БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Б. В. Фалейчик

МЕТОДЫ ВЫЧИСЛЕНИЙ

Рекомендовано
Учебно-методическим объединением
по естественнонаучному образованию
в качестве пособия для студентов
учреждения высшего образования,
обучающихся по специальности
1-31 03 07 «Прикладная информатика
(по направлениям)»

МИНСК
БГУ
2014

УДК 519.6 (075.8)
ББК 22.19 я73
Ф19

Рецензенты:
член-корреспондент НАН Беларуси, доктор физико-математических наук, профессор *Л. А. Янович*;
доктор физико-математических наук, профессор *Е. А. Ровба*

Фалейчик, Б. В.

Ф19 Методы вычислений : пособие / Б. В. Фалейчик. – Минск : БГУ, 2014. – 224 с.

ISBN 978-985-566-097-3.

Рассматриваются численные методы решения задач линейной алгебры, нелинейных уравнений и систем, методы аппроксимации функций, приближения интегралов и решения обыкновенных дифференциальных уравнений.

Предназначено для студентов учреждения высшего образования, обучающихся по специальности 1-31 03 07 «Прикладная информатика (по направлениям)».

УДК 519.6 (075.8)
ББК 22.19 я73

ISBN 978-985-566-097-3

ПРЕДИСЛОВИЕ

Методы вычислений (численные методы, вычислительные методы, вычислительная математика) – наука, находящаяся на стыке прикладной математики и информатики и занимающаяся построением и анализом методов и алгоритмов приближенного (численного) решения разнообразных задач математики. У ее истоков стояли такие великие ученые, как И. Ньютон, Ж. Лагранж, К. Гаусс, Л. Эйлер. Говоря о нашем времени, можно утверждать, что гигантский рывок в развитии вычислительной техники в последние несколько десятилетий был обусловлен потребностями вычислительной математики и ее приложений. Кроме традиционных задач моделирования физических, химических и других процессов, сегодня методы вычислительной математики применяются в совершенно неожиданных областях. Например, сеточные методы решения уравнения Пуассона используются в компьютерной графике, методы линейной алгебры – в интеллектуальном анализе данных.

Предлагаемое пособие основано на материалах учебного курса, который в течение нескольких лет читался автором по дисциплинам «Вычислительные методы алгебры», «Методы численного анализа» и «Методы вычислений» для студентов факультета прикладной математики и информатики Белорусского государственного университета. Его содержание наиболее близко к учебной программе курса «Методы вычислений» для специальности 1-31 03 07 «Прикладная информатика (по направлениям)».

Для успешного усвоения материала студентам необходимы базовые знания линейной алгебры и математического анализа, а также основ программирования. В данном пособии сделан акцент на теоретических аспектах построения и анализа вычислительных методов. Важно отметить, что формулы, которые определяют метод, являются лишь базой для построения *вычислительного алгоритма*. Именно вычислительный алгоритм представляет интерес для конечного пользователя. В тексте пособия простые алгоритмы даются в виде псевдокода, более сложные описаны словесно. При этом следует понимать, что наиболее эффективные вычислительные

алгоритмы, реализующие тот или иной численный метод, весьма сложны и не могут быть рассмотрены в рамках вводного курса.

В большинстве случаев каждый пункт пособия содержит материал одной лекции. По ходу изложения материала даны упражнения различной сложности, отмеченные значком $\triangleright_n$, где n — порядковый номер упражнения в текущем пункте. Выполнение упражнений имеет большое значение для усвоения материала, а также служит инструментом самостоятельного контроля знаний. Кроме этого, отдельные упражнения могут использоваться в качестве задач для практических занятий или контрольных работ в дополнение к базовым задачам, приведенным в конце книги.

Автор выражает признательность рецензентам профессору Л. А. Яновичу и профессору Е. А. Ровбе за ценные замечания и предложения по улучшению рукописи.

Условные обозначения

$\mathbb{N}$ — множество натуральных чисел

$\mathbb{Z}$ — множество целых чисел

$\mathbb{R}$ — множество вещественных чисел

$\mathbb{R}^n$ — n-мерное вещественное векторное пространство

$\mathbb{C}$ — множество комплексных чисел

$\mathbb{P}_n$ — множество алгебраических многочленов степени не выше n

$\underline{a}_i$ — i-я строка матрицы A

$f : X \to Y$ — функция f, отображающая элементы множества X в элементы множества Y

$f : x \mapsto y$ — функция f, отображающая x в y

$a \leftarrow b$ — переменной a присваивается значение переменной b

$\operatorname{span}\{v_1, v_2, \ldots, v_n\}$ — множество всех линейных комбинаций вида $\sum_{i=1}^{n} a_i v_i$, $a_i \in \mathbb{R}$, (линейная оболочка множества векторов $\{v_1, v_2, \ldots, v_n\}$)

$\operatorname{diag}\{a_1, a_2, \ldots, a_n\}$ — диагональная матрица размерности n с элементами $a_1, a_2, \ldots, a_n$ на диагонали

A^T — транспонированная к матрице A

A^* — сопряженная к матрице A

$[A]_k$ — квадратная матрица размерности k, составленная из элементов первых k строк и столбцов матрицы A

$\sigma(A)$ — множество всех собственных значений матрицы A (спектр матрицы A)

$\rho(A)$ — спектральный радиус матрицы A

$\overline{m, n}$ — диапазон целых чисел от m до n (включительно)

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

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

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.

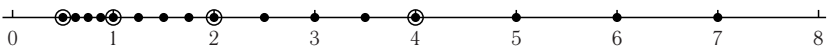
Определение 1.2. Число с плавающей точкой с ненулевым первым разрядом ($d_0 \neq 0$) называется *нормализованным*. Множество всех

нормализованных ЧПТ с основанием β, p-разрядной мантиссой и $e_{\min} \leqslant \leqslant e \leqslant e_{\max}$ 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 2–6, 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 условиям

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

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. Параллельная реализация метода Гаусса

2.2.1. Введение

Как известно, существует два основных класса параллельных вычислительных систем: системы с *общей* и системы с *распределенной* памятью. Наиболее распространенные представители первого класса — персональные компьютеры с многоядерными процессорами. Системы с распределенной памятью состоят из нескольких узлов — процессоров, каждый из которых имеет свою собственную память. Такие системы менее производительны, поскольку требуют выполнения операций

обмена данными между узлами. С другой стороны, системы с распределенной памятью позволяют привлекать гораздо большее количество процессоров, чем системы с общей памятью. Простейший пример системы с распределенной памятью — компьютеры, соединенные локальной сетью.

Мы рассмотрим лишь простейший вариант реализации метода Гаусса для параллельных систем с распределенной памятью. При этом описанный метод легко может быть адаптирован и к системе с общей памятью.

Итак, наша задача — решить СЛАУ $Ax = b$ размерности n методом Гаусса с выбором главного элемента по столбцу. Предположим, что параллельная вычислительная система имеет M узлов, причем для простоты положим $n = mM, m \in \mathbb{N}$. Пронумеруем все узлы от 0 до $M - 1$.

2.2.2. Распределение данных между узлами

Поскольку мы работаем с распределенной памятью, важно выбрать грамотный способ хранения данных, ведь полностью хранить матрицу A на каждом узле нерационально (в самом «страшном» случае матрица целиком просто может не поместиться). Поступим следующим образом: каждый узел будет хранить m *столбцов* матрицы A, но столбцы распределим не подряд, а циклически, т. е. узел номер 0 хранит столбцы с номерами $1, M + 1, 2M + 1, \ldots$, узел номер 1 — столбцы с номерами $2, M + 2, 2M + 2, \ldots$, и т. д. (рис. 2.1).

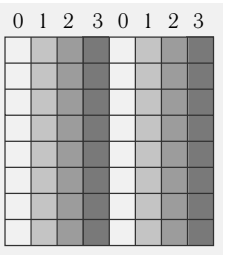


Рис. 2.1

Сделано это для того, чтобы по мере приведения матрицы к треугольному виду узлы не выключались из работы. Кроме самой матрицы системы,

передадим на каждый узел вектор правой части b. Это не обязательно, но затраты на его обработку минимальны, а алгоритм будет проще.

Таким образом, узел с номером p хранит матрицу A_p, состоящую из n строк и m столбцов, а также вектор b.

2.2.3. Прямой ход

Основная вычислительная нагрузка приходится именно на этот этап метода. Прямой ход состоит из $n-1$ шагов: на k-м шаге обнуляются элементы k-го столбца матрицы A, находящиеся ниже главной диагонали.

Предположим, что уже выполнено $k-1$ шагов. Опишем ход выполнения k-го шага.

1) Начинается шаг с того, что узел, хранящий k-й столбец матрицы A (обозначим этот столбец буквой c), рассылает его всем остальным узлам[1].

2) Теперь необходимо выбрать главный элемент в столбце c. Эта операция выполняется на каждом узле самостоятельно (каждый работает со своей локальной копией). Затем *параллельно* меняются местами соответствующие строки в матрицах A_p, $p = \overline{0, M-1}$, а также элементы локальных векторов b и c.

3) Наконец, осуществляются непосредственно элементарные операции со строками матриц A_p, $p = \overline{0, M-1}$: все узлы одновременно выполняют следующий алгоритм:

1: **for** $i = \overline{k+1, n}$ **do**
2: $\quad l \leftarrow c_i / c_k$
3: $\quad \underline{a}_i \leftarrow \underline{a}_i - l\,\underline{a}_k$
4: $\quad b_i \leftarrow b_i - l\,b_k$
5: **end for**

Здесь $\underline{a}_i$ обозначает i-ю строку матрицы A_p.

Осуществив таким образом $n-1$ шагов, приведем матрицу A к верхнетреугольному виду.

2.2.4. Обратный ход

Обратный ход занимает гораздо меньше ресурсов, чем прямой, поэтому его можно выполнить последовательно. Значит для завершения

[1] Можно рассылать не весь столбец, а лишь его элементы с индексами от k до n.

решения СЛАУ нужно просто собрать столбцы матрицы со всех узлов на один и выполнить стандартный обратный ход.

2.3. *LU*-разложение. Метод квадратного корня

2.3.1. Базовый алгоритм *LU*-разложения

Рассмотрим последовательность СЛАУ

$$Ax^{(i)} = b^{(i)}, \quad i = \overline{1, N}, \tag{2.12}$$

и предположим, что векторы $b^{(i)}$ неизвестны заранее и поступают *по одному* (например, $b^{(i)}$ зависит от $x^{(i-1)}$). В таком случае нельзя свести (2.12) к матричному уравнению. При решении каждой такой СЛАУ методом Гаусса будет тратиться $O(n^3)$ операций, причем к матрице A будут применяться *одни и те же* преобразования L_k.

Поэтому разумнее, однажды проделав прямой ход (или его аналог), построить LU-разложение, $A = LU$, и в дальнейшем вычислять x путем решения двух СЛАУ с треугольными матрицами:

$$LUx = b \quad \Leftrightarrow \quad \begin{cases} Ly = b, \\ Ux = y. \end{cases} \tag{2.13}$$

$\triangleright_1$ Запишите алгоритм вычисления x по формулам (2.13).

Рассмотрим алгоритм построения LU-разложения в предположении, что $|[A]_k| \neq 0 \; \forall k = \overline{1, n}$. По определению имеем

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \ell_{21} & 1 & 0 & \cdots & 0 \\ \ell_{31} & \ell_{32} & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \cdots & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} u_{11} & u_{12} & u_{13} & \ldots & u_{1n} \\ 0 & u_{22} & u_{23} & \ldots & u_{2n} \\ 0 & 0 & u_{33} & \ldots & u_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{bmatrix}}_{U} = \underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn} \end{bmatrix}}_{A}.$$

При машинной реализации алгоритма матрицы L и U будем хранить на месте матрицы A:

$$A \leftarrow \tilde{A} = \underbrace{\begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ \ell_{21} & u_{22} & u_{23} & \dots & u_{2n} \\ \ell_{31} & \ell_{32} & u_{33} & \dots & u_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \cdots & u_{nn} \end{bmatrix}}_{L-I+U}, \quad \text{т. е.} \quad \tilde{a}_{ij} = \begin{cases} u_{ij}, & i \leqslant j, \\ \ell_{ij}, & i > j. \end{cases}$$

Треугольная структура матриц L и U влечет следующее тождество (рис. 2.2):

$$a_{ij} = \sum_{k=1}^{n} \ell_{ik} u_{kj} = \sum_{k=1}^{\min(i,j)} \ell_{ik} u_{kj}. \tag{2.14}$$

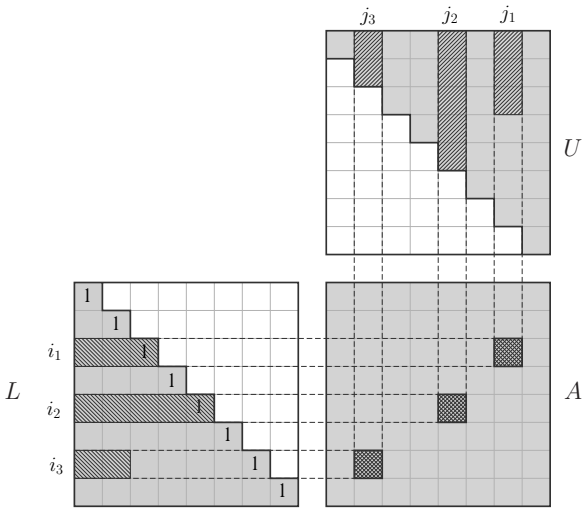


Рис. 2.2

Выделяя последние слагаемые в суммах (2.14), получим

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} \ell_{ik} u_{kj} \qquad \text{при } i \leqslant j; \tag{2.15}$$

$$\ell_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} u_{kj} \right) \qquad \text{при } i > j, \tag{2.16}$$

или

$$\tilde{a}_{ij} = \begin{cases} a_{ij} - \sum\limits_{k=1}^{i-1} \tilde{a}_{ik} \tilde{a}_{kj} & \text{при } i \leqslant j; \\ \dfrac{1}{\tilde{a}_{jj}} \left(a_{ij} - \sum\limits_{k=1}^{j-1} \tilde{a}_{ik} \tilde{a}_{kj} \right) & \text{при } i > j. \end{cases} \tag{2.17}$$

Таким образом, неизвестные элементы матриц L и U последовательно выражаются через a_{ij} и уже найденные ℓ_{ik} и u_{kj}.

Базовый алгоритм *LU*-разложения

1: **for** $j = \overline{1, n}$ **do**
2: **for** $i = \overline{1, n}$ **do**
3: $a_{ij} \leftarrow a_{ij} - \sum\limits_{k=1}^{\min(i,j)-1} a_{ik} a_{kj}$
4: **if** $i > j$ **then**
5: $a_{ij} \leftarrow a_{ij} / a_{jj}$
6: **end if**
7: **end for**
8: **end for**

Замечание 2.2. Данный алгоритм, иногда называемый алгоритмом Краута (Crout), не является единственно возможным. Альтернативный способ построения LU-разложения можно найти, например, в [6, 21].

2.3.2. Выбор главного элемента

По аналогии с методом Гаусса, этап алгоритма LU-разложения, определяемый циклом в строках 2–7, будем называть j-м шагом LU-разложения. Для того чтобы алгоритм был универсальным, необходимо ре-

ализовать выбор главного элемента $\tilde{a}_{jj} = u_{jj}$, на который происходит деление в строке 5.

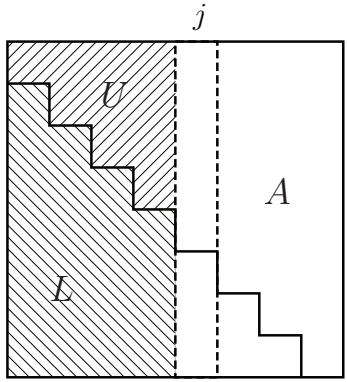


Рис. 2.3. Вид матрицы системы перед j-м шагом алгоритма LU-разложения.

Рассмотрим матрицу $A^{(j)}$, которая получается из A после $(j-1)$-го шага разложения (рис. 2.3). К этому моменту столбцы с 1-го по $(j-1)$-й уже содержат часть матриц L и U, а оставшиеся столбцы являются столбцами исходной матрицы A. Имеем ли мы право переставлять в этой «составной» матрице строки и если да, то какие? Строки с 1 по $(j-1)$-ю переставлять нельзя, иначе нарушится структура матрицы $\tilde{A}$. Перестановка же строк с j по n эквивалентна преобразованию

$$LU = A \quad \mapsto \quad PLU = PA,$$

где P — матрица перестановки.

Итак, на j-м шаге алгоритма мы имеем право переставлять строки с номерами от j до n. Поэтому элементы u_{ij} для i от 1 до $j-1$, вычисляемые по формуле (2.15), можно найти сразу. После этого нужно осуществить перестановку строк, но проблема в том, что ведущий элемент u_{jj} *еще неизвестен*, как неизвестны и возможные кандидаты на его место.

Поэтому перестановка должна быть выполнена таким образом, *чтобы элемент* $a_{jj}^{(j)} = u_{jj}$, вычисляемый по формуле

$$u_{jj} = a_{jj} - \sum_{k=1}^{j-1} \ell_{jk} u_{kj}, \tag{2.18}$$

был максимальным по модулю [18]. Заметим, что элементы ℓ_{ij} вычисляются по формуле (2.16), которая при $i = j$ отличается от (2.18) только множителем $1/u_{jj}$. Поэтому выбор главного элемента на j-м шаге LU-разложения осуществляется следующим образом:

1) вычисляются кандидаты на роль ведущего элемента: для всех i от j до n находим $a_{ij}^{(j)} = \tilde{a}_{ij}$ по второй формуле из (2.17), только *без деления на* $\tilde{a}_{jj}$;

2) среди полученных значений $a_{ij}^{(j)}$ для $i \geqslant j$ выбирается максимальный по модулю $a_{i^*j}^{(j)}$;

3) меняются местами j-я и i^*-я строки матрицы $A^{(j)}$;

4) для всех i от $j+1$ до n делим $a_{ij}^{(j)}$ на $a_{jj}^{(j)}$.

Для того чтобы после получения LU-разложения корректно решить СЛАУ (2.13), необходимо предварительно переставить элементы вектора b в соответствии с перестановками, которые происходили в ходе разложения. Поэтому стандартная процедура должна возвращать не только матрицу $\tilde{A}$, но и вектор перестановок p. Этот вектор получается применением перестановок, осуществляемых в ходе алгоритма, к вектору $(1, 2, \ldots, n)^T$. После этого для корректного решения СЛАУ $Ax = b$ нужно будет по схеме (2.13) решить систему

$$LUx = b',$$

где вектор b' получается путем применения к b перестановок, «сохраненных» в векторе p:

$$b'_i = b_{p_i}.$$

Кроме этого, для корректного вычисления определителя необходимо возвращать $s = \pm 1$ — значение четности числа перестановок.

Таким образом, метод LU-разложения фактически представляет собой «законсервированный» метод Гаусса.

2.4. Метод квадратного корня

2.4.1. Разложение Холецкого

Теорема 2.4 (разложение Холецкого)**.** *Пусть A — самосопряженная матрица над полем $\mathbb{C}$: $A = A^*$. Если все главные миноры $|[A_k]|$ отличны от нуля, то существует разложение*

$$A = R^* D R, \tag{2.19}$$

где R — верхнетреугольная матрица, $D = \operatorname{diag}(d_1, d_2, \ldots, d_n)$, $|d_k| = 1\ \forall k = \overline{1,n}$. Формула (2.19) *называется* разложением Холецкого (Cholesky decomposition).

Доказательство. Поскольку $|[A]_k| \neq 0$, по теореме 2.2 существует LU-разложение

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

откуда $L = U^*(L^*U^{-1}) = U^*H$, и

$$A = LU = U^*HU. \tag{2.20}$$

Рассмотрим матрицу $H = L^*U^{-1}$. С одной стороны, H — верхнетреугольная, так как является произведением верхнетреугольных матриц L^* и U^{-1}. С другой стороны, $H = (U^*)^{-1}L$, т. е. H является еще и нижнетреугольной. Следовательно,

$$H = \operatorname{diag}(h_1, h_2, \ldots, h_n).$$

Положим $d_k = h_k/|h_k|$, $D = \operatorname{diag}(d_1, \ldots, d_n)$, $\tilde{H} = \operatorname{diag}(\sqrt{|h_1|}, \sqrt{|h_2|}, \ldots, \sqrt{|h_n|})$. Тогда

$$H = \tilde{H}D\tilde{H},$$

и тождество (2.20) дает

$$A = U^*HU = U^*(\tilde{H}^*D\tilde{H})U = (\tilde{H}U)^*D(\tilde{H}U) = R^*DR,$$

что и требовалось доказать. □

Определение 2.3. Квадратная матрица A над полем $\mathbb{R}(\mathbb{C})$ называется *положительно определенной* ($A > 0$), если

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

В комплексном случае мы подразумеваем, что все (Ax, x) вещественны.

В дальнейшем нам понадобятся следующие свойства положительно определенных матриц:

1) если $A > 0$, то $|[A]_k| \neq 0\ \forall k = \overline{1,n}$;

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

Теорема 2.5. *Если $A = A^*$ и $A > 0$, то существует разложение*

$$A = R^* R,$$

где R — верхнетреугольная матрица.

Доказательство. Согласно свойству 1, для матрицы A существует разложение (2.19). Значит, нам достаточно показать, что матрица D — единичная. По условию имеем

$$(Ax, x) = (R^* DRx, x) = (DRx, Rx) > 0 \quad \forall x \neq 0.$$

Поскольку R не вырождена, $\forall y \in \mathbb{C}^n \quad \exists\, x :\ y = Rx$, т. е.

$$(Dy, y) = \sum_{i=1}^{n} d_i y_i \bar{y}_i > 0 \quad \forall y \neq 0. \tag{2.21}$$

Возьмем в качестве y k-й единичный орт: $y_i = \delta_{ik}$. Тогда с учетом того, что $|d_k| = 1$, из (2.21) получим $d_k = 1\ \forall k = \overline{1, n}$. □

Таким образом, в случае вещественной матрицы A из теорем 2.4 и 2.5 вытекают следующие утверждения: если A симметричная ($A = A^T$) и все $|[A]_k| \neq 0$, то существует разложение вида

$$A = R^T DR, \tag{2.22}$$

где R — вещественная верхнетреугольная; D — диагональная матрица с элементами ± 1 на диагонали. Если к тому же $A > 0$, то $D = I$.

2.4.2. Алгоритм метода

Методом квадратного корня называется метод решения вещественной СЛАУ $Ax = b$ с симметричной матрицей A путем построения разложения Холецкого

$$A = R^T DR.$$

Обозначим $L = R^T, U = DR$. Тогда аналогично методу LU-разложения имеем

$$a_{ij} = \sum_{k=1}^{\min(i,j)} \ell_{ik} u_{kj} = \left[\begin{array}{c} \ell_{ik} = r_{ki}, \\ u_{kj} = d_k r_{kj} \end{array} \right] = \sum_{k=1}^{\min(i,j)} d_k r_{ki} r_{kj}. \tag{2.23}$$

В силу симметрии достаточно рассмотреть (2.23) только для верхнего треугольника матрицы A ($i \leqslant j$):

$$i = j: \quad d_i r_{ii}^2 = a_{ii} - \sum_{k=1}^{i-1} d_k r_{ki}^2 = \omega_i \;\Rightarrow\; \begin{cases} d_i = \operatorname{sign} \omega_i, \\ r_{ii} = \sqrt{|\omega_i|}; \end{cases} \tag{2.24а}$$

$$i < j: \quad r_{ij} = \frac{1}{d_i r_{ii}} \left(a_{ij} - \sum_{k=1}^{i-1} d_k r_{ki} r_{kj} \right). \tag{2.24б}$$

Вычисления организуются следующим образом: для $i = \overline{1,n}$ по формулам (2.24) поочередно находятся r_{ij} для $j = \overline{i,n}$.

$\triangleright_1$ Запишите алгоритм прямого и обратного хода метода квадратного корня.

$\triangleright_2$ Оцените число мультипликативных операций в прямом ходе метода и сравните с методом Гаусса.

Детали программной реализации

1) Так как матрица A симметрична, достаточно хранить в памяти только ее верхний треугольник.
2) Аналогично случаю LU-разложения расчетные формулы (2.24) позволяют последовательно (построчно) находить элементы матрицы R и хранить их на месте исходной матрицы: $A \leftarrow R$.
3) Решение получаемой в итоге СЛАУ $R^T D R x = b$ осуществляется путем применения двух обратных подстановок: $R^T y = b$, затем $Rx = Dy$ (так как $D = D^{-1}$).

Замечание 2.3. Альтернативный алгоритм построения разложения Холецкого можно найти в [21].

2.5. Итерационные методы решения СЛАУ

2.5.1. 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[2]. 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. При этом ключевым требованием является то, что все приближения строятся по единому правилу:

$$x^{k+1} = \varphi(x^k), \quad k = 0, 1, 2, \ldots.$$

В общем случае x^{k+1} может зависеть от нескольких предыдущих приближений:

$$x^{k+1} = \varphi(x^k, x^{k-1}, \ldots, x^{k-m}), \quad k = m, m+1, m+2, \ldots.$$

2.5.2. Итерационные методы общего вида

Принцип построения

Рассмотрим СЛАУ

$$Ax = b$$

с невырожденной матрицей A. Приведем данную систему к виду

$$x = Bx + g \quad \Leftrightarrow \quad Ax = b, \tag{2.25}$$

где B — матрица; g — вектор соответствующей размерности. Отсюда автоматически получается итерационный процесс

$$x^{k+1} = Bx^k + g, \quad k = 0, 1, 2, \ldots. \tag{2.26}$$

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

Нетрудно заметить, что если такой процесс сходится, т. е. сходится последовательность (x^k), то предел этой последовательности x^* удовлетворяет условию $x^* = Bx^* + g$, а в силу (2.25) это означает, что $x^* = A^{-1}b$ — искомое решение. Процессы типа (2.26) будем называть *итерационными процессами общего вида*. Рассмотрим наиболее известные примеры таких процессов.

2.5.3. Классические итерационные методы

Метод простой итерации. Самый простой способ приведения СЛАУ $Ax = b$ к виду $x = Bx + g$ состоит в добавлении к обеим частям вектора x:

$$x = (I - A)x + b.$$

Соответствующий итерационный метод

$$x^{k+1} = (I - A)x^k + b \tag{2.27}$$

будем называть *методом простой итерации* (МПИ).

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

$$x_i = \frac{1}{a_{ii}}\Big(b_i - \sum_{j \neq i} a_{ij}x_j\Big), \quad i = \overline{1,n}. \tag{2.28}$$

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

$$A = L + D + R. \tag{2.29}$$

Тогда (2.28) примет вид

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

где

$$B_J = -D^{-1}(L + R), \quad g_J = D^{-1}b. \tag{2.30}$$

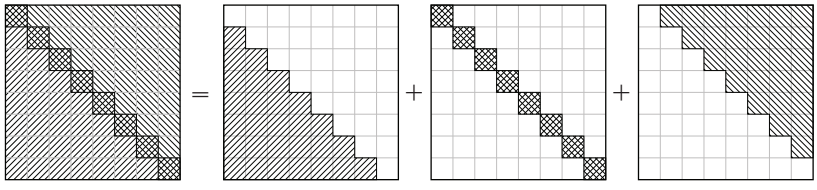


Рис. 2.4

Соответствующий системе (2.28) итерационный метод

$$x_i^{k+1} = \frac{1}{a_{ii}}\Big(b_i - \sum_{j\neq i} a_{ij}x_j^k\Big), \quad i = \overline{1,n}, \quad k = 0, 1, 2, \ldots \tag{2.31а}$$

называется *методом Якоби*. Его векторная форма имеет вид

$$x^{k+1} = B_J x^k + g_J, \tag{2.31б}$$

где B_J и g_J определяются по формуле (2.30).

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 (2.31а):

$$x_i^{k+1} = \frac{1}{a_{ii}}\Big(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k\Big). \tag{2.32а}$$

Векторный вариант (2.32а) имеет вид

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

откуда

$$x^{k+1} = B_S x^k + g_S,$$
$$B_S = -(D+L)^{-1}R, \quad g_S = (D+L)^{-1}b. \tag{2.32б}$$

Формулы (2.32а), (2.32б) определяют *метод Гаусса – Зейделя*.

$\triangleright_1$ Примените идею построения метода Гаусса – Зейделя к методу простой итерации (2.27). Запишите векторную и скалярную формы метода.

$\triangleright_2$ 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:

$$x_i^{k+1} = (1-\omega)x_i^k + \frac{\omega}{a_{ii}}\Big(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k\Big), \tag{2.33а}$$

где ω — весовой коэффициент, обычно $\omega \in (0,2)$. Формула (2.33а) 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

$$x^{k+1} = B_\omega x^k + g_\omega,$$
$$B_\omega = (D+\omega L)^{-1}((1-\omega)D - \omega R), \quad g_\omega = (D+\omega L)^{-1}b. \tag{2.33б}$$

Замечание 2.4. При $\omega = 1$ метод релаксации очевидно превращается в метод Гаусса – Зейделя.

Внимание! *Для программной реализации классических итерационных методов используются исключительно их скалярные формы (2.31а), (2.32а), (2.33а). Соответствующие векторные формы записи (2.31б), (2.32б), (2.33б) используются для анализа сходимости методов (см. далее).*

2.5.4. Сходимость итерационных процессов общего вида

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

сходимость итерационных процессов общего вида

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

Лемма 2.1. *Итерационный процесс общего вида сходится тогда и только тогда, когда последовательность матриц*

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

сходится к нулевой матрице.

Доказательство. Пусть итерационный процесс $x^{k+1} = Bx^k + g$ сходится к x^*. Рассмотрим

$$x^{k+1} - x^k = B(x^k - x^{k-1}) = \ldots = B^k(x^1 - x^0) = B^k(\underbrace{(B - I)x^0 + g}_{v}).$$

Поскольку последовательность (x^k) сходится, имеем $\|x^{k+1} - x^k\| \to 0$ $\forall x^0 \in \mathbb{R}^n$, откуда $\|B^k v\| \to 0\ \forall v \in \mathbb{R}^n$, т. е. $B^k \to 0$.

Пусть теперь $B^k \to 0$. Имеем

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

откуда сразу следует, что $\|x^k - x^*\| \to 0$ при $k \to \infty$. □

Определение 2.4. 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 2.5. *Спектральным радиусом* $\rho(A)$ называется величина наибольшего по модулю собственного значения матрицы A:

$$\rho(A) = \max_i |\lambda_i|.$$

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*, т. е. представима в виде

$$B = X^{-1}DX,$$

где матрица X невырождена, а

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

λ_i — собственные значения B.

В этом случае имеем

$$B^2 = X^{-1}DXX^{-1}DX = X^{-1}D^2X,$$

или

$$B^k = X^{-1}D^kX,$$

откуда

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

Следствие 2.2 (критерий сходимости итерационных процессов общего вида)**.** *Итерационный процесс $x^{k+1} = Bx^k + g$ сходится тогда и только тогда, когда $\rho(B) < 1$.*

Следствие 2.3 (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, в силу следствия 2.2 итерационный процесс $x^{k+1} = Bx^k + g$ сходится. $\square$

Замечание 2.5. Из доказательства леммы 2.2 можно увидеть, что скорость сходимости итерационного процесса общего вида зависит от величины спектрального радиуса матрицы B: чем он меньше, тем быстрее (в асимптотике) сходится процесс.

Замечание 2.6. Как видим, сходимость итерационного метода не зависит от начального приближения x_0 и полностью определяется матрицей B.

Сходимость классических итерационных методов

Применим полученные выше общие результаты о сходимости к методу Якоби (2.31).

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 2.3. □

Замечание 2.7. Аналогичное утверждение можно доказать и для метода Гаусса – Зейделя.

Следующий важный результат мы приведем без доказательства.

Теорема 2.6. *Если матрица A является симметричной и положительно определенной, то метод релаксации* (2.33) *сходится при любом* $\omega \in (0, 2)$.

Следствие 2.4. *Если матрица A является симметричной и положительно определенной, то метод Гаусса – Зейделя* (2.32) *сходится.*

2.6. Форматы хранения разреженных матриц

Определение 2.6. *Разреженными* называют матрицы, содержащие большой процент нулевых элементов.

Разреженные матрицы очень часто возникают в приложениях, в частности, при численном моделировании различных физических процессов. Количество ненулевых элементов матрицы в дальнейшем будем обозначать n_z.

Хранить в памяти ЭВМ все нулевые элементы разреженных матриц нерационально. Они, во-первых, зря занимают память, и, во-вторых, замедляют операции с матрицами. Поэтому существует ряд общепринятых способов хранения разреженных матриц.

2.6.1. Координатный формат

Самый простой способ — так называемый *координатный формат*. В этом формате для хранения вещественной матрицы A используется три массива:

- `AA` — массив вещественных чисел для хранения ненулевых элементов матрицы A;
- `IA` — массив целых чисел для хранения номеров строк соответствующих элементов массива `AA`;
- `JA` — аналогичный массив для номеров столбцов.

Длина всех трех массивов равна n_z.

Пример 2.1. Записать в координатном формате матрицу

2.				
		5.	7.	
		9.		
1.	4.		3.	
	8.			6.

Решение. Представление матрицы в координатном формате задается с точностью до перестановки элементов:

AA	1.	2.	3.	4.	5.	6.	7.	8.	9.
IA	4	1	4	4	2	5	2	5	3
JA	1	1	4	2	3	5	4	2	3

□

2.6.2. Форматы CSR и CSC

Если в примере 2.1 упорядочить элементы матрицы построчно, то информация в массиве IA окажется избыточной:

AA	2.	5.	7.	9.	1.	4.	3.	8.	6.
IA	*1*	*2*	*2*	*3*	*4*	*4*	*4*	*5*	*5*
JA	1	3	4	3	1	2	4	2	5

В этом случае достаточно хранить лишь *указатель* на элемент массива AA, с которого начинается хранение i-й строки:

IA	1	2	4	5	8	10

Здесь последний элемент вектора IA служит для того, чтобы можно было определить, где заканчивается n-я строка. Описанный способ хранения матриц называется *форматом CSR* (Compressed Sparse Row).

Если ненулевые элементы в массиве AA упорядочить не по строкам, а по столбцам, по аналогии получится *формат CSC* (Compressed Sparse Column).

Описание формата CSR (CSC)

- `AA` — массив вещественных чисел длины n_z, в котором хранятся упорядоченные по строкам (по столбцам) ненулевые элементы A.
- `JA` — массив целых чисел длины n_z для хранения номеров столбцов (строк) соответствующих элементов массива `AA`.
- `IA` — массив целых чисел длины $n+1$. На i-й позиции массива хранится *номер позиции* в массиве `AA`, с которой начинается хранение элементов i-й строки (столбца). Более конкретно: `IA[1]=1`, `IA[i+1]=IA[i]+` $+n_i$, где n_i — число ненулевых элементов в i-й строке (столбце).

Формат CSR является одним из наиболее популярных.

$\triangleright_1$ Укажите достоинства и недостатки этого формата по сравнению с координатным форматом.

Алгоритм умножения матрицы в формате CSR на вектор

Вход: `n, AA, JA, IA, x`.
Выход: `y`.

```
for i=1 to n do
    y[i]=0
    for j=IA[i] to IA[i+1]-1 do
        y[i]=y[i]+AA[j]*x[JA[j]]
    end for
end for
```

Алгоритм умножения матрицы в формате CSC на вектор

```
y=0
for j=1 to n do
    for i=IA[j] to IA[j+1]-1 do
        y[?]=y[?]+AA[?]*x[?]
    end for
end for
```

$\triangleright_2$ Что должно стоять вместо знаков вопроса?

2.6.3. Формат MSR

Во многих итерационных методах решения СЛАУ диагональные элементы матрицы A играют особую роль: доступ к ним нужно осуществлять

чаще, чем к другим элементам матрицы. Рассмотренные ранее форматы не позволяют быстро найти диагональные элементы. Решить эту проблему помогает модификация формата CSR — *формат MSR* (Modified Sparse Row).

Основные отличия этого формата от формата CSR состоят в следующем:

1) изменен формат массива `AA`: сначала в него *полностью* записывается главная диагональ матрицы A, а затем — недиагональные ненулевые элементы;

2) массивы `IA` и `JA` объединены в один (назовем его `IJ`).

Рассмотрим матрицу из примера 2.1:

2.				
		5.	7.	
		9.		
1.	4.		3.	
	8.			6.

Запишем ее представление в формате MSR:

	1	2	3	4	5	6	7	8	9	10	11
`AA`	2.	0.	9.	3.	6.	$\times$	5.	7.	1.	4.	8.
`IJ`	7	7	9	9	11	12	3	4	1	2	2

Массив `AA`: первые n элементов занимает главная диагональ A. Элемент на позиции $n+1$ *не используется*. Начиная с $(n+2)$-го элемента построчно хранятся недиагональные ненулевые элементы A.

Массив `IJ`: `IJ[1]=n+2`, `IJ[i+1]=IJ[i]+`n_i для $i = \overline{1,n}$, где n_i — количество недиагональных ненулевых элементов в i-й строке. Далее — номера столбцов для соответствующих элементов массива `AA`.

Замечание 2.8. Массивы `AA` и `IJ` полностью описывают матрицу A. Размерность матрицы легко найти по значению `IJ[1]`:

`n=IJ[1]-2`.

Кроме этого, длина обоих массивов равна `IJ[n+1]-1`.

Алгоритм умножения матрицы в формате MSR на вектор

Вход: `AA, IJ, x`.
Выход: `y`.

```
n=IJ[1]-2
for i=1 to n do
    y[i]=AA[i]*x[i]
    for j=IJ[i] to IJ[i+1]-1 do
        y[i]=y[i]+AA[j]*x[IJ[j]]
    end for
end for
```

$\triangleright_3$ Почему в массиве `AA` пустует элемент номер $n+1$?

Глава 3

МЕТОДЫ РЕШЕНИЯ ПРОБЛЕМЫ СОБСТВЕННЫХ ЗНАЧЕНИЙ

3.1. Проблема собственных значений: общая характеристика

3.1.1. Сведения из линейной алгебры

Пусть A — квадратная матрица над полем $\mathbb{C}$. Вектор $x \neq 0 \in \mathbb{C}^n$ 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 $\sigma(A)$.

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

$$Ax = \lambda x \quad \Rightarrow \quad A(\alpha x) = \lambda(\alpha x).$$

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.

Пример 3.1. Рассмотрим матрицу $A = I$. Она очевидно имеет одно собственное значение, равное 1, а собственным вектором является любой вектор $x \in \mathbb{C}^n$. Следовательно, геометрическая кратность собственного значения равна n. Характеристическое уравнение имеет вид $(1 - \lambda)^n = 0$, т. е. геометрическая кратность равна алгебраической.

Пример 3.2. Рассмотрим матрицу

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

Алгебраическая кратность собственного значения 1 равна 2. Собственным вектором является любой вектор вида $(\xi, 0)^T$, следовательно, геометрическая кратность равна 1.

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*.

Свойства диагонализируемых матриц

1) Диагонализируемая матрица может быть приведена к диагональному виду преобразованием подобия:

$$A = S^{-1}DS, \qquad \det S \neq 0, \quad D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

2) Матрица диагонализируема тогда и только тогда, когда алгебраическая и геометрическая кратности каждого собственного значения совпадают.

3.1.2. Общая характеристика проблемы собственных значений

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, $x^1, \ldots, x^n$ — соответствующий базис из собственных векторов. 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 $\{x^i\}$ произвольный вектор $y^0 \in \mathbb{C}^n$: $y^0 = \sum_1^n \alpha_i x^i$. Предположим, что $\alpha_1 \neq 0$ и рассмотрим последовательность (y^k) :

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

Тогда

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

Значит, при $k \to \infty$ имеем

$$y^k \sim \lambda_1^k \alpha_1 x^1,$$

т. е. вектор y^k все сильнее приближается к собственному подпространству, соответствующему x^1. Для практического применения необходимо на каждом шаге нормировать y^k. Существует несколько вариантов нормировки, мы рассмотрим один из них, самый простой и достаточно эффективный.

Обозначим $\max(x)$ максимальную по модулю компоненту вектора x. Тогда процесс степенного метода примет вид

$$v^{k+1} = Au^k, \quad u^{k+1} = v^{k+1} / \max(v^{k+1}), \quad u^0 = y^0.$$

При таком способе нормировки получаем $\max(u^k) = 1$ при всех $k \geqslant 1$, а также

$$u^k = y^k / \max(y^k) = A^k y^0 / \max(A^k y^0).$$

$\triangleright_1$ Докажите это.

Обозначим

$$\xi^1 = x^1 / \max(x^1).$$

Тогда в силу (3.1) по построению при $k \to \infty$ будем иметь

$$u^k \to \xi^1.$$

Кроме этого, так как максимальная по модулю компонента вектора ξ^1 равна 1, из тождества

$$A\xi^1 = \lambda_1 \xi^1$$

имеем

$$\max(v^{k+1}) = \max(Au^k) \to \max(A\xi^1) = \max(\lambda_1 \xi^1) = \lambda_1,$$

или просто

$$\max(v^k) \to \lambda_1.$$

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

$u = y^0$
$\lambda = 0$
while $\|Au - \lambda u\| > \varepsilon$ **do**
    $v \leftarrow Au$
    $\lambda \leftarrow \max(v)$
    $u \leftarrow v/\lambda$
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 $x^1, \ldots, x^m$ линейно

независимы и образуют собственное подпространство X_m размерности m. Тогда формула (3.1) примет вид

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

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 $x^1, \ldots, x^m$.

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

$$y^k = \sum_{i=1}^{n} \alpha_i \lambda_i^k x^i = \alpha_1 \lambda_1^k x^1 + \alpha_2 (-\lambda_1)^k x^2 + \sum_{i=3}^{n} \alpha_i \lambda_i^k x^i = \\ = \lambda_1^k \left(\alpha_1 x^1 + (-1)^k \alpha_2 x^2 + \sum_{i=3}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^k x^i \right),$$

откуда получим

$$\begin{aligned} y^{2k} &\sim \lambda_1^{2k} (\alpha_1 x^1 + \alpha_2 x^2), \\ y^{2k+1} &\sim \lambda_1^{2k+1} (\alpha_1 x^1 - \alpha_2 x^2), \end{aligned} \tag{3.2}$$

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

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

$$u^{2k} = y^{2k} / \max(y^{2k}), \quad k = 1, 2, \ldots.$$

Согласно (3.2), имеем

$$u^{2k} \to \eta = (\alpha_1 x^1 + \alpha_2 x^2)/M, \quad M = \max(\alpha_1 x^1 + \alpha_2 x^2), \tag{3.3}$$

при этом

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

Таким образом можно найти λ_1.

Теперь рассмотрим, как можно найти соответствующий собственный вектор. Для этого заметим, что

$$Au^{2k} \to A\eta = A(\alpha_1 x^1 + \alpha_2 x^2)/M = \lambda_1(\alpha_1 x^1 - \alpha_2 x^2)/M. \tag{3.5}$$

Тогда из (3.3)–(3.5) имеем следующую формулу для приближенного вычисления собственного вектора x^1:

$$\sqrt{\max(A^2u^{2k})}Au^{2k} + A^2u^{2k} \to 2\alpha_1\lambda_1^2x^1 = \tilde{v}.$$

Таким образом, $\tilde{v} \in \operatorname{span}(x^1)$. По желанию его можно пронормировать.

Согласно вышесказанному, можно построить следующую модификацию степенного алгоритма:

$u = y^0$
$\lambda = 0$
while $\|Au - \lambda u\| > \varepsilon$ **do**
    $v \leftarrow Au; \quad u \leftarrow Av$
    $\lambda \leftarrow \sqrt{\max(u)}$
    $v \leftarrow \lambda v + u$
    $u \leftarrow u/\max(u); \quad v \leftarrow v/\max(v)$
end while

В отличие от базового варианта, на выходе этого алгоритма будем иметь $v \approx \xi^1 = x^1/\max(x^1)$, а также по-прежнему $\lambda \approx \lambda_1$. Аналогично можно найти x^2.

3.2.4. Общий случай

Степенной метод можно применять не только для диагонализируемых матриц. Если геометрическая и алгебраическая кратности λ_1 совпадают, этот метод будет работать. В противном случае метод тоже может сходиться, но очень медленно.

3.2.5. Степенной метод со сдвигом

Степенной метод теоретически можно применять для отыскания произвольного собственного значения λ_j и x^j, пользуясь соотношением

$$\sigma(\alpha A + \beta I) = \alpha\sigma(A) + \beta.$$

За счет выбора α и β можно сместить спектр таким образом, что $\mu_j = \alpha\lambda_j + \beta$ станет максимальным собственным значением матрицы $\alpha A + \beta I$. Вычислив μ_j степенным методом, найдем соответственно и λ_j.

Опишем схему нахождения минимального собственного значения в случае вещественного спектра.

1) Пусть $0 \leqslant \lambda_i < m$ (когда m неизвестно, можно взять $m = \|A\|$). Заменим A на $B = mI - A$. Максимальное собственное значение B равно $m - \lambda_n$. Аналогично поступаем, когда все λ_i отрицательны.

2) Если λ_i имеют различные знаки, рассмотрим матрицу A^2. У нее те же собственные векторы, а собственные значения равны λ_i^2. Таким образом задача свелась к первому случаю.

Замечание 3.2. В общем случае минимальное по модулю собственное значение может быть найдено с помощью так называемого *метода обратной итерации*, который представляет собой степенной метод, примененный к матрице A^{-1}. При этом обращать матрицу нет необходимости, нужно лишь на каждой итерации дополнительно решать СЛАУ [5, с. 97].

3.3. Метод Данилевского

3.3.1. Преобразования подобия

Многие методы решения проблемы собственных значений используют принцип, аналогичный прямым методам решения СЛАУ: исходная матрица A путем некоторых преобразований приводится к эквивалентной матрице A', для которой проблема собственных значений решается просто. Естественно, решения задачи для матриц A и A' должны совпадать, т. е. преобразование $A \mapsto A'$ должно сохранять спектр. Другими словами, это преобразование должно быть *преобразованием подобия*.

Определение 3.2. Пусть S — невырожденная матрица. Преобразованием подобия квадратной матрицы A называется преобразование

$$A \mapsto S^{-1}AS.$$

Из курса линейной алгебры известно, что преобразование подобия сохраняет характеристический многочлен, а следовательно, и спектр матрицы. Следует понимать, однако, что такое преобразование изменяет собственные векторы:

$$A'x' = \lambda x' \quad \Leftrightarrow \quad S^{-1}ASx' = \lambda x' \quad \Leftrightarrow \quad ASx' = \lambda Sx',$$

т. е. если x' — собственный вектор A', то ему соответствует собственный вектор $x = Sx'$ матрицы A.

Таким образом, можно сформулировать следующее **общее правило**: если к матрице A применяется преобразование вида

$$A \mapsto TA = A',$$

то для сохранения спектра необходимо после этого выполнить преобразование

$$A' \mapsto A'T^{-1},$$

и наоборот.

В качестве преобразования T, как правило, используются те же преобразования, что и для СЛАУ, т. е., в частности, элементарные преобразования. Напомним, что эти преобразования бывают двух типов. Элементарное преобразование *первого рода* — это умножение i-й строки матрицы A на произвольный скаляр $\alpha \neq 0$. Данное преобразование эквивалентно умножению слева на *элементарную матрицу первого типа*

$$T_i(\alpha) = \operatorname{diag}(1, \ldots, 1, \alpha, 1, \ldots, 1),$$

где элемент α находится на i-й позиции.

Элементарное преобразование *второго рода* — это добавление к i-й строке матрицы A ее j-й строки, умноженной на α. Такое преобразование задается матрицей $S_{ij}(\alpha)$, которая представляет собой матрицу с единицами на главной диагонали и элементом α на позиции (i, j). Эту матрицу будем называть *элементарной матрицей второго типа*.

Чтобы в дальнейшем можно было использовать эти преобразования для решения проблемы собственных значений, нам необходимо (в соответствии с приведенным выше общим правилом) разобраться с тем, как действуют соответствующие операции «дополнения до подобия», т. е. чему эквивалентны операции

$$A \mapsto A(T_i(\alpha))^{-1} \quad \text{и} \quad A \mapsto A(S_{ij}(\alpha))^{-1}.$$

Элементарное преобразование подобия первого типа. Для начала вычислим обратную к матрице $T(\alpha_i)$. Очевидно, что

$$(T_i(\alpha))^{-1} = T_i(\alpha^{-1}).$$

Также нетрудно убедиться, что преобразование $A \mapsto AT_i(\alpha^{-1})$ представляет собой умножение i-го столбца матрицы A на α^{-1}. Таким образом, *если i-я строка (столбец) матрицы A умножается на скаляр α, то для сохранения спектра необходимо после этого разделить i-й столбец (строку) полученной матрицы на α.* Описанное преобразование будем называть *элементарным преобразованием подобия первого типа*.

Элементарное преобразование подобия второго типа. Обратная матрица к $S_{ij}(\alpha)$ вычисляется легко:

$$(S_{ij}(\alpha))^{-1} = S_{ij}(-\alpha).$$

Выяснить, что происходит с матрицей A при умножении справа на $S_{ij}(-\alpha)$, нам поможет известное тождество

$$(AB)^T = B^T A^T.$$

Имеем

$$AS_{ij}(-\alpha) = \left(S_{ij}(-\alpha)^T A^T\right)^T = \left(S_{ji}(-\alpha)A^T\right)^T,$$

т. е. *если к i-й строке (столбцу) матрицы A прибавлена j-я ее строка (столбец), умноженная на α, то для сохранения спектра необходимо после этого вычесть из j-го столбца (строки) полученной матрицы i-й столбец (строку), умноженный на α.*

Преобразование подобия с матрицей перестановки. Помимо элементарных преобразований мы также будем использовать преобразование перестановки местами строк i и j, которое задается матрицей P_{ij}. Эта матрица представляет собой единичную матрицу с переставленными i-й и j-й строками. Легко видеть, что

$$P_{ij}^{-1} = P_{ij}.$$

Кроме этого,

$$AP_{ij} = \left(P_{ij}A^T\right)^T,$$

поэтому третье правило преобразований подобия звучит так: *если в матрице переставлены строки (столбцы) с номерами i и j, то*

для сохранения спектра необходимо после этого у полученной матрицы переставить i-й и j-й столбцы (строки).

Теперь мы готовы приступить к изучению метода Данилевского.

3.3.2. Метод Данилевского

Метод Данилевского позволяет вычислить характеристический многочлен

$$P(\lambda) = |A - \lambda I|$$

для произвольной квадратной матрицы A.

Рассмотрим матрицу вида

$$\begin{bmatrix} p_1 & p_2 & \dots & p_{n-1} & p_n \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}. \tag{3.6}$$

Такой вид матрицы называется *формой Фробениуса*. Ее характеристический многочлен легко вычисляется путем рекурсивного разложения определителя по столбцу:

$$\begin{vmatrix} p_1 - \lambda & p_2 & \dots & p_{n-1} & p_n \\ 1 & -\lambda & 0 & \dots & 0 \\ 0 & 1 & -\lambda & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & -\lambda \end{vmatrix} = (p_1 - \lambda)(-\lambda)^{n-1} - \begin{vmatrix} p_2 & \dots & p_{n-1} & p_n \\ 1 & -\lambda & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 1 & -\lambda \end{vmatrix} =$$

$$= \dots = (-1)^n(\lambda^n - p_1\lambda^{n-1} - \dots - p_{n-1}\lambda - p_n). \tag{3.7}$$

Идея метода Данилевского проста: с помощью элементарных преобразований подобия привести данную матрицу A к форме Фробениуса A'. Поскольку преобразования подобия сохраняют спектр матрицы, характеристические многочлены матриц A и A' будут совпадать, т. е. искомый характеристический многочлен может быть вычислен по формуле (3.7).

Рассмотрим алгоритм на примере матрицы A размерности 4. Будем последовательно приводить строки матрицы к нужному виду, начиная с последней.

1. Для начала «делаем единицу» на позиции (4,3): делим третий столбец на $\alpha = a_{43}$. Чтобы сохранить спектр в соответствии с полученным выше правилом, нужно дополнить это преобразование до преобразования подобия, т. е. умножить третью строку на α. В результате указанных операций получим матрицу

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & 1 & \times \end{bmatrix}.$$

2. «Делаем нули» в последней строке: для $j \neq 3$ вычтем из j-го столбца 3-й, умноженный на $\alpha = a_{4j}$. Каждая такая операция должна быть дополнена до преобразования подобия, для чего необходимо к 3-й строке добавлять j-ю, умноженную на α. Получим в итоге

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

3. Аналогично поступим с третьей и второй строками. При этом сделанные ранее нули и единицы не «портятся»:

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & 0 & 1 & 0 \end{bmatrix} \mapsto \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mapsto \begin{bmatrix} \times & \times & \times & \times \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Базовый алгоритм метода Данилевского

1: **for** $i = \overline{n, 2}$ **do**
2: $\quad c \leftarrow a_{i,i-1}$
3: $\quad a_{i-1} \leftarrow a_{i-1}/c$;
4: $\quad a_{i-1} \leftarrow c\, a_{i-1}$
5: $\quad$ **for** $j \neq i$ **do**
6: $\quad\quad c \leftarrow a_{ij}$
7: $\quad\quad a_j \leftarrow a_j - c a_{i-1}$

8: $\underline{a_{i-1}} \leftarrow \underline{a_{i-1}} + c\underline{a_j}$
9: **end for**
10: **end for**

Напомним, что здесь a_j — j-й вектор-столбец матрицы A; $\underline{a_i}$ — i-я строка матрицы A; n — размерность матрицы A.

Для эффективной программной реализации строк 6 и 7, как и в методе Гаусса, следует учитывать, что с ходом преобразований нижняя часть матрицы будет содержать все больше нулевых элементов. После выполнения приведенного алгоритма (в невырожденном случае) на месте матрицы A будет находится ее форма Фробениуса.

Выбор главного элемента. Для минимизации погрешностей округления при машинной реализации необходимо выбирать главный элемент. Для этого перед началом i-го шага нужно выбрать максимальный по модулю элемент a_{ij^*} среди a_{ij} для $j = \overline{1, i-1}$. После этого меняем местами столбцы $i-1$ и j^*, а также соответствующие строки (для сохранения спектра).

Вырожденный случай. Предположим, на i-м этапе не удается выбрать главный элемент. Это означает, что матрица имеет вид

$$A = \left[\begin{array}{c|c} A_1 & \boxtimes \\ \hline 0 & A_2 \end{array}\right],$$

где A_1, A_2 — квадратные блоки размерности $n-i$ и i соответственно, причем A_2 имеет форму Фробениуса. Тогда имеем

$$|A - \lambda I| = |A_1 - \lambda I_1| \cdot |A_2 - \lambda I_2|.$$

Здесь I_1 и I_2 — единичные матрицы, размерность которых соответствует блокам A_1 и A_2. Второй множитель мы можем вычислить сразу, поэтому остается лишь привести к форме Фробениуса матрицу A_1.

3.4. Метод вращений Якоби

3.4.1. Преобразование вращения

Для изучения следующего метода решения проблемы собственных значений нам необходимо предварительно изучить важный тип линейных преобразований (матриц) — преобразования вращения, которые являются частным случаем ортогональных преобразований.

Определение 3.3. Линейное преобразование $A : \mathbb{R}^n \to \mathbb{R}^n$ называется *ортогональным*, если оно сохраняет длины векторов:

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

Матрица ортогонального преобразования называется ортогональной.

Возможны также следующие эквивалентные определения ортогонального преобразования:

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

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

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

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

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.

Преобразование вращения

Определение 3.4. *Матрицей элементарного вращения* называется матрица вида

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

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.

$\triangleright_1$ 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:

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

3.4.2. Симметричная проблема собственных значений

Пусть матрица A — вещественная и симметричная: $A^T = A$. Проблема собственных значений для такой матрицы является более простой, чем в общем случае, поскольку:

- A диагонализируема: существует матрица X такая, что

$$X^{-1}AX = D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n), \tag{3.9}$$

причем X — ортогональна;

- все λ_i вещественны.

Также из (3.9) имеем $AX = XD$, т. е. столбцы матрицы X являются собственными векторами A:

$$Ax_i = \lambda_i x_i.$$

3.4.3. Общая схема вращений Якоби

Метод вращений Якоби позволяет вычислить все собственные значения и векторы вещественной симметричной матрицы, т. е. решает полную проблему собственных значений для такой матрицы. Суть метода состоит в построении последовательности матриц

$$A^{(0)} = A, \quad A^{(m)} = X_m^T A^{(m-1)} X_m, \quad m = 1, 2, \ldots,$$

где X_m — матрицы элементарного вращения вида (3.8), которые строятся таким образом, что

$$A^{(m)} \xrightarrow[m\to\infty]{} D, \tag{3.10}$$

где D — некоторая диагональная матрица. Таким образом, для достаточно большого M получим

$$A^{(M)} \approx \operatorname{diag}(\lambda_1, \ldots, \lambda_n), \quad X \approx X^{(M)} = X_1 X_2 \ldots X_M.$$

В результате на диагонали матрицы $A^{(M)}$ будут находиться приближенные собственные значения матрицы A, а приближенные собственные векторы будут столбцами матрицы $X^{(M)}$.

Основной вопрос теперь состоит в том, каким образом следует выбирать матрицы вращения X_m, чтобы достичь сходимости к диагональной матрице. Для ответа нам понадобится следующее понятие.

Определение 3.5. *Нормой Фробениуса* называется матричная норма $\|\cdot\|_F$, определяемая как

$$\|A\|_F = \sqrt{\sum_{i,j}^{n} |a_{ij}|^2}.$$

$\triangleright_2$ Докажите, что ортогональные преобразования подобия сохраняют норму Фробениуса.

Рассмотрим величины

$$\operatorname{off}(A) = \sum_{i\neq j}^{n} a_{ij}^2, \quad \operatorname{on}(A) = \sum_{i=1}^{n} a_{ii}^2.$$

По определению имеем

$$\operatorname{on}(A) + \operatorname{off}(A) = \|A\|_F^2.$$

Условие сходимости (3.10) теперь можно записать в эквивалентом виде

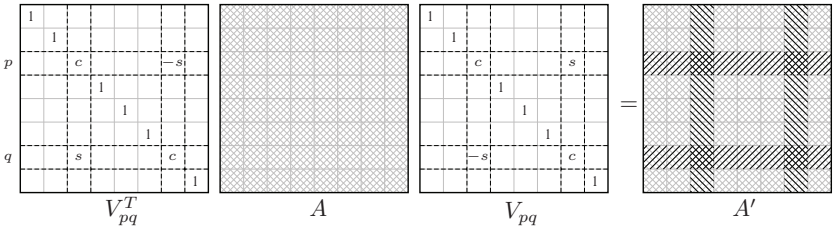
$$\operatorname{off}(A^{(m)}) \xrightarrow[m\to\infty]{} 0.$$

Как уже говорилось, матрица X_m, определяющая переход от $A^{(m-1)}$ к $A^{(m)}$, является матрицей элементарного вращения (3.8). Таким образом, $X_m = V_{pq}(\theta)$ определяется тремя параметрами: p, q и θ. Чтобы

понять, каким образом следует выбирать эти параметры, рассмотрим, как преобразование подобия

$$A \mapsto A' = V_{pq}^T A V_{pq} \tag{3.11}$$

изменяет матрицу A (см. рисунок).



Видно, что преобразование (3.11) изменяет в матрице A только строки и столбцы с индексами p и q. В частности, диагонали матриц A и A' отличаются лишь элементами на позициях (p,p) и (q,q). Введем следующие обозначения: $a_{pp} = a$, $a_{qq} = b$, $a'_{pp} = \alpha$, $a'_{qq} = \beta$. Тогда

$$\mathrm{on}(A') = \mathrm{on}(A) - a^2 - b^2 + \alpha^2 + \beta^2. \tag{3.12}$$

Обозначим также $a_{pq} = a_{qp} = e$, $a'_{pq} = a'_{qp} = \varepsilon$. Из (3.11) имеем соотношение

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a & e \\ e & b \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} \alpha & \varepsilon \\ \varepsilon & \beta \end{bmatrix}. \tag{3.13}$$

Поскольку ортогональное преобразование подобия сохраняет норму Фробениуса, получим

$$\alpha^2 + \beta^2 + 2\varepsilon^2 = a^2 + b^2 + 2e^2.$$

Используем это тождество в (3.12):

$$\mathrm{on}(A') = \mathrm{on}(A) + 2(e^2 - \varepsilon^2),$$

что с учетом

$$\|A\|_F = \|A'\|_F$$

равносильно

$$\mathrm{off}(A') = \mathrm{off}(A) - 2(e^2 - \varepsilon^2). \tag{3.14}$$

Из равенства (3.14) видно, что наименьшее значение $\mathrm{off}(A')$ достигается, если величины s и c (т. е. угол θ) выбраны таким образом, что $\varepsilon = a'_{pq} = 0$. Для того чтобы величина $\mathrm{off}(A)$ уменьшилась как можно больше, нужно выбирать в качестве обнуляемого элемента a_{pq} максимальный по модулю недиагональный элемент матрицы A.

Исходя из вышесказанного, получим следующий общий алгоритм.

Базовый алгоритм метода Якоби. Пока $\mathrm{off}(A)$ недостаточно мало:
1) среди a_{ij} для $i < j$ выбрать максимальный по модулю a_{pq};
2) выбрать c и s таким образом, чтобы после преобразования (3.11) получилось $a'_{pq} = 0$;
3) $A \leftarrow V_{pq}^T A V_{pq}$.

Согласно (3.14), для этого алгоритма имеем

$$\mathrm{off}(A') = \mathrm{off}(A) - 2e^2. \tag{3.15}$$

Здесь $A' = A^{(m+1)}$, $A = A^{(m)}$, $e = a_{pq}^{(m)}$. *Важно понимать, что каждый шаг алгоритма в общем случае «портит» нули, сделанные на предыдущем шаге.*

Оценим скорость сходимости алгоритма. Если $a_{pq} = e$ — максимальный по модулю недиагональный элемент матрицы A, то справедлива оценка

$$\mathrm{off}(A) \leqslant 2Ne^2,$$

где $N = n(n-1)/2$. Тогда из (3.15) получим

$$\mathrm{off}(A') \leqslant \left(1 - \frac{1}{N}\right) \mathrm{off}(A),$$

что означает

$$\mathrm{off}(A^{(m)}) \leqslant \left(1 - \frac{1}{N}\right)^m \mathrm{off}(A). \tag{3.16}$$

В частности, при $n = 2$ очевидно имеем сходимость за одну итерацию.

Согласно (3.16), метод вращений Якоби сходится со скоростью геометрической прогрессии. Однако эта оценка слишком груба, и на практике метод сходится быстрее.

3.4.4. Расчетные формулы метода

Для окончательного определения метода осталось вывести формулы, по которым вычисляются элементы матрицы V_{pq} во втором пункте

алгоритма. Из (3.13) получим

$$\alpha = a\,c^2 + b\,s^2 - 2e\,sc, \tag{3.17а}$$
$$\beta = a\,s^2 + b\,c^2 + 2e\,sc, \tag{3.17б}$$
$$\varepsilon = e(c^2 - s^2) + (a - b)sc = 0. \tag{3.17в}$$

Для решения уравнения (3.17в) введем переменные

$$t = \operatorname{tg}\theta = \frac{s}{c}, \quad z = \frac{b-a}{2e}.$$

После простых преобразований из (3.17в) получим

$$t^2 + 2zt - 1 = 0,$$

откуда

$$t = -z \pm \sqrt{z^2 + 1}. \tag{3.18}$$

Использование этой формулы на практике приводит к большим ошибкам округления, поэтому ее нужно переписать в более подходящем для машинных вычислений виде. Домножив (3.18) на $z \pm \sqrt{z^2+1}$, получим

$$t = \frac{1}{z \pm \sqrt{z^2 + 1}}.$$

Важно выбрать из этих двух корней наименьший по модулю. При $z \neq 0$ он равен

$$t = \frac{\operatorname{sign} z}{|z| + \sqrt{z^2 + 1}}. \tag{3.19}$$

После этого вычислим

$$c = \frac{1}{\sqrt{1 + t^2}}, \quad s = tc. \tag{3.20}$$

Замечание 3.3. В случае $z = 0$ необходимо использовать формулу (3.18), которая дает $t = \pm 1$, или $s = c = 1/\sqrt{2}$. Обратите внимание, что формула (3.19) в этом случае даст неверный результат.

Теоретически теперь нам остается с помощью найденных значений s и c вычислить A' по формуле (3.11). Однако для того чтобы метод был эффективен, вычисления организуются следующим образом.

Как видно на приведенном выше рисунке, преобразование $A \mapsto A'$ заключается в изменении только строк и столбцов с индексами p и q в

матрице A. Для диагональных элементов $a'_{pp} = \alpha$ и $a'_{qq} = \beta$ имеют место формулы (3.17а), (3.17б), $a'_{pq} = a'_{qp} = 0$ по построению, а для $j \neq p$, $j \neq q$ из (3.11) получим

$$\begin{aligned} a'_{jp} &= a'_{pj} = ca_{pj} - sa_{qj}, \\ a'_{jq} &= a'_{qj} = sa_{pj} + ca_{qj}. \end{aligned} \tag{3.21}$$

Для вычислительной устойчивости нужно представить вышеперечисленные формулы в виде

$$a'_{ij} = a_{ij} + \delta_{ij},$$

где δ_{ij} — некоторая поправка. Так, из (3.17а), (3.17б) с учетом (3.17в) получим

$$\begin{aligned} a'_{pp} &= a_{pp} - ta_{pq}, \\ a'_{qq} &= a_{qq} + ta_{pq}, \end{aligned} \tag{3.22}$$

а вместо (3.21) имеем

$$\begin{aligned} a'_{pj} &= a_{pj} - s(a_{qj} + \tau a_{pj}), \\ a'_{qj} &= a_{qj} + s(a_{pj} - \tau a_{qj}), \text{ где } \quad \tau = \frac{s}{1+c}. \end{aligned} \tag{3.23}$$

▷3 Запишите часть вычислительного алгоритма, отвечающую за построение системы собственных векторов матрицы A.

Замечания по практической реализации

1) В памяти следует хранить только верхний треугольник матрицы A.
2) Матрицы вращения V_{pq} в память записывать не следует, все преобразования осуществляются с использованием только величин c и s.
3) Преобразования (3.11) осуществляются по формулам (3.22), (3.23), при этом нужно грамотно учитывать симметрию матрицы.
4) При больших n выбор максимального по модулю элемента требует слишком много времени, поэтому, как правило, элементы a_{pq} для обнуления выбирают циклически: $a_{12}, \ldots, a_{1n}, a_{23}, \ldots, a_{2n}, \ldots, a_{n-1,n}$. Обычно для получения решения в пределах машинной точности достаточно 5-6 таких проходов. При этом на первых 2-3 проходах, если модуль a_{pq} достаточно мал, его пропускают.

3.5. *QR*-алгоритм

QR-алгоритм является одним из наиболее известных и популярных методов для вычисления всех (в том числе и комплексных) собственных значений произвольной квадратной матрицы. Как это обычно бывает, для его изучения необходимы некоторые предварительные сведения.

3.5.1. Предварительные сведения

QR-разложение

Определение 3.6. *QR-разложением* квадратной матрицы A называется ее представление в виде произведения ортогональной матрицы Q и верхнетреугольной R:

$$A = QR.$$

Такое разложение существует всегда, причем не только для квадратных матриц. Если исходная матрица вырождена, то на диагонали у матрицы R будет как минимум один нулевой элемент. В противном случае таких элементов не будет.

Построение QR-разложения по сути не отличается от LU-разложения: просто вместо элементарных преобразований для приведения матрицы A к верхнетреугольному виду следует использовать ортогональные (например, известные нам уже преобразования вращения). Тогда после некоторого числа преобразований получим

$$Q_N \dots Q_2 Q_1 A = R,$$

где матрицы Q_k — ортогональные, а R — верхнетреугольная. Отсюда имеем искомое разложение $A = QR$, где

$$Q = (Q_N \dots Q_2 Q_1)^{-1} = Q_1^T Q_2^T \dots Q_N^T.$$

Матрица Q очевидно ортогональна.

Форма Хессенберга

Определение 3.7. Говорят, что квадратная матрица A имеет *форму Хессенберга*, если

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

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

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

Теорема 3.1. *Любая квадратная матрица может быть приведена к форме Хессенберга преобразованием подобия.*

Доказательство. Докажем теорему с помощью элементарных преобразований подобия по схеме, аналогичной методу Данилевского, причем сходство с методом Гаусса в данном случае будет еще более явным.

Преобразование матрицы к форме Хессенберга будем осуществлять по столбцам слева направо. Предположим, что в первых $(k-1)$ столбцах уже обнулены все необходимые элементы. Рассмотрим для наглядности случай $n = 6$, $k = 3$:

$$\left[\begin{array}{cc|c|ccc} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ \hline & \times & \times & \times & \times & \times \\ & & \otimes & \times & \times & \times \\ & & * & \times & \times & \times \\ & & * & \times & \times & \times \end{array}\right].$$

На данной схеме звездочками обозначены подлежащие обнулению элементы k-го столбца, а кружком обведен главный элемент, с помощью которого эти обнуления и осуществляются элементарными преобразованиями строк. Единственное отличие от метода Гаусса при этом состоит в выполнении «дополняющих» преобразований со столбцами, сохраняющих спектр исходной матрицы. При этом столбцы от 1 до $k-1$ затрагиваться не будут.

Соответствующий алгоритм k-го шага алгоритма выглядит следующим образом:

1: **for** $i = \overline{k+2, n}$ **do**
2: $\quad \gamma \leftarrow a_{ik}/a_{k+1,k}$
3: $\quad \underline{a}_i \leftarrow \underline{a}_i - \gamma \underline{a}_{k+1}$
4: $\quad a_{k+1} \leftarrow a_{k+1} + \gamma a_i$
5: **end for**

При этом для исключения деления на ноль и повышения вычислительной устойчивости перед каждым шагом необходимо выбирать главный элемент, как и в методе Гаусса, но только среди элементов $a_{i,k}$ для

$i = \overline{k+1, n}$. После перестановки строк переставляются и соответствующие столбцы.

За $(n-2)$ таких шагов исходная матрица будет приведена к форме Хессенберга с сохранением спектра. □

Замечание 3.4. Вместо элементарных в доказательстве теоремы можно использовать ортогональные преобразования.

3.5.2. Общая схема *QR*-алгоритма

Теперь рассмотрим сам алгоритм. 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 3.2 (QR-алгоритм, упрощенная формулировка)**.** *Последовательность матриц $\{A_k\}_{k=0}^{\infty}$, построенная по описанному выше правилу, при выполнении определенных условий*[3] *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 3.5. Коэффициенты блоков $\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}$ не обязаны сходиться к какому-то пределу, сходятся их собственные значения.

Замечание 3.6. QR-алгоритм *никогда* не применяется к «неподготовленной» матрице A, потому что это слишком трудоемко. Сначала матрицу необходимо привести в форме Хессенберга и лишь потом начинать итерации QR-алгоритма.

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

3.5.3. *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:

$$\underbrace{V_{n-1,n}\dots V_{23}V_{12}}_{Q^{-1}}A = R, \tag{3.24}$$

где матрицы $V_{i,i+1} = V_{i,i+1}(\theta_i)$ имеют вид (3.8) и как обычно 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 (3.24), 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).

$\triangleright_1$ Докажите это утверждение.

Чтобы полностью определить алгоритм, осталось вывести формулы для параметров $\{c_i, s_i\}$, обеспечивающие обнуление «поддиагональных»

элементов в ходе (3.24). 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

$$c = \frac{\alpha}{\sqrt{\alpha^2 + \beta^2}}, \quad s = \frac{\beta}{\sqrt{\alpha^2 + \beta^2}}. \tag{3.25}$$

$\triangleright_2$ Запишите алгоритм одной итерации QR-алгоритма для матрицы в форме Хессенберга. Оцените его сложность.

3.5.4. 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 (теорема 3.1). То, как выполнять QR-итерации с использованием преобразований вращения (пункт 2), тоже (подп. 3.5.3).

Для полной ясности приведем пример «извлечения» собственных значений в пункте 3 (см. основную теорему 3.2). 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 3.7. 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.

Глава 4

МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ И СИСТЕМ

4.1. Решение нелинейных уравнений

4.1.1. Введение

Рассмотрим задачу нахождения корней нелинейных уравнений вида

$$f(x) = 0, \quad f : \mathbb{R} \to \mathbb{R}. \tag{4.1}$$

Определение 4.1. Если $f^{(i)}(x^*) = 0 \; \forall i = \overline{0, m-1}$, и $f^{(m)}(x^*) \neq 0$, то говорят, что x^* — корень уравнения (4.1) кратности m.

В общем случае задача (4.1) весьма сложна, поэтому перед тем как приступать к ее решению, необходимо провести следующий анализ:
1) определить количество корней и, желательно, их кратность;
2) *отделить (локализовать) корни* — определить непересекающиеся интервалы, в каждом из которых находится по одному корню.

Чем точнее отделены корни, тем больше шансов на успешное решение задачи. Основное средство, которое мы будем применять для отделения корней — известный из анализа факт: *если непрерывная на отрезке* $[a, b]$ *функция* f *принимает на концах этого отрезка значения противоположных знаков, то на* $[a, b]$ *находится как минимум один корень уравнения* $f(x) = 0$.

4.1.2. Основные теоретические сведения

Все рассматриваемые нами далее методы являются *итерационными*. Это означает, что они заключаются в построении *по единому прави-*

лу последовательности чисел (x_k) таким образом, чтобы $x_k \to x^*$ при $k \to \infty$, где x^* — искомый корень уравнения $f(x) = 0$. Большинство этих методов можно будет представить в виде

$$x_{k+1} = \varphi(x_k), \quad k = 0, 1, 2, \ldots, \tag{4.2}$$

где x_0 — начальное приближение; $\varphi : \mathbb{R} \to \mathbb{R}$ — некоторая функция, однозначно определяющая метод. Функция φ, естественно, должна зависеть от f. Переходя к пределу в (4.2), видим, что искомое решение x^* является также и корнем уравнения

$$x = \varphi(x). \tag{4.3}$$

Определение 4.2. Величину $\varepsilon_k = x_k - x^*$ будем называть погрешностью k-й итерации.

Основная характеристика качества метода — скорость, с которой ε_k стремится к 0.

Определение 4.3. Если при всех достаточно больших k имеет место неравенство

$$|\varepsilon_{k+1}| \leqslant C|\varepsilon_k^p|,$$

где $C < \infty$, $p \in \mathbb{R}$, то говорят, что метод, по которому построена последовательность (x_k), имеет *порядок (скорость) сходимости* p. Если $p = 1$, то сходимость называют *линейной*, при $1 < p < 2$ — *сверхлинейной*, $p = 2$ — *квадратичной* и т. д.

Для методов вида (4.2) существует относительно простой способ определения порядка сходимости.

Лемма 4.1 (Условие порядка)**.** *Пусть функция φ имеет p непрерывных производных в окрестности корня x^* и*

$$\varphi'(x^*) = \varphi''(x^*) = \ldots = \varphi^{(p-1)}(x^*) = 0, \quad \varphi^{(p)}(x^*) \neq 0.$$

Тогда итерационный процесс (4.2) *имеет порядок сходимости p и справедлива оценка*

$$|\varepsilon_{k+1}| \leqslant \frac{M_p}{p!}|\varepsilon_k^p|, \tag{4.4}$$

где M_p — максимум $|\varphi^{(p)}(x)|$ в некоторой фиксированной окрестности x^*.

Доказательство. Воспользуемся формулой Тейлора:

$$\varepsilon_{k+1} = x_{k+1} - x^* = \varphi(x_k) - x^* = \varphi(x^* + \varepsilon_k) - x^* = \varphi(x^*) + \\ + \varphi'(x^*)\varepsilon_k + \ldots + \frac{\varphi^{(p-1)}(x^*)}{(p-1)!}\varepsilon_k^{p-1} + \frac{\varphi^{(p)}(\xi)}{p!}\varepsilon_k^p - x^* = \frac{\varphi^{(p)}(\xi)}{p!}\varepsilon_k^p,$$

где $\xi \in (x^*, x^* + \varepsilon_k)$. Отсюда получим оценку (4.4). □

4.1.3. Метод бисекции

Если корень локализован на отрезке $[a, b]$ и $f(a)f(b) < 0$, то наиболее простым способом приближенного вычисления такого корня является метод *бисекции* (*дихотомии*, *половинного деления*). При этом данный метод отличается высокой надежностью: в случае непрерывной функции f он всегда сходится к одному из корней уравнения, расположенных на $[a, b]$.

Алгоритм метода приведен ниже и проиллюстрирован на рис. 4.1.

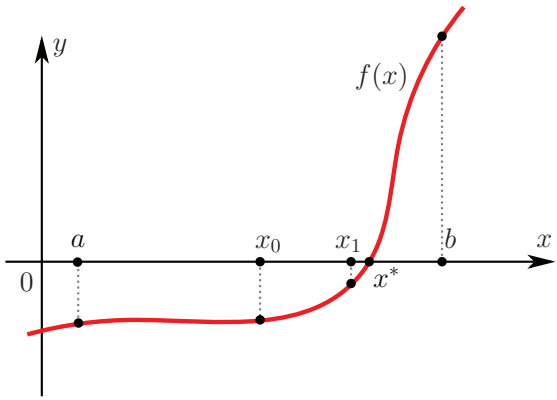


Рис. 4.1. Метод бисекции

Метод бисекции, общая схема

```
while |b − a| > 2ε do
    x ← (a + b)/2
    if f(x)f(a) < 0 then
        b ← x
```

```
5:      else
6:          a ← x
7:      end if
8:  end while
```

Здесь ε — точность, с которой требуется найти корень.

$\triangleright_1$ Определите порядок сходимости метода бисекции.

$\triangleright_2$ Найдите априорную оценку погрешности $|\varepsilon_k| = |x_k - x^*|$ для метода бисекции.

4.1.4. Метод Ньютона

В случаях, когда известен вид $f'(x)$, метод Ньютона является одним из наиболее эффективных. Рассмотрим некоторое приближение x_k. Приблизим график функции f касательной в точке x_k:

$$f(x) \approx y(x) = f(x_k) + (x - x_k)f'(x_k).$$

В качестве следующего приближения к решению уравнения $f(x) = 0$ возьмем корень уравнения $y(x) = 0$ (рис. 4.2):

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \ldots . \tag{4.5}$$

Формула (4.5) и определяет метод Ньютона.

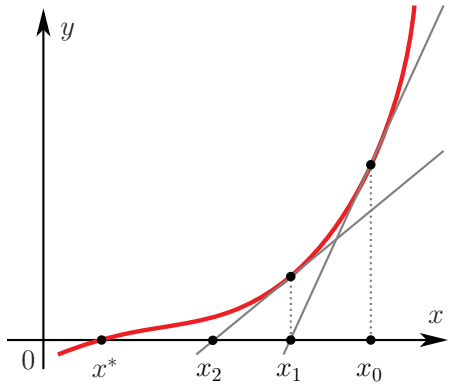


Рис. 4.2. Метод Ньютона

Исследуем порядок сходимости метода Ньютона. Запишем (4.5) в виде

$$x_{k+1} = \varphi(x_k), \quad \varphi(x) = x - \frac{f(x)}{f'(x)}. \tag{4.6}$$

Из (4.6) по лемме 4.1 легко получить порядок сходимости метода Ньютона:

$$\varphi'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)^2}, \tag{4.7}$$

откуда *при условии* $f'(x^*) \neq 0$ получим

$$\varphi'(x^*) = 0. \tag{4.8}$$

Вычисляя $\varphi''(x)$, увидим, что $\varphi''(x^*) \neq 0$, следовательно, *если* x^* — *корень кратности 1, то метод Ньютона имеет второй порядок сходимости*.

Предположим теперь, что x^* — корень кратности m. Тогда для достаточно гладкой функции f в его окрестности для точки $x = x^* + \delta$ имеем

$$f(x) = \underbrace{f(x^*) + f'(x^*)\delta + \ldots + \frac{f^{(m-1)}(x^*)}{(m-1)!}\delta^{m-1}}_{=0} + O(\delta^m),$$

т. е.

$$f(x) \approx c(x - x^*)^m,$$

где c — некоторая константа. Подставляя это представление в (4.7), получим

$$\varphi'(x) \approx \frac{m-1}{m},$$

следовательно, по формуле (4.4) имеем, что *в случае корня кратности* m *метод Ньютона сходится со скоростью геометрической прогрессии со знаменателем* $\dfrac{m-1}{m}$, *т. е. линейно*.

4.1.5. Модификации метода Ньютона

Метод Ньютона с постоянной производной

В случаях, когда производная $f'(x)$ неизвестна или ее слишком дорого вычислять, можно в методе Ньютона (4.5) заменить $f'(x_k)$ на какую-то

константу μ:

$$x_{k+1} = x_k - \frac{f(x_k)}{\mu}, \quad k = 0, 1, \ldots . \tag{4.9}$$

Очевидный выбор — $\mu \approx f'(x_0)$. Геометрическая интерпретация метода в данном случае изображена на рис. 4.3.

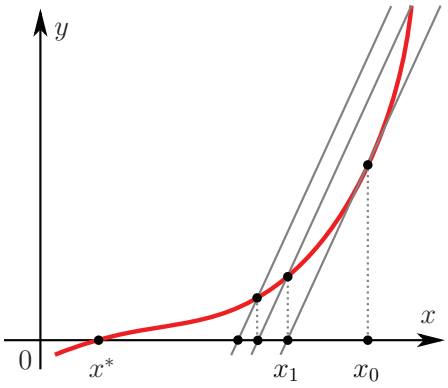


Рис. 4.3. Метод Ньютона с постоянной производной

▷3 Определите порядок сходимости метода.

Метод секущих

Наиболее естественное видоизменение метода Ньютона состоит в замене касательной прямой на секущую при геометрическом построении метода.

Пусть имеется два приближения к корню: x_k и x_{k-1}. Составим уравнение секущей для функции f по этим точкам:

$$y(x) = f(x_k) + (x - x_k)\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

Положим x_{k+1} равным корню уравнения $y(x) = 0$:

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}, \quad k = 0, 1, \ldots . \tag{4.10}$$

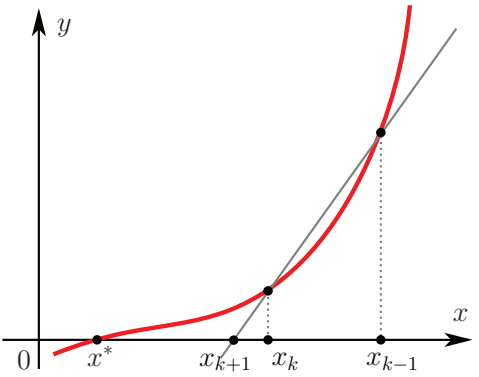


Рис. 4.4. Метод секущих

Это — метод секущих. Его геометрическая интерпретация изображена на рис. 4.4. Этот метод нельзя представить в виде (4.2), поэтому порядок сходимости метода вычисляется нетривиально. Он равен золотому сечению:

$$m = \frac{1+\sqrt{5}}{2} \approx 1{,}618,$$

т. е. метод секущих сходится сверхлинейно. В качестве начальных приближений x_1 и x_0 можно брать концы отрезка $[a, b]$, на котором локализован корень.

Метод хорд

У метода секущих есть недостаток: члены последовательности x_k могут выходить за пределы отрезка локализации $[a, b]$, что может привести к расходимости итерационного процесса.

$\triangleright_4$ Приведите пример, когда при $x_0 = a$ и $x_1 = b$ приближение x_3, построенное по методу секущих, не лежит в $[a, b]$.

Избежать этой проблемы можно, зафиксировав один из концов секущей линии:

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_0}{f(x_k) - f(x_0)}, \tag{4.11}$$

причем в качестве x_0 нужно брать тот конец отрезка $[a, b]$, в котором знаки f и f'' совпадают.

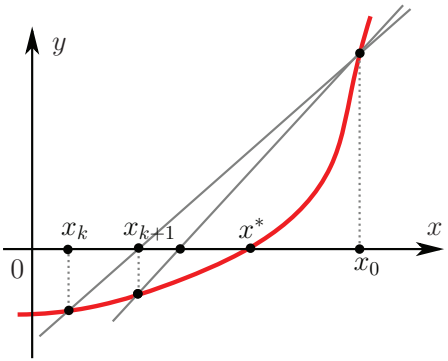


Рис. 4.5. Метод хорд

$\triangleright_5$ Обоснуйте это правило.

$\triangleright_6$ Определите порядок сходимости метода хорд.

Геометрический смысл метода хорд (4.11) изображен на рис. 4.5.

4.2. Решение систем нелинейных уравнений

4.2.1. Постановка задачи

Систему нелинейных уравнений

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0, \\ \cdots\cdots\cdots\cdots\cdots \\ f_n(x_1, \ldots, x_n) = 0 \end{cases} \tag{4.12}$$

запишем в векторном виде

$$f(x) = 0, \tag{4.13}$$

где $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}^n$,

$$f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{bmatrix}, \quad f_i : \mathbb{R}^n \to \mathbb{R}. \tag{4.14}$$

Как и ранее, точное решение уравнения (4.13) обозначим x^*. В дальнейшем будем предполагать, что f достаточно гладкая, т. е. существует необходимое число частных производных от f_i.

4.2.2. Метод Ньютона

В скалярном случае метод Ньютона имеет вид

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

Для начала можно *формально* обобщить этот метод на случай системы (4.13):

$$x^{k+1} = x^k - f'(x^k)^{-1} f(x^k), \tag{4.15}$$

где $f' = \dfrac{\partial f}{\partial x}$ — матрица Якоби,

$$f' = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} \dfrac{\partial f_1}{\partial x_2} \cdots \dfrac{\partial f_1}{\partial x_n} \\ \dfrac{\partial f_2}{\partial x_1} \dfrac{\partial f_2}{\partial x_2} \cdots \dfrac{\partial f_2}{\partial x_n} \\ \cdots\cdots\cdots\cdots \\ \dfrac{\partial f_n}{\partial x_1} \dfrac{\partial f_n}{\partial x_n} \cdots \dfrac{\partial f_n}{\partial x_n} \end{bmatrix}. \tag{4.16}$$

Для обоснования формулы (4.15) рассмотрим $x \in \mathbb{R}^n$, приращение $\Delta x \in \mathbb{R}^n$ и воспользуемся формулой Тейлора для функции нескольких переменных:

$$f_i(x + \Delta x) = f_i(x) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(x)\Delta x_j + O(\|\Delta x\|^2), \quad i = \overline{1, n}.$$

В векторной форме эти равенства можно записать как

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + O(\|\Delta x\|^2). \tag{4.17}$$

Пусть x^k — некоторое приближение к x^*. В идеале нам нужно найти такую поправку Δx^*, для которой $x^k + \Delta x^* = x^*$, или

$$f(x^k + \Delta x^*) = 0.$$

Отбрасывая остаточный член в формуле (4.17), имеем

$$0 = f(x^k + \Delta x^*) \approx f(x^k) + f'(x^k)\Delta x^* \Rightarrow \Delta x^* \approx -f'(x^k)^{-1} f(x^k) = \Delta x^k.$$

Вычисляя

$$x^* \approx x^{k+1} = x^k + \Delta x^k,$$

получим формулу (4.15).

Замечание 4.1. Важно понимать, что *формула* (4.15) *в явном виде практически никогда не используется для расчетов*, так как для этого нужно иметь аналитический вид матрицы $f'(x)^{-1}$.

Вместо этого каждую итерацию метода Ньютона реализуют в два этапа:

1) сначала находят поправку Δx^k как решение СЛАУ

$$f'(x^k)\Delta x^k = -f(x^k); \tag{4.18а}$$

2) затем вычисляют

$$x^{k+1} = x^k + \Delta x^k. \tag{4.18б}$$

$\triangleright_1$ Оцените сложность реализации метода Ньютона по формулам (4.18).

При больших n метод Ньютона становится слишком трудоемким. Кроме этого, он не применим, если нет возможности вычислить f', что довольно часто случается на практике. Для таких случаев разработан ряд модификаций метода Ньютона, направленных на снижение вычислительных затрат.

4.2.3. Метод Ньютона с постоянным якобианом

Для того чтобы снизить затраты на решение СЛАУ (4.18а), рассматривают метод Ньютона с «замороженной» матрицей Якоби:

$$x^{k+1} = x^k - J_0^{-1} f(x_k), \tag{4.19}$$

где $J_0 = f'(x_0)$.

Для реализации такого метода достаточно один раз построить LU-разложение матрицы J_0, а затем с помощью него решать СЛАУ вида

$$J_0\Delta x^k = -f(x_k)$$

для нахождения поправок за $O(n^2)$ операций. Данная модификация на порядок снижает вычислительные затраты, но при этом, конечно, снижается скорость сходимости метода.

4.2.4. Обобщения метода секущих

Метод секущих (4.10) можно рассматривать с двух сторон.
1) С одной стороны, он может быть получен путем модификации метода Ньютона с помощью замены

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}},$$

т. е. метод секущих основан на приближенном вычислении f'.
2) С другой стороны, данный метод определяет уравнение секущей $y(x)$, которая пересекает график f в точках x_k и x_{k-1} (именно так мы этот метод строили в предыдущем пункте). С этой точки зрения мы строим приближение для f.

В одномерном случае два описанных подхода дают один и тот же метод, однако в многомерном случае мы получим два разных семейства методов.

Первый способ: приближенное вычисление якобиана. Рассмотрим набор приращений $\{h_1, \ldots, h_n\}$, $h_i \in \mathbb{R}$, и определим приближения для частных производных f по формуле

$$\frac{\partial f_i}{\partial x_j}(x) \approx \frac{f_i(x_1, \ldots, x_j + h_j, \ldots x_n) - f_i(x_1, \ldots, x_n)}{h_j} = \psi_{ij}(x), \quad (4.20)$$

откуда имеем $f'(x) \approx \Psi(x)$. Соответствующий метод имеет вид

$$x^{k+1} = x^k - \Psi(x^k)^{-1} f(x_k). \quad (4.21)$$

$\triangleright_2$ Укажите достоинства и недостатки этого метода.

Кроме (4.20) можно рассматривать и другие варианты приближенного вычисления $f'(x)$.

Второй способ приводит к методу Бройдена.

4.2.5. Метод Бройдена

Обобщением понятия прямой, задаваемой уравнением $y = ax + b$, для пространства $\mathbb{R}^n$ является множество пар $(x, y) \in \mathbb{R}^n \times \mathbb{R}^n$, таких, что

$$y = Ax + b, \quad (4.22)$$

где x, y, b — векторы из $\mathbb{R}^n$; A — квадратная матрица.

Пусть нам известны два приближения к x^*: x^k и x^{k-1}. Тогда секущая вида (4.22) определяется соотношениями

$$\begin{cases} Ax^{k-1} + b = f(x^{k-1}), \\ Ax^k + b = f(x^k). \end{cases} \tag{4.23}$$

Очевидно, что эти условия не определяют A и b однозначно, но пока предположим, что мы каким-то образом нашли A. Тогда $f(x) \approx Ax + b$ и следующее приближение x^{k+1} строится из соотношения

$$Ax^{k+1} + b = 0 \quad \Rightarrow \quad x^{k+1} = -A^{-1}b,$$

откуда, выражая из (4.23) $b = f(x^k) - Ax^k$, получим знакомую формулу

$$x^{k+1} = x^k - A^{-1}f(x^k).$$

Общая схема итерации многомерного метода секущих

1) По двум предыдущим приближениям x^k и x^{k-1} вычисляется матрица $A = A_k$, определяющая уравнение секущей.

2) Находится новое приближение по формуле

$$x^{k+1} = x^k - A_k^{-1}f(x^k). \tag{4.24}$$

Рассмотрим теперь главный вопрос: как определять матрицу A из условий (4.23)? Это можно делать по-разному. Один из наиболее удачных способов был предложен Ч. Бройденом в 1965 г.

Обозначим $\Delta x = x^k - x^{k-1}$, $\Delta f = f(x^k) - f(x^{k-1})$. Из (4.23) имеем

$$A_k \Delta x = \Delta f. \tag{4.25}$$

Основная идея заключается в том, чтобы представить A_k в виде

$$A_k = A_{k-1} + u\Delta x^T, \tag{4.26}$$

где $u \in \mathbb{R}^n$ — вектор неизвестных параметров.

Подставляя (4.26) в (4.25), получим

$$(A_{k-1} + u\Delta x^T)\Delta x = \Delta f,$$

откуда

$$u = \frac{1}{\|\Delta x\|_2^2}\Big(\Delta f - A_{k-1}\Delta x\Big). \tag{4.27}$$

Здесь уже можно остановиться: по u находим A_k, после чего найдем x^{k+1} по формуле (4.24). При этом необходимо решить СЛАУ вида

$$A_k(x^{k+1} - x^k) = -f(x^k). \tag{4.28}$$

При определенной схеме организации вычислений решение таких систем можно получить за $O(n^2)$ операций, но описание такого алгоритма не входит в рамки нашего курса [18].

В самом начале процесса для получения приближения x^1 можно использовать метод Ньютона или метод (4.21). Тогда матрицей A_1 будет матрица $f'(x^0)$ или $\Psi(x^0)$ соответственно.

4.2.6. Другие методы

В заключение рассмотрим *нелинейный метод Гаусса – Зейделя*. По аналогии со случаем СЛАУ здесь в цикле по i от 1 до n по очереди вычисляются компоненты x_i^{k+1} путем решения скалярного уравнения вида

$$f_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i, x_{i+1}^k, \ldots, x_n^k) = 0, \tag{4.29}$$

относительно x_i. После этого полагается $x_i^{k+1} = x_i$. Для решения скалярных уравнений (4.29) могут быть использованы соответствующие методы, рассмотренные ранее.

$\triangleright_3$ Постройте нелинейные аналоги методов Якоби и релаксации.

4.3. Анализ сходимости итерационных процессов

Как уже не раз упоминалось, многие из рассмотренных нами методов решения нелинейных уравнений и систем представимы в виде $x^{k+1} = \varphi(x^k)$. Рассмотрим классический способ доказательства сходимости таких итерационных процессов.

4.3.1. Принцип сжимающих отображений

Расстояние между двумя точками $x, y \in \mathbb{R}^n$ условимся обозначать

$$\rho(x, y) = \|x - y\|. \tag{4.30}$$

Определение 4.4. Если для функции $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ существует константа $L < \infty$, такая, что

$$\rho(\varphi(x), \varphi(y)) \leqslant L\rho(x, y), \quad \forall x, y \in D \subset \mathbb{R}^n,$$

то говорят, что φ удовлетворяет *условию Липшица* (или просто *липшицева*) на множестве D. Число L называют *константой Липшица*.

Определение 4.5. *Сжимающим отображением* называют функцию φ, удовлетворяющую условию Липшица на $\mathbb{R}^n$ с константой $L < 1$. Если же это условие выполнено лишь на некоторой области $D \subset \mathbb{R}^n$, то такое отображение будем называть *локально-сжимающим*.

Схема действия сжимающего отображения $\mathbb{R}^2 \to \mathbb{R}^2$ изображена на рис. 4.6.

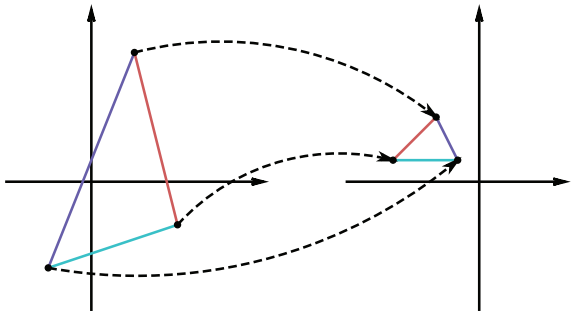


Рис. 4.6

Определение 4.6. Последовательность точек (x^k), $x^k \in \mathbb{R}^n$, называется *фундаментальной* или *последовательностью Коши*, если

$$\rho(x^k, x^m) \to 0 \quad \text{при} \quad k, m \to \infty.$$

Согласно критерию Коши, последовательность (x^k) сходится тогда и только тогда, когда она фундаментальна.

Теорема 4.1 (принцип сжимающих отображений)**.** *Пусть* $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ *— сжимающее отображение. Тогда уравнение*

$$\varphi(x) = x \tag{4.31}$$

имеет единственное решение $x = x^*$ *и для любого* $x_0 \in \mathbb{R}^n$ *последовательность*

$$x^{k+1} = \varphi(x^k), \quad k = 0, 1, 2, \ldots \tag{4.32}$$

сходится к x^**:*

$$\rho(x^k, x^*) \xrightarrow[k\to\infty]{} 0.$$

Доказательство. Рассмотрим последовательность

$$x^1 = \varphi(x^0),\ x^2 = \varphi(x^1),\ \ldots,\ x^{k+1} = \varphi(x^k),\ \ldots$$

и докажем, что она является фундаментальной:

$$\rho(x^k, x^{k+1}) = \rho\left(\varphi(x^{k-1}), \varphi(x^k)\right) \leqslant L\rho(x^{k-1}, x^k) \leqslant \ldots \leqslant L^k\rho(x^0, x^1).$$

Пусть $m > k$. Тогда по неравенству треугольника

$$\begin{aligned}\rho(x^k, x^m) &\leqslant \rho(x^k, x^{k+1}) + \rho(x^{k+1}, x^{k+2}) + \ldots + \rho(x^{m-1}, x^m) \leqslant \\ &\leqslant (L^k + L^{k+1} + \ldots + L^{m-1})\rho(x^0, x^1),\end{aligned}$$

откуда получим

$$\rho(x^k, x^m) \leqslant \frac{L^k}{1-L}\rho(x^0, x^1), \tag{4.33}$$

т. е. (x^k) фундаментальна ($L < 1$ по условию), и, следовательно, сходится к какому-то вектору $x^* \in \mathbb{R}^n$. Переходя к пределу в (4.32), получим, что x^* удовлетворяет уравнению (4.31).

Докажем единственность. Пусть $\exists\, x^{**} \neq x^*$ такой, что $\varphi(x^{**}) = x^{**}$. Это приводит к противоречию:

$$\rho(x^*, x^{**}) = \rho\left(\varphi(x^*), \varphi(x^{**})\right) \leqslant L\rho(x^*, x^{**}) < \rho(x^*, x^{**}). \qquad \square$$

Следствие 4.1. *Устремляя* $m \to \infty$ *в формуле* (4.33)*, получим априорную оценку погрешности:*

$$\rho(x^k, x^*) \leqslant \frac{L^k}{1-L}\rho(x^0, x^1). \tag{4.34}$$

$\triangleright_1$ Получите из формулы (4.34) оценку для количества итераций N, достаточного для получения решения с погрешностью, не превышающей ϵ.

Кроме оценки (4.34) можно получить и *апостериорную* оценку погрешности, которая, как правило, более точна, но для получения которой необходимо проделать k итераций. Положив в (4.34) $x^0 = x^{k-1}$, имеем

$$\rho(x^k, x^*) \leqslant \frac{L}{1-L}\rho(x^{k-1}, x^k). \tag{4.35}$$

Замечание 4.2. Функция ρ (метрика), которая измеряет расстояние между двумя точками, не обязательно должна иметь вид (4.30). Достаточно лишь, чтобы она удовлетворяла аксиомам метрики:
1) $\rho(x, y) = 0 \Leftrightarrow x = y$;
2) $\rho(x, y) = \rho(y, x)$;
3) $\rho(x, y) \leqslant \rho(x, z) + \rho(z, y)$.

Замечание 4.3. Принцип сжимающих отображений играет важную роль в функциональном анализе и других разделах математики. Он может быть сформулирован не только для $\mathbb{R}^n$, но и для произвольных метрических пространств, обладающих свойством *полноты*. В частности, с его помощью легко доказывается существование решений для обыкновенных дифференциальных уравнений, интегральных уравнений и т. д.

4.3.2. Локальный принцип сжимающих отображений

Рассмотрим теперь случай, когда отображение φ является лишь локально-сжимающим.

Определение 4.7. Гипершар радиусом r с центром в точке $x \in \mathbb{R}^n$ будем обозначать

$$B(x, r) = \{y \in \mathbb{R}^n \mid \rho(x, y) \leqslant r\}.$$

Лемма 4.2. *Пусть отображение φ имеет неподвижную точку $x^* = \varphi(x^*)$ и является локально-сжимающим в некоторой ее окрестности $B^* = B(x^*, r)$. Тогда для всех $x_0 \in B^*$ итерационный процесс* (4.32) *сходится к x^*.*

$\triangleright_2$ Докажите лемму.

Для применения леммы 4.2 необходимо обладать достаточно точной информацией о местоположении корня x^*. Следующий признак сходимости опирается лишь на информацию о φ в окрестности начального приближения x^0.

Теорема 4.2. *Пусть отображение φ является локально-сжимающим на множестве $B = B(x, r)$ с константой $L < 1$. Если имеет место неравенство*

$$\rho(x, \varphi(x)) \leqslant (1 - L)r, \tag{4.36}$$

то x^* — искомый корень уравнения $\varphi(x) = x$ — лежит в B и итерационный процесс (4.32) *сходится к x^* для всех $x_0 \in B$.*

Доказательство. Если мы докажем, что $\varphi(y) \in B$ при любых $y \in B$, то по общему принципу сжимающих отображений из этого автоматически будет следовать утверждение теоремы. Итак, пусть $y \in B$, тогда

$$\begin{aligned}\rho(\varphi(y), x) &\leqslant \rho(\varphi(y), \varphi(x)) + \rho(x, \varphi(x)) \leqslant \\ &\leqslant L\rho(y, x) + \rho(x, \varphi(x)) \leqslant Lr + (1 - L)r = r.\end{aligned}$$ □

4.3.3. Применение принципа сжимающих отображений

Рассмотрим случай функции одной переменной (одного нелинейного уравнения).

Теорема 4.3 (Лагранжа о среднем значении)**.** *Если функция* $\varphi : [a, b] \to \mathbb{R}$ *непрерывна на* $[a, b]$ *и дифференцируема на* (a, b)*, то* $\exists\, \xi \in (a, b)$ *такое, что*

$$\varphi(b) - \varphi(a) = \varphi'(\xi)(b - a).$$

В случае $\varphi : \mathbb{R} \to \mathbb{R}$ существует простое достаточное условие липшицевости: если φ достаточно гладкая, то согласно теореме Лагранжа имеем

$$|\varphi(x) - \varphi(y)| = |\varphi'(\xi)| \cdot |x - y|, \quad \xi \in (x, y),$$

поэтому если

$$|\varphi'(x)| \leqslant M \quad \forall\, x \in [a, b],$$

то φ — липшицева на $[a, b]$ с константой $L = M$.

Таким образом, получаем **признак сходимости 0**: если

$$|\varphi'(x)| \leqslant M < 1 \quad \forall\, x \in \mathbb{R}, \tag{4.37}$$

то итерационный процесс (4.32) сходится.

Очевидно, что ограничение вида (4.37) является слишком сильным. Как правило, мы можем рассчитывать на выполнение условий такого рода лишь на некотором подмножестве $\mathbb{R}$. Лемма 4.2 дает **признак сходимости 1**: если

$$|\varphi'(x)| \leqslant M < 1 \quad \forall\, x \in B^* = [x^* - r, x^* + r], \tag{4.38}$$

то итерационный процесс (4.32) сходится к x^* при всех $x_0 \in B^*$.

Из теоремы 4.2 получаем также **признак сходимости 2**: если

$$|\varphi'(x)| \leqslant M < 1 \quad \forall\, x \in B = [\xi - r, \xi + r]$$

и при этом

$$|\xi - \varphi(\xi)| \leqslant (1 - M)r,$$

то итерационный процесс (4.32) сходится к $x^* = \varphi(x^*)$ при всех $x_0 \in B$.

Важно понимать, что все рассмотренные выше признаки являются лишь *достаточными* условиями сходимости.

Применение к методу Ньютона

Напомним, что метод Ньютона для решения уравнения $f(x) = 0$ представляет собой метод типа (4.32), где

$$\varphi(x) = x - \frac{f(x)}{f'(x)}$$

и

$$\varphi'(x) = \frac{f(x)f''(x)}{f'(x)^2}.$$

Пусть $x^* \in [a, b]$, производные $f'(x)$ и $f''(x)$ непрерывны и не обращаются в нуль при всех $x \in [a, b]$. Тогда $\varphi'(x)$ непрерывна и из тождества $\varphi'(x^*) = 0$ следует, что существует целая окрестность $B^* = B(x^*, r)$ такая, что

$$|\varphi'(x)| \leqslant M < 1 \quad \forall\, x \in B^*.$$

Следовательно, по признаку сходимости 1 метод Ньютона будет сходиться при всех x_0 из B^*.

Глава 5

ПРИБЛИЖЕНИЕ ФУНКЦИЙ

5.1. Интерполяция функций. Интерполяционный многочлен

5.1.1. Общая постановка задачи интерполяции

Определение 5.1. Рассмотрим набор различных точек $\{x_i\}_{i=0}^n$, $x_i \in [a,b]$. Пусть $\{y_i\}_{i=0}^n$ — значения некоторой функции $f : [a,b] \to \mathbb{R}$ в этих точках: $y_i = f(x_i)$. Рассмотрим также набор линейно независимых базисных функций $\varphi_i : [a,b] \to \mathbb{R}$, $i = \overline{0,n}$. *Задача интерполяции* заключается в нахождении функции

$$\varphi = \sum_{i=0}^{n} \alpha_i \varphi_i, \quad \alpha_i \in \mathbb{R},$$

такой, что

$$\varphi(x_i) = y_i \quad \forall\, i = \overline{0,n}. \tag{5.1}$$

Функция f называется *интерполируемой функцией*, φ — *интерполирующей функцией*, $\{x_i\}$ — *узлами интерполяции*, точки декартовой плоскости (x_i, y_i) — *точками интерполяции*.

Замечание 5.1. Строго говоря, мы дали здесь определение *линейной* задачи интерполяции. В общем случае интерполирующая функция φ может зависеть от коэффициентов α_i нелинейно.

Таким образом, задача интерполяции сводится к нахождению неизвестных коэффициентов $\{\alpha_i\}_{i=0}^n$ из условий (5.1). По определению φ это

эквивалентно решению СЛАУ

$$\sum_{j=0}^{n} \alpha_j \varphi_j(x_i) = y_i, \quad i = \overline{0,n},$$

матричный вид которой запишем как

$$\Phi\alpha = y, \tag{5.2}$$

где $\alpha = (\alpha_0, \ldots, \alpha_n)^T$; $y = (y_0, \ldots, y_n)^T$;

$$\Phi = \begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \ldots & \varphi_n(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \ldots & \varphi_n(x_1) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \varphi_0(x_n) & \varphi_1(x_n) & \ldots & \varphi_n(x_n) \end{bmatrix}. \tag{5.3}$$

Очевидно, что решение задачи интерполяции существует и единственно тогда и только тогда, когда $\det \Phi \neq 0$. Система (5.2) имеет наиболее простой вид в случае, когда $\Phi = I$. Отсюда вытекает следующее определение.

Определение 5.2. Пусть $\{x_i\}_{i=0}^n$ — узлы интерполяции. Система функций $\{\varphi_i\}_{i=0}^n$ называется *фундаментальным базисом* для данного набора узлов, если

$$\varphi_i(x_j) = \delta_{ij} = \begin{cases} 1, & i = j; \\ 0, & i \neq j; \end{cases} \qquad \forall i, j = \overline{0,n}.$$

Таким образом, если $\{\varphi_i\}$ — фундаментальный базис, то задача интерполяции решается очень просто:

$$\varphi = \sum_{i=0}^{n} y_i \varphi_i.$$

5.1.2. Полиномиальная интерполяция

Классическим способом аппроксимации (приближения) функций является интерполяция алгебраическими многочленами:

$$\varphi(x) = P_n(x) = \sum_{i=0}^{n} \alpha_i x^i, \quad \text{т. е.} \quad \varphi_i(x) = x^i. \tag{5.4}$$

В этом случае интерполирующую функцию φ называют *интерполяционным многочленом*.

Пусть заданы узлы интерполяции. Рассмотрим для базиса из (5.4) матрицу интерполяции (5.3):

$$\Phi = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} = V(x_0, \dots, x_n). \tag{5.5}$$

Матрица такого вида называется *матрицей Вандермонда*.

Лемма 5.1. *Определитель матрицы Вандермонда может быть вычислен по формуле*

$$|V(x_0, \dots, x_n)| = \prod_{0 \leqslant j < i \leqslant n} (x_i - x_j). \tag{5.6}$$

Доказательство. Рассмотрим

$$|V(x_0, \dots, x_n)| = \begin{vmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{vmatrix}.$$

Преобразуем этот определитель: вычтем из j-го столбца $(j-1)$-й, умноженный на x_0, для j от $n+1$ до 2. В результате получим

$$|V(x_0, \dots, x_n)| = \begin{vmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & x_1 - x_0 & x_1^2 - x_1 x_0 & \dots & x_1^n - x_1^{n-1} x^0 \\ 1 & x_2 - x_0 & x_2^2 - x_2 x_0 & \dots & x_2^n - x_2^{n-1} x^0 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n - x_0 & x_n^2 - x_n x_0 & \dots & x_n^n - x_n^{n-1} x^0 \end{vmatrix} =$$

$$= (x_1 - x_0)(x_2 - x_0)\dots(x_n - x_0) \underbrace{\begin{vmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} \end{vmatrix}}_{|V(x_1,\,\dots,\,x_n)|}.$$

Преобразуя аналогичным образом по рекурсии $|V(x_1,\,\dots,\,x_n)|$, $|V(x_2,\,\dots,\,x_n)|$ и т. д., окончательно получим тождество (5.6). □

Следствие 5.1. *Если все узлы интерполяции $\{x_i\}$ различны, то при любых значениях $\{y_i\}$ интерполяционный многочлен существует и единствен.*

Следствие 5.2. *Любой многочлен степени n однозначно определяется своими значениями в $n+1$ различных точках.*

$\triangleright_1$ Докажите данные следствия.

5.1.3. Интерполяционный многочлен в форме Лагранжа

Построим фундаментальный базис многочленов для узлов $x_0, \dots, x_n$. Пусть Λ_i — i-й многочлен из этого базиса. По определению

$$\Lambda_i(x_j) = 0 \quad \forall j \neq i,$$

т. е.

$$\Lambda_i(x) = C_i \prod_{j \neq i} (x - x_j), \quad C_i \in \mathbb{R}.$$

Здесь и далее используется обозначение

$$\prod_{j \neq i} (\dots) = \prod_{\substack{j=0 \\ j \neq i}}^{n} (\dots).$$

Неизвестную константу C_i найдем из оставшегося условия

$$\Lambda_i(x_i) = 1 \quad \Rightarrow \quad C_i = \left(\prod_{j \neq i} (x_i - x_j) \right)^{-1},$$

откуда

$$\Lambda_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}. \tag{5.7}$$

Таким образом, получим знаменитую формулу *интерполяционного многочлена в форме Лагранжа*:

$$P_n(x) = \sum_{i=0}^{n} y_i \Lambda_i(x) = \sum_{i=0}^{n} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}. \tag{5.8}$$

Для многочленов Λ_i существует альтернативная форма записи. Рассмотрим многочлен

$$\omega_{n+1}(x) = (x - x_0)(x - x_1) \dots (x - x_n). \tag{5.9}$$

Тогда числитель в (5.7) можно записать как

$$\frac{\omega_{n+1}(x)}{x - x_i},$$

а знаменатель как

$$\omega'_{n+1}(x_i).$$

Отсюда

$$\Lambda_i(x) = \frac{\omega_{n+1}(x)}{(x - x_i)\omega'_{n+1}(x_i)}. \tag{5.10}$$

5.1.4. Барицентрическая интерполяционная формула

Интерполяционная формула Лагранжа (5.8) становится громоздкой и неудобной для вычислений при больших n. Поэтому долгое время считалось, что эта формула малопригодна для практических вычислений. Но оказывается, что при правильном способе записи ситуация кардинально изменяется.

Запишем многочлен Лагранжа с использованием формулы (5.10):

$$P_n(x) = \sum_{i=0}^{n} y_i \frac{\omega_{n+1}(x)}{(x - x_i)\omega'_{n+1}(x_i)}. \tag{5.11}$$

Чтобы избежать громоздких умножений в каждом слагаемом формулы (5.11), достаточно просто вынести общий множитель $\omega_{n+1}(x)$:

$$P_n(x) = \omega_{n+1}(x) \sum_{i=0}^{n} \frac{y_i}{(x - x_i)\omega'_{n+1}(x_i)}.$$

Эту формулу удобно записать в виде

$$P_n(x) = \omega_{n+1}(x) \sum_{i=0}^{n} y_i \frac{v_i}{x - x_i}, \tag{5.12}$$

где

$$v_i = \frac{1}{\omega'_{n+1}(x_i)} = \frac{1}{\prod_{j \neq i}(x_i - x_j)}, \quad i = \overline{0, n}. \tag{5.13}$$

Коэффициенты v_i будем называть *весовыми*. Полученная форма записи интерполяционного многочлена уже намного удобнее традиционной формулы Лагранжа, но ее можно еще улучшить. Для этого построим по формуле (5.12) интерполяционный многочлен для $f(x) = 1$. Этот многочлен, как нетрудно заметить, тождественно равен единице при любом n:

$$\omega_{n+1}(x) \sum_{i=0}^{n} \frac{v_i}{x - x_i} = 1.$$

Выражая отсюда $\omega_{n+1}(x)$ и подставляя в (5.12), получим замечательную формулу

$$P_n(x) = \frac{\displaystyle\sum_{i=0}^{n} y_i \frac{v_i}{x - x_i}}{\displaystyle\sum_{i=0}^{n} \frac{v_i}{x - x_i}}, \tag{5.14}$$

которая называется *барицентрической интерполяционной формулой*. Обращаем внимание, что несмотря на то, что правая часть (5.14) выглядит как рациональная функция, по построению это не что иное, как интерполяционный многочлен.

Легко убедиться, что барицентрическая интерполяционная формула обладает следующими свойствами:

- для ее построения, т. е. для вычисления коэффициентов $\{v_i\}_{i=0}^{n}$, требуется $O(n^2)$ операций. Однако в п. 5.3 мы увидим, что при специальном выборе узлов интерполяции количество операций составляет всего $O(n)$;

• для вычисления $P_n(x)$ (при известных весовых коэффициентах) по этой формуле требуется всего $O(n)$ операций;

• формула может быть легко обновлена при добавлении дополнительного узла интерполяции x_{n+1}: каждое v_i необходимо домножить на $(x_i - x_{n+1})^{-1}$, после чего по определению вычислить v_{n+1};

• весовые коэффициенты не зависят от $\{y_i\}$, т. е. однажды построенная формула для заданного набора узлов может легко применяться для интерполяции различных функций f;

• весовые коэффициенты v_i можно спокойно домножать на любую константу. Это большой плюс с точки зрения вычислительной устойчивости.

Помимо указанных свойств, барицентрическая формула обладает выдающейся вычислительной устойчивостью (при условии отдельной обработки случая, когда знаменатель в формуле (5.14) обращается в нуль).

5.2. Интерполяционный многочлен в форме Ньютона

5.2.1. Построение

Рассмотрим еще раз СЛАУ (5.2), решение которой является решением задачи интерполяции в общем случае:

$$\Phi\alpha = \begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \dots & \varphi_n(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \dots & \varphi_n(x_1) \\ \dots & \dots & \dots & \dots \\ \varphi_0(x_n) & \varphi_1(x_n) & \dots & \varphi_n(x_n) \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

При построении формулы Лагранжа мы выбирали базис $\{\varphi_i\}$ так, чтобы матрица Φ была единичной. Теперь же построим базис, в котором эта матрица будет *нижнетреугольной*. Такой базис должен удовлетворять очевидным условиям

$$\varphi_i(x_j) = 0 \quad \forall j = \overline{0, i-1},$$

откуда с точностью до постоянного множителя получим $\varphi_i(x) = \omega_i(x)$, где

$$\omega_i(x) = (x - x_0)(x - x_1)\dots(x - x_{i-1}) = \prod_{j=0}^{i-1}(x - x_j) \tag{5.15}$$

(сравните с формулой (5.9)). Здесь при $i = 0$ полагаем $\omega_0(x) = 1$.

Определение 5.3. Интерполяционный многочлен вида

$$P_n(x) = \sum_{i=0}^{n} \alpha_i \omega_i(x) = \alpha_0 + \alpha_1(x - x_0) + \alpha_2(x - x_0)(x - x_1) + \dots + \\ + \alpha_n(x - x_0)(x - x_1)\dots(x - x_{n-1}) \tag{5.16}$$

называется *интерполяционным многочленом в форме Ньютона*. Правило вычисления коэффициентов α_i будет рассмотрено ниже.

Следует понимать, что формулы (5.16), (5.14) и (5.8) суть разные формы записи одного и того же многочлена.

По определению интерполяции коэффициенты интерполяционного многочлена в форме Ньютона удовлетворяют СЛАУ

$$\begin{bmatrix} \omega_0(x_0) & 0 & 0 & \dots & 0 \\ \omega_0(x_1) & \omega_1(x_1) & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \omega_0(x_n) & \omega_1(x_n) & \omega_2(x_n) & \dots & \omega_n(x_n) \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}. \tag{5.17}$$

▷1 При каких условиях СЛАУ с такой матрицей имеет единственное решение?

Следовательно, α_i можно найти, напрямую решив (5.17) путем обратной подстановки, которая приводит к следующим рекуррентным соотношениям:

$$\alpha_i = \Big(y_i - \sum_{j=0}^{i-1} \alpha_j \omega_j(x_i)\Big) \Big/ \omega_i(x_i), \quad i = \overline{0, n}. \tag{5.18}$$

Однако использование формул (5.18) не является общепринятым способом вычисления α_i, так как существует более эффективный способ, основанный на понятии разделенных разностей (см. далее).

По построению интерполяционный многочлен в форме Ньютона обладает рядом полезных свойств. Во-первых, базисные функции ω_i зависят только от узлов $x_0, \dots, x_{i-1}$, а коэффициенты α_i — только от $x_0, \dots, x_i$.

Это позволяет легко «обновлять» формулу многочлена при добавлении дополнительных узлов интерполяции. Другими словами, если известен многочлен P_n (5.16), интерполирующий f по узлам $x_0, \ldots, x_n$, то интерполяционный многочлен по узлам $x_0, \ldots, x_{n+1}$ может быть найден по формуле

$$P_{n+1}(x) = P_n(x) + \alpha_{n+1}\omega_{n+1}(x). \tag{5.19}$$

Во-вторых, форма (5.16) более удобна для вычислений, чем классическая форма Лагранжа: последовательно вынося за скобки общие множители $(x - x_i)$, можно вычислить $P_n(x)$ по так называемой схеме Горнера

$$P_n(x) = \alpha_0 + (x - x_0)\Big(\alpha_1 + (x - x_1)\big(\alpha_2 + \ldots\big)\Big). \tag{5.20}$$

$\triangleright_2$ Подсчитайте количество арифметических операций, нужных для вычисления $P_n(x)$ по формуле (5.20).

5.2.2. Разделенные разности

Пусть $\{y_i\}_{i=0}^n$ являются значениями некоторой функции f в точках $\{x_i\}_{i=0}^n$ соответственно. Тогда каждый из коэффициентов α_i, определяемых по формулам (5.18), можно рассматривать как выражение, зависящее от функции f и узлов $x_0, \ldots, x_i$. Обозначим (пока формально) это выражение как

$$\alpha_i = f[x_0, \ldots, x_i]. \tag{5.21}$$

Лемма 5.2. *Пусть $\{x_0, x_1, \ldots\}$ — последовательность различных узлов интерполяции, P_k — интерполяционный многочлен для функции f по узлам $\{x_0, \ldots, x_k\}$, и P_k^+ — аналогичный многочлен, но построенный по узлам $\{x_1, \ldots, x_{k+1}\}$. Тогда*

$$P_{k+1}(x) = \frac{(x - x_0)P_k^+(x) - (x - x_{k+1})P_k(x)}{x_{k+1} - x_0}, \quad \forall\, k \geqslant 0. \tag{5.22}$$

$\triangleright_3$ Докажите лемму простой проверкой условий $P_{k+1}(x_j) = f(x_j)$, $j = \overline{0, k+1}$.

Для $k = i - 1$ запишем интерполяционные многочлены P_k и P_k^+ из

леммы 5.2 в форме Ньютона, используя обозначение (5.21):

$$P_{i-1}(x) = \sum_{j=0}^{i-1} \alpha_j \omega_j(x) = \sum_{j=0}^{i-1} f[x_0, \ldots, x_j]\omega_j(x),$$
$$P^+_{i-1}(x) = \sum_{j=0}^{i-1} \alpha^+_j \omega^+_j(x) = \sum_{j=0}^{i-1} f[x_1, \ldots, x_{j+1}]\omega^+_j(x).$$

Здесь $\omega^+_j(x) = (x - x_1)(x - x_2)\ldots(x - x_j)$. Тогда (5.22) примет вид

$$P_i(x) = \sum_{j=0}^{i} f[x_0, \ldots, x_j]\omega_j(x) =$$
$$= \frac{(x - x_0)\sum_{j=0}^{i-1} f[x_1, \ldots, x_{j+1}]\omega^+_j(x) - (x - x_i)\sum_{j=0}^{i-1} f[x_0, \ldots, x_j]\omega_j(x)}{x_i - x_0}.$$

Приравнивая коэффициенты при старшей степени в обеих частях, получим важное тождество

$$f[x_0, \ldots, x_i] = \frac{f[x_1, \ldots, x_i] - f[x_0, \ldots, x_{i-1}]}{x_i - x_0}.$$

Отсюда вытекает следующее определение.

Определение 5.4. Разделенной разностью порядка i для функции f по различным узлам $\{x_j\}_{j=0}^i$ называется выражение $f[x_0, \ldots, x_i]$, определяемое по рекуррентным соотношениям

$$f[x_0, \ldots, x_i] = \frac{f[x_1, \ldots, x_i] - f[x_0, \ldots, x_{i-1}]}{x_i - x_0}, \quad \forall i \geqslant 1; \qquad (5.23а)$$
$$f[x_j] = f(x_j) \quad \forall\, j. \qquad (5.23б)$$

Таким образом, коэффициенты (5.21) *интерполяционного многочлена в форме Ньютона являются разделенными разностями и традиционно вычисляются по определению* (5.23).

$\triangleright_4$ Вычислите общий вид коэффициентов α_0, α_1 и α_2 сначала по формуле (5.18), затем по формуле (5.23).

$\triangleright_5$ Сравните вычислительную сложность вычисления α_i по формулам (5.18) и (5.23).

$\triangleright_6$ Докажите, что значение разделенной разности не зависит от порядка расположения ее аргументов.

5.2.3. Алгоритм вычисления разделенных разностей

Коэффициенты интерполяционного многочлена (5.16) удобно вычислять по определению разделенных разностей (5.23) путем построения треугольной таблицы следующего вида:

$$
\begin{array}{llllll}
x_0 & f[x_0] & & & & \\
 & & f[x_0,x_1] & & & \\
x_1 & f[x_1] & & f[x_0,x_1,x_2] & & \\
 & & f[x_1,x_2] & & \dots & \\
x_2 & f[x_2] & & f[x_1,x_2,x_3] & \dots\ f[x_0,\ \dots,x_{n-1}] & \\
 & \cdot & f[x_2,x_3] & \cdot & \dots & f[x_0,\ \dots,x_n] \\
 & \cdot & \cdot & \cdot & \dots\ \ f[x_1,\ \dots,x_n] & \\
 & \cdot & \cdot & \cdot & \dots & \\
x_{n-1} & f[x_{n-1}] & \cdot & f[x_{n-2},x_{n-1},x_n] & & \\
 & & f[x_{n-1},x_n] & & & \\
x_n & f[x_n] & & & &
\end{array}
$$

Вычисления осуществляются следующим образом: сначала записываются первые два столбца таблицы, затем по определению вычисляется третий (разделенные разности первого порядка), затем четвертый и т. д. При этом каждый новый элемент таблицы вычисляется с использованием двух ближайших к нему элементов предыдущего столбца и соответствующих им значений x_i.

$\triangleright_7$ Каким минимальным количеством ячеек памяти можно обойтись при вычислении разделенных разностей, необходимых для построения интерполяционного многочлена в форме Ньютона? Постройте соответствующий алгоритм.

$\triangleright_8$ Сравните трудоемкость построения и вычисления интерполяционного многочлена в форме Ньютона с барицентрической интерполяционной формулой (5.14).

5.3. Остаток интерполяции. Многочлены Чебышева

5.3.1. Остаток алгебраического интерполирования

Определение 5.5. C-нормой для непрерывной на отрезке $[a,b]$ функ-

ции f называется величина $\|f\|_C$, вычисляемая по формуле

$$\|f\|_C = \|f\|_{C[a,b]} = \max_{a \leqslant x \leqslant b} |f(x)|.$$

Определение 5.6. Пусть $f \in C[a,b]$ — интерполируемая функция, P_n — интерполяционный многочлен для f по узлам $\{x_i \in [a,b]\}_{i=0}^n$. *Остатком интерполирования* называют функцию

$$r_n = f - P_n. \tag{5.24}$$

Погрешностью интерполирования назовем C-норму остатка:

$$\varepsilon_n = \|r_n\|_C = \max_{x \in [a,b]} |r_n(x)|. \tag{5.25}$$

Теорема 5.1. *Остаток интерполирования имеет вид*

$$r_n(x) = f[x_0, \ldots, x_n, x]\, \omega_{n+1}(x) \quad \forall x \in \mathbb{R}. \tag{5.26}$$

Доказательство. Пусть P_n — интерполяционный многочлен по узлам $x_0, \ldots, x_n$ для f. Рассмотрим произвольную точку $\xi \neq x_i$, $i = \overline{0,n}$. Построим для f интерполяционный многочлен P_{n+1} по узлам $\{x_0, \ldots, x_n, \xi\}$. По формуле (5.19) имеем

$$P_{n+1}(\xi) = f(\xi) = P_n(\xi) + f[x_0, \ldots, x_n, \xi]\omega_{n+1}(\xi).$$

Меняя в этой формуле обозначение ξ на x, получим (5.26). □

Замечание 5.2. Если точка x совпадает с одним из узлов интерполяции x_i, то в формуле (5.26) возникнет разделенная разность с двумя одинаковыми узлами, для вычисления которой нужно обобщить определение 5.4. Такое обобщение можно найти в большинстве пособий по численному анализу, например в [2]. Для строгого же доказательства теоремы 5.1 нам достаточно принять на веру, что такая разделенная разность будет ограниченной величиной: тогда (5.26) в соответствии с реальностью даст $r_n(x) = 0$.

Чтобы получить еще одно полезное выражение для остатка, необходимо вспомнить следующую теорему из курса анализа.

Теорема 5.2 (Ролля). *Если функция g непрерывна на $[a,b]$, дифференцируема на (a,b), и $g(a) = g(b)$, то существует по крайней мере одна такая точка $\xi \in (a,b)$, что $g'(\xi) = 0$.*

С помощью этой теоремы можно установить связь между разделенными разностями и производными функции f.

Лемма 5.3. *Пусть* $\underline{x} = \min\{x_i\}_{i=0}^n$, $\overline{x} = \max\{x_i\}_{i=0}^n$ *и* $f \in C^n[\underline{x}, \overline{x}]$. *Тогда* $\exists\, \xi \in (\underline{x}, \overline{x})$ *такое, что*

$$f[x_0, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!}. \tag{5.27}$$

Доказательство. Построим для f интерполяционный многочлен P_n в форме Ньютона по узлам $x_0, \ldots, x_n$ и рассмотрим остаток $r_n = f - P_n$. По построению имеем

$$r_n(x_0) = r_n(x_1) = \ldots = r_n(x_n) = 0,$$

значит, по теореме Ролля между $\underline{x}$ и $\overline{x}$ существует как минимум n точек, в которых r_n' обращается в нуль.

Продолжая аналогичные рассуждения для r_n'' и т. д., получим, что существует $\xi \in (\underline{x}, \overline{x})$, такое, что

$$r_n^{(n)}(\xi) = f^{(n)}(\xi) - P_n^{(n)}(\xi) = f^{(n)}(\xi) - n!\, f[x_0, \ldots, x_n] = 0. \qquad \square$$

Теорема 5.3. *Пусть* $f \in C^{(n+1)}[a, b]$, P_n *— интерполяционный многочлен для* f *по узлам* $\{x_i\}_{i=0}^n \subset [a, b]$. *Тогда* $\forall x \in [a, b]\ \exists\, \xi \in (\underline{x}, \overline{x})$, $(\underline{x} = \min\{x_0, \ldots, x_n, x\}, \overline{x} = \max\{x_0, \ldots, x_n, x\})$, *такое, что*

$$r_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x). \tag{5.28}$$

Доказательство. Данная теорема представляет собой тривиальное следствие теоремы 5.1 и леммы 5.3. $\square$

Формула (5.28) является основным инструментом при оценке погрешности интерполяции. Из нее, в частности, сразу получим

$$\varepsilon_n = \|r_n\| \leqslant \frac{\|f^{(n+1)}\|}{(n+1)!}\|\omega_{n+1}\|, \tag{5.29}$$

где $\|\cdot\| = \|\cdot\|_{C[a,b]}$ (см. (5.25)).

5.3.2. Задача выбора оптимальных узлов интерполирования

Пусть P_n — интерполяционный многочлен для f по узлам $\{x_i\}_{i=0}^n$ из отрезка $[a, b]$. Согласно теореме 5.3, точность приближения оценивается по формуле

$$\|f - P_n\| \leqslant \frac{\|f^{(n+1)}\|}{(n+1)!}\|\omega_{n+1}\|, \tag{5.30}$$

где

$$\omega_{n+1}(x) = (x - x_0)(x - x_1)(x - x_n).$$

Здесь и далее в этом подпункте $\|\cdot\| = \|\cdot\|_C$.

Часто требуется приблизить заранее неизвестную функцию интерполяционным многочленом как можно точнее, т. е. сделать правую часть (5.30) как можно меньше. И если в общем случае о величине $\|f^{(n+1)}\|$ ничего не известно, то $\|\omega_{n+1}\|$ можно минимизировать за счет выбора узлов $\{x_i\}$. Задачу выбора оптимальных узлов интерполяции можно поставить так: найти многочлен ω_{n+1} со *старшим коэффициентом, равным единице*, и минимально возможной нормой $\|\omega_{n+1}\|$ на отрезке интерполяции $[a, b]$. Такой многочлен называется *многочленом, наименее отклоняющимся от нуля* на $[a, b]$. Корни такого многочлена и будут оптимальными[4] узлами интерполяции.

Оказывается, что решением поставленной задачи для отрезка $[-1, 1]$ являются многочлены Чебышева — пожалуй, самое известное и часто используемое в вычислениях семейство многочленов.

5.3.3. Многочлены Чебышева

Рассмотрим на отрезке $[a, b] = [-1, 1]$ семейство функций $\{T_n\}_{n=0}^{\infty}$, определяемое формулой

$$T_n(x) = \cos(n \arccos x), \quad n = 0, 1, 2, \ldots . \tag{5.31}$$

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

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

[4] Необходимо понимать, что «оптимальные» в данном случае означает «универсальные», не зависящие от вида интерполируемой функции. Для каждой конкретной функции f узлы интерполяции, дающие минимальную погрешность, вообще говоря, заранее не известны.

а для произвольного n справедливо рекуррентное соотношение

$$T_{n+1}(x) = 2x\,T_n(x) - T_{n-1}(x). \tag{5.32}$$

$\triangleright_1$ Вычислите $T_i(x)$ для $i = 2, 3, 4, 5$.

$\triangleright_2$ Докажите рекуррентное соотношение (5.32), используя тригонометрическую формулу $\cos\alpha\cos\beta = \frac{1}{2}(\cos(\alpha+\beta) + \cos(\alpha-\beta))$.

Нетрудно видеть из определения, что C-норма всех T_n равна единице. Основное свойство многочленов Чебышева состоит в том, что их графики «равномерно колеблются» между -1 и 1 (рис. 5.1). Говоря точнее, многочлен T_n имеет $n+1$ локальных экстремумов, которые поочередно равны ± 1.

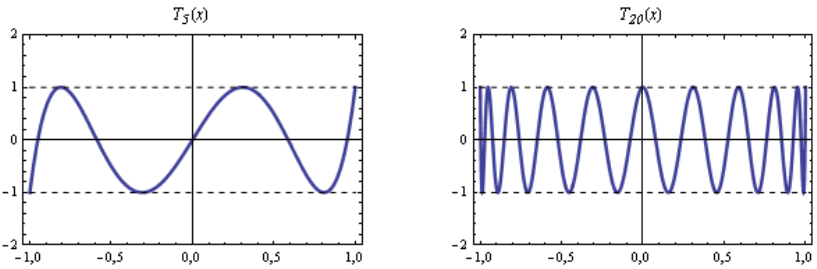


Рис. 5.1. Многочлены Чебышева

Следствием этого свойства является то, что *если многочлен T_n разделить на его старший коэффициент, то получится искомый нами многочлен, наименее отклоняющийся от нуля на* $[-1, 1]$. Доказательство этого факта выходит за рамки нашего курса.

5.3.4. Оптимальные узлы интерполирования

Итак, нули многочлена Чебышева (5.31) являются оптимальными узлами интерполирования на отрезке $[-1, 1]$, т. е. оптимальный многочлен ω_{n+1} определяется формулой

$$\omega_{n+1}(x) = 2^{-n} T_{n+1}(x). \tag{5.33}$$

Множитель 2^{-n} — это обратное значение старшего коэффициента многочлена T_{n+1}, в чем нетрудно убедиться из рекуррентного соотношения (5.32). Нули этого многочлена легко находятся из уравнения

$$T_{n+1}(x) = \cos((n+1)\arccos x) = 0$$

и равны

$$x_i = \cos\frac{\pi(2i+1)}{2n+2}, \quad i = \overline{0,n}. \tag{5.34}$$

$\triangleright_3$ Как надо изменить формулу (5.34), чтобы узлы интерполяции расположились в порядке возрастания?

Это и есть оптимальные узлы интерполирования, называемые также *чебышевскими*. При таком выборе узлов из (5.33) получим

$$\|\omega_{n+1}\| = 2^{-n},$$

т. е. оценка погрешности (5.35) для интерполяционного многочлена по чебышевским узлам на отрезке $[-1,1]$ имеет вид

$$\|r_n\| \leqslant \frac{\|f^{(n+1)}\|}{2^n(n+1)!}. \tag{5.35}$$

Весовые коэффициенты барицентрической формулы

Использование чебышевских узлов интерполяции, помимо прочего, существенно повышает эффективность барицентрической интерполяционной формулы (5.14). Напомним, что для ее использования необходимо вычислять весовые коэффициенты по формуле

$$v_i = \frac{1}{\prod_{j\neq i}(x_i - x_j)}, \quad i = \overline{0,n}.$$

Если в этой формуле использовать чебышевские узлы (5.34), то с учетом (5.33), после сокращения множителей, не зависящих от i, она принимает простой вид

$$v_i = (-1)^i \sin\frac{2i+1}{2n+2}\pi. \tag{5.36}$$

$\triangleright_4$ Выведите эту формулу

Еще более простой вид имеют коэффициенты для так называемых *чебышевских узлов второго рода*

$$x_i = \cos\frac{i\pi}{n}, \quad i = \overline{0,n}. \tag{5.37}$$

Соответствующие весовые коэффициенты для барицентрической формулы определяются формулой

$$v_i = (-1)^i \delta_i, \quad \delta_i = \begin{cases} 1/2, & \text{если } i = 0 \text{ или } i = n, \\ 1, & \text{в противном случае.} \end{cases} \tag{5.38}$$

Как видим, для узлов такого типа барицентрическая интерполяционная формула практически не требует никаких предварительных вычислений.

Чебышевские узлы на отрезке $[a, b]$

Чтобы применить полученные результаты для интерполяции на произвольном отрезке $[a, b]$, нужно воспользоваться заменой переменной:

$$x = \frac{a+b}{2} + \frac{b-a}{2} t, \quad t \in [-1, 1],\ x \in [a, b].$$

Тогда многочлен Чебышева, смасштабированный на $[a, b]$, имеет вид

$$\begin{aligned} \widehat{T}_{n+1}(x) = T_{n+1}(t) = 2t\, T_n(t) - T_{n-1}(t) = 2t\, \widehat{T}_n(x) - \widehat{T}_{n-1}(x) = \\ = 2\frac{2x-a-b}{b-a} \widehat{T}_n(x) - \widehat{T}_{n-1}(x). \end{aligned} \tag{5.39}$$

Кроме этого, имеют место очевидные соотношения

$$\widehat{T}_0(x) = 1, \quad \widehat{T}_1(x) = \frac{2x-b-a}{b-a}.$$

Следовательно, согласно (5.39), старший коэффициент многочлена $\widehat{T}_n(x)$ при всех $n \geqslant 1$ равен

$$\hat{a}_n = \frac{2}{b-a} \left(\frac{4}{b-a} \right)^{n-1} = \frac{1}{2} \left(\frac{4}{b-a} \right)^n,$$

откуда получим

$$\omega_{n+1}(x) = 2 \left(\frac{b-a}{4} \right)^{n+1} T_{n+1}\left(\frac{2x-b-a}{b-a} \right). \tag{5.40}$$

Корни этого многочлена (*оптимальные узлы интерполяции на отрезке* $[a, b]$), получаются масштабированием узлов (5.34):

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos \frac{\pi(2i+1)}{2n+2}, \quad i = \overline{0, n}. \tag{5.41}$$

Кроме этого, из (5.40) следует, что при выборе чебышевских узлов интерполирования (5.41) имеем

$$\|\omega_{n+1}\| = 2\left(\frac{b-a}{4}\right)^{n+1}.$$

Это равенство используется при оценке погрешности интерполирования по формуле (5.30):

$$\|r_n\| \leqslant 2\left(\frac{b-a}{4}\right)^{n+1}\frac{\|f^{(n+1)}\|}{(n+1)!}. \tag{5.42}$$

5.3.5. О сходимости интерполяционного процесса

Центральный вопрос теории приближения функций — сходимость интерполяционного процесса. Говоря простым языком, необходимо определить, будет ли интерполяционный многочлен стремиться к интерполируемой функции при неограниченном увеличении количества узлов интерполяции. На языке формул это требование записывается как

$$\|f - P_n\|_C \xrightarrow[n\to\infty]{} 0.$$

Здесь P_n — интерполяционный многочлен, построенный по набору узлов

$$X_n = \{x_0^{(n)}, x_1^{(n)}, \ldots, x_n^{(n)}\}.$$

Таким образом, последовательность $\{P_n\}$, т. е. сам интерполяционный процесс, определяется таблицей узлов интерполяции

$$\begin{aligned}
&X_0 = \{x_0^{(0)}\},\\
&X_1 = \{x_0^{(1)}, x_1^{(1)}\},\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&X_n = \{x_0^{(n)}, x_1^{(n)}, \ldots, x_n^{(n)}\},\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{aligned} \tag{5.43}$$

Основные факты о сходимости полиномиальной интерполяции

Теорема 5.4. *Для любой $f \in C[a,b]$ существует такая последовательность сеток, для которой интерполяционный процесс равномерно сходится к f.*

Эта теорема носит формальный характер, так как построение нужной последовательности сеток в общем случае представляет серьезную проблему. Следующий результат говорит о том, что не существует «универсальной» последовательности сеток, хорошей для всех непрерывных функций.

Теорема 5.5. *Для любой последовательности сеток вида* (5.43) *существует такая* $f \in C[a,b]$*, для которой интерполяционный процесс не сходится равномерно к* f*.*

Таким образом, класс $C[a,b]$ слишком широк. Поэтому рассмотрим результаты о сходимости интерполяционного процесса для более узкого класса функций.

Определение 5.7. Функция f называется целой, если существует ее разложение в степенной ряд вида

$$f(x) = \sum_{i=0}^{\infty} c_i (x - x_0)^i,$$

которое сходится при любом x.

Типичные примеры целых функций: многочлены, exp, sin, cos и их линейные комбинации.

Теорема 5.6. *Если функция* f *целая, то интерполяционный процесс по любой последовательности сеток вида* (5.43) *равномерно сходится на* $[a,b]$ *к* f*.*

Тем не менее класс целых функций достаточно узок. Например, интерполяция по равноотстоящим узлам расходится для известной функции Рунге $f(x) = (1 + 25x^2)^{-1}$ и функции $g(x) = |x|$, обе на отрезке $[-1,1]$. Для последней интерполяционный многочлен по равноотстоящим узлам степени $2n$ неограниченно растет в любой части отрезка $[-1,1]$.

Кроме этого, интерполяция по равноотстоящим узлам обладает еще одним неприятным свойством: она плохо обусловлена, т. е. при достаточно большом количестве точек небольшие погрешности округления будут приводить к большим погрешностям. Значит, даже если для функции cos интерполяция по равноотстоящим узлам сходится *теоретически* по теореме 5.6, на практике (в машинной арифметике) с ростом числа узлов интерполяционный процесс будет расходиться.

Более привлекательна интерполяция по чебышевским узлам.

Теорема 5.7. *Для любой абсолютно непрерывной на* $[a,b]$ *функции* f *интерполяционный процесс по чебышевским узлам равномерно сходится.*

Точное определение абсолютно непрерывной функции нам не нужно, достаточно знать, что любая функция с ограниченной производной на отрезке абсолютно непрерывна на нем. Помимо этого, интерполяция по чебышевским узлам хорошо обусловлена.

5.4. Сплайны

5.4.1. Определение

Рассмотрим набор узлов

$$\{x_i\}_{i=0}^{n}, \quad a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b,$$

и соответствующее ему разбиение отрезка $[a, b]$ на n частей:

$$\Delta = \{\Delta_i\}_{i=1}^{n}, \quad \Delta_i = [x_{i-1}, x_i]. \tag{5.44}$$

Пример такого разбиения для случая $n = 4$ изображен на рис. 5.2. На каждом отрезке Δ_i зададим многочлен s_i фиксированной степени $m \geqslant \geqslant 1$ и рассмотрим полученный набор многочленов как единую кусочно-непрерывную функцию s, определенную на всем отрезке $[a, b]$:

$$s(x) = s_i(x), \quad \text{если} \quad x \in \Delta_i.$$

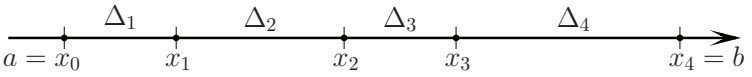


Рис. 5.2. Пример разбиения отрезка $[a, b]$

Для того чтобы это определение было однозначным при $x = x_i$, будем считать, что $s_i(x_i) = s_{i+1}(x_i)$ для всех $i = \overline{1, n-1}$, т. е. что функция s непрерывна. Такая кусочная функция и называется полиномиальным сплайном. Сформулируем это определение более строго.

Определение 5.8. Пусть Δ — некоторое разбиение отрезка $[a, b]$, определяемое формулой (5.44), $\mathbb{P}_m$ — множество всех многочленов степени не выше m. Для некоторого $m \geqslant 1$ рассмотрим функцию s, определенную на $[a, b]$ и обладающую следующими свойствами:

1) $s(x)|_{x\in\Delta_i} = s_i(x), \quad s_i \in \mathbb{P}_m$;
2) функция s имеет $m-1$ непрерывных производных:

$$s \in C^{m-1}[a,b],$$

что эквивалентно условиям

$$s_i^{(j)}(x_i - 0) = s_{i+1}^{(j)}(x_i + 0),$$
$$i = \overline{1, n-1}, \quad j = \overline{0, m-1}. \tag{5.45}$$

Такая функция s называется *полиномиальным сплайном порядка (степени)* m. Множество всех сплайнов степени m на разбиении Δ будем обозначать S_Δ^m.

5.4.2. Интерполяционные сплайны

Определение 5.9. Сплайн $s \in S_\Delta^m$ называется интерполяционным для функции f, если

$$s(x_i) = f(x_i) = y_i \quad \forall i = \overline{0, n}. \tag{5.46}$$

Интерполяционный сплайн $s \in S_\Delta^1$ представляет собой кусочно-линейную функцию (график — ломаная линия, рис. 5.3), построенную по точкам $\{(x_i, y_i)\}_{i=0}^n$.

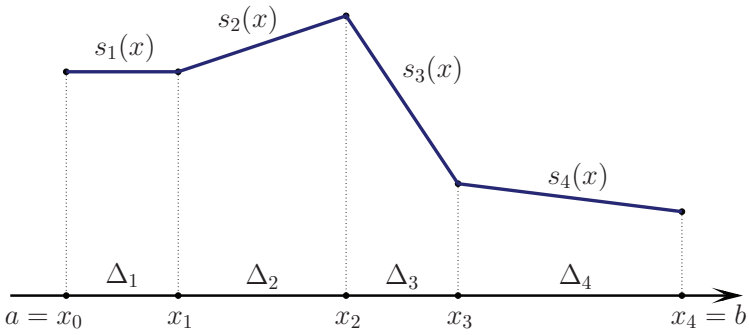


Рис. 5.3. Интерполяционный линейный сплайн

Больший интерес, конечно, представляют сплайны высших степеней. Наиболее распространенными являются кубические интерполяционные сплайны ($m = 3$, рис. 5.4).

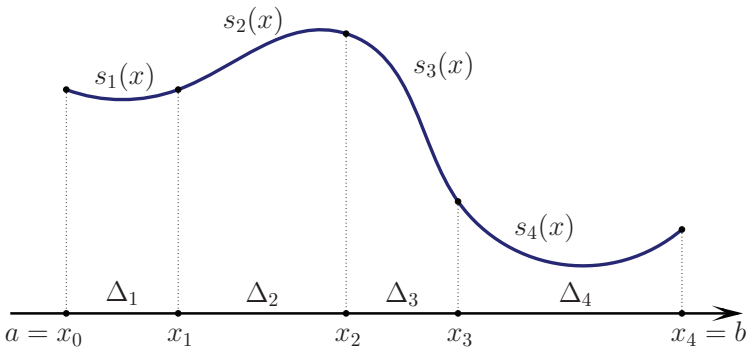


Рис. 5.4. Интерполяционный кубический сплайн

Построение кубического интерполяционного сплайна

Рассмотрим задачу нахождения $s \in S^3_\Delta$, удовлетворяющего условиям (5.46). Каждый «кусок» сплайна будем искать в виде

$$s_i(x) = \alpha_i + \beta_i(x - x_i) + \frac{\gamma_i}{2}(x - x_i)^2 + \frac{\delta_i}{6}(x - x_i)^3, \quad x \in \Delta_i = [x_{i-1}, x_i].$$

Такое представление в виде многочлена по степеням $(x - x_i)$ выбрано не случайно: при этом имеем

$$\begin{aligned} s_i'(x) &= \beta_i + \gamma_i(x - x_i) + \frac{\delta_i}{2}(x - x_i)^2, \\ s_i''(x) &= \gamma_i + \delta_i(x - x_i), \\ s_i'''(x) &= \delta_i \end{aligned}$$

и

$$s_i(x_i) = \alpha_i, \quad s_i'(x_i) = \beta_i, \quad s_i''(x_i) = \gamma_i.$$

Для построения всего сплайна нам нужно определить $4n$ неизвестных $\{\alpha_i, \beta_i, \gamma_i, \delta_i\}_{i=1}^n$. Обозначим

$$\alpha_0 = s_1(x_0), \quad \beta_0 = s_1'(x_0), \quad \gamma_0 = s_1''(x_0), \quad \delta_0 = s_1'''(x_0), \qquad (5.47)$$

после чего с полным правом можем записать

$$s(x_i) = \alpha_i, \quad s'(x_i) = \beta_i, \quad s''(x_i) = \gamma_i, \quad \forall\, i = \overline{0, n}.$$

Выпишем уравнения, которым должны удовлетворять искомые коэффициенты. Обозначим $h_i = x_i - x_{i-1}$. Прежде всего используем условия интерполяции (5.46). Имеем $s_i(x_i) = y_i$, откуда

$$\alpha_i = y_i, \quad i = \overline{1, n}, \tag{5.48а}$$

и $s_i(x_{i-1}) = y_{i-1}$, или

$$\beta_i = \frac{y_i - y_{i-1}}{h_i} + \frac{\gamma_i}{2} h_i - \frac{\delta_i}{6} h_i^2, \quad i = \overline{1, n}. \tag{5.48б}$$

Отметим, что выполнение этих двух условий сразу гарантирует непрерывность s. Теперь используем условия непрерывности s':

$$s_i'(x_{i-1}) = s_{i-1}'(x_{i-1}),$$

или

$$\beta_i - \gamma_i h_i + \frac{\delta_i}{2} h_i^2 = \beta_{i-1}, \quad i = \overline{2, n}. \tag{5.48в}$$

Наконец, требование непрерывности второй производной

$$s_i''(x_{i-1}) = s_{i-1}''(x_{i-1})$$

по аналогии дает

$$\delta_i = \frac{\gamma_i - \gamma_{i-1}}{h_i}, \quad i = \overline{2, n}. \tag{5.48г}$$

Важно: если в последних двух формулах положить $i = 1$, то мы получим просто определение величин β_0 и γ_0 согласно (5.47).

Формулы (5.48) дают $4n - 2$ уравнений для нахождения $4n$ неизвестных, поэтому без двух дополнительных условий сплайн однозначно не определить. Как правило, эти условия задаются на концах отрезка $[a, b]$ и поэтому называются *граничными*.

Величины γ_i называются *моментами* сплайна. Прежде чем рассматривать различные варианты задания дополнительных условий для определения интерполяционного кубического сплайна, заметим, что если известны значения всех моментов, то остальные коэффициенты сразу

находятся по формулам (5.48а), (5.48б) и (5.48г) (в последней нужно взять еще $i = 1$). В частности, (5.48б) дает

$$\beta_i = \frac{y_i - y_{i-1}}{h_i} + \frac{2\gamma_i + \gamma_{i-1}}{6} h_i. \qquad (5.48б')$$

Для нахождения $\{\gamma_i\}$ подставим (5.48б') и (5.48г) в (5.48в). После сдвига в нумерации получим

$$h_i\gamma_{i-1} + 2(h_i + h_{i+1})\gamma_i + h_{i+1}\gamma_{i+1} = 6\left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}\right),$$

что можно записать как

$$c_i\gamma_{i-1} + 2\gamma_i + e_i\gamma_{i+1} = b_i, \quad i = \overline{1, n-1}, \qquad (5.49)$$

где

$$c_i = \frac{h_i}{h_i + h_{i+1}}, \quad e_i = \frac{h_{i+1}}{h_i + h_{i+1}}, \quad b_i = 6f[x_{i-1}, x_i, x_{i+1}].$$

Формулы (5.49) дают $(n-1)$ линейных уравнений для нахождения $n+1$ неизвестных $\{\gamma_i\}_{i=0}^n$.

Таким образом, **общая схема вычисления интерполяционного кубического сплайна** такова:

1) вычисляются моменты $\{\gamma_i\}_{i=0}^n$ как решение СЛАУ (5.49), дополненной двумя дополнительными уравнениями — граничными условиями;
2) находятся остальные коэффициенты по формулам (5.48а), (5.48б') и (5.48г).

Виды граничных условий

Естественные граничные условия. Удобные граничные условия $s''(a) = s''(b) = 0$, как и соответствующий интерполяционный сплайн, называются *естественными*. В этом случае имеем $\gamma_0 = \gamma_n = 0$, а остальные $\{\gamma_i\}_{i=1}^{n-1}$ легко найти из (5.49), которая имеет вид

$$\begin{bmatrix} 2 & e_1 & & & & \\ c_2 & 2 & e_2 & & & \\ & c_3 & 2 & e_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{n-2} & 2 & e_{n-2} \\ & & & & c_{n-1} & 2 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \vdots \\ \gamma_{n-2} \\ \gamma_{n-1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix}.$$

Данную систему решают методом прогонки.

$\triangleright_1$ Докажите, что метод прогонки в данном случае применим.

$\triangleright_2$ Запишите вид аналогичной СЛАУ для граничных условий вида $s''(x_0) = f''(x_0)$, $s''(x_n) = f''(x_n)$.

Граничные условия на значение первой производной. Пусть в концах отрезка сплайн s должен удовлетворять дополнительным условиям

$$s'(a) = f'(a), \quad s'(b) = f'(b),$$

что равносильно $\beta_0 = f'(a)$, $\beta_n = f'(b)$. Рассмотрим (5.48в) при $i = 1$ с учетом (5.48б'):

$$\gamma_1 h_1 - \frac{\gamma_1 - \gamma_0}{2} h_1 = \frac{y_1 - y_0}{h_1} + \frac{2\gamma_1 + \gamma_0}{6} h_1 - f'(a),$$

откуда получим

$$2\gamma_0 + \gamma_1 = \frac{6}{h_1}\left(\frac{y_1 - y_0}{h_1} - f'(a)\right).$$

Аналогично для $i = n - 1$ имеем

$$\gamma_{n-1} + 2\gamma_n = \frac{6}{h_n}\left(f'(b) - \frac{y_n - y_{n-1}}{h_n}\right).$$

$\triangleright_3$ Запишите соответствующую СЛАУ для нахождения $\{\gamma_i\}_{i=0}^n$ и докажите, что к ней применим метод прогонки.

Другие граничные условия. На практике используется еще несколько видов граничных условий, например периодические:

$$s'(a) = s'(b), \quad s''(a) = s''(b)$$

или так называемые условия «отсутствия узлов», накладывающие требования непрерывности s''' в узлах x_1 и x_{n-1}.

$\triangleright_4$ Почему в этом случае говорят об «отсутствии узлов»?

Интересен также вариант

$$s'(a) = f'(a), \quad s''(a) = f''(a).$$

В этом случае вообще не нужно решать никакой СЛАУ — все s_i вычисляются явно по очереди от 1 до n.

5.4.3. Экстремальное свойство кубического сплайна

Рассмотрим функционал $\Phi : C^2[a,b] \to \mathbb{R}$,

$$\Phi(f) = \int_a^b (f''(x))^2 dx.$$

Экстремальное свойство интерполяционного сплайна $s \in S^3_\Delta$ состоит в том, что он доставляет минимум Φ среди всех функций из $C^2[a,b]$, интерполирующих f и удовлетворяющих требуемым граничным условиям. Сформулируем это утверждение более строго для случая естественных граничных условий.

Теорема 5.8. *Пусть $F_0 \subset C^2[a,b]$ — множество функций f, таких, что*

$$f(x_i) = y_i, \quad i = \overline{0,n}, \quad \textit{и} \quad f''(a) = f''(b) = 0.$$

Если кубический сплайн $s \in S^3_\Delta$ принадлежит F_0, то

$$\Phi(s) \leqslant \Phi(f) \quad \forall f \in F_0.$$

Доказательство. Для произвольной $f \in F_0$ рассмотрим функцию $g = f - s$. По условию имеем $g \in F_0$ и

$$g(x_i) = 0 \quad \forall i = \overline{0,n}.$$

Расписывая $\Phi(f) = \Phi(s+g)$, получим

$$\Phi(f) = \Phi(s) + \Phi(g) + 2\int_a^b s''g''dx.$$

Для доказательства теоремы теперь достаточно показать, что

$$\int_a^b s''g''dx \geqslant 0.$$

Воспользуемся интегрированием по частям, а также тем фактом, что s'''

на интервале (x_{i-1}, x_i) равна константе, которую обозначим δ_i:

$$\int_a^b s''g''dx = \sum_{i=1}^n \int_{x_{i-1}}^{x_i} s''g''dx = \sum_{i=1}^n \left(s''g' \Big|_{x_{i-1}}^{x_i} - \int_{x_{i-1}}^{x_i} s'''g'dx \right) =$$

$$= \underbrace{s''(b)g'(b)}_{0} - \underbrace{s''(a)g'(a)}_{0} - \sum_{i=1}^n \int_{x_{i-1}}^{x_i} s'''g'dx = -\sum_{i=1}^n \int_{x_{i-1}}^{x_i} s'''g'dx =$$

$$= -\sum_{i=1}^n \delta_i \int_{x_{i-1}}^{x_i} g'dx = -\sum_{i=1}^n \delta_i \Big(g(x_i) - g(x_{i-1}) \Big) dx = 0. \qquad \square$$

Отметим, что в аналитической геометрии кривизна кривой — графика $y = f(x)$ — вычисляется по формуле

$$\frac{|f''(x)|}{(1 + f'(x)^2)^{3/2}},$$

так что величину $\Phi(f)$ можно характеризовать как приближенное значение для усредненной кривизны графика. Поэтому часто говорят (пусть и не совсем строго), что интерполяционный сплайн обладает минимальной кривизной среди всех интерполирующих функций с указанными граничными условиями.

5.4.4. Сходимость интерполяции кубическим сплайном

Приведем без доказательства следующий результат.

Теорема 5.9. *Рассмотрим $f \in C^4[a,b]$ и разбиение Δ отрезка $[a,b]$, такое, что $\max_i h_i = h$. Пусть $s \in S_\Delta^3$ — естественный интерполяционный сплайн для f. Тогда*

$$\|f - s\| \leqslant \frac{5}{384} h^4 \|f^{(4)}\|.$$

Здесь $\|\cdot\| = \|\cdot\|_{C[a,b]}$.

Таким образом, для достаточно гладких f погрешность интерполирования кубическим сплайном $\varepsilon(h)$ равна $O(h^4)$.

5.5. Приближение кривых

5.5.1. Интерполяция кривых на плоскости

Постановка задачи

Рассмотрим на декартовой плоскости кривую ℓ. На данной кривой зададим направление движения и $n+1$ точек $p_i = (x_i, y_i)^T$, $i = \overline{0,n}$, расположенных последовательно в выбранном направлении (рис. 5.5).

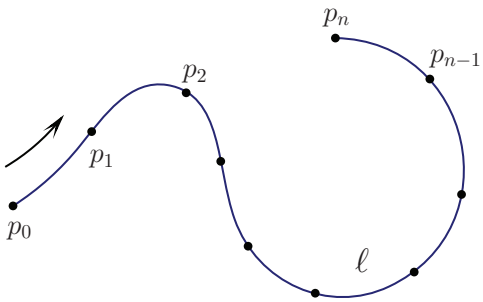


Рис. 5.5. Задача интерполяции кривой на плоскости

Определение 5.10. *Задача интерполяции кривой* ℓ по точкам $\{p_i\}$ заключается в построении такой параметрической функции $g : \mathbb{R} \to \mathbb{R}^2$,

$$g(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}, \tag{5.50}$$

что

$$g(t_i) = p_i, \quad \forall\, i = \overline{0,n},$$

где $\{t_i\}$ — некоторая возрастающая последовательность значений параметра:

$$t_0 < t_1 < \ldots < t_{n-1} < t_n,$$

которые будем называть *параметрическими узлами*.

В качестве параметрических узлов чаще всего используются следующие:

- *равномерные параметрические узлы* на отрезке $[0, n]$ или $[0, 1]$: $t_i = i$ и $t_i = i/n$ соответственно;

• *естественные параметрические узлы*, зависящие от расстояния между точками интерполяции p_i:

$$t_0 = 0, \quad t_{i+1} = t_i + \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, \quad i = \overline{1, n}.$$

Все эти определения, а также методы приближения плоских кривых, рассмотренные далее, могут быть естественным образом перенесены и на случай интерполяции кривой в пространстве размерности три и более.

5.5.2. Построение интерполирующих кривых

В дальнейшем будем считать, что множество параметрических узлов задано. Нетрудно видеть, что поставленная задача интерполяции кривой легко сводится к решению двух задач обыкновенной интерполяции: для построения отображения g в (5.50) достаточно построить функции $x : [t_0, t_n] \to \mathbb{R}$ и $y : [t_0, t_n] \to \mathbb{R}$, удовлетворяющие условиям

$$\begin{cases} x(t_i) = x_i, \\ y(t_i) = y_i, \end{cases} \quad i = \overline{0, n}. \tag{5.51}$$

Для построения таких x и y можно воспользоваться, например, алгебраической интерполяцией.

Алгебраическая интерполяция

Ввиду того что обе интерполирующие функции x и y строятся по одинаковой сетке узлов, аналог интерполяционного многочлена Лагранжа (5.8) можно записать следующим образом:

$$g(t) = \sum_{i=0}^{n} \Lambda_i(t) p_i = \sum_{i=0}^{n} \Lambda_i(t) \begin{bmatrix} x_i \\ y_i \end{bmatrix},$$

где базисная функция Λ_i определяется как обычно:

$$\Lambda_i(t) = \prod_{i \neq j} \frac{t - t_j}{t_i - t_j}.$$

$\triangleright_1$ Запишите аналоги интерполяционной формулы Ньютона и барицентрических интерполяционных формул.

Сплайновые интерполяционные кривые

Более эффективным инструментом интерполяции кривых являются кубические сплайны. Как обычно, рассмотрим разбиение интервала $[t_0, t_n]$:

$$\Delta = \{[t_{i-1}, t_i]\}_{i=1}^{n}.$$

Построение сплайновой кривой g сводится к построению двух интерполяционных сплайнов на этом разбиении:

$$x = s^1 \in S_\Delta^3, \quad y = s^2 \in S_\Delta^3$$

по условиям (5.51). Строятся эти сплайны по общей схеме, описанной в подп. 5.4.2. При этом, конечно, нужно задать граничные условия. В итоге мы получим кусочно-кубическую параметрическую функцию вида

$$g(t)|_{t\in\Delta_i} = \begin{bmatrix}\alpha_i^1\\ \alpha_i^2\end{bmatrix} + (t-t_i)\begin{bmatrix}\beta_i^1\\ \beta_i^2\end{bmatrix} + \frac{1}{2}(t-t_i)^2\begin{bmatrix}\gamma_i^1\\ \gamma_i^2\end{bmatrix} + \frac{1}{6}(t-t_i)^3\begin{bmatrix}\delta_i^1\\ \delta_i^2\end{bmatrix},$$

где $\{\alpha_i^1, \beta_i^1, \gamma_i^1, \delta_i^1\}$ — коэффициенты соответствующих «кусков» кубического сплайна $s^1 = x$, а $\{\alpha_i^2, \beta_i^2, \gamma_i^2, \delta_i^2\}$ — то же самое для $s^2 = y$.

5.5.3. Кривые Безье

Интерактивный дизайн кривой

Рассмотрим теперь задачу *интерактивного дизайна* кривой: требуется построить кривую, обладающую нужной *формой*. Эта задача существенно отличается от рассматриваемой ранее задачи интерполяции кривой тем, что здесь нет строго определенного объекта для приближения. Нужно путем визуального подбора построить кривую, обладающую определенными свойствами (например, имеющую форму кузова автомобиля).

Интерполяция многочленами плохо подходит для решения такой задачи. Это связано с тем, что базисные функции Лагранжа Λ_i хоть и обладают свойством $\Lambda_i(t_j) = \delta_{ij}$, но на всем остальном отрезке интерполяции $[a, b]$ могут принимать значения, намного большие единицы. Это приводит к тому, что небольшое изменение одной точки p_i может сильно изменить значение интерполяционного многочлена в точке t, расположенной достаточно далеко от узла t_i. Этот недостаток можно выразить так: i-я базисная функция Лагранжа Λ_i *не локализована вблизи i-го узла интерполяции*. Гораздо более удобным с этой точки зрения является базис из многочленов Бернштейна.

Базисные многочлены Бернштейна

Определение 5.11. Многочлен

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{5.52}$$

называют i-м (базисным) *многочленом Бернштейна* степени n, $0 \leqslant i \leqslant n$. Здесь $\binom{n}{i} = \dfrac{n!}{i!(n-i)!} = C_n^i$ — биномиальные коэффициенты. При $i < 0$ и $i > n$ полагаем $B_i^n(t) = 0$.

Множество $\{B_i^n\}_{i=0}^n$ образует базис в пространстве многочленов степени не выше n.

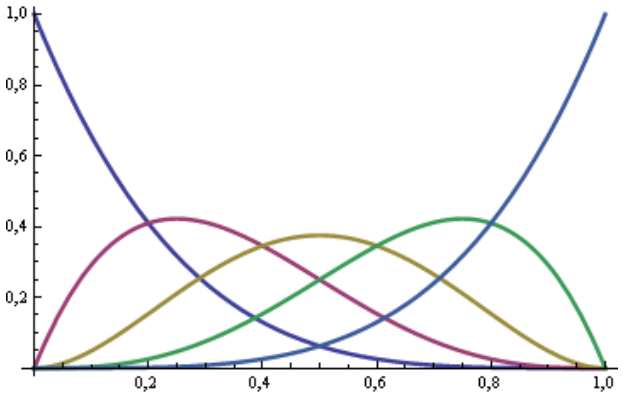


Рис. 5.6. Многочлены Бернштейна степени 4

Графики всех многочленов B_i^4 приведены на рис. 5.6. Многочлены Бернштейна обладают множеством полезных свойств и достаточно широко используются в численном анализе. Наиболее интересны для нас следующие свойства этих многочленов:

1) $B_i^n(t) \geqslant 0 \quad \forall\, t \in [0,1], \quad \forall\, n, i$;

2) $B_0^n(0) = B_n^n(1) = 1$;
$B_i^n(0) = B_i^n(1) = 0 \quad \forall i = \overline{1, n-1}$;

3) $B_i^n(t) = B_{n-i}^n(1-t)$;

4) $\dfrac{d}{dt} B_i^n(t) = n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t))$;

5) B_i^n имеет на отрезке $[0,1]$ единственный локальный максимум, который достигается в точке i/n (B_i^n локализована в точке i/n);

6) $\sum_{i=0}^{n} B_i^n(t) = 1 \quad \forall\, t \in \mathbb{R}, \quad n \geqslant 0$;
7) $B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$.

$\triangleright_2$ Докажите свойства 4–7.

Кривые Безье

Кривая Безье — это образ единичного отрезка под действием линейной комбинации базисных функций Бернштейна с векторными коэффициентами. Кривые Безье широко применяются в компьютерной графике, с их помощью работают практически все компьютерные векторные шрифты и др.

Определение 5.12. Рассмотрим набор точек плоскости $Q = \{q_i \in \mathbb{R}^2\}_{i=0}^n$, которые в дальнейшем будем называть *контрольными точками*. Кривой Безье называется множество точек

$$\{B(t) \mid t \in [0,1]\},$$

где

$$B(t) = \sum_{i=0}^{n} B_i^n(t) q_i. \tag{5.53}$$

Примеры кривых Безье для $n = 1, 2, 3$ изображены на рис. 5.7.

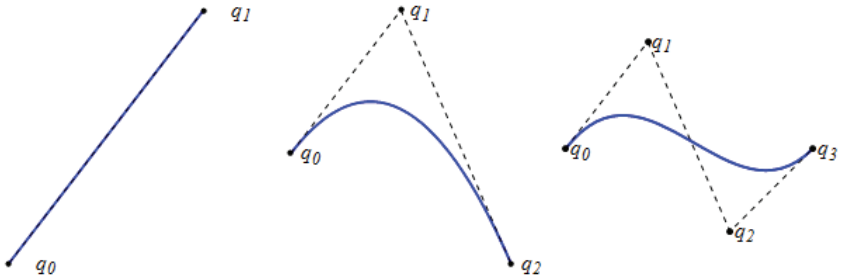


Рис. 5.7. Простейшие кривые Безье

Данный рисунок наглядно иллюстрирует следующие свойства кривых Безье:
1) $B(0) = q_0$, $B(1) = q_n$;
2) касательные к кривой в точках $t = 0$ и $t = 1$ коллинеарны векторам $q_1 - q_0$ и $q_n - q_{n-1}$ соответственно;

3) кривая Безье не выходит за пределы выпуклой оболочки множества контрольных точек Q.

Свойство 2 можно сформулировать более конкретно. Согласно свойству 4 многочленов Бернштейна, имеем

$$\frac{d}{dt}B(t) = \sum_{i=0}^{n} \frac{d}{dt}B_i^n(t)q_i = n\sum_{i=0}^{n}(B_{i-1}^{n-1}(t) - B_i^{n-1}(t))q_i = \\ = n\sum_{i=0}^{n-1} B_i^{n-1}(t)(q_{i+1} - q_i).$$

Отсюда с учетом свойства 2 многочленов Бернштейна получим

$$B'(0) = n(q_1 - q_0), \qquad B'(1) = n(q_n - q_{n-1}).$$

5.5.4. Алгоритмы построения кривых Безье

Рассмотрим задачу построения кривой Безье, задаваемой уравнением (5.53), которая сводится к вычислению значений многочлена B в точках $t \in [0, 1]$. Оба рассматриваемых способа основываются на рекуррентных соотношениях для многочленов Бернштейна (свойство 7).

Прямой алгоритм

Вычисление $B(t)$ для данного t можно разбить на два этапа:

1) вычисление $b_i = B_i^n(t)$ для всех i от 0 до n;

2) вычисление $B(t) = \sum_{i=0}^{n} b_i q_i$.

Основная работа выполняется на первом этапе, поэтому рассмотрим его. Непосредственное вычисление b_i по определению (5.52) приводит к вычислительной неустойчивости. Согласно свойству 7, имеем

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t). \tag{5.54}$$

Для того чтобы вычислить все b_i по такой схеме, необходимо вычислить все $B_j^k(t)$ для $k = \overline{0, n}$, $j = \overline{0, k}$. Осуществляется это последовательным

(слева направо) заполнением следующей треугольной таблицы:

$$\begin{array}{ccccccc} B_0^0 & B_0^1 & B_0^2 & \dots & B_0^{n-1} & B_0^n \\ & B_1^1 & B_1^2 & \dots & B_1^{n-1} & B_1^n \\ & & B_2^2 & \dots & \vdots & \vdots \\ & & & & B_{n-1}^{n-1} & B_{n-1}^n \\ & & & & & B_n^n \end{array}$$

Хранить всю таблицу не обязательно. Достаточно завести массив b размера $n+1$ и, начиная с

$$b = (B_0^0, 0, 0, \dots, 0) = (1, 0, 0, \dots, 0),$$

последовательно заполнить все его компоненты, в итоге получив

$$b = (B_0^n, B_1^n, \dots, B_n^n).$$

$\triangleright_3$ Запишите соответствующий алгоритм.

При построении графика кривой можно в два раза сократить объем вычислений, если учесть симметрию базисных многочленов Бернштейна (свойство 3): один раз вычислив b_i в точке t, можно найти не только $B(t)$, но и

$$B(1-t) = \sum_{i=0}^{n} b_{n-i} q_i.$$

Алгоритм de Casteljau

Формулу (5.54) можно применить непосредственно к многочлену (5.53). Если обозначить

$$B(t) = B(q_0, \dots, q_n, t),$$

то из указанных двух формул получим

$$\begin{aligned} B(q_0, \dots, q_n, t) &= (1-t) \sum_{i=0}^{n} B_i^{n-1} q_i + t \sum_{i=0}^{n} B_{i-1}^{n-1} q_i = \\ &= (1-t) B(q_0, \dots, q_{n-1}, t) + t\, B(q_1, \dots, q_n, t). \end{aligned} \tag{5.55}$$

Таким образом, по рекуррентной схеме (3) мы можем напрямую вычислить $B(t)$, заполняя треугольную таблицу практически аналогично схеме вычисления разделенных разностей. Начнем с массива

$$\big(B(q_0,t), B(q_1,t), \ldots, B(q_n,t)\big) = (q_0, q_1, \ldots, q_n)$$

и постепенно вычислим $B(q_i, \ldots, q_{i+k}, t)$, складывая соседние элементы массива с весами $(1-t)$ и t. Полученная схема вычислений приведена ниже (точку t в обозначении $B(\ldots)$ опустим):

$$\begin{array}{lllll}
q_0 & & & & \\
 & B(q_0,q_1) & & & \\
q_1 & & B(q_0,q_1,q_2) & & \\
 & B(q_1,q_2) & & \ldots & \\
q_2 & & B(q_1,q_2,q_3) & B(q_0,\ldots,q_{n-1}) & \\
\cdot & B(q_2,q_3) & \cdot & \ldots & B(q_0,\ldots,q_n) \\
\cdot & \cdot & \cdot & B(q_1,\ldots,q_n) & \\
\cdot & \cdot & \cdot & \ldots & \\
q_{n-1} & \cdot & B(q_{n-2},q_{n-1},q_n) & & \\
 & B(q_{n-1},q_n) & & & \\
q_n & & & &
\end{array}$$

$\triangleright_4$ Запишите алгоритм.

$\triangleright_5$ Сравните трудоемкость обоих алгоритмов.

5.6. Среднеквадратичные приближения

Сейчас мы рассмотрим принципиально отличный от интерполяции способ приближения функций — среднеквадратичное приближение. Такие приближения можно строить как по заданному набору точек, так и по заданной функции. Начнем с первого, дискретного, случая.

5.6.1. Дискретный случай

Постановка задачи

Рассмотрим на декартовой плоскости $N+1$ точек (x_k, y_k), $k = \overline{0, N}$ (рис. 5.8, слева). Можно считать, что эти точки представляют собой значения некоторой неизвестной функции f, заданные с какой-то погрешностью: $f(x_k) \approx y_k$. Часто в таком случае требуется найти приближение к функции f. Понятно, что интерполяция в данном случае подходит плохо.

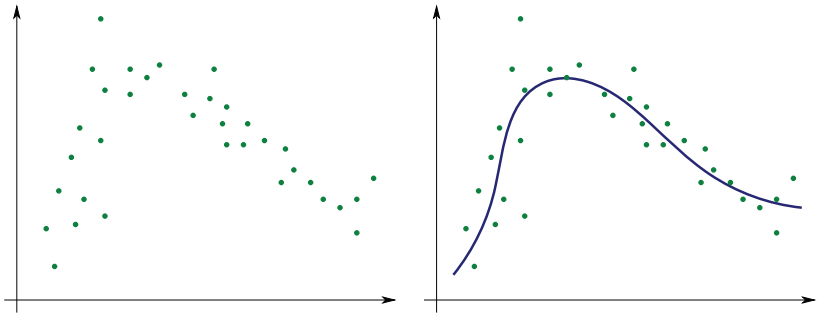


Рис. 5.8. Задача дискретного среднеквадратичного приближения

Вместо нее традиционно применяют метод *среднеквадратичного приближения*, он же *метод наименьших квадратов*. Этот метод состоит в поиске такой функции φ, для которой *сумма квадратов отклонений от y_i в точках x_i была бы минимальной* (рис. 5.8, справа).

Определение 5.13. *Задача дискретного среднеквадратичного приближения* для данного набора точек (x_k, y_k), $k = \overline{0, N}$, заключается в построении функции φ вида

$$\varphi(x) = \sum_{i=0}^{n} \alpha_i \varphi_i(x),$$

для которой выражение

$$\sum_{k=0}^{N} (y_k - \varphi(x_k))^2 \tag{5.56}$$

принимает минимальное возможное значение. Здесь $\{\varphi_i\}_{i=0}^{n}$ — некоторый заданный базис функций.

Искомая функция φ однозначно определяется своими коэффициентами $\{\alpha_i\}$, которые представим в виде вектора $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_n)^T$. Запишем выражение (5.56) в виде функционала $\Phi : \mathbb{R}^{n+1} \to \mathbb{R}$,

$$\Phi(\alpha) = \sum_{k=0}^{N} \left(y_k - \sum_{i=0}^{n} \alpha_i \varphi_i(x_k) \right)^2. \tag{5.57}$$

Задача свелась к нахождению вектора α, доставляющего минимум $\Phi(\alpha)$.

Вывод расчетных формул

Рассмотрим внимательнее структуру функционала Φ:

$$\Phi(\alpha) = \sum_{k=0}^{N} \left(y_k - \sum_{i=0}^{n} \alpha_i \varphi_i(x_k) \right)^2 =$$

$$= \sum_{k=0}^{N} \left(y_k^2 - 2y_k \sum_{i=0}^{n} \alpha_i \varphi_i(x_k) + \left(\sum_{i=0}^{n} \alpha_i \varphi_i(x_k) \right)^2 \right) =$$

$$= \sum_{k=0}^{N} y_k^2 - 2 \sum_{k=0}^{N} y_k \sum_{i=0}^{n} \alpha_i \varphi_i(x_k) + \sum_{k=0}^{N} \sum_{i=0}^{n} \alpha_i \varphi_i(x_k) \sum_{j=0}^{n} \alpha_j \varphi_j(x_k) =$$

$$= \sum_{k=0}^{N} y_k^2 - 2 \sum_{i=0}^{n} \alpha_i \sum_{k=0}^{N} y_k \varphi_i(x_k) + \sum_{i=0}^{n} \sum_{j=0}^{n} \alpha_i \alpha_j \sum_{k=0}^{N} \varphi_i(x_k) \varphi_j(x_k).$$

Если ввести обозначения

$$y = (y_0, y_1, \ldots, y_N)^T, \quad \beta_i = \sum_{k=0}^{N} y_k \varphi_i(x_k), \quad \gamma_{ij} = \sum_{k=0}^{N} \varphi_i(x_k) \varphi_j(x_k),$$

то полученное выражение можно переписать в виде

$$\Phi(\alpha) = (y, y) - 2 \sum_{i=0}^{n} \beta_i \alpha_i + \sum_{i=0}^{n} \sum_{j=0}^{n} \gamma_{ij} \alpha_i \alpha_j. \tag{5.58}$$

Как видим, $\Phi(\alpha)$ представляет собой квадратичный многочлен по каждой переменной α_i, причем $\Phi(\alpha) \geqslant 0$ при любых $\alpha \in \mathbb{R}^{n+1}$.

$\triangleright_1$ В каком случае минимальное значение Φ равно нулю?

Значит, единственный глобальный минимум Φ может быть найден как решение системы уравнений

$$\frac{\partial}{\partial \alpha_l}\Phi(\alpha) = 0, \quad l = \overline{0,n}.$$

Непосредственным дифференцированием получим

$$\frac{\partial}{\partial \alpha_l}\Phi(\alpha) = -2\beta_l + 2\sum_{j=0}^{n}\gamma_{lj}\alpha_j.$$

$\triangleright_2$ Выведите эту формулу.

Следовательно, искомый вектор α является решением системы линейных уравнений

$$\sum_{j=0}^{n}\gamma_{lj}\alpha_j = \beta_l, \quad l = \overline{0,n},$$

которая в матричном виде записывается просто как

$$\Gamma\alpha = \beta, \tag{5.59}$$

$$\begin{aligned} &\Gamma = \big(\gamma_{ij}\big)_{i,j=0}^{n}, && \gamma_{ij} = \sum_{k=0}^{N}\varphi_i(x_k)\varphi_j(x_k), \\ &\beta = (\beta_0, \beta_1, \ldots, \beta_n)^T, && \beta_i = \sum_{k=0}^{N} y_k\varphi_i(x_k). \end{aligned} \tag{5.60}$$

Заметим, что матрица Γ представляет собой матрицу скалярных произведений (матрицу Грама) для системы векторов

$$\phi_i = \big(\varphi_i(x_0), \varphi_i(x_1), \ldots, \varphi_i(x_N)\big)^T, \quad i = \overline{0,n}, \tag{5.61}$$

т. е.

$$\gamma_{ij} = (\phi_i, \phi_j),$$

а

$$\beta_i = (y, \phi_i).$$

Таким образом, *решение задачи дискретного среднеквадратичного приближения сводится к вычислению вектора неизвестных* α, *удовлетворяющего СЛАУ* (5.59), *где матрица* Γ *и вектор* β *определяются формулами* (5.60). Эта задача однозначно разрешима тогда

и только тогда, когда матрица Γ невырождена. В силу вышесказанного для этого необходимо и достаточно, чтобы система векторов $\{\phi_i\}$ (5.61) была линейно независимой. Матрица Γ очевидно является симметричной. Поэтому для решения СЛАУ (5.59) естественно применить метод квадратного корня.

Замечание 5.3. При больших значениях N формулы (5.60) могут давать очень большие по модулю значения коэффициентов матрицы Γ и вектора β, что приведет к большим вычислительным погрешностям. Чтобы этого избежать, следует пронормировать эти коэффициенты, разделив их на N, т. е. фактически решать СЛАУ

$$\frac{1}{N}\Gamma\alpha = \frac{1}{N}\beta.$$

5.6.2. Среднеквадратичное приближение функций

Как уже говорилось, среднеквадратичное приближение можно строить не только для набора точек, но и для аналитически заданной функции f на отрезке $[a, b]$. При этом общая схема решения задачи останется практически без изменений. Главным условием применимости этого метода является *интегрируемость с квадратом* функции f.

Постановка задачи

Определение 5.14. Функция f называется *интегрируемой с квадратом на отрезке* $[a, b]$, если

$$\int_a^b |f(x)|^2 dx < \infty.$$

В этом случае говорят, что f принадлежит пространству $L_2[a, b]$.

Это определение подходит и для комплекснозначных функций f. Мы же в дальнейшем будем считать, что f принимает на $[a, b]$ только вещественные значения.

Определение 5.15. *Задача среднеквадратичного приближения интегрируемой с квадратом функции* f на отрезке $[a, b]$ заключается в поиске функции φ вида

$$\varphi(x) = \sum_{i=0}^{n} \alpha_i \varphi_i(x),$$

для которой интеграл

$$\int_a^b \big(f(x) - \varphi(x)\big)^2 dx \tag{5.62}$$

принимает наименьшее возможное значение. Как обычно, здесь $\{\varphi_i\}$ — заданная система базисных функций.

Замечание 5.4. Иногда вместо (5.62) рассматривают интеграл более общего вида

$$\int_a^b \big(f(x) - \varphi(x)\big)^2 \rho(x) dx, \tag{5.63}$$

где ρ — некоторая неотрицательная функция, называемая *весовой функцией*.

Вывод расчетных формул

Совершенно аналогично дискретному случаю задача среднеквадратичного приближения функции f сводится к нахождению вектора коэффициентов $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_n)^T$, доставляющего минимум функционала, получаемого подстановкой $\varphi(x) = \sum_{i=0}^n \alpha_i \varphi_i(x)$ в формулу (5.62):

$$\Psi(\alpha) = \int_a^b \left(f(x) - \sum_{i=0}^n \alpha_i \varphi_i(x) \right)^2 dx.$$

Этот функционал имеет ту же структуру, что и Φ для дискретного случая (5.57). Разница лишь в том, что теперь вместо конечной суммы по точкам $\{x_i\}$ мы имеем сумму бесконечную — интеграл. Линейность интеграла позволяет точно так же, как это было сделано выше для $\Phi(\alpha)$, записать $\Psi(\alpha)$ в виде

$$\Psi(\alpha) = \int_a^b f(x)^2 dx - 2\sum_{i=0}^n \beta_i \alpha_i + \sum_{i=0}^n \sum_{j=0}^n \gamma_{ij} \alpha_i \alpha_j, \tag{5.64}$$

где

$$\beta_i = \int_a^b f(x)\varphi_i(x) dx, \quad \gamma_{ij} = \int_a^b \varphi_i(x)\varphi_j(x) dx, \quad i, j = \overline{0, n}. \tag{5.65}$$

$\triangleright_3$ Выведите формулы (5.65).

Заметим, что здесь мы использовали те же обозначения (β_i и γ_{ij}), что и ранее, так как они имеют схожий смысл.

Далее по уже известной схеме приравняем $\dfrac{\partial \Psi(\alpha)}{\partial \alpha_l}$ к нулю и получим следующий результат. *Решение задачи среднеквадратичного приближения функции f сводится к вычислению вектора неизвестных α, удовлетворяющего системе линейных уравнений* (5.59),

$$\Gamma \alpha = \beta,$$

где элементы матрицы $\Gamma = \left(\gamma_{ij}\right)_{i,j=0}^{n}$ *и вектора* $\beta = (\beta_0, \beta_1, \ldots, \beta_n)^T$ *определяются формулами* (5.65). Матрица Γ в данном случае также называется матрицей Грама, так как ее элементы $\gamma_{ij} = \int_a^b \varphi_i(x)\varphi_j(x)dx$ представляют собой *скалярные произведения функций* φ_i и φ_j. Эта матрица невырождена тогда и только тогда, когда система функций $\{\varphi_i\}$ линейно независима.

$\triangleright_4$ Получите выражения для β_i и γ_{ij} в случае, когда функционал Ψ определяется формулой (5.63).

5.6.3. Базисные функции

Обсудим теперь вопрос выбора базисных функций $\{\varphi_i\}$ для построения среднеквадратичного приближения.

Степенной базис

Если мы хотим, чтобы приближающая функция φ была алгебраическим многочленом, самый простой базис в этом случае имеет вид

$$\varphi_i(x) = x^i, \quad i = \overline{0, n}. \tag{5.66}$$

Матрица Грама Γ для этого базиса вычисляется легко (предположим, что $[a, b] = [0, 1]$):

$$\gamma_{ij} = \int_0^1 x^{i+j} dx = \frac{1}{i+j+1}, \quad \text{т. е.}$$

$$\Gamma = H_{n+1} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \dots & \frac{1}{n+1} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \dots & \frac{1}{n+2} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \dots & \frac{1}{n+3} \\ \dots & \dots & \dots & \dots & \dots \\ \frac{1}{n+1} & \frac{1}{n+2} & \frac{1}{n+3} & \dots & \frac{1}{2n+1} \end{bmatrix}.$$

Это знаменитая матрица Гильберта, классический пример плохо обусловленной матрицы. Ее число обусловленности очень быстро растет с ростом n, поэтому на практике базис (5.66) применяется лишь при небольших n.

$\triangleright_5$ Вычислите матрицу Грама для степенного базиса в случае произвольного отрезка $[a, b]$.

$\triangleright_6$ Запишите вид соответствующего вектора β.

Вместо базиса (5.66) для более эффективного вычисления среднеквадратичного приближения полиномами используются различные системы *ортогональных многочленов*, с которыми мы познакомимся в п. 6.4.

Классический тригонометрический базис

На практике часто используется тригонометрический базис на отрезке $[-\pi, \pi]$:

$$\{\varphi_i(x)\}_{i=0}^{n} = \{1, \cos x, \sin x, \cos 2x, \sin 2x, \dots, \cos Kx, \sin Kx\},$$

где $n = 2K$. Данный базис замечателен тем, что он является *ортогональным*, т. е. базисом, для которого матрица Грама диагональна (в случае единичной весовой функции ρ). Действительно, при таком выборе φ_i имеем

$$\gamma_{ij} = (\varphi_i, \varphi_j) = \int_{-1}^{1} \varphi_i(x)\varphi_j(x)dx = \begin{cases} 0, & i \neq j, \\ 2\pi, & i = j = 0, \\ \pi, & i = j \neq 0. \end{cases}$$

Следовательно, решение системы $\Gamma\alpha = \beta$ для нахождения коэффициентов среднеквадратичного приближения можно выписать сразу:

$$\alpha_0 = \frac{1}{2\pi}\beta_0 = \frac{1}{2\pi}\int\limits_{-\pi}^{\pi} f(x)dx,$$

$$\alpha_{2k-1} = \frac{1}{\pi}\beta_{2k-1} = \frac{1}{\pi}\int\limits_{-\pi}^{\pi} f(x)\cos kx\,dx,$$

$$\alpha_{2k} = \frac{1}{\pi}\beta_{2k} = \frac{1}{\pi}\int\limits_{-\pi}^{\pi} f(x)\sin kx\,dx, \quad k = \overline{1, K}.$$

Полученное таким образом приближение φ является отрезком *ряда Фурье* для функции f. Ряды Фурье широко используются в прикладной математике, подробности можно найти, например, в [9, гл. IV].

Фундаментальный кусочно-линейный базис

Рассмотрим разбиение отрезка $[a, b] = [x_0, x_n]$ на n частей:

$$\Delta = \{\Delta_i\}_{i=1}^{n}, \quad \Delta_i = [x_{i-1}, x_i]$$

и S_Δ^1 — множество сплайнов первого порядка на этом разбиении. Напомним, что элементами этого множества являются непрерывные кусочно-линейные функции. Предположим, что необходимо найти среднеквадратичное приближение к функции f среди элементов S_Δ^1.

Для решения этой задачи необходимо задать базис пространства S_Δ^1. Наиболее популярным является фундаментальный базис, i-я функция в котором принимает значение 1 в узле x_i и 0 во всех остальных узлах. Этого описания достаточно для определения базиса, так как любая функция из множества S_Δ^1 однозначно определяется своими значениями в узлах $\{x_i\}$. Аналитические формулы для базисных функций (обозначим их $\{\psi_i\}$) выглядят следующим образом:

$$\psi_0(x) = \begin{cases} \dfrac{x_1 - x}{h_1}, & x \in \Delta_1, \\ 0, & x \notin \Delta_1, \end{cases} \qquad \psi_n(x) = \begin{cases} \dfrac{x - x_{n-1}}{h_n}, & x \in \Delta_n, \\ 0, & x \notin \Delta_n, \end{cases}$$

$$\psi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{h_i}, & x \in \Delta_i, \\ \dfrac{x_{i+1} - x}{h_{i+1}}, & x \in \Delta_{i+1}, \\ 0, & x \notin \Delta_i \cup \Delta_{i+1}, \end{cases} \qquad i = \overline{1, n-1}.$$

Напомним обозначения: здесь $\Delta_i = [x_{i-1}, x_i]$, $h_i = x_i - x_{i-1}$.

Графики $\{\psi_i\}$ для частного случая $n = 3$ приведены на рис. 5.9.

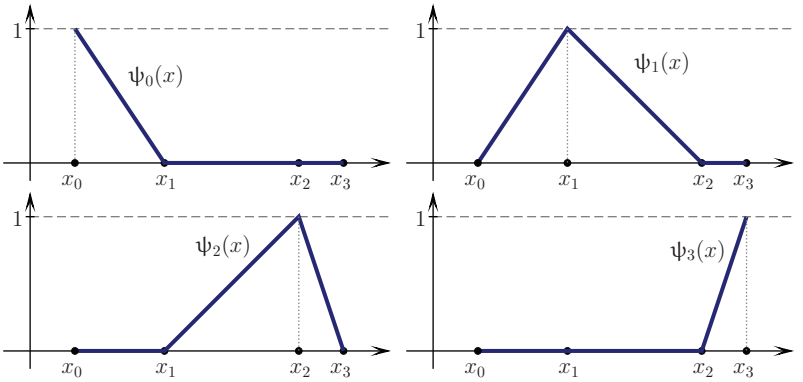


Рис. 5.9. Базис пространства кусочно-линейных функций на сетке

Интересно вычислить матрицу Грама для построенного базиса. Эта матрица будет трехдиагональной, так как при $|i - j| \geqslant 2$ имеем $\psi_i(x)\psi_j(x) = 0$ для всех x. Прямым вычислением получим

$$\Gamma = \begin{bmatrix} d_0 & c_1 & & & \\ c_1 & d_1 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-1} & d_{n-1} & c_n \\ & & & c_n & d_n \end{bmatrix}, \tag{5.67}$$

где

$$\begin{aligned} &d_i = \frac{h_i + h_{i+1}}{3}, \quad i = \overline{1, n-1}, \quad d_0 = \frac{h_1}{3}, \quad d_n = \frac{h_n}{3}, \\ &c_i = \frac{h_i}{6}, \quad i = \overline{1, n}. \end{aligned} \tag{5.68}$$

▷7 Выведите эти формулы.

5.7. Приближение поверхностей

5.7.1. Интерполяция на прямоугольнике

Задача приближения функции одной переменной, которую мы рассматривали до сих пор, естественным образом обобщается на случай функций нескольких переменных. Основное внимание в дальнейшем мы уделим задаче интерполяции функций двух переменных на прямоугольной области.

Определение 5.16. Рассмотрим на декартовой плоскости прямоугольник $\Pi = [a, b] \times [c, d]$. На этом прямоугольнике зададим множество точек — *прямоугольную сетку*

$$(x_i, y_j), \quad i = \overline{0, n}, \quad j = \overline{0, m},$$

порождаемую одномерными сетками на $[a, b]$ и $[c, d]$ соответственно (рис. 5.10):

$$a = x_0 < x_1 < \ldots < x_n = b, \quad \text{и} \quad c = y_0 < y_1 < \ldots < y_m = d.$$

Задача интерполяции функции $f : \Pi \to \mathbb{R}$ на данной сетке узлов заключается в построении функции $\varphi : \Pi \to \mathbb{R}$, такой, что

$$\varphi(x_i, y_j) = z_{ij} = f(x_i, y_j) \quad i = \overline{0, n}, \quad j = \overline{0, m}.$$

Вид функции φ, как правило, задается заранее.

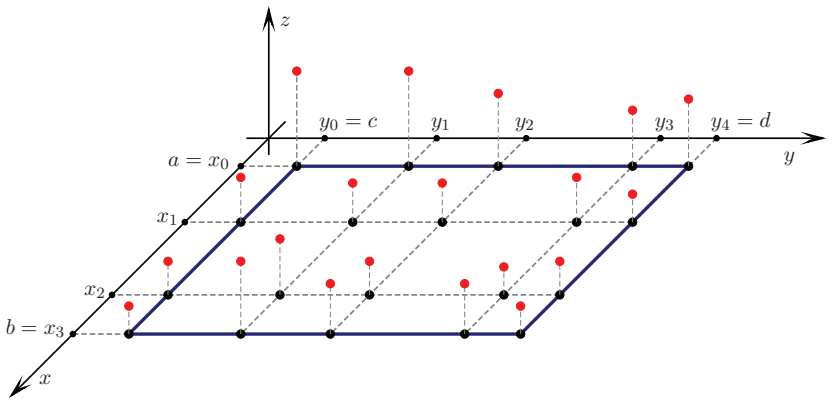


Рис. 5.10. Двумерная интерполяция на прямоугольной сетке

Преимущество такой постановки задачи в ее простоте: для интерполяции по прямоугольнику легко адаптируются одномерные методы интерполяции, которые мы успели рассмотреть.

5.7.2. Полиномиальная интерполяция

Рассмотрим случай, когда искомая интерполирующая функция φ является алгебраическим многочленом двух переменных, а точнее многочленом степени n по переменной x и степени m по y:

$$\varphi(x,y) = P_{n,m}(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} \alpha_{ij} x^i y^j.$$

Такую задачу интерполяции можно решить «методом грубой силы»: составить систему из $(n+1)(m+1)$ линейных уравнений для нахождения неизвестных коэффициентов $\{\alpha_{ij}\}$. Уравнения будут иметь вид

$$\sum_{i=0}^{n}\sum_{j=0}^{m} \alpha_{ij} x_p^i y_q^j = z_{pq}, \quad p = \overline{0,n}, \quad q = \overline{0,m}.$$

Но это слишком нерационально. Гораздо проще обобщить формулу интерполяционного многочлена Лагранжа (5.8) на случай двух переменных.

Сначала построим одномерные базисные функции Лагранжа (5.7) отдельно для узлов $\{x_i\}_{i=0}^n$ и $\{y_j\}_{j=0}^m$:

$$\Lambda_i^1(x) = \prod_{p\neq i} \frac{x - x_p}{x_i - x_p}, \quad i = \overline{0,n},$$

$$\Lambda_j^2(y) = \prod_{q\neq j} \frac{y - y_q}{y_j - y_q}, \quad j = \overline{0,m}.$$

Теперь рассмотрим базисные многочлены двух переменных

$$\Lambda_{ij}(x,y) = \Lambda_i^1(x)\Lambda_j^2(y).$$

Это *фундаментальный базис двумерной алгебраической интерполяции*, так как по построению имеем

$$\Lambda_{ij}(x_p, y_q) = \begin{cases} 1, & \text{если } p = i \text{ и } q = j \\ 0, & \text{в противном случае.} \end{cases}$$

Следовательно, многочлен

$$P_{n,m}(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} z_{ij}\Lambda_{ij}(x,y) \tag{5.69}$$

удовлетворяет условиям интерполяции

$$P_{n,m}(x_i,y_j) = z_{ij}, \quad \forall\, i = \overline{0,n}, \quad j = \overline{0,m},$$

т. е. является искомым интерполяционным многочленом в форме Лагранжа.

$\triangleright_1$ Постройте аналог барицентрической формулы Лагранжа для двумерного случая.

5.7.3. Интерполяция билинейными сплайнами

Теперь рассмотрим обобщение понятия линейного сплайна на случай двух переменных. Начнем с обозначений для отрезков:

$$\Delta_i^1 = [x_{i-1}, x_i], \quad i = \overline{1,n},$$
$$\Delta_j^2 = [y_{j-1}, y_j], \quad j = \overline{1,m}.$$

Соответствующие разбиения обозначим $\Delta^1 = \{\Delta_i^1\}_{i=1}^n$, $\Delta^2 = \{\Delta_j^2\}_{j=1}^m$. Эти разбиения порождают разбиение прямоугольника $\Pi = [a,b]\times[c,d]$ на множество прямоугольников

$$\Delta = \Delta^1 \times \Delta^2 = \{\Delta_{ij} = \Delta_i^1 \times \Delta_j^2\}_{i=1,j=1}^{n,m}.$$

Определение 5.17. *Билинейным сплайном* на разбиении $\Delta = \Delta^1 \times \times\Delta^2$ называется непрерывная функция $s : \Pi \to \mathbb{R}$, которая на каждом прямоугольнике Δ_{ij} является *билинейной функцией* (рис. 5.11):

$$s(x,y)\Big|_{(x,y)\in\Delta_{ij}} = s_{ij}(x,y) = \alpha_{ij}^{00} + \alpha_{ij}^{10}x + \alpha_{ij}^{01}y + \alpha_{ij}^{11}xy. \tag{5.70}$$

Заметим, что можно дать и другое, эквивалентное, определение билинейного сплайна: это функция двух переменных s такая, что при любом фиксированном $x \in [a,b]$ выражение $s(x,y)$ как функция от y является сплайном первого порядка на разбиении Δ^2. А при любом фиксированном $y \in [c,d]$ выражение $s(x,y)$ как функция от x является сплайном первого порядка на разбиении Δ^1.

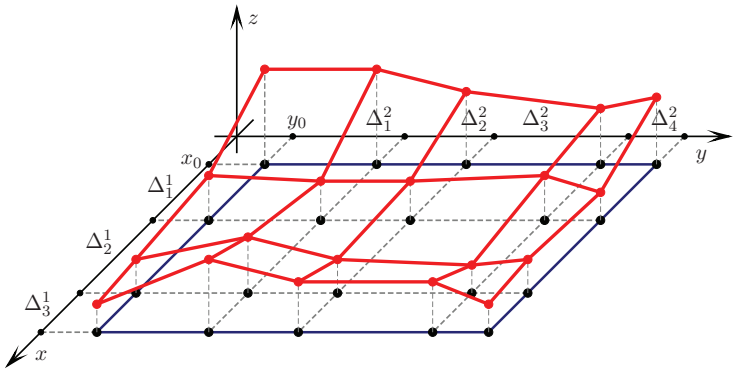


Рис. 5.11. Билинейный сплайн

Построение интерполяционного билинейного сплайна, удовлетворяющего условиям

$$s(x_i, y_j) = z_{ij},$$

не представляет проблем. На каждом прямоугольнике Δ_{ij} «кусок» сплайна s_{ij} (5.70) можно построить, используя формулу Лагранжа (5.69) по очевидным условиям

$$s_{ij}(x_{i-1}, y_{j-1}) = z_{i-1,j-1}, \qquad s_{ij}(x_i, y_{j-1}) = z_{i,j-1},$$
$$s_{ij}(x_{i-1}, y_j) = z_{i-1,j}, \qquad s_{ij}(x_i, y_j) = z_{i,j}.$$

$\triangleright_2$ Выведите формулы для вычисления $s_{ij}(x, y)$.

5.7.4. Интерполяция бикубическими сплайнами

Немного сложнее обстоят дела с кубическими сплайнами двух переменных — бикубическими сплайнами, которые будут сейчас рассмотрены. Для начала дадим определение с учетом введенных выше обозначений.

Определение 5.18. *Бикубическим сплайном*, определенном на разбиении $\Delta = \Delta^1 \times \Delta^2$, называется функция $s : \Pi \to R$, удовлетворяющая следующим двум условиям:

- при любом фиксированном значении $x \in [a, b]$ выражение $s(x, y)$ как функция переменной y является кубическим сплайном на разбиении Δ^2;

• при любом фиксированном значении $y \in [c, d]$ выражение $s(x, y)$ как функция переменной x является кубическим сплайном на разбиении Δ^1.

Из этого определения следует, что:

1) на каждом прямоугольнике Δ_{ij} бикубический сплайн s задается многочленом третьей степени по каждой переменной;

2) первая и вторая производные бикубического сплайна, включая смешанные, непрерывны на всем прямоугольнике Π.

Эти два свойства можно считать эквивалентным определением бикубического сплайна (рис. 5.12).

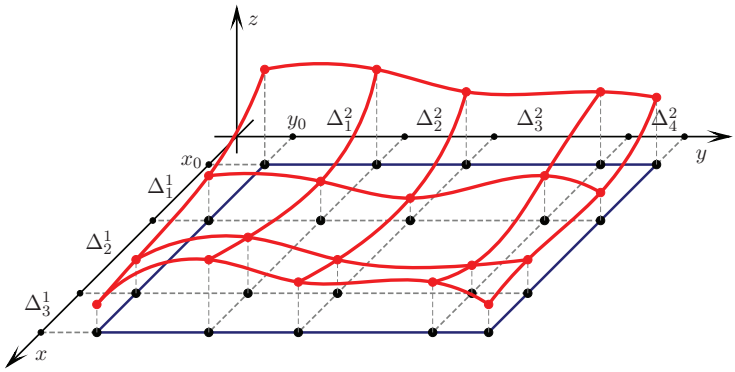


Рис. 5.12. Бикубический сплайн

Приступим теперь к построению интерполяционного бикубического сплайна s, удовлетворяющего условиям

$$s(x_i, y_j) = z_{ij}.$$

Для решения этой задачи достаточно уметь строить одномерные кубические сплайны (см. подп. 5.4.2).

Зафиксируем произвольное $y \in [c, d]$. Тогда по определению $s(x, y)$ представляет собой обычный кубический сплайн по переменной x, который можно записать, как мы делали это ранее:

$$s(x,y)\Big|_{x\in\Delta_i^1} = \alpha_i(y)+\beta_i(y)(x-x_i)+\frac{\gamma_i(y)}{2}(x-x_i)^2+\frac{\delta_i(y)}{6}(x-x_i)^3. \quad (5.71)$$

Здесь коэффициенты $\alpha_i, \beta_i, \gamma_i, \delta_i$, $i = \overline{1,n}$, очевидно зависят от зафиксированной нами точки y. Более того, из определения бикубического сплайна следует, что эти коэффициенты, как функции переменной y, являются кубическими сплайнами на разбиении Δ^2:

$$\alpha_i, \beta_i, \gamma_i, \delta_i \in S^3_{\Delta^2}, \quad i = \overline{1,n}.$$

Как только мы определим эти сплайны, а их в общей сложности $4n$ штук, искомый бикубический сплайн будет построен.

Необходимые для этого условия получаются из (5.71) подстановкой $y = y_j$, $j = \overline{0,m}$. В обозначениях

$$\alpha_i(y_j) = \alpha_{ij}, \quad \beta_i(y_j) = \beta_{ij}, \quad \delta_i(y_j) = \delta_{ij}, \quad \gamma_i(y_j) = \gamma_{ij} \qquad (5.72)$$

эти условия для каждого j имеют вид

$$s(x, y_j)\Big|_{x\in\Delta^1_i} = u_j(x)\Big|_{x\in\Delta^1_i} =$$
$$= \alpha_{ij} + \beta_{ij}(x - x_i) + \frac{\gamma_{ij}}{2}(x - x_i)^2 + \frac{\delta_{ij}}{6}(x - x_i)^3, \quad i = \overline{1,n}.$$

Здесь u_j — кубический сплайн на разбиении Δ^1. Его коэффициенты $\{\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij}\}_{i=1}^n$ легко найти, так как по условию u_j должен удовлетворять условиям интерполяции

$$u_j(x_i) = z_{ij}, \quad i = \overline{0,n}.$$

При этом, как и в одномерном случае, для однозначного определения сплайна нужно будет задать дополнительные граничные условия. Это могут быть, в принципе, любые условия из рассмотренных нами ранее для кубических сплайнов. Вычисляя сплайны u_j для $j = \overline{0,m}$ по схеме, описанной в подп. 5.4.2, мы находим $\{\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij}\}_{i=1}^n$.

После этого остается вычислить искомые сплайны $\alpha_i, \beta_i, \gamma_i, \delta_i$ по условиям интерполяции (5.72) для j от 0 до m, а также по соответствующим граничным условиям.

▷3 Укажите количество и размерность СЛАУ, которые нужно решить для построения бикубического сплайна таким способом.

Замечание 5.5. Строить бикубический сплайн можно и в другой последовательности, если вместо y в самом начале зафиксировать переменную x.

5.7.5. Поверхности Безье

Теперь от интерполяции перейдем к задаче интерактивного дизайна поверхностей, используя обобщение кривой Безье — поверхность Безье.

Определение 5.19. Рассмотрим набор точек в трехмерном пространстве $Q = \{q_{ij} \in \mathbb{R}^3\}_{i=0,j=0}^{n,m}$, которые в дальнейшем будем называть *контрольными точками*. Поверхностью Безье называется множество точек

$$\{B(u,v) \mid u,v \in [0,1]\},$$

где

$$B(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u) B_j^m(v) q_{ij}. \tag{5.73}$$

Здесь B_i^n и B_j^m — многочлены Бернштейна (5.52).

Контрольные точки $\{q_{ij}\}$ играют здесь ту же роль, что и в случае кривой Безье: они контролируют форму поверхности. В частности, поверхность Безье проходит через угловые точки q_{00}, q_{n0}, q_{m0} и q_{mn}, а также не выходит за пределы выпуклой оболочки контрольных точек.

Для построения поверхностей Безье используются те же вычислительные алгоритмы, что и для кривых.

$\triangleright_4$ Запишите адаптацию алгоритма de Casteljau для поверхностей Безье.

Глава 6

ПРИБЛИЖЕННОЕ ВЫЧИСЛЕНИЕ ИНТЕГРАЛОВ

6.1. Интерполяционные квадратурные формулы

6.1.1. Постановка задачи численного интегрирования

Рассмотрим задачу вычисления интеграла

$$\int_a^b f(x)dx,$$

где f — интегрируемая по Риману функция на отрезке $[a, b]$. Как известно, далеко не всегда значение интеграла может быть выражено через элементарные функции, поэтому необходимы методы приближенного интегрирования, которые будут рассматриваться далее.

Теоретически для приближенного вычисления $\int_a^b f(x)dx$ можно воспользоваться определением интеграла Римана как предела интегральных сумм вида

$$\sum_i f(\xi_i)\Delta x_i,$$

где $\{x_i\}$ — разбиение отрезка интегрирования; $\Delta x_i = x_i - x_{i-1}$; $\xi_i \in \in [x_{i-1}, x_i]$. Но такой способ, как правило, сходится очень медленно. Гораздо эффективнее *приблизить подынтегральную функцию некоторой функцией* φ*, интеграл от которой вычисляется легко, и заменить интеграл от* f *на интеграл от* φ. Формулы для прибли-

женного вычисления интегралов называются *квадратурными формулами (КФ)*.

6.1.2. Простейшие квадратурные формулы

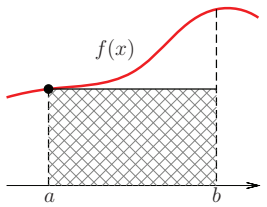
Формулы прямоугольников

Самые простые квадратурные формулы получаются при замене подынтегральной функции f на константу, а точнее на значение $f(x_0)$, где x_0 — некоторая точка из интервала интегрирования $[a, b]$. Площадь криволинейной трапеции, которой по определению равен искомый интеграл $\int_a^b f(x)dx$, приближенно равна площади прямоугольника $[a, b] \times [0, f(x_0)]$, т. е.

$$\int_a^b f(x)dx \approx f(x_0)(b-a).$$

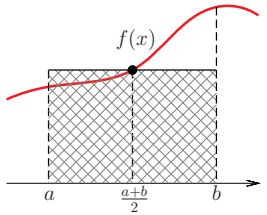
Наиболее известны три квадратурные формулы такого типа:

- *формула левых прямоугольников:*



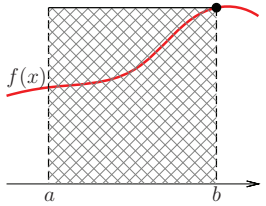
$$\int_a^b f(x)dx \approx (b-a)f(a); \qquad (6.1)$$

- *формула средних прямоугольников:*



$$\int_a^b f(x)dx \approx (b-a)f\left(\frac{a+b}{2}\right); \qquad (6.2)$$

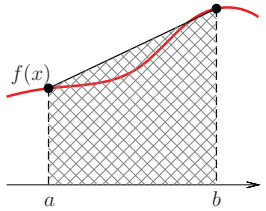
- *формула правых прямоугольников:*



$$\int_a^b f(x)dx \approx (b-a)f(b). \qquad (6.3)$$

Квадратурная формула трапеций

Если график подынтегральной функции приблизить секущей, проходящей через точки $f(a)$ и $f(b)$, что при интегрировании эквивалентно замене площади криволинейной трапеции на площадь трапеции обыкновенной, то в результате получится *квадратурная формула трапеций*:



$$\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b)}{2}. \qquad (6.4)$$

Эта формула есть не что иное, как школьная формула для вычисления площади трапеции.

6.1.3. Определение

Перейдем теперь от частных случаев к общему определению квадратурной формулы. Начнем с того, что запишем интеграл в более общем виде

$$S(f) = \int_a^b f(x)\rho(x)dx. \qquad (6.5)$$

Здесь ρ — некоторая неотрицательная весовая функция, которая может принимать нулевые значения лишь в конечном числе точек. Отображение $S : f \mapsto \int_a^b f(x)\rho(x)dx$ представляет собой линейный функционал.

Определение 6.1. На отрезке $[a, b]$ рассмотрим множество из $n+1$ попарно различных точек x_0, $x_1, \ldots x_n$. Формула для приближенного вычисления интеграла $S(f)$ вида

$$\int_a^b f(x)\rho(x)dx \approx \sum_{i=0}^{n} A_i f(x_i) \tag{6.6}$$

называется *квадратурной формулой*. Вещественные числа $\{A_i\}_{i=0}^n$ называются *коэффициентами квадратурной формулы*, а точки $\{x_i\}_{i=0}^n$ — ее *узлами*. Функционал, стоящий в правой части квадратурной формулы,

$$Q_n(f) = \sum_{i=0}^{n} A_i f(x_i),$$

называется *квадратурной суммой*.

Таким образом, любая квадратурная формула может быть кратко записана в виде

$$S(f) \approx Q_n(f).$$

6.1.4. Интерполяционные квадратурные формулы

Как уже говорилось ранее, традиционный способ построения квадратурных формул заключается в замене подынтегральной функции f на легко интегрируемую функцию φ, т. е.

$$Q_n(f) = S(\varphi).$$

Если φ является интерполяционным многочленом, то получаемая в результате квадратурная формула называется *интерполяционной*.

Определение 6.2. Квадратурная формула вида

$$S(f) \approx S(P_n),$$

где P_n — интерполяционный многочлен для f, построенный по узлам $\{x_i\}_{i=0}^n$, называется *интерполяционной квадратурной формулой*.

Запишем интерполяционную квадратурную формулу в каноническом виде (6.6), используя представление интерполяционного многочлена в форме Лагранжа:

$$P_n(x) = \sum_{i=0}^{n} \Lambda_i(x) f(x_i).$$

Здесь Λ_i — базисные функции (5.7). Тогда

$$S(P_n) = S\left(\sum_{i=0}^{n} \Lambda_i f(x_i)\right) = \sum_{i=0}^{n} S(\Lambda_i) f(x_i) = \sum_{i=0}^{n} A_i f(x_i) = Q_n(f).$$

Таким образом, коэффициенты *интерполяционной квадратурной формулы с узлами* $\{x_i\}_{i=0}^n$ *могут быть вычислены по правилу*

$$\begin{aligned} A_i &= S(\Lambda_i) = \int_a^b \Lambda_i(x)\rho(x)dx, \\ \Lambda_i(x) &= \prod_{j\neq i} \frac{x - x_j}{x_i - x_j}. \end{aligned} \tag{6.7}$$

Очевидно, что уже знакомые формулы прямоугольников и трапеций являются интерполяционными и соответствуют весовой функции $\rho(x) \equiv 1$.

$\triangleright_1$ Получите формулу трапеций, используя правило (6.7).

6.2. Остаток квадратурных формул

Определение 6.3. Остатком квадратурной формулы $S(f) \approx Q_n(f)$ называется функционал

$$R_n(f) = S(f) - Q_n(f), \tag{6.8}$$

или, в развернутой форме,

$$R_n(f) = \int_a^b f(x)\rho(x)dx - \sum_{i=0}^{n} A_i f(x_i).$$

Величина остатка — естественная характеристика точности квадратурной формулы.

6.2.1. Остаток интерполяционных квадратурных формул

Имея выражение для остатка полиномиального интерполирования

$$r_n = f - P_n,$$

где P_n — интерполяционный многочлен для f, легко получить выражение для остатка интерполяционных квадратур. Действительно:

$$R_n(f) = S(f) - Q_n(f) = S(f) - S(P_n) = S(f - P_n) = S(r_n),$$

т. е. *остаток интерполяционной квадратурной формулы равен интегралу от остатка интерполяции*. Отсюда, используя формулу (5.28), которая имеет вид

$$r_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x),$$

для достаточно гладких f получим

$$R_n(f) = \frac{1}{(n+1)!}\int\limits_a^b f^{(n+1)}(\xi)\omega_{n+1}(x)\rho(x)dx. \tag{6.9}$$

Напомним, что здесь $\omega_{n+1}(x) = (x - x_0)\dots(x - x_n)$, а величина ξ зависит от x, поэтому выносить $f^{(n+1)}(\xi)$ за знак интеграла, вообще говоря, нельзя. Однако в некоторых случаях это возможно. Например, если многочлен ω_{n+1} сохраняет знак на $[a, b]$, то применима одна из теорем о среднем для интеграла Римана.

Теорема 6.1 (о среднем)**.** *Если функция f непрерывна на отрезке $[a, b]$, а функция g интегрируема и знакопостоянна на $[a, b]$, то существует такая точка $\eta \in [a, b]$, что*

$$\int\limits_a^b f(x)g(x)dx = f(\eta)\int\limits_a^b g(x)dx.$$

Таким образом, если многочлен ω_{n+1} знакопостоянен, то по теореме о среднем из (6.9) получим

$$R_n(f) = \frac{f^{(n+1)}(\eta)}{(n+1)!}\int\limits_a^b \omega_{n+1}(x)\rho(x)dx, \quad \eta \in [a, b]. \tag{6.10}$$

Здесь также использован тот факт, что весовая функция ρ знакопостоянна по условию.

6.2.2. Остаток простейших квадратурных формул

Применим полученные формулы для исследования остатка рассмотренных в подп. 6.1.2 простейших квадратурных формул. Все они являются интерполяционными с единичной весовой функцией. Начнем с формул прямоугольников.

Остаток формул левых и правых прямоугольников

Формула (6.1) – это интерполяционная квадратурная формула с $n = 0$ (подынтегральная функция приближается многочленом нулевой степени) и $x_0 = a$:

$$S(f) \approx Q_0(f) = (b-a)f(a).$$

В силу знакопостоянства многочлена $\omega_1(x) = x - a$ остаток этой квадратурной формулы может быть представлен в виде (6.10). В результате получается

$$R_0(f) = \frac{(b-a)^2}{2} f'(\eta), \quad \eta \in [a,b]. \tag{6.11}$$

$\triangleright_1$ Запишите остаток квадратурной формулы правых прямоугольников (6.3).

Остаток формулы средних прямоугольников

Формула (6.2) наиболее интересна. Перепишем ее в виде

$$S(f) \approx Q_0(f) = (b-a)f\Big(\frac{a+b}{2}\Big).$$

В этом случае многочлен $\omega_1 = x - (a+b)/2$ уже не сохраняет знак на отрезке интегрирования, т. е. формула (6.10) не работает. Однако хорошее представление для остатка можно получить следующим образом. Предположим, что f'' существует и непрерывна. Тогда по формуле Тейлора имеем

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(\xi)}{2}(x - x_0)^2, \quad \xi \in [a,b].$$

Вычислим $R_0(f) = S(f - P_0)$, где $P_0(x) \equiv f(x_0)$, $x_0 = (a+b)/2$:

$$S(f-P_0) = \int_a^b \Big(f'(x_0)(x-x_0) + \frac{f''(\xi)}{2}(x-x_0)^2 \Big) dx =$$

$$= \int_a^b \frac{f''(\xi)}{2}(x-x_0)^2 dx = \frac{f''(\eta)}{2} \int_a^b (x-x_0)^2 dx,$$

откуда

$$R_0(f) = \frac{(b-a)^3}{24} f''(\eta). \tag{6.12}$$

$\triangleright_2$ Какой вид будет иметь остаток формулы средних прямоугольников в случае, когда f'' не существует, т. е. когда работает только формула (6.9)?

Остаток формулы трапеций

Для формулы трапеций (6.4), которую запишем в виде

$$S(f) \approx Q_1(f) = \frac{b-a}{2} f(a) + \frac{b-a}{2} f(b),$$

имеем $n = 1$, $x_0 = a$ и $x_1 = b$. Многочлен $\omega_2(x) = (x-a)(x-b)$ знакопостоянен на $[a, b]$, поэтому снова имеем право применить формулу (6.10) для остатка. В результате получим

$$R_1(f) = -\frac{(b-a)^3}{12} f''(\eta), \quad \eta \in [a, b]. \tag{6.13}$$

$\triangleright_3$ Выведите эту формулу. При взятии интеграла будет удобно сделать замену переменных $x = a + (b-a)t$.

Если сравнить формулы (6.12) и (6.13), видно, что формула средних прямоугольников, которая требует лишь одного вычисления функции f, точнее формулы трапеций при условии достаточной гладкости f.

6.3. Квадратурные формулы Ньютона – Котеса

6.3.1. Алгебраическая степень точности

Помимо величины остатка существует еще одна характеристика квадратурных формул — алгебраическая степень точности.

Определение 6.4. Рассмотрим квадратурную формулу $S(f) \approx Q_n(f)$. Если для любого многочлена p степени m и ниже выполняется тождество

$$S(p) = Q_n(p),$$

и при этом существует такой многочлен $\tilde{p}$ степени $m+1$, что

$$S(\tilde{p}) \neq Q_n(\tilde{p}),$$

то говорят, что алгебраическая степень точности (АСТ) такой формулы равна m.

Замечание 6.1. В силу линейности функционалов S и Q_n можно дать эквивалентное определение АСТ. Любой многочлен степени m является линейной комбинацией одночленов

$$\chi_j(x) = x^j, \quad j = \overline{0, m},$$

поэтому квадратурная формула имеет АСТ m, если и только если она точно интегрирует все многочлены χ_j:

$$S(\chi_j) = Q_n(\chi_j) \quad \forall\, j = \overline{0, m},$$

и при этом

$$S(\chi_{m+1}) \neq Q_n(\chi_{m+1}).$$

$\triangleright_1$ Докажите это.

Поскольку интерполяционные квадратурные формулы получаются заменой подынтегральной функции на интерполяционный многочлен степени n, а все многочлены степени $m \leqslant n$ при этом будут интерполироваться точно, получается следующий простой, но важный результат.

Теорема 6.2. *Квадратурная формула с $(n+1)$ узлом является интерполяционной, если и только если она имеет алгебраическую степень точности не менее n.*

Доказательство. Если КФ является интерполяционной, то $Q_n(f) = S(P_n)$, где P_n — интерполяционный многочлен степени n для f. Если f является многочленом степени $m \leqslant n$, то имеем $P_n = f$ и $Q_n(f) = S(f)$ — формула точна для всех многочленов степени n и ниже.

Пусть квадратурная формула

$$S(f) \approx \sum_i A_i f(x_i)$$

имеет АСТ не менее n. Покажем, что ее коэффициенты находятся по формуле (6.7): $A_i = S(\Lambda_i)$, где Λ_i — фундаментальная базисная функция Лагранжа (5.7). Это и будет означать, что формула интерполяционная. Нет ничего проще: так как $\deg \Lambda_i = n$, имеем

$$S(\Lambda_i) = Q_n(\Lambda_i) = \sum_{j=0}^{n} A_j \Lambda_i(x_j) = A_i. \qquad \square$$

$\triangleright_2$ Найдите АСТ всех трех формул прямоугольников и формулы трапеций.

6.3.2. Симметричные квадратурные формулы

Определение 6.5. Квадратурную формулу

$$S(f) \approx \sum_{i=0}^{n} A_i f(x_i)$$

будем называть *симметричной*, если она имеет симметричные узлы относительно середины отрезка $[a, b]$,

$$x_i - a = b - x_{n-i}, \quad \forall i = \overline{0, n},$$

а также симметричные коэффициенты:

$$A_i = A_{n-i}, \quad \forall i = \overline{0, n}.$$

Лемма 6.1. *Пусть весовая функция* ρ *является четной относительно середины* $[a, b]$*, а квадратурная формула* $S(f) \approx Q_n(f)$ *— симметричной. Тогда для всякой нечетной относительно* $(a + b)/2$ *функции* f*,*

$$f(x) = -f(a + b - x),$$

справедливо

$$S(f) = Q_n(f).$$

Доказательство. В силу нечетности f имеем $S(f) = 0$, а симметрия узлов дает $f(x_i) = -f(x_{n-i})$. Покажем, что $Q_n(f) = 0$:

$$Q_n(f) = \sum_{i=0}^{n} A_i f(x_i) = -\sum_{i=0}^{n} A_{n-i} f(x_{n-i}) = -Q_n(f),$$

откуда сразу следует утверждение леммы. $\square$

Теорема 6.3 (о повышенной АСТ симметричных КФ). *Пусть весовая функция* ρ *является четной относительно середины* $[a,b]$, *а квадратурная формула* $S(f) \approx Q_n(f)$ *точна для всех многочленов степени* $2M$ *и ниже,* $M \in \mathbb{Z}$. *Если к тому же эта КФ симметрична, то ее алгебраическая степень точности равна как минимум* $2M+1$.

Доказательство. Нам достаточно показать, что $Q_n(P) = S(P)$ для любого многочлена P степени $2M+1$. Итак, пусть $\deg P = 2M+1$. Его старший коэффициент обозначим α и рассмотрим многочлен

$$U(x) = \alpha\Big(x - \frac{a+b}{2}\Big)^{2M+1}.$$

В силу нечетности этого многочлена $S(U) = 0$, а также в силу симметричности по лемме 6.1 имеем $Q_n(U) = 0$.

Осталось представить P в виде

$$P = U + \tilde{P},$$

где $\deg \tilde{P} = \deg(P-U) \leqslant 2M$. Тогда по условию $S(\tilde{P}) = Q_n(\tilde{P})$ и

$$S(P) = S(U) + S(\tilde{P}) = S(\tilde{P}) = Q_n(\tilde{P}) = Q_n(\tilde{P}) + Q_n(U) = Q_n(P). \ \square$$

$\triangleright_3$ Укажите, в каком месте доказательства используется четность ρ.

6.3.3. Симметрия интерполяционных квадратурных формул

Теорема 6.4. *Пусть функция* ρ *является четной относительно середины отрезка* $[a,b]$. *Тогда если узлы* $\{x_i\}$ *симметричны, то соответствующая им интерполяционная квадратурная формула с весом* ρ *является симметричной.*

Доказательство. Нам нужно доказать, что если узлы удовлетворяют свойству $x_i - a = b - x_{n-i}$, то коэффициенты соответствующей КФ, которые вычисляются по формуле (6.7):

$$A_i = \int_a^b \Lambda_i(x)\rho(x)dx,$$

обладают свойством $A_i = A_{n-i}$. Доказательство основывается на факте попарной симметричности базисных функций Λ_i:

$$\Lambda_i(x) = \Lambda_{n-i}(a+b-x).$$

Докажем это:

$$\Lambda_{n-i}(a+b-x) = \prod_{j\neq n-i} \frac{a+b-x-x_j}{x_{n-i}-x_j} =$$
$$= \prod_{j\neq n-i} \frac{x_{n-j}-x}{(a+b-x_i)-(a+b-x_{n-j})} =$$
$$= \prod_{j\neq n-i} \frac{x-x_{n-j}}{x_i-x_{n-j}} = \prod_{j\neq i} \frac{x-x_j}{x_i-x_j} = \Lambda_i(x).$$

Осталось сделать замену переменных в интеграле:

$$A_i = \int_a^b \Lambda_i(x)\rho(x)dx = \int_a^b \Lambda_{n-i}(a+b-x)\rho(x)dx = [z = a+b-x] =$$
$$= -\int_b^a \Lambda_{n-i}(z)\rho(a+b-z)dz = \int_a^b \Lambda_{n-i}(z)\rho(z)dz = A_{n-i}. \qquad \square$$

Следствие 6.1. *Интерполяционная квадратурная формула с четной относительно середины $[a,b]$ весовой функцией ρ и симметричными узлами $\{x_i\}$ при $n = 2M$ (т. е. при нечетном количестве узлов) имеет алгебраическую степень точности не менее $2M+1$.*

Доказательство. Поскольку рассматриваемая формула интерполяционная, она точна для всех многочленов степени $n = 2M$ и ниже. По теореме 6.4 она также является симметричной. Значит, по теореме 6.3 АСТ этой формулы равна как минимум $2M+1$. $\square$

6.3.4. Квадратурные формулы Ньютона – Котеса

Определение 6.6. Интерполяционные квадратурные формулы для весовой функции $\rho \equiv 1$ по равноотстоящим узлам на отрезке $[a,b]$,

$$x_i = a + \frac{b-a}{n} i, \quad i = \overline{0,n},$$

называются *формулами Ньютона – Котеса*.

По определению и теореме 6.4 все формулы Ньютона – Котеса являются симметричными. С одной такой формулой мы уже знакомы: при $n = 1$ получается формула трапеций. Под формальное определение попадает также формула левых прямоугольников, хотя по смыслу больше подходит формула прямоугольников средних. Сейчас мы построим еще одну классическую квадратурную формулу Ньютона – Котеса.

Формула Симпсона

Квадратурная формула Симпсона — это формула Ньютона – Котеса с тремя узлами:

$$x_0 = a, \quad x_1 = \frac{a+b}{2}, \quad x_2 = b.$$

Иногда эту формулу также называют формулой парабол (подынтегральная функция приближается параболой). Для вычисления коэффициентов этой формулы можно воспользоваться определением (6.7), но мы сделаем это по-другому.

Начнем с того, что выведем формулу Симпсона для частного случая $[a, b] = [-1, 1]$. Мы знаем, что эта формула является симметричной, т. е. имеет вид

$$\int_{-1}^{1} f(x)dx = S(f) \approx \bar{Q}_2(f) = \bar{A}_0 f(-1) + \bar{A}_1 f(0) + \bar{A}_0 f(1).$$

Эта формула по построению точна для всех многочленов степени 2 и ниже, а значит при $f(x) = x^i$ для $i = 0, 1, 2$ будет иметь место тождество $S(f) = \bar{Q}_2(f)$. Эти три условия дают систему уравнений

$$\begin{cases} \bar{A}_0 + \bar{A}_1 + \bar{A}_0 = 2, \\ -\bar{A}_0 + \bar{A}_0 = 0, \\ \bar{A}_0 + \bar{A}_0 = \dfrac{2}{3}, \end{cases}$$

откуда легко находим

$$\bar{A}_0 = \frac{1}{3}, \quad \bar{A}_1 = \frac{4}{3}.$$

Мы получили формулу Симпсона для отрезка $[-1, 1]$:

$$\int_{-1}^{1} f(x)dx \approx \frac{1}{3} f(-1) + \frac{4}{3} f(0) + \frac{1}{3} f(1). \tag{6.14}$$

Для вывода формулы в общем случае теперь достаточно сделать замену переменной интегрирования:

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f\Big(\frac{a+b}{2} + \frac{b-a}{2}t\Big)dt.$$

Применяя к последнему интегралу формулу (6.14), окончательно получим *квадратурную формулу Симпсона*

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left(f(a) + 4f\Big(\frac{a+b}{2}\Big) + f(b)\right). \qquad (6.15)$$

$\triangleright_4$ Аналогичным образом выведите формулы Ньютона – Котеса с четырьмя и пятью узлами.

$\triangleright_5$ Определите АСТ формулы Симпсона и используя эту информацию получите выражение для ее остатка путем разложения в ряд Тейлора в точке $(a+b)/2$.

6.4. Квадратурные формулы Гаусса

6.4.1. Постановка задачи

Рассмотрим задачу построения квадратурных формул

$$S(f) = \int_a^b f(x)\rho(x)dx \approx \sum_{i=0}^{n} A_i f(x_i) = Q_n(f),$$

имеющих максимально возможную алгебраическую степень точности. Такие формулы называются *гауссовыми* или *квадратурными формулами наивысшей АСТ*. Как мы помним, по определению АСТ квадратурной формулы равна m, если

$$S(p) = Q_n(p) \quad \forall p \in \mathbb{P}_m \qquad (6.16а)$$

и

$$\exists\, \tilde{p} \in \mathbb{P}_{m+1}: \quad S(\tilde{p}) \neq Q_n(\tilde{p}). \qquad (6.16б)$$

Здесь и далее $\mathbb{P}_m$ — множество всех многочленов степени не выше m. Мы помним также, что в силу линейности для выполнения условия

(6.16а) необходимо и достаточно потребовать, чтобы квадратурная формула работала точно на базисе пространства $\mathbb{P}_m$. Проще всего выбрать степенной базис, поэтому условия (6.16) эквивалентны (в частности) следующим:

$$\sum_{i=0}^{n} A_i x_i^j = \int_a^b x^j \rho(x)dx, \quad j = \overline{0,m}; \tag{6.17а}$$

$$\sum_{i=0}^{n} A_i x_i^{m+1} \neq \int_a^b x^{m+1} \rho(x)dx. \tag{6.17б}$$

В дальнейшем мы будем рассматривать только условия (6.17а).

Какова же будет максимально возможная АСТ для данного n? Условия (6.17) представляют собой систему из $(m+1)$ нелинейных уравнений относительно $(2n+2)$ неизвестных $\{x_i\}_{i=0}^n$, $\{A_i\}_{i=0}^n$. Поэтому можно по крайней мере надеяться, что эта система будет иметь решение, когда количество уравнений и неизвестных совпадает, т. е. когда

$$m = 2n + 1.$$

Как будет показано далее, это и есть максимально возможная АСТ.

6.4.2. Построение квадратурных формул Гаусса

Итак, построение квадратурных формул Гаусса заключается в нахождении узлов $\{x_i\}$ и коэффициентов $\{A_i\}$, удовлетворяющих условиям (6.17а) для $m = 2n + 1$. Самый очевидный способ их нахождения — это непосредственное решение указанной системы. Для каждого n мы будем иметь $2n + 2$ нелинейных уравнений. Случай $n = 0$ прост:

$$\begin{cases} A_0 = \int_a^b \rho(x)dx, \\ A_0 x_0 = \int_a^b x\rho(x)dx, \end{cases}$$

откуда, в частности, при $\rho(x) \equiv 1$ получим формулу средних прямоугольников. При $n = 1$ систему можно решить относительно легко, а при больших значениях n задача становится практически неподъемной.

$\triangleright_1$ Решите систему при $n = 1$, $[a, b] = [-1, 1]$ и $\rho(x) \equiv 1$.

Следующее тривиальное следствие теоремы 6.2 существенно упрощает задачу: *если квадратурная формула имеет наивысшую АСТ, то*

она является интерполяционной, т. е. ее коэффициенты имеют вид

$$A_i = \int\limits_a^b \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \rho(x) dx, \quad i = \overline{0, n}.$$

$\triangleright_2$ Докажите.

Таким образом, задача *упростилась в два раза*: достаточно найти узлы формулы Гаусса, а коэффициенты можно определить по формуле, приведенной выше. О том, какими должны быть узлы формулы, говорит следующая далее теорема. Она является наиболее важным результатом этого пункта, и для ее формулировки понадобится определение скалярного произведения и ортогональности функций.

Определение 6.7. Скалярным произведением интегрируемых с квадратом функций u и v называется выражение

$$(u, v) = \int\limits_a^b u(x) v(x) \rho(x) dx,$$

Здесь ρ — заданная неотрицательная на $[a, b]$ весовая функция. Если $(u, v) = 0$, то говорят, что функции u и v являются *ортогональными* по весу ρ.

Теорема 6.5 (К. Ф. Гаусс)**.** *Для того чтобы квадратурная формула с $(n+1)$ узлами была точной для любых алгебраических многочленов степени $(2n+1)$ и ниже, необходимо и достаточно, чтобы:*

1) она была интерполяционной;

2) многочлен $\omega_{n+1}(x) = (x - x_0) \dots (x - x_n)$ был ортогонален по весу ρ на отрезке $[a, b]$ ко всем многочленам степени n и ниже:

$$(\omega_{n+1}, p) = \int\limits_a^b \omega_{n+1}(x) p(x) \rho(x) dx = 0 \quad \forall p \in \mathbb{P}_n. \tag{6.18}$$

Доказательство. Пусть формула имеет АСТ $2n+1$. Значит, она будет точна для всех многочленов вида $q(x) = \omega_{n+1}(x) p(x)$, $p \in \mathbb{P}_n$, так как $\deg q \leqslant 2n+1$. С учетом того, что $\omega_{n+1}(x_i) = 0$ при всех i, получим

$$S(q) = (\omega_{n+1}, p) = Q_n(q) = \sum_{i=0}^{n} A_i \omega_{n+1}(x_i) p(x_i) = 0.$$

То, что формула интерполяционная, было доказано ранее.

Теперь пусть верно (6.18). Возьмем любой $q \in \mathbb{P}_{2n+1}$ и разделим его на ω_{n+1} с остатком:

$$q(x) = \omega_{n+1}(x)p(x) + r(x),$$

где $\deg p \leqslant n$, $\deg r \leqslant n$. Тогда с учетом $q(x_i) = r(x_i)$ имеем

$$S(q) = (\omega_{n+1}, p) + S(r) = S(r) = \sum_{i=0}^{n} A_i r(x_i) = \sum_{i=0}^{n} A_i q(x_i),$$

т. е. формула точна для любого $q \in \mathbb{P}_{2n+1}$. □

Таким образом, *узлами квадратурной формулы Гаусса являются корни многочлена, ортогонального по весу* ρ *всем многочленам степени* n *и ниже*. Многочлен, удовлетворяющий этим условиям, в дальнейшем будем обозначать ω^*_{n+1}.

Для строгости доказательства полученного результата нужно еще: а) показать, что нельзя достичь АСТ, большей $2n+1$; б) доказать существование узлов формул Гаусса на отрезке $[a, b]$ (вдруг они вообще комплексные?).

Теорема 6.6. *Если весовая функция* ρ *знакопостоянна на отрезке* $[a, b]$*, то не существует квадратурных формул с* $(n+1)$ *узлами, имеющих АСТ* $2n+2$.

Доказательство. Предположим существование такой формулы. Пусть $\{x_i\}_{i=0}^n$ — ее узлы. Рассмотрим многочлен

$$\nu(x) = (x - x_0)^2 \dots (x - x_n)^2.$$

Для него очевидно имеем $S(\nu) \neq 0$, так как подынтегральное выражение знакопостоянно и не равно нулю тождественно. С другой стороны, имеем $Q_n(f) = \sum_{i=0}^n A_i \nu(x_i) = 0$. Полученное противоречие доказывает теорему. □

Теорема 6.7. *Если весовая функция* ρ *сохраняет знак на отрезке* $[a, b]$*, то многочлен* ω^*_{n+1} *степени* $(n+1)$*, ортогональный на данном отрезке по весу* ρ *ко всем многочленам степени* n *и ниже, существует и единствен для любого фиксированного* n*. При этом все его корни действительны, различны и лежат внутри* $[a, b]$.

Замечание 6.2. Рассмотрим множество многочленов $\{\omega_i^*\}_{i=1}^{\infty}$. Поскольку по определению $(\omega_{n+1}^*, p) = 0$ для всех $p \in \mathbb{P}_n$, то имеем

$$(\omega_i^*, \omega_j^*) = 0, \quad \text{если} \quad i \neq j,$$

т. е. множество $\{\omega_i^*\}$ представляет собой *систему ортогональных многочленов* (по весу ρ). Такие системы играют важную роль во многих приложениях. Поэтому результат главной теоремы 6.5 можно сформулировать и так: *квадратурная формула с* $(n+1)$ *узлами точна для любых алгебраических многочленов степени* $(2n+1)$ *и ниже тогда и только тогда, когда она интерполяционная, а ее узлы являются корнями ортогонального по весу* ρ *многочлена степени* $n+1$.

6.4.3. Классические квадратурные формулы Гаусса

Единичная весовая функция

Рассмотрим случай единичной весовой функции на отрезке $[-1, 1]$, т. е. квадратурные формулы наивысшей АСТ для интеграла общего вида $\int_a^b f(x)dx$. Такие формулы иногда называют формулами Гаусса – Лежандра, так как их узлами являются корни ортогональных на $[-1, 1]$ с единичным весом *многочленов Лежандра*. Эти многочлены можно определить явной формулой

$$L_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} \left((x^2 - 1)^n \right) \tag{6.19}$$

или рекуррентным соотношением

$$L_{n+1}(x) = \frac{2n+1}{n+1} x L_n(x) - \frac{n}{n+1} L_{n-1}(x), \tag{6.20}$$
$$L_0(x) = 1, \quad L_1(x) = x.$$

$\triangleright_3$ Постройте квадратурные формулы Гаусса – Лежандра с двумя и тремя узлами.

Для применения квадратурных формул Гаусса – Лежандра на произвольном отрезке $[a, b]$ достаточно сделать уже известную нам замену переменных

$$x = \frac{a+b}{2} + \frac{b-a}{2} t, \quad t \in [-1, 1],$$

как это было при выводе формулы Симпсона. Покажем подробно, как это делается. Пусть имеется формула Гаусса вида

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=0}^{n} A_i f(x_i).$$

Тогда

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} \underbrace{f\Big(\frac{a+b}{2}+\frac{b-a}{2}t\Big)}_{g(t)}dt = \frac{b-a}{2}\int_{-1}^{1} g(t)dt \approx$$

$$\approx \frac{b-a}{2}\sum_{i=0}^{n} A_i g(x_i) = \frac{b-a}{2}\sum_{i=0}^{n} A_i f\Big(\frac{a+b}{2}+\frac{b-a}{2}x_i\Big).$$

Другие классические формулы

Для некоторых весовых функций известны явные формулы, определяющие ортогональные многочлены, а также явные рекуррентные соотношения. Для этих многочленов давно составлены таблицы коэффициентов и узлов соответствующих формул наивысшей АСТ. Эти коэффициенты выводить слишком долго, а запоминать бессмысленно, но нужно о них знать, чтобы при необходимости найти в литературе [7] (см. таблицу).

Классические системы ортогональных многочленов

$\rho(x)$	Отрезок	Название
$\frac{1}{\sqrt{1-x^2}}$	$[-1,1]$	Многочлены Чебышева
$x^{\alpha}e^{-x}$	$[0,+\infty)$	Многочлены Лагерра
e^{-x^2}	$(-\infty,+\infty)$	Многочлены Эрмита
$(1-x)^{\alpha}(1+x)^{\beta}$	$[-1,1]$	Многочлены Якоби

В частности, знакомые нам многочлены Чебышева

$$T_n(x) = \cos(n \arccos x),$$

оказывается, ортогональны на отрезке $[-1, 1]$ с весом $\rho(x) = (1-x^2)^{-1/2}$. Следовательно, узлы квадратурной формулы Гаусса – Чебышева, которая имеет наивысшую АСТ для интегралов с этой весовой функцией, являются корнями многочленов T_n (чебышевскими узлами):

$$x_i = \cos\frac{\pi(2i+1)}{2n+2}, \quad i = \overline{0,n}.$$

Замечателен тот факт, что коэффициенты таких квадратурных формул при данном n будут равны между собой:

$$A_0 = \ldots = A_n = \frac{\pi}{n+1}.$$

Другие весовые функции

Возникает вопрос: каким образом строить гауссовы квадратурные формулы в общем случае, когда неизвестны соответствующие системы ортогональных многочленов? Ответ таков: нужные системы ортогональных многочленов можно построить по специальным рекуррентным соотношениям, рассмотрение которых выходит за рамки нашего курса.

Однако при малых n можно использовать следующий простой подход. Он позволяет для данного n найти непосредственно многочлен ω^*_{n+1} в явном виде

$$\omega^*_{n+1}(x) = x^{n+1} + c_n x^n + c_{n-1}x^{n-1} + \ldots + c_1 x + c_0. \tag{6.21}$$

Воспользуемся определением: ω^*_{n+1} должен быть ортогонален всем многочленам степени n и ниже, а для этого необходимо и достаточно потребовать его ортогональности одночленам $\chi_i(x) = x^i$:

$$\int_a^b \omega^*_{n+1}(x)x^i\rho(x)dx = 0, \quad i = \overline{0,n}.$$

Подставляя сюда представление (6.21), получим СЛАУ вида

$$\gamma_{i0}c_0 + \gamma_{i1}c_1 + \ldots + \gamma_{i,n-1}c_{n-1} + \gamma_{i,n}c_n = -g_i, \quad i = \overline{0,n}, \tag{6.22}$$

где

$$\gamma_{ij} = \int_a^b x^{i+j}\rho(x)dx, \quad g_i = \int_a^b x^{n+i+1}\rho(x)dx.$$

$\triangleright_4$ Выведите эти формулы.

После того как найден многочлен ω_{n+1}^*, остается найти его корни и вычислить коэффициенты квадратурной формулы по правилу (6.7).

6.5. Составные квадратурные формулы

6.5.1. Введение

Пусть $\rho \equiv 1$. Рассмотрим задачу вычисления интеграла

$$S(f) = \int_a^b f(x)dx$$

с некоторой заданной точностью. Для этого разобьем отрезок $[a, b]$ на N равных частей точками $\{\xi_k\}_{k=0}^N$,

$$\xi_k = a + kh, \quad k = \overline{0, N}, \quad h = \frac{b-a}{N},$$

и получим

$$S(f) = \sum_{k=1}^N S_k(f), \qquad S_k(f) = \int_{\xi_{k-1}}^{\xi_k} f(x)dx.$$

Приближая каждое слагаемое какой-нибудь простой квадратурной формулой

$$S_k(f) \approx Q_{n,k}(f),$$

получим *составную квадратурную формулу* для всего интеграла, которую будем обозначать как

$$S(f) \approx Q_n^N(f) = \sum_{k=1}^N Q_{n,k}(f).$$

6.5.2. Простейшие составные формулы

Возьмем в качестве базовой, например, формулу средних прямоугольников (6.2):

$$\int_{\xi_{k-1}}^{\xi_k} f(x)dx \approx (\xi_k - \xi_{k-1}) f\Big(\frac{\xi_{k-1} + \xi_k}{2}\Big).$$

С учетом того, что $\xi_k = a + kh$, после суммирования всех частей получим *составную формулу средних прямоугольников* (рис. 6.1):

$$\int_a^b f(x)dx \approx h \sum_{k=1}^{N} f\left(a + (k - \tfrac{1}{2})h\right). \tag{6.23}$$

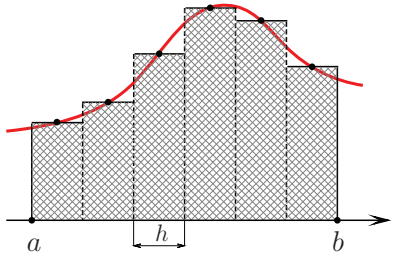


Рис. 6.1. Составная квадратурная формула средних прямоугольников

Аналогично строится *составная формула трапеций* (рис. 6.2):

$$\int_{\xi_{k-1}}^{\xi_k} f(x)dx \approx \frac{h}{2}(f(\xi_{k-1}) + f(\xi_k)),$$

откуда

$$\int_a^b f(x)dx \approx \frac{h}{2}\Big(f(a) + 2\sum_{k=1}^{N-1} f(a + kh) + f(b)\Big). \tag{6.24}$$

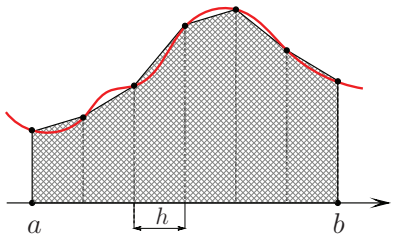


Рис. 6.2. Составная квадратурная формула трапеций

$\triangleright_1$ Постройте составные формулы левых, правых прямоугольников и Симпсона.

6.5.3. Остаток составных квадратурных формул

Теперь наша задача — вычислить остаток составных формул, который обозначим

$$R_n^N(f) = S(f) - Q_n^N(f).$$

Согласно введенным в начале пункта обозначениям, имеем

$$R_n^N(f) = S(f) - Q_n^N(f) = \sum_{k=1}^{N} S_k(f) - \sum_{k=1}^{N} Q_{n,k}(f) =$$
$$= \sum_{k=1}^{N} \Big(S_k(f) - Q_{n,k}(f) \Big) = \sum_{k=1}^{N} R_{n,k}(f).$$

Таким образом, как нетрудно было догадаться сразу, *остаток составной квадратурной формулы равен сумме остатков ее составных частей*.

Предположим, что для каждой квадратурной суммы $Q_{n,k}$ имеет место представление остатка в виде

$$R_{n,k}(f) = Ch^l f^{(m)}(\eta_k),$$

где $\eta_k \in [\xi_{k-1}, \xi_k]$; C — известная константа; l и m — фиксированные натуральные числа. Как мы помним, такая форма остатка справедлива, в частности, для всех уже рассмотренных нами интерполяционных формул (6.11), (6.12), (6.13). Тогда для всего остатка получаем очевидное представление

$$R_n^N(f) = \sum_{k=1}^{N} R_{n,k}(f) = Ch^l \sum_{k=1}^{N} f^{(m)}(\eta_k).$$

Эту формулу можно упростить, если предположить, что $f^{(m)}$ непрерывна на $[a, b]$. В этом случае на этом отрезке она достигает своих экстремальных значений $y_{\min}$ и $y_{\max}$. Тогда

$$y_{\min} \leqslant \frac{1}{N} \sum_{k=1}^{N} f^{(m)}(\eta_k) \leqslant y_{\max}.$$

По теореме о промежуточном значении, в силу непрерывности $f^{(m)}$ на отрезке $[a, b]$ эта функция «проходит» все значения от $y_{\min}$ до $y_{\max}$, а в

частности, существует такая точка $\eta \in [a, b]$, что

$$f^{(m)}(\eta) = \frac{1}{N} \sum_{k=1}^{N} f^{(m)}(\eta_k). \tag{6.25}$$

Следовательно, формула для остатка составной КФ примет вид

$$R_n^N(f) = Ch^l N f^{(m)}(\eta) \tag{6.26а}$$

или, с учетом $N = (b - a)/h$,

$$R_n^N(f) = C(b - a)h^{l-1} f^{(m)}(\eta). \tag{6.26а}$$

Замечание 6.3. Напомним, что здесь η — некоторая точка из отрезка $[a, b]$, причем эта точка зависит от разбиения, т. е. для различных N эти точки будут разными (η зависит от h).

Отсюда для составной формулы средних прямоугольников (6.23) из (6.12) получаем остаток

$$R_0^N(f) = \frac{b - a}{24} h^2 f''(\eta), \tag{6.27}$$

а для составной формулы трапеций (6.24) из (6.13) имеем

$$R_1^N(f) = -\frac{b - a}{12} h^2 f''(\eta). \tag{6.28}$$

Имея формулы такого рода, теоретически можно заранее оценить необходимое число отрезков разбиения N, которое следует взять для вычисления интеграла с требуемой точностью. Для этого, естественно, нужно располагать информацией о функции $f^{(m)}$ на отрезке $[a, b]$. На практике такая информация доступна не всегда, поэтому в универсальных программах численного интегрирования используется несколько иной способ представления погрешности составных формул.

6.5.4. Выделение главной части погрешности

Рассмотрим формулу остатка составной формулы прямоугольников (6.27). Она говорит, что остаток имеет вид $O(h^2)$, если f'' существует и непрерывна на $[a, b]$:

$$S(f) = Q_0^N(f) + O(h^2). \tag{6.29}$$

Если же f имеет производные более высоких порядков, то из этого остатка можно выделить *главную часть*, которая, в отличие от $f''(\eta)$, не зависит от h и может быть вычислена. Покажем, как это делается на примере той же составной формулы средних прямоугольников (6.23).

Рассмотрим точку $z_k = (\xi_{k-1} + \xi_k)/2$ — середину отрезка $[\xi_{k-1}, \xi_k]$. Предположим, что f четырежды непрерывно дифференцируема и представим ее формулой Тейлора в точке z_k:

$$S_k(f) = \int\limits_{\xi_{k-1}}^{\xi_k} f(x)dx = \int\limits_{\xi_{k-1}}^{\xi_k} \Big[f(z_k) + (x - z_k)f'(z_k) + \\ + \frac{(x - z_k)^2}{2} f''(z_k) + \frac{(x - z_k)^3}{6} f'''(z_k) + \frac{(x - z_k)^4}{24} f^{(4)}(\zeta_k) \Big] dx.$$

Здесь ζ_k — некоторая точка из $[\xi_{k-1}, \xi_k]$. Так как интеграл от нечетных степеней $(x - z_k)$ равен нулю, после интегрирования каждого слагаемого и применения теоремы о среднем к последнему из них получим

$$S_k(f) = hf(z_k) + \frac{h^3}{24} f''(z_k) + \frac{h^5}{1920} f^{(4)}(\eta_k),$$

η_k — точка из $[\xi_{k-1}, \xi_k]$, отличная, вообще говоря, от ζ_k. Просуммируем теперь все $S_k(f)$:

$$S(f) = \sum_{k=1}^{N} S_k(f) = h \sum_{k=1}^{N} f(z_k) + \frac{h^3}{24} \sum_{k=1}^{N} f''(z_k) + \frac{h^5}{1920} \sum_{k=1}^{N} f^{(4)}(\eta_k).$$

Итак, наш интеграл состоит из трех слагаемых. Рассмотрим отдельно каждое из них. Первое, $h \sum_{k=1}^{N} f(z_k) = Q_0^N(f)$, — это как раз составная квадратурная сумма средних прямоугольников (6.23). Второе,

$$\frac{h^3}{24} \sum_{k=1}^{N} f''(z_k) = \frac{h^2}{24} \left(h \sum_{k=1}^{N} f''(z_k) \right),$$

содержит составную сумму средних прямоугольников для f'', так что с учетом (6.29) это слагаемое примет вид

$$\frac{h^3}{24} \sum_{k=1}^{N} f''(z_k) = \frac{h^2}{24} \left(\int\limits_a^b f''(x)dx + O(h^2) \right).$$

В третьем слагаемом применим формулу (6.25) и получим

$$\frac{h^5}{1920}\sum_{k=1}^{N} f^{(4)}(\eta_k) = \frac{h^5}{1920} N f^{(4)}(\eta) = \frac{h^4}{1920}(b-a) f^{(4)}(\eta).$$

Собирая все вместе, имеем

$$S(f) = Q_0^N(f) + \frac{h^2}{24}\int_a^b f''(x)dx + O(h^4),$$

или

$$R_0^N(f) = \frac{f'(a) - f'(b)}{24} h^2 + O(h^4). \tag{6.30}$$

Таким образом, мы выделили главную часть остатка составной квадратурной формулы средних прямоугольников: она равна

$$\frac{f'(a) - f'(b)}{24} h^2.$$

Еще раз обратим внимание, что константа при h^2, как и было обещано, зависит только от f и, в принципе, может быть вычислена. Эту главную часть при желании можно добавить к исходной квадратурной сумме и получить уточненное значение интеграла $S(f)$. Аналогичную операцию выделения главной части можно провести и с другими составными квадратурными формулами.

6.5.5. Практическая оценка погрешности: правило Рунге

Составные квадратурные формулы по построению представляют собой инструмент, позволяющий регулировать точность вычисления интеграла за счет увеличения числа отрезков разбиения N (уменьшения шага h). Для вычислений на практике нужно уметь выбирать такое N, которое бы позволило достичь требуемой точности ε, и в то же время не приводило к избыточным вычислениям (не было слишком велико). Один из возможных способов реализации этих требований называется *правилом Рунге* или *методом двойного пересчета*.

Пусть $Q_n^N = Q^N$ — некоторая составная квадратурная сумма. Предположим, что для остатка соответствующей квадратурной формулы справедливо представление, аналогичное (6.30):

$$R^N(f) = S(f) - Q^N(f) = Kh^p + o(h^p), \quad h = \frac{b-a}{N}, \tag{6.31}$$

где константа K не зависит от h. Здесь Kh^p представляет собой главную часть погрешности, которую можно найти, если известно значение константы K.

Вычислим приближенное значение интеграла $S(f)$ по данной формуле *дважды*, на двух разных разбиениях с числом отрезков N_1 и N_2, $N_2 > N_1$. Полученные приближения обозначим соответственно

$$q_1 = Q^{N_1}(f) \quad \text{и} \quad q_2 = Q^{N_2}(f).$$

Обозначив $h_i = (b-a)/N_i$, согласно (6.31) можно записать приближенные равенства

$$S(f) - q_1 \approx Kh_1^p,$$
$$S(f) - q_2 \approx Kh_2^p.$$

Отсюда, исключая $S(f)$, имеем

$$K \approx \frac{q_2 - q_1}{h_1^p - h_2^p} =: \tilde{K},$$

а это очень важная информация. Во-первых, с ее помощью можно получить оценку погрешности для обеих квадратурных сумм:

$$R^{N_i}(f) \approx Kh_i^p \approx \tilde{K}h_i^p = \tilde{R}_i,$$

где

$$\tilde{R}_i = \frac{(q_2 - q_1)h_i^p}{h_1^p - h_2^p}. \tag{6.32}$$

Во-вторых, имея $\tilde{R}_i$, мы можем уточнить значения q_i:

$$S(f) = q_i + R^{N_i}(f) \approx q_i + \tilde{R}_i. \tag{6.33}$$

Для получения максимальной точности лучше всего уточнять q_2.

В-третьих, с помощью $\tilde{K}$ можно оценить оптимальную величину шага h^* для достижения точности ε. Делается это с помощью очевидного требования

$$|R^N(f)| \approx |\tilde{K}|h^p \leqslant \varepsilon,$$

из которого получим

$$h \leqslant \left(\frac{\varepsilon}{|\tilde{K}|}\right)^{1/p} = \left(\frac{\varepsilon}{|\tilde{R}_i|}\right)^{1/p} h_i = h^*,$$

или

$$N \geqslant N^* = \frac{b-a}{h^*} = \left(\frac{|\tilde{R}_i|}{\varepsilon}\right)^{1/p} N_i. \tag{6.34}$$

Практические рекомендации. Теперь поговорим о том, как все это реализуется на практике. Простейший алгоритм выглядит следующим образом:

1) выбираем квадратурную формулу Q^N, N_1 и N_2, точность ε;
2) $q_1 \leftarrow Q^{N_1}(f), \quad q_2 \leftarrow Q^{N_2}(f)$;
3) по формуле (6.32) вычисляем оценку погрешности для приближения q_2:

$$\tilde{R} \leftarrow \frac{(q_2 - q_1)h_2^p}{h_1^p - h_2^p}.$$

Как мы помним, здесь p — показатель точности КФ, определяемый (6.31), $h_i = (b-a)/N_i$;
4) если $|\tilde{R}| \leqslant \varepsilon$, полагаем $q \leftarrow q_2$ и завершаем алгоритм. В противном случае переходим к следующему шагу;
5) вычисляем «оптимальное» количество разбиений N^* по формуле (6.34):

$$N^* \leftarrow \left\lceil \left(\frac{|\tilde{R}|}{\varepsilon}\right)^{1/p} N_2 \right\rceil.$$

Здесь $\lceil x \rceil$ — минимальное целое, большее x;
6) полагаем $N_1 \leftarrow N_2$, $q_1 \leftarrow q_2$, $N_2 \leftarrow N^*$, $q_2 \leftarrow Q^{N^*}(f)$;
7) переходим к шагу 3.

Результатом работы алгоритма является значение $q \approx S(f)$. Этот алгоритм можно совершенствовать в следующих направлениях:

• **экономия вычислений f:** в зависимости от используемой базовой КФ подбирать N_1 и N_2 так, чтобы в соответствующих сетках для вычисления q_1 и q_2 было как можно больше одинаковых узлов. Например, в случае составной формулы трапеций (6.24) обычно берут $N_{i+1} = 2N_i$. Тогда получается

$$Q_1^N(f) = \frac{h}{2}\Big(f(a) + 2\sum_{k=1}^{N-1} f(a+kh) + f(b)\Big),$$

$$Q_1^{2N}(f) = \frac{h}{4}\Big(f(a) + 2\sum_{k=1}^{2N-1} f(a+k\frac{h}{2}) + f(b)\Big) =$$

$$= \frac{1}{2}\left(Q_1^N(f) + h\sum_{k=1}^{N} f(a+(2k-1)\frac{h}{2})\right);$$

$\triangleright_2$ Постройте соответствующую модификацию алгоритма.

• **использование уточненного приближения:** на шаге 4 можно возвращать не q_2, а его уточненное по формуле (6.33) значение $q_2 + \tilde{R}$;

• **использование неравномерных сеток** — наиболее радикальное изменение, которое по сути представляет собой совершенно иной алгоритм. Мотивация тут следующая: численное интегрирование по равномерно расположенным узлам является неоптимальным, если функция f на отрезке интегрирования имеет участки с различным поведением (например, сначала сильно осциллирует, а потом изменяется очень медленно). Поэтому современные программы численного интегрирования используют алгоритмы с неравномерно расположенными узлами. При этом принцип оценки погрешности остается прежним.

Глава 7

ЧИСЛЕННОЕ РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

7.1. Одношаговые методы решения задачи Коши

7.1.1. Постановка задачи

Рассмотрим систему обыкновенных дифференциальных уравнений (ОДУ)

$$y'(x) = f(x, y(x)) \tag{7.1а}$$

и начальное условие

$$y(x_0) = y_0, \quad x_0 \in \mathbb{R},\ y_0 \in \mathbb{R}^n. \tag{7.1б}$$

Здесь $y : \mathbb{R} \to \mathbb{R}^n$ — искомая вектор-функция, $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ — функция, которую иногда называют просто *правой частью* ОДУ. Формулы (7.1) вместе определяют *задачу Коши* для системы ОДУ.

Для развернутой покомпонентной записи данной задачи введем обозначения

$$y(x) = \begin{bmatrix} y^1(x) \\ y^2(x) \\ \vdots \\ y^n(x) \end{bmatrix}, \quad f(x, y(x)) = \begin{bmatrix} f^1(x, y(x)) \\ f^2(x, y(x)) \\ \vdots \\ f^n(x, y(x)) \end{bmatrix}.$$

Здесь $y^i : \mathbb{R} \to \mathbb{R}$ — скалярные функции-компоненты вектор-функции y, аналогично $f^i : (\mathbb{R} \times \mathbb{R}^n) \to \mathbb{R}$ — компоненты вектор-функции f. Мы вынуждены использовать верхние индексы для обозначения компонент вектор-функций, так как нижние индексы в дальнейшем будут заняты. В таких обозначениях задача Коши (7.1) принимает вид

$$\begin{cases} \frac{d}{dx}y^1(x) = f^1(x, y^1(x), \ldots, y^n(x)), & y^1(x_0) = y_0^1; \\ \frac{d}{dx}y^2(x) = f^2(x, y^1(x), \ldots, y^n(x)), & y^1(x_0) = y_0^2; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \frac{d}{dx}y^n(x) = f^n(x, y^1(x), \ldots, y^n(x)), & y^n(x_0) = y_0^n; \end{cases}$$

Во многих случаях формулы (7.1) удобно объединить в одну путем интегрирования:

$$y(x) = y_0 + \int\limits_{x_0}^{x} f(z, y(z))dz. \tag{7.2}$$

Пусть необходимо найти значение решения в некоторой точке $x_1 > x_0$. На практике редко удается вычислить $y(x_1)$ точно, поэтому приходится прибегать к приближенным (численным) методам.

7.1.2. Методы Эйлера

Явный метод Эйлера

Рассмотрим скалярный случай $n = 1$. Начальное условие (7.1б) позволяет составить уравнение касательной к точному решению y в точке x_0: если (x_1, y_1) — точка на касательной, то справедливо соотношение

$$\frac{y_1 - y_0}{x_1 - x_0} = y'(x_0) = f(x_0, y_0).$$

Полагая $y(x_1) \approx y_1$, получим самый простой и самый известный метод — *явный метод Эйлера*

$$y_1 = y_0 + (x_1 - x_0)f(x_0, y_0). \tag{7.3}$$

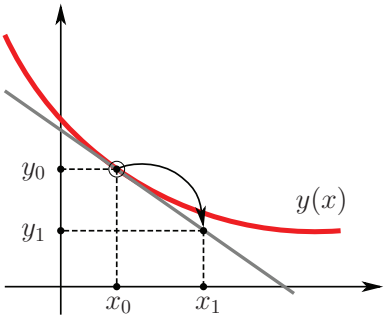


Рис. 7.1.

Геометрический смысл этого метода представлен на рис. 7.1.

Неявный метод Эйлера

Рассмотрим соотношение (7.2) для $x = x_1$:

$$y(x_1) = y_0 + \int\limits_{x_0}^{x_1} f(z, y(z))dz.$$

Приблизим интеграл по правилу правых прямоугольников[5]

$$\int\limits_a^b f(x)dx \approx (b-a)f(b)$$

и получим неявное соотношение для вычисления $y_1 \approx y(x_1)$:

$$y_1 = y_0 + (x_1 - x_0)f(x_1, y_1). \tag{7.4}$$

Это *неявный* метод Эйлера. Очевидно, что для вычисления y_1 необходимо решать нелинейное уравнение (систему из n нелинейных уравнений в общем случае).

$\triangleright_1$ В чем состоит геометрический смысл неявного метода Эйлера? Изобразите соответствующий рисунок.

$\triangleright_2$ Примените для аппроксимации интеграла (7.2) квадратурную формулу трапеций и запишите формулу для соответствующего метода.

[5]Заметим, что если применить формулу левых прямоугольников, получится явный метод Эйлера (7.3).

7.1.3. Одношаговые методы: терминология

Итак, методы Эйлера позволяют вычислить приближенное значение решения в точке x_1 по известному значению y_0 в точке x_0. Методы такого типа называются *одношаговыми*.

Определение 7.1. *Одношаговым методом* численного интегрирования задачи Коши (7.1) называется отображение

$$\Phi \ : \ \{x_0, y_0, x_1\} \mapsto y_1,$$

которое данному начальному условию (x_0, y_0) ставит в соответствие значение приближенного решения задачи (7.1) в данной точке $x_1 \in \mathbb{R}$:

$$\Phi(x_0, y_0, x_1) = y_1 \approx y(x_1).$$

Главное различие между двумя методами Эйлера состоит в том, что в одном случае y_1 задается явной формулой, а в другом оно определяется как решение нелинейного уравнения. Аналогично все методы численного решения ОДУ разделяются на явные и неявные.

Определение 7.2. Если отображение Φ является неявной функцией, т. е. задано уравнением вида

$$\Psi(x_0, y_0, x_1, y_1) = 0,$$

то соответствующий одношаговый метод называется *неявным*. В противном случае, т. е. если y_1 явно выражается через x_0, y_0 и x_1, метод называется *явным*.

Как правило, отображение Φ определено для всех x_1, лежащих в некоторой окрестности x_0, что позволяет рассматривать $\Phi(x_0, y_0, x)$ как функцию непрерывного аргумента x. Поэтому иногда для краткости будем рассматривать отображение $\varphi : \mathbb{R} \to \mathbb{R}^n$, определенное по правилу

$$\varphi(x) = \Phi(x_0, y_0, x).$$

Обычно вычислителю необходимо получить приближенное решение задачи Коши (7.1) на «большом» отрезке $[x_0, x_0 + H]$. Поскольку функция φ дает приемлемое приближение лишь в достаточно малой окрестности точки x_0, отрезок интегрирования разбивается на N частей сеткой узлов

$$x_0 < x_1 < x_2 \ldots < x_N = x_0 + H \tag{7.5}$$

и строится набор приближенных значений $y_k \approx y(x_k)$ по правилу

$$y_{k+1} = \Phi(x_k, y_k, x_{k+1}), \quad k = \overline{0, N-1}. \tag{7.6}$$

7.1.4. Порядок метода

Классическим показателем точности методов численного интегрирования ОДУ является порядок метода. Это понятие связано с разложением точного и приближенного решений в ряд Тейлора.

Рассмотрим сначала y — точное решение уравнения (7.1) в скалярном случае и предположим, что оно достаточно гладкое. Введем следующие традиционные обозначения:

$$f = f(x_0, y_0), \quad f_x = \frac{\partial f}{\partial x}(x_0, y_0), \quad f_y = \frac{\partial f}{\partial y}(x_0, y_0), \tag{7.7}$$

$$f_{\underbrace{x \ldots x}_{M}\,\underbrace{y \ldots y}_{N}} = \frac{\partial^{M+N} f}{\partial x^M \partial y^N}(x_0, y_0). \tag{7.8}$$

Разложение Тейлора для $y(x_0 + h)$ в точке x_0 имеет вид

$$y(x) = y_0 + hy'(x_0) + \frac{h^2}{2!}y''(x_0) + \frac{h^3}{3!}y'''(x_0) + \ldots \tag{7.9}$$

Тот факт, что y удовлетворяет (7.1), позволяет вычислить неизвестные коэффициенты $y^{(k)}(x_0)$ для любого k:

$$\begin{aligned} &y'(x_0) = f, \\ &y''(x_0) = \frac{d}{dx} f(x, y(x))\bigg|_{x=x_0} = f_x + f_y y'(x_0) = f_x + f_y f, \\ &y'''(x_0) = \frac{d^2}{dx^2} f(x, y(x))\bigg|_{x=x_0} = f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y(f_x + f_y f), \end{aligned} \tag{7.10}$$

и т. д. Таким образом, мы имеем теоретическую возможность *точно* вычислить разложение Тейлора (7.9).

Теперь зафиксируем начальное условие (x_0, y_0), рассмотрим произвольный одношаговый метод Φ и $\varphi(x) = \Phi(x_0, y_0, x)$. Традиционный способ оценки точности такого приближения заключается в сравнении разложения Тейлора для $\varphi(x)$ с разложением (7.9). Чем больше членов в этих разложениях совпадают, тем выше порядок метода.

Определение 7.3. *Локальной погрешностью* одношагового метода Φ называется функция

$$r(x) = y(x) - \varphi(x) = y(x) - \Phi(x_0, y_0, x).$$

Определение 7.4. Одношаговый метод имеет порядок p, если для всех достаточно гладких задач его локальная погрешность представима в виде

$$r(x_0 + h) = Ch^{p+1} + O(h^{p+2}), \quad C \neq 0, \tag{7.11}$$

т. е. разложения Тейлора в точке x_0 для точного решения $y(x_0 + h)$ и приближенного $\varphi(x_0 + h)$ совпадают до члена h^p включительно:

$$y^{(k)}(x_0) = \varphi^{(k)}(x_0) \quad \forall k = \overline{0, p}.$$

Слагаемое Ch^{p+1} называют *главным членом локальной погрешности*, а константу C — *константой погрешности* метода.

Замечание 7.1. Следует понимать, что высокий порядок метода не всегда гарантирует высокую точность приближения.

Порядок явного метода Эйлера. Согласно определению (7.3), имеем

$$\varphi(x) = \Phi(x_0, y_0, x) = y_0 + (x - x_0)f(x_0, y_0).$$

Отсюда получим

$$\begin{aligned}
&\varphi(x_0) = y_0,\\
&\varphi'(x_0) = f(x_0, y_0) = y'(x_0),\\
&\varphi^{(k)}(x_0) = 0 \neq y^{(k)}(x_0) \quad \forall k \geqslant 2.
\end{aligned}$$

Следовательно,

$$r(x_0 + h) = y(x_0 + h) - \varphi(x_0 + h) = \frac{h^2}{2!}y''(x_0) + O(h^3).$$

Таким образом, порядок явного метода Эйлера равен единице, а константа погрешности согласно (7.10) равна

$$C = \frac{1}{2}(f_x + f_y f). \tag{7.12}$$

Порядок неявного метода Эйлера. Рассмотрим (7.4). Для этого метода приближенное решение $\varphi(x)$ определяется неявным соотношением

$$\varphi(x) = y_0 + (x - x_0)f(x, \varphi(x)).$$

Вычислим производные $\varphi(x)$:

$$\begin{aligned}
\varphi'(x) &= f(x, \varphi(x)) + (x - x_0)\frac{d}{dx}f(x, \varphi(x)),\\
\varphi''(x) &= 2\frac{d}{dx}f(x, \varphi(x)) + (x - x_0)\frac{d^2}{dx^2}f(x, \varphi(x)).
\end{aligned}$$

Отсюда с учетом того, что $\varphi(x_0) = y_0$, получим

$$\varphi'(x_0) = f(x_0, y_0) = y'(x_0),$$
$$\varphi''(x_0) = 2\frac{d}{dx}f(x_0, y_0) = 2(f_x + f_y f) \neq y''(x_0).$$

Следовательно, разложение локальной погрешности в ряд Тейлора имеет вид

$$r(x_0 + h) = y(x_0 + h) - \varphi(x_0 + h) = -\frac{h^2}{2!}y''(x_0) + O(h^3).$$

Итак, неявный метод Эйлера имеет первый порядок, а его константа погрешности равна

$$C = -\frac{1}{2}(f_x + f_y f). \tag{7.13}$$

Замечание 7.2. На рассмотренных примерах видно, что в общем случае константа погрешности C представляет собой вектор, который выражается через значения частных производных f в точке (x_0, y_0).

$\triangleright_3$ Найдите порядок метода, построенного в $\triangleright_2$, а также главный член его локальной погрешности.

7.2. Методы Рунге – Кутты

Методы типа Рунге – Кутты (РК) являются наиболее популярными одношаговыми методами численного решения обыкновенных дифференциальных уравнений в настоящее время. Название этих методов связано с именами немецких математиков Карла Рунге (1856–1927) и Мартина Кутты (1867–1944).

7.2.1. Простейшие методы Рунге – Кутты

До сих пор мы рассматривали лишь два простейших метода численного решения ОДУ — явный и неявный методы Эйлера. Рассмотрим теперь более совершенные одношаговые методы.

Рассмотрим задачу Коши в форме интегрального уравнения (7.2) для $x = x_0 + h$:

$$y(x_0 + h) = y_0 + \int_{x_0}^{x_0+h} f(z, y(z))dz.$$

В интеграле удобно сделать замену переменных $z = x_0 + th$:

$$y(x_0 + h) = y_0 + h \int_0^1 f\big(x_0 + th, y(x_0 + th)\big)dt. \qquad (7.14)$$

Приблизим интеграл квадратурной формулой средних прямоугольников:

$$y(x_0 + h) \approx y_0 + hf\Big(x_0 + \frac{h}{2}, y\big(x_0 + \frac{h}{2}\big)\Big).$$

Использовать эту формулу для вычислений пока нельзя, так как неизвестно значение $y(x_0+h/2)$. Но его можно приблизить любым из методов Эйлера (7.3), (7.4). Для начала возьмем явный метод:

$$y\Big(x_0 + \frac{h}{2}\Big) \approx y_0 + \frac{h}{2} f(x_0, y_0),$$

и получим численный метод вида

$$Y_2 = y_0 + \frac{h}{2} f(x_0, y_0), \qquad (7.15а)$$

$$y_1 = y_0 + hf\Big(x_0 + \frac{h}{2}, Y_2\Big). \qquad (7.15б)$$

Этот метод будем называть *явным методом средних прямоугольников (средней точки)*.

Если же приблизить $y\left(x_0 + \dfrac{h}{2}\right)$ неявным методом Эйлера

$$y\Big(x_0 + \frac{h}{2}\Big) \approx Y_1 = y_0 + \frac{h}{2} f\Big(x_0 + \frac{h}{2}, Y_1\Big),$$

получим *неявный метод средних прямоугольников*:

$$Y_1 = y_0 + \frac{h}{2} f\Big(x_0 + \frac{h}{2}, Y_1\Big), \qquad (7.16а)$$

$$y_1 = y_0 + hf\Big(x_0 + \frac{h}{2}, Y_1\Big). \qquad (7.16б)$$

Оба полученных метода, а также и сами методы Эйлера являются частными случаями методов Рунге – Кутты.

7.2.2. Общий случай

В общем случае приблизим интеграл в (7.14) какой-то абстрактной квадратурной формулой с узлами $\{c_1, \ldots, c_s\}$ и весами $\{b_1, \ldots, b_s\}$:

$$\int_0^1 f\big(x_0 + th, y(x_0 + th)\big)dt \approx \sum_{i=1}^{s} b_i f\big(x_0 + c_i h, y(x_0 + c_i h)\big).$$

Для нахождения приближений к неизвестным величинам $y(x_0 + c_i h)$ используем такой же подход:

$$y(x_0 + c_i h) = y_0 + \int_{x_0}^{x_0 + c_i h} f(z, y(z))dz = y_0 + h\int_0^{c_i} f(x_0 + th, y(x_0 + th))dt,$$

а интеграл приблизим квадратурной формулой *по тем же самым узлам* $\{c_1, \ldots, c_s\}$, *но с другими коэффициентами* $\{a_{i1}, \ldots, a_{is}\}$:

$$y(x_0 + c_i h) \approx y_0 + h\sum_{j=1}^{s} a_{ij} f(x_0 + c_j h, y(x_0 + c_j h)).$$

Обозначая $y(x_0 + c_i h) \approx Y_i$, в итоге получим *s-стадийный метод Рунге – Кутты общего вида*:

$$\begin{aligned} Y_i &= y_0 + h\sum_{j=1}^{s} a_{ij} f(x_0 + c_j h, Y_j), \quad i = \overline{1, s}, \\ y_1 &= y_0 + h\sum_{i=1}^{s} b_i f\big(x_0 + c_i h, Y_i\big). \end{aligned} \tag{7.17}$$

Запись метода РК в виде (7.17) называется *симметричной*. Более распространенной является другая, эквивалентная форма записи, которая получается заменой переменных:

$$\kappa_i = f(x_0 + c_i h, Y_i), \tag{7.18}$$

или

$$Y_i = y_0 + h\sum_{j=1}^{s} a_{ij}\kappa_j. \tag{7.19}$$

В результате имеем

$$\begin{aligned} \kappa_i &= f\Big(x_0 + c_i h, y_0 + h\sum_{j=1}^{s} a_{ij}\kappa_j\Big), \quad i = \overline{1,s}, \\ y_1 &= y_0 + h\sum_{i=1}^{s} b_i\kappa_i. \end{aligned} \tag{7.20}$$

Таким образом, s-стадийный метод РК определяется $s^2 + 2s$ параметрами $\{a_{ij}\}_{i,j=1}^{s}$, $\{b_1, \ldots, b_s\}$ и $\{c_1, \ldots, c_s\}$. Традиционно эти параметры размещают в виде следующей таблицы:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|c} c_1 & a_{11} \ldots a_{1s} \\ \vdots & \ldots\ldots\ldots \\ c_s & a_{s1} \ldots a_{ss} \\ \hline & b_1 \ \ldots \ b_s \end{array}. \tag{7.21}$$

Матрицу $A = (a_{ij})$ называют *матрицей Бутчера*[6], а всю таблицу в формуле (7.21) — *таблицей Бутчера*.

$\triangleright_1$ Постройте таблицу Бутчера для методов Эйлера, а также для явного и неявного метода средних прямоугольников.

Заметим, что первая формула в (7.20), как и ее симметричный аналог из (7.17), задает систему нелинейных уравнений относительно неизвестных $\{\kappa_i\}_{i=1}^{s}$ ($\{Y_i\}_{i=1}^{s}$ соответственно), т. е. методы РК являются в общем случае неявными. Однако если коэффициенты метода удовлетворяют условиям

$$a_{ij} = 0 \quad \forall i \leqslant j$$

и

$$c_1 = 0,$$

то такой метод РК очевидно будет явным.

7.2.3. Условия порядка методов Рунге – Кутты

Неизвестные коэффициенты методов типа Рунге – Кутты (7.21) традиционно выбираются таким образом, чтобы получившийся метод имел как можно более высокий порядок точности.

[6] Джон Бутчер (John Butcher, университет Окленда) — новозеландский математик, внесший большой вклад в развитие и популяризацию методов РК.

Найдем условия, которым должен удовлетворять произвольный метод РК второго порядка. Рассмотрим приближенное решение y_1 как функцию переменной h. Поскольку функция метода имеет вид $\varphi(x_0+h)=y_1$, то

$$\frac{d^k\varphi}{dx^k}\Big|_{x=x_0}=\frac{d^k y_1}{dh^k}\Big|_{h=0}.$$

Тогда для того чтобы метод имел порядок 2, по определению необходимо, чтобы выполнялись условия

$$y_1|_{h=0}=y_0, \tag{7.22а}$$
$$y_1'|_{h=0}=y'(x_0)=f, \tag{7.22б}$$
$$y_1''|_{h=0}=y''(x_0)=f_x+f_y f. \tag{7.22в}$$

Условие (7.22а) очевидно выполняется всегда. Дифференцируя вторую формулу из (7.20) по h, имеем

$$\begin{aligned} y_1' &= \textstyle\sum_i b_i\kappa_i + h\sum_i b_i\kappa_i', \\ y_1'' &= 2\textstyle\sum_i b_i\kappa_i' + h\sum_i b_i\kappa_i''. \end{aligned} \tag{7.23}$$

Во всех суммах здесь и далее суммирование идет от 1 до s. Используя (7.19), вычислим

$$\begin{aligned} \kappa_i' &= \frac{d}{dh}f(x_0+c_ih,Y_i)=c_i f_x'()+f_y'()Y_i'= \\ &= c_i f_x'()+f_y'()\Big(\sum_j a_{ij}\kappa_j + h\sum_j a_{ij}\kappa_j'\Big). \end{aligned} \tag{7.24}$$

Здесь $()=(x_0+c_ih,Y_i)$. Используя (7.2.3), из (7.23) получим

$$\begin{aligned} y_1'|_{h=0} &= \textstyle\sum_i b_i\kappa_i|_{h=0}=\sum_i b_i f, \\ y_1''|_{h=0} &= 2\textstyle\sum_i b_i\kappa_i'|_{h=0}=2\sum_i b_i\left[c_i f_x + f_y f\sum_j a_{ij}\right]. \end{aligned}$$

Сопоставляя эти формулы с (7.22б), (7.22в), получим следующие условия второго порядка для методов РК:

$$\begin{aligned} &\textstyle\sum_i b_i = 1, \\ &\textstyle\sum_i b_i c_i = \frac{1}{2}, \\ &\textstyle\sum_i b_i\sum_j a_{ij} = \frac{1}{2}. \end{aligned} \tag{7.25}$$

$\triangleright_2$ Выведите условия третьего порядка.

Замечание 7.3. В подавляющем большинстве случаев при построении методов РК априори полагают $c_i = \sum_j a_{ij}$, так что последние два условия в (7.25) эквивалентны.

Замечание 7.4. При выводе условий (7.25) мы рассматривали случай скалярного ОДУ. Следует понимать, что в общем случае условия порядка для систем ОДУ не будут совпадать со скалярными.

7.2.4. Явные методы Рунге – Кутты

Общий вид

Явные методы Рунге – Кутты можно смело назвать классическими, так как именно такого рода формулы строили в свое время Рунге и Кутта.

Определение 7.5. Метод Рунге – Кутты называется явным, если его таблица Бутчера имеет вид, представленный в табл. 7.1.

Таблица 7.1

0	
c_2	a_{21}
c_3	$a_{31}\ a_{32}$
$\vdots$	$\ldots\ldots\ldots$
c_s	$a_{s1}\ a_{s2}\ \ldots\ a_{s,s-1}$
	$b_1\ \ b_2\ \ \ldots\ \ b_{s-1}\ \ b_s$

Общая схема вычисления $y_1 \approx y(x_0+h)$ по формулам (7.20) для явных методов РК имеет простой вид:

$$
\begin{aligned}
&\kappa_1 = f(x_0, y_0)\\
&\kappa_2 = f(x_0 + c_2 h, y_0 + h a_{21}\kappa_1),\\
&\kappa_3 = f(x_0 + c_3 h, y_0 + h(a_{31}\kappa_1 + a_{32}\kappa_2)),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&\kappa_s = f(x_0 + c_s h, y_0 + h(a_{s1}\kappa_1 + \ldots + a_{s,s-1}\kappa_{s-1}));\\
&y_1 = y_0 + h\sum_{i=1}^{s} b_i\kappa_i.
\end{aligned}
\qquad (7.26)
$$

Явные методы — наиболее простые в реализации методы типа РК: вспомогательные значения κ_i вычисляются последовательно, каждое по-

следующее из них явно выражается через уже найденные. Такой процесс очевидно невозможен, если в верхнем треугольнике матрицы A (включая диагональ) есть ненулевые элементы.

Примеры явных методов Рунге – Кутты

Второй порядок. Используя условия (7.25), легко получить общий вид явных двустадийных методов второго порядка. Поскольку $c_1 = a_{11} = a_{12} = a_{22} = 0$, для оставшихся коэффициентов метода имеем условия

$$b_1 + b_2 = 1,$$
$$b_2 c_2 = b_2 a_{21} = \frac{1}{2},$$

откуда получим следующий общий вид таблицы Бутчера:

$$\begin{array}{c|cc} 0 & & \\ \beta^{-1}/2 & \beta^{-1}/2 & \\ \hline & 1-\beta & \beta \end{array} \qquad (7.27)$$

Здесь $b_2 = \beta$ — параметр, как правило, лежащий в полуинтервале $(0, 1]$.

Третий порядок. Приведем в табл. 7.2 без вывода некоторые методы порядка 3.

Таблица 7.2

$$\begin{array}{c|ccc} 0 & & & \\ 1/3 & 1/3 & & \\ 2/3 & 0 & 2/3 & \\ \hline & 1/4 & 0 & 3/4 \end{array} \qquad \begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1 & 0 & 1 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/6 & 0 & 1/6 \end{array}$$

▷3 Постройте какой-нибудь трехстадийный явный метод РК третьего порядка.

Четвертый порядок. Наиболее популярный «классический» метод четвертого порядка определяется табл 7.3.

Таблица 7.3

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	2/6	2/6	1/6

7.3. Выбор шага численного интегрирования ОДУ

Пусть нам необходимо приближенно решить (или, как еще говорят, численно проинтегрировать) задачу Коши:

$$y'(x) = f(x, y(x)), \quad y(x_0) = y_0,$$

с помощью некоторого одношагового метода Ф. Как и ранее, приближенное значение решения в точке $x_i > x_0$ обозначаем y_i:

$$y_i \approx y(x_i).$$

Наиболее простой вариант реализации численного интегрирования состоит в использовании равномерной сетки узлов,

$$x_i = x_0 + ih,$$

где h — некоторое фиксированное число, которое также называют шагом интегрирования. В таком случае процесс решения имеет вид

$$y_{i+1} = \Phi(x_i, y_i, x_i + h), \quad i = 0, 1, 2, \ldots .$$

Такой подход обладает теми же недостатками, что и применение составных квадратурных формул с постоянным шагом: как правило, на

практике невозможно «угадать» нужную величину шага, которая обеспечивала бы требуемую точность и, с другой стороны, не приводила бы к чрезмерным вычислительным затратам.

Поэтому численное интегрирование задачи Коши для ОДУ обычно проводят с адаптивной[7] длиной шага:

$$x_{i+1} = x_i + h_{i+1}, \quad i = 0, 1, 2, \ldots .$$

где длина шага h_{i+1} зависит от результата вычислений на предыдущем, i-м шаге.

7.3.1. Адаптивный выбор шага

Существует два широко применяемых способа адаптивного выбора шага. Все они основаны на оценке константы погрешности. Первый способ — это уже знакомый нам метод двойного пересчета, или *правило Рунге*.

Правило Рунге

Правило Рунге для численного решения ОДУ практически полностью аналогично случаю приближенного вычисления определенного интеграла. Рассмотрим первый шаг процесса численного интегрирования методом Φ порядка p. Выберем какую-то величину начального шага h и сделаем сначала один «большой» шаг длиной $2h$:

$$\tilde{y}_2 = \Phi(x_0, y_0, x_0 + 2h),$$

а также два шага длиной h:

$$y_1 = \Phi(x_0, y_0, x_0 + h), \quad y_2 = \Phi(x_0 + h, y_1, x_0 + 2h).$$

Наша цель — оценить погрешность приближенного решения в точке $x_0 + 2h$. Для этого сравним погрешности

$$\tilde{e}_2 = y(x_0 + 2h) - \tilde{y}_2$$

и

$$e_2 = y(x_0 + 2h) - y_2.$$

[7] С длиной шага, которая автоматически приспосабливается к поведению решения.

По определению локальной погрешности (7.11) имеем

$$\tilde{e}_2 = y(x_0 + 2h) - \tilde{y}_2 = C_0(2h)^{p+1} + O(h^{p+2}). \qquad (7.28)$$

Выражение погрешности e_2 имеет более сложный вид (рис. 7.2). Оно состоит из двух частей:

$$e_2 = y(x_0 + 2h) - y_2 = r_2 + \varepsilon_2.$$

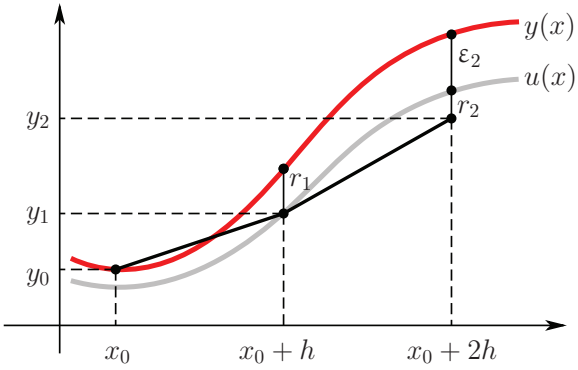


Рис. 7.2

Для записи r_2 (локальной погрешности второго шага) необходимо ввести в рассмотрение u — функцию, удовлетворяющую исходному дифференциальному уравнению

$$u'(x) = f(x, u(x)),$$

но с начальным условием $u(x_0 + h) = y_1$. Тогда по определению имеем

$$r_2 = u(x_0 + 2h) - y_2 = C_1 h^{p+1} + O(h^{p+2}), \qquad (7.29)$$

а

$$\varepsilon_2 = y(x_0 + 2h) - u(x_0 + 2h).$$

Можно показать, что для величины ε_2 справедлива оценка

$$\varepsilon_2 = \big(1 + O(h)\big) r_1,$$

где r_1 — локальная ошибка первого шага:

$$r_1 = y(x_0 + h) - y_1 = C_0 h^{p+1} + O(h^{p+2}),$$

а константа погрешности C_1 в (7.29) имеет вид

$$C_1 = C_0 + O(h).$$

Таким образом,

$$e_2 = \varepsilon_2 + r_2 = \Big(r_1 + O(h^{p+2})\Big) + \Big((C_0 + O(h))h^{p+1} + O(h^{p+2})\Big),$$

откуда

$$e_2 = y(x_0 + 2h) - y_2 = 2C_0 h^{p+1} + O(h^{p+2}). \tag{7.30}$$

Вычитая из (7.28) выражение (7.30), получим

$$y_2 - \tilde{y}_2 = 2C_0(2^p - 1)h^{p+1} + O(h^{p+2}).$$

Это равенство позволяет вычислить оценку главной части погрешности для y_2 (см. (7.30)), которую традиционно обозначают err:

$$y(x_0 + 2h) = y_2 + \mathrm{err} + O(h^{p+2}), \tag{7.31}$$

где

$$\mathrm{err} = \frac{y_2 - \tilde{y}_2}{2^p - 1}, \tag{7.32}$$

а также приближенное значение константы погрешности

$$C_0 = \frac{\mathrm{err}}{2h^{p+1}} + O(h). \tag{7.33}$$

Две последние формулы позволяют нам, во-первых, оценить погрешность e_2, во-вторых, уточнить приближенное решение y_2, и, в-третьих, выбрать оптимальный шаг интегрирования для дальнейших вычислений. Остановимся на каждом из этих пунктов подробнее.

Величина err, вычисляемая по формуле (7.32), по построению характеризует точность приближенного решения $y_2 \approx y(x_0 + 2h)$:

$$e_2 = \mathrm{err} + O(h^{p+2}).$$

Требуемую точность в дальнейшем будем обозначать tol. Если $|\mathrm{err}| < \mathrm{tol}$, то полученное приближение приемлемо, в противном случае необходимо повторить вычисления с меньшим шагом (см. ниже).

Далее, согласно (7.31), величина

$$\hat{y}_2 = y_2 + \mathrm{err} \tag{7.34}$$

является приближением к $y(x_0 + 2h)$ порядка $p + 1$:

$$y(x_0 + 2h) - \hat{y}_2 = O(h^{p+2}).$$

Таким образом, мы фактически получили способ повышения порядка метода на единицу. Величину $\hat{y}_2$ иногда называют *экстраполированным приближением*.

$\triangleright_1$ Пользуясь формулами (7.34), (7.32), постройте общую формулу, по которой из любого одношагового метода Φ порядка p можно получить метод $\widetilde{\Phi}$ порядка $p + 1$.

$\triangleright_2$ Запишите методы, полученные таким образом из явного и неявного методов Эйлера.

После того как вычислена оценка погрешности err, возможны два варианта развития событий в зависимости от выполнения неравенства

$$|\mathrm{err}| < \mathrm{tol}. \tag{7.35}$$

Предположим, что неравенство не выполняется, т. е. приближенное решение y_2 не является достаточно точным. Стандартный способ повышения точности в этом случае таков: y_2 «отбрасывается» и вычисления повторяются с меньшим шагом $\hat{h}$. Выбор величины $\hat{h}$ упрощается тем, что нам известна оценка константы погрешности (7.33). Согласно (7.30), мы должны выбрать $\hat{h}$ из условия

$$|2C_0\hat{h}^{p+1}| < \mathrm{tol},$$

откуда сразу же получаем

$$\hat{h} < \delta\, h, \tag{7.36}$$

где

$$\delta = \left(\frac{\mathrm{tol}}{|\mathrm{err}|}\right)^{\frac{1}{p+1}}. \tag{7.37}$$

Если же неравенство (7.35) выполняется, мы принимаем приближение y_2 (или его уточненное значение $\hat{y}_2$) и продолжаем процесс численного интегрирования из точки $x_0 + 2h$ с новым значением шага. Выбирают этот шаг из следующих соображений. Предположим, что константа погрешности на следующем шаге $\tilde{C}_0$ будет незначительно отличаться от C_0 (это будет верно, если шаг достаточно мал, а функция f достаточно

гладкая). Тогда для того чтобы на новом шаге выполнилась оценка (7.35), мы совершенно аналогично предыдущему случаю должны выбрать $\hat{h}$ из условий (7.36), (7.37). Только теперь величина δ будет больше единицы, т. е. шаг численного интегрирования *увеличится*.

Таким образом, **алгоритм автоматического (адаптивного) выбора шага численного интегрирования** имеет следующий общий вид:

1) $x \leftarrow x_0$, $y \leftarrow y_0$, $h \leftarrow h_0$, $X \leftarrow x_0 + H$;
2) если $x = X$, то завершаем алгоритм;
3) вычисляем $y_2 \leftarrow \Phi(x + h, \Phi(x, y, x + h), x + 2h)$, $\tilde{y}_2 \leftarrow \Phi(x, y, x + 2h)$;
4) находим оценку погрешности err (7.32) и коэффициент δ (7.37);
5) вычисляем $\hat{h} \leftarrow \alpha\delta h$, где $\alpha < 1$ — «страховочный множитель». Как правило, α выбирают в пределах от 0,7 до 0,9;
6) если $\delta < 1$, то полагаем $h \leftarrow \hat{h}$ и возвращаемся к пункту 3;
7) если же $\delta \geqslant 1$, то принимаем шаг: запоминаем пару значений $(x + 2h, y_2)$. Вместо y_2 здесь можно взять $\hat{y}_2$;
8) полагаем $x \leftarrow x + 2h$, $y \leftarrow y_2$ ($\hat{y}_2$), $h \leftarrow \min\{\hat{h}, \dfrac{X - x}{2}\}$;
9) возвращаемся к пункту 2.

Замечание 7.5. В силу погрешностей округления условие окончания алгоритма в пункте 2 следует использовать с осторожностью.

Замечание 7.6. В случае системы ОДУ оценка погрешности err будет векторной величиной. Поэтому в формулах вида (7.37) следует использовать какую-либо векторную норму. Кроме этого, для некоторых задач величина относительной погрешности может быть более информативной, поэтому используют также формулы типа

$$\mathrm{err} = \frac{1}{2^p - 1}\|D^{-1}(y_2 - \tilde{y}_2)\|,$$

где $D = \mathrm{diag}(\tilde{y}_2)$. Если используется абсолютная погрешность, то $D = I$ — единичная матрица.

Замечание 7.7. Общепринято накладывать ограничения на величину шага h: он не должен быть слишком мал. Если же такой случай возникает, программы численного интегрирования обычно выдают сообщение об ошибке и прекращают работу. Кроме этого, накладываются ограничения на скорость увеличения шага, т. е. на величину δ:

$$\delta = \min\left(\delta_{\max}, \left(\frac{\mathrm{tol}}{|\mathrm{err}|}\right)^{\frac{1}{p+1}}\right).$$

Оценка погрешности с помощью вложенных методов

Данный способ оценки главной части погрешности более прост, но менее универсален, чем правило Рунге. Он требует наличия двух методов: метода Φ порядка p и метода $\widehat{\Phi}$ порядка $q > p$. Обозначим

$$y_1 = \Phi(x_0, y_0, x_0 + h),$$
$$\hat{y}_1 = \widehat{\Phi}(x_0, y_0, x_0 + h).$$

По определению имеем

$$y(x_0 + h) - y_1 = Ch^{p+1} + O(h^{p+2}),$$
$$y(x_0 + h) - \hat{y}_1 = \widehat{C}h^{q+1} + O(h^{q+2}).$$

Отсюда с учетом того, что $q > p$, получим

$$\hat{y}_1 - y_1 = Ch^{p+1} + O(h^{p+2}).$$

Таким образом, величина

$$\text{err} = \hat{y}_1 - y_1 \tag{7.38}$$

с точностью до $O(h^{p+2})$ равна главной части локальной погрешности $y(x_0+h)-y_1$. Дальнейшие рассуждения полностью аналогичны правилу Рунге. В частности, правило выбора оптимального шага (7.36), (7.37), а также общая схема алгоритма адаптивного выбора шага остаются без изменений. Единственное отличие заключается в способе оценки погрешности: вместо (7.32) имеем (7.38).

В заключение приведем один из наиболее популярных вложенных методов Рунге – Кутты — метод Дормана – Принса порядка 5(4) (табл. 7.4).

Таблица 7.4

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
$\hat{b}_i$	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
b_i	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Удобство таких методов состоит в том, что для получения оценки погрешности требуется минимальное количество дополнительных вычислений: первая строчка коэффициентов $\hat{b}_i$ в табл. 7.4 определяет приближение $\hat{y}_1$ порядка 5:

$$\hat{y}_1 = y_0 + h \sum_i \hat{b}_i \kappa_i,$$

а вторая — y_1 порядка 4 для оценки погрешности, т. е.

$$\mathrm{err} = h \sum_i (\hat{b}_i - b_i) \kappa_i.$$

7.4. Многошаговые методы

7.4.1. Введение

Как и ранее, рассмотрим проблему численного решения задачи Коши

$$y'(x) = f(x, y(x)), \quad y(x_0) = y_0.$$

Предположим, что кроме начального значения y_0 нам известно еще и значение $y_1 = y(x_1)$, $x_1 = x_0 + h$. Каким образом можно использовать эту дополнительную информацию при вычислении приближения в точке $x_2 = x_1 + h$?

Проинтегрируем дифференциальное уравнение от x_1 до x_2:

$$y(x_2) = y(x_1) + \int_{x_1}^{x_2} f(x, y(x))dx = y_1 + h \int_0^1 F(t)dt,$$

где

$$F(t) = f(x_1 + th, y(x_1 + th)).$$

По условию нам известно, что

$$F(0) = f(x_1, y_1) = f_1, \quad F(-1) = f(x_0, y_0) = f_0,$$

поэтому можно приблизить подынтегральную функцию F интерполяционным многочленом P в форме Лагранжа:

$$P(t) = f_0\Lambda_0(t) + f_1\Lambda_1(t),$$

и получить приближенное равенство вида

$$y(x_2) \approx y_2 = y_1 + h \int_0^1 (f_0\Lambda_0(t) + f_1\Lambda_1(t))dt = y_1 + h(\beta_0 f_0 + \beta_1 f_1),$$

где

$$\beta_0 = \int_0^1 \Lambda_0(t)dt = \int_0^1 \frac{t-0}{-1-0}dt = -\frac{1}{2}, \quad \beta_1 = \int_0^1 \Lambda_1(t)dt = \int_0^1 \frac{t+1}{0+1}dt = \frac{3}{2}.$$

В итоге получим формулу

$$y_2 = y_1 + \frac{h}{2}(3f_1 - f_0). \tag{7.39}$$

Это — *двухшаговый метод*, а точнее, двухшаговый явный метод Адамса. Использованное здесь обозначение

$$f_i = f(x_i, y_i),$$

будет активно применяться в дальнейшем.

Определение 7.6. Многошаговым (k-шаговым) методом решения задачи Коши называется отображение

$$\Phi : \left\{ \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}, \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}, \ldots, \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix}, x_k \right\} \mapsto y_k \approx y(x_k).$$

При $k = 1$ очевидно данное определение превращается в определение одношагового метода.

В общем случае задачи Коши нам известно только y_0, следовательно, многошаговые методы при численной реализации нуждаются в процедуре вычисления «стартовых» значений $y_1, \ldots, y_{k-1}$. Обычно для этого используются одношаговые методы. После этого уже возможно пошаговое применение метода Φ:

$$y_{n+1} = \Phi \left(\begin{bmatrix} x_{n-k+1} \\ y_{n-k+1} \end{bmatrix}, \ldots, \begin{bmatrix} x_n \\ y_n \end{bmatrix}, x_{n+1} \right), \quad n = k-1, k, \ldots. \tag{7.40}$$

7.4.2. Явные методы Адамса

Нетрудно обобщить идею, которая использовалась при построении метода (7.39), на случай большего количества точек. Пусть известны значения $y_0, y_1, \ldots, y_{k-1}$,

$$y_i \approx y(x_i) = y(x_0 + ih).$$

Тогда имеем

$$y(x_k) = y(x_{k-1}) + h \int_0^1 F(t)dt, \tag{7.41}$$

$$F(t) = f(x_{k-1} + th, y(x_{k-1} + th)).$$

Функция F очевидно обладает свойством

$$F(0) = f_{k-1},\ F(-1) = f_{k-2},\ \ldots,\ F(1-k) = f_0,$$

или просто $F(t_j) = f_j$, где

$$t_j = 1 - k + j, \quad j = \overline{0, k-1}. \tag{7.42}$$

Строим для F интерполяционный многочлен P по узлам $\{t_j\}$:

$$F(t) \approx P(t) = \sum_{j=0}^{k-1} f_j \Lambda_j(t).$$

Заменяя F на P в интеграле (7.41), получим многошаговый метод вида

$$y_k = y_{k-1} + h \sum_{j=0}^{k-1} \beta_j f_j. \tag{7.43}$$

Коэффициенты β_j представляют собой интегралы от базисных функций Лагранжа Λ_i:

$$\beta_j = \int_0^1 \Lambda_j(t)dt = \int_0^1 \prod_{\substack{l=0 \\ l \neq j}}^{k-1} \frac{t - t_l}{t_j - t_l} dt, \tag{7.44}$$

т. е. по сути являются коэффициентами специальной интерполяционной квадратурной формулы, которая отличается от классических формул лишь тем, что использует узлы, лежащие за пределами отрезка интегрирования. Поэтому β_j можно найти не только по определению (7.44), но и используя тот факт, что по построению квадратурная формула

$$\int_0^1 F(t)dt \approx \sum_{j=0}^{k-1} \beta_j F(t_j)$$

должна иметь алгебраическую степень точности, равную $k - 1$. Подставляя в эту формулу $F(t) = t^m$, получим систему уравнений

$$\sum_{j=0}^{k-1} \beta_j (1 - k + j)^m = \frac{1}{m+1}, \quad m = 0, 1, \ldots, k-1. \tag{7.45}$$

$\triangleright_1$ Постройте явные методы Адамса для $k = 3$ и $k = 4$.

Замечание 7.8. В некоторых случаях строить формулы Адамса удобнее, применяя для представления интерполяционного многочлена P форму Ньютона.

7.4.3. Неявные методы Адамса

При интерполяции подынтегральной функции F в формуле (7.41) можно использовать не только известные значения F в точках $t = 0, -1, \ldots, 1-k$, но и неизвестное, «неявное» значение в точке $t = 1$, т. е.

$$F(1) = f(x_k, y_k).$$

Здесь y_k — искомое значение, которое нужно будет найти, решая полученное в итоге уравнение (систему уравнений). В результате такой модификации мы получим многошаговые методы Адамса, которые имеют общий вид

$$y_k = y_{k-1} + h\sum_{j=0}^{k} \hat{\beta}_j f_j. \tag{7.46}$$

Коэффициенты $\hat{\beta}_j$ вычисляются по формуле, аналогичной (7.44):

$$\hat{\beta}_j = \int_0^1 \hat{\Lambda}_j(t)dt = \int_0^1 \prod_{\substack{l=0 \\ l\neq j}}^{k} \frac{t-t_l}{t_j-t_l}dt. \tag{7.47}$$

Они могут быть найдены и как решение системы

$$\sum_{j=0}^{k} \hat{\beta}_j (1-k+j)^m = \frac{1}{m+1}, \quad m = 0, 1, \ldots, k. \tag{7.48}$$

▷$_2$ Постройте двухшаговый и трехшаговый неявные методы Адамса.

7.4.4. Точность методов Адамса

Рассмотрим некоторый k-шаговый явный метод Адамса. Предположим, что все начальные значения $y_0, y_1, \ldots, y_{k-1}$ заданы точно, т. е.

$$y_j = y(x_j) \quad \forall\, j = \overline{0, k-1},$$

а y_k вычислено с помощью метода. Наша задача — оценить величину погрешности

$$y(x_k) - y_k.$$

По построению имеем

$$y(x_k) = y(x_{k-1}) + h\int\limits_0^1 F(t)dt,$$

$$y_k = y_{k-1} + h\int\limits_0^1 P(t)dt,$$

где

$$F(t) = f(x_{k-1} + th, y(x_{k-1} + th)),$$

а многочлен P интерполирует F по узлам $\{t_j\}_{j=0}^{k-1}$ (7.42). Отсюда получим

$$y(x_k) - y_k = h\int\limits_0^1 r(t)dt,$$

где $r(t) = F(t) - P(t)$ — остаток полиномиальной интерполяции, который согласно формуле (5.28) можно представить в виде

$$r(t) = \frac{1}{k!}\frac{d^k}{dt^k}F(\xi)\ \underbrace{t(t+1)\dots(t+k-1)}_{\omega_k(t)},$$

где ξ — некоторая точка из интервала $(1-k, 0)$, зависящая от t. В силу знакопостоянства многочлена ω_k на отрезке $[0, 1]$ по уже неоднократно применявшейся нами теореме о среднем имеем

$$\int\limits_0^1 r(t)dt = \frac{1}{k!}\frac{d^k}{dt^k}F(\eta)\int\limits_0^1 t(t+1)\dots(t+k-1)dt = C\frac{d^k}{dt^k}F(\eta).$$

Здесь C — константа, зависящая только от k, а $\eta \in (1-k, 0)$. Дифференцируя F как сложную функцию, нетрудно убедиться в том, что

$$\frac{d^k}{dt^k}F(\eta) = O(h^k).$$

Отсюда окончательно получим

$$y(x_k) - y_k = hC\frac{d^k}{dt^k}F(\eta) = O(h^{k+1}),$$

т. е. *явный k-шаговый метод Адамса имеет порядок точности, равный k.*

В случае неявного k-шагового метода Адамса интерполяционный многочлен P имеет степень $k+1$. Проводя полностью аналогичные рассуждения, легко показать, что *неявный k-шаговый метод Адамса имеет порядок точности, равный $k+1$.*

7.5. Численное решение краевых задач. Метод стрельбы

7.5.1. Двухточечные краевые задачи

Знакомство с краевыми задачами для систем ОДУ начнем с рассмотрения примера. В плоскости xOy рассмотрим полет снаряда, выпущенного из точки $(0,0)$ с начальной абсолютной скоростью v_0 под углом α к оси Ox. Траектория полета описывается системой ОДУ

$$\begin{cases} y'(x) = \operatorname{tg}\theta(x), \\ v'(x) = -g\dfrac{\operatorname{tg}\theta(x)}{v(x)} - k\dfrac{v(x)}{\cos\theta(x)}, \\ \theta'(x) = -\dfrac{g}{v(x)^2} \end{cases} \tag{7.49}$$

с начальными условиями

$$y(0) = 0, \quad v(0) = v_0, \quad \theta(0) = \alpha. \tag{7.50}$$

Здесь $(x, y(x))$ — координаты снаряда, $v(x)$ — скорость снаряда, $\theta(x)$ — угол между вектором скорости и осью Ox в точке $(x, y(x))$, g — гравитационная постоянная (смасштабированная должным образом), k — коэффициент сопротивления воздуха.

Решить данную задачу Коши можно с помощью любого из рассмотренных ранее численных методов. Однако с практической точки зрения более интересна другая задача: *определить траекторию полета снаряда, при которой он поражает наземную цель, находящуюся в точке $x = X$.* Соответствующее решение уравнения (7.49) вместо начальных условий (7.50) должно удовлетворять условиям

$$y(0) = 0, \quad y(X) = 0, \quad v(0) = v_0, \tag{7.51}$$

которые называются *краевыми*. Уравнения (7.49) вместе с условиями (7.51) представляют собой типичный пример *двухточечной краевой задачи*.

Определение 7.7. Рассмотрим систему ОДУ

$$y'(x) = f(x, y(x)), \quad x \in [a, b], \tag{7.52а}$$

где y и f — вектор-функции:

$$y(x) = \begin{bmatrix} y^1(x) \\ \vdots \\ y^n(x) \end{bmatrix}, \quad f(x, y(x)) = \begin{bmatrix} f^1(x, y(x)) \\ \vdots \\ f^n(x, y(x)) \end{bmatrix}.$$

Зададим n условий (связей), которым должно удовлетворять решение системы (7.52а):

$$G_i\Big(y(a), y(b)\Big) = 0, \quad i = \overline{1, n}. \tag{7.52б}$$

Уравнения (7.52) определяют *двухточечную краевую задачу для ОДУ* общего вида.

Замечание 7.9. Помимо (вместо) краев отрезка $[a, b]$ в условиях (7.52б) могут фигурировать и другие точки из этого отрезка. В этом случае получим более общую задачу, которую называют *многоточечной задачей*.

Следует понимать, что краевая задача может иметь несколько или ни одного решения даже в том случае, когда соответствующая начальная задача (задача Коши) однозначно разрешима.

7.5.2. Метод стрельбы

Приступим к рассмотрению первого метода решения граничных задач вида (7.52) — метода стрельбы. Это один из методов, основанных на сведении краевой задачи (которую мы не умеем решать) к последовательности задач Коши (которые мы решать уже научились).

Рассмотрим артиллерийскую краевую задачу (7.49), (7.51). Эту задачу можно переформулировать так: *найти такой угол наклона пушки* $\alpha = \alpha^*$*, при котором решение задачи Коши с условиями* (7.50) *проходит через точку* $(X, 0)$. Обозначим $y(x, \alpha)$ решение системы (7.49) с начальными условиями (7.50), зависящее от параметра α. Тогда алгоритм нахождения нужного угла α^* может выглядеть следующим образом:

1) выбираем два значения α_1 и α_2, для которых заведомо выполняется условие $y(X, \alpha_1) < 0 < y(X, \alpha_2)$;
2) полагаем $\alpha \leftarrow (\alpha_1 + \alpha_2)/2$ и делаем пробный выстрел: решаем задачу Коши, находя $Y_\alpha = y(X, \alpha)$;
3) если $Y_\alpha = 0$ (или $|Y_\alpha| < \varepsilon$) — цель поражена, заканчиваем стрельбу;
4) если перелет ($Y_\alpha > 0$), полагаем $\alpha_2 \leftarrow \alpha$, в противном случае полагаем $\alpha_1 \leftarrow \alpha$;
5) переходим к шагу 2.
Это и есть простейший вариант метода стрельбы.

Описанная схема очевидно есть не что иное, как алгоритм метода бисекции для решения уравнения

$$F(\alpha) = y(X, \alpha) = 0.$$

Для ускорения сходимости можно применить более совершенные методы решения нелинейных уравнений, например метод секущих или Ньютона. Применение последнего, однако, затруднено необходимостью вычисления $F'(\alpha) = \dfrac{\partial}{\partial \alpha} y(X, \alpha)$.

Подведем локальный итог: *метод стрельбы заключается в подборе таких начальных условий, при которых решение данной системы ОДУ будет удовлетворять поставленным краевым условиям.*

Общая схема метода стрельбы

В общем случае ситуация осложняется тем, что неизвестных начальных условий может быть несколько. Например, вместо (7.51) можно рассмотреть условия

$$y(0) = 0, \quad y(X) = 0, \quad v(X) = v_X.$$

В этом случае нужно подбирать два начальных условия: $\theta(0)$ и $v(0)$, поэтому приходится использовать методы решения систем нелинейных уравнений: различные модификации метода Ньютона, метод Бройдена и т. п. Запишем общую схему метода стрельбы в таком случае.

Пусть дана система ОДУ (7.52а), которую запишем в покомпонентной форме

$$\frac{d}{dx} y^i(x) = f^i(x, y^1(x), \ldots, y^n(x)), \quad i = \overline{1, n}, \tag{7.53а}$$

и краевые условия, первые m из которых заданы в точке $x = b$, а остальные — в точке $x = a$, т. е. являются начальными:

$$\begin{cases} y^i(b) = \beta_i, & i = \overline{1, m}, \\ y^i(a) = \alpha_i, & i = \overline{m+1, n}. \end{cases} \tag{7.53б}$$

Для решения данной краевой задачи определим вектор-функцию

$$F = (F^1, \ldots, F^m)^T : \mathbb{R}^m \to \mathbb{R}^m$$

следующим образом:

$$F^i(\alpha_1, \ldots, \alpha_m) = y^i(b, \alpha_1, \ldots, \alpha_m) - \beta_i, \quad i = \overline{1, m},$$

где $y^i(x, \alpha_1, \ldots, \alpha_m)$ — решение исходной системы ОДУ (7.53а) с *начальными условиями*

$$y^i(a) = \alpha_i, \quad i = \overline{1, n} \tag{7.54}$$

(напомним, что начальные значения α_{m+1}, …, α_n заданы по условию). Метод стрельбы решения краевой задачи (7.53) заключается в нахождении вектора $\alpha^* = (\alpha_1^*, \ldots, \alpha_m^*)$, такого, что

$$F(\alpha^*) = 0.$$

Полученное решение $y(x, \alpha_1^*, \ldots, \alpha_m^*)$ по построению будет являтся решением краевой задачи (7.53).

Обратим особое внимание, что каждое вычисление $F(\alpha)$ требует решения задачи Коши (7.53а), (7.54), что весьма трудоемко. Более того, для реализации метода Ньютона, который считается наиболее эффективным методом решения систем уравнений, необходимо вычислять матрицу Якоби $\dfrac{\partial F}{\partial \alpha}$, вид которой, как правило, неизвестен. Эту матрицу приходится приближать с помощью конечных разностей, что, во-первых, ненадежно, и, во-вторых, очень накладно. Поэтому следует помнить, что метод стрельбы для краевых задач большой размерности очень трудоемок и ненадежен.

7.6. Решение краевых задач конечно-разностным методом

7.6.1. Общий нелинейный случай

Рассмотрим нелинейную краевую задачу для ОДУ второго порядка:

$$u''(x) = g(x, u(x), u'(x)), \quad u(a) = A, \quad u(b) = B. \tag{7.55}$$

Здесь $u : \mathbb{R} \to \mathbb{R}$, $g : \mathbb{R}^3 \to \mathbb{R}$. *Метод конечных разностей* или, как его чаще называют в русскоязычной литературе, *метод сеток* позволяет найти приближенное решение такой задачи в точках сетки $\{x_i\}_{i=0}^n$, которую чаще всего берут равномерной с шагом $h = (b - a)/n$:

$$x_i = a + ih, \quad i = \overline{0, n}. \tag{7.56}$$

Приближенное решение в этих узлах традиционно обозначают

$$y_i \approx u(x_i).$$

Поскольку по условию можно сразу положить $y_0 = A$, $y_n = B$, для нахождения $\{y_i\}_{i=1}^{n-1}$ необходимо из исходного дифференциального уравнения (7.55) получить $n - 1$ уравнений, которые бы связывали между собой эти неизвестные. Для этого необходимо выразить приближенные значения производных $u'(x_i)$ и $u''(x_i)$ через значения u в узлах сетки. Осуществить это можно разными способами, но все они в конечном итоге приводят к одинаковым результатам. Мы воспользуемся способом интерполяции.

Разностные производные

Для приближения производных применяется та же идея, что и при аппроксимации интегралов квадратурными формулами, а именно: *производная функции приближается производной многочлена, интерполирующего эту функцию по некоторому заранее заданному набору точек.*

Рассмотрим простейшие случаи. Пусть нам известны значения функции u в точках x_i и $x_{i+1} = x_i + h$. Интерполяционный многочлен в форме Ньютона по этим точкам имеет вид

$$P(x) = u(x_i) + \frac{u(x_{i+1}) - u(x_i)}{h}(x - x_i).$$

Отсюда получим

$$u'(x_i) \approx P'(x_i) = \frac{u(x_{i+1}) - u(x_i)}{h}. \tag{7.57}$$

Такая аппроксимация называется *правой разностной производной*.

Повторяя описанные действия для узлов интерполяции $\{x_{i-1}, x_i\}$, получим *левую разностную производную*

$$u'(x_i) \approx \frac{u(x_i) - u(x_{i-1})}{h}. \tag{7.58}$$

Более точное приближение получается при использовании трех интерполяционных узлов $\{x_{i-1}, x_i, x_{i+1}\}$:

$$P(x) = u(x_{i-1}) + \frac{u(x_i) - u(x_{i-1})}{h}(x - x_{i-1}) + u[x_{i-1}, x_i, x_{i+1}](x - x_{i-1})(x - x_i).$$

Дифференцируя этот многочлен, получим *центральную разностную производную*

$$u'(x_i) \approx \frac{u(x_{i+1}) - u(x_{i-1})}{2h} \tag{7.59}$$

и *вторую разностную производную*

$$u''(x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2}. \tag{7.60}$$

$\triangleright_1$ Выведите эти формулы.

Разностная схема

Приближая производные в дифференциальном уравнении (7.55) по формулам (7.59) и (7.60), мы получим набор приближенных равенств вида

$$\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} \approx g\Big(x_i, u(x_i), \frac{u(x_{i+1}) - u(x_{i-1})}{2h}\Big).$$

Здесь индекс i изменяется от 1 до $n-1$. Теперь заменим $u(x_i)$ на их приближенные значения y_i и окончательно получим следующую систему нелинейных уравнений, а точнее семейство систем, зависящих от параметра n:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} = g\Big(x_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}\Big), \quad i = \overline{1, n-1}, \tag{7.61а}$$

$$y_0 = A, \quad y_n = B. \tag{7.61б}$$

Такое семейство систем уравнений называется *разностной схемой*. Решая систему любым из известных нам численных методов при заданном значении шага h (числа n), получим приближенное решение краевой задачи (7.55).

7.6.2. Линейный случай

Наиболее просто конечно-разностный метод реализуется и исследуется в случае, когда уравнение линейно. Для простоты мы рассмотрим уравнение теплопроводности, которое описывает стационарное распределение тепла в стержне, на концах которого поддерживается постоянная температура A и B соответственно.

$$-u''(x) + q(x)u(x) = f(x), \quad u(a) = A, \quad u(b) = B. \tag{7.62}$$

Здесь $q(x)$ — коэффициент теплоотдачи, $f(x)$ — плотность источников тепла в точке x. Решение этого уравнения — функция u — определяет температуру стержня в точке x.

В данном случае уравнения (7.61а) превращаются в систему линейных уравнений

$$-\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + q_i y_i = f_i, \tag{7.63}$$

где $f_i = f(x_i)$; $q_i = q(x_i)$. Ее можно переписать в виде

$$-y_{i-1} + \left(2 + q_i h^2\right) y_i - y_{i+1} = h^2 f_i. \tag{7.64}$$

Обозначая

$$d_i = 2 + q_i h^2,$$

а также учитывая краевые условия $y_0 = A$, $y_n = B$, окончательно получим СЛАУ

$$L_h y_h = f_h, \tag{7.65}$$

где

$$L_h = \frac{1}{h^2}\begin{bmatrix} h^2 & 0 & & & & & \\ -1 & d_1 & -1 & & & & \\ & -1 & d_2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & d_{n-2} & -1 & \\ & & & & -1 & d_{n-1} & -1 \\ & & & & & 0 & h^2 \end{bmatrix}; \tag{7.66}$$

$y_h = (y_0, y_1, \ldots, y_{n-1}, y_n)^T$; $f_h = (A, f_1, \ldots, f_{n-1}, B)^T$. Нижний индекс h при записи системы (7.65) призван акцентировать факт зависимости размерности этой задачи и ее коэффициентов от величины шага сетки.

7.6.3. Устойчивость и сходимость метода сеток

Исследуем сходимость метода сеток для задачи (7.62). Будем считать $q(x) > 0\ \forall x \in [a, b]$, так что имеем

$$d_i = 2 + q_i h^2 > 2 \quad \forall i = \overline{1, n-1}. \tag{7.67}$$

Это свойство гарантирует диагональное преобладание для матрицы L_h, а значит, существование и единственность решения СЛАУ (7.65), а также применимость метода прогонки для ее решения.

Теперь наша задача — доказать, что метод сеток сходится, т. е.

$$\max_{0 \leqslant i \leqslant n} |u(x_i) - y_i| = \|u_h - y_h\| \xrightarrow[h \to 0]{} 0. \tag{7.68}$$

Здесь и далее $u_h = (u(x_0), u(x_1), \ldots, u(x_{n-1}), u(x_n))^T$ — вектор точных значений решения в узлах сетки, $\|\cdot\|$ — любая векторная норма.

Рассмотрим невязку, полученную при подстановке u_h вместо y_h в систему (7.65):

$$\psi_h = L_h u_h - f_h.$$

Такой вектор принято называть *погрешностью аппроксимации*. Поскольку по условию $L_h y_h = f_h$, имеем

$$L_h(u_h - y_h) = \psi_h,$$

откуда

$$\|u_h - y_h\| \leqslant \|L_h^{-1}\|\, \|\psi_h\|.$$

Из последнего соотношения становятся очевидными **достаточные условия сходимости метода**: *для того чтобы выполнялось* (7.68)*, достаточно выполнения следующих двух условий:*

1) $\|\psi_h\| \xrightarrow[h \to 0]{} 0$ *(условие аппроксимации);*

2) $\|L_h^{-1}\| \leqslant M \leqslant \infty\ \forall h \leqslant h_0$*, где* M *не зависит от* h *(условие устойчивости), а* h_0 *— некоторое положительное число.*

Аппроксимация

Докажем выполнение первого условия сходимости. Рассмотрим ψ_i — i-ю компоненту вектора $\psi_h = L_h u_h - f_h$. Пусть $i = \overline{1, n-1}$, тогда

$$\psi_i = -\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} + q_i u_i - f_i =$$
$$= -\frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2} + q(x_i)u(x_i) - f(x_i) =$$
$$= u''(x_i) - \frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2}.$$

Раскладывая в ряд Тейлора величины $u(x_i \pm h)$, в итоге получим отсюда

$$\psi_i = O(h^2), \quad i = \overline{1, n-1}.$$

$\triangleright_2$ Проделайте это.

Для $i = 0$ и $i = n$ имеем $\psi_i = 0$, так что окончательно

$$\|\psi_h\| = \max_i |\psi_i| = O(h^2) \xrightarrow[h \to 0]{} 0.$$

В таких случаях говорят, что *метод (схема) аппроксимирует дифференциальное уравнение со вторым порядком*.

Устойчивость

Для доказательства устойчивости, т. е. равномерной ограниченности нормы матрицы L_h^{-1}, достаточно показать, что если

$$L_h y_h = f_h,$$

то

$$\|y_h\| \leqslant M \|f_h\|, \quad M < \infty,$$

для всех h, меньших некоторого h_0, причем константа M не должна зависеть от h. Можно показать, что в нашем случае

$$M = \left(1 + \frac{(b-a)^2}{8}\right)$$

(доказательство этого факта выходит за рамки нашего курса).

Таким образом, конечно-разностный метод для линейной краевой задачи теплопроводности (7.62) сходится.

ЗАДАЧИ

Машинная арифметика

1. Для указанных значений $\{\beta, p, e_{\min}, e_{\max}\}$ изобразите на числовой прямой соответствующие множества нормализованных и денормализованных чисел с плавающей точкой, вычислите ε_M:
а) $\{2, 3, -2, 1\}$; б) $\{3, 2, -1, 1\}$; в) $\{5, 2, -1, 1\}$.

2. Известны два подряд идущих нормализованных машинных числа из некоторой двоичной арифметики с плавающей точкой:
а) 4, 4,25; б) 8, 8,25.
Чему равен ε_M (в зависимости от типа округления)?

3. Пусть ε_M — машинный эпсилон в некоторой машинной арифметике с плавающей точкой по основанию β. Оцените абсолютную погрешность округления $\Delta(x)$ для произвольного x в этой арифметике.

4. Рассмотрим машинную арифметику с параметрами $\{\beta, p, e_{\min}, e_{\max}\}$. Определите следующие величины:
1) максимальное число, точно представимое в данной арифметике;
2) минимальное положительное целое число, не представимое точно в данной арифметике;
3) минимальное нормализованное число, точно представимое в данной арифметике;
4) минимальное денормализованное число, точно представимое в данной арифметике.

5. Каким условиям должны удовлетворять параметры $\{\beta, p, e_{\min}, e_{\max}\}$, чтобы все натуральные числа, входящие в диапазон соответствующей машинной арифметики, были точно представимыми?

Обусловленность

6. Вычислите число обусловленности матрицы в норме $\|\cdot\|_\infty$:

а) $\begin{bmatrix} 2 & 0 & 0 \\ 1 & -1 & 0 \\ 2 & 1 & -2 \end{bmatrix}$; б) $\begin{bmatrix} 1 & 3 & 2 \\ -3 & -1 & 1 \\ 2 & 1 & -2 \end{bmatrix}$; в) $\begin{bmatrix} 2 & 0 & \dots & 0 \\ a & 2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ a & 0 & \dots & 2 \end{bmatrix}$;

г) $\begin{bmatrix} a & 0 & \dots & 0 \\ 1 & a & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 1 & 0 & \dots & a \end{bmatrix}$; д) $\begin{bmatrix} a & 1 & 0 & \dots & 0 \\ 0 & a & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a & 1 \\ 0 & 0 & \dots & 0 & a \end{bmatrix}$.

7. Исследуйте обусловленность задачи вычисления определителя матрицы размерности 2 относительно погрешности, вносимой в элемент на позиции а) $(1,1)$; б) $(2,1)$. Приведите примеры хорошо и плохо обусловленной задачи такого типа.

Прямые методы решения СЛАУ

8. Пусть известно LU-разложение матрицы A. Найдите решение системы $Ax = [0, 5, 0, -15]^T$.

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ -3 & -3 & -1 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} -4 & -4 & -3 & 5 \\ 0 & -5 & -5 & -5 \\ 0 & 0 & 5 & -4 \\ 0 & 0 & 0 & -3 \end{bmatrix}.$$

9. Постройте LU-разложение матрицы A и решите с его помощью СЛАУ $Ax = b$:

а) $A = \begin{bmatrix} 4 & -3 & 0 \\ 4 & -4 & -2 \\ 16 & -14 & -1 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$; б) $A = \begin{bmatrix} 3 & -1 & 4 \\ -3 & -2 & -9 \\ -6 & 5 & -4 \end{bmatrix}, b = \begin{bmatrix} 1 \\ -6 \\ 2 \end{bmatrix}$;

в) $A = \begin{bmatrix} 1 & 1 & -1 \\ -4 & -3 & 3 \\ -3 & 1 & -3 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ -4 \\ -5 \end{bmatrix}$.

10. Постройте LU-разложение матрицы вида $\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}$.

11. Пусть известно разложение Холецкого: $A = R^T D R$. Найдите решение СЛАУ $Ax = [-3, -2, 3, 1]^T$.

$$R = \begin{bmatrix} 3 & 1 & -1 & -1 \\ 0 & 1 & -2 & 0 \\ 0 & 0 & 4 & -3 \\ 0 & 0 & 0 & -4 \end{bmatrix}, \quad D = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

12. Решите методом квадратного корня СЛАУ:

а) $\left[\begin{array}{ccc|c} 9 & 3 & 3 & 0 \\ 3 & 5 & -5 & 10 \\ 3 & -5 & 19 & -24 \end{array}\right]$; б) $\left[\begin{array}{ccc|c} 4 & -2 & 8 & 2 \\ -2 & 5 & 2 & 3 \\ 8 & 2 & 34 & 10 \end{array}\right]$;

в) $\left[\begin{array}{cccc|c} -9 & -3 & 3 & 3 & 0 \\ -3 & -2 & 3 & 1 & -1 \\ 3 & 3 & 11 & -13 & -10 \\ 3 & 1 & -13 & 24 & 25 \end{array}\right]$; г) $\left[\begin{array}{cccc|c} -4 & 6 & -2 & 8 & 2 \\ 6 & -18 & 6 & -21 & -3 \\ -2 & 6 & 7 & 7 & 1 \\ 8 & -21 & 7 & -9 & 12 \end{array}\right]$.

13. Дана трехдиагональная матрица A, определяемая тремя векторами c (диагональ под главной), d (главная диагональ) и e (диагональ над главной). Запишите алгоритм построения LU-разложения для такой матрицы, а также алгоритм решения СЛАУ $Ax = b$ с помощью построенного разложения. Оцените сложность алгоритмов.

14. Дана трехдиагональная матрица A, задаваемая тремя векторами c, d и e (см. предыдущую задачу). Запишите алгоритм построения

QR-разложения *методом вращений* для такой матрицы и оцените его сложность.

15. Дана симметричная трехдиагональная матрица A, задаваемая вектором d (главная диагональ) и вектором c (диагональ над и под главной). Запишите алгоритм построения разложения Холецкого $A = R^T DR$ для такой матрицы, а также алгоритм решения СЛАУ $Ax = b$ с помощью построенного разложения. Оцените сложность алгоритмов.

Итерационные методы решения СЛАУ

16. Выбрав произвольное начальное приближение, выполните две итерации: а) метода Якоби; б) метода Гаусса – Зейделя; в) метода релаксации для приведенных ниже СЛАУ. Теоретически исследуйте сходимость каждого метода.

а) $\left[\begin{array}{cc|c} 2 & -1 & 0 \\ -2 & 2 & 1 \end{array}\right]$; б) $\left[\begin{array}{ccc|c} 3 & 1 & 1 & 4 \\ 1 & 5 & 1 & 6 \\ 2 & 0 & 9 & 2 \end{array}\right]$; в) $\left[\begin{array}{ccc|c} 1 & 2 & 1 & 3 \\ 2 & -5 & -3 & -8 \\ 1 & -3 & 2 & -1 \end{array}\right]$.

17. Сходится ли указанный итерационный процесс и если да, то к какому вектору?

1) $\begin{cases} x_1^{k+1} = (2 + x_2^k - x_3^k)/3, \\ x_2^{k+1} = (3 - x_1^k)/2, \\ x_3^{k+1} = (-1 - x_1^k + 2x_2^k)/4 \end{cases}$; 2) $\begin{cases} x_1^{k+1} = (2 + x_2^k - x_3^k)/3, \\ x_2^{k+1} = (3 - x_1^{k+1})/2, \\ x_3^{k+1} = (-1 - x_1^{k+1} + 2x_2^{k+1})/4 \end{cases}$.

18. Исследуйте сходимость методов Якоби и Гаусса – Зейделя для СЛАУ с матрицей общего вида $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$. Сравните результаты и сделайте выводы.

19. Запишите в общем виде критерий сходимости итерационного процесса вида $x_i^{k+1} = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^k - \sum_{j=i+1}^{n} a_{ij}x_j^{k+1}\right)/a_{ii}$, $i = n$, $n-1, \ldots, 1$, для решения СЛАУ $Ax = b$.

Форматы хранения разреженных матриц

20. Запишите матрицу в форматах CSR, CSC и MSR:

а) $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$; б) $\begin{bmatrix} 9 & 2 & 5 \\ 2 & 7 & 8 \\ 3 & 1 & 6 \end{bmatrix}$; в) $\begin{bmatrix} -2 & 2 & 0 & 1 \\ 2 & 1 & 3 & 0 \\ 4 & 5 & 0 & -2 \\ -6 & 9 & 8 & 7 \end{bmatrix}$; г) $\begin{bmatrix} 3 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 \end{bmatrix}$;

д) $\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 0 \\ 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 \end{bmatrix}$; е) $\begin{bmatrix} 3 & 0 & 0 & 4 & 0 \\ 2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 7 & 8 \\ 0 & 9 & 0 & 0 & 0 \end{bmatrix}$; ж) $\begin{bmatrix} 0 & 0 & 0 & 1 & 2 \\ 7 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 \\ 3 & 0 & 7 & 0 & 0 \end{bmatrix}$.

21. Восстановите исходный вид матрицы A, записанной в формате MSR:

а)

AA	5	3	0	2	8	×	3	5	6
IJ	7	7	8	8	9	10	1	3	2

;

б)

AA	5	3	0	2	8	×	3	5	6
IJ	7	7	7	7	9	10	1	3	2

.

Методы решения проблемы собственных значений

22. Методом Данилевского найдите характеристический многочлен матрицы.

а) $\begin{bmatrix} 3 & -2 & -1 \\ 2 & 2 & 2 \\ -4 & 2 & 0 \end{bmatrix}$; б) $\begin{bmatrix} 2 & -2 & -1 \\ 3 & 2 & 2 \\ -1 & 2 & 4 \end{bmatrix}$; в) $\begin{bmatrix} 3 & -1 & -1/2 \\ 2 & 2 & 3/2 \\ -4 & 0 & 1 \end{bmatrix}$;

г) $\begin{bmatrix} 0 & 2 & 1 & 1 \\ 1 & 4 & 0 & 0 \\ -1 & 2 & 1 & 3 \\ 0 & 0 & 0 & -1 \end{bmatrix}$; д) $\begin{bmatrix} 0 & 2 & 1 & 1 & 2 \\ 1 & 4 & 2 & 0 & 1 \\ -1 & 2 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix}$.

23. Вычислите приближенно максимальное по модулю собственное значение матрицы и соответствующий ему собственный вектор с помощью степенного метода:

а) $\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$; б) $\begin{bmatrix} 2 & -2 \\ 1 & 5 \end{bmatrix}$; в) $\begin{bmatrix} 2 & -2 & 4 \\ 1 & 5 & 1 \\ 2 & 3 & 4 \end{bmatrix}$; г) $\begin{bmatrix} -1 & -2 & 4 \\ 1 & 2 & 1 \\ 2 & 3 & -4 \end{bmatrix}$.

24. Известно, что матрица имеет два противоположных по знаку максимальных по модулю собственных значения. Найдите приближенно эти значения и соответствующие им собственные векторы с помощью степенного метода:

а) $\begin{bmatrix} 1 & 12 & 2 \\ 1 & -3 & -1 \\ 2 & 12 & 1 \end{bmatrix}$; б) $\begin{bmatrix} 3 & 10 & 2 \\ 2 & -5 & -5 \\ 4 & 10 & 1 \end{bmatrix}$.

25. Максимальное по модулю собственное значение матрицы $\begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}$ равно 5. Выполните несколько итераций степенного метода для начального приближения $y^0 = (2, -2)^T$ и объясните полученный парадоксальный результат.

26. Найдите все собственные значения и собственные векторы матрицы методом вращений Якоби:

а) $\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$; б) $\begin{bmatrix} -3 & 2 \\ 2 & -3 \end{bmatrix}$; в) $\begin{bmatrix} 4 & -1 \\ -1 & 2 \end{bmatrix}$.

27. Приведите матрицу к форме Хессенберга элементарными преобразованиями подобия с выбором главного элемента:

а) $\begin{bmatrix} 3 & 2 & -1 \\ 1 & 8 & 2 \\ -1 & 2 & 0 \end{bmatrix}$; б) $\begin{bmatrix} 2 & 2 & -1 & 0 \\ 1 & 8 & 1 & 3 \\ -2 & 3 & 0 & 1 \\ 1 & -1 & 4 & 7 \end{bmatrix}$; в) $\begin{bmatrix} 3 & 2 & -1 & 1 \\ 1 & 8 & 2 & 3 \\ -1 & 2 & 0 & 1 \\ 2 & 0 & 4 & 7 \end{bmatrix}$.

28. Постройте QR-разложение матрицы:

а) $\begin{bmatrix} 3 & -1 \\ 4 & 1 \end{bmatrix}$; б) $\begin{bmatrix} -3 & 1 \\ 4 & -2 \end{bmatrix}$; в) $\begin{bmatrix} -1 & 3 & 1 \\ 0 & -2 & 0 \\ 0 & 2 & -1 \end{bmatrix}$.

29. Выполните одну итерацию QR-алгоритма для матриц из предыдущего задания.

Методы решения нелинейных уравнений и систем

30. Постройте сходящийся итерационный процесс для вычисления вещественного корня уравнения:
а) $x^5 + \cos 2x = 0$; б) $2x^3 + x + 4 = 0$.

31. Постройте итерационный процесс вида $x_{k+1} = \varphi(x_k)$ для приближенного вычисления числа π, докажите его сходимость и укажите порядок сходимости.

32. Методом бисекции вычислите $\sqrt{3}$ с точностью 10^{-3}. Сколько итераций нужно сделать, чтобы достичь точности 10^{-10} с теми же начальными приближениями?

33. Оцените количество итераций метода бисекции, необходимое для вычисления корня уравнения $\cos x = x$ с точностью 10^{-6}, начиная с $[a, b] = [0, 1]$, $x_0 = 0{,}5$.

34. Исследуйте сходимость метода Ньютона для уравнения

$$3x^2 - 7x - 6 = 0$$

в зависимости от выбора начального приближения.

35. С помощью метода Ньютона постройте итерационный процесс приближенного вычисления $\sqrt[p]{a}$, $p \in \mathbb{N}$, и укажите множество начальных приближений x_0, для которых этот процесс сходится.

36. Определите порядок сходимости итерационного процесса

$$x_{k+1} = \frac{x_k \cos x_k - \sin x_k}{\cos x_k - 1}.$$

37. Докажите, что итерационный процесс

$$x_{k+1} = \frac{x_k^2 + 3}{2x_k}$$

имеет порядок сходимости не ниже 2.

38. Определите порядок сходимости метода

$$x_{k+1} = x_k - \frac{2h\, f(x_k)}{f(x_k + h) - f(x_k - h)}, \quad 0 < h < 1,$$

для решения уравнения $f(x) = 0$.

39. Выполните одну итерацию: а) метода Гаусса – Зейделя; б) метода Якоби для системы уравнений $\begin{cases} x\sqrt{y} + \sin z = 1, \\ x\cos z = 2(y-3), \\ x^2 + y^2 - z = 3, \end{cases}$ начиная с $x_0 = 2$, $y_0 = 1$, $z_0 = 0$.

40. Выполните одну итерацию метода Ньютона для системы

$$\begin{cases} a = bc, & a_0 = 0, \\ b + c = d, & b_0 = 1, \\ c = d^2, & c_0 = 2, \\ d = \sqrt{1+a}, & d_0 = 3. \end{cases}$$

41. Выполните две итерации метода Ньютона с постоянным якобианом для системы

$$\begin{cases} x = y^2 - z, & x_0 = 0, \\ y + x^2 = 1, & y_0 = 1, \\ z^2 = y, & z_0 = 2. \end{cases}$$

42. Выполните две итерации метода Ньютона для системы уравнений

$$\begin{cases} x^2 + y^2 = 1, \\ xy^2 = -1 \end{cases}$$

с начальным приближением $x_0 = 1$, $y_0 = -1$.

Приближение функций

43. Задайте произвольным образом на плоскости 3–5 точек (x_i, y_i) с целочисленными координатами и постройте по ним интерполяционный многочлен: а) в форме Лагранжа; б) барицентрической форме; в) форме Ньютона. Проверьте полученный ответ.

44. Оцените погрешность интерполяции функции $f(x) = xe^{2x}$ алгебраическим многочленом второй степени на отрезке $[-0{,}5, 0{,}5]$ по трем узлам: $-0{,}5; 0; 0{,}5$.

45. Оцените погрешность интерполяции функции $f(x) = \sin 2x$ алгебраическим многочленом четвертой степени на отрезке $[0, 1]$ по чебышевским узлам.

46. Определите степень интерполяционного многочлена на равномерной сетке, обеспечивающего точность приближения функции e^x на отрезке $[0, 1]$ не менее 10^{-3}.

47. Постройте интерполяционный сплайн первой степени по точкам $(-1, 2)$, $(2, 3)$, $(3, 2)$.

48. Постройте естественный кубический интерполяционный сплайн по точкам $(-1, 0)$, $(0, 1)$, $(1, 0)$.

49. Постройте кубический интерполяционный сплайн по точкам $(-2, 2)$, $(1, 0)$, $(3, 2)$ с граничными условиями $s'(-2) = 1, s''(-2) = 0$.

50. Задайте произвольным образом на плоскости 3–5 точек (x_i, y_i). Постройте алгебраическую интерполяционную кривую, проходящую через эти точки: а) по равномерным; б) естественным параметрическим узлам.

51. Пусть B — функция, задающая кривую Безье для контрольных точек: а) $(1,2), (2,-1), (2,3)$; б) $(-1,2), (2,1), (0,3), (-1,4)$. Вычислите координаты точек $B(0{,}5)$ и $B(0{,}2)$ тремя разными способами.

52. Для функции $f(x,y) = \dfrac{x}{y^2+1}$ Постройте интерполяционный многочлен в форме Лагранжа по узлам $\{0,1,3\} \times \{-1,1\}$.

53. Для функции $f(x,y) = x\sqrt{y^2+1}$ Постройте интерполяционный многочлен в форме Лагранжа по узлам $\{0,3\} \times \{-1,1,2\}$.

54. Постройте многочлен двух переменных P, удовлетворяющий условиям $P(0,0) = 1$, $P(1,2) = -1$, $P(2,1) = 2$.

55. Постройте наилучшее среднеквадратичное приближение в виде многочлена: а) нулевой; б) первой; в) второй; г) третьей степени к набору точек $(1,1)$, $(2,3)$, $(3,1)$, $(4,4)$. Изобразите графики построенных приближений.

56. Постройте наилучшее среднеквадратичное приближение вида $\varphi(x) = \alpha x$ к набору точек $(1,1)$, $(2,3)$, $(3,1)$, $(4,0)$.

57. Постройте наилучшее среднеквадратичное приближение вида $\varphi(x) = \alpha x + \beta\dfrac{1}{x}$ к набору точек $(1,1)$, $(2,3)$, $(3,1)$, $(4,4)$.

58. Постройте наилучшее среднеквадратичное приближение вида $\varphi(x) = \alpha + \beta x^2$ к набору точек $(-1,1)$, $(0,0)$, $(1,1)$, $(2,2)$.

59. Найдите наилучшее среднеквадратичное приближение для функции $f(x) = \sin\dfrac{\pi}{2}x$ на отрезке $[0,1]$ среди всех многочленов первой степени. Постройте графики.

60. Найдите наилучшее среднеквадратичное приближение для функции $f(x) = \cos \pi x$ на отрезке $[0,1]$ среди всех многочленов первой степени. Постройте графики.

61. Найдите наилучшее среднеквадратичное приближение для функции $f(x) = x$ на отрезке $[-\pi, \pi]$ среди всех тригонометрических многочленов вида $a + b\cos x + c\sin x$. Постройте графики.

62. Может ли интерполяционный многочлен по чебышевским узлам приближать функцию хуже, чем интерполяционный многочлен той же степени по равноотстоящим узлам? Ответ обоснуйте.

63. Может ли интерполяционный многочлен нулевой степени приближать функцию лучше, чем интерполяционный многочлен первой степени для той же функции на том же отрезке? Ответ обоснуйте.

64. Известно, что многочлен P_2 интерполирует функцию f по точкам 0, 1 и 2, а многочлен Q_2 интерполирует эту же функцию по точкам 1, 2 и 3. Постройте интерполяционный многочлен для функции f по точкам 0, 1, 2 и 3.

65. На плоскости даны три точки: $q_0 = (0, 0)$, $q_1 = (1, -1)$, $q_2 = (1, 1)$. Как нужно выбрать точку q_3, чтобы кривая Безье, построенная по множеству контрольных точек $\{q_0, q_1, q_2, q_3, q_0\}$, была гладкой?

Квадратурные формулы

66. Постройте аналог квадратурной формулы левых прямоугольников для приближенного вычисления интегралов вида $\int_0^1 f(x)\sqrt{x}dx$. Определите АСТ этой формулы.

67. Постройте аналог квадратурной формулы правых прямоугольников для приближенного вычисления интегралов вида $\int_0^1 f(x)\sqrt{x}dx$. Определите АСТ этой формулы.

68. Постройте квадратурную формулу с одним узлом и максимально возможной АСТ для приближенного вычисления интегралов вида $\int_0^1 f(x)\sqrt{x}dx$. Укажите АСТ этой формулы.

69. Постройте квадратурную формулу с одним узлом и максимально возможной АСТ для приближенного вычисления интегралов вида $\int_0^1 f(x)e^x dx$. Укажите АСТ этой формулы.

70. Определите алгебраическую степень точности квадратурной формулы

$$\int_{-1}^{1} f(x)dx \approx \frac{1}{2}(f(-1) + 3f(1/3)).$$

71. Постройте квадратурную формулу вида

$$\int_{-1}^{1} f(x)dx \approx A_1 f(x_0) + A_2 f(1)$$

с максимально возможной алгебраической степенью точности.

72. Оцените число отрезков, на которое нужно разбить отрезок $[0, 1]$ для вычисления интеграла $\int_{-1}^{1} e^{2x}dx$ по составной квадратурной формуле трапеций.

73. Оцените количество частей, на которое нужно разбить отрезок $[0, 1]$ для вычисления интеграла $\int_{-1}^{1} e^{2x}dx$ с точностью 10^{-3} по составной квадратурной формуле: а) средних прямоугольников; б) трапеций.

74. Вычислите приближенное значение интеграла $\int_0^1 \sqrt{x}dx$ по составной формуле трапеций на двух отрезках и оцените погрешность по правилу Рунге.

75. Рассмотрим интерполяционную квадратурную формулу

$$\int_{-1}^{1} f(x)|x|dx \approx \sum_{i=0}^{7} A_i f(x_i).$$

Чему равна сумма всех ее коэффициентов?

76. Рассмотрим интерполяционную квадратурную формулу

$$\int_{0}^{1} f(x)\sqrt{x}dx \approx \sum_{i=0}^{7} A_i f(x_i).$$

Найдите $\sum_{i=0}^{7} A_i x_i$.

Численное решение ОДУ

77. Дана задача Коши: $y'(x) = x - y(x)$, $y(0) = 1$. Вычислите приближенное значение $y(0{,}5)$: а) явным; б) неявным методом Эйлера. Оцените погрешность по правилу Рунге.

78. Рассмотрим следующий метод Рунге – Кутты:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{3} & \frac{1}{3} & & & \\ \frac{2}{3} & -\frac{1}{3} & 1 & & \\ 1 & 1 & -1 & 1 & \\ \hline & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{array}$$

При его программной реализации была допущена ошибка, в результате которой y_1 вычислилось как $y_1 = y_0 + h(b_1k_1 + b_2k_1 + b_3k_2 + b_4k_3)$. Постройте таблицу Бутчера для получившегося метода и определите его порядок.

79. Рассмотрим следующий одношаговый метод: отрезок $[x_0, x_0 + h]$ разбивается на s равных частей, и в качестве приближенного решения в точке $x_0 + h$ берется значение, вычисленное соответствующим количеством шагов явного метода Эйлера на полученной сетке. Оказывается, что такой метод будет методом Рунге – Кутты. Постройте его таблицу Бутчера для произвольного s. Определите порядок этого метода.

80. Даны два метода Рунге – Кутты: $\begin{array}{c|c} \alpha & \alpha \\ \hline & 1 \end{array}$ и $\begin{array}{c|c} \beta & \beta \\ \hline & 1 \end{array}$. Запишите таблицу Бутчера для метода, который получается последовательным применением этих методов с шагом $h/2$. Определите максимально возможный порядок получившегося метода.

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 ; под ред. И. П. Мысовских. – Минск, 1972.– Т. 1. – 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 c.

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.

ОГЛАВЛЕНИЕ

Учебное издание

Фалейчик Борис Викторович

МЕТОДЫ ВЫЧИСЛЕНИЙ

Пособие

Редактор *Н. Ф. Акулич*
Художник обложки *Т. Ю. Таран*
Технический редактор *Т. К. Раманович*
Компьютерная верстка *Б. В. Фалейчика*
Корректор *Е. В. Гордейко*

Подписано в печать 31.12.2014.
Формат 60×84/16. Бумага офсетная.
Ризография. Усл. печ. л. 13,02.
Уч.-изд. л. 12,19. Тираж 100 экз. Заказ 33.

Белорусский государственный университет.
Свидетельство о государственной регистрации
издателя, изготовителя, распространителя
печатных изданий № 1/270 от 03.04.2014.
Пр. Независимости, 4, 220030, Минск.

Республиканское унитарное предприятие
«Издательский центр Белорусского
государственного университета».
Свидетельство о государственной регистрации
издателя, изготовителя, распространителя
печатных изданий № 2/63 от 19.03.2014.
Ул. Красноармейская, 6, 220030, Минск.