

Белорусский государственный университет  
факультет прикладной математики и информатики

**Б. В. Фалейчик**

# **МЕТОДЫ ВЫЧИСЛЕНИЙ**

*Курс лекций*

Минск  
БГУ  
2012

# ОГЛАВЛЕНИЕ

<b>Глава 1. ВВЕДЕНИЕ</b>	<b>8</b>
<b>1.1. Машинная арифметика</b>	<b>9</b>
1.1.1. Числа с плавающей точкой	9
1.1.2. Двоичные числа с плавающей точкой	9
1.1.3. Способы округления	11
1.1.4. Расширение множества чисел с плавающей точкой	12
1.1.5. Машинный эпсилон	13
1.1.6. Стандарт IEEE 754	14
1.1.7. Проблемы машинных вычислений	15
<b>1.2. Обусловленность задачи</b>	<b>16</b>
1.2.1. Некорректные и плохо обусловленные задачи	16
1.2.2. Векторные нормы	18
1.2.3. Матричные нормы	18
1.2.4. Число обусловленности матрицы	20
1.2.5. Резюме	22
<b>Глава 2. МЕТОДЫ РЕШЕНИЯ СЛАУ</b>	<b>23</b>
<b>2.1. Метод Гаусса</b>	<b>24</b>
2.1.1. Введение	24
2.1.2. Базовый метод Гаусса	25
2.1.3. Связь метода Гаусса и $LU$ -разложения	28
2.1.4. Метод Гаусса с выбором главного элемента	28
2.1.5. Матричные уравнения	30
2.1.6. Обращение матрицы и вычисление определителя	30
2.1.7. Метод прогонки	31
<b>2.2. <math>LU</math>-разложение</b>	<b>33</b>
2.2.1. Базовый алгоритм $LU$ -разложения	33
2.2.2. Выбор главного элемента	35

2.3. Метод квадратного корня . . . . .	37
2.3.1. Разложение Холецкого . . . . .	37
2.3.2. Алгоритм метода . . . . .	39
2.4. Параллельная реализация метода Гаусса . . . . .	40
2.4.1. Введение . . . . .	40
2.4.2. Распределение данных между узлами . . . . .	40
2.4.3. Прямой ход . . . . .	41
2.4.4. Обратный ход . . . . .	42
2.5. Итерационные методы решения СЛАУ . . . . .	43
2.5.1. Общая характеристика . . . . .	43
2.5.2. Итерационные методы общего вида . . . . .	43
2.5.3. Сходимость итерационных процессов общего вида . . . .	47
2.6. Форматы хранения разреженных матриц . . . . .	51
2.6.1. Координатный формат . . . . .	51
2.6.2. Форматы CSR и CSC . . . . .	52
2.6.3. Формат MSR . . . . .	53
<b>Глава 3. МЕТОДЫ РЕШЕНИЯ ПРОБЛЕМЫ СОБСТВЕННЫХ ЗНАЧЕНИЙ</b> . . . . .	<b>56</b>
3.1. Проблема собственных значений: общая характеристика . . . .	57
3.1.1. Сведения из линейной алгебры . . . . .	57
3.1.2. Общая характеристика проблемы собственных значений . . .	58
3.2. Степенной метод . . . . .	59
3.2.1. Случай 1 . . . . .	59
3.2.2. Случай 2 . . . . .	60
3.2.3. Случай 3 . . . . .	61
3.2.4. Случай 4 . . . . .	62
3.2.5. Общий случай . . . . .	63
3.2.6. Степенной метод со сдвигом . . . . .	63
3.3. Метод Данилевского . . . . .	64
3.3.1. Преобразования подобия . . . . .	64
3.3.2. Метод Данилевского . . . . .	66
3.4. Метод вращений Якоби . . . . .	70
3.4.1. Преобразование вращения . . . . .	70
3.4.2. Симметричная проблема собственных значений . . . . .	72
3.4.3. Общая схема вращений Якоби . . . . .	72
3.4.4. Расчётные формулы метода . . . . .	75

3.5.	<b>QR-алгоритм</b>	78
3.5.1.	Предварительные сведения	78
3.5.2.	Общая схема QR-алгоритма	80
3.5.3.	QR-разложение для формы Хессенберга	81
3.5.4.	Детали реализации алгоритма	82
<b>Глава 4.</b>	<b>МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ И СИСТЕМ</b>	<b>84</b>
4.1.	<b>Решение нелинейных уравнений</b>	<b>85</b>
4.1.1.	Введение	85
4.1.2.	Основные теоретические сведения	86
4.1.3.	Метод бисекции	87
4.1.4.	Метод Ньютона	88
4.1.5.	Модификации метода Ньютона	90
4.2.	<b>Решение систем нелинейных уравнений</b>	<b>93</b>
4.2.1.	Постановка задачи	93
4.2.2.	Метод Ньютона	93
4.2.3.	Метод Ньютона с постоянным якобианом	95
4.2.4.	Обобщения метода секущих	95
4.2.5.	Метод Бройдена	96
4.2.6.	Методы спуска	98
4.2.7.	Еще немного	100
4.3.	<b>Анализ сходимости итерационных процессов</b>	<b>101</b>
4.3.1.	Принцип сжимающих отображений	101
4.3.2.	Локальный принцип сжимающих отображений	103
4.3.3.	Применение принципа сжимающих отображений	104
<b>Глава 5.</b>	<b>ПРИБЛИЖЕНИЕ ФУНКЦИЙ</b>	<b>107</b>
5.1.	<b>Интерполяция функций. Интерполяционный многочлен</b>	<b>108</b>
5.1.1.	Общая постановка задачи интерполяции	108
5.1.2.	Полиномиальная интерполяция	109
5.1.3.	Интерполяционный многочлен в форме Лагранжа	111
5.1.4.	Итого	112
5.2.	<b>Интерполяционный многочлен в форме Ньютона</b>	<b>113</b>
5.2.1.	Построение	113
5.2.2.	Разделённые разности	115
5.2.3.	Алгоритм вычисления разделённых разностей	117
5.2.4.	Остаток интерполирования	117

5.3. Минимизация остатка интерполирования. Многочлены Чебышева . . . . .	120
5.3.1. Задача выбора оптимальных узлов интерполирования . . . . .	120
5.3.2. Многочлены Чебышева . . . . .	120
5.3.3. Оптимальные узлы интерполирования . . . . .	121
5.3.4. О сходимости интерполяционного процесса . . . . .	123
5.3.5. Барцентрическая форма записи многочлена Лагранжа . . . . .	125
5.4. Сплаины . . . . .	128
5.4.1. Определение . . . . .	128
5.4.2. Интерполяционные сплайны . . . . .	129
5.4.3. Экстремальное свойство кубического сплайна . . . . .	134
5.4.4. Сходимость интерполяции кубическим сплайном . . . . .	135
5.5. Приближение кривых . . . . .	136
5.5.1. Интерполяция кривых на плоскости . . . . .	136
5.5.2. Построение интерполирующих кривых . . . . .	137
5.5.3. Кривые Безье . . . . .	138
5.5.4. Алгоритмы построения кривых Безье . . . . .	141
5.6. Среднекватратичные приближения . . . . .	144
5.6.1. Дискретный случай . . . . .	144
5.6.2. Среднекватратичное приближение функций . . . . .	147
5.6.3. Базисные функции . . . . .	149
5.7. Приближение поверхностей . . . . .	154
5.7.1. Интерполяция на прямоугольнике . . . . .	154
5.7.2. Полиномиальная интерполяция . . . . .	155
5.7.3. Интерполяция билинейными сплайнами . . . . .	156
5.7.4. Интерполяция бикубическими сплайнами . . . . .	157
5.7.5. Поверхности Безье . . . . .	160
Глава 6. ПРИБЛИЖЕННОЕ ВЫЧИСЛЕНИЕ ИНТЕГРАЛОВ . . . . .	161
6.1. Интерполяционные квадратурные формулы . . . . .	162
6.1.1. Постановка задачи численного интегрирования . . . . .	162
6.1.2. Простейшие квадратурные формулы . . . . .	162
6.1.3. Определение . . . . .	164
6.1.4. Интерполяционные квадратурные формулы . . . . .	165
6.1.5. Остаток простейших квадратурных формул . . . . .	167
6.2. Квадратурные формулы Ньютона–Котеса . . . . .	170
6.2.1. Алгебраическая степень точности . . . . .	170
6.2.2. Симметричные квадратурные формулы . . . . .	171
6.2.3. Симметрия интерполяционных КФ . . . . .	172

6.2.4. Квадратурные формулы Ньютона—Котеса . . . . .	174
<b>6.3. Квадратурные формулы Гаусса . . . . .</b>	<b>176</b>
6.3.1. Постановка задачи . . . . .	176
6.3.2. Построение квадратурных формул Гаусса . . . . .	177
6.3.3. Классические квадратурные формулы Гаусса . . . . .	180
<b>6.4. Составные квадратурные формулы . . . . .</b>	<b>183</b>
6.4.1. Введение . . . . .	183
6.4.2. Простейшие составные формулы . . . . .	183
6.4.3. Остаток составных квадратурных формул . . . . .	185
6.4.4. Выделение главной части погрешности . . . . .	187
6.4.5. Практическая оценка погрешности: правило Рунге . . . . .	189
<b>Глава 7. ЧИСЛЕННОЕ РЕШЕНИЕ ОДУ . . . . .</b>	<b>193</b>
<b>7.1. Одношаговые методы решения задачи Коши . . . . .</b>	<b>194</b>
7.1.1. Постановка задачи . . . . .	194
7.1.2. Методы Эйлера . . . . .	195
7.1.3. Одношаговые методы: терминология . . . . .	196
7.1.4. Порядок метода . . . . .	197
<b>7.2. Методы Рунге—Кутты . . . . .</b>	<b>201</b>
7.2.1. Простейшие методы Рунге—Кутты . . . . .	201
7.2.2. Общий случай . . . . .	202
7.2.3. Условия порядка для методов РК . . . . .	204
7.2.4. Явные методы Рунге—Кутты . . . . .	205
<b>7.3. Выбор шага численного интегрирования ОДУ . . . . .</b>	<b>208</b>
7.3.1. Адаптивный выбор шага . . . . .	208
<b>7.4. Многошаговые методы . . . . .</b>	<b>215</b>
7.4.1. Введение . . . . .	215
7.4.2. Явные методы Адамса . . . . .	216
7.4.3. Неявные методы Адамса . . . . .	218
7.4.4. Точность методов Адамса . . . . .	218
<b>7.5. Численное решение краевых задач.</b>	
<b>Метод стрельбы . . . . .</b>	<b>221</b>
7.5.1. Двухточечные краевые задачи . . . . .	221
7.5.2. Метод стрельбы . . . . .	222
<b>7.6. Решение краевых задач конечно-разностным методом . . . . .</b>	<b>225</b>
7.6.1. Общий нелинейный случай . . . . .	225
7.6.2. Линейный случай . . . . .	227
7.6.3. Устойчивость и сходимость метода сеток . . . . .	228

---

**ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ . . . . . 231**





## **Глава 1**

# **ВВЕДЕНИЕ**

## 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 распространённым 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

$$\pm \underbrace{d_0.d_1d_2\dots d_{p-1}}_m \times \beta^e, \quad 0 \leq d_i < \beta, \quad (1.1)$$

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»:

$$0.0001234 = 0.0012340 \times 10^{-1} = 1.2340000 \times 10^{-4}.$$

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  $\beta$ ,  $p$ -разрядной мантиссой, и  $e_{\min} \leq e \leq e_{\max}$  условимся обозначать  $\mathbb{F}_1(\beta, p, e_{\min}, e_{\max})$  или просто  $\mathbb{F}_1$ .

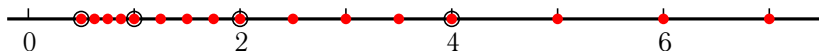
### 1.1.2. Двоичные числа с плавающей точкой

Рассмотрим подробно случай  $\beta = 2$ , так как именно такая арифметика используется в большинстве современных компьютеров. Возьмём

$p = 3$ ,  $e_{\min} = -1$ ,  $e_{\max} = 2$ . Все соответствующие положительные нормализованные ЧПТ приведены в таблице, они же изображены на рисунке. Степени двойки, соответствующие единичному значению мантиссы, обведены окружностями.

Таблица 1.1 Все положительные элементы множества  $\mathbb{F}_1(2, 3, -1, 2)$ .

$m \setminus e$	-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)



Приведённые данные наглядно демонстрируют следующие важные свойства, общие для всех нормализованных ЧПТ.

### Свойства множества нормализованных чисел с плавающей точкой

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  $\xi$  и соседними элементами  $\mathbb{F}_1$ .*
4. *Между нулём 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

Понятно, что если представление  $x$  в  $\beta$ -ичной системе счисления содержит больше  $p$  значащих цифр, мы не можем точно записать его в виде (1.1). В этом случае можно лишь приблизить  $x$  некоторым ЧПТ, которое в дальнейшем будем обозначать  $R(x)$ .

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

$$R : \mathbb{R} \rightarrow \mathbb{F}$$

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

Рассмотрим несколько способов задания  $R$ , считая  $\beta = 10$ ,  $p = 3$ .

- 1) Отбрасывание «лишних» знаков ( $R = R_d$ ):  
 $R_d(0.12345) = 1.23 \times 10^{-1}$ .
- 2) Округление вверх («школьное округление»,  $R = R_u$ ).  
 $R_u(543.21) = 5.43 \times 10^2$ ,  $R_u(5678) = 5.68 \times 10^3$ . В случае, когда запись  $x$  заканчивается на 5, его округляют до большего ЧПТ (вверх):  $R_u(23.45) = 2.35 \times 10^1$ .
- 3) Округление до чётного ( $R = R_e$ ). Этот способ отличается от предыдущего только трактовкой «спорного» случая, когда  $x$  находится ровно между двумя ЧПТ  $\underline{x}$  и  $\bar{x}$ . Оба эти приближения на самом деле равноправны, поэтому вместо того, чтобы всегда выбирать  $\bar{x}$ , мы будем с 50% вероятностью брать  $R_e(x) = \underline{x}$  либо  $R_e(x) = \bar{x}$ .  
 Реализовать это можно, всегда выбирая из  $\underline{x}$  и  $\bar{x}$  то, мантисса которого заканчивается на чётную цифру. Таким образом, получаем:  $R_e(23.45) = 2.34 \times 10^1$ , но  $R_e(23.55) = 2.36 \times 10^1$ .

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

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

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

*Если  $R = R_e$ , то либо  $x_i = x \forall i \geq 0$ , либо  $x_i = x_1 \forall i \geq 1$ .*

Заметим, что по стандарту IEEE 754 в современных ЭВМ используется  $R = R_e$ .

### 1.1.4. Расширение множества чисел с плавающей точкой

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

#### Денормализованные числа

Наличие «зазора» между нулём и минимальным положительным нормализованным ЧПТ (свойство 4) может привести к серьёзным проблемам на практике. Рассмотрим, например, ЧПТ  $x = 0.75$  и  $y = 0.625$  из рассмотренного модельного множества  $\mathbb{F}_1(2, 3, -1, 2)$ . Так как  $x - y = 0.125$ , при любом разумном способе округления мы имеем  $R(x - y) = 0$ . То есть, например, выполнение обычного кода типа

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

в нашем случае приведёт к плачевному результату.

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

**Определение 1.4.** Вещественные числа вида

$$0.d_1d_2\dots d_{p-1} \times \beta^{e_{\min}},$$

где  $d_i$  — произвольные  $\beta$ -ичные цифры, 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  $\mathbb{F}_0$ .

Введением денормализации мы сразу решаем две проблемы: получаем свойство

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

а также добавляем 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 определёнными: 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 не-числа (NaN-ы, от «not a number»), которые обозначают результаты некорректных арифметических операций, таких как, например, извлечение корня из отрицательного числа.

## Определение машинной арифметики

**Определение 1.5.** *Машинными числами* будем называть элементы множества

$$M = \mathbb{F}_0 \cup \mathbb{F}_1.$$

**Определение 1.6.** *Машинной арифметикой с плавающей точкой* (МАПТ) будем называть множество машинных чисел  $M$  в совокупности с правилом округления  $R$ .

### 1.1.5. Машинный эпсилон

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

**Определение 1.7.** *Абсолютной погрешностью округления* для числа  $x \in \mathbb{R}$  в данной МАПТ называется число

$$\Delta(x) = |x - R(x)|, \quad (1.2)$$

а *относительной погрешностью округления* — число

$$\delta(x) = \frac{|x - R(x)|}{|x|} = \frac{\Delta(x)}{|x|}. \quad (1.3)$$

Иногда относительную погрешность измеряют в процентах, умножая её на 100. Важно понимать, что при работе с машинной арифметикой уместнее всего оперировать относительными погрешностями, так как чем больше модуль  $x$ , тем больше  $\Delta(x)$  (см. свойство 3).

**Определение 1.8.** *Машинным эpsilon  $\varepsilon_M$  для МАПТ называется наименьшее положительное число  $\varepsilon$ , удовлетворяющее условию*

$$R(1 + \varepsilon) > 1.$$

### Свойства машинного эpsilon

1 (главное свойство). *Для всех вещественных  $x$  таких, что  $\xi_{\min} \leq |x| \leq \xi_{\max}$ , где  $\xi_{\min}$  и  $\xi_{\max}$  — минимальное и максимальное положительное нормализованное ЧПТ соответственно, справедливо*

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

2. *В диапазоне денормализованных чисел свойство 1, вообще говоря, не выполняется.*

3. *Значение машинного эpsilon зависит от правила округления и от количества бит в мантиссе.*

4. *Ни в коем случае не следует путать машинный эpsilon с машинным нулем.*

Таким образом, чем меньше величина  $\varepsilon_M$ , тем точнее вещественные числа приближаются в машинной арифметике.

### 1.1.6. Стандарт IEEE 754

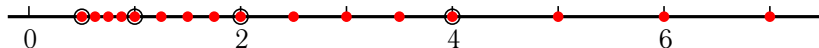
Международный стандарт *IEEE 754 floating point standard* определяет правила организации машинной арифметики с плавающей точкой. В настоящее время ему соответствует большинство вычислительных машин. В частности, наиболее распространённый тип данных, известный как *double precision floating point* (тип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

$\beta$	$p$	$e_{\min}$	$e_{\max}$	$R$
2	53	-1022	1023	$R_e$

### 1.1.7. Проблемы машинных вычислений

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

Приведём пример на модельной арифметике с  $\mathbb{F}_1(2, 3, -1, 2)$  и  $R = R_u$ : пусть  $x = 4.51$ ,  $y = 4.49$ . Имеем  $R(x) = 5$ ,  $R(y) = 4$ , и  $R(R(x) - R(y)) = 1$ , тогда как  $x - y = 0.02$ . Таким образом мы имеем 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.** При работе 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  $R((a \circ b) \circ c) \neq R(a \circ (b \circ c))$ .

## Резюме

Итак, при использовании машинной арифметики вычислитель всегда должен помнить о том, что как только он записывает в память ЭВМ число  $x$ , оно автоматически превращается в число  $\tilde{x} = R(x)$ , 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 свойству 1, почти всегда ограничена величиной  $\varepsilon_M$ .



## 1.2. Обусловленность задачи

### 1.2.1. Некорректные и плохо обусловленные задачи

Постоянное присутствие ошибок округления при работе с машиной 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 1.9.** Задача называется *корректно поставленной*, или просто *корректной*, если её решение (а) существует, (б) единственно и (в) непрерывно зависит от начальных данных. Если нарушено хотя бы одно из этих условий, задачу называют *некорректной*.

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

Существует ещё 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 даёт большую *относительную* погрешность в решении. Упор на относительную погрешность делается потому, что, как мы знаем из предыдущего раздела, при округлении вещественного числа  $x$  до машинного  $R(x)$  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  $P(a_0, \dots, a_n, x) = \sum_{i=0}^n a_i x^i$ . 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  $\tilde{a}_k$ . То есть 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:

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

Коэффициент

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

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

Численный пример:

$$\begin{aligned} p(x) = (x - 2)^{10} = x^{10} - 20x^9 + 180x^8 - 960x^7 + 3360x^6 - 8064x^5 + \\ + 13440x^4 - 15360x^3 + 11520x^2 - 5120x + 1024. \end{aligned}$$

Пусть  $x = 3$ . Тогда  $P(a_0, \dots, a_{10}, x) = 1$ . Изменим коэффициент  $a_9 = -20$  на 0.01 (что составляет 0.05%):  $\tilde{a}_9 = -19.99$ . Тогда получим  $P(a_0, \dots, \tilde{a}_9, a_{10}, x) = 197.83$ , что на 19683% больше точного значения.  $\square$

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 ещё точнее — от величины машинного эпсилон  $\varepsilon_M$ , так как эта величина ограничивает относительную погрешность округления.

▷<sub>1</sub> Пусть известно значение  $\varepsilon_M$ . Найдите нижнюю границу числа обусловленности для плохо обусловленной задачи.

### 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*

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|, \quad (1.4)$$

и *евклидовой нормой*

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

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

На следующей лекции мы приступим к рассмотрению задачи решения системы линейных алгебраических уравнений (СЛАУ) вида

$$Ax = b, \quad (1.6)$$

где  $A$  — невырожденная квадратная матрица размерности  $n$ ,  $x, b \in \mathbb{R}^n$ . Прежде, чем приступить к алгоритмам численного решения этой задачи, исследуем её 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*, то есть отображение  $A : \mathbb{R}^m \rightarrow \mathbb{R}^n$ , 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», которые многие считают аксиомой, есть ни что иное, как алгоритм вычисления композиции линейных операторов. В таком контексте вопрос об определении «величины» матрицы сводится к определению нормы соответствующего линейного оператора.

**Определение 1.10.** 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. То есть 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

$$\|Ax\| \leq \|A\| \|x\|. \quad (1.7)$$

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

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

$$\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|. \quad (1.8)$$

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

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

$$\|A\|_2 = \max_{1 \leq i \leq n} \sqrt{\lambda_i}, \quad (1.9)$$

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, очевидно, сводится к вычислению

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

Исследуем обусловленность этой задачи, считая параметром вектор правой части  $b$ . Действуем по схеме, аналогичной примеру 1.1. Относительная погрешность начальных данных имеет вид

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

а возмущенного решения, очевидно,

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

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

$$\begin{aligned} \delta_x &= \frac{\|A^{-1}(b - \tilde{b})\|}{\|A^{-1}b\|} \leq \frac{\|A^{-1}\| \|b - \tilde{b}\|}{\|A^{-1}b\|} \leq \\ &\leq \left[ \begin{array}{l} \|b\| = \|AA^{-1}b\| \leq \|A\| \|A^{-1}b\| \Rightarrow \\ \Rightarrow \|A^{-1}b\| \geq \|A\|^{-1} \|b\| \end{array} \right] \leq \|A^{-1}\| \|A\| \frac{\|b - \tilde{b}\|}{\|b\|}, \end{aligned}$$

то есть

$$\delta_x \leq \kappa(A) \delta_b, \quad (1.10)$$

где  $\kappa(A) = \|A^{-1}\| \|A\|$ .

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

**Определение 1.11.** Числом обусловленности невырожденной матрицы  $A$  называется число

$$\kappa(A) = \|A\| \|A^{-1}\|. \quad (1.11)$$

Если матрица  $A$  вырождена, её число обусловленности полагается равным бесконечности.

*Замечание 1.2.* Несмотря на то, что это определение ассоциировано с матрицей  $A$ , необходимо чётко понимать, что речь идёт 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  $\kappa_2(A)$  и  $\kappa_\infty(A)$ .

### Свойства числа обусловленности матрицы

1.  $\kappa(A) \geq 1$  :  $1 = \|A^{-1}A\| \leq \|A\| \|A^{-1}\|$ .
2.  $\kappa(AB) \leq \kappa(A)\kappa(B)$  :  $\|AB\| \|(AB)^{-1}\| \leq \|A\| \|A^{-1}\| \|B\| \|B^{-1}\|$ .

3. Если  $A = A^*$ , то  $\kappa_2(A) = \frac{|\lambda_{\max}|}{|\lambda_{\min}|}$ , 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  $A = \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix}$  равны 1, в то время как  $\kappa_2(A) \rightarrow \infty$  при  $|\alpha| \rightarrow \infty$ .

*Замечание 1.5.* Укажем на одно часто встречающееся заблуждение. Так как  $\kappa(A)$  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.

### 1.2.5. Резюме

- 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

Мы приступаем к изучению методов решения систем линейных алгебраических уравнений (СЛАУ)

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

которые будем записывать в матричном виде

$$Ax = b, \quad (2.1)$$

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

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 (то есть, к системе с тем же решением)

$$Vx = g, \quad (2.2)$$

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), очевидно, имеет вид

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

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

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

Такой вычислительный процесс 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.

Поговорим теперь о том, 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  $T_k$ ):

$$\underbrace{T_N \dots T_2 T_1}_V Ax = \underbrace{T_N \dots T_2 T_1}_g b.$$

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      // Прямой ход метода Гаусса
2:   for  $i = \overline{k+1, n}$  do
3:      $\underline{a}_i \leftarrow \underline{a}_i - \frac{a_{ik}}{a_{kk}} \underline{a}_k$ 
4:      $b_i \leftarrow b_i - \frac{a_{ik}}{a_{kk}} b_k$ 
5:   end for
6: end for
```

Понятно, что после этого для решения системы остается применить формулы обратной подстановки (2.3) (где  $v_{ij} = a_{ij}$ ,  $g_i = b_i$ ).

Этап алгоритма, определяемый строками 2–5, 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  $A^{(k+1)} = G_k A^{(k)}$ , где

$$G_k = \left[ \begin{array}{c|ccc} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & \alpha_{k+1}^{(k)} & 1 \\ & & \alpha_{k+2}^{(k)} & & \ddots \\ & & \vdots & & & \ddots \\ & & \alpha_n^{(k)} & & & & 1 \end{array} \right]. \quad (2.4)$$

Умножение произвольной матрицы  $M$  слева на матрицу  $G_k$  равносильно добавлению к  $i$ -й строке матрицы  $M$   $k$ -ой строки, умноженной на  $\alpha_i^{(k)}$  для всех  $i$  от  $k+1$  до  $n$ . Согласно алгоритму метода Гаусса (см. строку 3) имеем

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

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

Очевидно, что если хотя бы один из элементов

$$\theta_k = a_{kk}^{(k)}$$

равен нулю, то прямой ход в базовом алгоритме неосуществим. В дальнейшем элементы  $\theta_k$  будем называть *главными* или *ведущими*. Если же все главные элементы отличны от нуля, то приведённый 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  $G_1$  корректно определено и первый шаг метода Гаусса выполним если (и только если)  $a_{11}^{(1)} = a_{11} = |[A]_1| \neq 0$ .

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

$$\tilde{G}_k = G_k G_{k-1} \dots G_1, \quad \text{и} \quad A^{(k+1)} = \tilde{G}_k A.$$

Нетрудно увидеть, что матрицы  $\tilde{G}_k$  имеют блочный вид

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

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

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

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

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  $G_{k+1}$  (см. (2.4), (2.5)).

Из (2.6) имеем

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

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

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

□

**Трудоёмкость 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).

▷<sub>1</sub> Докажите это. Оцените трудоёмкость обратного хода.

### 2.1.3. Связь метода Гаусса и $LU$ -разложения

**Определение 2.1.**  $LU$ -разложением невырожденной матрицы  $A$  называется её 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.**  $\Rightarrow$  Пусть осуществим базовый алгоритм. Тогда из формулы (2.6) при  $k = n - 1$  получаем  $\tilde{G}_{n-1}A = A^{(n)}$ , причём по построению  $\tilde{G}_{n-1}$  — нижнетреугольная с единичной главной диагональю, а  $A^{(n)}$  — верхнетреугольная. Отсюда получаем  $A = LU$ , где  $L = (\tilde{G}_{n-1})^{-1}$ ,  $U = A^{(n)}$ .

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

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

□

### 2.1.4. Метод Гаусса с выбором главного элемента

Для того, чтобы выполнение алгоритма метода Гаусса не обрывалось при  $\theta_k = 0$  (и не только), *перед каждым шагом* метода применяется процедура, называемая *выбором главного элемента*. Суть процедуры: путём 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 ведущий 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 ведущий 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 ведущий 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 выбрать главный элемент?
- 2) Какой способ выбора главного элемента лучше?

Для ответа рассмотрим ещё раз матрицу  $G_k$  (2.4). Имеем

$$\kappa_{\infty}(G_k) = (1 + \max_i |\alpha_i^{(k)}|)^2, \quad (2.7)$$

▷<sub>2</sub> Докажите

откуда с учётом свойства 2 числа обусловленности имеем

$$\kappa_{\infty}(A^{(n)}) = \kappa_{\infty}(\tilde{G}_{n-1}A) \leq \kappa_{\infty}(A) \prod_{k=1}^{n-1} (1 + \max_i |\alpha_i^{(k)}|)^2.$$

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

Для исправления ситуации мы должны минимизировать величины (2.7). С учётом (2.5), получаем 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

$$AX = B, \quad (2.8)$$

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 4 заменить на

$$\underline{b}_i \leftarrow \underline{b}_i - \frac{a_{ik}}{a_{kk}} \underline{b}_k$$

(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.

Определитель матрицы также вычисляется с помощью метода Гаусса:

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

Однако, 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:

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

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

### 2.1.7. Метод прогонки

Рассмотрим СЛАУ

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

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

При выполнении алгоритма прогонки мы лишены возможности выбора главного элемента, так как при этом нарушилась бы трёхдиагональная 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

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

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

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

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

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

то алгоритм прогонки выполним.

## 2.2. LU-разложение

### 2.2.1. Базовый алгоритм $LU$ -разложения

Рассмотрим последовательность СЛАУ

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

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, причём к матрице  $A$  будут применяться *одни и те же* преобразования  $G_k$ .

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

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

▷<sub>1</sub> 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

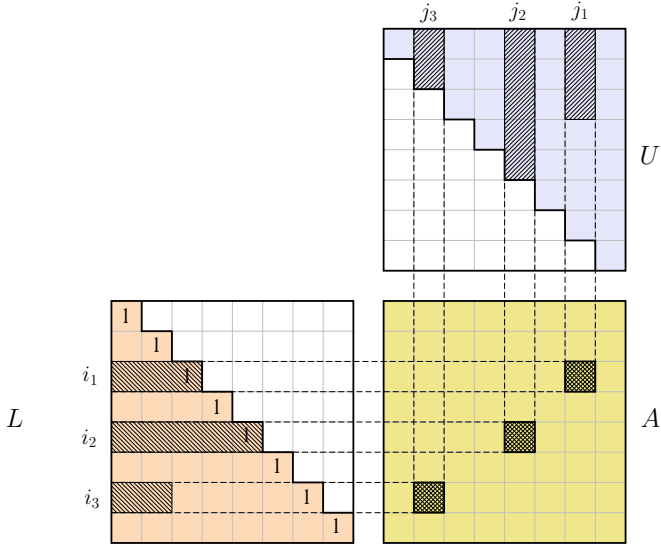
$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \ell_{21} & 1 & 0 & \cdots & 0 \\ \ell_{31} & \ell_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \cdots & 1 \end{bmatrix}}_L \underbrace{\begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{bmatrix}}_U = \underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}}_A.$$

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

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

Треугольная структура матриц  $L$  и  $U$  влечет следующее тождество (см. иллюстрацию):

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



Выделяя последние слагаемые в суммах (2.14), получаем

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} \ell_{ik} u_{kj} \quad \text{при } i \leq j; \quad (2.15)$$

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

или

$$\tilde{a}_{ij} = \begin{cases} a_{ij} - \sum_{k=1}^{i-1} \tilde{a}_{ik} \tilde{a}_{kj} & \text{при } i \leq j; \\ \frac{1}{\tilde{a}_{jj}} \left( a_{ij} - \sum_{k=1}^{j-1} \tilde{a}_{ik} \tilde{a}_{kj} \right) & \text{при } i > j. \end{cases} \quad (2.17)$$

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

### Базовый алгоритм $LU$ -разложения

```

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

```

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

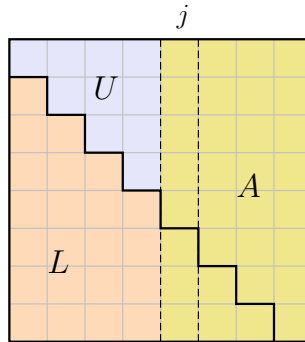


Рисунок 2.1 Вид матрицы системы перед  $j$ -м шагом алгоритма  $LU$ -разложения.

Рассмотрим матрицу  $A^{(j)}$ , которая получается из  $A$  после  $(j-1)$ -го шага разложения (рис. 2.1). 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  $u_{jj}$  ещё 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

$$u_{jj} = a_{jj} - \sum_{k=1}^{j-1} \ell_{jk} u_{kj} \quad (2.18)$$

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

- 1) Вычисляем кандидатов на роль ведущего элемента: для всех  $i$  от  $j$  до  $n$  вычисляем  $a_{ij}^{(j)} = \tilde{a}_{ij}$  по второй формуле из (2.17), только без деления на  $\tilde{a}_{jj}$ ;
- 2) Среди полученных значений  $a_{ij}^{(j)}$  для  $i \geq j$  выбираем максимальный по модулю  $a_{i^*j}^{(j)}$ ;
- 3) Меняем местами  $j$ -ю и  $i^*$ -ю строки матрицы  $A^{(j)}$ ;
- 4) Для всех  $i$  от  $j + 1$  до  $n$  делим  $a_{ij}^{(j)}$  на  $a_{jj}^{(j)}$ .

**Замечание 2.1.** 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  $p$ . Кроме этого, для корректного вычисления определителя необходимо возвращать  $s = \pm 1$  — значение чётности числа перестановок.

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

## 2.3. Метод квадратного корня

### 2.3.1. Разложение Холецкого

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

$$A = R^*DR, \quad (2.19)$$

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

*Доказательство.* Так как  $||A||_k \neq 0$ , по теореме 2.2 существует  $LU$ -разложение

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

откуда  $L = U^*(L^*U^{-1}) = U^*H$ , и

$$A = LU = U^*HU. \quad (2.20)$$

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

$$H = \text{diag}(h_1, h_2, \dots, h_n).$$

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

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

и тождество (2.20) даёт

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

что и требовалось доказать.  $\square$

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

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

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

### Свойства положительно определённых матриц

1. Если  $A > 0$  то  $|[A]_k| \neq 0 \quad \forall k = \overline{1, n}$ .
2. Если  $A^* = A$ , то  $A > 0 \Leftrightarrow$  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

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

Так как  $R$  невырождена,  $\forall y \in \mathbb{C}^n \quad \exists x : y = Rx$ , то есть

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

Возьмём в качестве  $y$   $k$ -й единичный орт:  $y_i = \delta_{ik}$ . Тогда с учётом того, что  $|d_k| = 1$ , из (2.21) получаем  $d_k = 1 \quad \forall k = \overline{1, n}$ .  $\square$

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

$$A = R^T DR, \quad (2.22)$$

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

### 2.3.2. Алгоритм метода

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

$$A = R^T DR.$$

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

$$a_{ij} = \sum_{k=1}^{\min(i,j)} \ell_{ik} u_{kj} = \begin{bmatrix} \ell_{ik} = r_{ki}, \\ u_{kj} = d_k r_{kj} \end{bmatrix} = \sum_{k=1}^{\min(i,j)} d_k r_{ki} r_{kj}. \quad (2.23)$$

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

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

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

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

▷<sub>1</sub> Запишите алгоритм прямого и обратного хода метода квадратного корня.

▷<sub>2</sub> 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 её верхний треугольник.
- 2) Аналогично случаю  $LU$ -разложения расчётные 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  $R^T DRx = b$  осуществляется путём применения двух обратных подстановок:  $R^T y = b$ , затем  $Rx = Dy$  (так как  $D = D^{-1}$ ).



## 2.4. Параллельная реализация метода Гаусса

### 2.4.1. 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. В таких системах несколько процессоров работают с общей на всех памятью. Системы с распределенной памятью состоят из нескольких узлов — процессоров, каждый из которых имеет свою собственную память. Такие системы менее производительны, так как требуют выполнения операций обмена данными между узлами. С другой стороны, в силу очевидных причин 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.

Сейчас мы рассмотрим один из способов 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

$$n = mM, \quad m \in \mathbb{N}.$$

Пронумеруем все узлы от 0 до  $M - 1$ .

### 2.4.2. Распределение данных между узлами

Так как мы работаем с распределенной памятью, важно выбрать грамотный способ хранения данных: понятно ведь, что полностью хранить матрицу  $A$  на каждом узле — не рационально (в самом страшном случае матрица целиком просто может не поместиться). Поступаем следующим образом: каждый узел будет хранить  $m$  столбцов матрицы  $A$ , но столбцы распределяем не подряд, а циклически. То есть, узел номер 0 хранит столбцы с номерами  $1, M+1, 2M+1, \dots$ , узел номер 1 — столбцы с номерами  $2, M+2, 2M+2, \dots$ , и так далее (см. рисунок). Сделано это



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

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

Здесь  $\underline{a}_i$ , понятное дело, —  $i$ -я строка матрицы  $A_p$ .

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

#### 2.4.4. Обратный ход

Обратный ход занимает гораздо меньше ресурсов, чем прямой, поэтому его можно выполнить последовательно<sup>2</sup>. То есть просто собираем столбцы со всех узлов на один и выполняем стандартный обратный ход.

---

<sup>2</sup>Естественно, существуют и параллельные алгоритмы обратного хода, но их описание выходит за рамки нашего курса.

## 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*, то есть 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<sup>3</sup>. То есть 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

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

такой, что если существует её предел  $x^*$ ,

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

то  $x^* = A^{-1}b$  — искомое решение. При этом ключевым требованием является то, что все приближения строятся по единому правилу:

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

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

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

### 2.5.2. Итерационные методы общего вида

#### Принцип построения

Рассмотрим СЛАУ

$$Ax = b$$

---

<sup>3</sup>Естественно, чем выше требуемая точность, тем больше вычислительной работы придется проделать.

с невырожденной матрицей  $A$ . Приведем данную систему к виду

$$x = Bx + g \quad \Leftrightarrow \quad Ax = b, \quad (2.25)$$

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

$$x^{k+1} = Bx^k + g, \quad k = 0, 1, 2, \dots \quad (2.26)$$

Нетрудно заметить, что если такой процесс сходится, то есть если 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.

### 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

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

будем называть *методом простой итерации* (МПИ).

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

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

Выражая из него  $x_i$ , получаем

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

Для того, чтобы записать это тождество в векторном виде, рассмотрим разбиение матрицы  $A$  на слагаемые согласно рис. 2.2:

$$A = L + D + R. \quad (2.29)$$

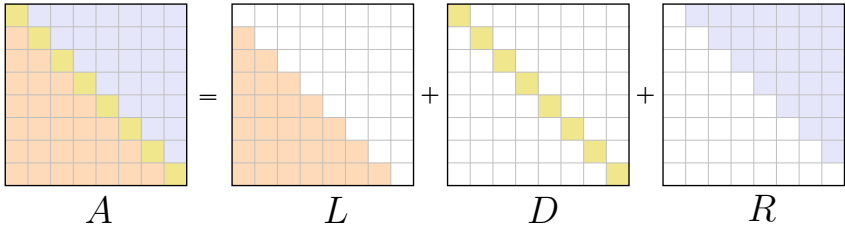


Рисунок 2.2

Тогда (2.28) примет вид

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

где

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

Соответствующий итерационный метод

$$x_i^{k+1} = \frac{1}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij} x_j^k \right), \quad i = \overline{1, n}, \quad k = 0, 1, 2, \dots \quad (2.31a)$$

называется *методом Якоби*. Его векторная форма имеет вид

$$x^{k+1} = B_J x^k + g_J, \quad (2.31b)$$

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  $k$ -ой итерации метода Якоби:

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

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 соот-

ветствующие компоненты текущего приближения  $x^k$ , поэтому их можно использовать в сумме (2.31a):

$$x_i^{k+1} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^n a_{ij} x_j^k \right) \quad (2.32a)$$

Векторный вариант (2.32a) имеет вид

$$\begin{aligned} x^{k+1} &= D^{-1}(b - Lx^{k+1} - Rx^k), \quad \text{откуда} \\ x^{k+1} &= B_S x^k + g_S, \\ B_S &= -(D + L)^{-1}R, \quad g_S = (D + L)^{-1}b. \end{aligned} \quad (2.32b)$$

Формулы (2.32a), (2.32b) определяют *метод Гаусса–Зейделя*.

▷<sub>1</sub> Примените идею построения метода Гаусса–Зейделя к методу простой итерации (2.27). Запишите векторную и скалярную формы метода.

▷<sub>2</sub> 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 путём взвешенного осреднения текущего приближения и приближения, построенного по методу Гаусса–Зейделя:

$$x_i^{k+1} = (1 - \omega)x_i^k + \frac{\omega}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^n a_{ij} x_j^k \right), \quad (2.33a)$$

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

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

*Замечание 2.2.* При  $\omega = 1$  метод релаксации, очевидно, превращается в метод Гаусса–Зейделя.

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

### 2.5.3. 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*

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

*сходится к нулевой матрице.*

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

$$x^{k+1} - x^k = B(x^k - x^{k-1}) = \dots = B^k(x^1 - x^0) = B^k(\underbrace{(B - I)x_0 + g}_v).$$

Так как последовательность  $(x^k)$  сходится, имеем  $\|x^{k+1} - x^k\| \rightarrow 0 \forall x_0 \in \mathbb{R}^n$ , откуда  $\|B^k v\| \rightarrow 0 \forall v \in \mathbb{R}^n$ , то есть  $B^k \rightarrow 0$ .

$\Leftarrow$  Пусть  $B^k \rightarrow 0$ . Имеем

$$\begin{aligned} \|x^k - x^*\| &= \|Bx^{k-1} + g - Bx^* - g\| = \|B(x^{k-1} - x^*)\| = \dots = \\ &= \|B^k(x^0 - x^*)\| \leq \|B^k\| \|x^0 - x^*\|, \end{aligned}$$

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



**Определение 2.4.** Пусть  $A$  — квадратная матрица. Вектор  $\xi \neq 0 \in \mathbb{C}^n$  называется *собственным вектором* матрицы  $A$ , если существует  $\lambda \in \mathbb{C}$  такое, что

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

Число  $\lambda$  называется *собственным значением*, соответствующим  $\xi$ . 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*

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

*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*, то есть представима в виде

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

где матрица  $X$  невырождена, а

$$D = \text{diag}(\lambda_1, \dots, \lambda_n),$$

$\lambda_i$  — собственные значения  $B$ .

В этом случае имеем

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

или

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

откуда

$$B^k \rightarrow 0 \Leftrightarrow D^k \rightarrow 0 \Leftrightarrow |\lambda_i|^k \rightarrow 0 \Leftrightarrow \rho(B) < 1. \quad \square$$

**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  $\lambda$  матрицы  $B$  и соответствующий ему собственный вектор  $\xi$  с единичной нормой. Тогда в любой подчиненной матричной норме имеем

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

откуда

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

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

*Замечание 2.3.* 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 сходится процесс.

*Замечание 2.4.* 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  $B_J = I - D^{-1}A$ :

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

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, то есть

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

Отсюда сразу следует, что

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

то есть метод Якоби сходится по следствию 2.3. □

*Замечание 2.5.* 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

### 2.6.1. Координатный формат

**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  $n_z$ .

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

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

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

Длина всех трёх 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 задаётся 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  $i$ -ой строке (столбце).

Формат CSR является одним из наиболее популярных.

▷<sub>1</sub> Укажите достоинства и недостатки этого формата по сравнению с координатным форматом.

**Алгоритм умножения матрицы в формате CSR на вектор**    Ввод:  $n$ , **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
```

▷<sub>2</sub> 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 фор-

маты, очевидно, 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) Изменён 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 (назовём 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
<b>AA</b>	2.	0.	9.	3.	6.	×	5.	7.	1.	4.	8.
<b>IJ</b>	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 2.6.* 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
```

▷<sub>3</sub> 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  $\mathbb{C}$ . Как мы помним, 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  $\lambda$  называется *собственным значением*, соответствующим  $x$ . Множество всех собственных значений  $A$  называется *спектром* и обозначается  $\sigma(A)$ .

*Замечание 3.1.* Собственный вектор определён с точностью до постоянного множителя:

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

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

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

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

Если  $\lambda$  — корень многочлена  $\varphi$  кратности  $k$ , то говорят, что *алгебраическая кратность*  $\lambda$  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  $(1 - \lambda)^n = 0$ , то есть 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  $(\xi, 0)^T$ , то есть 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:*

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

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 (ПСЗ) 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 ей собственный вектор вещественной диагоналируемой матрицы. Он используется, в частности, в алгоритме PageRank, который применяет компания Google при расчете релевантности страницы поисковому запросу.

Пусть  $A$  — диагоналируемая матрица,  $\lambda_1, \dots, \lambda_n$  — упорядоченные по убыванию модуля собственные значения,  $x^1, \dots, x^n$  — соответствующий базис из собственных векторов. Рассмотрим несколько возможных случаев.

### 3.2.1. Случай 1

Пусть  $\lambda_1 \in \mathbb{R}$ ,  $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$ .

Разложим базису  $\{x^i\}$  произвольный вектор  $y^0 \in \mathbb{C}^n$ :  $y^0 = \sum_{i=1}^n \alpha_i x^i$ . Предположим, что  $\alpha_1 \neq 0$  и рассмотрим последовательность  $(y^k)$ :

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

Тогда

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

Значит, при  $k \rightarrow \infty$  имеем

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

то есть вектор  $y^k$  всё сильнее приближается к собственному подпространству, соответствующему  $x^1$ . Для практического применения, естественно, 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  $k \geq 1$ , а также

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

▷<sub>1</sub> Докажите это.

Обозначим

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

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

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

Кроме этого, так как максимальная по модулю компонента вектора  $\xi^1$  равна 1, из тождества

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

имеем

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

или просто

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

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

```

 $u = y^0$ 
while не сошлось do
     $v \leftarrow Au$ 
     $\lambda \leftarrow \max(v)$ 
     $u \leftarrow v/\lambda$ 
end while

```

На выходе будем иметь  $u \approx \xi^1$ ,  $\lambda \approx \lambda_1$ .

Из (3.1) видно, что степенной метод, сходится со скоростью геометрической прогрессии со знаменателем  $\left| \frac{\lambda_2}{\lambda_1} \right|$ .

### 3.2.2. Случай 2

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

Так как матрица  $A$  по условию диагонализуема, геометрическая кратность  $\lambda_1$  равна  $m$ , то есть собственные векторы  $x^1, \dots, x^m$  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 остаётся без изменений. Последовательность  $(u^k)$  сходится к какому-то вектору  $u \in X_m$  — он ничем не хуже других собственных векторов  $x^1, \dots, x^m$ .

### 3.2.3. Случай 3

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

В этом случае (3.1) превращается в

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

откуда получаем

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

то есть 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  $\text{span}(x^1, x^2)$ , причём, в отличие от предыдущего случая, ни один из них не является, вообще говоря, собственным.

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

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

Согласно (3.2) имеем

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

а

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

Таким образом можно найти  $\lambda_1$ .

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

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

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

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

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

Согласно вышесказанному можно построить следующую модификацию степенного алгоритма:

```

u = y0
while не сошлось do
    v ← Au;   u ← Av
    λ ← √max(u)
    v ← λv + u
    u ← u / max(u);   v ← 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. Случай 4

Пусть  $\lambda_2 = \bar{\lambda}_1 \in \mathbb{C}$ ,  $|\lambda_1| > |\lambda_3| \geq \dots \geq |\lambda_n|$ .

В этом случае разложение вектора  $y^0$  по базису  $x^i$  имеет вид

$$y^0 = \alpha_1 x_1 + \bar{\alpha}_1 \bar{x}_1 + \sum_{i=3}^n \alpha_i x^i.$$

Обозначим  $\lambda_1 = \rho e^{i\phi}$ ,  $\alpha_1 x_j^1 = r_j e^{i\theta_j}$  тогда

$$y^k \sim \lambda_1^k \alpha_1 x_j^1 + \bar{\lambda}_1^k \bar{\alpha}_1 \bar{x}_j^1 = \rho^k r_j (e^{\phi k + \theta_j} + e^{-i(\phi k + \theta_j)}) = 2\rho^k r_j \cos(\phi k + \theta_j).$$

Компоненты вектора  $u^k$  осциллируют и не стремятся ни к какому пределу. В этом случае всё равно можно найти  $\lambda_1$  и  $x_1$ , см. [Дж. Уилкинсон. Алгебраическая проблема собственных значений (Наука, 1970.)].

### 3.2.5. Общий случай

Вообще говоря, степенной метод можно применять не только для диагоналируемых матриц. Если геометрическая и алгебраическая кратности  $\lambda_1$  совпадают, этот метод будет работать. В противном случае метод тоже может сходиться, но очень медленно.

### 3.2.6. Степенной метод со сдвигом

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

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

За счёт выбора  $\alpha$  и  $\beta$  можно сместить спектр таким образом, что  $\mu_j = \alpha \lambda_j + \beta$  станет максимальным собственным значением матрицы  $\alpha A + \beta I$ . Найдя  $\mu_j$  степенным методом, соответственно, найдём и  $\lambda_j$ .

Опишем схему нахождения минимального собственного значения в случае вещественного спектра.

- 1) Пусть  $0 \leq \lambda_i < m$  (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  $\lambda_i$  отрицательны.
- 2) Если  $\lambda_i$  имеют различные знаки, рассмотрим матрицу  $A^2$ . У неё те же собственные векторы, а собственные значения равны  $\lambda_i^2$ . Таким образом задача свелась к первому случаю.



### 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, то есть преобразование  $A \mapsto A'$  должно сохранять спектр. Другими словами, это преобразование должно быть *преобразованием подобия*.

**Определение 3.2.** Преобразованием подобия квадратной матрицы  $A$  называется преобразование

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

где  $S$  — невырожденная матрица.

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:

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

то есть 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, то есть, в частности, элементарные

преобразования. Напомним, что эти преобразования бывают двух типов. Элементарное преобразование *первого рода* — это умножение  $i$ -ой строки матрицы  $A$  на произвольный скаляр  $\alpha \neq 0$ . Данное преобразование эквивалентно умножению слева на *элементарную матрицу первого типа*

$$T_i^\alpha = \text{diag}(1, \dots, 1, \alpha, 1, \dots, 1),$$

где элемент  $\alpha$  находится на  $i$ -ой позиции.

Элементарное преобразование *второго рода* это добавление к  $i$ -ой строке матрицы  $A$  её  $j$ -ой строки, умноженной на  $\alpha$ . Такое преобразование задается матрицей  $S_{ij}^\alpha$ , которая представляет собой матрицу с единицами на главной диагонали и элементом  $\alpha$  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», то есть чему эквивалентны операции

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

Начнем с первого.

**Элементарное преобразование подобия первого типа.** Для начала вычислим обратную к матрице  $T_i^\alpha$ . Очевидно, что

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

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

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

то есть *если к  $i$ -ой строке (столбцу) матрицы  $A$  прибавлена  $j$ -я ее строка (столбец), умноженная на  $\alpha$ , то для сохранения спектра необходимо после этого вычесть из  $j$ -ого столбца (строки) полученной матрицы  $i$ -й столбец (строку), умноженный на  $\alpha$ .*

**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  $i$ -ой и  $j$ -ой строками. Понятно, что

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

Кроме этого,

$$AP_{ij} = (P_{ij}A^T)^T$$

так что 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 матрицы  $A$ .

Рассмотрим матрицу вида

$$\begin{bmatrix} p_1 & p_2 & \cdots & p_{n-1} & p_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \quad (3.6)$$

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

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

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

Идея метода Данилевского проста: с помощью элементарных преобразований подобия привести данную матрицу  $A$  к форме Фробениуса  $A'$ . Так как преобразования подобия сохраняют спектр матрицы, характеристические многочлены матриц  $A$  и  $A'$  будут совпадать, то есть искомым 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 в полученным выше правилом, нужно дополнить это преобразование до преоб-

разования подобия, то есть умножить третью строку на  $\alpha$ . В результате указанных операций получаем 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  $j \neq 3$  вычитаем 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  $\alpha$ . Получаем в итоге

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

3. Аналогично поступаем 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:    $\alpha \leftarrow a_{i, i-1}$ 
3:    $a_{i-1} \leftarrow a_{i-1}/\alpha$ ;  $\underline{a_{i-1}} \leftarrow \alpha \underline{a_{i-1}}$ 
4:   for  $j \neq i$  do
5:      $\alpha \leftarrow a_{ij}$ 
6:      $a_j \leftarrow a_j - \alpha a_{i-1}$ 
7:      $\underline{a_{i-1}} \leftarrow \underline{a_{i-1}} + \alpha \underline{a_j}$ 
8:   end for
9: end for

```

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

После выполнения приведенного алгоритма (в невырожденном случае) на месте матрицы  $A$  будет находиться 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  $i$ -ом этапе не удаётся 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, причём  $A_2$  имеет форму Фробениуса. Тогда имеем

$$|A - \lambda I| = |A_{n-i} - \lambda I_{n-i}| \cdot |A_2 - \lambda I_2|.$$

Второй множитель мы можем вычислить сразу, поэтому остаётся лишь привести к форме Фробениуса матрицу  $A_1$ .

## 3.4. Метод вращений Якоби

### 3.4.1. Преобразование вращения

Пришло время изучить важный тип линейных преобразований (матриц) — преобразования вращения, которые являются частным случаем ортогональных преобразований.

#### Ортогональные преобразования

**Определение 3.3.** Линейное преобразование  $A : \mathbb{R}^n \rightarrow \mathbb{R}^n$  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 равен 1.*

3. *Число обусловленности ортогональной матрицы в спектральной норме равно 1.*

### Преобразование вращения

**Определение 3.4.** Матрицей элементарного вращения называется матрица вида

$$V_{pq}(\theta) = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & c & s \\ & & & -s & c \\ & & & & 1 & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \begin{matrix} p \\ q \end{matrix}, \quad (3.8)$$

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

▷<sub>1</sub> 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, так как

- $A$  диагонализуема: существует матрица  $X$  такая, что

$$X^{-1}AX = D = \text{diag}(\lambda_1, \dots, \lambda_n), \quad (3.9)$$

причем  $X$  — ортогональна.

- все  $\lambda_i$  вещественны.

Также из (3.9) имеем  $AX = XD$ , то есть 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, то есть решает полную проблему собственных значений для такой матрицы. Суть метода состоит в построении последовательности матриц

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

где  $X_m$  — матрицы элементарного вращения (см. (3.8)), которые строятся таким образом, что

$$A^{(m)} \xrightarrow{m \rightarrow \infty} D, \quad (3.10)$$

где  $D$  — некоторая диагональная матрица. Таким образом, для достаточно большого  $M$  мы будем иметь

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

*Главный вопрос:* каким образом выбирать  $X_m$ , чтобы достичь сходимости?

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

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

▷<sub>2</sub> Докажите, что ортогональные преобразования подобия сохраняют норму Фробениуса.

Рассмотрим величины

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

По определению имеем

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

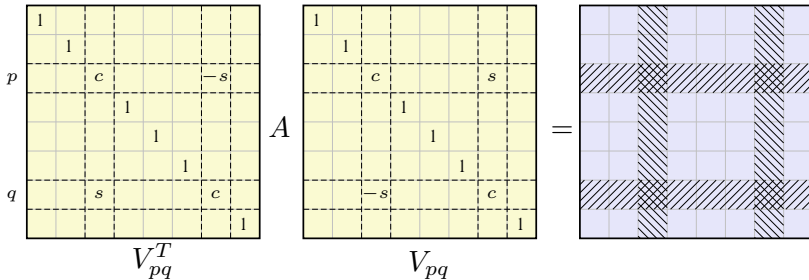
Условие сходимости (3.10) можно теперь записать в эквивалентной форме

$$\text{off}(A^{(m)}) \xrightarrow{m \rightarrow \infty} 0.$$

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  $\theta$ . Чтобы понять, каким образом следует выбирать эти параметры, рассмотрим как преобразование подобия

$$A \mapsto A' = V_{pq}^T A V_{pq} \quad (3.11)$$

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  $(q, q)$ . Введём следующие обозначения:  $a_{pp} = a$ ,  $a_{qq} = b$ ,  $a'_{pp} = \alpha$ ,  $a'_{qq} = \beta$ . Тогда

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

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

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

Так как ортогональное преобразование подобия сохраняет норму Фробениуса, получаем

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

Используем это тождество в (3.12):

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

что с учётом

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

равносильно

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

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

**Базовый алгоритм метода Якоби.** Пока  $\text{off}(A)$  не достаточно 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

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

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

$$\text{off}(A) \leq 2Ne^2,$$

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

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

что означает

$$\text{off}(A^{(m)}) \leq \left(1 - \frac{1}{N}\right)^m \text{off}(A). \quad (3.16)$$

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. Расчётные формулы метода

Для окончательного определения метода осталось вывести формулы, по которым вычисляются элементы матрицы  $V_{pq}$  во втором пункте алгоритма. Из (3.13) получаем

$$\alpha = a c^2 + b s^2 - 2e sc, \quad (3.17a)$$

$$\beta = a s^2 + b c^2 + 2e sc, \quad (3.17b)$$

$$\varepsilon = e(c^2 - s^2) + (a - b)sc = 0. \quad (3.17c)$$

Для решения уравнения (3.17c) введём переменные

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

После простых преобразований из (3.17с) получаем

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

и

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

Использование этой формулы на практике приводит к большим ошибкам округления, поэтому нужно ее 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. Он равен

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

После этого вычисляем

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

Теоретически на этом этапе нам остаётся с помощью найденных значений  $s$  и  $c$  вычислить  $A'$  по формуле (3.11). Однако для того, чтобы метод был эффективен, следует организовать вычисления следующим образом.

Как мы видели, 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) имеем

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

Для вычислительной устойчивости нужно представить вышеперечисленные формулы в виде

$$a'_{ij} = a_{ij} + [\text{какая-то поправка}].$$

Так, из (3.17a), (3.17b) с учётом (3.17c) получаем

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

а вместо (3.21) имеем

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

▷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  $V_{pq}$ , очевидно, в память не записываются, используются только величины  $c$  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:  $a_{12}, \dots, a_{1n}, a_{23}, \dots, a_{2n}, \dots, a_{n-1,n}$ . Обычно для получения решения в пределах машинной точности достаточно 5–6 таких проходов. При этом на первых двух-трёх проходах, если модуль  $a_{pq}$  достаточно мал, его пропускают.

## 3.5. QR-алгоритм

QR-алгоритм в настоящее время является одним из наиболее известных и популярных методов вычисления всех (в том числе и комплексных, конечно) 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 будем иметь

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

где матрицы  $Q_k$  — ортогональные, а  $R$  — верхнетреугольная. Отсюда получаем 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

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

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  $n = 6, k = 3$ .

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

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:    $\gamma \leftarrow a_{ik} / a_{k+1,k}$ 
3:    $\underline{a}_i \leftarrow \underline{a}_i - \gamma \underline{a}_{k+1}$ 
4:    $a_{k+1} \leftarrow a_{k+1} + \gamma a_i$ 
5: end for
```



Понятно, что 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  $i = k + 1, n$ . После перестановки строк переставляются и соответствующие столбцы.

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

*Замечание 3.2.* Вместо элементарных в доказательстве теоремы можно использовать ортогональные преобразования.

### 3.5.2. Общая схема QR-алгоритма

Теперь мы готовы рассмотреть сам алгоритм. Построим последовательность матриц

$$A_0 = A, \quad A_1, \quad A_2, \quad \dots,$$

по следующему правилу.

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

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, упрощённая формулировка). *Последовательность матриц  $\{A_k\}_{k=0}^{\infty}$ , построенная по описанному выше правилу, почти всегда сходится к верхнетреугольной матрице  $\tilde{R}$ , у которой на диагонали стоят либо  $\lambda_i$  — собственные значения  $A$ , либо блоки вида  $\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}$ , собственные значения которых равны комплексно-сопряжённым собственным значениям матрицы  $A$ .*

*Замечание 3.3.* Коэффициенты блоков  $\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}$  не обязаны сходиться к какому-то пределу, сходятся их собственные значения.

*Замечание 3.4.* QR-алгоритм *никогда* не применяется к «неподготовленной» матрице  $A$ , потому что это слишком дорого. Сначала матрицу необходимо привести в форме Хессенберга и лишь потом начинать итерации QR-алгоритма.

### 3.5.3. QR-разложение для формы Хессенберга

Пусть  $A$  имеет форму Хессенберга. Для приведения её 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:

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

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  $n = 4$ :

$$\begin{aligned} \begin{bmatrix} c_1 & s_1 & & \\ -s_1 & c_1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times \\ * & \times & \times & \times \\ & * & \times & \times \\ & & * & \times \end{bmatrix} &= \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & * & \times & \times \\ & & & * & \times \end{bmatrix}, \\ \begin{bmatrix} 1 & & & \\ & c_2 & s_2 & \\ & -s_2 & c_2 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & * & \times & \times \\ & & * & \times \end{bmatrix} &= \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & * & \times \end{bmatrix}, \\ \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & c_3 & s_3 \\ & & -s_3 & c_3 \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & & * & \times \end{bmatrix} &= \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{bmatrix}. \end{aligned}$$

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, то есть 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).

▷<sub>1</sub> Докажите.

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  $2 \times 2$ :

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

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 (см. (3.20))

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

▷<sub>2</sub> 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 следующий

#### Примитивный QR-алгоритм

- 1) Приводим  $A$  к форме Хессенберга  $A_0$ .
- 2) Выполняем итерации QR-алгоритма с начальной матрицей  $A_0$  до тех пор, пока не достигнем нужной точности.
- 3) Извлекаем из полученной матрицы  $A_N \approx \tilde{R}$  собственные значения.

Обсудим данный алгоритм. Первый пункт мы разобрали (теорема 3.1). То, как выполнять QR-итерации с использованием преобразований вращения (пункт 2), тоже разобрали (раздел 3.5.3).

Для полной ясности приведем пример «извлечения» собственных значений в пункте 3 (см. еще раз 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:  $\alpha_1$  и  $\alpha_4$ . 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 3.5.* 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**

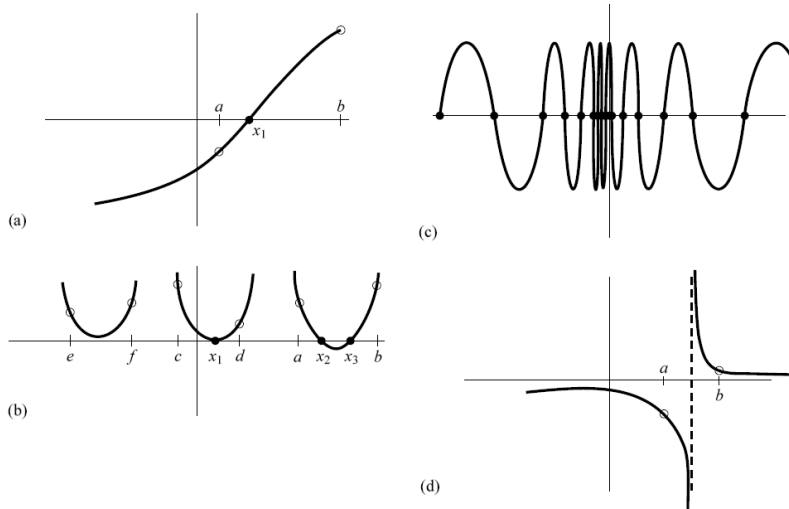
## 4.1. Решение нелинейных уравнений

### 4.1.1. Введение

Мы приступаем к рассмотрению задачи нахождения корней нелинейных уравнений вида

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

**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 приступить к её решению, необходимо провести 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. Основным средством, которое мы будем применять для отделения корней, является 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 одному правилу* последовательности чисел  $(x_k)$  таким образом, чтобы  $x_k \rightarrow x^*$  при  $k \rightarrow \infty$ , где  $x^*$  — искомый корень уравнения  $f(x) = 0$ . Большинство этих методов можно будет представить в виде

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

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

$$x = \varphi(x). \quad (4.3)$$

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

Основная характеристика качества метода — скорость, с которой  $\varepsilon_k$  стремится к 0.

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

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

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 рецепт определения порядка сходимости.

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

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

Тогда итерационный процесс (4.2) имеет порядок сходимости  $p$  и имеет место оценка

$$|\varepsilon_{k+1}| \leq \frac{M_p}{p!} |\varepsilon_k^p|, \quad (4.4)$$

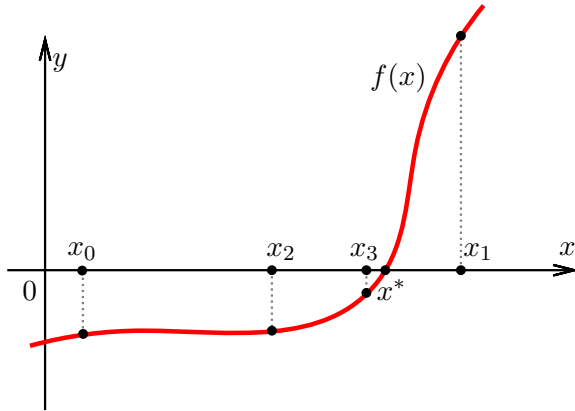
где  $M$  — максимум  $|\varphi^{(p)}(x)|$  в некоторой фиксированной окрестности  $x^*$ .

*Доказательство.* По формуле Тейлора имеем

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

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

### 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  $[a, b]$ .



**Метод бисекции (дихотомии), общая схема**

```

1: while  $|b - a| > 2\varepsilon$  do
2:    $x \leftarrow (a + b)/2$ 
3:   if  $f(x)f(a) < 0$  then
4:      $b \leftarrow x$ 
5:   else
6:      $a \leftarrow x$ 
7:   end if
8: end while

```

Здесь  $\varepsilon$  — точность, с которой требуется найти корень.

▷<sub>1</sub> Определите порядок сходимости метода бисекции.

**4.1.4. Метод Ньютона**

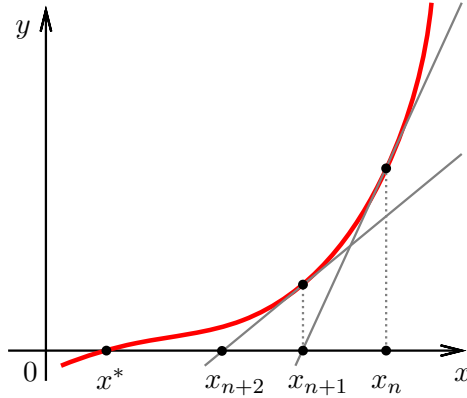
В случаях, когда известен вид  $f'(x)$ , метод Ньютона является одним из наиболее эффективных. Рассмотрим некоторое приближение  $x_k$ . Приближим график функции  $f$  касательной в точке  $x_k$ :

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

В качестве следующего приближения к решению уравнения  $f(x) = 0$  возьмём корень уравнения  $y(x) = 0$ :

$$\boxed{x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}}. \quad (4.5)$$

Формула (4.5) и определяет метод Ньютона.



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

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

Из (4.6) по лемме 2 легко получить порядок сходимости метода Ньютона:

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

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

$$\varphi'(x^*) = 0. \quad (4.8)$$

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

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

то есть

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

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

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

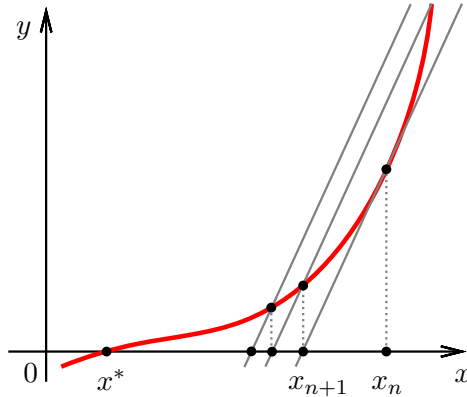
следовательно по формуле (4.4) получаем, что в случае корня кратности  $m$  метод Ньютона сходится со скоростью геометрической прогрессии со знаменателем  $\frac{m-1}{m}$ , то есть линейно.

#### 4.1.5. Модификации метода Ньютона

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

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

Очевидный выбор —  $\mu \approx f'(x_0)$ .

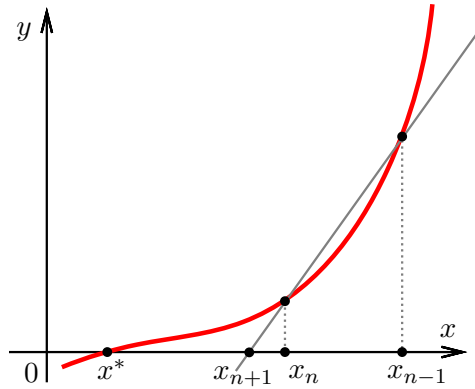


▷<sub>2</sub> Определите порядок сходимости метода.

**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  $y(x) = 0$ :

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

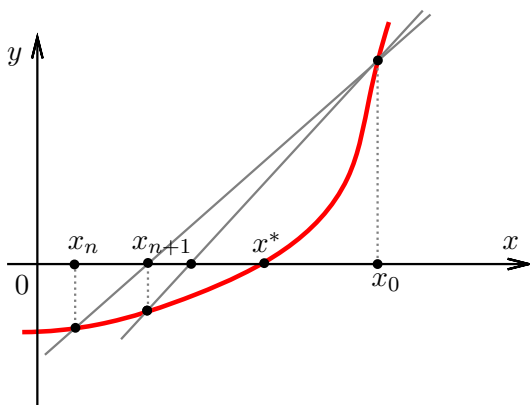
Это — метод секущих. Его, очевидно, нельзя представить в виде (4.2), поэтому порядок сходимости метода секущих вычисляется не тривиально. Он равен золотому сечению:

$$m = \frac{1 + \sqrt{5}}{2} \approx 1.618,$$

то есть 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.

▷<sub>3</sub> 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:

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

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

▷<sub>4</sub> Обоснуйте это правило.

▷<sub>5</sub> Определите порядок сходимости метода хорд.

## 4.2. Решение систем нелинейных уравнений

### 4.2.1. Постановка задачи

Систему нелинейных уравнений

$$\begin{cases} f_1(x_1, \dots, x_n) = 0, \\ \vdots \\ f_n(x_1, \dots, x_n) = 0 \end{cases} \quad (4.12)$$

запишем в векторном виде

$$f(x) = 0, \quad (4.13)$$

где  $x = [x_1, \dots, x_n]^T \in \mathbb{R}^n$ ,  $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ ,

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

Как и ранее, точное решение уравнения (4.13) обозначим  $x^*$ . В дальнейшем будем предполагать, что  $f$  достаточно гладкая, то есть 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):

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

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

$$f' = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}. \quad (4.16)$$

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.

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

В векторной форме эти равенства можно записать как

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

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

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

Отбрасывая остаточный член в формуле (4.17) получаем

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

Вычисляя

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

получаем формулу (4.15)

*Замечание 4.1.* Важно понимать, что формула (4.15) в явном виде практически никогда не используется для расчётов, 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  $\Delta x^k$  как решение СЛАУ

$$f'(x^k)\Delta x^k = -f(x^k), \quad (4.18a)$$

2) а затем вычисляют

$$x^{k+1} = x^k + \Delta x^k. \quad (4.18b)$$

▷<sub>1</sub> Оцените сложность реализации метода Ньютона по формулам (4.18).

При больших  $n$  метод Ньютона становится слишком дорогим удовольствием. 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 (4.18a), рассматривают метод Ньютона с «замороженной» матрицей Якоби:

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

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 (12.13) можно рассматривать с двух точек зрения.

- 1) С одной стороны, он может быть получен из метода Ньютона путём замены

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

то есть метод секущих основан на приближённом 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 предыдущей лекции). 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.

**1 способ: приближённое вычисление якобиана.** Рассмотрим набор приращений  $[h_1, \dots, h_n] \in \mathbb{R}^n$  и определим приближения для частных производных  $f$  по формуле

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



откуда имеем

$$f'(x) \approx \Psi(x).$$

Соответствующий метод имеет вид

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

▷<sub>2</sub> Перечислите достоинства и недостатки этого метода.

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

## 2 способ: метод Бroyдена!

### 4.2.5. Метод Бroyдена

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

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

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)  $b = f(x^k) - Ax^k$ , получаем до боли знакомую формулу

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

Таким образом получаем метод следующего вида.

**Общая схема итерации многомерного метода секущих**

1) По двум предыдущим приближениям  $x^k$  и  $x^{k-1}$  находим матрицу  $A = A_k$ , определяющую уравнение секущей.

2) Находим

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

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

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

$$A_k \Delta x = \Delta f. \quad (4.25)$$

Основная идея Бройдена заключается в том, чтобы представить  $A_k$  в виде

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

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

Подставляя (4.26) в (4.25) получаем

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

откуда

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

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

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

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.

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. Методы спуска

Следующее большое семейство методов основано на замене задачи решения системы нелинейных уравнений  $f(x) = 0$  задачей нахождения минимума некоторого функционала.

Для системы (4.13) рассмотрим функционал  $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}$ ,

$$\Phi(x) = \|f(x)\|_2^2 = \sum_i f_i(x)^2. \quad (4.29)$$

Очевидно, что  $\Phi(x) \geq 0$  и что глобальный минимум  $\Phi(x)$  достигается на точном решении системы:  $x = x^*$ . Заметим, что возможны и другие варианты выбора функционала.

Пусть  $x^k$  — текущее приближение,  $p$  — некоторый вектор, который будем называть *направлением спуска*. Следующее приближение  $x^{k+1}$  будем искать в виде

$$x^{k+1} = x^k + \alpha p,$$

где  $\alpha$  — вещественный параметр, который нужно как-то определить. Будем выбирать такое значение  $\alpha$ , которое сильнее всего уменьшит значение  $\Phi(x^{k+1})$ , то есть минимизирует величину

$$\psi(\alpha) = \Phi(x^k + \alpha p).$$

Таким образом, для определения  $\alpha$  нужно решить скалярное уравнение

$$\frac{d\psi}{d\alpha}(\alpha) = 0.$$

Понятно, что в общем случае для нахождения  $\alpha$  нужно использовать какой-то приближённый метод (Ньютона, секущих и т. п.).

Найдём явный вид функции  $\psi'(\alpha)$ :

$$\begin{aligned} \psi'(\alpha) &= \frac{d}{d\alpha} \Phi(x^k + \alpha p) = \frac{d}{d\alpha} \sum_{i=1}^n f_i(x_1^k + \alpha p_1, \dots, x_n^k + \alpha p_n)^2 = \\ &= 2 \sum_{i=1}^n f_i(x^k + \alpha p) \sum_{j=1}^n \frac{\partial f_i}{\partial x_j}(x^k + \alpha p) p_j = 2 \left( f(x^k + \alpha p), f'(x^k + \alpha p) p \right). \end{aligned} \quad (4.30)$$

### Метод градиентного спуска

Напомним, что *градиентом* функции  $n$  переменных  $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}$  называется вектор-функция

$$\text{grad } \Phi = \nabla \Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n,$$

определяемая формулой

$$\nabla \Phi = \left[ \frac{\partial \Phi}{\partial x_1}, \frac{\partial \Phi}{\partial x_2}, \dots, \frac{\partial \Phi}{\partial x_n} \right]^T.$$

Как известно, основное свойство вектора  $\nabla \Phi(x)$  состоит в том, что он показывает направление и величину наибольшей скорости возрастания  $\Phi$  в точке  $x$ . Соответственно, вектор  $-\nabla \Phi(x)$  указывает направление наискорейшего убывания.

Таким образом, если в качестве направления спуска  $p = p^k$  взять антиградиент  $\Phi$  в точке  $x^k$ , получится *метод градиентного (наискорейшего) спуска*:

$$p^k = \nabla \Phi(x^k), \quad p_i^k = \frac{\partial \Phi}{\partial x_i}(x^k) = 2 \sum_{j=1}^n f_j(x^k) \frac{\partial f_j}{\partial x_i}(x^k),$$

откуда

$$p^k = \nabla \Phi(x^k) = 2(f'(x^k))^T f(x^k). \quad (4.31)$$

Заметим, что на самом деле знак вектора, задающего направление спуска, не имеет значения, поэтому мы для простоты записи берем  $p^k = \nabla \Phi(x^k)$ , а не  $-\nabla \Phi(x^k)$ .

Итак, **итерация метода градиентного спуска** для решения системы (4.13) имеет следующий вид:

- 1) Вычисляем  $p^k = \nabla \Phi(x^k)$  по формуле (4.31).
- 2) Находим  $\alpha^*$  как решение скалярного уравнения  $\psi'(\alpha) = 0$ , где  $\psi'(\alpha)$  определяется формулой (4.30),  $p = p^k$ .
- 3) Вычисляем  $x^{k+1} = x^k + \alpha^* p^k$ .

▷3 Охарактеризуйте трудоёмкость метода градиентного спуска.

### Метод покоординатного спуска

Менее трудоёмкий вариант выбора направления  $p^k$  состоит в циклическом повторении  $p^k = e^i, i = \overline{1, n}$ , где  $e^i \in \mathbb{R}^n$  —  $i$ -й единичный орт,  $e_j^i = \delta_{ij}$ .

В этом случае одна итерация сводится к уточнению  $i$ -й компоненты вектора  $x^k$ . Прогоняя  $i$  от 1 до  $n$ , получаем, одну итерацию *метода покоординатного спуска*.

#### 4.2.7. Еще немного

Ну и напоследок — **нелинейный метод Гаусса–Зейделя**. По аналогии со случаем СЛАУ здесь мы в цикле по  $i$  от 1 до  $n$  по очереди вычисляем компоненты  $x_i^{k+1}$ , решая относительно  $x_i$  скалярное уравнение вида

$$f_i(x_1^{k+1}, \dots, x_{i-1}^{k+1}, x_i, x_{i+1}^k, \dots, x_n^k) = 0, \quad (4.32)$$

после чего полагаем  $x_i^{k+1} = x_i$ .

▷<sub>4</sub> 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  $x^{k+1} = \varphi(x^k)$ . Сейчас мы 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

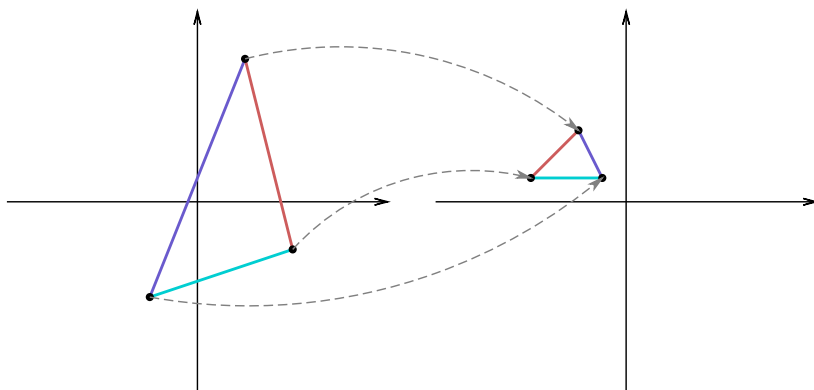
$$\rho(x, y) = \|x - y\|. \quad (4.33)$$

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

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

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

**Определение 4.5.** *Сжимающим отображением* называют функцию  $\varphi$ , 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*.



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

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

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

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

$$\varphi(x) = x \tag{4.34}$$

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

$$x^{k+1} = \varphi(x^k), \quad k = 0, 1, 2, \dots \tag{4.35}$$

сходится к  $x^*$ :

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

*Доказательство.* Рассмотрим последовательность

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

и докажем, что она является фундаментальной:

$$\rho(x^k, x^{k+1}) = \rho(\varphi(x^{k-1}), \varphi(x^k)) \leq L\rho(x^{k-1}, x^k) \leq \dots \leq L^k \rho(x^0, x^1).$$

Пусть  $m > k$ . Тогда по неравенству треугольника

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

откуда получаем

$$\rho(x^k, x^m) \leq \frac{L^k}{1-L} \rho(x^0, x^1), \tag{4.36}$$

то есть  $(x^k)$  фундаментальна ( $L < 1$  по условию), и, следовательно, сходится к какому-то вектору  $x^* \in \mathbb{R}^n$ . Переходя к пределу в (4.35) получаем, что  $x^*$  удовлетворяет уравнению (4.34).

Для доказательства единственности предположим, что  $\exists x^{**} \neq x^* : \varphi(x^{**}) = x^{**}$ . Это приводит к противоречию:  $\rho(x^*, x^{**}) = \rho(\varphi(x^*), \varphi(x^{**})) \leq L\rho(x^*, x^{**}) < \rho(x^*, x^{**})$ . □

**Следствие 4.1.** Устремляя  $t \rightarrow \infty$  в формуле (4.36), получаем априорную оценку погрешности:

$$\rho(x^k, x^*) \leq \frac{L^k}{1-L} \rho(x^0, x^1). \quad (4.37)$$

▷<sub>1</sub> Получите из формулы (4.37) оценку для количества итераций  $N$ , достаточного для получения решения с погрешностью, не превышающей  $\epsilon$ .

Кроме оценки (4.37) можно получить и *апостериорную* оценку погрешности, которая как правило более точна, но для получения которой необходимо проделать  $k$  итераций. Положив в (4.37)  $x^0 = x^{k-1}$  получаем

$$\rho(x^k, x^*) \leq \frac{L}{1-L} \rho(x^{k-1}, x^k). \quad (4.38)$$

**Замечание 4.2.** Функция  $\rho$  (метрика), которая измеряет «расстояние» между двумя точками, не обязательно должна иметь вид (4.33). Достаточно лишь, чтобы она удовлетворяла аксиомам метрики:

- 1)  $\rho(x, y) = 0 \Leftrightarrow x = y$ ;
- 2)  $\rho(x, y) = \rho(y, x)$ ;
- 3)  $\rho(x, y) \leq \rho(x, z) + \rho(z, y)$ .

**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 помощью него 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  $\varphi$  является лишь локально-сжимающим.

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

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



**Лемма 4.2.** Пусть отображение  $\varphi$  имеет неподвижную точку  $x^* = \varphi(x^*)$  и является локально-сжимающим в некоторой её окрестности  $B^* = B(x^*, r)$ . Тогда для всех  $x_0 \in B^*$  итерационный процесс (4.35) сходится к  $x^*$ .

▷<sub>2</sub> Докажите лемму.

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

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

$$\rho(x, \varphi(x)) \leq (1 - L)r, \quad (4.39)$$

то  $x^*$  — искомый корень уравнения  $\varphi(x) = x$  — лежит в  $B$  и итерационный процесс (4.35) 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

$$\begin{aligned} \rho(\varphi(y), x) &\leq \rho(\varphi(y), \varphi(x)) + \rho(x, \varphi(x)) \leq \\ &\leq L\rho(y, x) + \rho(x, \varphi(x)) \leq [(4.39)] \leq Lr + (1 - L)r = r. \end{aligned}$$

□

### 4.3.3. Применение принципа сжимающих отображений

Рассмотрим случай функции одной переменной (одного нелинейного уравнения).

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

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

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

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

поэтому если

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

то  $\varphi$  липшицева на  $[a, b]$  с константой  $L = M$ .

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

$$|\varphi'(x)| \leq M < 1 \quad \forall x \in \mathbb{R}, \quad (4.40)$$

то итерационный процесс (4.35) сходится.

Очевидно, что ограничение вида (4.40) слишком сурово. Как правило мы можем рассчитывать на выполнение условий такого рода лишь на некотором подмножестве  $\mathbb{R}$ . Лемма 4.2 дает **признак сходимости 1**: если

$$|\varphi'(x)| \leq M < 1 \quad \forall x \in B^* = [x^* - r, x^* + r], \quad (4.41)$$

то итерационный процесс (4.35) сходится к  $x^*$  при всех  $x_0 \in B^*$ .

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

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

и при этом

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

то итерационный процесс (4.35) сходится к  $x^* = \varphi(x^*)$  при всех  $x_0 \in B$ .

**Важно понимать** 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 (4.35), 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

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

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

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

такой, что

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

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

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

$$\Phi \alpha = y, \quad (5.2)$$

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

$$\Phi = \begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \cdots & \varphi_n(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \cdots & \varphi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_n) & \varphi_1(x_n) & \cdots & \varphi_n(x_n) \end{bmatrix}. \quad (5.3)$$

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

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

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:

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

В этом случае интерполирующую функцию  $\varphi$  называют *интерполяционным многочленом*.

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

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

Матрица такого вида называется *матрицей Вандермонда*.

**Лемма 5.1.**

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

*Доказательство.*

$$|V(x_0, \dots, x_n)| = \begin{vmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{vmatrix} = \left[ \begin{array}{l} \text{вычитаем из } j\text{-го столбца} \\ (j-1)\text{-й, умноженный на} \\ x_0, \text{ для } j \text{ от } n \text{ до } 2. \end{array} \right] =$$

$$= \begin{vmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & x_1 - x_0 & x_1^2 - x_1 x_0 & \cdots & x_1^n - x_1^{n-1} x_0 \\ 1 & x_2 - x_0 & x_2^2 - x_2 x_0 & \cdots & x_2^n - x_2^{n-1} x_0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n - x_0 & x_n^2 - x_n x_0 & \cdots & x_n^n - x_n^{n-1} x_0 \end{vmatrix} =$$

$$= (x_1 - x_0)(x_2 - x_0) \dots (x_n - x_0) \begin{vmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{vmatrix} =$$

$$= (x_1 - x_0)(x_2 - x_0) \dots (x_n - x_0) \cdot |V(x_1, \dots, x_n)| = [\dots] = \prod_{0 \leq j < i \leq n} (x_i - x_j).$$

□

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

▷<sub>1</sub> Докажите.

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

▷<sub>2</sub> Докажите.

### 5.1.3. Интерполяционный многочлен в форме Лагранжа

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

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

то есть

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

Неизвестную константу  $C_i$  находим 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 окончательно получаем

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

Таким образом получаем знаменитую формулу *интерполяционного многочлена в форме Лагранжа*:

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



Для многочленов  $\Lambda_i$  существует альтернативная форма записи. Рассмотрим многочлен

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

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

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

#### 5.1.4. Итого

- 1) Интерполяция — наиболее простой и популярный способ приближения функций.
- 2) Понимание понятия фундаментального базиса очень пригодится в дальнейшем.
- 3) В частности потому, что если известен фундаментальный базис, то задача интерполяции решается легко и непринуждённо.
- 4) Кто не будет уметь строить интерполяционный многочлен Лагранжа — останется на второй год!

## 5.2. Интерполяционный многочлен в форме Ньютона

### 5.2.1. Построение

Интерполяционная формула Лагранжа (5.8) становится громоздкой и неудобной для вычислений при больших  $n$ . Более удобным на практике оказывается представление интерполяционного многочлена в так называемой форме Ньютона.

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

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

При построении формулы Лагранжа мы выбирали базис  $\{\varphi_i\}$  так, чтобы матрица  $\Phi$  была единичной. Теперь же построим базис, в котором эта матрица будет *нижнетреугольной*:

$$\Phi \alpha = \begin{bmatrix} \varphi_0(x_0) & 0 & 0 & \cdots & 0 \\ \varphi_0(x_1) & \varphi_1(x_1) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_n) & \varphi_1(x_n) & \varphi_2(x_n) & \cdots & \varphi_n(x_n) \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}. \quad (5.11)$$

Такой базис должен, очевидно, удовлетворять условиям

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

откуда с точностью до постоянного множителя получаем

$$\varphi_i(x) = (x - x_0)(x - x_1) \cdots (x - x_{i-1}) = \prod_{j=0}^{i-1} (x - x_j) = \omega_i(x) \quad (5.12)$$

(сравните с формулой (5.9)). Здесь при  $i = 0$  как обычно полагаем  $\omega_0(x) = 1$ . Следовательно, матрица СЛАУ из (5.11) будет иметь вид

$$\Phi = \Omega = \left( \omega_j(x_i) \right)_{i,j=0}^n =$$

$$= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & x_1 - x_0 & 0 & \cdots & 0 \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n - x_0 & (x_n - x_0)(x_n - x_1) & \cdots & (x_n - x_0) \cdots (x_n - x_{n-1}) \end{bmatrix}.$$

▷<sub>1</sub> При каких условиях СЛАУ с такой матрицей имеет единственное решение?  
Таким образом получаем интерполяционный многочлен в виде

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

$$+ \alpha_n(x - x_0)(x - x_1) \dots (x - x_{n-1}), \quad (5.13)$$

где коэффициенты  $\alpha_i$  можно найти из (5.11) по рекуррентным соотношениям:

$$\alpha_i = \left( y_i - \sum_{j=0}^{i-1} \alpha_j \omega_j(x_i) \right) / \omega_i(x_i), \quad i = \overline{0, n}. \quad (5.14)$$

*Внимание! Использование формул (5.14) не является общепринятым способом вычисления  $\alpha_i$  (см. далее).*

**Определение 5.3.** Многочлен  $P_n$ , определяемый формулами (5.13), (5.14) называется *интерполяционным многочленом в форме Ньютона*.

*Следует понимать также, что формулы (5.13) и (5.8) — суть разные формы записи одного и того же многочлена!*

По построению интерполяционный многочлен в форме Ньютона обладает рядом полезных свойств. Во-первых, базисные функции

$\omega_i$  зависят только от узлов  $x_0, \dots, x_{i-1}$ , а коэффициенты  $\alpha_i$  — только от  $x_0, \dots, x_i$ . Это позволяет легко «обновлять» формулу многочлена при добавлении дополнительных узлов интерполяции. Другими словами, если известен многочлен  $P_n$  (5.13), интерполирующий  $f$  по узлам  $x_0, \dots, x_n$ , то интерполяционный многочлен по узлам  $x_0, \dots, x_{n+1}$  может быть найден по формуле

$$P_{n+1}(x) = P_n(x) + \alpha_{n+1}\omega_{n+1}(x). \quad (5.15)$$

Во-вторых, форма (5.13) более удобна для вычислений, чем форма Лагранжа: последовательно вынося за скобки общие множители  $(x - x_i)$ , можно вычислить  $P_n(x)$  по т. н. схеме Горнера

$$P_n(x) = \alpha_0 + (x - x_0)(\alpha_1 + (x - x_1)(\alpha_2 + \dots)). \quad (5.16)$$

▷<sub>2</sub> Подсчитайте количество арифметических операций, нужных для вычисления  $P_n(x)$  по формуле (5.16).

### 5.2.2. Разделённые разности

Пусть  $\{y_i\}_{i=0}^n$  являются значениями некоторой функции  $f$  в точках  $\{x_i\}_{i=0}^n$  соответственно. Тогда каждый из коэффициентов  $\alpha_i$ , определяемых по формулам (5.14) можно рассматривать как выражение, зависящее от функции  $f$  и узлов  $x_0, \dots, x_i$ . Обозначим (пока что чисто формально) это выражение

$$\alpha_i = f[x_0, \dots, x_i]. \quad (5.17)$$

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

$$P_{k+1}(x) = \frac{(x - x_0)P_k^+(x) - (x - x_{k+1})P_k(x)}{x_{k+1} - x_0}, \quad \forall k \geq 0. \quad (5.18)$$

▷<sub>3</sub> 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 (5.17):

$$P_{i-1}(x) = \sum_{j=0}^{i-1} \alpha_j \omega_j(x) = \sum_{j=0}^{i-1} f[x_0, \dots, x_j] \omega_j(x),$$

$$P_{i-1}^+(x) = \sum_{j=0}^{i-1} \alpha_j^+ \omega_j^+(x) = \sum_{j=0}^{i-1} f[x_1, \dots, x_{j+1}] \omega_j^+(x).$$

Здесь  $\omega_j^+(x) = (x - x_1)(x - x_2) \dots (x - x_j)$ . Тогда (5.18) примет вид

$$P_i(x) = \sum_{j=0}^i f[x_0, \dots, x_j] \omega_j(x) =$$

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

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

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

Отсюда получаем следующее определение.

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

$$f[x_0, \dots, x_i] = \frac{f[x_1, \dots, x_i] - f[x_0, \dots, x_{i-1}]}{x_i - x_0}, \quad \forall i \geq 1; \quad (5.19a)$$

$$f[x_j] = f(x_j) \quad \forall j. \quad (5.19b)$$

Таким образом, коэффициенты (5.17) интерполяционного многочлена в форме Ньютона являются разделёнными разностями и традиционно вычисляются по определению (5.19).

▷<sub>4</sub> Вычислите общий вид коэффициентов  $\alpha_0$ ,  $\alpha_1$  и  $\alpha_2$  сначала по формуле (5.14), затем по формуле (5.19).

▷<sub>5</sub> Сравните вычислительную сложность вычисления  $\alpha_i$  по формулам (5.14) и (5.19).

▷<sub>6</sub> Докажите, что значение разделённой разности не зависит от порядка расположения её аргументов.

### 5.2.3. Алгоритм вычисления разделённых разностей

Коэффициенты интерполяционного многочлена (5.13) удобно вычислять по определению разделённых разностей (5.19) путём построения треугольной таблицы следующего вида.

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

▷<sub>7</sub> Каким минимальным количеством ячеек памяти можно обойтись при вычислении разделённых разностей, необходимых для построения интерполяционного многочлена в форме Ньютона? Постройте соответствующий алгоритм.

### 5.2.4. Остаток интерполирования

**Определение 5.5.**  $C$ -нормой для непрерывной на отрезке  $[a, b]$  функции  $f$  называется величина  $\|f\|_C$ , вычисляемая по формуле

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

**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

$$r_n = f - P_n. \quad (5.20)$$

Погрешностью интерполирования назовём  $C$ -норму остатка:

$$\varepsilon_n = \|r_n\|_C = \max_{x \in [a, b]} |r_n(x)|. \quad (5.21)$$

**Теорема 5.1.** *Остаток интерполирования имеет вид*

$$r_n(x) = f[x_0, \dots, x_n, x] \omega_{n+1}(x) \quad \forall x \in \mathbb{R}. \quad (5.22)$$

*Доказательство.* Пусть  $P_n$  — интерполяционный многочлен по узлам  $x_0, \dots, x_n$  для  $f$ . Если  $x \in \{x_i\}_{i=0}^n$ , то по определению имеем  $r_n(x) = 0$ , что почти<sup>1</sup> соответствует (5.22).

Пусть теперь  $x \notin \{x_i\}_{i=0}^n$ . Рассмотрим для  $f$  интерполяционный многочлен  $P_{n+1}$  по узлам  $x_0, \dots, x_n, x$ . По формуле (5.15) имеем

$$P_{n+1}(x) = f(x) = P_n(x) + f[x_0, \dots, x_n, x] \omega_{n+1}(x). \quad \square$$

Для того, чтобы вывести еще одно полезное выражение для остатка, необходимо вспомнить следующую теорему из курса анализа.

**Теорема 5.2** (Теорема 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  $\underline{x} = \min\{x_i\}_{i=0}^n$ ,  $\bar{x} = \max\{x_i\}_{i=0}^n$  и  $f \in C^n[\underline{x}, \bar{x}]$ . Тогда  $\exists \xi \in (\underline{x}, \bar{x})$  такое, что*

$$f[x_0, \dots, x_n] = \frac{f^{(n)}(\xi)}{n!}. \quad (5.23)$$

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

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

---

<sup>1</sup>Почти — потому что в этом случае пока что не определено значение разделенной разности  $f[x_0, \dots, x_n, x_i]$ . Это будет сделано в следующей лекции.

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

Продолжая аналогичные рассуждения для  $r''_n$  и так далее, получаем, что  $\exists \xi \in (\underline{x}, \bar{x})$  такое, что

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

**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  $\forall x \in [a, b] \exists \xi \in (\underline{x}, \bar{x})$ ,  $(\underline{x} = \min\{x_0, \dots, x_n, x\}, \bar{x} = \max\{x_0, \dots, x_n, x\})$ , такое, что

$$r_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x). \quad (5.24)$$

*Доказательство.* Данная теорема является тривиальным следствием теоремы 5.1 и леммы 5.3.  $\square$

Формула (5.24) является основным инструментом при оценке погрешности интерполяции. Из неё, в частности, сразу получаем

$$\varepsilon_n = \|r_n\| \leqslant \frac{\|f^{(n+1)}\|}{(n+1)!} \|\omega_{n+1}\|, \quad (5.25)$$

где  $\|\cdot\| = \|\cdot\|_{C[a,b]}$  (см. (5.21)).



## 5.3. Минимизация остатка интерполирования. Многочлены Чебышева

### 5.3.1. 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

$$\|f - P_n\| \leq \frac{\|f^{(n+1)}\|}{(n+1)!} \|\omega_{n+1}\|, \quad (5.26)$$

где

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

Здесь и далее  $\|\cdot\| = \|\cdot\|_C$ .

Наша задача — сделать правую часть (5.26) как можно меньше. И если в общем случае о величине  $\|f^{(n+1)}\|$  ничего не известно, то  $\|\omega_{n+1}\|$  можно минимизировать за счёт выбора узлов  $\{x_i\}$ . Задачу выбора оптимальных узлов интерполяции можно поставить так: найти многочлен  $\omega_{n+1}$  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  $[a, b]$ . Понятно, что корни такого многочлена и будут оптимальными<sup>2</sup> узлами интерполяции.

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

### 5.3.2. Многочлены Чебышева

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

$$T_n(x) = \cos(n \arccos x), \quad n = 0, 1, 2, \dots \quad (5.27)$$

<sup>2</sup>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, не совпадают с чебышевскими.

Как это ни кажется странным на первый взгляд,  $T_n$  является алгебраическим многочленом степени  $n$ :

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

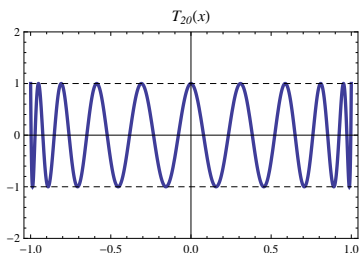
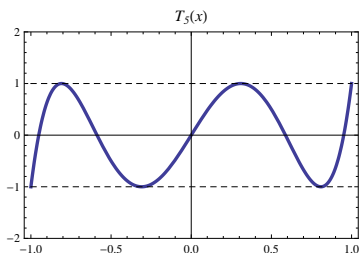
а для произвольного  $n$  справедливо рекуррентное соотношение

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x). \quad (5.28)$$

▷<sub>1</sub> Вычислите  $T_i(x)$  для  $i = 2, 3, 4, 5$ .

▷<sub>2</sub> Докажите рекуррентное соотношение 5.28, используя тригонометрическую формулу  $\cos \alpha \cos \beta = \frac{1}{2}(\cos(\alpha + \beta) + \cos(\alpha - \beta))$ .

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



Следствием этого свойства является то, что *если многочлен  $T_n$  разделить на его старший коэффициент, то получится искомым нами многочлен, наименее отклоняющийся от нуля на  $[-1, 1]$ .*

### 5.3.3. Оптимальные узлы интерполирования

Таким образом, нули многочлена Чебышева (5.27) являются оптимальными узлами интерполирования на отрезке  $[-1, 1]$ , то есть оптимальный многочлен  $\omega_{n+1}$  определяется формулой

$$\omega_{n+1}(x) = 2^{-n}T_{n+1}(x). \quad (5.29)$$

Множитель  $2^{-n}$  — это обратное значение старшего коэффициента многочлена  $T_{n+1}$ , в чем нетрудно убедиться из рекуррентного соотношения (5.28). Нули этого многочлена легко находятся из уравнения

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

и равны

$$x_i = \cos \frac{\pi(2i+1)}{2n+2}, \quad i = \overline{0, n}. \quad (5.30)$$

▷<sub>3</sub> Как надо изменить формулу (5.30), чтобы узлы интерполяции расположились в порядке возрастания?

Это и есть оптимальные узлы интерполирования, называемые также *чебышевскими*. При таком выборе узлов из (5.29) получаем

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

то есть оценка погрешности (5.31) для интерполяционного многочлена по чебышевским узлам на отрезке  $[-1, 1]$  будет иметь вид

$$\|r_n\| \leq \frac{\|f^{(n+1)}\|}{2^n(n+1)!}. \quad (5.31)$$

### Чебышевские узлы на отрезке $[a, b]$

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

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

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

$$\begin{aligned} \hat{T}_{n+1}(x) &= T_{n+1}(t) = 2tT_n(t) - T_{n-1}(t) = 2t\hat{T}_n(x) - \hat{T}_{n-1}(x) = \\ &= 2\frac{2x-a-b}{b-a}\hat{T}_n(x) - \hat{T}_{n-1}(x). \end{aligned} \quad (5.32)$$

Кроме этого имеют место очевидные соотношения

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

Следовательно, согласно (5.32) старший коэффициент многочлена  $\hat{T}_n(x)$  при всех  $n \geq 1$  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 получаем

$$\omega_{n+1}(x) = 2 \left( \frac{b-a}{4} \right)^{n+1} T_{n+1} \left( \frac{2x-b-a}{b-a} \right). \quad (5.33)$$

Корни этого многочлена (*оптимальные узлы интерполяции на отрезке*  $[a, b]$ ), очевидно, получаются масштабированием узлов (5.30):

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

Кроме этого, из (5.33) следует, что при выборе чебышевских узлов интерполирования (5.34) имеем

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

Это равенство используется при оценке погрешности интерполирования по формуле (5.26):

$$\|r_n\| \leq 2 \left( \frac{b-a}{4} \right)^{n+1} \frac{\|f^{(n+1)}\|}{(n+1)!}. \quad (5.35)$$

### 5.3.4. 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

$$\|f - P_n\|_C \xrightarrow{n \rightarrow \infty} 0.$$

Здесь  $P_n$  — интерполяционный многочлен, построенный по набору узлов

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

Таким образом, последовательность  $\{P_n\}$ , то есть сам интерполяционный процесс, определяется таблицей узлов интерполяции

$$\begin{aligned} X_0 &= \{x_0^{(0)}\}, \\ X_1 &= \{x_0^{(1)}, x_1^{(1)}\}, \\ &\dots \\ X_n &= \{x_0^{(n)}, x_1^{(n)}, \dots, x_n^{(n)}\}, \\ &\dots \end{aligned} \tag{5.36}$$

### 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 серьёзную 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 (5.36) 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 её 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 (5.36) 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  $f(x) = \frac{1}{1+25x^2}$  и функции  $g(x) = |x|$ , обе на отрезке  $[-1, 1]$ . Для последней интерполяционный многочлен по равноотстоящим узлам степени  $2n$  неограниченно растёт в любой части отрезка  $[-1, 1]$ .

Кроме этого интерполяция по равноотстоящим узлам обладает еще одним неприятным свойством: она плохо обусловлена, то есть при достаточно большом количестве точек небольшие погрешности округления будут приводить к большим погрешностям. То есть, 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.

Гораздо более выгодно выглядит 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 является равномерно непрерывной на нем. Помимо этого, интерполяция по чебышевским узлам хорошо обусловлена.

### 5.3.5. Барицентрическая форма записи многочлена Лагранжа

Оказывается, что при правильной форме записи интерполяционный многочлен в форме Лагранжа может конкурировать, а иногда и быть более эффективен, чем представление в форме Ньютона.

Запишем многочлен Лагранжа с использованием формулы (5.10):

$$P_n(x) = \sum_{i=0}^n y_i \frac{\omega_{n+1}(x)}{(x - x_i)\omega'_{n+1}(x_i)}, \quad (5.37)$$

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

Чтобы избежать громоздких умножений в каждом слагаемом формулы (5.37), 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

$$P_n(x) = \omega_{n+1}(x) \sum_{i=0}^n y_i \frac{v_i}{x - x_i}, \quad (5.38)$$

где

$$v_i = \frac{1}{\omega'_{n+1}(x_i)} = \frac{1}{\prod_{j \neq i} (x_i - x_j)}, \quad i = \overline{0, n}. \quad (5.39)$$

Коэффициенты  $v_i$  называются весовыми. Формула (5.38) называется *первой формой барицентрической интерполяционной формулы*. Нетрудно убедиться, что эта формула обладает следующими достоинствами по сравнению с (5.37):

- Для вычисления  $P_n(x)$  по этой формуле требуется  $O(n)$  операций.
- Для её построения, то есть для вычисления коэффициентов  $\{v_i\}_{i=0}^n$ , требуется  $O(n^2)$  операций, как и для многочлена в форме Ньютона.
- Формула может быть легко обновлена при добавлении дополнительного узла интерполяции  $x_{n+1}$ : каждое  $v_i$  необходимо домножить на  $\frac{1}{x_i - x_{n+1}}$ , после чего по определению вычислить  $v_{n+1}$ .
- При этом остается достоинство, которого нет у многочлена Ньютона: весовые коэффициенты, в отличие от разделенных разностей, не зависят от  $\{y_i\}$ , то есть однажды построенная формула может легко применяться для интерполяции различных функций  $f$ .

Но и это еще не все. Существует вторая, еще более эффективная форма записи барицентрической формулы. Строится она так: строим по формуле (5.38) интерполяционный многочлен для  $f(x) = 1$ , который, очевидно, тождественно равен 1. То есть,

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

Выражая отсюда  $\omega_{n+1}(x)$  и подставляя в (5.38), получаем чудесную формулу

$$P_n(x) = \frac{\sum_{i=0}^n y_i \frac{v_i}{x - x_i}}{\sum_{i=0}^n \frac{v_i}{x - x_i}} \quad (5.40)$$

Эту формулу называют просто *барицентрической интерполяционной формулой*.

Помимо того, что данная формула требует меньше арифметических операций, большой плюс состоит в том, что весовые коэффициенты  $v_i$  можно спокойно домножать на любую константу. Это полезно с точки зрения вычислительной устойчивости.

Кроме этого, весовые коэффициенты для наиболее часто используемых узлов могут быть вычислены очень быстро, за  $O(n)$  операций. Например, для чебышевских узлов на  $[-1, 1]$  (5.30), с учетом (5.29), после сокращения множителей, не зависящих от  $i$ , получается формула

$$v_i = (-1)^i \sin \frac{2i+1}{2n+2} \pi. \quad (5.41)$$

▷<sub>4</sub> Выведите эту формулу

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

$$x_i = \cos \frac{i\pi}{n}, \quad i = \overline{0, n}. \quad (5.42)$$

Соответствующие весовые коэффициенты для барицентрической формулы определяются формулой

$$v_i = (-1)^i \delta_i, \quad \delta_i = \begin{cases} 1/2, & \text{если } i = 0 \text{ или } i = n, \\ 1, & \text{иначе.} \end{cases} \quad (5.43)$$

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

Помимо всех указанных замечательных свойств, барицентрические формулы обладают выдающейся вычислительной устойчивостью (при условии отдельной обработки случая, когда знаменатель в формуле (5.40) обращается в нуль.)



## 5.4. Сплаины

### 5.4.1. Определение

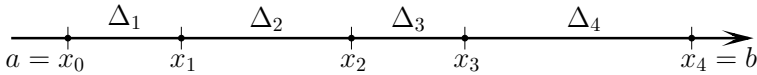
Рассмотрим набор узлов  $\{x_i\}_{i=0}^n$ ,

$$a = x_0 < x_1 < \dots < x_{n-1} < x_n = b$$

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

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

Пример такого разбиения для случая  $n = 4$  выглядит следующим образом.



На каждом отрезке  $\Delta_i$  зададим многочлен  $s_i$  фиксированной степени  $m \geq 1$  и рассмотрим полученный набор многочленов как единую кусочно-непрерывную функцию  $s$ , определенную на всем отрезке  $[a, b]$ :

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

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

**Определение 5.8.** Пусть  $\Delta$  — некоторое разбиение отрезка  $[a, b]$ , определяемое формулой (5.67),  $\mathbb{P}_m$  — множество всех многочленов степени не выше  $m$ . Для некоторого  $m \geq 1$  рассмотрим функцию  $s$ , определённую 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

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

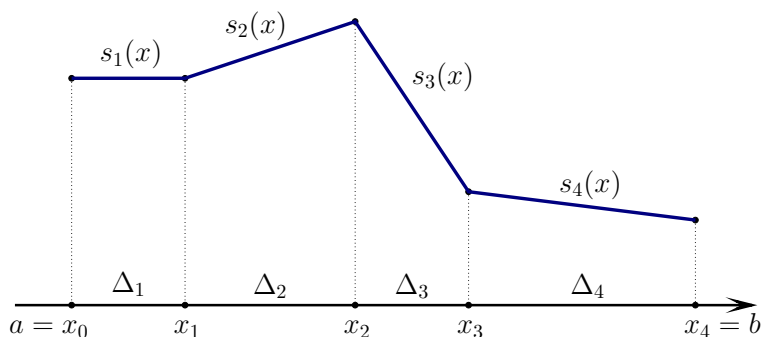
Такая функция  $s$  называется *полиномиальным сплайном порядка (степени)  $m$* . Множество всех сплайнов степени  $m$  на разбиении  $\Delta$  будем обозначать  $S_\Delta^m$ .

### 5.4.2. Интерполяционные сплайны

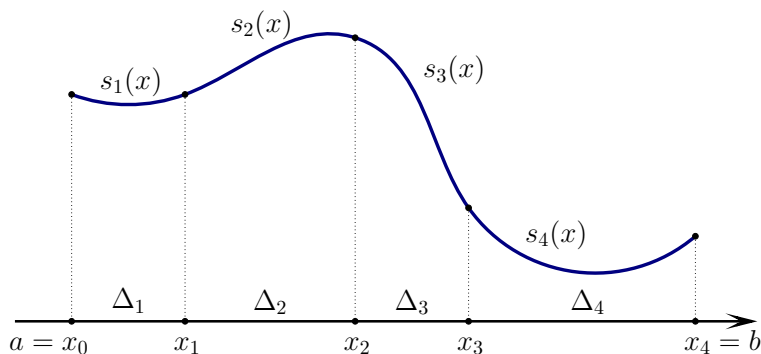
**Определение 5.9.** Сплайн  $s \in S_{\Delta}^m$  называется интерполяционным для функции  $f$ , если

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

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



Большой интерес, конечно, представляют сплайны высших степеней. Наиболее распространёнными являются кубические интерполяционные сплайны ( $m = 3$ ):



### Построение кубического интерполяционного сплайна

Рассмотрим задачу нахождения  $s \in S_{\Delta}^3$ , 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: в этом случае мы имеем

$$s'_i(x) = \beta_i + \gamma_i(x - x_i) + \frac{\delta_i}{2}(x - x_i)^2,$$

$$s''_i(x) = \gamma_i + \delta_i(x - x_i),$$

$$s'''_i(x) = \delta_i,$$

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

$$\alpha_0 = s_1(x_0), \quad \beta_0 = s'_1(x_0), \quad \gamma_0 = s''_1(x_0), \quad \delta_0 = s'''_1(x_0), \quad (5.47)$$

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

$$\alpha_i = y_i, \quad i = \overline{1, n}, \quad (5.48a)$$

и  $s_i(x_{i-1}) = y_{i-1}$ , или

$$\beta_i = \frac{y_i - y_{i-1}}{h_i} + \frac{\gamma_i}{2}h_i - \frac{\delta_i}{6}h_i^2, \quad i = \overline{1, n}. \quad (5.48b)$$

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

$$\beta_i - \gamma_i h_i + \frac{\delta_i}{2} h_i^2 = \beta_{i-1}, \quad i = \overline{2, n}. \quad (5.48c)$$

Ну и, наконец, требование непрерывности второй производной

$$s''_i(x_{i-1}) = s''_{i-1}(x_{i-1})$$

по аналогии даёт

$$\delta_i = \frac{\gamma_i - \gamma_{i-1}}{h_i}, \quad i = \overline{2, n}. \quad (5.48d)$$

Важно: если в последних двух формулах положить  $i = 1$ , то мы получим просто определение величин  $\beta_0$  и  $\gamma_0$  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  $\gamma_i$  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 (5.48a), (5.48b) и (5.48d) (в последней нужно взять ещё  $i = 1$ ). В частности, (5.48b) даёт

$$\beta_i = \frac{y_i - y_{i-1}}{h_i} + \frac{2\gamma_i + \gamma_{i-1}}{6} h_i. \quad (5.48b')$$

Для нахождения  $\{\gamma_i\}$  подставим (5.48b') и (5.48d) в (5.48c). После сдвига в нумерации получаем

$$h_i \gamma_{i-1} + 2(h_i + h_{i+1})\gamma_i + h_{i+1}\gamma_{i+1} = 6 \left( \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right),$$

что можно записать как

$$c_i \gamma_{i-1} + 2\gamma_i + e_i \gamma_{i+1} = b_i, \quad i = \overline{1, n-1}, \quad (5.49)$$

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 (5.48a), (5.48b') и (5.48d).

### Виды граничных условий

**Естественные граничные условия.** Самые 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 СЛАУ решаем, естественно, методом прогонки.

▷<sub>1</sub> Докажите, что метод прогонки в данном случае применим.

▷<sub>2</sub> 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 (5.48с) при  $i = 1$  с учётом (5.48b’):

$$\gamma_1 h_1 - \frac{\gamma_1 - \gamma_0}{2} h_1 = \frac{y_1 - y_0}{h_1} + \frac{2\gamma_1 + \gamma_0}{6} h_1 - f'(a),$$

откуда получаем

$$2\gamma_0 + \gamma_1 = \frac{6}{h_1} \left( \frac{y_1 - y_0}{h_1} - f'(a) \right).$$

Аналогично для  $i = n - 1$  имеем

$$\gamma_{n-1} + 2\gamma_n = \frac{6}{h_n} \left( f'(b) - \frac{y_n - y_{n-1}}{h_n} \right).$$

▷<sub>3</sub> Запишите соответствующую СЛАУ для нахождения  $\{\gamma_i\}_{i=0}^n$  и докажите, что к ней применим метод прогонки.

**Другие граничные условия.** На практике используется ещё 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  $x_{n-1}$ .

▷<sub>4</sub> 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  $\Phi : C^2[a, b] \rightarrow \mathbb{R}$ ,

$$\Phi(f) = \int_a^b (f''(x))^2 dx.$$

Экстремальное свойство интерполяционного сплайна  $s \in S_\Delta^3$  состоит в том, что он доставляет минимум  $\Phi$  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

$$f(x_i) = y_i, \quad i = \overline{0, n}, \quad \text{и} \quad f''(a) = f''(b) = 0.$$

Если кубический сплайн  $s \in S_\Delta^3$  принадлежит  $F_0$ , то

$$\Phi(s) \leq \Phi(f) \quad \forall f \in F_0.$$

*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  $\Phi(f) = \Phi(s + g)$ , получаем

$$\Phi(f) = \Phi(s) + \Phi(g) + 2 \int_a^b s'' g'' dx.$$

Для доказательства теоремы теперь достаточно показать, что

$$\int_a^b s'' g'' dx \geq 0.$$

Для этого воспользуемся интегрированием по частям, а также тем фак-

том, что  $s'''$  на интервале  $(x_{i-1}, x_i)$  равна константе, которую обозначим  $\delta_i$ . Вычисляем:

$$\begin{aligned} \int_a^b s'' g'' dx &= \sum_{i=1}^n \int_{x_{i-1}}^{x_i} s'' g'' dx = \sum_{i=1}^n \left( s'' g' \Big|_{x_{i-1}}^{x_i} - \int_{x_{i-1}}^{x_i} s''' g' dx \right) = \\ &= \underbrace{s''(b)g'(b)}_0 - \underbrace{s''(a)g'(a)}_0 - \sum_{i=1}^n \int_{x_{i-1}}^{x_i} s''' g' dx = - \sum_{i=1}^n \int_{x_{i-1}}^{x_i} s''' g' dx = \\ &= - \sum_{i=1}^n \delta_i \int_{x_{i-1}}^{x_i} g' dx = - \sum_{i=1}^n \delta_i (g(x_i) - g(x_{i-1})) dx = 0. \quad \square \end{aligned}$$

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 приближённое значение для усреднённой кривизны графика. Поэтому часто говорят (пусть и не совсем корректно), что интерполяционный сплайн обладает минимальной кривизной среди всех интерполирующих функций с указанными граничными условиями.

#### 5.4.4. Сходимость интерполяции кубическим сплайном

Приведём без доказательства следующий результат.

**Теорема 5.9.** *Рассмотрим  $f \in C^4[a, b]$  и разбиение  $\Delta$  отрезка  $[a, b]$  такое, что  $\max_i h_i = h$ . Пусть  $s \in S_{\Delta}^3$  — естественный интерполяционный сплайн для  $f$ . Тогда*

$$\|f - s\| \leq \frac{5}{384} h^4 \|f^{(4)}\|.$$

Здесь  $\|\cdot\| = \|\cdot\|_{C[a, b]}$ .

Таким образом, для достаточно гладких  $f$  погрешность интерполирования кубическим сплайном  $\varepsilon(h)$  равна  $O(h^4)$ .

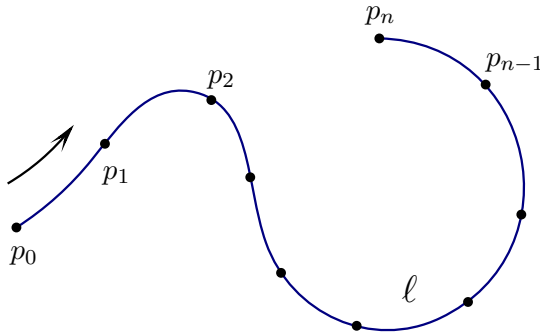


## 5.5. Приближение кривых

### 5.5.1. Интерполяция кривых на плоскости

#### Постановка задачи

Рассмотрим на декартовой плоскости кривую  $\ell$ . 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.



**Определение 5.10.** Задача интерполяции кривой  $\ell$  по точкам  $\{p_i\}$  заключается в построении такой параметрической функции  $g: \mathbb{R} \rightarrow \mathbb{R}^2$ ,

$$g(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}, \quad (5.50)$$

что

$$g(t_i) = p_i, \quad \forall i = \overline{0, n},$$

где  $\{t_i\}$  — некоторая возрастающая последовательность значений параметра:

$$t_0 < t_1 < \dots < t_{n-1} < t_n,$$

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 следующие ниже методы приближения плоских кривых, 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  $x : [t_0, t_n] \rightarrow \mathbb{R}$  и  $y : [t_0, t_n] \rightarrow \mathbb{R}$ , удовлетворяющие условиям

$$\begin{cases} x(t_i) = x_i, \\ y(t_i) = y_i, \end{cases} \quad i = \overline{0, n}. \quad (5.51)$$

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  $\Lambda_i$  определяется как обычно:

$$\Lambda_i(t) = \prod_{i \neq j} \frac{t - t_j}{t_i - t_j}.$$

▷<sub>1</sub> 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 разделе 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 требуемой *формой*. Эта задача существенно отличается от рассматриваемой ранее задачи интерполяции кривой тем, что здесь нету строго определённого объекта для приближения. Нужно путём визуального подбора построить кривую, обладающую определёнными свойствами (например, имеющую форму кузова автомобиля).

Интерполяция многочленами плохо подходит для решения такой задачи. Это связано с тем, что базисные функции Лагранжа  $\Lambda_i$  хоть и обладают свойством  $\Lambda_i(t_j) = \delta_{ij}$ , но на всём остальном отрезке интерполяции  $[a, b]$  могут принимать значения, намного бóльшие 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:  $i$ -ая базисная функция Лагранжа  $\Lambda_i$  *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

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad (5.52)$$

называют  $i$ -м (базисным) *многочленом Бернштейна* степени  $n$ ,  $0 \leq i \leq n$ . Здесь  $\binom{n}{i} = \frac{n!}{i!(n-i)!}$  — 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  $n$ . На рисунке 5.1 приведены графики всех многочленов Бернштейна для  $n = 4$ . Многочлены Бернштейна обладают множе-

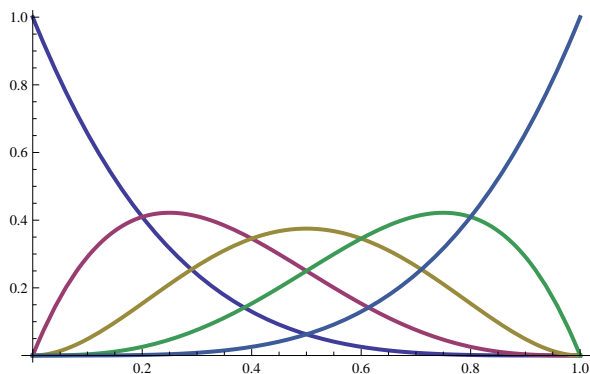


Рисунок 5.1

ством полезных свойств и достаточно широко используются в численном анализе. Наиболее интересны для нас следующие свойства этих многочленов.

- 1)  $B_i^n(t) \geq 0 \quad \forall t \in [0, 1], \quad \forall n, i;$
- 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)  $\frac{d}{dt} B_i^n(t) = n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t))$ ;
- 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)  $\sum_{i=0}^n B_i^n(t) = 1 \quad \forall t \in \mathbb{R}, \quad n \geq 0$ ;
- 7)  $B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$ .

▷<sub>2</sub> Докажите свойства 4-7.

### Кривые Безье

В двух словах, 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 много чего ещё.

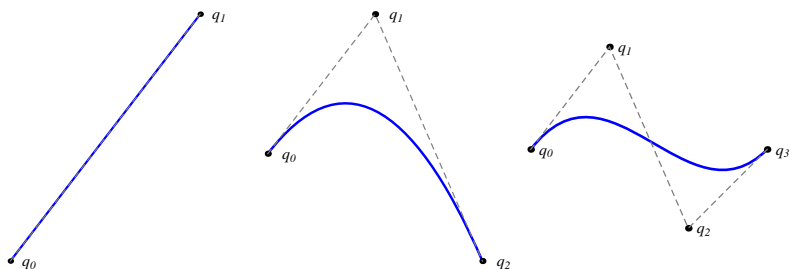
**Определение 5.12.** Рассмотрим набор точек плоскости  $Q = \{q_i \in \mathbb{R}^2\}_{i=0}^n$ , которые в дальнейшем будем называть *контрольными точками*. Кривой Безье называется множество точек

$$\{B(t) \mid t \in [0, 1]\},$$

где

$$B(t) = \sum_{i=0}^n B_i^n(t) q_i. \quad (5.53)$$

Перед тем, как обсудить свойства кривой Безье, рассмотрим вид кривых для  $n = 1, 2, 3$ :



Рисунки наглядно иллюстрируют 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

$$\begin{aligned} \frac{d}{dt}B(t) &= \sum_{i=0}^n \frac{d}{dt}B_i^n(t)q_i = n \sum_{i=0}^n (B_{i-1}^{n-1}(t) - B_i^{n-1}(t))q_i = \\ &= n \sum_{i=0}^{n-1} B_i^{n-1}(t)(q_{i+1} - q_i). \end{aligned}$$

Отсюда с учётом свойства 2 многочленов Бернштейна получаем

$$B'(0) = n(q_1 - q_0), \quad B'(1) = n(q_n - q_{n-1}).$$

#### 5.5.4. Алгоритмы построения кривых Безье

Рассмотрим задачу построения кривой Безье, задаваемой уравнением (5.76), которая сводится к вычислению значений многочлена  $B$  в точках  $t \in [0, 1]$ . Оба способа, которые мы рассмотрим, основываются на рекуррентных соотношениях для многочленов Бернштейна (свойство 7).

##### Прямой алгоритм

Вычисление  $B(t)$  для данного  $t$ , очевидно, 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. Понятно, что грубое вычисление  $b_i$  по определению (5.52) — не наш метод. Согласно свойству 7 имеем

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t). \quad (5.54)$$

Для того, чтобы вычислить все  $b_i$  по такой схеме, необходимо вычислить все  $B_j^k(t)$  для  $k = \overline{0, n}$ ,  $j = \overline{0, k}$ . Делается это последовательным (слева направо) заполнением следующей треугольной таблицы:

$$\begin{array}{cccccc}
 B_0^0 & B_0^1 & B_0^2 & \cdots & B_0^{n-1} & B_0^n \\
 & B_1^1 & B_1^2 & \cdots & B_1^{n-1} & B_1^n \\
 & & B_2^2 & \cdots & \vdots & \vdots \\
 & & & & B_{n-1}^{n-1} & B_{n-1}^n \\
 & & & & & B_n^n
 \end{array}$$

Хранить всю таблицу не обязательно. Достаточно завести вектор  $b$  длины  $n + 1$  и начиная с

$$b = (B_0^0, 0, 0, \dots, 0) = (1, 0, 0, \dots, 0)$$

последовательно заполнить все его компоненты, в итоге получив

$$b = (B_0^n, B_1^n, \dots, B_n^n).$$

▷<sub>3</sub> Запишите соответствующий алгоритм.

При построении графика кривой можно в два раза сократить объём вычислений, если учесть симметрию базисных многочленов Бернштейна (свойство 3): один раз вычислив  $b_i$  в точке  $t$  можно вычислить 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 (5.76). Если обозначить

$$B(t) = B(q_0, \dots, q_n, t),$$

то из указанных двух формул получаем

$$\begin{aligned} B(q_0, \dots, q_n, t) &= (1-t) \sum_{i=0}^n B_i^{n-1} q_i + t \sum_{i=0}^n B_{i-1}^{n-1} q_i = \\ &= (1-t)B(q_0, \dots, q_{n-1}, t) + tB(q_1, \dots, q_n, t). \end{aligned} \quad (5.55)$$

Таким образом, по рекуррентной схеме (5.55) мы можем напрямую вычислить  $B(t)$ , заполняя треугольную таблицу практически аналогично схеме вычисления разделённых разностей. Начинаем с массива

$$(B(q_0, t), B(q_1, t), \dots, B(q_n, t)) = (q_0, q_1, \dots, q_n)$$

постепенно вычисляем  $B(q_i, \dots, q_{i+k}, t)$ , складывая соседние элементы массива с весами  $(1-t)$  и  $t$ . Схема получается такая (точку  $t$  в обозначении  $B(\dots)$  опускаем):

$$\begin{array}{ccccccc} q_0 & & & & & & \\ & B(q_0, q_1) & & & & & \\ & & & & & & \\ q_1 & & B(q_0, q_1, q_2) & & & & \\ & B(q_1, q_2) & & \ddots & & & \\ & & & & & & \\ q_2 & & B(q_1, q_2, q_3) & & B(q_0, \dots, q_{n-1}) & & \\ \vdots & B(q_2, q_3) & \vdots & & & B(q_0, \dots, q_n) & \\ \vdots & \vdots & \vdots & & & & \\ \vdots & \vdots & \vdots & & B(q_1, \dots, q_n) & & \\ \vdots & \vdots & \vdots & \ddots & & & \\ q_{n-1} & \vdots & B(q_{n-2}, q_{n-1}, q_n) & & & & \\ & B(q_{n-1}, q_n) & & & & & \\ & & & & & & \\ q_n & & & & & & \end{array}$$

▷<sub>4</sub> Запишите алгоритм.

▷<sub>5</sub> Сравните трудоёмкость обоих алгоритмов.



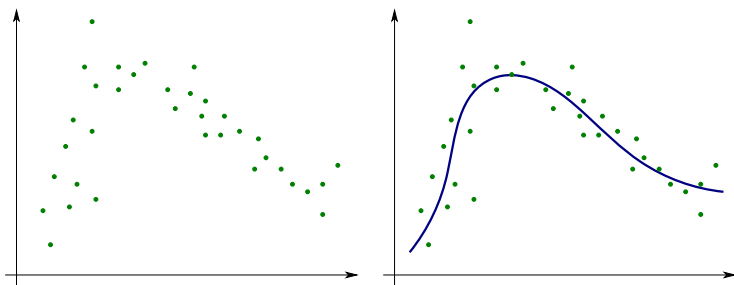
## 5.6. Среднеквадратичные приближения

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  $(x_k, y_k)$ ,  $k = \overline{0, N}$ . Можно считать, что эти точки представляют собой значения некоторой неизвестной функции  $f$ , заданные с какой-то погрешностью:  $f(x_k) \approx y_k$ . Наша задача — найти приближение к функции  $f$ .



Понятно, что интерполяция в данном случае подходит плохо. Вместо нее традиционно применяют метод *среднеквадратичного приближения*, он же *метод наименьших квадратов*. Этот метод состоит в поиске такой функции  $\varphi$ , для которой *сумма квадратов отклонений от  $y_i$  в точках  $x_i$  была бы минимальной*.

**Определение 5.13.** *Задача дискретного среднеквадратичного приближения* для данного набора точек  $(x_k, y_k)$ ,  $k = \overline{0, N}$ , заключается в построении функции  $\varphi$  вида,

$$\varphi(x) = \sum_{i=0}^n \alpha_i \varphi_i(x),$$

для которой выражение

$$\sum_{k=0}^N (y_k - \varphi(x_k))^2 \quad (5.56)$$

принимает минимальное возможное значение. Здесь  $\{\varphi_i\}_{i=0}^n$  — некоторый заданный базис функций.

Искомая функция  $\varphi$  однозначно определяется своими коэффициентами  $\{\alpha_i\}$ , которые представим в виде вектора  $\alpha = (\alpha_0, \alpha_1, \dots, \alpha_n)^T$ . Запишем выражение (5.56) в виде функционала  $\Phi : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ ,

$$\Phi(\alpha) = \sum_{k=0}^N \left( y_k - \sum_{i=0}^n \alpha_i \varphi_i(x_k) \right)^2. \quad (5.57)$$

Задача свелась к нахождению вектора  $\alpha$ , доставляющего минимум  $\Phi(\alpha)$ .

### Вывод расчетных формул

Рассмотрим внимательнее структуру функционала  $\Phi$ :

$$\begin{aligned} \Phi(\alpha) &= \sum_{k=0}^N \left( y_k - \sum_{i=0}^n \alpha_i \varphi_i(x_k) \right)^2 = \\ &= \sum_{k=0}^N \left( y_k^2 - 2y_k \sum_{i=0}^n \alpha_i \varphi_i(x_k) + \left( \sum_{i=0}^n \alpha_i \varphi_i(x_k) \right)^2 \right) = \\ &= \sum_{k=0}^N y_k^2 - 2 \sum_{k=0}^N y_k \sum_{i=0}^n \alpha_i \varphi_i(x_k) + \sum_{k=0}^N \sum_{i=0}^n \alpha_i \varphi_i(x_k) \sum_{j=0}^n \alpha_j \varphi_j(x_k) = \\ &= \sum_{k=0}^N y_k^2 - 2 \sum_{i=0}^n \alpha_i \sum_{k=0}^N y_k \varphi_i(x_k) + \sum_{i=0}^n \sum_{j=0}^n \alpha_i \alpha_j \sum_{k=0}^N \varphi_i(x_k) \varphi_j(x_k). \end{aligned}$$

Если ввести обозначения

$$y = (y_0, y_1, \dots, y_N)^T, \quad \beta_i = \sum_{k=0}^N y_k \varphi_i(x_k), \quad \gamma_{ij} = \sum_{k=0}^N \varphi_i(x_k) \varphi_j(x_k),$$

то полученное выражение можно переписать в виде

$$\Phi(\alpha) = (y, y) - 2 \sum_{i=0}^n \beta_i \alpha_i + \sum_{i=0}^n \sum_{j=0}^n \gamma_{ij} \alpha_i \alpha_j. \quad (5.58)$$

Как видим,  $\Phi(\alpha)$  представляет собой квадратичную форму, причем  $\Phi(\alpha) \geq 0$  при любых  $\alpha \in \mathbb{R}^{n+1}$ .

▷<sub>1</sub> В каком случае минимальное значение  $\Phi$  равно нулю?

Значит, единственный глобальный минимум  $\Phi$  может быть найден как решение системы уравнений

$$\frac{\partial}{\partial \alpha_l} \Phi(\alpha) = 0, \quad l = \overline{0, n}.$$

Непосредственным дифференцированием получаем

$$\frac{\partial}{\partial \alpha_l} \Phi(\alpha) = -2\beta_l + 2 \sum_{j=0}^n \gamma_{lj} \alpha_j.$$

▷<sub>2</sub> Выведите эту формулу.

Следовательно, искомый вектор  $\alpha$  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

$$\Gamma \alpha = \beta, \quad (5.59)$$

$$\begin{aligned} \Gamma &= (\gamma_{ij})_{i,j=0}^n, & \gamma_{ij} &= \sum_{k=0}^N \varphi_i(x_k) \varphi_j(x_k), \\ \beta &= (\beta_0, \beta_1, \dots, \beta_n)^T, & \beta_i &= \sum_{k=0}^N y_k \varphi_i(x_k). \end{aligned} \quad (5.60)$$

Заметим, что матрица  $\Gamma$  представляет собой матрицу скалярных произведений (матрицу Грама) для системы векторов

$$\phi_i = (\varphi_i(x_0), \varphi_i(x_1), \dots, \varphi_i(x_N))^T, \quad i = \overline{0, n}, \quad (5.61)$$

то есть

$$\gamma_{ij} = (\phi_i, \phi_j),$$

а

$$\beta_i = (y, \phi_i).$$

Таким образом, решение задачи дискретного среднеквадратичного приближения сводится к вычислению вектора неизвестных  $\alpha$ , удовлетворяющего СЛАУ (5.59), где матрица  $\Gamma$  и вектор  $\beta$  определяются формулами (5.60). Эта задача однозначно разрешима тогда и только тогда, когда матрица  $\Gamma$  невырождена. В силу вышесказанного, для этого необходимо и достаточно, чтобы система векторов  $\{\phi_i\}$  (5.61) была линейно независимой. Матрица  $\Gamma$ , очевидно, является симметричной. Поэтому для решения СЛАУ (5.59) естественно применить метод квадратного корня.

### 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  $\varphi$  вида,

$$\varphi(x) = \sum_{i=0}^n \alpha_i \varphi_i(x),$$

для которой интеграл

$$\int_a^b (f(x) - \varphi(x))^2 dx \quad (5.62)$$

принимает наименьшее возможное значение. Как обычно, здесь  $\{\varphi_i\}$  — заданная система базисных функций.

*Замечание 5.1.* Иногда вместо (5.62) рассматривают интеграл более общего вида

$$\int_a^b (f(x) - \varphi(x))^2 \rho(x) dx, \quad (5.63)$$

где  $\rho$  — некоторая неотрицательная функция, называемая *весовой функцией*.

### Вывод расчетных формул

Совершенно аналогично дискретному случаю, задача среднеквадратичного приближения функции  $f$  сводится к нахождению вектора коэффициентов  $\alpha = (\alpha_0, \alpha_1, \dots, \alpha_n)^T$ , 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  $\Phi$  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

$$\Psi(\alpha) = \int_a^b f(x)^2 dx - 2 \sum_{i=0}^n \beta_i \alpha_i + \sum_{i=0}^n \sum_{j=0}^n \gamma_{ij} \alpha_i \alpha_j, \quad (5.64)$$

где

$$\beta_i = \int_a^b f(x)\varphi_i(x)dx, \quad \gamma_{ij} = \int_a^b \varphi_i(x)\varphi_j(x)dx, \quad i, j = \overline{0, n}. \quad (5.65)$$

▷<sub>3</sub> Выведите эти формулы.

Заметим, что здесь мы использовали те же обозначения ( $\beta_i$  и  $\gamma_{ij}$ ), что и ранее, так как они имеют схожий смысл.

Далее следуем обкатанной схеме: приравниваем  $\frac{\partial}{\partial \alpha_i}\Psi(\alpha)$  к нулю и получаем следующий результат. *Решение задачи среднеквадратичного приближения функции  $f$  сводится к вычислению вектора неизвестных  $\alpha$ , удовлетворяющего системе линейных уравнений (5.59),*

$$\Gamma\alpha = \beta,$$

где элементы матрицы  $\Gamma = (\gamma_{ij})_{i,j=0}^n$  и вектора  $\beta = (\beta_0, \beta_1, \dots, \beta_n)^T$  определяются формулами (5.65). Матрица  $\Gamma$  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  $\varphi_i$  и  $\varphi_j$ . Эта матрица невырождена тогда и только тогда, когда система функций  $\{\varphi_i\}$  линейно независима.

▷<sub>4</sub> Получите выражения для  $\beta_i$  и  $\gamma_{ij}$  в случае, когда функционал  $\Psi$  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  $\varphi$  была алгебраическим многочленом, самый простой базис в этом случае имеет вид

$$\varphi_i(x) = x^i, \quad i = \overline{0, n}. \quad (5.66)$$

Матрица Грама  $\Gamma$  для этого базиса вычисляется легко (предположим, что  $[a, b] = [0, 1]$ ):

$$\gamma_{ij} = \int_0^1 x^{i+j} dx = \frac{1}{i+j+1},$$

то есть

$$\Gamma = H_{n+1} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+1} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+2} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \frac{1}{n+3} & \cdots & \frac{1}{2n+1} \end{bmatrix}.$$

Это — знаменитая матрица Гильберта, классический пример плохо обусловленной матрицы. Её число обусловленности очень быстро растёт с ростом  $n$ , поэтому на практике базис (5.66) применяется лишь при небольших  $n$ .

▷<sub>5</sub> Вычислите матрицу Грама для степенного базиса в случае произвольного отрезка  $[a, b]$ .

▷<sub>6</sub> Запишите вид соответствующего вектора  $\beta$ .

Вместо базиса (5.66) для более эффективного вычисления среднеквадратичного приближения полиномами используются различные системы *ортogonalных многочленов*, с которыми мы близко познакомимся позже.

### Фундаментальный кусочно-линейный базис

Рассмотрим разбиение отрезка  $[a, b] = [x_0, x_n]$  на  $n$  частей:

$$\Delta = \{\Delta_i\}_{i=1}^n, \quad \Delta_i = [x_{i-1}, x_i] \quad (5.67)$$

и  $S_{\Delta}^1$  — 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  $S_{\Delta}^1$ .

Для решения этой задачи необходимо задать базис пространства  $S_{\Delta}^1$ . Наиболее популярным является фундаментальный базис,  $i$ -ая функ-

ция 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  $S_{\Delta}^1$  однозначно определяются своими значениями в узлах  $\{x_i\}$ .

Аналитические формулы для базисных функций (обозначим их  $\{\psi_i\}$ ) выглядят следующим образом.

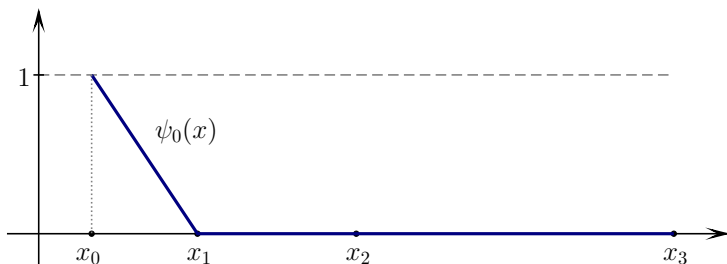
$$\psi_0(x) = \begin{cases} \frac{x_1 - x}{h_1}, & x \in \Delta_1, \\ 0, & x \notin \Delta_1; \end{cases} \quad (5.68a)$$

$$\psi_i(x) = \begin{cases} \frac{x - x_{i-1}}{h_i}, & x \in \Delta_i, \\ \frac{x_{i+1} - x}{h_{i+1}}, & x \in \Delta_{i+1}, \\ 0, & x \notin \Delta_i \cup \Delta_{i+1}, \end{cases} \quad i = \overline{1, n-1}; \quad (5.68b)$$

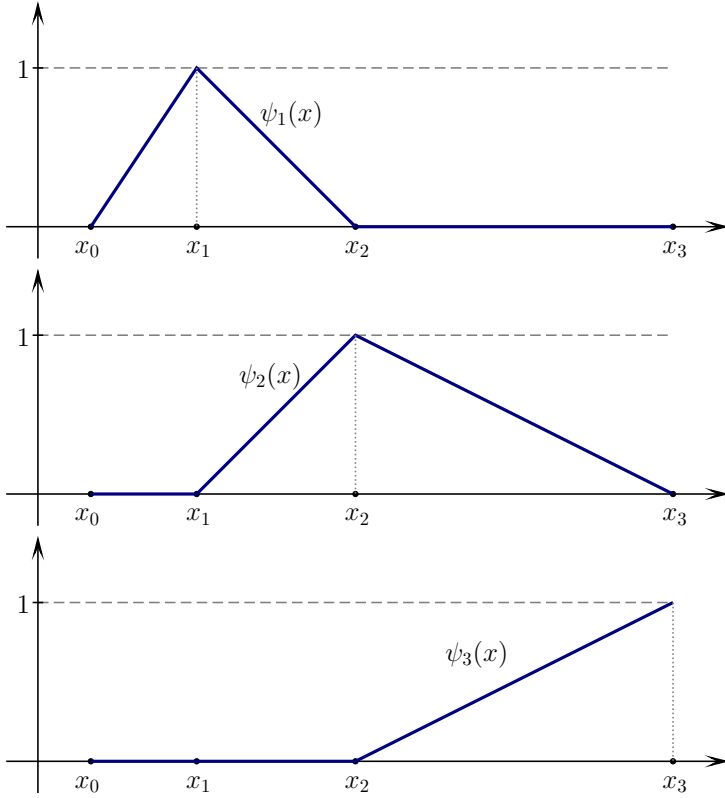
$$\psi_n(x) = \begin{cases} \frac{x - x_{n-1}}{h_n}, & x \in \Delta_n, \\ 0, & x \notin \Delta_n. \end{cases} \quad (5.68c)$$

Напомним обозначения: здесь  $\Delta_i = [x_{i-1}, x_i]$ ,  $h_i = x_i - x_{i-1}$ .

Ниже приведены графики  $\{\psi_i\}$  для частного случая  $n = 3$ .







Интересно вычислить матрицу Грама для построенного базиса. Эта матрица будет трёхдиагональной, так как при  $|i - j| \geq 2$  имеем  $\psi_i(x)\psi_j(x) = 0$  для всех  $x$ . Непосредственным вычислением получаем

$$\Gamma = \begin{bmatrix} d_0 & c_1 & 0 & \cdots & 0 \\ c_1 & d_1 & c_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & c_{n-1} & d_{n-1} & c_n \\ 0 & 0 & 0 & c_n & d_n \end{bmatrix}, \quad (5.69)$$

где

$$\begin{aligned}d_i &= \frac{h_i + h_{i+1}}{3}, \quad i = \overline{1, n-1}, \quad d_0 = \frac{h_1}{3}, \quad d_n = \frac{h_n}{3}, \\c_i &= \frac{h_i}{6}, \quad i = \overline{1, n}.\end{aligned}\tag{5.70}$$

▷<sub>7</sub> 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*)

$$(x_i, y_j), \quad i = \overline{0, n}, \quad j = \overline{0, m},$$

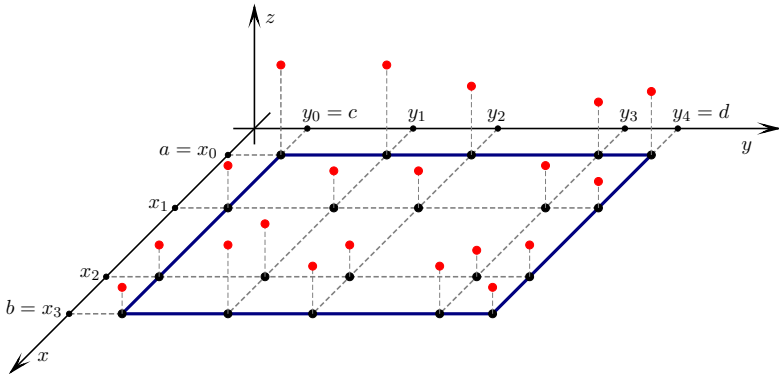
порождаемую одномерными сетками на  $[a, b]$  и  $[c, d]$  соответственно:

$$a = x_0 < x_1 < \dots < x_n = b, \quad \text{и} \quad c = y_0 < y_1 < \dots < y_m = d.$$

Задача интерполяции функции  $f : \Pi \rightarrow \mathbb{R}$  на данной сетке узлов заключается в построении функции  $\varphi : \Pi \rightarrow \mathbb{R}$  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  $\varphi$  как правило задается заранее.



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  $\varphi$  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 «в лоб»: составить систему из  $(n+1)(m+1)$  линейных уравнений для нахождения неизвестных коэффициентов  $\{\alpha_{ij}\}$ . Уравнения, очевидно, 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  $\{y_j\}_{j=0}^m$ :

$$\begin{aligned} \Lambda_i^1(x) &= \prod_{p \neq i} \frac{x - x_p}{x_i - x_p}, \quad i = \overline{0, n}, \\ \Lambda_j^2(y) &= \prod_{q \neq j} \frac{y - y_q}{y_j - y_q}, \quad j = \overline{0, m}. \end{aligned}$$

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

$$\Lambda_{ij}(x_p, y_q) = \begin{cases} 1, & \text{если } p = i \text{ и } q = j \\ 0, & \text{иначе.} \end{cases}$$

Следовательно, многочлен

$$P_{n,m}(x, y) = \sum_{i=0}^n \sum_{j=0}^m z_{ij} \Lambda_{ij}(x, y) \quad (5.71)$$

удовлетворяет условиям интерполяции

$$P_{n,m}(x_i, y_j) = z_{ij}, \quad \forall i = \overline{0, n}, \quad j = \overline{0, m},$$

то есть является искомым интерполяционным многочленом в форме Лагранжа.

▷<sub>1</sub> 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:

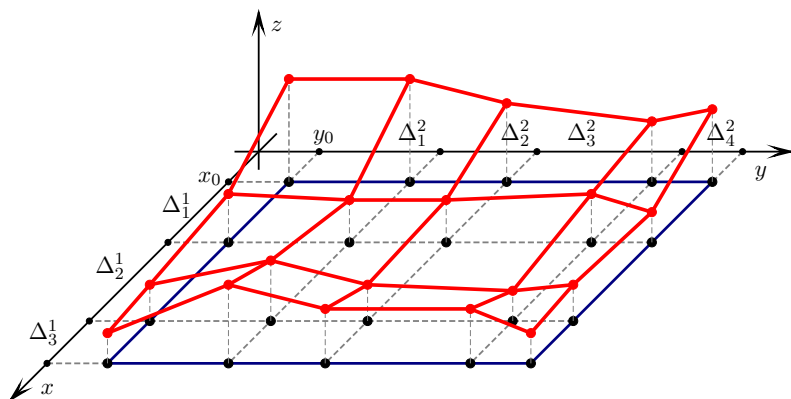
$$\begin{aligned} \Delta_i^1 &= [x_{i-1}, x_i], \quad i = \overline{1, n}, \\ \Delta_j^2 &= [y_{j-1}, y_j], \quad j = \overline{1, m}. \end{aligned}$$

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  $\Delta = \Delta^1 \times \Delta^2$  называется непрерывная функция  $s : \Pi \rightarrow \mathbb{R}$ , которая на каждом прямоугольнике  $\Delta_{ij}$  является *билинейной функцией*:

$$s(x, y) \Big|_{(x, y) \in \Delta_{ij}} = s_{ij}(x, y) = \alpha_{ij}^{00} + \alpha_{ij}^{10}x + \alpha_{ij}^{01}y + \alpha_{ij}^{11}xy. \quad (5.72)$$



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  $\Delta^2$ . А при любом фиксированном  $y \in [c, d]$  выражение  $s(x, y)$ , как функция от  $x$ , является сплайном первого порядка на разбиении  $\Delta^1$ .

Построение интерполяционного билинейного сплайна, удовлетворяющего условиям

$$s(x_i, y_j) = z_{ij},$$

не представляет проблем. На каждом прямоугольнике  $\Delta_{ij}$  «кусоч» сплайна  $s_{ij}$  (5.72) можно построить используя формулу Лагранжа (5.71) по очевидным условиям

$$\begin{aligned} s_{ij}(x_{i-1}, y_{j-1}) &= z_{i-1, j-1}, & s_{ij}(x_i, y_{j-1}) &= z_{i, j-1} \\ s_{ij}(x_{i-1}, y_j) &= z_{i-1, j}, & s_{ij}(x_i, y_j) &= z_{i, j}. \end{aligned}$$

▷<sub>2</sub> 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 мы сейчас рассмотрим. Для начала дадим определение с учетом введенных выше обозначений.

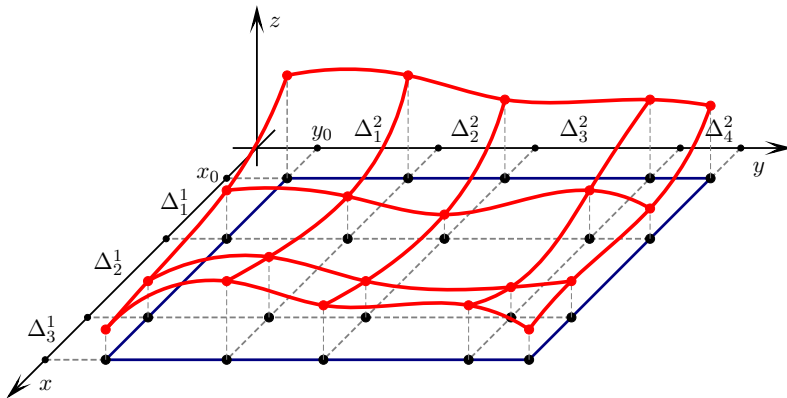
**Определение 5.18.** Бикубическим сплайном, определенным на разбиении  $\Delta = \Delta^1 \times \Delta^2$ , называется функция  $s : \Pi \rightarrow R$  удовлетворяющая следующим двум условиям.

- При любом фиксированном значении  $x \in [a, b]$  выражение  $s(x, y)$ , как функция переменной  $y$ , является кубическим сплайном на разбиении  $\Delta^2$ .
- При любом фиксированном значении  $y \in [c, d]$  выражение  $s(x, y)$ , как функция переменной  $x$ , является кубическим сплайном на разбиении  $\Delta^1$ .

Из этого определения следует, что

- 1) на каждом прямоугольнике  $\Delta_{ij}$  бикубический сплайн  $s$  задается многочленом третьей степени по каждой переменной;
- 2) первая и вторая производные бикубического сплайна, включая смешанные, непрерывны на всем прямоугольнике  $\Pi$ .

Эти два свойства можно считать эквивалентным определением бикубического сплайна.



Займемся теперь построением интерполяционного бикубического сплайна  $s$ , удовлетворяющего условиям

$$s(x_i, y_j) = z_{ij}.$$

Для решения этой задачи, как мы сейчас убедимся, достаточно уметь строить одномерные кубические сплайны (см. раздел 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 это мы делали ранее:

$$s(x, y) \Big|_{x \in \Delta_i^1} = \alpha_i(y) + \beta_i(y)(x - x_i) + \frac{\gamma_i(y)}{2}(x - x_i)^2 + \frac{\delta_i(y)}{6}(x - x_i)^3. \quad (5.73)$$

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  $\Delta^2$ :

$$\alpha_i, \beta_i, \gamma_i, \delta_i \in S_{\Delta^2}^3, \quad i = \overline{1, n}.$$

Как только мы определим эти сплайны, а их в общей сложности  $4n$  штук, искомым бикубический сплайн будет построен.

Необходимые для этого условия получаются из (5.73) подстановкой  $y = y_j, j = \overline{0, m}$ . В обозначениях

$$\alpha_i(y_j) = \alpha_{ij}, \quad \beta_i(y_j) = \beta_{ij}, \quad \delta_i(y_j) = \delta_{ij}, \quad \gamma_i(y_j) = \gamma_{ij}, \quad (5.74)$$

эти условия для каждого  $j$  имеют вид

$$\begin{aligned} s(x, y_j) \Big|_{x \in \Delta_i^1} &= u_j(x) \Big|_{x \in \Delta_i^1} = \\ &= \alpha_{ij} + \beta_{ij}(x - x_i) + \frac{\gamma_{ij}}{2}(x - x_i)^2 + \frac{\delta_{ij}}{6}(x - x_i)^3, \quad i = \overline{1, n}. \end{aligned} \quad (5.75)$$

Здесь  $u_j$  — кубический сплайн на разбиении  $\Delta^1$ . 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 мы помним, 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 разделе 5.4.2, мы находим  $\{\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij}\}_{i=1}^n$ .



После этого остаётся вычислить искомые сплайны  $\alpha_i, \beta_i, \gamma_i, \delta_i$  по условиям интерполяции (5.74) для  $j$  от 0 до  $m$ , а также по соответствующим граничным условиям.

▷<sub>3</sub> Укажите количество и размерность СЛАУ, которых нужно решить для построения бикубического сплайна таким способом.

*Замечание 5.2.* 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. В этом нам поможет 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

$$B(u, v) = \sum_{i=0}^n \sum_{j=0}^m B_i^n(u) B_j^m(v) q_{ij}. \quad (5.76)$$

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.

▷<sub>4</sub> 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 приближённого интегрирования, которые мы и будем рассматривать далее.

Вообще говоря, 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,  $\Delta x_i = x_i - x_{i-1}$ ,  $\xi_i \in [x_{i-1}, x_i]$ . Но такой способ как правило сходится очень медленно. Гораздо эффективнее *приблизить подынтегральную функцию некоторой функцией  $\varphi$ , интеграл от которой вычисляется легко, и заменить интеграл от  $f$  на интеграл от  $\varphi$* . Такие формулы для приближенного вычисления интегралов называются *квадратурными формулами* (строгое определение см. ниже).

### 6.1.2. Простейшие квадратурные формулы

#### Формулы прямоугольников

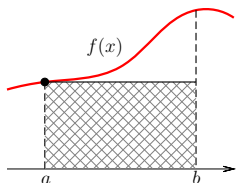
Самые простые квадратурные формулы получаются при замене подынтегральной функции  $f$  на константу, а точнее на значение  $f(x_0)$ , где  $x_0$  — некоторая точка из интервала интегрирования  $[a, b]$ . Площадь криволинейной трапеции, которой по определению равен искомый интеграл

$\int_a^b f(x)dx$ , приближается площадью прямоугольника  $[a, b] \times [0, f(x_0)]$ , то есть

$$\int_a^b f(x)dx \approx f(x_0)(b-a).$$

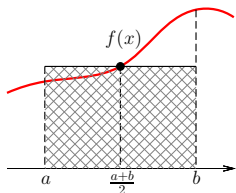
Наиболее известны три квадратурные формулы такого типа:

- *формула левых прямоугольников*



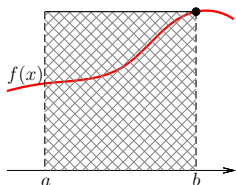
$$\int_a^b f(x)dx \approx (b-a)f(a), \quad (6.1)$$

- *формула средних прямоугольников*



$$\int_a^b f(x)dx \approx (b-a)f\left(\frac{a+b}{2}\right), \quad (6.2)$$

- *формула правых прямоугольников*

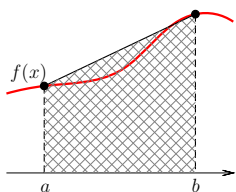


$$\int_a^b f(x)dx \approx (b-a)f(b). \quad (6.3)$$

### 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*:



$$\int_a^b f(x)dx \approx (b-a) \frac{f(a) + f(b)}{2}. \quad (6.4)$$

Эта формула есть ни 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

$$S(f) = \int_a^b f(x)\rho(x)dx. \quad (6.5)$$

Здесь  $\rho$  — некоторая неотрицательная весовая функция, которая может принимать нулевые значения лишь в конечном числе точек. Отображение  $S$ ,

$$S : f \mapsto \int_a^b f(x)\rho(x)dx,$$

представляет собой линейный функционал (отображает функцию в число).

**Определение 6.1.** На отрезке  $[a, b]$  рассмотрим множество из  $n + 1$  парно различных точек  $x_0, x_1, \dots, x_n$ . Формула для приближенного вычисления интеграла  $S(f)$  вида

$$\int_a^b f(x)\rho(x)dx \approx \sum_{i=0}^n A_i f(x_i) \quad (6.6)$$

называется *квадратурной формулой*. Вещественные числа  $\{A_i\}_{i=0}^n$  называются *коэффициентами квадратурной формулы*, а точки

$\{x_i\}_{i=0}^n$  — её *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  $\varphi$ , то есть

$$Q_n(f) = S(\varphi).$$

Если  $\varphi$  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  $\Lambda_i$  — наши любимые базисные функции (5.7). Тогда

$$\begin{aligned} S(P_n) &= S\left(\sum_{i=0}^n \Lambda_i f(x_i)\right) = \left[\int_a^b \sum_{i=0}^n \Lambda_i(x) f(x_i) \rho(x) dx = \right. \\ &= \left. \sum_{i=0}^n f(x_i) \int_a^b \Lambda_i(x) \rho(x) dx\right] = \sum_{i=0}^n S(\Lambda_i) f(x_i) = \sum_{i=0}^n A_i f(x_i) = Q_n(f). \end{aligned}$$

Таким образом, коэффициенты интерполяционной квадратурной формулы с узлами  $\{x_i\}_{i=0}^n$  могут быть вычислены по правилу

$$A_i = S(\Lambda_i) = \int_a^b \Lambda_i(x) \rho(x) dx, \quad (6.7)$$

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

Очевидно, что уже знакомые нам формулы прямоугольников и трапеций являются интерполяционными и соответствуют весовой функции  $\rho(x) \equiv 1$ .

▷<sub>1</sub> Получите формулу трапеций, используя правило (6.7).

### Остаток интерполяционных квадратурных формул

**Определение 6.3.** Остатком квадратурной формулы  $S(f) \approx Q_n(f)$  называется функционал

$$R_n(f) = S(f) - Q_n(f), \quad (6.8)$$

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.

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:

$$R_n(f) = S(f) - Q_n(f) = S(f) - S(P_n) = S(f - P_n) = S(r_n),$$

то есть *остаток интерполяционной квадратурной формулы равен интегралу от остатка интерполяции*. Отсюда, используя формулу (5.24), которая имеет вид

$$r_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x),$$

для достаточно гладких  $f$  получаем

$$R_n(f) = \frac{1}{(n+1)!} \int_a^b f^{(n+1)}(\xi) \omega_{n+1}(x) \rho(x) dx. \quad (6.9)$$

Напомним, что здесь  $\omega_{n+1}(x) = (x - x_0) \dots (x - x_n)$ , а величина  $\xi$  зависит от  $x$ , поэтому острое желание вынести  $f^{(n+1)}(\xi)$  за знак интеграла следует, вообще говоря, подавлять. Однако в некоторых случаях это желание осуществимо. Например, если многочлен  $\omega_{n+1}$  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*

$$\int_a^b f(x)g(x)dx = f(\eta) \int_a^b g(x)dx.$$

Таким образом, если многочлен  $\omega_{n+1}$  знакопостоянен, то по теореме о среднем из (6.9) получаем

$$R_n(f) = \frac{f^{(n+1)}(\eta)}{(n+1)!} \int_a^b \omega_{n+1}(x) \rho(x) dx, \quad \eta \in [a, b]. \quad (6.10)$$

Здесь мы также использовали тот факт, что весовая функция  $\rho$  знакопостоянна по условию.

### 6.1.5. Остаток простейших квадратурных формул

Применим полученные формулы для исследования остатка рассмотренных в пункте 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 вот что:

$$R_0(f) = \frac{(b-a)^2}{2} f'(\eta), \quad \eta \in [a, b]. \quad (6.11)$$

▷<sub>2</sub> Запишите остаток квадратурной формулы правых прямоугольников (6.3).

### Остаток формулы средних прямоугольников

Формула (6.2) наиболее интересна. Перепишем ее в виде

$$S(f) \approx Q_0(f) = (b-a)f\left(\frac{a+b}{2}\right).$$

В этом случае многочлен  $\omega_1 = x - (a+b)/2$  уже не сохраняет знак на отрезке интегрирования, то есть 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  $P_0(x) \equiv f(x_0)$ ,  $x_0 = (a+b)/2$ :

$$\begin{aligned} S(f - P_0) &= \int_a^b \left( f'(x_0)(x - x_0) + \frac{f''(\xi)}{2}(x - x_0)^2 \right) dx = \\ &= \int_a^b \frac{f''(\xi)}{2}(x - x_0)^2 dx = \frac{f''(\eta)}{2} \int_a^b (x - x_0)^2 dx, \end{aligned}$$

откуда

$$R_0(f) = \frac{(b-a)^3}{24} f''(\eta). \quad (6.12)$$

▷<sub>3</sub> Какой вид будет иметь остаток формулы средних прямоугольников в случае, когда  $f''$  не существует, то есть 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

$$R_1(f) = -\frac{(b-a)^3}{12}f''(\eta), \quad \eta \in [a, b]. \quad (6.13)$$

▷<sub>4</sub> Выведите эту формулу. При взятии интеграла будет удобно сделать замену переменных  $x = a + (b-a)t$ .

Сравнивая формулы (6.12) и (6.13) замечаем, формула средних прямоугольников, которая требует лишь одного вычисления функции  $f$ , вообще говоря, точнее формулы трапеций.

## 6.2. Квадратурные формулы Ньютона–Котеса

### 6.2.1. 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 многочлен  $\tilde{p}$  степени  $m + 1$  такой, 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  $\chi_j$ :

$$S(\chi_j) = Q_n(\chi_j) \quad \forall j = \overline{0, m},$$

и при этом

$$S(\chi_{m+1}) \neq Q_n(\chi_{m+1}).$$

▷<sub>1</sub> Докажите это.

Так как интерполяционные квадратурные формулы получаются заменой подынтегральной функции на интерполяционный многочлен степени  $n$ , а все многочлены степени  $m \leq n$  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.**  $\Rightarrow$  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  $m \leq n$ , то имеем  $P_n = f$  и  $Q_n(f) = S(f)$  — формула точна для всех многочленов степени  $n$  и ниже.

$\Leftarrow$  Пусть квадратурная формула

$$S(f) \approx \sum_i A_i f(x_i) = Q_n(f)$$

имеет АСТ не менее  $n$ . Покажем, что её коэффициенты находятся по формуле (6.7):  $A_i = S(\Lambda_i)$ , где  $\Lambda_i$  — фундаментальная базисная функция Лагранжа (5.7). Это и будет означать, что формула интерполяционная. Нет ничего проще: так как  $\deg \Lambda_i = n$ , имеем

$$S(\Lambda_i) = Q_n(\Lambda_i) = \sum_{j=0}^n A_j \Lambda_i(x_j) = A_i. \quad \square$$

▷2 Найдите АСТ всех трех формул прямоугольников и формулы трапеций.

### 6.2.2. Симметричные квадратурные формулы

**Определение 6.5.** Квадратурную формулу

$$S(f) \approx \sum_{i=0}^n A_i f(x_i) = Q_n(f)$$

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  $\rho$  является чётной относительно середины  $[a, b]$ , а КФ  $S(f) \approx Q_n(f)$  является симметричной. Тогда для всякой нечётной относительно  $(a+b)/2$  функции  $f$ ,

$$f(x) = -f(a+b-x),$$

справедливо

$$S(f) = Q_n(f).$$

*Доказательство.* В силу нечётности  $f$  имеем  $S(f) = 0$ , а симметрия узлов даёт  $f(x_i) = -f(x_{n-i})$ . 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** (Повышение АСТ для симметричных КФ). *Пусть весовая функция  $\rho$  является чётной 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 её 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  $\alpha$  и рассмотрим многочлен

$$U(x) = \alpha \left( x - \frac{a+b}{2} \right)^{2M+1}.$$

В силу нечётности 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  $\deg \tilde{P} = \deg(P - U) \leq 2M$ . Тогда по условию  $S(\tilde{P}) = Q_n(\tilde{P})$  и

$$S(P) = S(U) + S(\tilde{P}) = S(\tilde{P}) = Q_n(\tilde{P}) = Q_n(\tilde{P}) + Q_n(U) = Q_n(P). \quad \square$$

▷<sub>3</sub> Укажите в каком месте доказательства используется чётность  $\rho$ .

### 6.2.3. Симметрия интерполяционных КФ

#### Достаточное условие симметричности

**Теорема 6.4.** *Пусть функция  $\rho$  является чётной относительно середины отрезка  $[a, b]$ . Тогда если узлы  $\{x_i\}$  симметричны, то соответствующая им интерполяционная квадратурная формула с весом  $\rho$  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  $\Lambda_i$ :

$$\Lambda_i(x) = \Lambda_{n-i}(a + b - x).$$

Докажем это.

$$\begin{aligned} \Lambda_{n-i}(a + b - x) &= \prod_{j \neq n-i} \frac{a + b - x - x_j}{x_{n-i} - x_j} = \\ &= \prod_{j \neq n-i} \frac{x_{n-j} - x}{(a + b - x_i) - (a + b - x_{n-j})} = \\ &= \prod_{j \neq n-i} \frac{x - x_{n-j}}{x_i - x_{n-j}} = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} = \Lambda_i(x). \end{aligned}$$

Осталось сделать замену переменных в интеграле:

$$\begin{aligned} A_i &= \int_a^b \Lambda_i(x) \rho(x) dx = \int_a^b \Lambda_{n-i}(a + b - x) \rho(x) dx = [z = a + b - x] = \\ &= - \int_b^a \Lambda(z) \rho(a + b - z) dz = \int_a^b \Lambda_{n-i}(z) \rho(z) dz = A_{n-i}. \quad \square \end{aligned}$$

**Следствие 6.1.** Интерполяционная квадратурная формула  $S(f) \approx Q_n(f)$  с четной относительно середины  $[a, b]$  весовой функцией  $\rho$  и симметричными узлами  $\{x_i\}$  при  $n = 2M$  (то есть при нечетном количестве узлов) имеет алгебраическую степень точности не менее  $2M + 1$ .

*Доказательство.* Так как 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  $2M + 1$ .  $\square$

### 6.2.4. 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 и 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, то есть 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

$$\begin{cases} \bar{A}_0 + \bar{A}_1 + \bar{A}_0 = 2, \\ -\bar{A}_0 + \bar{A}_0 = 0, \\ \bar{A}_0 + \bar{A}_0 = \frac{2}{3}, \end{cases}$$

откуда легко находим

$$\bar{A}_0 = \frac{1}{3}, \quad \bar{A}_1 = \frac{4}{3}.$$

Мы получили формулу Симпсона для отрезка  $[-1, 1]$ :

$$\int_{-1}^1 f(x)dx \approx \frac{1}{3}f(-1) + \frac{4}{3}f(0) + \frac{1}{3}f(1). \quad (6.14)$$

Для вывода формулы в общем случае теперь достаточно сделать замену переменной интегрирования:

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{a+b}{2} + \frac{b-a}{2}t\right)dt.$$

Применяя к последнему интегралу формулу (6.14), окончательно получаем *квадратурную формулу Симпсона*

$$\boxed{\int_a^b f(x)dx \approx \frac{b-a}{6} \left( f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right)}. \quad (6.15)$$

▷<sub>4</sub> Аналогичным образом выведите формулы Ньютона–Котеса с четырьмя и пятью узлами.

▷<sub>5</sub> Определите АСТ формулы Симпсона и используя эту информацию получите выражение для её остатка путем разложения в ряд Тейлора в точке  $(a+b)/2$ .



## 6.3. Квадратурные формулы Гаусса

### 6.3.1. 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

$$S(p) = Q_n(p) \quad \forall p \in \mathbb{P}_m \quad (6.16a)$$

и

$$\exists \tilde{p} \in \mathbb{P}_{m+1} : S(\tilde{p}) \neq Q_n(\tilde{p}). \quad (6.16b)$$

Здесь и далее  $\mathbb{P}_m$  — множество всех многочленов степени не выше  $m$ . Мы помним также, что в силу линейности для выполнения условия (6.16a) 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:

$$\boxed{\sum_{i=0}^n A_i x_i^j = \int_a^b x^j \rho(x) dx, \quad j = \overline{0, m};} \quad (6.17a)$$

$$\sum_{i=0}^n A_i x_i^{m+1} \neq \int_a^b x^{m+1} \rho(x) dx. \quad (6.17b)$$

В дальнейшем мы будем рассматривать только условия (6.17a).

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, то есть когда

$$m = 2n + 1.$$

Как мы увидим далее, это и есть максимально возможная АСТ.

### 6.3.2. Построение квадратурных формул Гаусса

Итак, построение квадратурных формул Гаусса, заключается в нахождении узлов  $\{x_i\}$  и коэффициентов  $\{A_i\}$ , удовлетворяющих условиям (6.17a) для  $m = 2n + 1$ . Самый тривиальный способ их нахождения это решение указанной системы «в лоб». 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  $\rho(x) \equiv 1$  получаем формулу средних прямоугольников. При  $n = 1$  систему можно решить относительно легко, а при больших значениях  $n$  задача становится практически неподъемной.

▷<sub>1</sub> Решите систему при  $n = 1$ ,  $[a, b] = [-1, 1]$  и  $\rho(x) \equiv 1$ .

Следующее тривиальное следствие теоремы 6.2 существенно упрощает нашу задачу: *если квадратурная формула имеет наивысшую АСТ, то она является интерполяционной, то есть её коэффициенты имеют вид*

$$A_i = \int_a^b \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \rho(x) dx, \quad i = \overline{0, n}.$$

▷<sub>2</sub> Докажите.

Таким образом задача *упростилась в два раза*: достаточно найти узлы формулы Гаусса, а коэффициенты можно определить по знакомой нам формуле, приведённой выше.

А вот о том, какими должны быть узлы формулы, говорит следующая далее теорема. Она является наиболее важным результатом всей лекции, 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

$$(u, v) = \int_a^b u(x)v(x)\rho(x)dx,$$

Здесь  $\rho$  — заданная неотрицательная на  $[a, b]$  весовая функция. Если  $(u, v) = 0$ , то говорят, что функции  $u$  и  $v$  являются *ортгональными* по весу  $\rho$ .

Перейдем теперь к обещанной теореме.

**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 и*
- 2) *многочлен  $\omega_{n+1}(x) = (x - x_0) \dots (x - x_n)$  был ортогонален по весу  $\rho$  на отрезке  $[a, b]$  ко всем многочленам степени  $n$  и ниже:*

$$(\omega_{n+1}, p) = \int_a^b \omega_{n+1}(x) p(x) \rho(x) dx = 0 \quad \forall p \in \mathbb{P}_n. \quad (6.18)$$

*Доказательство.*  $\Rightarrow$  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  $\deg q \leq 2n+1$ . С учётом того, что  $\omega_{n+1}(x_i) = 0$  при всех  $i$ , получаем

$$S(q) = (\omega_{n+1}, p) = Q_n(q) = \sum_{i=0}^n A_i \omega_{n+1}(x_i) p(x_i) = 0.$$

То, что формула интерполяционная, мы уже доказывали.

$\Leftarrow$  Пусть верно (6.18). Возьмём любой  $q \in \mathbb{P}_{2n+1}$  и разделим его на  $\omega_{n+1}$  с остатком:

$$q(x) = \omega_{n+1}(x)p(x) + r(x),$$

где  $\deg p \leq n$ ,  $\deg r \leq n$ . Тогда с учётом  $q(x_i) = r(x_i)$  имеем

$$S(q) = (\omega_{n+1}, p) + S(r) = S(r) = \sum_{i=0}^n A_i r(x_i) = \sum_{i=0}^n A_i q(x_i),$$

то есть формула точна для любого  $q \in \mathbb{P}_{2n+1}$ .  $\square$

Таким образом, узлами квадратурной формулы Гаусса являются корни многочлена, ортогонального по весу  $\rho$  всем многочленам степени  $n$  и ниже. Многочлен, удовлетворяющий этим условиям, в дальнейшем будем обозначать  $\omega_{n+1}^*$ .

Мы получили прекрасный результат, но для строгости его доказательства нужно а) показать, что нельзя достичь АСТ, большей  $2n + 1$  и б) доказать существование узлов формул Гаусса на отрезке  $[a, b]$  (а вдруг они вообще комплексные?).

**Теорема 6.6.** *Если весовая функция  $\rho$  знакопостоянна на отрезке  $[a, b]$ , то не существует квадратурных формул с  $(n + 1)$  узлами, имеющих АСТ  $2n + 2$ .*

*Доказательство.* Предположим существование такой формулы. Пусть  $\{x_i\}_{i=0}^n$  — её 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.  $\square$

**Теорема 6.7.** *Если весовая функция  $\rho$  сохраняет знак на отрезке  $[a, b]$ , то многочлен  $\omega_{n+1}^*$  степени  $(n + 1)$ , ортогональный на данном отрезке по весу  $\rho$  ко всем многочленам степени  $n$  и ниже, существует и единственен для любого фиксированного  $n$ . При этом все его корни действительны, различны и лежат внутри  $[a, b]$ .*

*Замечание 6.2.* Рассмотрим множество многочленов  $\{\omega_i^*\}_{i=1}^\infty$ . Так как по определению  $(\omega_{n+1}^*, p) = 0$  для всех  $p \in \mathbb{P}_n$ , имеем

$$(\omega_i^*, \omega_j^*) = 0 \quad \text{если} \quad i \neq j,$$

то есть множество  $\{\omega_i^*\}$  представляет собой систему ортогональных многочленов (по весу  $\rho$ ). 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  $\rho$  многочлена степени  $n + 1$ .*

### 6.3.3. Классические квадратурные формулы Гаусса

#### Единичная весовая функция

Рассмотрим случай единичной весовой функции на отрезке  $[-1, 1]$ , то есть 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 *многочленов Лежандра*, которые являются ортогональными на  $[-1, 1]$  с единичным весом. Многочлены Лежандра можно определить явной формулой

$$L_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} ((x^2 - 1)^n), \quad (6.19)$$

или рекуррентным соотношением

$$\begin{aligned} L_{n+1}(x) &= \frac{2n+1}{n+1} x L_n(x) - \frac{n}{n+1} L_{n-1}(x), \\ L_0(x) &= 1, \quad L_1(x) = x. \end{aligned} \quad (6.20)$$

▷<sub>3</sub> Постройте 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 мы имеем формулу Гаусса вида

$$\int_{-1}^1 f(x)dx \approx \sum_{i=0}^n A_i f(x_i).$$

Тогда

$$\begin{aligned} \int_a^b f(x)dx &= \frac{b-a}{2} \int_{-1}^1 \underbrace{f\left(\frac{a+b}{2} + \frac{b-a}{2}t\right)}_{g(t)} dt = \frac{b-a}{2} \int_{-1}^1 g(t)dt \approx \\ &\approx \frac{b-a}{2} \sum_{i=0}^n A_i g(x_i) = \frac{b-a}{2} \sum_{i=0}^n A_i f\left(\frac{a+b}{2} + \frac{b-a}{2}x_i\right). \end{aligned}$$

### Другие классические формулы

Для некоторых весовых функций известны явные формулы, определяющие ортогональные многочлены, а также явные рекуррентные соотношения. Для этих многочленов давным-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.

$\rho(x)$	Отрезок	Название системы многочленов
$\frac{1}{\sqrt{1-x^2}}$	$[-1, 1]$	Чебышева
$x^\alpha e^{-x}$	$[0, +\infty)$	Лагерра
$e^{-x^2}$	$(-\infty, +\infty)$	Эрмита
$(1-x)^\alpha(1+x)^\beta$	$[-1, 1]$	Якоби

В частности, мы видим, что знакомые уже 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:

$$A_0 = \dots = A_n = \frac{\pi}{n+1}.$$

### Другие весовые функции

Возникает вопрос: каким образом строить гауссовы квадратурные формулы в общем случае, когда не известны 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 бесхитростный подход. Он позволяет для данного  $n$  найти непосредственно многочлен  $\omega_{n+1}^*$  в явном виде

$$\omega_{n+1}^*(x) = x^{n+1} + c_n x^n + c_{n-1} x^{n-1} + \dots + c_1 x + c_0. \quad (6.21)$$

Для этого просто воспользуемся определением:  $\omega_{n+1}^*$  должен быть ортогонален всем многочленам степени  $n$  и ниже, а для этого необходимо и достаточно потребовать ортогональности для базиса  $\chi_i(x) = x^i$ :

$$\int_a^b \omega_{n+1}^*(x) x^i \rho(x) dx = 0, \quad i = \overline{0, n}.$$

Подставляя сюда представление (6.21), получаем систему линейных уравнений вида

$$\gamma_{i0} c_0 + \gamma_{i1} c_1 + \dots + \gamma_{i, n-1} c_{n-1} + \gamma_{i, n} c_n = -g_i, \quad i = \overline{0, n}, \quad (6.22)$$

где

$$\gamma_{ij} = \int_a^b x^{i+j} \rho(x) dx, \quad g_i = \int_a^b x^{n+i+1} \rho(x) dx.$$

▷<sub>4</sub> Выведите эти формулы.

После того, как найден многочлен  $\omega_{n+1}^*$ , остается найти его корни и вычислить коэффициенты квадратурной формулы по правилу (6.7).

## 6.4. Составные квадратурные формулы

### 6.4.1. Введение

Пусть  $\rho \equiv 1$ . Рассмотрим задачу вычисления интеграла

$$S(f) = \int_a^b f(x) dx$$

с некоторой заданной точностью. Для этого разобьём 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

$$S(f) = \sum_{k=1}^N S_k(f), \quad S_k(f) = \int_{\xi_{k-1}}^{\xi_k} f(x) dx.$$

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

$$S(f) \approx Q_n^N(f) = \sum_{k=1}^N Q_{n,k}(f).$$

### 6.4.2. Простейшие составные формулы

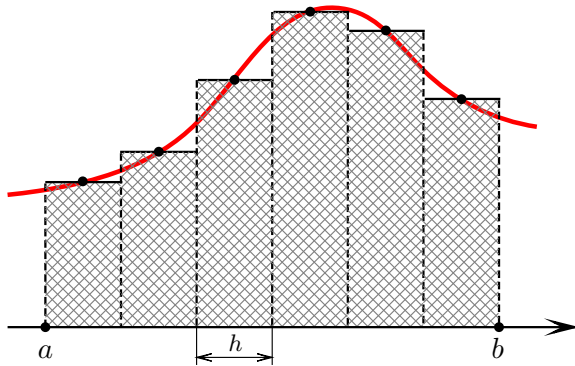
Возьмем в качестве базовой, например, формулу средних прямоугольников (6.2):

$$\int_{\xi_{k-1}}^{\xi_k} f(x) dx \approx (\xi_k - \xi_{k-1}) f\left(\frac{\xi_{k-1} + \xi_k}{2}\right).$$



С учетом того, что  $\xi_k = a + kh$ , после суммирования всех частей получаем *составную формулу средних прямоугольников*

$$\int_a^b f(x) dx \approx h \sum_{k=1}^N f\left(a + \left(k - \frac{1}{2}\right)h\right). \quad (6.23)$$

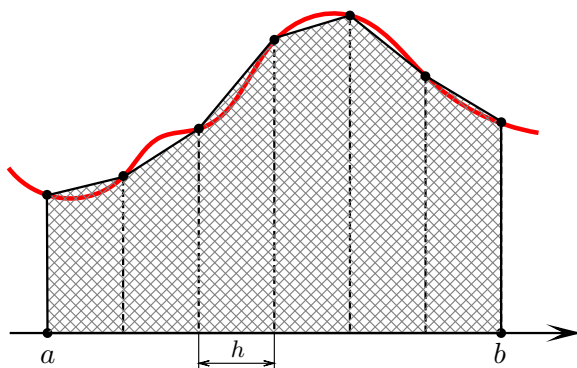


Аналогично строится *составная формула трапеций*:

$$\int_{\xi_{k-1}}^{\xi_k} f(x) dx \approx \frac{h}{2} (f(\xi_{k-1}) + f(\xi_k)),$$

откуда

$$\int_a^b f(x) dx \approx \frac{h}{2} \left( f(a) + 2 \sum_{k=1}^{N-1} f(a + kh) + f(b) \right). \quad (6.24)$$



▷<sub>1</sub> Постройте составные формулы левых, правых прямоугольников и Симпсона.

### 6.4.3. 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 лекции обозначениям, имеем

$$\begin{aligned} R_n^N(f) &= S(f) - Q_n^N(f) = \sum_{k=1}^N S_k(f) - \sum_{k=1}^N Q_{n,k}(f) = \\ &= \sum_{k=1}^N (S_k(f) - Q_{n,k}(f)) = \sum_{k=1}^N R_{n,k}(f). \end{aligned}$$

Таким образом, как нетрудно было догадаться сразу, *остаток составной квадратурной формулы равен сумме остатков её 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, см. (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

$$y_{\min} \leq \frac{1}{N} \sum_{k=1}^N f^{(m)}(\eta_k) \leq y_{\max}.$$

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

$$f^{(m)}(\eta) = \frac{1}{N} \sum_{k=1}^N f^{(m)}(\eta_k). \quad (6.25)$$

Следовательно, формула для остатка составной КФ примет вид

$$R_n^N(f) = Ch^l N f^{(m)}(\eta), \quad (6.26a)$$

или, с учётом  $N = (b - a)/h$ ,

$$\boxed{R_n^N(f) = C(b - a)h^{l-1} f^{(m)}(\eta).} \quad (6.26b)$$

*Замечание 6.3.* Напомним, что здесь  $\eta$  — некоторая точка из отрезка  $[a, b]$ , причем эта точка зависит от разбиения, то есть для различных  $N$  эти точки будут разными. Другими словами,  $\eta$  зависит от  $h$ .

Отсюда для составной формулы средних прямоугольников (6.23) из (6.12) получаем остаток

$$\boxed{R_0^N(f) = \frac{b-a}{24} h^2 f''(\eta),} \quad (6.27)$$

а для составной формулы трапеций (6.24) из (6.13) имеем

$$R_1^N(f) = -\frac{b-a}{12}h^2 f''(\eta). \quad (6.28)$$

Имея формулы такого рода, теоретически можно заранее оценить необходимое число отрезков разбиения  $N$ , которое необходимо взять для вычисления интеграла с требуемой точностью. Для этого, естественно, нужно располагать информацией о функции  $f^{(m)}$  на отрезке  $[a, b]$ . Понятно, что на практике такая информация доступна не всегда, поэтому в универсальных программах численного интегрирования используется несколько иной способ представления погрешности составных формул.

#### 6.4.4. Выделение главной части погрешности

Рассмотрим формулу остатка составной формулы прямоугольников (6.28). Она говорит, что остаток имеет вид  $O(h^2)$  если  $f''$  существует и непрерывна на  $[a, b]$ :

$$S(f) = Q_0^N(f) + O(h^2). \quad (6.29)$$

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  $z_k$ :

$$\begin{aligned} S_k(f) = \int_{\xi_{k-1}}^{\xi_k} f(x) dx &= \int_{\xi_{k-1}}^{\xi_k} \left[ f(z_k) + (x - z_k)f'(z_k) + \right. \\ &\quad \left. + \frac{(x - z_k)^2}{2} f''(z_k) + \frac{(x - z_k)^3}{6} f'''(z_k) + \frac{(x - z_k)^4}{24} f^{(4)}(\zeta_k) \right] dx. \end{aligned}$$

Здесь  $\zeta_k$  — 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, получаем

$$S_k(f) = hf(z_k) + \frac{h^3}{24} f''(z_k) + \frac{h^5}{1920} f^{(4)}(\eta_k),$$

$\eta_k$  — точка из  $[\xi_{k-1}, \xi_k]$ , отличная, вообще говоря, от  $\zeta_k$ . 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

$$\frac{h^3}{24} \sum_{k=1}^N f''(z_k) = \frac{h^2}{24} \left( \int_a^b f''(x) dx + O(h^2) \right).$$

В третьем слагаемом применим формулу (6.25) и получим

$$\frac{h^5}{1920} \sum_{k=1}^N f^{(4)}(\eta_k) = \frac{h^5}{1920} N f''(\eta) = \frac{h^4}{1920} (b-a) f''(\eta).$$

Собирая все вместе, получаем

$$S(f) = Q_0^N(f) + \frac{h^2}{24} \int_a^b f''(x) dx + O(h^4),$$

или

$$\boxed{R_0^N(f) = \frac{f'(a) - f'(b)}{24} h^2 + O(h^4)} \quad (6.30)$$

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.

### 6.4.5. Практическая оценка погрешности: правило Рунге

Составные квадратурные формулы по построению представляют собой инструмент, позволяющий регулировать точность вычисления интеграла за счёт увеличения числа отрезков разбиения  $N$  (уменьшения шага  $h$ ). Понятно, что для вычислений на практике нужно уметь выбирать такое  $N$ , которое бы позволило достичь требуемой точности  $\varepsilon$ , 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 пересчёта*.

Пусть  $Q_n^N = Q^N$  — некоторая составная квадратурная сумма. Предположим, что для остатка соответствующей квадратурной формулы справедливо представление, аналогичное (6.30):

$$R^N(f) = S(f) - Q^N(f) = Kh^p + o(h^p), \quad h = \frac{b-a}{N}, \quad (6.31)$$

где константа  $K$  не зависит от  $h$ . Здесь  $Kh^p$  представляет собой главную часть погрешности, которую можно найти, если известно значение константы  $K$ .

Вычислим приближённое 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 приближённые 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 помощью нее получаем оценку погрешности для обеих квадратурных сумм:

$$R^{N_i}(f) \approx K h_i^p \approx \tilde{K} h_i^p = \tilde{R}_i,$$

где

$$\boxed{\tilde{R}_i = \frac{(q_2 - q_1)h_i^p}{h_1^p - h_2^p}}. \quad (6.32)$$

Во-вторых, имея  $\tilde{R}_i$ , мы можем уточнить значения  $q_i$ :

$$S(f) = q_i + R^{N_i}(f) \approx q_i + \tilde{R}_i. \quad (6.33)$$

Понятно, что для получения максимальной точности лучше всего уточнять  $q_2$ .

В-третьих, с помощью  $\tilde{K}$  мы имеем возможность оценить оптимальную величину шага  $h^*$  для достижения точности  $\varepsilon$ . Делается это с помощью очевидного требования

$$|R^N(f)| \approx |\tilde{K}|h^p \leq \varepsilon,$$

из которого получаем

$$h \leq \left( \frac{\varepsilon}{|\tilde{K}|} \right)^{1/p} = \left( \frac{\varepsilon}{|\tilde{R}_i|} \right)^{1/p} h_i = h^*,$$

или

$$N \geq N^* = \frac{b-a}{h^*} = \left( \frac{|\tilde{R}_i|}{\varepsilon} \right)^{1/p} N_i. \quad (6.34)$$

**Практические рекомендации.** Теперь поговорим о том, как всё это реализовывается на практике. Простейший алгоритм выглядит следующим образом.

- 1) Выбираем квадратурную формулу  $Q^N$ ,  $N_1$  и  $N_2$ , точность  $\varepsilon$ .

- 2)  $q_1 \leftarrow Q^{N_1}(f)$ ,  $q_2 \leftarrow Q^{N_2}(f)$ .
- 3) По формуле (6.32) вычисляем оценку погрешности для «более точного» 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  $|\tilde{R}| \leq \varepsilon$ , 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

$$\begin{aligned} Q_1^N(f) &= \frac{h}{2} \left( f(a) + 2 \sum_{k=1}^{N-1} f(a + kh) + f(b) \right), \\ Q_1^{2N}(f) &= \frac{h}{4} \left( f(a) + 2 \sum_{k=1}^{2N-1} f(a + k\frac{h}{2}) + f(b) \right) = \\ &= \frac{1}{2} \left( Q_1^N(f) + h \sum_{k=1}^N f(a + (2k-1)\frac{h}{2}) \right). \end{aligned}$$

▷<sub>2</sub> Постройте соответствующую модификацию алгоритма.



- **Использование уточнённого приближения:** на шаге 4 можно возвращать не  $q_2$ , а его уточнённое 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 остаётся 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)

$$\boxed{y'(x) = f(x, y(x))}, \quad (7.1a)$$

и начальное условие

$$\boxed{y(x_0) = y_0, \quad x_0 \in \mathbb{R}, \quad y_0 \in \mathbb{R}^n}. \quad (7.1b)$$

Здесь  $y : \mathbb{R} \rightarrow \mathbb{R}^n$  — искомая вектор-функция,  $f : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$  — функция, которую иногда называют просто *правой частью* ОДУ. Формулы (7.1) вместе определяют *задачу Коши* для системы ОДУ.

Для развернутой, то есть 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  $y^i : \mathbb{R} \rightarrow \mathbb{R}$  — скалярные функции-компоненты вектор-функции  $y$ , аналогично  $f^i : (\mathbb{R} \times \mathbb{R}^n) \rightarrow \mathbb{R}$  — 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

$$\begin{cases} \frac{d}{dx} y^1(x) = f^1(x, y^1(x), \dots, y^n(x)), & y^1(x_0) = y_0^1; \\ \frac{d}{dx} y^2(x) = f^2(x, y^1(x), \dots, y^n(x)), & y^2(x_0) = y_0^2; \\ \vdots \\ \frac{d}{dx} y^n(x) = f^n(x, y^1(x), \dots, y^n(x)), & y^n(x_0) = y_0^n; \end{cases}$$

Во многих случаях формулы (7.1) удобно объединить в одну путём интегрирования:

$$y(x) = y_0 + \int_{x_0}^x f(z, y(z)) dz. \quad (7.2)$$

Пусть нам необходимо найти значение решения в некоторой точке  $x_1 > x_0$ . На практике редко удаётся вычислить  $y(x_1)$  точно, поэтому приходится прибегать к приближённым (численным) методам.

### 7.1.2. Методы Эйлера

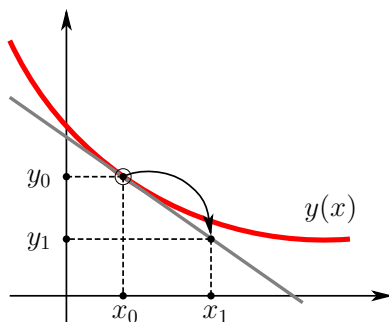
#### Явный метод Эйлера

Рассмотрим скалярный случай  $n = 1$ . Начальное условие (7.1b) 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  $y(x_1) \approx y_1$ , получаем самый простой и самый известный метод — *явный метод Эйлера*

$$y_1 = y_0 + (x_1 - x_0)f(x_0, y_0). \quad (7.3)$$



### Неявный метод Эйлера

Рассмотрим соотношение (7.2) для  $x = x_1$ :

$$y(x_1) = y_0 + \int_{x_0}^{x_1} f(z, y(z)) dz.$$

Приблизим интеграл по правилу правых прямоугольников<sup>1</sup>

$$\int_a^b f(x) dx \approx (b - a) f(b)$$

и получим неявное соотношение для вычисления  $y_1 \approx y(x_1)$ :

$$y_1 = y_0 + (x_1 - x_0) f(x_1, y_1). \quad (7.4)$$

Это *неявный* метод Эйлера. Очевидно, что для вычисления  $y_1$  необходимо решать нелинейное уравнение (систему из  $n$  нелинейных уравнений в общем случае).

▷<sub>1</sub> В чем состоит геометрический смысл неявного метода Эйлера? Изобразите соответствующий рисунок.

▷<sub>2</sub> Примените для аппроксимации интеграла (7.2) квадратурную формулу трапеций и запишите формулу для соответствующего метода.

### 7.1.3. Одношаговые методы: терминология

Итак, методы Эйлера позволяют вычислить приближённое 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 приближённого решения задачи (7.1) в данной точке  $x_1 \in \mathbb{R}$ :

$$\Phi(x_0, y_0, x_1) = y_1 \approx y(x_1).$$

<sup>1</sup>Заметим, что если применить формулу левых прямоугольников, получится явный метод Эйлера (7.3).

Главное различие между двумя методами Эйлера состоит в том, что в одном случае  $y_1$  задаётся 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  $\Phi$  является неявной функцией, то есть задано уравнением вида

$$\Psi(x_0, y_0, x_1, y_1) = 0,$$

то соответствующий одношаговый метод называется *неявным*. В противном случае, то есть если  $y_1$  явно выражается через  $x_0$ ,  $y_0$  и  $x_1$ , метод называется *явным*.

Как правило, отображение  $\Phi$  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  $\varphi : \mathbb{R} \rightarrow \mathbb{R}^n$ , определённое по правилу

$$\varphi(x) = \Phi(x_0, y_0, x).$$

Обычно вычислительно необходимо получить приближённое решение задачи Коши (7.1) на «большом» отрезке  $[x_0, x_0 + H]$ . Так как функция  $\varphi$  даёт приемлемое приближение лишь в достаточно малой окрестности точки  $x_0$ , отрезок интегрирования разбивается на  $N$  частей сеткой узлов

$$x_0 < x_1 < x_2 \dots < x_N = x_0 + H \quad (7.5)$$

и строится набор приближённых значений  $y_k \approx y(x_k)$  по правилу

$$y_{k+1} = \Phi(x_k, y_k, x_{k+1}), \quad k = \overline{0, N-1}. \quad (7.6)$$

#### 7.1.4. Порядок метода

Классическим показателем точности методов численного интегрирования ОДУ является порядок метода. Это понятие связано с разложением точного и приближённого решений в ряд Тейлора.

Рассмотрим сначала  $y$  — точное решение уравнения (7.1) в скалярном случае и предположим, что оно достаточно гладкое. Введём следующие традиционные обозначения:

$$f = f(x_0, y_0), \quad f_x = \frac{\partial f}{\partial x}(x_0, y_0), \quad f_y = \frac{\partial f}{\partial y}(x_0, y_0), \quad (7.7)$$

$$\underbrace{f x \dots x}_M \underbrace{y \dots y}_N = \frac{\partial^{M+N} f}{\partial x^M \partial y^N}(x_0, y_0). \quad (7.8)$$

Разложение Тейлора для  $y(x_0 + h)$  в точке  $x_0$  имеет вид

$$y(x) = y_0 + h y'(x_0) + \frac{h^2}{2!} y''(x_0) + \frac{h^3}{3!} y'''(x_0) + \dots \quad (7.9)$$

Тот факт, что  $y$  удовлетворяет (7.1), позволяет вычислить неизвестные коэффициенты  $y^{(k)}(x_0)$  для любого  $k$ . Действительно, мы имеем

$$y'(x_0) = f, \quad (7.10a)$$

$$y''(x_0) = \left. \frac{d}{dx} f(x, y(x)) \right|_{x=x_0} = f_x + f_y y'(x_0) = f_x + f_y f. \quad (7.10b)$$

$$y'''(x_0) = \left. \frac{d^2}{dx^2} f(x, y(x)) \right|_{x=x_0} = f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y(f_x + f_y f), \quad (7.10c)$$

и так далее. Таким образом, мы имеем теоретическую возможность *точно* вычислить разложение Тейлора (7.9).

Теперь зафиксируем начальное условие  $(x_0, y_0)$ , рассмотрим произвольный одношаговый метод  $\Phi$  и  $\varphi(x) = \Phi(x_0, y_0, x)$ . Традиционный способ оценки точности такого приближения заключается в сравнении разложений 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  $\Phi$  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 имеет вид

$$r(x_0 + h) = C h^{p+1} + O(h^{p+2}), \quad C \neq 0, \quad (7.11)$$

то есть разложения Тейлора в точке  $x_0$  для точного решения  $y(x_0 + h)$  и приближённого  $\varphi(x_0 + h)$  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.

### Примеры вычисления порядка

**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 получаем

$$\begin{aligned} \varphi(x_0) &= y_0, \\ \varphi'(x_0) &= f(x_0, y_0) = y'(x_0), \\ \varphi^{(k)}(x_0) &= 0 \neq y^{(k)}(x_0) \quad \forall k \geq 2. \end{aligned}$$

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 (7.10b), равна

$$C = \frac{1}{2}(f_x + f_y f). \quad (7.12)$$

**Порядок неявного метода Эйлера.** Рассмотрим (7.4). Для этого метода приближённое 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 учётом того, что  $\varphi(x_0) = y_0$ , получаем

$$\begin{aligned}\varphi'(x_0) &= f(x_0, y_0) = y'(x_0), \\ \varphi''(x_0) &= 2 \frac{d}{dx} f(x_0, y_0) = 2(f_x + f_y f) \neq y''(x_0).\end{aligned}$$

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

$$C = -\frac{1}{2}(f_x + f_y f). \quad (7.13)$$

*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  $(x_0, y_0)$ .

▷<sub>3</sub> Найдите порядок метода, построенного в упражнении 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. Настало время построить чуть 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  $z = x_0 + th$ :

$$y(x_0 + h) = y_0 + h \int_0^1 f(x_0 + th, y(x_0 + th)) dt. \quad (7.14)$$

Приближим интеграл квадратурной формулой средних прямоугольников:

$$y(x_0 + h) \approx y_0 + hf\left(x_0 + \frac{h}{2}, y(x_0 + \frac{h}{2})\right).$$

Использовать эту формулу для вычислений пока что нельзя, так как неизвестно значение  $y(x_0 + \frac{h}{2})$ . Но его можно приблизить любым из методов Эйлера (7.3), (7.4). Для начала возьмём явный метод:

$$y\left(x_0 + \frac{h}{2}\right) \approx y_0 + \frac{h}{2}f(x_0, y_0),$$

и получим численный метод вида

$$Y_2 = y_0 + \frac{h}{2}f(x_0, y_0), \quad (7.15a)$$

$$y_1 = y_0 + hf\left(x_0 + \frac{h}{2}, Y_2\right). \quad (7.15b)$$

Этот метод будем называть *явным методом средних прямоугольников (средней точки)*.

Если же приблизить  $y(x_0 + \frac{h}{2})$  неявным методом Эйлера

$$y\left(x_0 + \frac{h}{2}\right) \approx Y_1 = y_0 + \frac{h}{2}f\left(x_0 + \frac{h}{2}, Y_1\right),$$

получим *неявный метод средних прямоугольников*

$$Y_1 = y_0 + \frac{h}{2}f\left(x_0 + \frac{h}{2}, Y_1\right), \quad (7.16a)$$

$$y_1 = y_0 + hf\left(x_0 + \frac{h}{2}, Y_1\right). \quad (7.16b)$$

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  $\{c_1, \dots, c_s\}$  и весами  $\{b_1, \dots, b_s\}$ :

$$\int_0^1 f(x_0 + th, y(x_0 + th))dt \approx \sum_{i=1}^s b_i f(x_0 + c_i h, y(x_0 + c_i h)).$$

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 приближаем квадратурной формулой *по тем же самым узлам*  $\{c_1, \dots, c_s\}$ , но с другими коэффициентами  $\{a_{i1}, \dots, a_{is}\}$ :

$$y(x_0 + c_i h) \approx y_0 + h \sum_{j=1}^s a_{ij} f(x_0 + c_j h, y(x_0 + c_j h)).$$

Обозначая  $y(x_0 + c_i h) \approx Y_i$ , в итоге получаем  $s$ -стадийный метод Рунге–Кутты общего вида:

$$\boxed{\begin{aligned} Y_i &= y_0 + h \sum_{j=1}^s a_{ij} f(x_0 + c_j h, Y_j), \quad i = \overline{1, s}, \\ y_1 &= y_0 + h \sum_{i=1}^s b_i f(x_0 + c_i h, Y_i). \end{aligned}} \quad (7.17)$$

Запись метода РК в виде (7.17) называется *симметричной*. Более распространённой является другая, эквивалентная форма записи, которая получается заменой переменных

$$\kappa_i = f(x_0 + c_i h, Y_i), \quad (7.18)$$

или

$$Y_i = y_0 + h \sum_{j=1}^s a_{ij} \kappa_j. \quad (7.19)$$

В результате имеем

$$\boxed{\begin{aligned} \kappa_i &= f\left(x_0 + c_i h, y_0 + h \sum_{j=1}^s a_{ij} \kappa_j\right), \quad i = \overline{1, s}, \\ y_1 &= y_0 + h \sum_{i=1}^s b_i \kappa_i. \end{aligned}} \quad (7.20)$$

Таким образом,  $s$ -стадийный метод РК определяется  $s^2 + 2s$  параметрами  $\{a_{ij}\}_{i,j=1}^s$ ,  $\{b_1, \dots, b_s\}$  и  $\{c_1, \dots, c_s\}$ . Традиционно эти параметры размещают в таблицу вида

$$\frac{c}{A} \Big| \frac{b^T}{b} = \begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}. \quad (7.21)$$

Матрицу  $A = (a_{ij})$  называют *матрицей Бутчера*<sup>2</sup>, а всю таблицу (7.21) — *таблицей Бутчера*.

<sup>2</sup>Джон Бутчер (John Butcher), университет Окленда, — новозеландский математик, внесший большой вклад в развитие и популяризацию методов РК.

▷<sub>1</sub> Постройте таблицу Бутчера для методов Эйлера, а также для явного и неявного метода средних прямоугольников.

Заметим, что первая формула в (7.20), как и её симметричный аналог из (7.17), задаёт систему нелинейных уравнений относительно неизвестных  $\{\kappa_i\}_{i=1}^s$  ( $\{Y_i\}_{i=1}^s$  соответственно). То есть методы РК являются в общем случае неявными. Однако если коэффициенты метода удовлетворяют условиям

$$a_{ij} = 0 \quad \forall i \leq j$$

и

$$c_1 = 0,$$

то такой метод РК, очевидно, является явным.

### 7.2.3. Условия порядка для методов РК

Неизвестные коэффициенты методов типа Рунге–Кутты (7.21) традиционно выбираются таким образом, чтобы получившийся метод имел как можно более высокий порядок точности.

Найдём условия, которым должен удовлетворять произвольный метод РК второго порядка. Рассмотрим приближённое решение  $y_1$  как функцию переменной  $h$ . Так как функция метода имеет вид  $\varphi(x_0 + h) = y_1$ , имеем

$$\left. \frac{d^k \varphi}{dx^k} \right|_{x=x_0} = \left. \frac{d^k y_1}{dh^k} \right|_{h=0}.$$

Тогда для того, чтобы метод имел порядок 2, по определению необходимо, чтобы выполнялись условия

$$y_1|_{h=0} = y_0, \tag{7.22a}$$

$$y_1'|_{h=0} = y'(x_0) = f, \tag{7.22b}$$

$$y_1''|_{h=0} = y''(x_0) = f_x + f_y f. \tag{7.22c}$$

Условие (7.22a), очевидно, выполняется всегда. Дифференцируя вторую формулу из (7.20) по  $h$ , имеем

$$y_1' = \sum_i b_i \kappa_i + h \sum_i b_i \kappa_i', \tag{7.23}$$

$$y_1'' = 2 \sum_i b_i \kappa_i' + h \sum_i b_i \kappa_i''.$$

Во всех суммах здесь и далее суммирование идёт от 1 до  $s$ . Вычислим

$$\begin{aligned}\kappa'_i &= \frac{d}{dh} f(x_0 + c_i h, Y_i) = c_i f'_x() + f'_y() Y'_i = [\text{используем (7.19)}] = \\ &= c_i f'_x() + f'_y() \left( \sum_j a_{ij} \kappa_j + h \sum_j a_{ij} \kappa'_j \right). \quad (7.24)\end{aligned}$$

Здесь  $() = (x_0 + c_i h, Y_i)$ . Используя (7.24) из (7.23) получаем

$$\begin{aligned}y'_1|_{h=0} &= \sum_i b_i \kappa_i|_{h=0} = \sum_i b_i f, \\ y''_1|_{h=0} &= 2 \sum_i b_i \kappa'_i|_{h=0} = 2 \sum_i b_i \left[ c_i f_x + f_y f \sum_j a_{ij} \right].\end{aligned}$$

Сопоставляя эти формулы с (7.22b), (7.22c), получаем следующие условия второго порядка для методов РК:

$$\begin{aligned}\sum_i b_i &= 1, \\ \sum_i b_i c_i &= \frac{1}{2}, \\ \sum_i b_i \sum_j a_{ij} &= \frac{1}{2}.\end{aligned} \quad (7.25)$$

▷<sub>2</sub> 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

Именно явные методы Рунге–Кутты можно назвать классическими, так как именно такого рода формулы строили в своё время Рунге и Кутта.



откуда получаем следующий общий вид таблицы Бутчера:

$$\begin{array}{c|c} 0 & \\ \frac{1}{2}\beta^{-1} & \frac{1}{2}\beta^{-1} \\ \hline & 1 - \beta \quad \beta \end{array} \quad (7.28)$$

Здесь  $b_2 = \beta$  — параметр, как правило лежащий в полуинтервале  $(0, 1]$ .

**Третий порядок.** Приведём без вывода методы порядка 3.

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{3} & \frac{1}{3} & \\ \frac{2}{3} & 0 & \frac{2}{3} \\ \hline & \frac{1}{4} & 0 \quad \frac{3}{4} \end{array} \quad \begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & 0 & 1 & \\ 1 & 0 & 0 & 1 \\ \hline & \frac{1}{6} & \frac{2}{3} & 0 \quad \frac{1}{6} \end{array} \quad (7.29)$$

▷3 Постройте какой-нибудь трёхстадийный явный метод РК третьего порядка.

**Четвёртый порядок.** Наиболее популярный «классический» метод четвёртого порядка определяется таблицей

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array} . \quad (7.30)$$



## 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  $\Phi$ . 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

$$y_{i+1} = \Phi(x_i, y_i, x_i + h), \quad i = 0, 1, 2, \dots$$

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 требуемую точность и, с другой стороны, не приводила к чрезмерным вычислительным затратам.

Поэтому численное интегрирование задачи Коши для ОДУ как правило проводят с переменной, адаптивной<sup>3</sup>, длиной шага:

$$x_{i+1} = x_i + h_{i+1}, \quad i = 0, 1, 2, \dots$$

где длина шага  $h_{i+1}$  зависит от результата вычислений на предыдущем,  $i$ -ом 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 пересчёта, или *правило Рунге*.

---

<sup>3</sup>То есть, с длиной шага, которая автоматически «приспосабливается» к поведению решения.

### Правило Рунге

Правило Рунге для численного решения ОДУ практически полностью аналогично случаю приближённого вычисления определённого интеграла. Рассмотрим первый шаг процесса численного интегрирования методом  $\Phi$  порядка  $p$ . Выберем какую-то величину начального шага  $h$  и сделаем сначала один «большой» шаг длины  $2h$

$$\tilde{y}_2 = \Phi(x_0, y_0, x_0 + 2h),$$

а также два шага длины  $h$ :

$$y_1 = \Phi(x_0, y_0, x_0 + h), \quad y_2 = \Phi(x_0 + h, y_1, x_0 + 2h).$$

Наша цель — сравнить погрешности

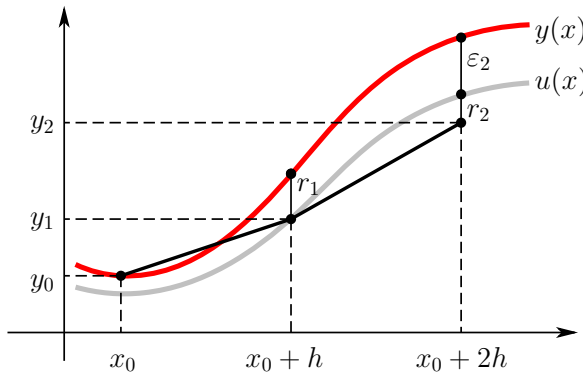
$$\tilde{e}_2 = y(x_0 + 2h) - \tilde{y}_2$$

и

$$e_2 = y(x_0 + 2h) - y_2.$$

По определению локальной погрешности (7.11) имеем

$$\tilde{e}_2 = y(x_0 + 2h) - \tilde{y}_2 = C_0(2h)^{p+1} + O(h^{p+2}). \quad (7.31)$$



Выражение погрешности  $e_2$  имеет более сложный вид (см. рисунок). Оно состоит из двух частей:

$$e_2 = y(x_0 + 2h) - y_2 = r_2 + \varepsilon_2.$$

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

$$r_2 = u(x_0 + 2h) - y_2 = C_1 h^{p+1} + O(h^{p+2}), \quad (7.32)$$

а

$$\varepsilon_2 = y(x_0 + 2h) - u(x_0 + 2h).$$

Можно показать, что для величины  $\varepsilon_2$  справедлива оценка

$$\varepsilon_2 = (1 + O(h))r_1,$$

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 (7.32) имеет вид

$$C_1 = C_0 + O(h).$$

Таким образом,

$$e_2 = \varepsilon_2 + r_2 = (r_1 + O(h^{p+2})) + ((C_0 + O(h))h^{p+1} + O(h^{p+2})),$$

откуда

$$e_2 = y(x_0 + 2h) - y_2 = 2C_0 h^{p+1} + O(h^{p+2}). \quad (7.33)$$

Вычитая из (7.31) выражение (7.33), получаем

$$y_2 - \tilde{y}_2 = 2C_0(2^p - 1)h^{p+1} + O(h^{p+2}).$$

Это равенство позволяет вычислить оценку главной части погрешности для  $y_2$  (см. (7.33)):

$$y(x_0 + 2h) = y_2 + err + O(h^{p+2}), \quad (7.34)$$

где

$$\boxed{err = \frac{y_2 - \tilde{y}_2}{2^p - 1}}, \quad (7.35)$$

а также приближённое значение константы погрешности

$$C_0 = \frac{err}{2h^{p+1}} + O(h). \quad (7.36)$$

Две последние формулы позволяют нам, во-первых, оценить погрешность  $e_2$ , во-вторых, уточнить приближённое 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 (7.35), по построению характеризует точность приближённого решения  $y_2 \approx y(x_0 + 2h)$ :

$$e_2 = err + O(h^{p+2}).$$

Если  $|err| < tol$ , то данное приближение приемлемо, в противном случае необходимо повторить вычисления с меньшим шагом (см. ниже).

Далее, согласно (7.34), величина

$$\hat{y}_2 = y_2 + err \quad (7.37)$$

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.

▷<sub>1</sub> Пользуясь формулами (7.37), (7.35) постройте общую формулу, по которой из любого одношагового метода  $\Phi$  порядка  $p$  можно получить метод  $\hat{\Phi}$  порядка  $p + 1$ .

▷<sub>2</sub> 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

$$|err| < tol. \quad (7.38)$$

Предположим, что неравенство не выполняется, то есть приближённое решение  $y_2$  не является достаточно точным. Стандартный способ повышения точности в этом случае таков:  $y_2$  «отбрасывается» и вычисления повторяются с меньшим шагом  $h_{new}$ . Выбор величины  $h_{new}$  упрощается тем, что нам известна оценка константы погрешности (7.36). Согласно (7.33) мы должны выбрать  $h_{new}$  из условия

$$|2C_0 h_{new}^{p+1}| < tol,$$

откуда сразу же получаем

$$h_{new} < \delta h, \quad (7.39)$$

где

$$\delta = \left( \frac{tol}{|err|} \right)^{\frac{1}{p+1}}. \quad (7.40)$$

Если же неравенство (7.38) выполняется, мы принимаем приближение  $y_2$  (или его уточнённое значение  $\hat{y}_2$ ) и продолжаем процесс численного интегрирования из точки  $x_0 + 2h$  с новым значением шага  $h_{new}$ . 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 выполнялась оценка (7.38), мы совершенно аналогично предыдущему случаю должны выбрать  $h_{new}$  из условий (7.39), (7.40). Только теперь величина  $\delta$  будет больше единицы, то есть 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  $y_2 \leftarrow \Phi(x + h, \Phi(x, y, x + h), x + 2h)$ ,  
 $\hat{y}_2 \leftarrow \Phi(x, y, x + 2h)$ .
- 4) Находим оценку погрешности  $err$  (7.35) и коэффициент  $\delta$  (7.40).
- 5) Вычисляем  $h_{new} \leftarrow \alpha \delta h$ , где  $\alpha < 1$  — «страховочный множитель». Как правило  $\alpha$  выбирают в пределах от 0.7 до 0.9.
- 6) Если  $\delta < 1$ , то полагаем  $h \leftarrow h_{new}$  и возвращаемся к пункту 3.
- 7) Если же  $\delta \geq 1$ , то принимаем шаг: запоминаем пару значений  $(x + 2h, y_2)$ . Вместо  $y_2$  здесь можно взять  $\hat{y}_2$ .
- 8) Полагаем  $x \leftarrow x + 2h, y \leftarrow y_2 (\hat{y}_2), h \leftarrow \min\{h_{new}, \frac{X-x}{2}\}$ .
- 9) Возвращаемся к пункту 2.

*Замечание 7.5.* Понятно, что в силу погрешностей округления условие окончания алгоритма в пункте 2 следует использовать с осторожностью.

*Замечание 7.6.* В случае системы ОДУ оценка погрешности  $err$ , очевидно, будет векторной величиной. Поэтому в формулах вида (7.40) следует использовать какую-либо векторную норму. Кроме этого, для некоторых задач величина относительной погрешности может быть более информативной, поэтому используют также формулы типа

$$err = \frac{1}{2^p - 1} \|D^{-1}(y_2 - \tilde{y}_2)\|,$$

где  $D = \text{diag}(\hat{y}_2)$ . 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, то есть на величину  $\delta$ :

$$\delta = \min \left( \delta_{\max}, \left( \frac{tol}{|err|} \right)^{\frac{1}{p+1}} \right).$$

### 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  $\Phi$  порядка  $p$  и метода  $\hat{\Phi}$  порядка  $q > p$ . Обозначим

$$y_1 = \Phi(x_0, y_0, x_0 + h), \quad \hat{y}_1 = \hat{\Phi}(x_0, y_0, x_0 + h).$$

По определению имеем

$$\begin{aligned} y(x_0 + h) - y_1 &= Ch^{p+1} + O(h^{p+2}), \\ y(x_0 + h) - \hat{y}_1 &= \hat{C}h^{q+1} + O(h^{q+2}). \end{aligned}$$

Отсюда с учётом того, что  $q > p$ , получаем

$$\hat{y}_1 - y_1 = Ch^{p+1} + O(h^{p+2}).$$

Таким образом, величина

$$\boxed{err = \hat{y}_1 - y_1} \tag{7.41}$$

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 (7.39), (7.40), а также общая схема алгоритма адаптивного выбора шага остаётся без изменений. Единственное отличие заключается в способе оценки погрешности: вместо (7.35) имеем (7.41).

В заключение раздела приведём один из наиболее успешных вложенных методов Рунге—Кутты — метод Дормана—Принса порядка 5(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	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$	

(7.42)

Удобство таких методов состоит в том, что для получения оценки погрешности требуется минимальное количество дополнительных вычислений: первая строчка коэффициентов  $\hat{b}_i$  в этой таблице 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, то есть

$$err = h \sum_i (\hat{b}_i - b_i) \kappa_i.$$

## 7.4. Многошаговые методы

### 7.4.1. Введение

Как и ранее, мы рассматриваем 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

$$F(0) = f(x_1, y_1) = f_1, \quad F(-1) = f(x_0, y_0) = f_0,$$

поэтому мы можем 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 получаем формулу

$$y_2 = y_1 + \frac{h}{2}(3f_1 - f_0). \quad (7.43)$$



Это — *двухшаговый метод*, а точнее двухшаговый явный метод Адамса. Напомним, что здесь использовано обозначение

$$f_i = f(x_i, y_i),$$

которое будет активно использоваться в дальнейшем.

**Определение 7.6.** Многошаговым ( $k$ -шаговым) методом решения задачи Коши называется отображение

$$\Phi : \left\{ \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}, \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}, \dots, \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix}, x_k \right\} \mapsto y_k \approx y(x_k).$$

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  $y_1, \dots, y_{k-1}$ . Обычно для этого используются одношаговые методы. После этого уже возможно пошаговое применение метода  $\Phi$ :

$$y_{n+1} = \Phi \left( \begin{bmatrix} x_{n-k+1} \\ y_{n-k+1} \end{bmatrix}, \dots, \begin{bmatrix} x_n \\ y_n \end{bmatrix}, x_{n+1} \right), \quad n = k-1, k, \dots \quad (7.44)$$

### 7.4.2. Явные методы Адамса

Нетрудно обобщить идею, которая использовалась при построении метода (7.43), на случай большего количества точек. Пусть известны значения  $y_0, y_1, \dots, y_{k-1}$ ,

$$y_i \approx y(x_i) = y(x_0 + ih).$$

Тогда имеем

$$y(x_k) = y(x_{k-1}) + h \int_0^1 F(t) dt, \quad (7.45)$$

$$F(t) = f(x_{k-1} + th, y(x_{k-1} + th)).$$

Функция  $F$ , очевидно, обладает свойством

$$F(0) = f_{k-1}, \quad F(-1) = f_{k-2}, \quad \dots, \quad F(1-k) = f_0,$$

или просто  $F(t_j) = f_j$ , где

$$t_j = k - 1 + j, \quad j = \overline{0, k-1}. \quad (7.46)$$

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 (7.45), получаем многошаговый метод вида

$$y_k = y_{k-1} + h \sum_{j=0}^{k-1} \beta_j f_j. \quad (7.47)$$

Коэффициенты  $\beta_j$  представляют собой интегралы от базисных функций Лагранжа  $\Lambda_i$ :

$$\beta_j = \int_0^1 \Lambda_j(t) dt = \int_0^1 \prod_{\substack{l=0 \\ l \neq j}}^{k-1} \frac{t - t_l}{t_j - t_l} dt, \quad (7.48)$$

то есть по сути являются коэффициентами специальной интерполяционной квадратурной формулы, которая отличается от классических формул лишь тем, что использует узлы, лежащие за пределами отрезка интегрирования. Поэтому  $\beta_j$  можно найти не только по определению (7.48), 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

$$\sum_{j=0}^{k-1} \beta_j (1 - k + j)^m = \frac{1}{m+1}, \quad m = 0, 1, \dots, k-1. \quad (7.49)$$

▷<sub>1</sub> Постройте явные методы Адамса для  $k = 3$  и  $k = 4$ .

Необходимо также отметить, что строить формулы Адамса можно используя для представления интерполяционного многочлена  $P$  форму Ньютона.

### 7.4.3. Неявные методы Адамса

При интерполяции подынтегральной функции  $F$  в формуле (7.45) можно использовать не только известные значения  $F$  в точках  $t = 0, -1, \dots, 1 - k$ , но и неизвестное, «неявное» значение в точке  $t = 1$ , то есть

$$F(1) = f(x_k, y_k).$$

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

$$y_k = y_{k-1} + h \sum_{j=0}^k \hat{\beta}_j f_j. \quad (7.50)$$

Коэффициенты  $\hat{\beta}_j$  вычисляются по формуле, аналогичной (7.48):

$$\hat{\beta}_j = \int_0^1 \hat{\Lambda}_j(t) dt = \int_0^1 \prod_{\substack{l=0 \\ l \neq j}}^k \frac{t - t_l}{t_j - t_l} dt. \quad (7.51)$$

Они могут быть также найдены как решение системы

$$\sum_{j=0}^k \hat{\beta}_j (1 - k + j)^m = \frac{1}{m + 1}, \quad m = 0, 1, \dots, k. \quad (7.52)$$

▷<sub>2</sub> 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  $y_0, y_1, \dots, y_{k-1}$  заданы точно, то есть

$$y_j = y(x_j) \quad \forall j = \overline{0, k-1},$$

а  $y_k$  вычислено с помощью метода. Наша задача — оценить величину погрешности

$$y(x_k) - y_k.$$

По построению имеем

$$y(x_k) = y(x_{k-1}) + h \int_0^1 F(t) dt,$$

$$y_k = y_{k-1} + h \int_0^1 P(t) dt,$$

где

$$F(t) = f(x_{k-1} + th, y(x_{k-1} + th)),$$

а многочлен  $P$  интерполирует  $F$  по узлам  $\{t_j\}_{j=0}^{k-1}$  (7.46). Отсюда получаем

$$y(x_k) - y_k = h \int_0^1 r(t) dt,$$

где  $r(t) = F(t) - P(t)$  — остаток полиномиальной интерполяции, который согласно формуле (5.24) 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  $\xi$  — некоторая точка из интервала  $(1-k, 0)$ , зависящая от  $t$ . В силу знакопостоянства многочлена  $\omega_k$  на отрезке  $[0, 1]$  по уже неоднократно применявшейся нами теореме о среднем имеем

$$\int_0^1 r(t) dt = \frac{1}{k!} \frac{d^k}{dt^k} F(\eta) \int_0^1 t(t+1) \dots (t+k-1) dt = C \frac{d^k}{dt^k} F(\eta).$$

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 получаем

$$y(x_k) - y_k = hC \frac{d^k}{dt^k} F(\eta) = O(h^{k+1}),$$

то есть 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 начнём с рассмотрения примера. В плоскости  $xOy$  рассмотрим полёт снаряда, выпущенного из точки  $(0, 0)$  с начальной абсолютной скоростью  $v_0$  под углом  $\alpha$  к оси  $Ox$ . Траектория полёта описывается системой ОДУ

$$\begin{cases} y'(x) = \operatorname{tg} \theta(x), \\ v'(x) = -g \frac{\operatorname{tg} \theta(x)}{v(x)} - k \frac{v(x)}{\cos \theta(x)}, \\ \theta'(x) = -\frac{g}{v(x)^2}. \end{cases} \quad (7.53)$$

с начальными условиями

$$y(0) = 0, \quad v(0) = v_0, \quad \theta(0) = \alpha. \quad (7.54)$$

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 нами ранее численных методов. Однако с практической точки зрения более интересна другая задача: *определить траекторию полёта снаряда, при которой он поражает наземную цель, находящуюся в точке  $x = X$* . Соответствующее решение уравнения (7.53) вместо начальных условий (7.54) должно, очевидно, удовлетворять условиям

$$y(0) = 0, \quad y(X) = 0, \quad v(0) = v_0, \quad (7.55)$$

которые называются *краевыми условиями*. Уравнения (7.53) вместе с условиями (7.55) представляют собой типичный пример *двухточечной краевой задачи*.

**Определение 7.7.** Рассмотрим систему ОДУ

$$y'(x) = f(x, y(x)), \quad x \in [a, b], \quad (7.56a)$$

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 (7.56a):

$$G_i(y(a), y(b)) = 0, \quad i = \overline{1, n}. \quad (7.56b)$$

Уравнения (7.56a), (7.56b) определяют *двухточечную краевую задачу для ОДУ* общего вида.

**Замечание 7.8.** Помимо (вместо) краёв отрезка  $[a, b]$  в условиях (7.56b) 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 (7.56) — 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 (7.53), (7.55). Эту задачу можно переформулировать так: *найти такой угол наклона пушки  $\alpha = \alpha^*$ , при котором решение задачи Коши с условиями (7.54) проходит через точку  $(X, 0)$* . Обозначим  $y(x, \alpha)$  решение системы (7.53) с начальными условиями (7.54), зависящее от параметра  $\alpha$ . Тогда алгоритм нахождения нужного угла  $\alpha^*$  может выглядеть следующим образом:

- 1) Выбираем два значения  $\alpha_1$  и  $\alpha_2$ , для которых заведомо выполняется

$$y(X, \alpha_1) < 0 < y(X, \alpha_2).$$

- 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  $|Y_\alpha| < \varepsilon$ ) — победа!
- 4) Если перелёт ( $Y_\alpha > 0$ ), 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

$$F(\alpha) = y(X, \alpha) = 0.$$

Понятно, что для ускорения сходимости можно применить более совершенные методы решения нелинейных уравнений. Например, метод секущих или Ньютона. Применение последнего, однако, затруднено необходимостью вычисления  $F'(\alpha) = \frac{\partial}{\partial \alpha} y(X, \alpha)$ .

Подведём 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 (7.55) 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  $v(0)$ ) и для этого 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 (7.56a), которую запишем в покомпонентной форме

$$\frac{d}{dx} y^i(x) = f^i(x, y^1(x), \dots, y^n(x)), \quad i = \overline{1, n}, \quad (7.57a)$$



и краевые условия, первые  $m$  из которых заданы в точке  $x = b$ , а остальные — в точке  $x = a$ , то есть являются начальными:

$$\begin{cases} y^i(b) = \beta_i, & i = \overline{1, m}, \\ y^i(a) = \alpha_i, & i = \overline{m+1, n}. \end{cases} \quad (7.57b)$$

Для решения данной краевой задачи определим вектор-функцию

$$F = (F^1, \dots, F^m)^T : \mathbb{R}^m \rightarrow \mathbb{R}^m$$

следующим образом:

$$F^i(\alpha_1, \dots, \alpha_m) = y^i(b, \alpha_1, \dots, \alpha_m) - \beta_i, \quad i = \overline{1, m},$$

где  $y^i(x, \alpha_1, \dots, \alpha_m)$  — решение исходной системы ОДУ (7.57a) с начальными условиями

$$y^i(a) = \alpha_i, \quad i = \overline{1, n} \quad (7.58)$$

(напомним, что начальные значения  $\alpha_{m+1}, \dots, \alpha_n$  заданы по условию). Метод стрельбы решения краевой задачи (7.57) заключается в нахождении вектора  $\alpha^* = (\alpha_1^*, \dots, \alpha_m^*)$ , такого, что

$$F(\alpha^*) = 0.$$

Полученное решение  $y(x, \alpha_1^*, \dots, \alpha_m^*)$  по построению будет являться решением краевой задачи (7.57).

Обратим особое внимание, что каждое вычисление  $F(\alpha)$  требует решения задачи Коши (7.57a), (7.58), что, вообще говоря, весьма трудоёмко. Более того, для реализации метода Ньютона, который считается наиболее эффективным методом решения систем уравнений, необходимо вычислять матрицу Якоби  $\frac{\partial F}{\partial \alpha}$ , которой у нас просто нет. Эту матрицу приходится приближать с помощью конечных разностей, что, во-первых, ненадёжно, и, во-вторых, очень накладно. Поэтому следует помнить, что метод стрельбы для краевых задач большой размерности очень трудоёмок и ненадёжен.

## 7.6. Решение краевых задач конечно-разностным методом

### 7.6.1. Общий нелинейный случай

Рассмотрим нелинейную краевую задачу для ОДУ второго порядка:

$$u''(x) = g(x, u(x), u'(x)), \quad u(a) = A, \quad u(b) = B. \quad (7.59)$$

Здесь  $u : \mathbb{R} \rightarrow \mathbb{R}$ , соответственно  $g : \mathbb{R}^3 \rightarrow \mathbb{R}$ . 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  $h = (b - a)/n$ :

$$x_i = a + ih, \quad i = \overline{0, n}. \quad (7.60)$$

Приближенное решение в этих узлах традиционно обозначают

$$y_i \approx u(x_i).$$

Так как по условию можно сразу положить  $y_0 = A$ ,  $y_n = B$ , для нахождения  $\{y_i\}_{i=1}^{n-1}$  необходимо из исходного дифференциального уравнения (7.59) 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 получаем

$$u'(x_i) \approx P'(x_i) = \frac{u(x_{i+1}) - u(x_i)}{h}. \quad (7.61)$$

Такая аппроксимация называется *правой разностной производной*.

Повторяя описанные действия для узлов интерполяции  $\{x_{i-1}, x_i\}$ , получаем *левую разностную производную*

$$u'(x_i) \approx \frac{u(x_i) - u(x_{i-1}))}{h}. \quad (7.62)$$

Более точное приближение получается при использовании трёх 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, получаем *центральную разностную производную*

$$u'(x_i) \approx \frac{u(x_{i+1}) - u(x_{i-1}))}{2h} \quad (7.63)$$

и *вторую разностную производную*

$$u''(x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1}))}{h^2}. \quad (7.64)$$

▷<sub>1</sub> Выведите эти формулы.

### Разностная схема

Приближая производные в дифференциальном уравнении (7.59) по формулам (7.63) и (7.64), мы получим набор приближенных равенств вида

$$\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1}))}{h^2} \approx g\left(x_i, u(x_i), \frac{u(x_{i+1}) - u(x_{i-1}))}{2h}\right).$$

Здесь индекс  $i$  изменяется от 1 до  $n - 1$ . Теперь заменяем  $u(x_i)$  на их приближенные значения  $y_i$  и окончательно получаем следующую систему нелинейных уравнений, а точнее семейство систем, зависящих от параметра  $n$ :

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} = g\left(x_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}\right), \quad i = \overline{1, n-1}, \quad (7.65a)$$

$$y_0 = A, \quad y_n = B. \quad (7.65b)$$

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 (7.59).

### 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.

$$-u''(x) + q(x)u(x) = f(x), \quad u(a) = A, \quad u(b) = B. \quad (7.66)$$

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 (7.65a) превращаются в систему линейных уравнений

$$-\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + q_i y_i = f_i, \quad (7.67)$$

где  $f_i = f(x_i)$ ,  $q_i = q(x_i)$ . Её можно переписать в виде

$$-y_{i-1} + (2 + q_i h^2) y_i - y_{i+1} = h^2 f_i. \quad (7.68)$$

Обозначая

$$d_i = 2 + q_i h^2,$$

а также учитывая краевые условия  $y_0 = A$ ,  $y_n = B$ , окончательно получаем СЛАУ

$$L_h y_h = f_h, \quad (7.69)$$

где

$$L_h = \frac{1}{h^2} \begin{bmatrix} h^2 & 0 & & & \\ -1 & d_1 & -1 & & \\ & -1 & d_2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & d_{n-2} & -1 \\ & & & & -1 & d_{n-1} & -1 \\ & & & & & 0 & h^2 \end{bmatrix}, \quad (7.70)$$

$y_h = (y_0, y_1, \dots, y_{n-1}, y_n)^T$ ,  $f_h = (f_0, f_1, \dots, f_{n-1}, f_n)^T$ . Нижний индекс  $h$  при записи системы (7.69) 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 (7.66). Будем считать  $q(x) > 0 \forall x \in [a, b]$ , так что имеем

$$d_i = 2 + q_i h^2 > 2 \quad \forall i = \overline{1, n-1}. \quad (7.71)$$

Это свойство гарантирует диагональное преобладание для матрицы  $L_h$ , а значит существование и единственность решения СЛАУ (7.69), а также применимость метода прогонки для ее решения.

Теперь наша задача — доказать, что метод сеток сходится, то есть что

$$\boxed{\max_{0 \leq i \leq n} |u(x_i) - y_i| = \|u_h - y_h\| \xrightarrow{h \rightarrow 0} 0.} \quad (7.72)$$

Здесь и далее  $u_h = (u(x_0), u(x_1), \dots, u(x_{n-1}), u(x_n))^T$  — вектор точных значений решения в узлах сетки,  $\|\cdot\|$  — любая векторная норма.

Рассмотрим невязку, полученную при подстановке  $u_h$  вместо  $y_h$  в систему (7.69):

$$\psi_h = L_h u_h - f_h.$$

Такой вектор принято называть *погрешностью аппроксимации*. Так как по условию  $L_h y_h = f_h$ , имеем

$$L_h (u_h - y_h) = \psi_h,$$

откуда

$$\|u_h - y_h\| \leq \|L_h^{-1}\| \|\psi_h\|.$$

Из последнего соотношения становятся очевидными **достаточные условия сходимости метода**: для того, чтобы выполнялось (7.72), достаточно выполнения следующих двух условий:

- 1)  $\|\psi_h\| \xrightarrow{h \rightarrow 0} 0$  (условие аппроксимации),
- 2)  $\|L_h^{-1}\| \leq M < \infty \forall h \leq h_0$ , где  $M$  не зависит от  $h$  (условие устойчивости).

### Аппроксимация

Докажем выполнение первого условия сходимости. Рассмотрим  $\psi_i$  —  $i$ -ю компоненту вектора  $\psi_h = L_h u_h - f_h$ . Пусть  $i = \overline{1, n-1}$ , тогда

$$\begin{aligned} \psi_i &= -\frac{u_{i-1} + 2u_i - u_{i+1}}{h^2} + q_i u_i - f_i = \\ &= -\frac{u(x_i - h) + 2u(x_i) - u(x_i + h)}{h^2} + q(x_i)u(x_i) - f(x_i) = \\ &= [f(x_i) = -u''(x_i) + q(x_i)u(x_i)] = u''(x_i) - \frac{u(x_i - h) + 2u(x_i) - u(x_i + h)}{h^2}. \end{aligned}$$

Раскладывая в ряд Тейлора величины  $u(x_i \pm h)$ , в итоге получаем отсюда

$$\psi_i = O(h^2), \quad i = \overline{1, n-1}.$$

▷<sub>2</sub> Прodelайте это.

Для  $i = 0$  и  $i = n$  имеем, очевидно,  $\psi_i = 0$ , так что окончательно

$$\|\psi_h\| = \max_i |\psi_i| = O(h^2) \xrightarrow{h \rightarrow 0} 0.$$

В таких случаях говорят, что *метод (схема) аппроксимирует дифференциальное уравнение со вторым порядком*.

### Устойчивость

Для доказательства устойчивости, то есть равномерной ограниченности нормы матрицы  $L_h^{-1}$ , достаточно показать, что если

$$L_h y_h = f_h,$$

то

$$\|y_h\| \leq M \|f_h\|, \quad M < \infty,$$

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 (7.66) сходится.

# ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

- $LU$ -разложение матрицы, 28  
 $a_i$ , 25  
, 86  
, 104
- QR-разложение, 78
- Барицентрическая интерполяционная формула  
— первая форма, 126  
Барицентрическая формула Лагранжа, 127
- Главная часть остатка квадратур, 187  
Главный член локальной погрешности, 199  
Градиент, 99
- Диагональное  
— преобладание, 32  
— — строгое, 32
- Задача  
— корректная, 16  
— некорректная, 16  
Задача интерполяции кривой, 136  
Задача линейной интерполяции, 108
- Интерполяционный многочлен, 109  
— в форме Лагранжа, 111  
— в форме Ньютона, 114  
Итерационный процесс общего вида, 44
- Квадратичная сходимость, 86  
Квадратурная сумма, 165  
Квадратурная формула, 164  
— Ньютона–Котеса, 174  
— Симпсона, 175  
— интерполяционная, 165  
— левых прямоугольников, 163  
— правых прямоугольников, 163  
— средних прямоугольников, 163  
— — составная, 184  
— трапеций, 164  
— — составная, 184  
Константа погрешности одношагового метода, 199  
Кратность собственного значения  
— алгебраическая, 57  
— геометрическая, 57
- Линейная сходимость, 86
- Мантисса, 9  
Матрица  
— диагонализируемая, 58  
— положительно определенная, 38  
— разреженная, 51  
— элементарного вращения, 71  
Матрица Вандермонда, 110  
Машинная арифметика с плавающей точкой, 13  
Машинное число, 13  
Машинный эpsilon, 14  
Метод



- Рунге—Кутты
- — общего вида, 203
- Гаусса, 25
- Гаусса—Зейделя
- — для СЛАУ, 46
- Данилевского, 66
- Рунге—Кутты
- — явный, 206
- Якоби, 45
- градиентного спуска, 99
- квадратного корня, 39
- покоординатного спуска, 100
- прогонки, 31
- простой итерации
- — для СЛАУ, 44
- релаксации, 46
- степенной метод, 59
- Многочлен
  - Бернштейна, 139
  - Лежандра, 180
  - наименее отклоняющийся от нуля, 120
- Направление спуска, 98
- Норма
  - векторная
  - —  $p$ -норма, 18
  - — евклидова, 18
  - — максимум-норма, 18
  - матричная
  - — Фробениуса, 73
  - — индуцированная, 19
  - — подчиненная, 19
  - — спектральная, 20
  - — строчная максимум-норма, 20
- Обратная подстановка, 25
- Одношаговый метод, 196
  - неявный, 197
  - явный, 197
- Оператор
  - простой структуры, 58
- Ортогональное преобразование, 70
- Ортогональные функции, 178
- Остаток интерполирования, 118
- Параметрические узлы, 136
  - естественные, 137
- Плохо обусловленная задача, 17
- Погрешность
  - аппроксимации, 228
  - локальная, 198
  - округления
  - — абсолютная, 13
  - — относительная, 13
- Погрешность интерполирования, 118
- Порядок сходимости, 86
- Правило
  - округления, 11
- Проблема собственных значений, 58
- Прямой ход, 25
- Разностная производная
  - вторая, 226
  - левая, 226
  - правая, 226
  - центральная, 226
- Разностная схема, 227
- Сверхлинейная сходимость, 86
- Следствие
  - Достаточное условие сходимости, 49
  - Критерий сходимости итерационных процессов общего вида, 48
- Собственное значение, 48
- Собственное подпространство, 57
- Собственный вектор, 48
- Составная квадратурная формула, 183
- Спектр матрицы, 48
- Спектральный радиус, 48
- Сплайн
  - бикубический, 158
  - билинейный, 156

- Среднеквадратичное приближение
- дискретное, [144](#)
  - функции, [147](#)
- Стандарт IEEE 754, [14](#)
- Теорема
- $QR$ -алгоритм, упрощённая формулировка, [80](#)
  - К. Ф. Гаусс, [178](#)
  - Лагранжа о среднем значении, [104](#)
  - О среднем, [167](#)
  - Повышение АСТ для симметричных КФ, [172](#)
  - Принцип сжимающих отображений, [102](#)
  - Теорема Ролля, [118](#)
  - разложение Холецкого, [37](#)
  - связь метода Гаусса и  $LU$ -разложения, [28](#)
- Точка интерполяции, [108](#)
- Узел интерполяции, [108](#)
- Условие Липшица, [101](#)
- Форма Фробениуса, [67](#)
- Форма Хессенберга, [78](#)
- Формат
- CSC, [52](#)
  - CSR, [52](#)
  - MSR, [54](#)
  - координатный, [51](#)
- Фундаментальный базис, [109](#)
- двумерной алгебраической интерполяции, [155](#)
- Функция
- билинейная, [156](#)
  - интегрируемая с квадратом, [147](#)
- Характеристический многочлен, [57](#)
- Чебышевские узлы, [122](#)
- Число
- обусловленности
  - — матрицы, [21](#)
  - с плавающей точкой, [9](#)
  - — денормализованное, [12](#)
  - — нормализованное, [9](#)
- Экспонента числа с плавающей точкой, [9](#)
- Элементарная матрица
- второго типа, [65](#)
  - первого типа, [65](#)