С. П. Иглин

Математические расчеты на базе MATLAB

Санкт-Петербург
«БХВ-Петербург»
2005

УДК 681.3.06(075.8)
ББК 32.973.26-018.2я73
И26

Иглин С. П.

И26 Математические расчеты на базе MATLAB. — СПб.: БХВ-Петербург, 2005. — 640 с.: ил.

ISBN 5-94157-290-5

Рассматриваются три раздела математики: вариационное исчисление (задачи с закрепленными и подвижными концами, условный экстремум функционалов, численные методы), математическая статистика (обработка массива данных, сравнение выборок, дисперсионный, регрессионный и корреляционный анализ) и теория графов (задачи о максимальном паросочетании, минимальном вершинном покрытии, минимальном остовном дереве, кратчайшем пути, правильной раскраске). Основная направленность книги — применение MATLAB и его расширений: Symbolic Math Toolbox, PDE Toolbox, Optimization Toolbox, Statistics Toolbox и разработанного автором Graph Theory Toolbox. Для всех рассмотренных задач приведены программы их решения, к каждой главе подобрано по 30 вариантов задач для самостоятельного решения. Компакт-диск содержит примеры программ, рассмотренных в книге, и механизм интерактивной работы с пакетом MATLAB.

Для студентов, аспирантов, преподавателей и научных работников технических, компьютерных и экономических специальностей

УДК 681.3.06(075.8)
ББК 32.973.26-018.2я73

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Григорий Добин*
Редактор	*Татьяна Лапина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 21.01.05.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 51,6.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию, товар № 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-290-5

© Иглин С. П., 2005
© Оформление, издательство "БХВ-Петербург", 2005

Содержание

Введение **11**

Структура и содержание книги 13
Необходимое программное обеспечение 14

ЧАСТЬ I. ВАРИАЦИОННОЕ ИСЧИСЛЕНИЕ **17**

Глава 1. Введение в вариационное исчисление **19**

1.1. Основная задача вариационного исчисления 19
1.2. Классические задачи и принципы вариационного исчисления 21
1.3. Классы функций 28
1.4. Экстремум функционала 39
1.5. Непрерывность и варьируемость функционала 40
1.6. Вариация функционала 45
1.7. Необходимое условие экстремума функционала 48
1.8. Основная лемма вариационного исчисления 48
1.9. Вопросы для самопроверки 52

Глава 2. Элементарная задача вариационного исчисления **54**

2.1. Дифференциальное уравнение Эйлера 54
2.2. Частные случаи уравнений Эйлера 61
2.2.1. Подынтегральная функция F не зависит явно от y' 61
2.2.2. Подынтегральная функция F линейно зависит от y' 62
2.2.3. Подынтегральная функция F не зависит явно от y' 63
2.2.4. Подынтегральная функция F зависит только от y' 65
2.2.5. Подынтегральная функция F не зависит явно от x 66
2.3. Вопросы для самопроверки 69
2.4. Примеры выполнения заданий 70
2.4.1. Задание 1 70
2.4.2. Задание 2 75
2.4.3. Задание 3 78
2.5. Задание 80

Глава 3. Функционалы, зависящие от нескольких функций 81

3.1. Система дифференциальных уравнений Эйлера 81
3.2. Вопросы для самопроверки 84
3.3. Пример выполнения задания 84
3.4. Задание 89

Глава 4. Функционалы, зависящие от производных высших порядков 90

4.1. Дифференциальное уравнение Эйлера — Пуассона 90
4.2. Вопросы для самопроверки 95
4.3. Пример выполнения задания 95
4.4. Задание 99

Глава 5. Функционалы, зависящие от функции нескольких переменных 100

5.1. Дифференциальное уравнение Эйлера — Остроградского 100
5.2. Вопросы для самопроверки 106
5.3. Пример выполнения задания 107
5.4. Задание 120

Глава 6. Вариационная задача в параметрической форме 121

6.1. Когда это нужно? 121
6.2. Переход к параметру в элементарной задаче вариационного исчисления 122
6.3. Вопросы для самопроверки 124

Глава 7. Естественные граничные условия 125

7.1. Элементарная задача вариационного исчисления без граничного условия 125
7.2. Функционал, зависящий от нескольких функций, без граничного условия 129
7.3. Вопросы для самопроверки 131
7.4. Примеры выполнения заданий 131
7.4.1. Задание 1 131
7.4.2. Задание 2 135
7.4.3. Задание 3 140
7.5. Задание 143

Глава 8. Условия трансверсальности 144

8.1. Условия трансверсальности в элементарной задаче вариационного исчисления 144
8.2. Условия трансверсальности для функционала, зависящего от двух функций 150
8.3. Вопросы для самопроверки 154
8.4. Примеры выполнения заданий 155
8.4.1. Задание 1 155
8.4.2. Задание 2 159
8.4.3. Задание 3 166
8.5. Задание 173

Глава 9. Отражение экстремалей 174

9.1. Отражение экстремалей в элементарной задаче вариационного исчисления 174
9.2. Вопросы для самопроверки 178

9.3. Пример выполнения задания ... 178
9.4. Задание ... 184

Глава 10. Преломление экстремалей ... 185

10.1. Преломление экстремалей в элементарной задаче ... 185
10.2. Вопрос для самопроверки ... 189
10.3. Пример выполнения задания ... 189
10.4. Задание ... 195

Глава 11. Экстремали с угловыми точками ... 196

11.1. Откуда берутся угловые точки? ... 196
11.2. Вопросы для самопроверки ... 200

Глава 12. Односторонние вариации ... 201

12.1. Запрет на пребывание экстремали в заданной области ... 201
12.2. Вопрос для самопроверки ... 206
12.3. Пример выполнения задания ... 206
12.4. Задание ... 210

Глава 13. Достаточные условия экстремума ... 211

13.1. Собственное и центральное поле ... 211
13.2. Поле экстремалей ... 214
13.3. Функция Вейерштрасса ... 216
13.4. Достаточные условия Вейерштрасса экстремума функционала ... 219
13.5. Достаточные условия Лежандра экстремума функционала ... 224
13.6. Вопросы для самопроверки ... 225

Глава 14. Условный экстремум функционалов ... 227

14.1. Вариационная задача для функционала с голономными ограничениями ... 227
14.2. Вариационная задача с неголономными ограничениями ... 234
14.3. Изопериметрические задачи ... 235
14.4. Вопросы для самопроверки ... 242
14.5. Примеры выполнения заданий ... 242
14.5.1. Задание 1 ... 242
14.5.2. Задание 2 ... 246
14.5.3. Задание 3 ... 249
14.6. Задание ... 252

Глава 15. Метод начальных параметров ... 253

15.1. Метод стрельбы, начальных параметров, матричной прогонки ... 253
15.2. Вопросы для самопроверки ... 258
15.3. Примеры выполнения заданий ... 258
15.3.1. Задание 1 ... 258
15.3.2. Задание 2 ... 262
15.3.3. Задание 3 ... 267
15.4. Задание ... 271

Глава 16. Метод конечных разностей (МКР) ... 272

16.1. МКР для вариационных задач с обыкновенными дифференциальными уравнениями ... 272
16.2. МКР для вариационной задачи в частных производных: прямоугольная сетка ... 274
16.3. МКР для вариационной задачи в частных производных: треугольная сетка ... 276
16.4. Вопросы для самопроверки ... 280
16.5. Примеры выполнения заданий ... 280
16.5.1. Задание 1 ... 280
16.5.2. Задание 2 ... 283
16.6. Задание ... 288

Глава 17. Метод Ритца ... 289

17.1. Применение метода Ритца к одномерным задачам ... 289
17.2. Метод Ритца в применении к двумерным задачам ... 290
17.3. МКР + метод Ритца $\Rightarrow$ МКЭ ... 292
17.4. Вопросы для самопроверки ... 295
17.5. Примеры выполнения заданий ... 295
17.5.1. Задание 1 ... 295
17.5.2. Задание 2 ... 299
17.6. Задание ... 310

ЧАСТЬ II. МАТЕМАТИЧЕСКАЯ СТАТИСТИКА ... 311

Глава 18. Основы выборочного метода ... 313

18.1. Генеральная совокупность и выборка ... 313
18.2. Оценки и требования к ним ... 317
18.3. Оценка математического ожидания ... 325
18.4. Оценка дисперсии ... 326
18.5. Другие выборочные параметры ... 329
18.6. Методика текущих измерений ... 330
18.7. Вопросы для самопроверки ... 332

Глава 19. Доверительные оценки параметров распределения ... 334

19.1. Квантили ... 334
19.2. Доверительный интервал и доверительная вероятность ... 336
19.3. Абсолютная и практическая достоверность ... 341
19.4. Проверка статистических гипотез ... 342
19.5. Односторонние и двухсторонние критерии ... 344
19.6. Вопросы для самопроверки ... 347

Глава 20. Оценки генеральных параметров распределения ... 348

20.1. Оценка генерального математического ожидания ... 348
20.2. Оценка генеральной дисперсии ... 352
20.3. Оценка других генеральных параметров ... 356
20.4. Выявление промахов ... 358
20.5. Вопросы для самопроверки ... 360

Глава 21. Анализ закона распределения ... 361

21.1. Простейший критерий проверки основной гипотезы ... 361
21.2. Подбор теоретического распределения и его параметров ... 362
21.2.1. Вид теоретического распределения ... 362
21.2.2. Параметры теоретического распределения ... 370
21.3. Критерии согласия ... 374
21.3.1. Критерий согласия Колмогорова ... 375
21.3.2. Критерий согласия Пирсона ... 378
21.4. Вопросы для самопроверки ... 381

Глава 22. Сравнение выборок ... 382

22.1. Сравнение двух дисперсий ... 382
22.2. Сравнение двух средних ... 386
22.3. Сравнение нескольких дисперсий ... 390
22.3.1. Критерий Бартлета ... 390
22.3.2. Критерий Кохрана ... 392
22.4. Сравнение нескольких средних ... 393
22.5. Вопросы для самопроверки ... 397

Глава 23. Дисперсионный анализ ... 399

23.1. 1-факторный дисперсионный анализ ... 402
23.2. 2-факторный дисперсионный анализ ... 405
23.3. Многофакторный дисперсионный анализ ... 410
23.4. Вопросы для самопроверки ... 411

Глава 24. Метод наименьших квадратов ... 412

24.1. МНК и его связь с ПМП ... 412
24.2. Система нормальных уравнений Гаусса ... 415
24.3. Доверительные интервалы для генеральных параметров аппроксимации ... 421
24.4. Аппроксимация степенными полиномами ... 424
24.5. Тригонометрическая аппроксимация ... 427
24.6. Аппроксимация функции нескольких переменных ... 431
24.6.1. Линейная модель для 2-факторного эксперимента ... 432
24.6.2. Полином для 2-факторного эксперимента ... 437
24.7. Нелинейная зависимость от параметров ... 437
24.8. Метод наименьших, но не квадратов ... 440
24.9. Вопросы для самопроверки ... 443

Глава 25. Корреляционный анализ ... 445

25.1. Понятие о корреляции ... 446
25.2. Оценка коэффициента корреляции по данным наблюдений ... 452
25.3. Вопросы для самопроверки ... 455

Глава 26. Генерация вариантов заданий ... 456

26.1. Обработка массива данных ... 456
26.2. Сравнение двух выборок ... 460
26.3. Сравнение нескольких выборок ... 461

26.4. Двухфакторный дисперсионный анализ ... 464
26.5. Аппроксимация степенными полиномами ... 466
26.6. Тригонометрическая аппроксимация ... 467
26.7. Линейная функция двух переменных ... 469
26.8. Кривая насыщения ... 470
26.9. Корреляционный анализ ... 471

Глава 27. Функции пакета Statistics Toolbox ... 474

27.1. Функции распределения ... 474
27.2. Плотности распределения ... 475
27.3. Квантили распределения ... 476
27.4. Генераторы случайных чисел ... 477
27.5. Статистические характеристики ... 477
27.6. Подбор параметров ... 478
27.7. Функции правдоподобия ... 479
27.8. Функции описательной статистики ... 479
27.9. Статистическая графика ... 480
27.10. Статистическое управление процессами ... 481
27.11. Линейные модели ... 481
27.12. Нелинейные модели ... 482
27.13. Планирование эксперимента ... 482
27.14. Кластерный анализ ... 483
27.15. Понижение размерности задач ... 483
27.16. Многомерный анализ данных ... 484
27.17. Анализ на основе дерева возможных решений ... 484
27.18. Проверка статистических гипотез ... 484
27.19. Проверка теоретического распределения ... 485
27.20. Функции непараметрической статистики ... 485
27.21. Демонстрационные примеры ... 485
27.22. Функции ввода-вывода ... 486
27.23. Вспомогательные функции ... 486

ЧАСТЬ III. ТЕОРИЯ ГРАФОВ ... 487

Глава 28. Графы и орграфы ... 489

28.1. Основные определения теории графов ... 490
28.2. Как задать граф ... 499
28.3. Описание процедуры *PlotGraph* ... 500
28.4. Пример обращения ... 509
28.5. Вопросы для самопроверки ... 510

Глава 29. Больше ребер, меньше вершин ... 511

29.1. Максимальное паросочетание ... 511
29.1.1. Основные определения ... 511
29.1.2. Сведение к задаче целочисленного линейного программирования ... 512
29.1.3. Описание процедуры *MaxMatch* ... 514
29.1.4. Пример обращения к процедуре *MaxMatch* ... 517

29.2. Минимальное вершинное покрытие ... 520
29.2.1. Основные определения и постановка задачи ... 520
29.2.2. Сведение к задаче ЦЛП ... 521
29.2.3. Описание процедуры *MinVerCover* ... 522
29.2.4. Пример обращения к процедуре *MinVerCover* ... 525
29.3. Немного о двойственности ... 527
29.4. Вопросы для самопроверки ... 528

Глава 30. Жадные алгоритмы и минимальные остовные деревья ... 530

30.1. Жадность помогает и губит ... 530
30.2. Остовное дерево минимального веса ... 532
30.3. Описание процедуры *MinSpanTree* ... 540
30.4. Пример обращения к процедуре *MinSpanTree* ... 542
30.5. Вопросы для самопроверки ... 545

Глава 31. Базис в пространстве циклов ... 546

31.1. Сколько нужно циклов? ... 546
31.2. Описание процедуры *CicleBasis* ... 550
31.3. Пример обращения к процедуре *CicleBasis* ... 553
31.4. Вопросы для самопроверки ... 558

Глава 32. Правильная раскраска вершин ... 559

32.1. Сколько нужно красок? ... 559
32.2. Правильная раскраска графа — задача ЦЛП ... 561
32.3. Описание процедуры *ColorGraph* ... 562
32.4. Пример обращения к процедуре *ColorGraph* ... 565
32.5. Вопросы для самопроверки ... 566

Глава 33. Кратчайший путь ... 567

33.1. Постановка задачи о кратчайшем пути ... 567
33.2. Алгоритм Дейкстры ... 568
33.3. Алгоритм Флойда — Уоршелла ... 570
33.4. Описание процедуры *ShortPath* ... 572
33.5. Пример обращения к процедуре *ShortPath* ... 574
33.6. Вопросы для самопроверки ... 575

Глава 34. Разбиваем и упорядочиваем ... 577

34.1. Бинарные отношения ... 577
34.2. Разбиение на классы эквивалентности ... 581
34.3. Алгоритмы упорядочения ... 583
34.4. Описание процедуры *DecompPartOrder* ... 584
34.5. Пример обращения к процедуре *DecompPartOrder* ... 586
34.6. Вопросы для самопроверки ... 590

Глава 35. Максимальный поток в сети ... 591

35.1. Задача о максимальном потоке как задача линейного программирования ... 591
35.2. Описание процедуры *MaxFlows* ... 593

35.3. Пример обращения к процедуре *MaxFlows* ... 595
35.4. Вопросы для самопроверки ... 597

Глава 36. Задача коммивояжера ... 598

36.1. Задача коммивояжера — задача ЦЛП ... 598
36.2. Описание процедуры *TravSale* ... 601
36.3. Пример обращения к процедуре *TravSale* ... 604
36.4. Вопросы для самопроверки ... 604

ПРИЛОЖЕНИЯ ... 607

Приложение 1. Учимся работать в MATLAB ... 609

П1.1. Символические вычисления ... 610
П1.2. Построение графиков ... 612
П1.3. Решение конечных уравнений ... 616
П1.4. Решение дифференциальных уравнений ... 619
П1.5. Вопросы для самопроверки ... 623

Приложение 2. Описание компакт-диска ... 625

Список литературы ... 627

Предметный указатель ... 631

Введение

Одно из направлений развития вычислительных технологий в настоящее время — это появление мощных математических пакетов, позволяющих максимально упростить процесс подготовки задачи, ее решения и анализа результатов. При использовании таких средств, как Maple, Mathcad, Mathematica или MATLAB, решение дифференциального или трансцендентного уравнения, аналитическое либо численное дифференцирование и интегрирование, обращение матрицы, решение проблемы собственных значений, вычисление предела, разложение в ряд и многие другие задачи решаются с помощью одной команды. Но, конечно, эту команду нужно правильно применить: надо корректно сформулировать задачу; знать, в каком виде искать решение и т. д. Иными словами, применение математических пакетов позволяет ускорить и упростить выполнение рутинных действий, выкладок и избавить от появления досадных ошибок, но:

✔ Математические пакеты не избавляют от необходимости думать!

Среди нескольких десятков математических пакетов (четыре из них перечислены ранее) особого внимания заслуживает система инженерных и научных расчетов MATLAB [15, 16, 36]. В отличие от других пакетов, MATLAB в одинаковой мере ориентирован на применение как символических, так и численных методов. Включенное в MATLAB ядро Maple хорошо справляется с символическими вычислениями [58], а сам MATLAB и его многочисленные инструментарии, описанные, например, в [16, 53, 54, 57], прекрасно работают с числами.

Интересная особенность, реализованная разработчиками MATLAB, — это его способность встраиваться в другие приложения Windows и динамически обмениваться с ними данными. Так, автором этой книги предложен механизм интеграции системы MATLAB в Web-страницу [20], который позволяет пуб-

ликовать в сети Интернет интерактивные учебники с прямым доступом к MATLAB. В частности, именно по такой технологии создана и размещена в Интернете электронная версия данной книги, доступная по адресу [21]. На ней читатель может найти все программы из этой книги, запустить их на счет прямо со страницы и получить нужные ему результаты. Переработанная версия этого сайта размещена на диске, прилагаемом к книге.

Немаловажным фактором широкого распространения этого продукта является также ценовая политика компании MathWorks Inc. (производителя MATLAB), которая позволяет вузам приобретать MATLAB со значительными скидками. Это делает систему MATLAB, на наш взгляд, наиболее подходящей средой для обучения студентов методам решения математических задач.

Всеобщая компьютеризация коснулась и сферы образования. Сегодня уже немыслимо представить изучение математики и различных ее спецкурсов без проведения лабораторных работ в компьютерном классе. Внедрение вычислительной техники в учебный процесс поставило на повестку дня задачу создания учебников по различным дисциплинам, ориентированных на применение компьютеров и, в частности, на использование математических пакетов. Например, широко распространенные учебники по вариационному исчислению [3, 7, 14, 28, 30, 41, 45, 48] были написаны в 70-е годы прошлого столетия или даже раньше. Несомненно, они остаются прекрасными в научном и методическом плане книгами. Но их авторы не предвидели и не могли предвидеть столь бурной компьютеризации. Поэтому необходима адаптация курса вариационного исчисления к использованию современных компьютерных технологий. Первые попытки в этом направлении — [19, 35]. *Часть I* этой книги, посвященная вариационному исчислению, написана на основе [19].

Несколько иная ситуация сложилась в математической статистике. Здесь также в классических учебниках, задачниках и пособиях [6, 8—11, 25, 31, 37, 38, 44, 47] мало затрагиваются проблемы использования вычислительной техники, хотя сама постановка задач математической статистики уже предполагает такое использование, и в каждом математическом пакете есть даже специальный инструментарий для решения задач статистики. В MATLAB это [57]. Многие его функции описаны в справочниках [16, 36] и на странице [42]. Из зарубежных изданий можно отметить [49, 50, 55]. Однако систематического изложения вузовского курса математической статистики с решением задач в среде MATLAB автору найти не удалось. Восполнить этот пробел призвана *часть II* книги, посвященная математической статистике. Ее основа — методическое пособие [23].

И уж совсем парадоксальная ситуация сложилась в теории графов. В классических учебниках [5, 18, 32, 43] реализации алгоритмов на ЭВМ уделяется

значительное внимание. Множество различных программ описано в [2]. Книга [33] посвящена анализу алгоритмов комбинаторной оптимизации, в том числе и теории графов. Однако создатели математических пакетов почему-то обошли эти задачи стороной. Только в Maple есть средства для решения задач теории графов [12]. Некоторые задачи дискретной математики решены с помощью MATLAB в [26]. Но отдельного инструментария MATLAB для решения задач теории графов до недавних пор не было. Лишь в конце 2003 года автором этой книги был разработан и размещен на сайте компании MathWorks Inc. набор функций (инструментарий) для решения задач на графах GrTheory Toolbox. Он доступен для свободного копирования по адресу [51], страница "Mathematics — General" ("Математика — общие вопросы"), и на момент написания этой книги (октябрь 2004 года) входит в десятку лидеров данной категории. Некоторые функции этого пакета описаны в [22]. В *части III* книги различные задачи теории графов решаются с помощью функций GrTheory Toolbox.

Структура и содержание книги

Настоящая книга написана на основе лекций, практических и лабораторных занятий по различным математическим курсам, которые автор на протяжении 20 лет преподает студентам Национального технического университета "Харьковский политехнический институт". Она состоит из введения, трех частей, двух приложений и списка литературы. Во введении, которое вы сейчас читаете, дается краткий обзор содержания книги и список программного обеспечения, которое необходимо для полноценной работы с ней. Каждая из трех частей включает главы, посвященные соответственно вариационному исчислению, математической статистике и теории графов. В *приложении 1* рассмотрены некоторые приемы программирования в среде MATLAB. Если вы раньше не работали с этим пакетом, чтение книги целесообразно начать с него. В *приложении 2* описано содержимое компакт-диска, прилагаемого к книге.

Почти все главы книги построены примерно одинаково. Вначале излагается теория. Здесь читатель найдет готовый конспект лекций по данной теме. Далее (или в конце главы) следуют вопросы для самопроверки. Затем рассматривается решение примеров и составление программ. Решение везде доводится до численного результата, обычно с построением графика или рисунка. *Глава 26* посвящена генерации вариантов индивидуальных домашних заданий (ИДЗ), а для *части I* на диске, приложенном к книге, имеются по 30 вариантов ИДЗ по каждой теме. Содержание диска описано в *приложении 2*.

Для облегчения перекрестных ссылок все формулы, определения, теоремы, рисунки и т. п. имеют двойную нумерацию, включающую номер главы. Конец примера, определения или доказательства отмечен символом □.

Необходимое программное обеспечение

Электронная версия, размещенная на диске, представляет переработанный вариант сайта автора [21], включая механизм взаимосвязи с MATLAB. Для ее чтения понадобится Internet Explorer версии 4.0 или выше. Другие браузеры, в частности Netscape Communicator, Bat! или Opera, не поддерживают в должной мере всевозможные динамические эффекты. Поэтому при написании электронного варианта этой книги был использован Internet Explorer и язык программирования VBScript. Как правило, при установке на компьютер системы Windows автоматически устанавливается и Internet Explorer.

В корневом каталоге диска-приложения находится главный файл книги index.html. Загрузив его, вы попадете в оглавление. Далее следуйте по обычным перекрестным ссылкам. Документ можно загрузить как непосредственно с компакт-диска, так и переписав предварительно все его содержимое на жесткий диск компьютера, сохраняя структуру папок. При этом никакой инсталляции не требуется: весь документ состоит только из Web-страниц, рисунков к ним, таблиц стилей и файлов сценариев.

В электронный вариант книги, размещенный на диске, встроен механизм взаимосвязи с MATLAB, описанный в [20]. Для правильной совместной работы с MATLAB нужно изменить некоторые настройки вашего обозревателя. Выполните следующие действия.

1. Войдите в меню **Сервис** | **Свойства обозревателя**.
2. Откройте страницу **Безопасность**.
3. Выберите зону **Интернет**.
4. Включите уровень безопасности **Другой**.
5. Найдите переключатель **Использование элементов ActiveX, не помеченных как безопасные** (это в конце страницы). Переключите флажок в положение **Предлагать**, как показано на рис. 1.
6. Нажмите кнопку **OK**, чтобы принять изменение, и обновите загрузку страницы.

Теперь ваш обозреватель, загрузив страницу с диска или через Интернет, сможет подключить MATLAB, установленный на вашем локальном компьютере (если он, конечно, есть). Не забудьте при загрузке страницы ответить **Да** на вопрос: **Разрешить ли загрузку ActiveX-элементов, не помеченных как безопасные?**

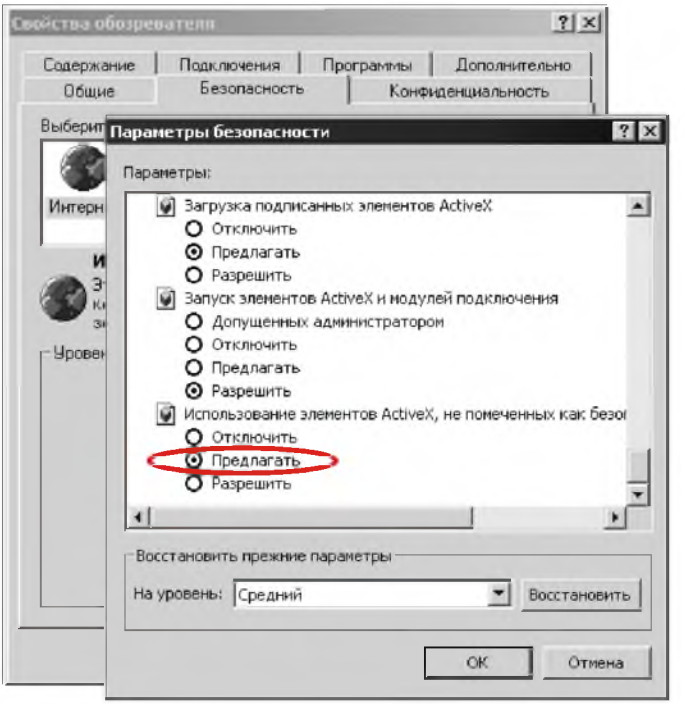


Рис 1. Настройка обозревателя на совместную работу с MATLAB

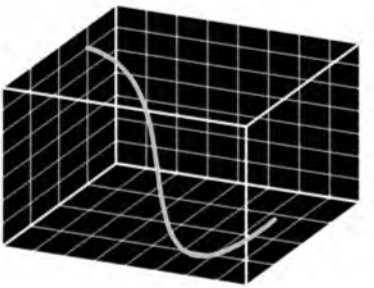
Далее, на вашем компьютере должен быть инсталлирован MATLAB версии 5, 6 или 7. Если у вас его нет, то вы сможете только читать электронную версию книги в интерактивном режиме, но не сможете считать свои варианты задач. Для полноценной работы с книгой установите MATLAB. Полная его версия занимает диск или два и требует для своей установки более 800 Мбайт. Но если отказаться от некоторых инструментариев (в MATLAB они называются Toolboxes), то можно значительно сократить объем дискового пространства. При решении задач из этой книги нам понадобятся следующие инструментарии:

- **Symbolic Toolbox** (инструментарий для символьных вычислений);
- **Partial Differential Equations Toolbox**, или сокращенно **PDE Toolbox** (инструментарий для дифференциальных уравнений в частных производных);
- **Optimization Toolbox** (инструментарий для оптимизации);
- **Statistics Toolbox** (инструментарий для статистики).

Их инсталлируйте обязательно. Для экономии места можно также ограничиться одной из двух справочных служб: или в формате HTML, или в форма-

те PDF. Первая доступна для просмотра через тот же Internet Explorer, а вторая — через Acrobat Reader. Они полностью дублируют друг друга.

К сожалению, в некоторых вариантах 6-й версии MATLAB есть проблемы с кодировкой символов кириллицы. Так, буква `я` (ASCII-код 255, т. е. все единицы в двоичной системе), видимо, используется разработчиками MATLAB в своих, служебных целях, т. к. она не воспринимается даже в комментариях. Поэтому, если у вас установлена такая версия, замените везде в областях ввода маленькую букву `я` на большую `Я`.



ЧАСТЬ I

Вариационное исчисление

Глава 1. Введение в вариационное исчисление

Глава 2. Элементарная задача вариационного исчисления

Глава 3. Функционалы, зависящие от нескольких функций

Глава 4. Функционалы, зависящие от производных высших порядков

Глава 5. Функционалы, зависящие от функции нескольких переменных

Глава 6. Вариационная задача в параметрической форме

Глава 7. Естественные граничные условия

Глава 8. Условия трансверсальности

Глава 9. Отражение экстремалей

Глава 10. Преломление экстремалей

Глава 11. Экстремали с угловыми точками

Глава 12. Односторонние вариации

Глава 13. Достаточные условия экстремума

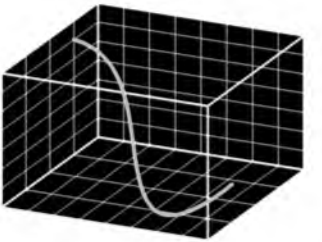
Глава 14. Условный экстремум функционалов

Глава 15. Метод начальных параметров

Глава 16. Метод конечных разностей (МКР)

Глава 17. Метод Ритца

ГЛАВА 1



Введение в вариационное исчисление

1.1. Основная задача вариационного исчисления

При рассмотрении новых объектов их стараются связать с уже знакомыми. И мы при изучении курса вариационного исчисления будем опираться на известные нам определения математического анализа.

До сих пор в обычном курсе анализа вы имели дело с *функциями*. В вариационном исчислении мы встретимся с новым видом математических объектов — *функционалами*. Чем же отличаются функционалы от функций? Давайте вспомним известное вам определение функции.

Определение 1.1. *Функцией* называется любое правило, по которому заданному числу x из некоторого их множества ставится в соответствие число y. □

Мы обозначаем функцию так:

$$y = y(x). \tag{1.1}$$

Множество чисел x, для которых определена функция y, называется *областью определения функции*. Обычно область определения функции — это интервал или система интервалов (открытых, полуоткрытых, закрытых), или вся числовая ось. Хотя, в общем случае, это не обязательно.

Если *числу* α ставится в соответствие *функция*, то α тоже является функцией, но уже зависящей от параметра:

$$y = y(x, \alpha). \tag{1.2}$$

Ее можно рассматривать как функцию двух аргументов.

А вот если *функции* ставится в соответствие *число*, то это будет функционал.

Определение 1.2. *Функционалом* называется любое правило, по которому заданной функции $y(x)$ из некоторого их множества ставится в соответствие число J. □

Обозначим функционал так:

$$J=J(y(x)). \tag{1.3}$$

Множество функций $y(x)$, на которых определен функционал J, назовем *классом функций*.

В (1.3) записан простейший случай зависимости. В общем случае функционал J может зависеть не от одной, а от нескольких функций, причем функции могут быть нескольких переменных. Функционал J может зависеть также от производных этих функций.

Функционал (1.3) можно рассматривать как функцию очень большого (в пределе — бесконечного) числа переменных — значений функции $y(x)$ в различных точках $x\in[a, b]$, как показано на рис. 1.1.

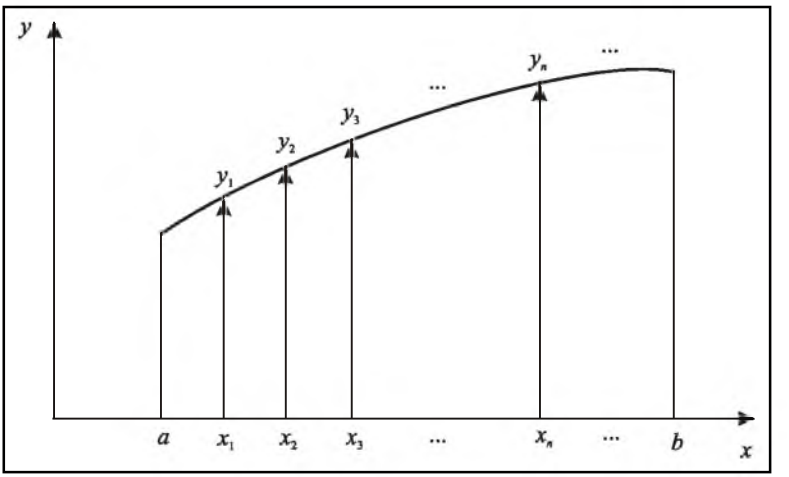


Рис. 1.1. Функционал — это предел функции многих переменных

Вместо $J(y(x))$, как было в (1.3), мы здесь записываем:

$$J=J(y_1, y_2, y_3, ..., y_n, ...). \tag{1.4}$$

Конечно, такая аналогия весьма условна. В обычной функции многих переменных значения, например, аргументов y_2 и y_3 могут сильно отличаться друг от друга (если это допускается областью определения функции). В функционале ситуация принципиально иная. Например, если функция $y(x)$ — непрерывная, то значения соседних аргументов не могут сильно

отличаться. Если $y(x)$ — дифференцируемая, то, к примеру, в точке y_2 не может быть излома, и т. д. Но с другой стороны, эта аналогия полезна, т. к. она используется в численных методах *(см. главу 16)*.

ПРИМЕР 1.1. Длина линии, соединяющей точки $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$, — это функционал:

$$J\big(y(x)\big) = \int_{x_1}^{x_2} \sqrt{1 + y'^2}\, dx. \tag{1.5}$$

Здесь классом функций, на котором определен наш функционал, является множество дифференцируемых на $[x_1, x_2]$ функций, проходящих через точки $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$. Дифференцируемых — потому что нам нужно вычислять производную от функций $y(x)$; проходящих через точки $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$ — потому что так задано в условии задачи. □

Определение 1.3. *Основной задачей вариационного исчисления* является исследование на экстремум функционалов, т. е. нахождение таких функций (из данного класса), которые доставляют функционалу наибольшее или наименьшее значение. Такие функции называются *экстремалями*. □

Так, в примере 1.1 экстремалью является прямая, соединяющая точки $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$: эта функция доставляет минимум функционалу (1.5) на соответствующем классе.

1.2. Классические задачи и принципы вариационного исчисления

Задачи, связанные с исследованием функционалов на экстремум, интересовали человечество давно. Из глубокой древности до нас дошли *четыре классические задачи вариационного исчисления*. Это задача Дидоны, задача о брахистохроне, задача о якорной цепи и задача о геодезических линиях.

Задача Дидоны. По одному из мифов, Дидона (рис. 1.2) была сестрой финикийского царя Пигмалиона, который убил ее мужа, чтобы воспользоваться его богатством. Она покинула Финикию и поселилась в Северной Африке, где царь Иарб обещал дать ей столько земли, сколько покроет воловья шкура. Дидона разрезала шкуру на тонкие ремни и отмерила ими участок, на котором основала город Карфаген (рис. 1.3). А задача, которую решила Дидона, стала классической задачей вариационного исчисления: линией заданной длины L, соединяющей точки $O(0, 0)$ и $M_2(x_2, 0)$, требуется охватить максимальную площадь S, расположенную под линией и выше оси Ox (рис. 1.4). Здесь мы выбрали систему координат так, чтобы ее начало совпало с одним

Рис. 1.2. Дидона

Рис. 1.3. Развалины Карфагена (Тунис)

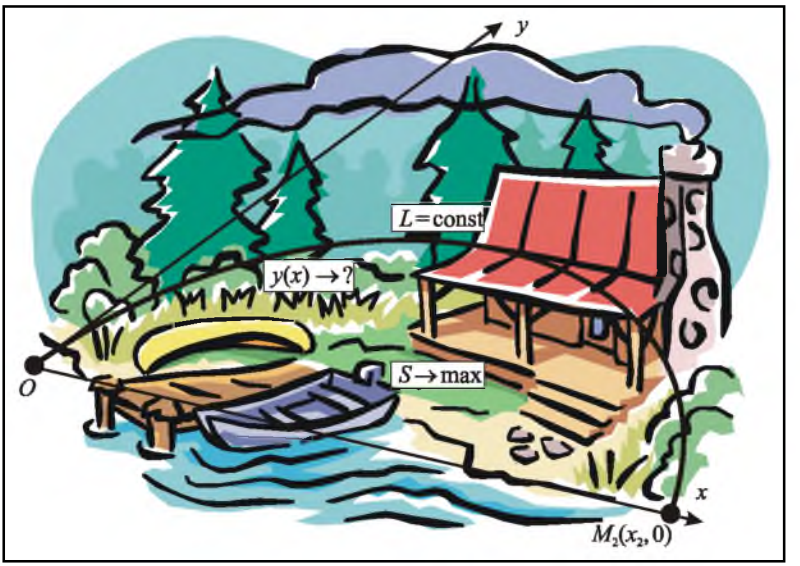


Рис. 1.4. Задача Дидоны

из концов линии, ось Ox направили вдоль береговой линии, а ось Oy — перпендикулярно ей вглубь суши. □

Если правый конец x_2 неподвижен, то решением задачи Дидоны будет *сегмент круга*, а если может скользить вдоль оси Ox — то *полукруг*. Эта задача относится к классу *изопериметрических* (линия имеет постоянный периметр, т. е. длину). Мы будем решать ее в *главе 14*.

Оценим, какую площадь имел Карфаген в момент его основания. Воловья шкура имеет площадь $S_ш \approx 2$ м^2. С помощью примитивных инструментов ее можно разрезать на полоски шириной $h \approx 5$ мм. Длина полученной веревки $l = S_ш/h \approx 400$ м. Этой веревкой можно охватить полукруг радиуса $R = l/\pi \approx 127{,}3$ м. Площадь такого полукруга $S_г = \pi R^2/2 \approx 25\,465$ м^2. Два с половиной гектара — не очень-то и много.

Обобщением этой задачи является *взвешенная задача Дидоны*. Здесь разные участки земли имеют разную стоимость:

$$c = c(x, y), \tag{1.6}$$

и требуется максимизировать не площадь $S(y(x))$, а суммарную стоимость земельного участка:

$$C\big(y(x)\big) = \int_0^{x_2} \left(\int_0^{y(x)} c(x, y)\, dy \right) dx \to \max. \tag{1.7}$$

Решение этой задачи — уже не обязательно дуга окружности. Если, например, прибрежные участки более ценные, чем расположенные вдали от берега, то граница города растянется вдоль береговой линии, как показано на рис. 1.5.

Рис. 1.5. Взвешенная задача Дидоны

Задача о брахистохроне. *Брахистохрона* (от греч. βραχιστοσ — кратчайший и χρονοσ — время) в узком смысле слова — это кривая наискорейшего спуска, т. е. та из всевозможных кривых, соединяющих две точки O и M_2, вдоль которой материальная точка, катящаяся без трения из точки O под действием силы тяжести, в кратчайшее время достигнет точки M_2 (рис. 1.6).

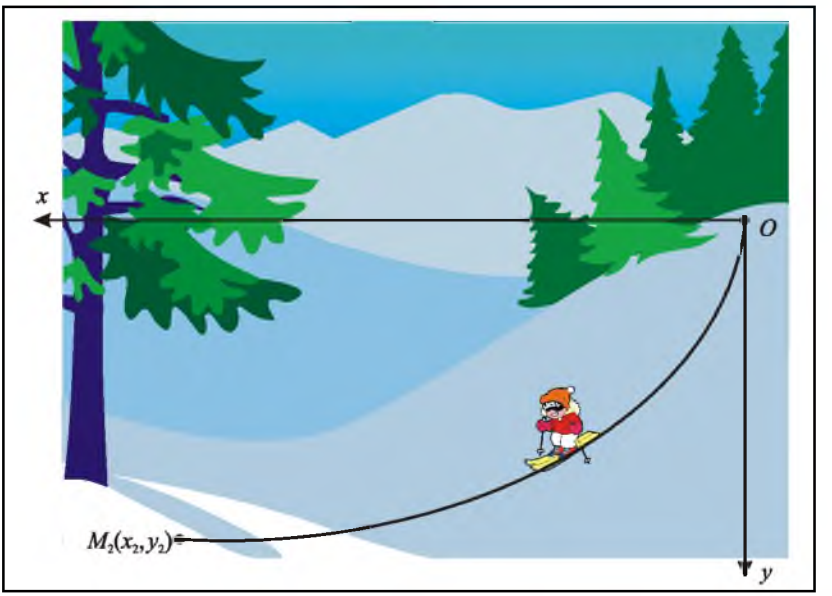


Рис. 1.6. Задача о брахистохроне

Начало системы координат на рисунке мы поместили в точку старта, ось Ox направили горизонтально вдоль траектории движения (она считается плоской), а ось Oy — вертикально вниз. □

Если вы заранее не знаете, то вряд ли догадаетесь, что решением этой задачи является *циклоида*. Да, да, та самая циклоида — след точки на ободе колеса, которое катится без трения по оси Ox. Мы решим эту задачу в *главе 2*.

Брахистохрона в широком смысле слова — это также линия наискорейшего прохождения дистанции, но уже не обязательно в поле силы тяжести. Например, согласно *принципу Ферма* (Pierre Férmat, 1601—1665, рис. 1.7), свет, распространяясь в среде с переменной оптической плотностью (показателем преломления) $n(x, y)$, движется именно по брахистохроне. Он стремится прийти из точки $M_1(x_1, y_1)$ в точку $M_2(x_2, y_2)$ за минимально возможное время. И линией, по которой он распространяется, уже будет не обязательно прямая (рис. 1.8). Чтобы минимизировать время прохождения дистанции, большую часть своего пути свет стремится пройти в среде с меньшей оптической плотностью (его скорость там выше). □

Рис. 1.7. П. Ферма

Рис. 1.8. Распространение света (принцип Ферма)

Задача о якорной цепи. Найти форму провисания якорной цепи. Или, что то же самое, найти форму провисания тонкой абсолютно гибкой нерастяжимой нити, соединяющей точки M_1 и M_2 (рис. 1.9). □

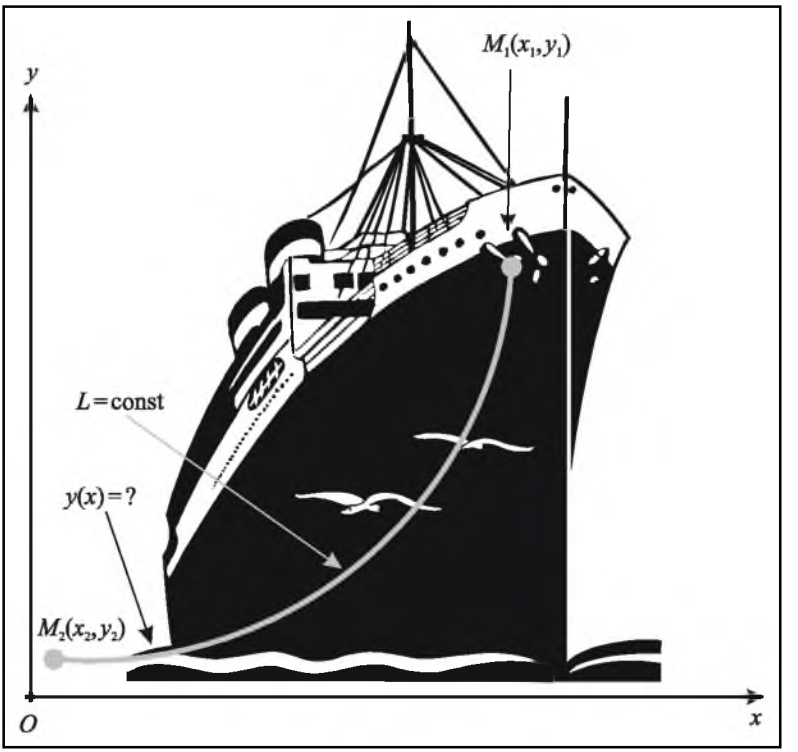


Рис. 1.9. Задача о якорной цепи

Решением этой задачи является гиперболический косинус, который иногда так и называют: цепная линия.

Как и задача Дидоны, эта задача является изопериметрической, и мы решим ее в *главе 14*.

Конечно, у этой задачи также есть обобщения: линия может быть переменной плотности, иметь конечную гибкость, быть растяжимой и т. д.

Задача о геодезической линии. На заданной поверхности требуется найти линию минимальной длины, которая соединяет две заданные точки поверхности M_1 и M_2 (рис. 1.10). □

Решение этой задачи зависит от вида поверхности. В *главе 14* мы рассмотрим пример решения для одной конкретной поверхности.

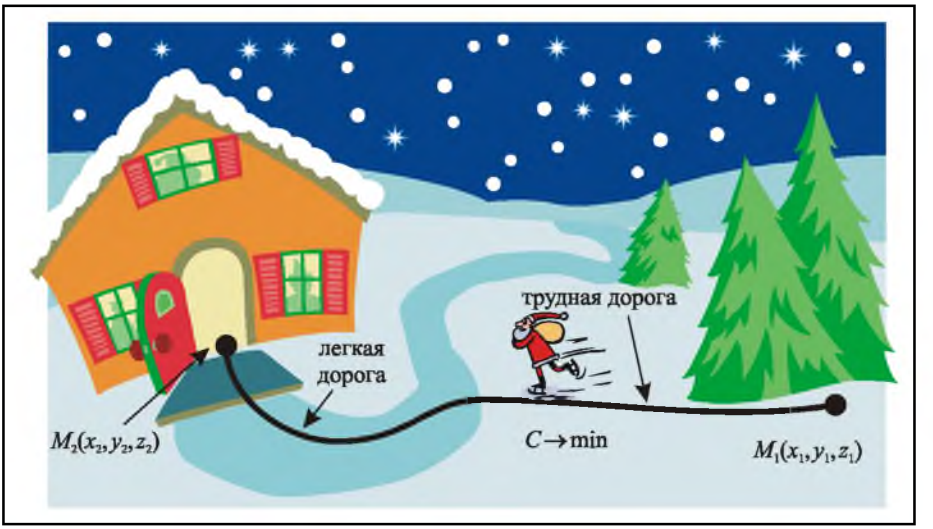


Рис. 1.10. Задача о взвешенной геодезической линии

Обобщение этой задачи — нахождение взвешенной геодезической линии, когда на поверхности распределена некоторая весовая функция c, и требуется минимизировать общий вес пути.

На рис. 1.10 показана как раз такая ситуация: по дорожке нашему Деду Морозу бежать легко, а по снегу — трудно. Поэтому он интуитивно выбирает более длинный, но быстрый путь: сначала как можно скорее стремится выйти на дорожку, а потом движется по ней. Хорошей иллюстрацией к задаче о геодезической линии является пословица "Умный в гору не пойдет, умный гору обойдет".

Если проанализировать эти классические задачи, то видно, что их можно разбить на две группы. В одних задачах мы сами пытаемся улучшить решение: максимизировать площадь, минимизировать путь и т. д. Типичные примеры — это задача Дидоны и задача о геодезической линии. Другие задачи являются следствием некоторых основополагающих принципов (законов природы). Так, распространение света по брахистохроне является следствием принципа Ферма, форма провисания якорной цепи — это следствие известного закона механики: любое тело стремится занять положение устойчивого равновесия, когда его энергия минимальна.

Подобные законы, сформулированные как задачи вариационного исчисления, носят название *вариационных принципов*. С одним из них мы уже познакомились — это принцип Ферма. Другой важный вариационный принцип, который часто применяется в механике, — это *принцип Остроградского — Га-*

Рис. 1.11. М. В. Остроградский

Рис. 1.12. У. Р. Гамильтон

мильтона (Михаил Васильевич Остроградский, 1801—1862, рис. 1.11, и Уильям Роуэн Гамильтон (William Rowan Hamilton), 1805—1865, рис. 1.12).

Принцип Остроградского — Гамильтона. Среди всех возможных (т. е. совместимых со связями) движений системы материальных точек в действительности осуществляется такое, которое доставляет стационарное значение функционалу S. Он называется "действие по Остроградскому — Гамильтону" и вычисляется по формуле:

$$S=\int_{t_1}^{t_2}\left(T-U\right)dt\,, \tag{1.8}$$

где T — кинетическая энергия системы, U — ее потенциальная энергия, t — время, $[t_1, t_2]$ — интервал времени, на котором рассматривается движение системы. □

Таким образом, законы движения механических систем могут быть получены как следствие вариационного принципа Остроградского — Гамильтона. В следующих главах мы научимся выводить эти законы.

1.3. Классы функций

Давайте вспомним, как мы определяли экстремум функции нескольких переменных

$$y=y(x_1, x_2, ..., x_n)=y(\mathrm{x}). \tag{1.9}$$

Здесь вектор аргументов обозначен x.

Определение 1.4. Точка x_0 называется точкой *минимума* функции $y(x)$, если эту точку x_0 можно окружить некоторой малой δ-окрестностью, в которой $\forall x$ выполняется условие: $y(x) \geq y(x_0)$. Если при этом $\forall x$ из этой δ-окрестности, кроме x_0, будет выполняться $y(x) > y(x_0)$, то говорят, что в точке x_0 достигается *строгий минимум*. Аналогично дается определение *максимума (строгого максимума)*. □

Хотелось бы таким же образом определить экстремум и для функционалов, но что такое δ-окрестность *функции*? Для точки $x \in R_n$ мы под δ-окрестностью точки x_0 понимаем n-мерный круг радиуса δ с центром в точке x_0.

Определение 1.5. δ-*окрестность* точки x_0 — это множество точек x, для которых выполняется условие:

$$\left|\mathbf{x}-\mathbf{x}_0\right| = \sqrt{\sum_{i=1}^{n}\left(x_i - x_{0i}\right)^2} < \delta\,. \ \square \tag{1.10}$$

Здесь x_i — координаты точки x, а x_{0i} — координаты x_0. Для $n=2$ малая δ-окрестность точки x_0 показана на рис. 1.13.

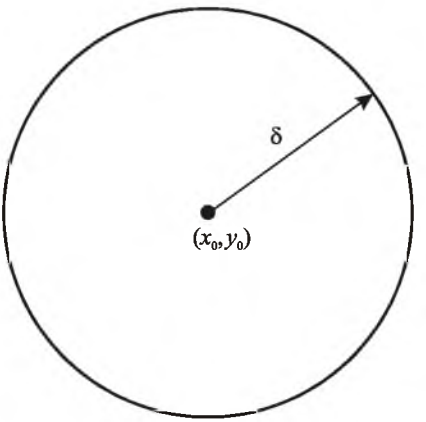


Рис. 1.13. δ-окрестность точки x_0 в смысле (1.10)

Правильнее было бы сказать так: мы вначале определили для точки $x \in R_n$ *норму*, как квадратный корень из суммы квадратов ее координат, потом ввели понятие *расстояния* (норма разности), а затем уже определили δ-окрестность точки x_0 как такое множество точек, для которых их расстояние до x_0 меньше δ. Попробуем пойти по этому же пути для функций. Более подробно эти вопросы рассматриваются в курсе *функционального анализа*. Здесь мы ограничимся только теми понятиями, которые будут нам нужны для решения задач вариационного исчисления.

Функцию $f(x)$ на $[a, b]$ можно рассматривать как ∞-мерный вектор (см. рис. 1.1): координатные оси — это значения $x \in [a, b]$, а сами значения координат — это значения функции $f(x)$. Вместо суммы, участвующей в определении 1.5, мы можем записать, например, интеграл. Тем самым мы определим некоторый *класс функций.*

Определение 1.6. *Классом функций* L_2 на $[a, b]$ называется множество функций, интегрируемых в квадрате на $[a, b]$, для которых норма вычисляется по формуле

$$\| f(x) \|_{L_2} = \sqrt{\int_{x_1}^{x_2} f^2(x)\, dx}\ . \square \qquad (1.11)$$

Если возможны различные толкования, название класса проставляют в виде индекса внизу после определения нормы. Саму норму обозначают не одинарными вертикальными черточками, как модуль, а двойными. В определении 1.6 требуется, чтобы функция была интегрируемой в квадрате на $[a, b]$, т. к. там вычисляется такой интеграл.

Используя определение 1.6, можно ввести понятие δ-окрестности функции $y_0(x)$ в классе L_2. Это такие функции $y(x)$, для которых норма разности их от $y_0(x)$ меньше δ:

$$\|y(x) - y_0(x)\|_{L_2} = \sqrt{\int_{x_1}^{x_2} \left(y(x) - y_0(x)\right)^2 dx} < \delta. \qquad (1.12)$$

ПРИМЕР 1.2. Посмотрите на рис. 1.14. Здесь сплошной жирной линией нарисована исходная функция $y_0(x)$, а еще две функции нарисованы так: сплошной тонкой линией — $y_1(x)$, а штриховой — $y_2(x)$.

Функция $y_2(x)$ отличается от $y_0(x)$ значительно, но только на небольшом интервале. Наоборот, $y_1(x)$ отличается от $y_0(x)$ хоть и не так сильно, но на всем отрезке $[a, b]$. Площадь между $y_1(x)$ и $y_0(x)$ больше площади между $y_2(x)$ и $y_0(x)$. Поэтому, если вычислять по (1.12) расстояние (норму разности) между $y_1(x)$ и $y_0(x)$, с одной стороны, и $y_2(x)$ и $y_0(x)$ с другой, то окажется, что

$$\| y_1(x) - y_0(x) \|_{L_2} > \| y_2(x) - y_0(x) \|_{L_2}\ . \square \qquad (1.13)$$

С классом функций L_2 вы имели дело в теории рядов Фурье. Там, в частности, доказывается, что из всех тригонометрических многочленов степени n наилучшее приближение к функции $y(x)$ в смысле (1.12) имеет такой многочлен, коэффициенты которого совпадают с коэффициентами Фурье функции $y(x)$. На этом далее строится доказательство сходимости рядов Фурье.

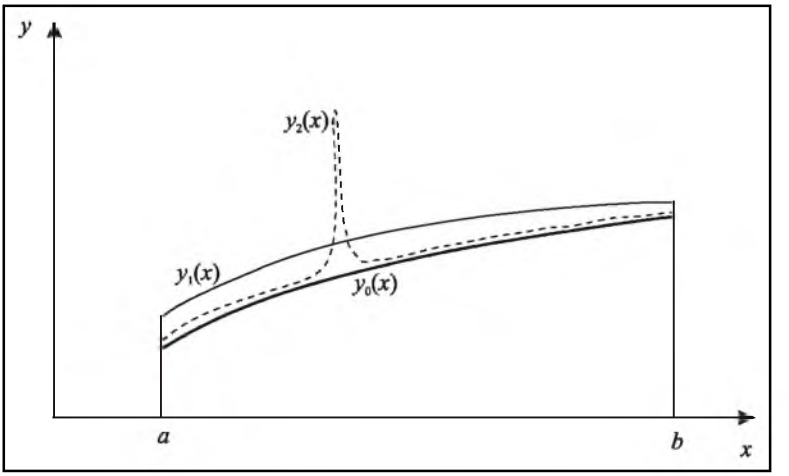


Рис. 1.14. В классе L_2 функция $y_1(x)$ находится дальше от $y_0(x)$, чем $y_2(x)$

В вариационном исчислении класс L_2 применяется редко. Здесь используются другие классы, которые мы сейчас и рассмотрим.

Определение 1.7. *Классом функций* C_0 на $[a, b]$ называется множество функций, непрерывных на $[a, b]$, для которых норма вычисляется по формуле

$$\| f(x) \|_{C_0} = \max_{\forall x \in [x_1, x_2]} | f(x) | . \ \square \tag{1.14}$$

При таком определении нормы две функции $y_0(x)$ и $y(x)$ считаются близкими, если максимум модуля их разности мал. Так, например, δ-окрестность функции $y_0(x)$ в классе C_0 — это такие функции $y(x)$, для которых норма разности их от $y_0(x)$ меньше δ:

$$\| y(x) - y_0(x) \|_{C_0} = \max_{\forall x \in [x_1, x_2]} | y(x) - y_0(x) | < \delta . \tag{1.15}$$

ПРИМЕР 1.3. Посмотрите еще раз на рис. 1.14. Здесь максимальная разность между $y_2(x)$ и $y_0(x)$ больше, чем между $y_1(x)$ и $y_0(x)$. Поэтому, если вычислять расстояние по (1.15), то

$$\| y_1(x) - y_0(x) \|_{C_0} < \| y_2(x) - y_0(x) \|_{C_0} . \tag{1.16}$$

Функция $y_1(x)$ в смысле C_0 находится ближе к $y_0(x)$, чем $y_2(x)$. □

Определение 1.8. *Близостью 0-го порядка* называется близость в смысле C_0. □

В формуле (1.15) $y(x)$ близка к $y_0(x)$ в смысле близости 0-го порядка: норма их разности в C_0 мала.

Определение 1.9. *Классом функций* C_1 на $[a, b]$ называется множество функций, дифференцируемых в $[a, b]$, для которых норма вычисляется:

$$\| f(x) \|_{C_1} = \max\left(\max_{\forall x \in [x_1, x_2]} |f(x)|, \max_{\forall x \in [x_1, x_2]} |f'(x)| \right). \square \qquad (1.17)$$

Близкими в C_1 будут такие функции, которые и сами мало отличаются друг от друга, и производные их отличаются мало.

Определение 1.10. *Близостью 1-го порядка* называется близость в смысле C_1. □

ПРИМЕР 1.4. На рис. 1.15 функции $y_1(x)$ (сплошная тонкая линия) и $y_2(x)$ (штриховая) мало отличаются от $y_0(x)$ (сплошная жирная линия) по модулю. Значит, обе они близки к $y_0(x)$ в смысле близости 0-го порядка. Но функция $y_2(x)$ сильно отличается от $y_0(x)$ по значениям производной, а $y_1(x)$ — нет. Поэтому $y_1(x)$ будет близка к $y_0(x)$ в смысле близости 1-го порядка, а $y_2(x)$ — нет. Получается, что множество функций, близких к данной в смысле близости 1-го порядка, является подмножеством множества функций, близких к данной в смысле близости 0-го порядка. Говорят, что *класс* C_1 *является подклассом класса* C_0. □

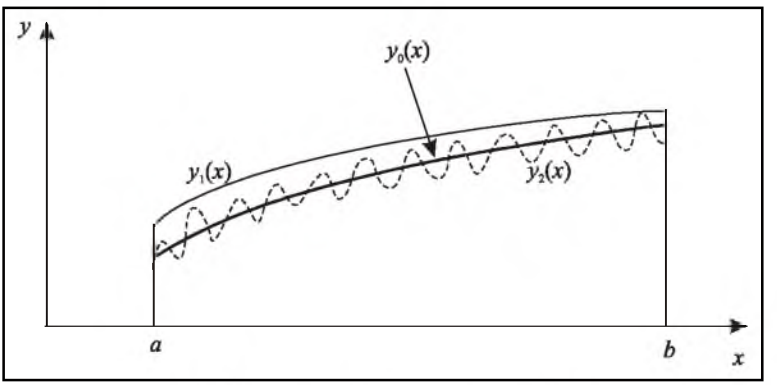


Рис. 1.15. Близость функций в классах C_0 и C_1

Отсюда следуют два важных вывода, которые мы будем далее использовать.

Вывод 1.1. Если какое-либо свойство выполняется для всех функций из C_0, то оно тем более будет выполняться и для всех функций из вложенного в C_0 класса C_1. □

Вывод 1.2. Если какая-либо функция принадлежит C_1, то она тем более будет принадлежать и охватывающему C_1 классу C_0. □

Так, на рис. 1.15 $y_1(x)$ близка к $y_0(x)$ в смысле C_1 (т. е. принадлежит множеству функций, близких к $y_0(x)$ в смысле C_1). Поэтому она будет также близка к $y_0(x)$ и в смысле C_0. Здесь мы применяем вывод 1.2 и говорим:

> ✔ Если две функции близки в смысле близости 1-го порядка, то они тем более будут близки в смысле близости 0-го порядка.

Действительно, пусть две функции мало отличаются одна от другой и по значениям функции, и по значениям производной, т. е. близки в C_1. Тогда они, в частности, мало отличаются по значениям функции, т. е. близки в C_0.

А вот обратная ситуация. Пусть мы показали, что какое-то свойство выполняется для всех функций, близких к $y_0(x)$ в смысле C_0, т. е. и для $y_1(x)$, и для $y_2(x)$ с рис. 1.15. Тогда в силу вывода 1.1 это свойство должно иметь место и для всех функций, близких к $y_0(x)$ в смысле C_1, в частности, для $y_1(x)$.

Иногда норма в C_1 вычисляется не по (1.17), а по формуле:

$$\| f(x) \|_{C_1} = \max_{\forall x \in [x_1, x_2]} |f(x)| + \max_{\forall x \in [x_1, x_2]} |f'(x)|. \qquad (1.18)$$

Несложно убедиться, что функции, близкие в смысле нормы (1.18), будут также близкими и в смысле (1.17), и наоборот. Сформулируем соответствующие теоремы.

■ **ТЕОРЕМА 1.1.** Если функции $y_1(x)$ и $y_2(x)$ близки в классе C_1 в смысле определения (1.17), то они будут близки и в смысле (1.18).

Доказательство. Зададим малое δ, и вычислим $\delta_1 = \delta/2$. Поскольку функции $y_1(x)$ и $y_2(x)$ близки в C_1 в смысле (1.17), то и значения самих функций, и значения их производных могут отличаться лишь на малую величину, например, на δ_1:

$$\begin{cases} \max\limits_{\forall x \in [x_1, x_2]} |y_1(x) - y_2(x)| < \delta_1; \\ \max\limits_{\forall x \in [x_1, x_2]} |y_1'(x) - y_2'(x)| < \delta_1; \end{cases} \qquad (1.19)$$

иначе максимальная из этих величин будет больше δ_1. Но тогда сумма этих малых величин будет равна δ — величине малой. □

■ **ТЕОРЕМА 1.2 (обратная).** Если функции $y_1(x)$ и $y_2(x)$ близки в классе C_1 в смысле определения (1.18), то они будут близки и в смысле (1.17).

Доказательство проведите самостоятельно. □

Определение 1.11. *Классом функций* C_k на $[a, b]$ называется множество функций, k раз дифференцируемых в $[a, b]$, для которых норма вычисляется:

$$\| f(x) \|_{C_k} = \max\left(\max_{\forall x \in [x_1, x_2]} |f(x)|, \max_{\forall x \in [x_1, x_2]} |f'(x)|, \ldots, \max_{\forall x \in [x_1, x_2]} |f^{(k)}(x)| \right). \square \quad (1.20)$$

Иногда вместо (1.20) используют формулу, аналогичную (1.18): берут сумму максимумов модулей функции и ее производных до k-го порядка включительно.

Близкими в C_k будут такие функции, которые и сами мало отличаются друг от друга, и их производные до k-го порядка включительно отличаются мало. Теоремы 1.1 и 1.2 имеют место и здесь. Сформулируйте и докажите их самостоятельно.

Определение 1.12. Близость в смысле C_k называется близостью k-го порядка. □

Видно, что класс C_{k+1} является подклассом класса C_k: в C_{k+1} входят не все функции из C_k, а только та их часть, для которых еще $(k+1)$-е производные отличаются мало. Классы C_0, C_1, ..., C_k вкладываются друг в друга, как матрешки (рис. 1.16). Самая большая матрешка — это класс C_0, следующая по вложенности — C_1, и т. д.

Рис. 1.16. Классы функций: C_0, C_1, ..., C_n, ...

Конечно, и самая большая наша матрешка C_0 может быть, в свою очередь, вложена в еще большую, например, в L_2.

Выводы 1.1 и 1.2, которые мы сделали для C_0 и C_1, можно обобщить.

Вывод 1.3. Если какое-либо свойство выполняется для всех функций из C_k, то оно тем более будет выполняться и для всех функций из вложенного в C_k класса C_{k+1}. □

Вывод 1.4. Если какая-либо функция принадлежит C_k, то она тем более будет принадлежать и охватывающему C_k классу C_{k-1}. □

В частности:

> ✔ Если две функции близки в смысле близости k-го порядка, то они тем более будут близки в смысле близости $(k-1)$-го порядка.

ПРИМЕР 1.5. Найти расстояние в C_0 и C_1 между функциями $y_1(x)=x$ и $y_2(x)=\ln x$ на отрезке $x\in[e^{-1}, e]$.

Для вычисления расстояния в классе C_0 используем формулу (1.15):

$$\| y_1(x)-y_2(x) \|_{C_0} = \max_{\forall x\in[e^{-1},e]} |x-\ln x| . \qquad (1.21)$$

Чтобы вычислить максимальную по модулю разность между двумя функциями, воспользуемся обычными приемами нахождения наибольшего значения функции на отрезке. Найдем все экстремумы функции $y_1(x)-y_2(x)$ в интервале $x\in(e^{-1}, e)$, затем вычислим значения этой функции на краях интервала и выберем наибольшее по модулю значение — это и будет расстояние между $y_1(x)$ и $y_2(x)$ в C_0. Исследуем на экстремум:

$$y_1'(x)-y_2'(x)=1-\frac{1}{x}-\frac{x-1}{x}=0\ ;\ \Rightarrow x=1\ ;\ \ y_1(1)-y_2(1)=1. \qquad (1.22)$$

Значения функции $y_1(x)-y_2(x)$ на краях отрезка $x\in[e^{-1}, e]$:

$$\begin{aligned} &y_1(e^{-1})-y_2(e^{-1})=\frac{1}{e}+1=\frac{1+e}{e}\approx 1{,}36787944117144;\\ &y_1(e)-y_2(e)=e-1\approx 1{,}71828182845905\ . \end{aligned} \qquad (1.23)$$

Из трех вычисленных значений функции (они все положительные) выбираем максимальное — это и есть *расстояние между функциями в* C_0:

$$\| y_1(x)-y_2(x) \|_{C_0} = e-1\approx 1{,}71828182845905. \qquad (1.24)$$

Нарисуем с помощью MATLAB графики наших функций $y_1(x)$ и $y_2(x)$ (см. рис. 1.17). Опишем функции в виде символических выражений, т. к. дальше нам нужно будет построить и графики их производных.

```
clear all % очистили все
syms x % символический аргумент
y1=x; % описали функции
y2=log(x);
xpl=linspace(exp(-1),exp(1)); % массив для графика
```

```
y1pl=subs(y1,x,xpl); % вычислили функции
y2pl=subs(y2,x,xpl);
plot(xpl,y1pl,'-b',xpl,y2pl,'-r') % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title(['\bfГрафики функций '...
  '\rm\ity\rm_1(\itx\rm) и \ity\rm_2(\itx\rm)'])
xlabel('\itx') % метка оси OX
ylabel('\ity\rm_1(\itx\rm), \ity\rm_2(\itx\rm)')
da=daspect; % получили масштаб
da(1:2)=min (da(1:2)); % уравниваем
daspect(da); % установили
xlim([exp(-1) exp(1)]); % пределы по оси OX
```

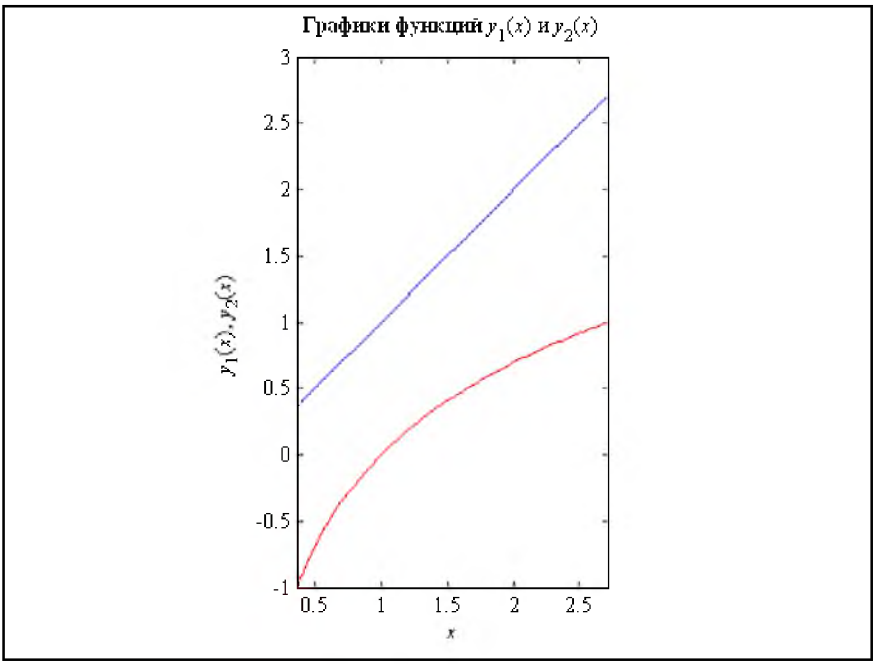


Рис. 1.17. Графики функций $y_1(x)$ и $y_2(x)$, нарисованные с помощью MATLAB

Действительно, на графике видно, что максимальная по модулю разность между функциями $y_1(x)=x$ и $y_2(x)=\ln x$ на отрезке $x\in[e^{-1},e]$ достигается на правом краю, при $x=e$.

Теперь найдем расстояние между нашими функциями в C_1. Для этого используем формулу (1.17) (расстояние в C_1 в смысле (1.18) найдите самостоятельно). Максимальная по модулю разность между значениями функции у нас

есть — это (1.24). Найдем максимальную по модулю разность между производными. Экстремумы внутри интервала (e^{-1}, e):

$$y_1''(x) - y_2''(x) = \frac{1}{x^2} = 0\,; \quad \Rightarrow x = \varnothing\,. \tag{1.25}$$

Значения на концах интервала:

$$\begin{aligned} y_1'\left(e^{-1}\right) - y_2'\left(e^{-1}\right) &= 1 - e \approx -1{,}71828182845905; \\ y_1'(e) - y_2'(e) &= 1 - \frac{1}{e} = \frac{e-1}{e} \approx 0{,}63212055882856\,. \end{aligned} \tag{1.26}$$

Максимальная по модулю разность между производными:

$$\max_{\forall x \in \left[e^{-1}, e\right]} \left| y_1'(x) - y_2'(x) \right| = \left| 1 - e \right| \approx 1{,}71828182845905\,; \tag{1.27}$$

а расстояние между $y_1(x)$ и $y_2(x)$ в C_1 — это максимальная из двух величин: (1.24) и (1.27):

$$\left\| y_1(x) - y_2(x) \right\|_{C_1} = e - 1 \approx 1{,}71828182845905. \tag{1.28}$$

Проверим этот результат с помощью MATLAB: нарисуем графики производных наших функций. Дифференцирование проводим аналитически, с помощью команды `diff`. Далее выполняем те же действия, что и в предыдущей зоне ввода: подставляем значения аргументов, рисуем график, подписываем заголовок и метки осей, выравниваем масштаб и устанавливаем границы по оси *Ox*. Результат показан на рис. 1.18.

```
Dy1=diff(y1,'x') % вычисляем производные
Dy2=diff(y2,'x')
Dy1pl=subs(Dy1,x,xpl); % подставляем аргументы
Dy2pl=subs(Dy2,x,xpl);
figure % новая фигура
plot(xpl,Dy1pl,'-b',xpl,Dy2pl,'-r') % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title(['\bfГрафики функций '...
  '\rm\ity\rm''_1(\itx\rm) и \ity\rm''_2(\itx\rm)'])
xlabel('\itx') % метка оси OX
ylabel('\ity\rm''_1(\itx\rm), \ity\rm''_2(\itx\rm)')
da=daspect; % получили масштаб
da(1:2)=min(da(1:2)); % уравниваем
```

```
daspect(da); % установили
xlim([exp(-1) exp(1)]); % пределы по оси OX
Dy1 =
1
Dy2 =
1/x
```

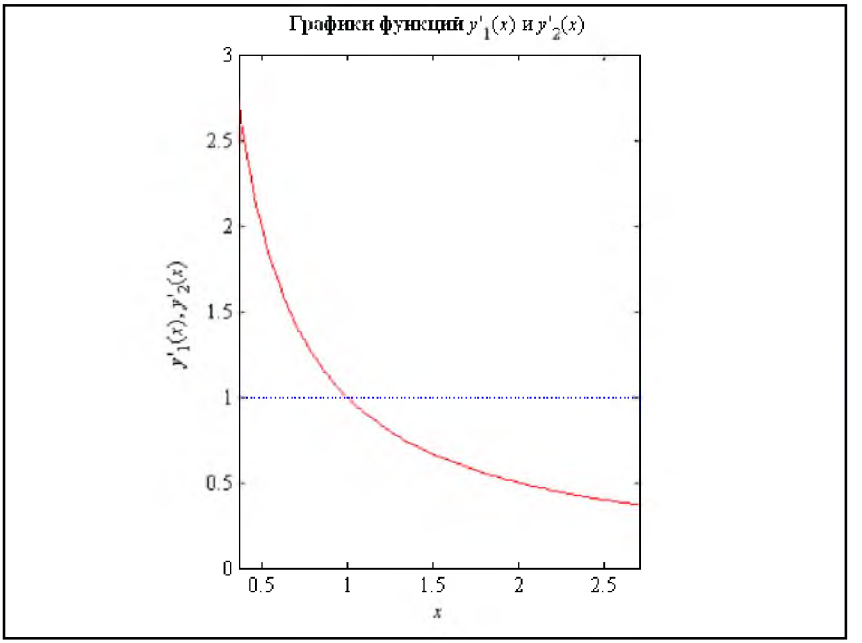


Рис. 1.18. Графики функций $y'_1(x)$ и $y'_2(x)$, нарисованные с помощью MATLAB

Мы видим, что максимальная по модулю разность значений производных достигается на левом краю отрезка, при $x = e^{-1}$. □

ПРИМЕР 1.6. Установить порядок близости кривых $y(x) = \dfrac{\cos nx}{n^2 + 1}$ и $y_0(x) = 0$ на отрезке $x \in [0, 2\pi]$ при $n \to \infty$.

Видно, что $\forall x \in [0, 2\pi]$: $\lim\limits_{n\to\infty} y(x) = y_0(x) = 0$, т. к. $|\cos nx| \le 1$, поэтому функции $y(x)$ и $y_0(x)$ будут близки в C_0: максимум модуля их разности при $n \to \infty$ можно сделать сколь угодно малым.

Оценим близость этих функций в C_1, C_2, ... Производные наших функций: $y'(x) = -\dfrac{n \sin nx}{n^2 + 1}$; $y'_0(x) = 0$, и максимальная разность (по модулю) между ни-

ми также стремится к нулю при $n \to \infty$: $\lim\limits_{n\to\infty} \left| y'(x) - y_0'(x) \right| = \lim\limits_{n\to\infty} \dfrac{n|\sin nx|}{n^2+1} = 0$.

Значит, наши функции близки и в C_1.

Продолжаем исследование. Вторые производные: $y''(x) = -\dfrac{n^2 \cos nx}{n^2+1}$;

$y''_0(x) = 0$. При $n \to \infty$ не существует предела от функции $|y''(x)|$, поэтому в классе C_2 (а тем более в C_3, C_4 и т. д.) наши функции близкими не являются.

Ответ. Функции $y(x) = \dfrac{\cos nx}{n^2+1}$ и $y_0(x) = 0$ на отрезке $x \in [0, 2\pi]$ при $n \to \infty$ *близки в смысле близости 1-го (и 0-го), но не 2-го порядка.* □

1.4. Экстремум функционала

Теперь мы по аналогии с определением 1.4 можем дать определение точки экстремума функционала. Конечно, такой точкой будет функция.

Определение 1.13. Функция $y_0(x)$ называется *точкой минимума* функционала $J(y(x))$, если $\forall y(x)$ из ее малой δ-окрестности выполняется условие: $J(y(x)) \geq J(y_0(x))$. Если при этом $\forall y(x)$ из этой δ-окрестности, кроме $y_0(x)$, будет выполняться $J(y(x)) > J(y_0(x))$, то говорят, что на функции $y_0(x)$ достигается *строгий минимум*. Аналогично дается определение *максимума (строгого максимума)* функционала. □

В этом определении участвует δ-окрестность функции, а мы определяли δ-окрестность для конкретного класса функций. Вот в соответствии с этим и говорят об экстремуме функционала в классах C_0, C_1, ..., C_k, ...

Определение 1.14. Функция $y_0(x)$ называется *точкой экстремума k-го порядка* функционала $J(y(x))$, если соответствующее условие из определения 1.13 выполняется в δ-окрестности функции $y_0(x)$ в C_k. □

Пусть на функции $y_0(x)$ достигается экстремум k-го порядка. Какой вывод мы тогда можем сделать из этого: что достигается экстремум $(k-1)$-го или $(k+1)$-го порядка? Давайте разберемся. Если, например, на $y_0(x)$ достигается минимум k-го порядка, то $\forall y(x)$, которые мало отличаются от $y_0(x)$ по значениям функции, значениям 1-й производной, значениям 2-й производной и т. д., значениям k-й производной, будет выполняться $J(y(x)) \geq J(y_0(x))$. Если при этом $y(x)$ будет мало отличаться от $y_0(x)$ еще и по значениям $(k+1)$-й производной, то хуже от этого не будет: другие условия малого отличия не нарушаются, появляется новое, дополнительное условие. Поэтому по-прежнему будет выполняться $J(y(x)) \geq J(y_0(x))$. Если же, наоборот, значения k-х производных отли-

чаются значительно, то может уже оказаться, что условие $J(y(x)) \geq J(y_0(x))$ не выполняется. Таким образом, мы должны здесь применить вывод 1.3:

> ✔ Если на функции $y_0(x)$ достигается экстремум k-го порядка, то будет достигаться и экстремум $(k+1)$-го порядка.

Здесь можно провести такую аналогию с функцией нескольких переменных (рис. 1.19). Пусть в точке (x_0, y_0) достигается строгий минимум функции двух переменных $f(x, y)$: в любой точке круга малого радиуса δ с центром в точке (x_0, y_0), кроме самой этой точки, будет выполняться условие: $f(x, y) > f(x_0, y_0)$. Но тогда минимум в точке (x_0, y_0) будет достигаться и на любом подмножестве точек этого круга, в котором содержится точка (x_0, y_0)! Например, в интервале $(x_0 - \delta, x_0 + \delta)$ при $y = y_0$.

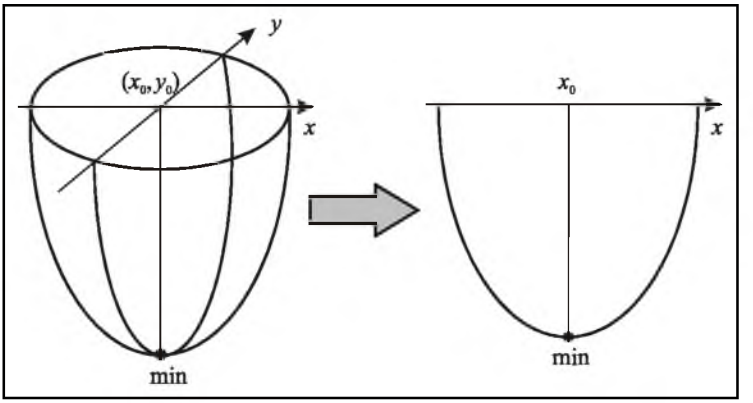


Рис. 1.19. Минимум в области и ее подобласти

Определение 1.15. Экстремум 0-го порядка называется *сильным*. Экстремум 1-го порядка называется *слабым* (или *слабым 1-го порядка*). Экстремум k-го порядка называется *слабым экстремумом k-го порядка*. □

Таким образом, если на какой-то функции достигается сильный экстремум, то достигается и слабый (любого порядка). Если на какой-то функции достигается слабый экстремум меньшего порядка, то достигается и слабый экстремум большего порядка.

1.5. Непрерывность и варьируемость функционала

Как же исследовать функционал на экстремум? Для этого нужны условия: необходимые — чтобы найти функции, на которых может достигаться экс-

тремум, и достаточные, чтобы проверить, достигается ли экстремум. Для вывода этих условий вновь обратимся к функции n переменных (1.9). Мы знаем, что необходимым условием экстремума такой функции является равенство нулю всех частных производных:

$$\begin{cases} y'_{x_i}(\mathbf{x}) = 0; \\ i \in [1, n]; \end{cases} \tag{1.29}$$

или, что то же самое, равенство нулю полного дифференциала:

$$dy(\mathrm{x}) = 0. \tag{1.30}$$

Разумеется, функция $y(\mathrm{x})$ при этом должна быть дифференцируемой: ее приращение Δy в некоторой точке должно быть представимо в виде суммы двух слагаемых: линейного относительно вектора малых приращений $d\mathrm{x}$ и бесконечно малой величины высшего порядка малости относительно $|d\mathrm{x}|$:

$$\Delta y = (\mathrm{C}, d\mathrm{x}) + \beta |d\mathrm{x}|. \tag{1.31}$$

Здесь C — вектор постоянных величин (частные производные в точке), β — бесконечно малая величина.

Для того чтобы мы вообще могли говорить о дифференцируемости, наша функция $y(\mathrm{x})$ обязательно должна быть непрерывной: бесконечно малому приращению аргументов $d\mathrm{x}$ должно соответствовать бесконечно малое приращение функции.

Давайте теперь разбираться с функционалами. Введем понятие, аналогичное понятию дифференциала аргумента, и дадим определение непрерывности функционала.

Определение 1.16. *Вариацией* (или *приращением*) функции $y(x)$ на функции $y_0(x)$ называется:

$$\delta y(x) = y(x) - y_0(x). \ \square \tag{1.32}$$

Определение 1.17. Вариация называется *малой* в смысле малости k-го порядка, если ее норма в классе C_k мала, т. е. не превышает сколь угодно малого заданного наперед числа η:

$$\|\delta y\|_{C_k} < \eta. \ \square \tag{1.33}$$

Для нашего функционала вариация функции — это аналог бесконечно малого приращения аргументов функции нескольких переменных $y(\mathrm{x})$, т. е. дифференциала аргументов $d\mathrm{x}$.

Если вариация δy — малая k-го порядка, то функции $y(x)$ и $y_0(x)$ и сами отличаются мало, и их производные до k-го порядка включительно также отлича-

ются мало. Однако производные более высокого, например, $(k+1)$-го порядка, могут уже отличаться значительно. Согласно выводу 1.4, мы можем утверждать:

> ✔ Если вариация δy — малая k-го порядка, то она будет малой и $(k-1)$-го порядка.

Из (1.32) следует, что операции варьирования и дифференцирования переставимы. Поэтому, когда мы будем писать $\delta y^{(k)}$, то это означает и $\delta(y^{(k)})$, и $(\delta y)^{(k)}$.

Вариация функции δy вызывает приращение функционала J.

Определение 1.18. *Приращением функционала* J на функции $y_0(x)$, соответствующим вариации функции δy, называется:

$$\Delta J(y_0) = J(y_0 + \delta y) - J(y_0). \ \square \tag{1.34}$$

Будет ли приращение функционала ΔJ малым, если вариация функции — малая? Если да, то мы может сказать, что наш функционал — непрерывный. Дадим точное определение непрерывности.

Определение 1.19. Функционал $J(y(x))$ называется *непрерывным* на функции $y_0(x)$ в смысле непрерывности k-го порядка, если малой (в смысле малости k-го порядка) вариации функции δy соответствует малое приращение функционала $J(y)$. Или функционал $J(y(x))$ называется непрерывным на функции $y_0(x)$ в смысле непрерывности k-го порядка, если для любого сколь угодно малого заданного наперед положительного числа ε существует такое малое число η, что, как только мы сделаем норму вариации функции δy (в смысле C_k) меньше η, так сразу модуль приращения функционала станет меньше ε. $\square$

У вас не возникает ощущения, что вы где-то уже это слышали? Конечно, это же обычное определение предела! В нашем случае этот предел выглядит так:

$$\lim_{\|\delta y\| \to 0} \left| \Delta J\left(y \right) \right| = 0 \ . \ \square \tag{1.35}$$

Здесь норма вариации функции $\|\delta y\|$ вычисляется в C_k, поэтому мы и говорим о непрерывности функционала в определенном классе C_k.

Что следует из непрерывности k-го порядка: непрерывность $(k-1)$-го порядка или $(k+1)$-го? Как всегда, давайте анализировать. Пусть, например, наш функционал непрерывен в смысле непрерывности 1-го порядка. Приращение функционала стремится к нулю, когда максимальная из двух величин: $\max|\delta y|$ и $\max|\delta y'|$ стремится к нулю. Если при этом еще и $\max|\delta y''| \to 0$, то хуже от этого не будет: все равно будет $\Delta J \to 0$. А вот если отказаться, например, от

требования $\max|\delta y'| \to 0$, то оставшегося условия $\max|\delta y| \to 0$ может уже оказаться недостаточно для того, чтобы $\Delta J \to 0$. Поэтому:

> ✔ Если функционал J непрерывен в смысле непрерывности k-го порядка, то он будет непрерывен и в смысле непрерывности $(k+1)$-го порядка.

ПРИМЕР 1.7. Исследовать на непрерывность функционал

$$J(y(x)) = y(x_0), \ \square \tag{1.36}$$

где $x_0 \in [a, b]$, а функция $y(x)$ непрерывна в $[a, b]$ и имеет непрерывные производные нужного порядка (рис. 1.20).

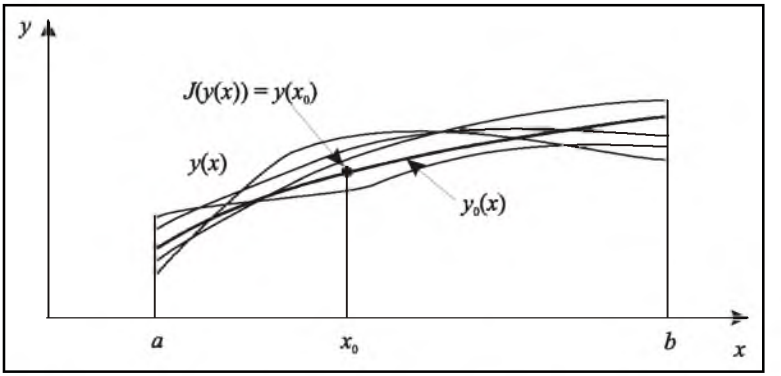


Рис. 1.20. Исследование на непрерывность функционала (1.36)

Сначала исследуем функционал (1.36) на непрерывность в C_0. Дадим малое приращение (вариацию) функции $y_0(x)$ в C_0: $\delta y(x)$. Поскольку $\delta y(x)$ — малая в C_0, то максимальное $\forall x \in [a, b]$ по модулю значение $\delta y(x)$ мало. Следовательно, мало и $\delta y(x_0)$. Приращение функционала в этом случае:

$$\Delta J(y_0(x)) = J(y_0(x) + \delta y(x)) - J(y_0(x)) = y_0(x_0) + \delta y(x_0) - y_0(x_0) = \delta y(x_0), \tag{1.37}$$

а это — величина малая. Таким образом, *функционал (1.36) непрерывен в C_0, а значит, и в C_1, C_2 и т. д.* На рис. 1.20 мы видим: малое изменение ординат функции мало меняет наш функционал, даже если производные меняются сильно. $\square$

ПРИМЕР 1.8. Исследовать на непрерывность функционал

$$J\big(y(x)\big) = \int_0^1 \big| y'(x) \big| \, dx, \tag{1.38}$$

где функция $y(x)$ непрерывна в [0; 1] и имеет непрерывные производные нужного порядка.

Дадим исходной функции $y_0(x)$ малую вариацию $\delta y(x)$. Найдем приращение функционала:

$$\Delta J\left(y_0(x)\right)=\int_0^1\left|\,y_0'(x)+\delta y'(x)\,\right|dx-\int_0^1\left|\,y_0'(x)\,\right|dx\le$$
$$\le\int_0^1\left(\left|\,y_0'(x)\,\right|+\left|\,\delta y'(x)\,\right|\right)dx-\int_0^1\left|\,y_0'(x)\,\right|dx=\int_0^1\left|\,\delta y'(x)\,\right|dx\,. \tag{1.39}$$

Если $\delta y(x)$ — малая в C_1, т. е. и $\delta y(x)$, и $\delta y'(x)$ — малые, то и приращение функционала будет малым:

$$\Delta J\left(y_0(x)\right)\le\int_0^1\left|\,\delta y'(x)\,\right|dx\le\max_{\forall x\in[0;1]}\left|\,\delta y'(x)\,\right|\le\left\|\,\delta y(x)\,\right\|_{C_1}. \tag{1.40}$$

Значит, наш *функционал непрерывен в C_1 (а значит, и в C_2, C_3 и т. д.)*. Пусть теперь $\delta y(x)$ — малая в C_0, но не в C_1, т. е. $\delta y(x)$ — малая, а $\delta y'(x)$ — нет. Покажем, что в этом случае функционал (1.38) разрывен. Возьмем $y_0(x)=0$, а $\delta y(x)=\dfrac{\sin nx}{n}$. Эта вариация — малая в C_0, но не в C_1 при $n\to\infty$ (докажите это — см. пример 1.6). Значение функционала на исходной функции, очевидно, равно нулю. Вычислим приращение функционала:

$$\Delta J\left(y_0(x)\right)=\int_0^1\left|\,\delta y'(x)\,\right|dx=\int_0^1\left|\,\cos nx\,\right|dx\ge\int_0^1\cos^2 nx\,dx=$$
$$=\frac{1}{2}\int_0^1\left(1+\cos 2nx\right)dx=\frac{1}{2}+\frac{1}{2}\int_0^1\cos 2nx\,dx. \tag{1.41}$$

Второе слагаемое (интеграл) по лемме Римана стремится к нулю при $n\to\infty$ (вспомните теорию рядов Фурье!), но первое — это константа. Поэтому получаем, что приращение функционала (1.38) на сколь угодно малой вариации функции превышает 0,5. Значит, *функционал (1.38) разрывен в C_0*. □

С непрерывностью мы разобрались. Теперь введем понятие, аналогичное дифференцируемости. Нам нужно будет выделять из приращения функционала главную, линейную часть. Но записать ее так, как в (1.31) эта часть записана для функции, мы не можем: у нас нет понятия скалярного произведения. Поэтому введем понятие линейного функционала, а затем используем его в определении варьируемости (это аналог дифференцируемости).

Определение 1.20. Функционал $L(y)$ называется *линейным*, если:

$$L(C_1y_1+C_2y_2)=C_1L(y_1)+C_2L(y_2). \square \tag{1.42}$$

ПРИМЕР 1.9 линейного функционала:

$$J(y)=\int_{x_1}^{x_2}\left(p(x)y+q(x)y'\right)dx\,. \square \tag{1.43}$$

Определение 1.21. Функционал $J(y)$ называется *варьируемым* на функции $y(x)$, если его приращение на этой функции ΔJ, соответствующее вариации функции δy, можно представить в виде суммы двух слагаемых:

1. Линейного относительно δy функционала $L(y, \delta y)$ (этот функционал зависит еще и от y — той функции, в окрестности которой вычисляется приращение, но линейность требуется только по второму аргументу δy).
2. Бесконечно малого функционала высшего порядка малости по сравнению с δy (мы его можем представить как произведение просто бесконечно малого относительно δy функционала $\beta(y, \delta y)$ на $\|\delta y\|$):

$$\Delta J(y) = L(y, \delta y) + \beta(y, \delta y)\|\delta y\|. \square \tag{1.44}$$

Здесь $\beta(y, \delta y)$ — бесконечно малый относительно δy функционал:

$$\lim_{\|\delta y\|\to 0}\left|\beta(y,\delta y)\right| = 0\,. \tag{1.45}$$

В определении варьируемости участвует норма вариации функции, поэтому мы должны рассматривать варьируемость в конкретном классе функций, т. е. говорить о варьируемости 0-го, 1-го, ..., k-го порядка. Согласно выводу 1.3, если функционал является варьируемым для всех функций из какого-либо класса, то он будет варьируемым и для всех функций из подкласса этого класса. Поэтому:

> ✔ Если функционал J варьируемый в C_k, то он будет варьируемым и в C_{k+1}.

1.6. Вариация функционала

Для дифференцируемой функции (ее приращение вычисляется по формуле (1.31)) дифференциал dy — это главная, линейная относительно dx, часть приращения функции: 1-е слагаемое формулы (1.31):

$$dy=(C, dx). \tag{1.46}$$

Для варьируемого функционала аналогом дифференциала функции будет вариация.

Определение 1.22. *Вариацией* $\delta J(y)$ функционала $J(y)$ на функции $y(x)$ называется главная, линейная относительно δy, часть его приращения на этой функции ΔJ. □

Согласно (1.44), вариация будет линейным относительно δy функционалом. Конечно, вариация будет также зависеть и от той функции $y(x)$, на которой она вычисляется. Применяя вывод 1.3, мы можем сказать:

✔ Если $\exists\, \delta J(y)$ в C_k, то она существует и в C_{k+1}.

Как же вычислить вариацию функционала? Здесь возможны различные подходы. Мы рассмотрим два способа, которые будем применять дальше при исследовании функционалов на экстремум.

Первый способ вычисления вариации функционала. Даем бесконечно малое приращение δy функции $y(x)$ — аргументу нашего функционала. Вычисляем приращение ΔJ, вызванное этой вариацией. Раскладываем его в ряд Тейлора и удерживаем только линейные члены. □

Этот способ мы будем использовать в следующих главах при выводе дифференциальных уравнений Эйлера. А сейчас рассмотрим второй способ, который нам понадобится при выводе необходимого условия экстремума.

Второй способ вычисления вариации функционала. Пусть $y_0(x)$ — та функция, на которой нам нужно вычислить вариацию функционала δJ. Рассмотрим класс функций вида

$$y(x) = y_0(x) + \alpha\eta(x), \tag{1.47}$$

где α — бесконечно малая величина, а $\eta(x)$ — фиксированная функция. Этот класс является более узким, чем исходный класс функций: в нем все ординаты меняются пропорционально одному числовому параметру α.

На рис. 1.21 жирной сплошной линией показана исходная функция $y_0(x)$, тонкими сплошными линиями — несколько функций из класса (1.47) и тонкой штриховой линией — функция, не принадлежащая классу (1.47).

Мы знаем, что если существует δJ на каком-либо классе функций, то она существует и на более узком классе. Поэтому по способу 2 мы будем вычислять вариацию функционала на классе функций (1.47). Так как в функциях (1.47) $y_0(x)$ и $\eta(x)$ — фиксированные, то функционал на функциях вида (1.47) становится фактически функцией одной переменной α:

$$J(y(x)) = J(y_0(x) + \alpha\eta(x)) = V(\alpha), \tag{1.48}$$

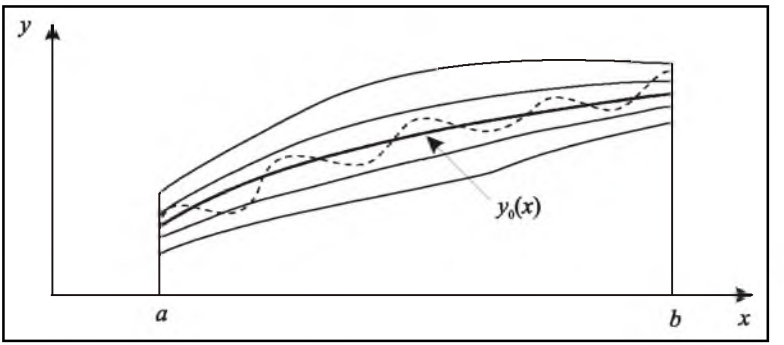


Рис. 1.21. Функции для 2-го способа вычисления δJ

причем $V(0)$ соответствует функционалу, вычисленному на исходной функции $y_0(x)$. Найдем приращение функционала:

$$\Delta J(y_0) = J(y_0 + \alpha\eta) - J(y_0) = V(\alpha) - V(0). \tag{1.49}$$

Так как функционал — варьируемый, то по (1.44) имеем:

$$\Delta J(y_0) = L(y_0, \alpha\eta) + \beta(y_0, \alpha\eta)\,\|\alpha\eta\|, \tag{1.50}$$

или, с учетом линейности (1.42):

$$\Delta J(y_0) = \alpha L(y_0, \eta) + |\alpha|\,\beta(y_0, \alpha\eta)\,\|\eta\|. \tag{1.51}$$

Нам нужно выделить в этом выражении линейный функционал $L(y_0, \eta)$, это и будет интересующая нас вариация функционала. Для этого разделим обе части (1.51) на α и перейдем к пределу при $\alpha \to 0$. В правой части 1-е слагаемое даст интересующую нас величину $L(y_0, \eta)$. Второе слагаемое стремится к нулю. Действительно, первый множитель $|\alpha|/\alpha$ ограничен по модулю, третий множитель $\|\eta\|$ — это норма фиксированной функции — постоянная величина, а вот второй множитель β — это бесконечно малая величина, она стремится к нулю при $\alpha \to 0$. Следовательно, предел правой части — это вариация функционала δJ. Найдем предел левой части. В силу (1.49):

$$\lim_{\alpha\to 0}\frac{\Delta J(y_0)}{\alpha} = \lim_{\alpha\to 0}\frac{V(\alpha) - V(0)}{\alpha - 0} = \frac{dV(\alpha)}{d\alpha}\bigg|_{\alpha=0}. \tag{1.52}$$

Таким образом, вычисление вариации функционала по второму правилу сводится к вычислению производной по α от $J(y_0 + \alpha\eta)$ при $\alpha = 0$:

$$\delta J(y_0) = \frac{d}{d\alpha} J(y_0 + \alpha\eta)\bigg|_{\alpha=0}. \square \tag{1.53}$$

1.7. Необходимое условие экстремума функционала

Как мы знаем, для дифференцируемой функции нескольких переменных необходимым условием экстремума является равенство нулю всех частных производных (1.29) или полного дифференциала (1.30). Для функционалов частные производные не определены, а аналогом дифференциала является вариация.

■ **ТЕОРЕМА 1.3 (необходимое условие экстремума функционала).** Если варьируемый функционал $J(y)$ достигает экстремума на $y_0(x)$ — "внутренней" функции области определения функционала, то вариация функционала на этой функции равна нулю: $\delta J(y_0(x)) = 0$.

Под внутренней функцией мы понимаем такую, которая не лежит на границе области определения, т. е. функцию, для которой вариация в любой точке может быть и положительной, и отрицательной.

Доказательство. Пусть функционал $J(y(x))$ достигает экстремума на $y_0(x)$ — "внутренней" функции области определения функционала. Выберем функцию $\eta(x)$ настолько малой, чтобы при любых малых α функции вида $V(\alpha) = J(y_0(x) + \alpha\eta(x))$ находились в области определения и при положительных, и при отрицательных α. Значению $\alpha = 0$ соответствует $J(y_0(x)) = V(0)$ — экстремальное значение. Необходимое условие экстремума функции одной переменной $V(\alpha)$ — это равенство нулю ее производной: $V'(0) = 0$. Но по (1.52)—(1.53) это соответствует тому, что равна нулю вариация функционала. □

Замечание 1.1. Эта теорема также имеет место, если функционал зависит не от одной, а от нескольких функций. Для доказательства достаточно зафиксировать все функции, кроме одной, на экстремалях (как на рис. 1.19). Тогда функционал будет зависеть только от одной функции и будет достигать экстремума, когда его вариация по этой функции будет равна нулю. □

Замечание 1.2. Эта теорема также имеет место, если функционал зависит от функции нескольких переменных. В этом случае в классе функций (1.47) функции y_0, y и η будут зависеть не от одной, а от нескольких переменных. □

1.8. Основная лемма вариационного исчисления

Мы сейчас сформулируем и докажем одну простую лемму. Она играет настолько важную роль в вариационном исчислении, что так и называется: *ос-*

новная лемма вариационного исчисления. Дальше она будет использоваться при выводе дифференциальных уравнений Эйлера. Вот эта лемма.

■ **ЛЕММА 1.1.** Если $\forall\eta(x)\in C_k$ на $[x_1, x_2]$ интеграл от произведения этой функции на другую функцию $\Phi(x)\in C_k$ по $[x_1, x_2]$ равен нулю:

$$\int_{x_1}^{x_2} \Phi(x)\,\eta(x)\,dx = 0, \tag{1.54}$$

то это возможно только в том случае, если $\Phi(x)\equiv 0\ \forall x\in[x_1, x_2]$.

Доказательство. Проведем доказательство от противного. Пусть в какой-либо точке $x_0\in[x_1, x_2]$: $\Phi(x)\neq 0$. Для определенности будем считать, что $\Phi(x_0) = A > 0$. Вы когда-то изучали свойства функций, непрерывных на интервале, и знаете: если непрерывная функция в какой-то точке x_0 отлична от нуля, то существует некоторая малая окрестность этой точки, в которой функция тоже отлична от нуля и имеет тот же знак, что и в точке x_0. У нас $\Phi(x)\in C_k$, т. е. является непрерывной. Поэтому наверняка существует некоторый отрезок $[x_-, x_+]$, в котором $\Phi(x) > 0$.

Покажем теперь, как можно построить такую функцию $\eta(x)\in C_k$, что (1.54) будет нарушаться. Возьмем $\eta(x)$ в виде:

$$\eta(x) = \begin{cases} k(x-x_-)^{2n}(x-x_+)^{2n}; & x\in[x_-; x_+]; \\ 0; & x\notin[x_-; x_+]. \end{cases} \tag{1.55}$$

За счет показателя n можно добиться дифференцируемости нужное число раз, а за счет k — сделать функцию сколь угодно большой или малой. Нарисуем с помощью MATLAB пример такой функции (рис. 1.22). Посмотрите в *приложении 1*, как строить графики.

```
clear all % очистили все
Xminus=0.4;
Xplus=0.6;
n=2;
k=1;
x=linspace(Xminus,Xplus);
y=k.*(x-Xminus).^(2*n).*(x-Xplus).^(2*n);
plot([0 x 1],[0 y 0]) % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfГрафик функции \rm\eta\rm(\itx\rm)')
xlabel('\itx') % метка оси OX
```

```
ylabel('\eta\rm(\itx\rm)') % метка оси OY
xlim([0 1]); % пределы по оси OX
```

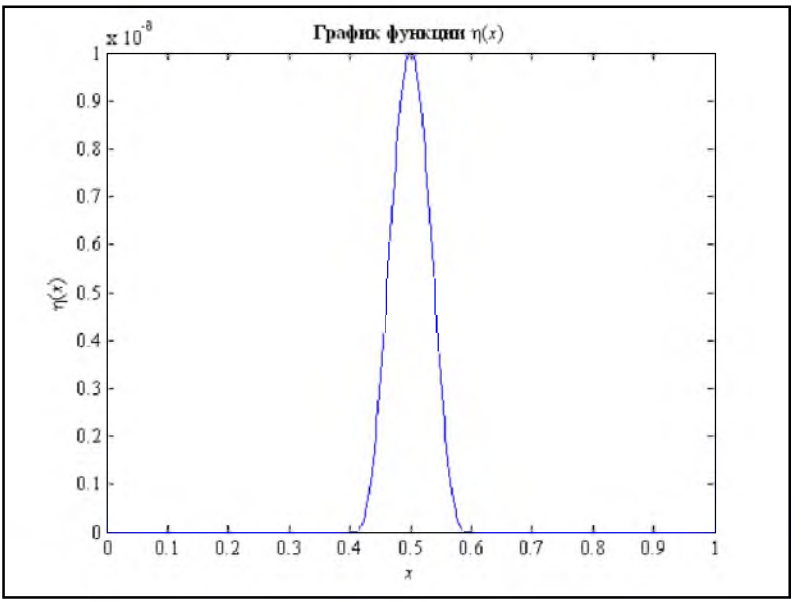


Рис. 1.22. График функции $\eta(x)$, нарисованный с помощью MATLAB

Тогда (1.54) будет нарушаться:

$$\int_{x_1}^{x_2} \Phi(x)\,\eta(x)\,dx = \int_{x_-}^{x_{2+}} \Phi(x)\,\eta(x)\,dx > 0, \tag{1.56}$$

т. к. каждый из сомножителей в подынтегральной функции положительный. Аналогично, если в какой-либо точке $\Phi(x_0) < 0$, то для этой же $\eta(x)$ интеграл (1.56) будет отрицательный. Отсюда по принципу от противного следует, что если $\forall\eta(x)$ будет выполняться (1.54), то это возможно, только если $\Phi(x) \equiv 0$ $\forall x \in [x_1, x_2]$. □

Замечание 1.3. Основная лемма вариационного исчисления справедлива и для функции нескольких переменных. Сформулируем и докажем ее для функции двух переменных. Формулируется она так: если $\forall\eta(x, y) \in C_k$ в области D имеет место

$$\iint_{(D)} \Phi(x, y)\,\eta(x, y)\,dS = 0, \tag{1.57}$$

то это возможно только в том случае, если $\Phi(x, y) \equiv 0$ $\forall (x, y) \in D$.

Доказательство проводится так же, методом от противного. Предположим, что в какой-то точке $(x_0, y_0) \in D$: $\Phi(x_0, y_0) = A > 0$. Значит, существует некоторая малая δ-окрестность точки (x_0, y_0), в которой $\Phi(x_0, y_0) > 0$. Построим функцию $\eta(x, y)$ в виде:

$$\eta(x,y) = \begin{cases} k\left((x-x_0)^2 + (y-y_0)^2 - \delta^2\right)^n; & x \in D; \\ 0; & x \notin D. \end{cases} \tag{1.58}$$

Нарисуем трехмерный график этой функции с помощью MATLAB. Посмотрите, как вычисляется функция `z` на заданной сетке. Мы проводим вычисления по первой части формулы (1.58), а затем умножаем поэлементно на булевский массив. При этом происходит автоматическое приведение типов, и булевский массив преобразуется в массив нулей и единиц. Таким образом, в области D вычисления проводятся по первой части (1.58), а вне D будет 0. Результат показан на рис. 1.23.

```
clear all % очистили все
x0=0.5;
y0=0.5;
delta=0.2;
n=2;
k=1;
[X,Y]=meshgrid(linspace(0,1),linspace(0,1));
z=k.*((X-x0).^2+(Y-y0).^2-delta^2).^n.*...
  ((delta^2-(X-x0).^2-(Y-y0).^2)>0);
surf(X,Y,z) % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title(['\bfГрафик функции '...
  '\rm\eta\rm(\itx\rm,\ity\rm)']) % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity') % метка оси OY
zlabel('\eta\rm(\itx\rm,\ity\rm)') % метка оси OZ
```

Для этой функции $\eta(x, y)$ условие (1.57) будет нарушаться: подынтегральная функция будет отлична от нуля (причем положительна) только в δ-окрестности точки (x_0, y_0), поэтому интеграл (1.57) будет положительный. Аналогично, если в какой-то точке $\Phi(x_0, y_0) < 0$, то для подобранной нами функции $\eta(x, y)$ интеграл (1.57) будет отрицательный. Следовательно, добиться выполнения (1.57) при произвольной $\eta(x, y)$ можно, только если $\Phi(x, y) \equiv \equiv 0 \ \forall (x, y) \in D$.

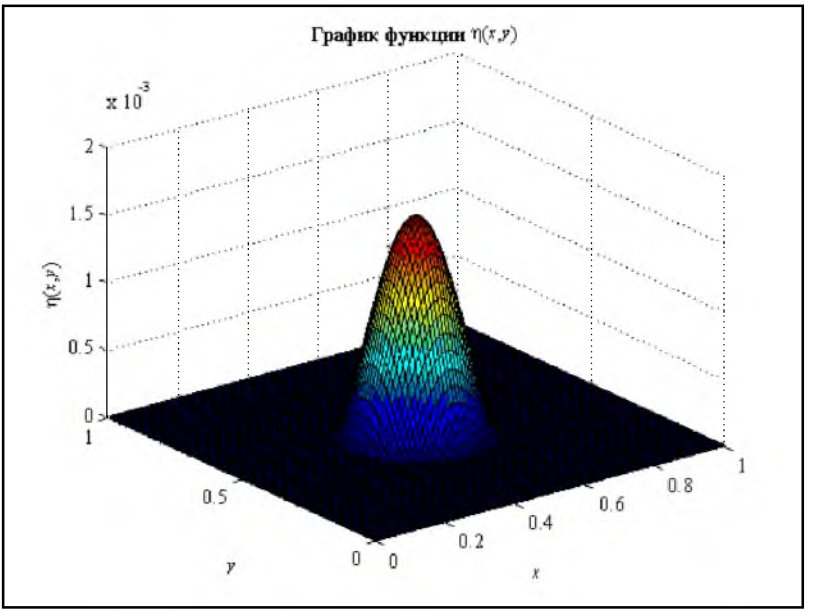


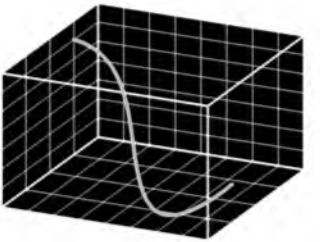
Рис. 1.23. График функции $\eta(x, y)$, нарисованный с помощью MATLAB

1.9. Вопросы для самопроверки

1. Что такое функционал? Чем он отличается от функции?
2. В чем состоит основная задача вариационного исчисления?
3. Какие классические задачи вариационного исчисления вы знаете?
4. Что называется классом функций? Какие классы вы знаете?
5. Какое утверждение правильное: $C_0 \subseteq C_1$ или $C_1 \subseteq C_0$?
6. Какое утверждение правильное: $C_k \subseteq C_{k+1}$ или $C_{k+1} \subseteq C_k$?
7. Сформулируйте и докажите теоремы 1.1 и 1.2 для C_k.
8. Найдите расстояние 2-го порядка между функциями $y_1(x) = x$ и $y_2(x) = -\cos x$ на отрезке $\left[0; \frac{\pi}{3}\right]$.
9. Найдите расстояние 1001-го порядка между функциями $y_1(x) = e^x$ и $y_2(x) = x$ на отрезке $[0; 1]$.
10. Установите порядок близости функций $y_1(x) = \frac{\sin x}{n}$ и $y_2(x) = \sin\frac{x}{n}$ к функции $y_0(x) = 0$ на отрезке $x \in [0, 2\pi]$ при $n \to \infty$.

11. Какой экстремум называется сильным? слабым? слабым k-го порядка?
12. Если на функции $y(x)$ достигается сильный экстремум, то достигается ли слабый? А если достигается слабый, то достигается ли сильный?
13. Если на функции $y(x)$ достигается экстремум k-го порядка, то будет ли на ней достигаться экстремум $(k+1)$-го порядка? $(k-1)$-го порядка?
14. Какой функционал называется непрерывным?
15. Что следует из непрерывности k-го порядка: непрерывность $(k+1)$-го или $(k-1)$-го порядка?
16. Исследуйте на непрерывность функционал $J(y(x)) = \max|y(x)|$, где функция $y(x)$ непрерывна на отрезке $[x_1, x_2]$ и имеет непрерывные производные любого нужного порядка.
17. Исследуйте на непрерывность функционал $J\left(y(x)\right) = \int_0^{\pi} \sqrt{1 + y'^2(x)}\, dx$ на функции $y_0(x) = 0$.
18. Какой функционал называется линейным?
19. Какой функционал называется варьируемым?
20. Если функционал J — варьируемый в C_k, то будет ли он варьируемым в C_{k+1}? в C_{k-1}?
21. Что называется вариацией функционала? Как она вычисляется? Какие вы знаете способы ее вычисления?
22. Сформулируйте необходимое условие экстремума функционала. Как оно доказывается?
23. Сформулируйте основную лемму вариационного исчисления и докажите ее.

ГЛАВА 2



Элементарная задача вариационного исчисления

2.1. Дифференциальное уравнение Эйлера

Рассмотрим функционал, зависящий от функции одной переменной $y(x)$ и ее производной $y'(x)$:

$$J(y)=\int_{x_1}^{x_2} F(x,y,y')dx \to \text{extr} \tag{2.1}$$

с заданными граничными условиями:

$$\begin{cases} y(x_1)=y_1; \\ y(x_2)=y_2, \end{cases} \tag{2.2}$$

где $F(x, y, y')$ — непрерывная функция трех переменных и дифференцируемая функция двух своих последних аргументов.

Определение 2.1. Задача вариационного исчисления (2.1—2.2) называется *элементарной*. □

Найдем решение этой задачи. Необходимым условием экстремума функционала является равенство нулю его вариации, вычисленной на экстремальной функции $y_0(x)$:

$$\delta J(y_0) = 0. \tag{2.3}$$

В нашем случае $\delta J(y)$ вызывается вариацией независимой переменной — функции $y(x)$ и ее производной $y'(x)$:

$$y(x)=y_0(x)+\delta y(x); \quad y'(x)=y'_0(x)+\delta y'(x), \tag{2.4}$$

причем в силу граничных условий (2.2) вариация функции на концах интервала равна нулю:

$$\delta y(x_1) = \delta y(x_2) = 0. \tag{2.5}$$

Экстремаль $y_0(x)$ и допустимые функции, удовлетворяющие условиям (2.5), показаны на рис. 2.1.

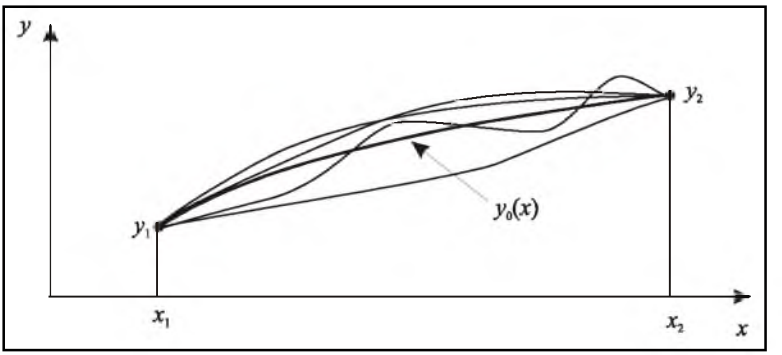


Рис. 2.1. Экстремаль и допустимые функции

Все допустимые функции удовлетворяют граничным условиям (2.2) и имеют малую вариацию (по крайней мере в смысле C_0) относительно экстремали $y_0(x)$. Необходимое условие экстремума функционала (2.3) требует, чтобы *на любой* такой вариации функции δy вариация функционала δJ обращалась в нуль.

Вычислим вариацию функционала как линейную часть его приращения ΔJ. Мы знаем два способа вычисления $\delta J(y)$. Воспользуемся первым способом: разложим $F(x, y_0 + \delta y, y'_0 + \delta y')$ в ряд Тейлора в окрестности экстремальной функции $y_0(x)$ с удержанием только линейных членов:

$$F(x, y_0 + \delta y, y'_0 + \delta y') = F(x, y_0, y'_0) + F_y(x, y_0, y'_0)\,\delta y + F_{y'}(x, y_0, y'_0)\,\delta y'. \tag{2.6}$$

Здесь символами F_y и $F_{y'}$ обозначены частные производные от функции F по ее второму и третьему аргументам (в вариационном исчислении в частных производных штрих вверху обычно не ставится). Вычислим функционал (2.1) на функции F с приращениями (т. е. вычисленной по (2.6)), и вычтем из него функционал, вычисленный на исходной функции без приращений. Поскольку мы в (2.6) удержали только линейные члены, то получим не приращение функционала, а линейную часть этого приращения, т. е. вариацию:

$$\delta J(y_0) = \int_{x_1}^{x_2} \left(F(x, y_0 + \delta y, y'_0 + \delta y') - F(x, y_0, y'_0) \right) dx = \int_{x_1}^{x_2} \left(F_y\,\delta y + F_{y'}\,\delta y' \right) dx. \tag{2.7}$$

Преобразуем полученное выражение, интегрируя 2-е слагаемое по частям:

$$\delta J(y_0) = \int_{x_1}^{x_2} F_y \delta y dx + \left(F_{y'} \delta y\right)\Big|_{x_1}^{x_2} - \int_{x_1}^{x_2} \frac{dF_{y'}}{dx} \delta y dx = \int_{x_1}^{x_2} \left(F_y - \frac{dF_{y'}}{dx} \right) \delta y dx = 0. \quad (2.8)$$

При интегрировании по частям внеинтегральное слагаемое равно нулю из-за граничных условий на концах интервала (2.5). Остается интеграл от произведения двух функций, который должен быть равен нулю. Так как вариация функции $\delta y(x)$ — произвольная, то в силу основной леммы вариационного исчисления первый сомножитель под интегралом должен равняться нулю. Таким образом, функция, на которой достигается экстремум, должна удовлетворять дифференциальному уравнению

$$F_y - \frac{dF_{y'}}{dx} = 0 . \quad (2.9)$$

Определение 2.2. Уравнение (2.9) называется *дифференциальным уравнением Эйлера* (Leonhard Euler, 1707—1783, рис. 2.2). □

Рис. 2.2. Л. Эйлер

Оно является в общем случае уравнением второго порядка. Действительно, функция F зависит от аргументов x, y и y'; F_y и $F_{y'}$ — это *частные* производные от F по ее второму и третьему аргументам, они также зависят только от этих аргументов. А $dF_{y'}/dx$ — это *полная* производная по x от частной производной $F_{y'}$, и в выражении для $dF_{y'}/dx$ появится уже *вторая* производная y''.

В нашем случае уравнение Эйлера дополняется двумя граничными условиями (2.2). Так как это не начальные, а граничные условия, то теорема Коши о существовании и единственности решения дифференциального уравнения

здесь неприменима. Иными словами, решение задачи (2.9) с граничными условиями (2.2) не обязательно существует, а если существует, то оно не обязательно единственное. Все зависит от вида уравнения Эйлера (2.9) и разрешимости системы уравнений для граничных условий (2.2).

Определение 2.3. Любое решение задачи (2.9) с граничными условиями (2.2) называется *экстремалью*. □

Это определение не противоречит *определению 1.3*, а обобщает его. Поскольку (2.3) — это только необходимое условие экстремума, но не достаточное, то экстремаль — это кривая, на которой может достигаться экстремум (а может и не достигаться). Для проверки наличия экстремума на найденной экстремали служат различные достаточные условия экстремума, которые мы изучим дальше, в *главе 13*.

ПРИМЕР 2.1. Рассмотрим пример, который легко решить аналитически. Требуется найти экстремум функционала

$$J(y)=\int_0^1\left(y'^2+12xy\right)dx\to\text{extr} \tag{2.10}$$

при граничных условиях

$$\begin{cases}y(0)=0;\\ y(1)=1.\end{cases} \tag{2.11}$$

Найдем частные производные F_y и $F_{y'}$:

$$F_y=12x;\quad F_{y'}=2y'. \tag{2.12}$$

Вычислим полную производную по x от $F_{y'}$:

$$\frac{dF_{y'}}{dx}=2y''. \tag{2.13}$$

Составим дифференциальное уравнение Эйлера вида (2.9):

$$12x-2y''=0, \tag{2.14}$$

или, после упрощений:

$$y''=6x. \tag{2.15}$$

Его общее решение имеет вид

$$y(x)=x^3+C_1x+C_2. \tag{2.16}$$

Для нахождения произвольных постоянных C_1 и C_2 подставим решение (2.16) в граничные условия (2.11):

$$\begin{cases} y(0) = C_2 = 0; \\ y(1) = 1 + C_1 + C_2 = 1. \end{cases} \tag{2.17}$$

Видно, что система (2.17) имеет единственное решение. Решая эту систему, найдем значения C_1 и C_2:

$$\begin{cases} C_1 = 0; \\ C_2 = 0, \end{cases} \tag{2.18}$$

и тогда уравнение экстремали имеет вид:

$$y(x) = x^3. \tag{2.19}$$

Действительно ли на этой кривой достигается экстремум? И если да, то какой: минимум или максимум? Далее, в *главе 13*, мы рассмотрим *достаточные условия экстремума*. В частности, мы выведем условие Лежандра: если на экстремали выполняется условие $F_{y'y'} > 0$, а на функциях, близких к экстремали, для произвольных y' имеет место $F_{y'y'} \geq 0$, то достигается сильный минимум. В нашем случае это выполняется:

$$F_{y'y'} = 2 > 0, \tag{2.20}$$

и, следовательно, на нашей экстремали достигается сильный минимум. Проверим этот результат: вычислим $J(y)$ на нескольких функциях вида $y_k = x^k$. Эти функции удовлетворяют граничным условиям (2.11) и, следовательно, являются допустимыми. Для вычислений применим MATLAB.

```
clear all % очистили все
syms x % описали символический аргумент
k=1:5; % показатели степени
y=x.^k; % символические функции
Dy=diff(y,x); % производные
F=Dy.^2+12*x.*y; % подынтегральные функции
J=int(F,x,0,1); % функционалы
Jm=eval(J); % посчитали значения
disp('k      J(x^k)') % печатаем заголовок
fprintf('%d   %12.10f\n',[k;Jm]) % печатаем результаты
k      J(x^k)
1   5.0000000000
2   4.3333333333
3   4.2000000000
4   4.2857142857
5   4.4920634921
```

Действительно, полученный результат не противоречит выводу о том, что на функции $y(x) = x^3$ достигается минимум. Но, конечно же, проведенная провер-

ка *не доказывает* этот факт. Ведь мы проверили только несколько из бесконечного числа функций, графики которых проходят через точки $M_1(0, 0)$ и $M_2(1, 1)$! Доказательством могут служить необходимые и достаточные условия экстремума функционала. □

ПРИМЕР 2.2. Найти экстремаль функционала

$$J(y) = \int_0^{2\pi} \left(y'^2 - y^2 \right) dx \to \text{extr} \tag{2.21}$$

при граничных условиях

$$\begin{cases} y(0) = 1; \\ y(2\pi) = 1. \end{cases} \tag{2.22}$$

Выводим уравнение Эйлера вида (2.9). Частные производные:

$$F_y = -2y; \quad F_{y'} = 2y'. \tag{2.23}$$

Уравнение Эйлера после упрощений имеет вид:

$$y'' + y = 0. \tag{2.24}$$

Его общее решение:

$$y(x) = C_1 \cos x + C_2 \sin x. \tag{2.25}$$

Находим произвольные постоянные из граничных условий (2.22). Подставляем решение (2.25) в эти граничные условия:

$$\begin{cases} y(0) = C_1 = 1; \\ y(2\pi) = C_1 = 1. \end{cases} \tag{2.26}$$

Мы видим, что из полученной системы уравнений можно найти только $C_1 = 1$, а C_2 может быть произвольной. Поэтому данная вариационная задача имеет бесчисленное множество решений вида

$$y(x) = \cos x + C_2 \sin x. \tag{2.27}$$

На любой из этих функций функционал $J(y)$ принимает постоянное значение (какое — мы сейчас посчитаем). Проверка по достаточному условию Лежандра дает:

$$F_{y'y'} = 2, \tag{2.28}$$

поэтому на экстремалях (2.27) достигается сильный минимум.

Посчитаем значение функционала (2.21) на функциях вида (2.27) и нарисуем несколько экстремалей с помощью MATLAB (рис. 2.3).

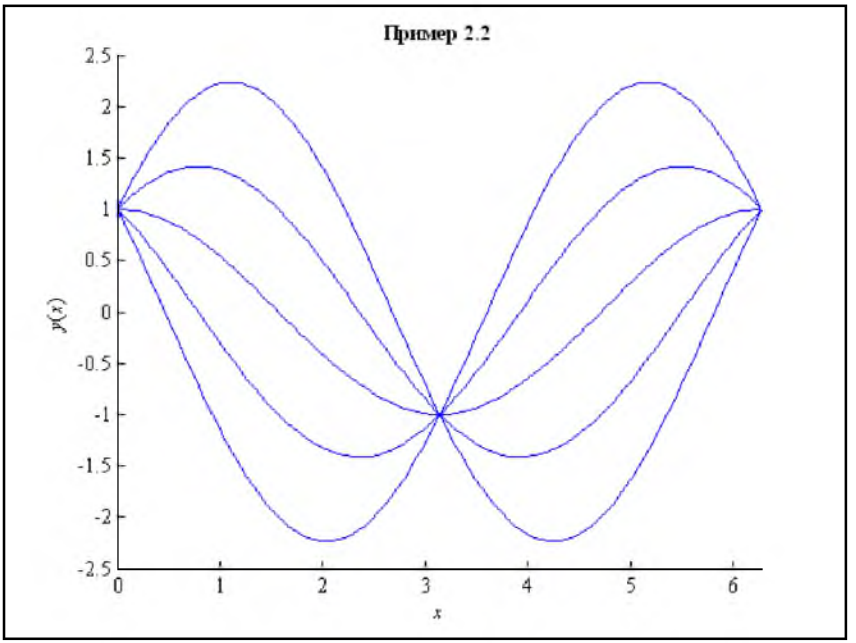


Рис. 2.3. Несколько экстремалей в примере 2.2

```
clear all % очистили память
syms x % описали символическую переменную
C2=-2:2; % несколько констант
y=cos(x)+C2*sin(x); % функции
Dy=diff(y,x); % производные
F=Dy.^2-y.^2; % подынтегральные функции
J=int(F,x,0,2*pi) % функционалы
xpl=linspace(0,2*pi); % абсциссы для графика
figure % новая фигура
hold on % для рисования нескольких графиков
for k=1:length(J), % заполняем ординаты и рисуем
  ypl=subs(y(k),x,xpl); % ординаты
  plot(xpl,ypl) % рисуем
end
hold off
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
xlim([0 2*pi]) % установили пределы по оси OX
da=daspect;
da(1:2)=min(da(1:2));
daspect(da); % одинаковый масштаб
```

```
title ('\bfПример 2.2')
xlabel ('\itx\rm')  % метка оси OX
ylabel ('\ity\rm(\itx\rm)')  % метка оси OY
J =
[ 0, 0, 0, 0, 0]
```

На каждой из наших функций функционал равен нулю. □

2.2. Частные случаи уравнений Эйлера

Иногда решение уравнения Эйлера существенно упрощается. Рассмотрим соответствующие частные случаи.

2.2.1. Подынтегральная функция *F* не зависит явно от *y'*

В этом случае $F_{y'}=0$, и уравнение Эйлера становится конечным. В его решении уже нет произвольных постоянных. Оно не обязательно удовлетворяет граничным условиям (2.2). Если эти условия удовлетворяются, то мы получим экстремаль, а если нет — то нет и решений у данной вариационной задачи.

ПРИМЕР 2.3. Решить вариационную задачу:

$$J(y)=\int_{x_1}^{x_2} y^2 dx \to \min; \quad \begin{cases} y(x_1)=y_1; \\ y(x_2)=y_2. \end{cases} \tag{2.29}$$

Здесь подынтегральная функция $F(x, y, y')=y^2$, т. е. не зависит от y'. Дифференциальное уравнение Эйлера (2.9) фактически стало конечным:

$$F_y - \frac{dF_{y'}}{dx} = 2y = 0. \tag{2.30}$$

Его решение:

$$y=0. \tag{2.31}$$

На непрерывных функциях (т. е. в классе C_0 и во всех вложенных в него классах) экстремум (минимум) достигается только при граничных условиях

$$\begin{cases} y(x_1)=0; \\ y(x_2)=0. \end{cases} \tag{2.32}$$

При других граничных условиях экстремума нет. □

2.2.2. Подынтегральная функция *F* линейно зависит от *y'*

Пусть подынтегральная функция F линейно зависит от y', т. е. имеет структуру:

$$F(x, y, y') = P(x, y) + Q(x, y)y'. \tag{2.33}$$

Выведем уравнение Эйлера.

$$\begin{aligned} F_y - \frac{dF_{y'}}{dx} &= \frac{\partial P}{\partial y} + y'\frac{\partial Q}{\partial y} - \frac{dQ}{dx} = \\ &= \frac{\partial P}{\partial y} + y'\frac{\partial Q}{\partial y} - \frac{\partial Q}{\partial x} - \frac{\partial Q}{\partial y}y' = \frac{\partial P}{\partial y} - \frac{\partial Q}{\partial x} = 0. \end{aligned} \tag{2.34}$$

Если условие

$$\frac{\partial P}{\partial y} - \frac{\partial Q}{\partial x} \equiv 0 \tag{2.35}$$

не выполняется, то (2.34) — конечное уравнение. Здесь возможны два варианта:

- если линия, определяемая этим уравнением, удовлетворяет граничным условиям, то в этом случае она является экстремалью;
- если граничные условия не удовлетворяются — то нет и решений у данной вариационной задачи.

Если же (2.35) выполняется, то вариационная задача теряет смысл: интеграл от функции вида (2.33) в этом случае вообще не зависит от линии интегрирования. На любой кривой, соединяющей граничные точки $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$, функционал принимает одинаковое значение. Вы ведь помните: (2.35) — это условие независимости криволинейного интеграла от линии интегрирования!

ПРИМЕР 2.4. Решить вариационную задачу:

$$J(y) = \int_0^1 \left(y^2 + x^2 y'\right) dx \to \text{extr}; \quad \begin{cases} y(0) = 0; \\ y(1) = a. \end{cases} \tag{2.36}$$

Выводим уравнение Эйлера:

$$2y - \frac{d\left(x^2\right)}{dx} = 0; \quad 2y - 2x = 0. \tag{2.37}$$

Его решение:

$$y = x. \tag{2.38}$$

Подставляем граничные условия:

$$\begin{cases} y(0)=0; & (\text{удовлетворяется}) \\ y(1)=a. \end{cases} \tag{2.39}$$

Если $a=1$, то линия $y=x$ является решением данной вариационной задачи. При $a\neq 1$ решений нет. □

ПРИМЕР 2.5. Решить вариационную задачу:

$$J(y)=\int_{x_1}^{x_2}(y+xy')dx\to \text{extr}; \quad \begin{cases} y(x_1)=y_1; \\ y(x_2)=y_2. \end{cases} \tag{2.40}$$

Выводим уравнение Эйлера:

$$F_y-\frac{dF_{y'}}{dx}=1-\frac{dx}{dx}=1-1\equiv 0. \tag{2.41}$$

Уравнение обратилось в тождество. Любая кривая, проходящая через граничные точки, доставляет функционалу одно и то же значение:

$$J(y)=\int_{x_1}^{x_2}\left(y+x\frac{dy}{dx}\right)dx=\int_{(x_1,y_1)}^{(x_2,y_2)}ydx+xdy=(xy)\Big|_{(x_1,y_1)}^{(x_2,y_2)}=x_2y_2-x_1y_1. \tag{2.42}$$

Функционал не зависит от линии интегрирования — вариационная задача теряет смысл. □

2.2.3. Подынтегральная функция *F* не зависит явно от *y'*

В этом случае в уравнении Эйлера (2.9) первое слагаемое равно нулю. Поскольку при этом полная производная по x от $F_{y'}$ обращается в нуль, то сама функция $F_{y'}$ должна быть постоянной. Эта функция зависит только от аргументов x и y'. Поэтому имеем первый интеграл:

$$F_{y'}(x,y')=C_1. \tag{2.43}$$

Это — уравнение уже не 2-го, а 1-го порядка, причем не содержащее в явном виде y. Возможно, оно решается легче, чем исходное уравнение 2-го порядка (2.9). Можно, например, попытаться разрешить (2.43) относительно y', а затем проинтегрировать.

ПРИМЕР 2.6. Материальная точка движется из положения $M_1(x_1,y_1)$ в положение $M_2(x_2,y_2)$ со скоростью, прямо пропорциональной координате x: $v=kx$.

По какой траектории она должна двигаться, чтобы прийти из $M_1(x_1, y_1)$ в $M_2(x_2, y_2)$ за минимально возможное время (рис. 2.4)?

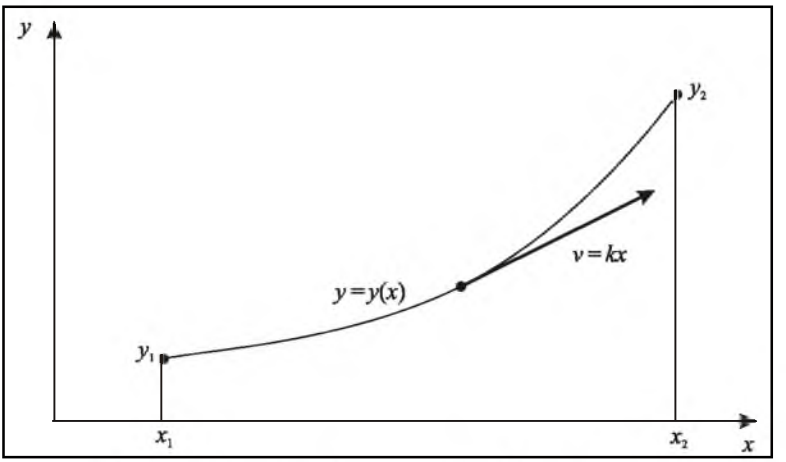


Рис. 2.4. Одна из постановок задачи о брахистохроне

Выведем функционал. Скорость — это производная пути по времени:

$$v = \frac{dl}{dt}. \tag{2.44}$$

Отсюда находим дифференциал времени:

$$dt = \frac{dl}{v} = \frac{\sqrt{1 + y'^2}\,dx}{kx}. \tag{2.45}$$

Все время прохождения траектории от $M_1(x_1, y_1)$ до $M_2(x_2, y_2)$:

$$T(y) = \int_{x_1}^{x_2} \frac{\sqrt{1 + y'^2}\,dx}{kx}. \tag{2.46}$$

Требуется минимизировать этот функционал при граничных условиях

$$\begin{cases} y(x_1) = y_1; \\ y(x_2) = y_2. \end{cases} \tag{2.47}$$

Подынтегральная функция не зависит от y. Запишем первый интеграл уравнения Эйлера (2.43) (множитель k внесем в C_1):

$$\frac{y'}{x\sqrt{1 + y'^2}} = C_1. \tag{2.48}$$

Полученное дифференциальное уравнение 1-го порядка решим путем введения параметра:

$$y' = \operatorname{ctg} t. \tag{2.49}$$

Из дифференциального уравнения (2.48) находим переменную x:

$$x = \frac{y'}{C_1\sqrt{1+y'^2}} = \frac{\operatorname{ctg} t}{C_1\sqrt{1+\operatorname{ctg}^2 t}} = \frac{\cos t}{C_1} = C_1^* \cos t. \tag{2.50}$$

Здесь мы переобозначили константу: $C_1^* = 1/C_1$. Теперь находим dy:

$$dy = \operatorname{ctg} t dx = -\operatorname{ctg} t C_1^* \sin t dt = -C_1^* \cos t dt. \tag{2.51}$$

Интегрируем:

$$y = -C_1^* \sin t + C_2. \tag{2.52}$$

Вновь переобозначим $C_1 = C_1^*$ и исключим параметр t из полученной системы уравнений:

$$\begin{cases} x = C_1 \cos t; \\ y = -C_1 \sin t + C_2. \end{cases} \tag{2.53}$$

Для этого во втором уравнении перенесем C_2 налево, возведем обе части в квадрат и сложим:

$$x^2 + (y - C_2)^2 = C_1^2. \tag{2.54}$$

Полученные экстремали — окружности с центрами на оси Oy. Произвольные постоянные C_1 и C_2 можно найти из граничных условий. Мы не будем этого делать: вы решали подобные задачи при изучении курса аналитической геометрии. □

2.2.4. Подынтегральная функция F зависит только от y'

Функция F зависит в общем случае от трех аргументов: x, y и y'. Поэтому полную производную по x от $F_{y'}$ (второе слагаемое уравнения Эйлера (2.9)) можно расписать так:

$$\frac{dF_{y'}}{dx} = \frac{\partial F_{y'}}{\partial x} + \frac{\partial F_{y'}}{\partial y}\frac{dy}{dx} + \frac{\partial F_{y'}}{\partial y'}\frac{dy'}{dx} = F_{xy'} + F_{yy'}y' + F_{y'y'}y'', \tag{2.55}$$

а само уравнение Эйлера записать в развернутом виде, через частные производные:

$$F_y - F_{xy'} - F_{yy'}y' - F_{y'y'}y'' = 0. \tag{2.56}$$

Если F зависит только от y', то в (2.56) частные производные по x и y обращаются в нуль, и остается только последнее слагаемое:

$$F_{y'y'}y''=0. \tag{2.57}$$

Произведение равно нулю, когда хотя бы один из сомножителей обращается в нуль. Поэтому экстремали находятся из уравнения

$$y''=0 \tag{2.58}$$

или уравнения

$$F_{y'y'}=0. \tag{2.59}$$

Решения уравнения (2.58) — это различные прямые:

$$y=C_1x+C_2. \tag{2.60}$$

Уравнение (2.59) — это конечное уравнение относительно y'; оно имеет какие-то корни:

$$\left[\begin{array}{l} y'=k_1; \\ y'=k_2; \\ \quad\cdots \\ y'=k_n. \end{array}\right. \tag{2.61}$$

Каждое из этих решений определяет прямую, т. е. является частным случаем общего уравнения прямой (2.60). Таким образом, экстремали функционала, зависящего только от y, — это прямые.

ПРИМЕР 2.7. Найти кратчайшее расстояние между двумя точками $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$.

Составим функционал:

$$L(y)=\int_{x_1}^{x_2}\sqrt{1+y'^2}\,dx\to\min. \tag{2.62}$$

Подынтегральная функция зависит только от y' — решением будет прямая (2.60). Подставив граничные условия, найдем C_1 и C_2. Вы ведь умеете проводить прямую через 2 точки? □

2.2.5. Подынтегральная функция *F* не зависит явно от *x*

В этом случае в уравнении (2.56) тождественно равно нулю второе слагаемое: там есть частная производная по x.

Остается:

$$F_y - F_{yy'}y' - F_{y'y'}y'' = 0. \tag{2.63}$$

Но такой же вид имеет и полная производная по x от выражения

$$F - y'F_{y'} = C_1. \tag{2.64}$$

Действительно, продифференцируем (2.64). При дифференцировании учтем, что F и ее частные производные от x не зависят:

$$\begin{aligned} &F_y y' + F_{y'} y'' - y''F_{y'} - y'\left(F_{yy'}y' + F_{y'y'}y''\right) = \\ &= F_y y' - F_{yy'}y'^2 - F_{y'y'}y'y'' = 0, \end{aligned} \tag{2.65}$$

что после сокращения на y' совпадает с (2.63). Таким образом, если подынтегральная функция не зависит явно от x, и решением не является горизонтальная прямая ($y' \neq 0$), то дифференциальное уравнение Эйлера (2.9) имеет первый интеграл (2.64). Он является уравнением 1-го порядка, которое решается проще, чем исходное уравнение (2.9).

ПРИМЕР 2.8. Задача о брахистохроне *(см. раздел 1.2)*. Выберем систему координат так, как показано на рис. 1.6. Поместим начало координат в точку старта $O(0, 0)$. Ось Ox направим горизонтально вдоль траектории движения, а Oy — вертикально вниз. Для вывода функционала воспользуемся законом сохранения энергии. Ввиду отсутствия диссипативных сил

$$T + U = T_0 + U_0, \tag{2.66}$$

где T — кинетическая энергия, U — потенциальная, индекс 0 означает начальный момент времени. В начальный момент скорость нулевая, поэтому

$$T_0 = 0. \tag{2.67}$$

Начало отсчета потенциальной энергии также выберем в точке $M_1(0, 0)$. Поэтому

$$U_0 = 0. \tag{2.68}$$

В любой точке $M(x, y)$ кинетическая энергия

$$T = \frac{mv^2}{2}. \tag{2.69}$$

Потенциальная энергия в этой же точке $M(x, y)$ равна

$$U = -mgy. \tag{2.70}$$

Здесь m — масса, g — ускорение силы тяжести. Знак "минус" появился потому, что ось Oy направлена вниз, и увеличению координаты y соответствует уменьшение потенциальной энергии U. Составляем уравнение закона сохранения энергии:

$$\frac{mv^2}{2} - mgy = 0. \tag{2.71}$$

Отсюда находим скорость v, которая является производной пути по времени:

$$\frac{dl}{dt} = v = \sqrt{2gy}. \tag{2.72}$$

Дифференциал времени равен:

$$dt = \frac{dl}{\sqrt{2gy}} = \frac{\sqrt{1+y'^2}\,dx}{\sqrt{2gy}}, \tag{2.73}$$

а все время прохождения дистанции будет:

$$T(y) = \frac{1}{\sqrt{2g}} \int_{x_1}^{x_2} \frac{\sqrt{1+y'^2}}{\sqrt{y}}\,dx \to \min. \tag{2.74}$$

Подынтегральная функция в этом функционале не зависит явно от x. Горизонтальные прямые не являются решениями: $y' \neq 0$. Запишем первый интеграл уравнения Эйлера (2.64). Постоянный множитель перед интегралом можно сократить (или внести в C_1). Имеем:

$$\frac{\sqrt{1+y'^2}}{\sqrt{y}} - y' \frac{y'}{\sqrt{y}\sqrt{1+y'^2}} = C_1. \tag{2.75}$$

Упрощаем. После приведения к общему знаменателю получим:

$$\frac{1}{\sqrt{y}\sqrt{1+y'^2}} = C_1. \tag{2.76}$$

Возведем в квадрат и переобозначим константу:

$$y(1+y'^2) = C_1. \tag{2.77}$$

Решим уравнение при помощи введения параметра:

$$y' = \operatorname{ctg} t. \tag{2.78}$$

Из уравнения (2.77) переменная y выражается через параметр:

$$\begin{aligned} y\left(1+\operatorname{ctg}^2 t\right) &= \frac{y}{\sin^2 t} = C_1; \\ y &= C_1 \sin^2 t = \frac{C_1}{2}\left(1-\cos 2t\right). \end{aligned} \tag{2.79}$$

Теперь найдем x. Вначале находим dx:

$$dx = \frac{dy}{y'} = \frac{C_1 2\sin t\cos t dt}{\operatorname{ctg} t} = 2C_1 \sin^2 t dt = C_1\left(1-\cos 2t\right)dt, \tag{2.80}$$

а затем интегрируем:

$$x = \frac{C_1}{2}\left(2t - \sin 2t\right). \tag{2.81}$$

Еще раз переобозначим константу: $C_1/2 \Rightarrow C_1$ и параметр: $2t \Rightarrow t$. Из (2.79) и (2.81) получаем уравнения циклоиды:

$$\begin{cases} x = C_1\left(t - \sin t\right) + C_2; \\ y = C_1\left(1 - \cos t\right). \end{cases} \tag{2.82}$$

Произвольные постоянные C_1 и C_2 находим из граничных условий. Подставляем левую точку $M_1(0, 0)$:

$$y = 0;\ \Rightarrow\ \cos t = 1;\ \Rightarrow\ t = 0;\ \Rightarrow\ x = C_2 = 0. \tag{2.83}$$

Теперь подставим правую точку $M_2(x_2, y_2)$ и найдем C_1 и соответствующее значение параметра t_2 из решения системы нелинейных уравнений:

$$\begin{cases} C_1\left(t_2 - \sin t_2\right) = x_2; \\ C_1\left(1 - \cos t_2\right) = y_2. \end{cases} \tag{2.84}$$

Мы будем решать эту систему дальше в этой главе, в *разделе 2.4.3*.

Исследуем полученную экстремаль на выполнение достаточных условий экстремума по Лежандру. Вычислим $F_{y'y'}$:

$$F_{y'y'} = \frac{1}{\sqrt{y}\left(1 + y'^2\right)^{\frac{3}{2}}}. \tag{2.85}$$

У нас $y > 0$: второй сомножитель знаменателя всегда положительный, поэтому $F_{y'y'} > 0$ для всех y, близких к экстремали, и для всех y'. В соответствии с достаточным условием Лежандра достигается сильный минимум. □

2.3. Вопросы для самопроверки

1. Какую вариационную задачу мы решаем?
2. Как выводится дифференциальное уравнение Эйлера?

3. Где используется в выводе дифференциального уравнения Эйлера основная лемма вариационного исчисления?
4. Почему обращается в нуль внеинтегральное слагаемое в формуле (2.8) при интегрировании по частям?
5. Чем отличается частная производная от полной?
6. Какие вы знаете методы решения дифференциальных уравнений 2-го порядка?
7. Всегда ли решение вариационной задачи будет единственным? От чего это зависит?
8. Какие частные случаи уравнения Эйлера вы знаете?
9. В каких случаях уравнение Эйлера перестает быть дифференциальным и становится конечным?
10. В каких случаях вариационная задача теряет смысл?
11. Как записывается 1-й интеграл уравнения Эйлера, если подынтегральная функция F не зависит явно от y?
12. Каким будет решение уравнения Эйлера, если подынтегральная функция F зависит только от y'?
13. Как решается уравнение Эйлера, если подынтегральная функция F не зависит явно от y'?
14. Как решается задача о брахистохроне?

2.4. Примеры выполнения заданий

2.4.1. Задание 1

Найти экстремаль функционала

$$J(y)=\int_{-1}^{1}\left(x^2+y^2+y'^2\right)dx\to\text{extr};\quad\begin{cases}y(-1)=1;\\ y(1)=2.\end{cases}\tag{2.86}$$

Исследовать полученную экстремаль на достаточные условия экстремума. Вычислить значение функционала на найденной экстремали и, для сравнения, на прямой, соединяющей точки $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$. Построить график решения.

В этом примере подынтегральная функция $F(x, y, y')$ является функцией общего вида, поэтому составим уравнение Эйлера в виде (2.9) и решим его. Затем построим график решения. Попутно исследуем на выполнение достаточных условий экстремума и вычислим значение функционала на экстремали и отрезке прямой M_1M_2.

Применим для решения задачи MATLAB. Посмотрите в *приложении 1*, как выполняются символические вычисления в MATLAB, решаются конечные и дифференциальные уравнения, строятся графики.

Очистим память. Напечатаем заголовок решаемой задачи. Если хотите, задайте другую строку для вывода (например, свою фамилию). Опишем символические переменные [58]. Для решения уравнения Эйлера используем принятые в MATLAB обозначения производных: `Dy` для y' и `D2y` для y''. Аргумент обозначим `x`, а функцию — `y`.

```
clear all % очистили память
disp('Решаем задание 2.1') % выводим заголовок задачи
syms x y Dy D2y % описали символические переменные
Решаем задание 2.1
```

Вводим подынтегральную функцию и граничные условия. Печатаем их. Чтобы напечатать символ `'`, нужно его удвоить. Здесь вы должны поставить свои исходные данные: подынтегральную функцию F и граничные условия x_1, y_1, x_2, y_2.

```
F=x^2+y^2+Dy^2; % подынтегральная функция
x1=-1; % граничные условия
y1=1;
x2=1;
y2=2;
disp('Исходные данные:')
fprintf('Подынтегральная функция F(x,y,y'')=%s\n',char(F))
fprintf('Граничное условие слева: y(%d)=%d\n',x1,y1)
fprintf('Граничное условие справа: y(%d)=%d\n',x2,y2)
Исходные данные:
Подынтегральная функция F(x,y,y')=x^2+y^2+Dy^2
Граничное условие слева: y(-1)=1
Граничное условие справа: y(1)=2
```

Начинаем вывод дифференциального уравнения Эйлера (2.9). Найдем частные производные F_y и $F_{y'}$. Напечатаем их.

```
dFdy=diff(F,y); % вычислили Fy
dFdy1=diff(F,Dy); % вычислили Fy'
fprintf('Fy=%s\n',char(dFdy))
fprintf('Fy''=%s\n',char(dFdy1))
Fy=2*y
Fy'=2*Dy
```

В уравнение Эйлера (2.9) входит полная производная $dF_{y'}/dx$. Вычислим ее по обычной формуле дифференцирования сложной функции:

$$\frac{dF_{y'}}{dx} = \frac{\partial F_{y'}}{\partial x} + \frac{\partial F_{y'}}{\partial y} y' + \frac{\partial F_{y'}}{\partial y'} y''. \tag{2.87}$$

Напечатаем ее. Напечатаем также величину $F_{y'y'}$, необходимую для проверки достаточных условий экстремума по признаку Лежандра.

```
d_dFdy1_dx=diff(dFdy1,x); % Fxy'x
d_dFdy1_dy=diff(dFdy1,y); % Fyy'
d_dFdy1_dy1=diff(dFdy1,Dy); % Fy'y'-условие Лежандра
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+d_dFdy1_dy1*D2y;
fprintf('dFy''/dx=%s\n',char(dFy1dx))
disp('Условие Лежандра:')
fprintf('Fy''y''=%s\n',char(d_dFdy1_dy1))
dFy'/dx=2*D2y
Условие Лежандра:
Fy'y'=2
```

Составим левую часть дифференциального уравнения Эйлера (2.9) и упростим ее. Преобразуем символическую переменную `Euler` в строку.

```
Euler=simple(dFdy-dFy1dx); % уравнение Эйлера
deqEuler=[char(Euler) '=0']; % в строку; добавили =0
fprintf('Уравнение Эйлера:\n%s\n',deqEuler)
Уравнение Эйлера:
2*y-2*D2y=0
```

Мы составили уравнение Эйлера, теперь решим его. Команда `dsolve` позволяет находить как общее решение дифференциального уравнения, так и частное его решение, удовлетворяющее заданным начальным или граничным условиям. В следующих главах при решении других заданий нам нужно будет иметь общее решение уравнения Эйлера. Найдем его.

```
Sol=dsolve(deqEuler,'x'); % решаем уравнение Эйлера
if length(Sol)~=1 % решений нет или более одного
  error('Нет решений или более одного решения!');
else
  disp('Общее решение уравнения Эйлера:')
  fprintf('y(x)=%s\n',char(Sol))
end
Общее решение уравнения Эйлера:
y(x)=C1*sinh(x)+C2*cosh(x)
```

Сформируем теперь уравнения для граничных условий. Подставим в найденное аналитическое решение `Sol` граничные точки x_1 и x_2, и приравняем их соответственно y_1 и y_2.

```
SolLeft=subs(Sol,x,x1); % подставили x1
SolRight=subs(Sol,x,x2); % подставили x2
EqLeft=[char(SolLeft) '=' char(sym(y1))]; % =y1
EqRight=[char(SolRight) '=' char(sym(y2))]; % =y2
disp('Уравнения для граничных условий:')
fprintf('%s\n',EqLeft,EqRight)
Уравнения для граничных условий:
-C1*sinh(1)+C2*cosh(1)=1
C1*sinh(1)+C2*cosh(1)=2
```

Решаем полученную систему конечных уравнений — находим значения произвольных постоянных `C1` и `C2`. Присваиваем найденные решения символическим константам, полученным при решении дифференциального уравнения. Теперь вычисляем аналитическое решение `Sol21`. Такое вычисление сводится к тому, что в него будут подставлены найденные значения констант `C1` и `C2`. Печатаем найденное уравнение экстремали.

```
Con=solve(EqLeft,EqRight,'C1,C2'); % решаем систему
C1=Con.C1; % присваиваем полученные решения
C2=Con.C2; % символическим константам C1 и C2
Sol21=vpa(eval(Sol),14); % подставили C1,C2
disp('Уравнение экстремали:')
fprintf('y(x)=%s\n',char(Sol21))
Уравнение экстремали:
y(x)=.42545906411967*sinh(x)+.97208141049585*cosh(x)
```

Вычислим значения функционала (2.86) на найденной экстремали и на прямой, соединяющей точки M_1 и M_2. Подставим в подынтегральную функцию F аналитические выражения для этих линий и их производных, а затем проинтегрируем. Напечатаем результаты.

```
Fextr=subs(F,{y,Dy},{Sol21,diff(Sol21,x)});
Jextr=eval(int(Fextr,x,x1,x2))
ylin=(x-x1)*(y2-y1)/(x2-x1)+y1;
Flin=subs(F,{y,Dy},{ylin,diff(ylin,x)});
Jlin=eval(int(Flin,x,x1,x2))
Jextr =
   4.75035801121746
Jlin =
   5.83333333333333
```

В данном примере условие Лежандра говорит о сильном минимуме, что подтверждается полученным результатом: значение функционала на экстремали меньше, чем на другой допустимой функции. А как в вашем варианте: какой экстремум достигается? И подтверждается ли этот результат сравнением величин `Jextr` и `Jlin`? Если нет, то не забудьте, что найденный экстремум — только локальный, а не глобальный! Попробуйте вычислить значение функционала не на прямой M_1M_2, а на какой-нибудь другой допустимой кривой, достаточно близкой к экстремали. Например, можно наложить на экстремаль несколько полуволн синусоиды, смещенной и деформированной вдоль оси Ox так, чтобы $[0, \pi] \Rightarrow [x_1, x_2]$.

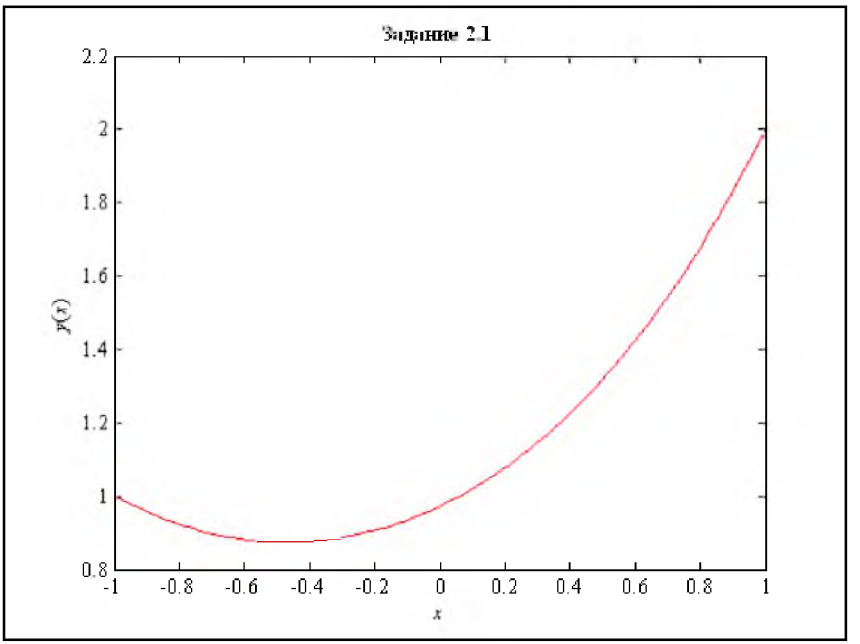


Рис. 2.5. График экстремали в задании 2.1

И наконец, строим график. Задаем массив аргументов для рисования графика функции и вычисляем значения функции. Рисуем график (рис. 2.5), подписываем заголовок и координатные оси установленным шрифтом.

```
xpl=linspace(x1,x2); % задаем массив абсцисс
y21=subs(Sol21,x,xpl); % вычислили ординаты
figure % фигура
plot(xpl,y21,'-r') % рисуем график красной линией
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
```

```
title('\bfЗадание 2.1') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
```

2.4.2. Задание 2

Найти экстремаль функционала

$$J(y)=\int_{-1}^{1}\left(y'^2+2y'\operatorname{sh}x-5x^2\right)dx\to\text{extr};\quad\begin{cases}y(-1)=2;\\ y(1)=3.\end{cases}\tag{2.88}$$

Исследовать на выполнение достаточных условий экстремума. Построить график решения.

В этом примере подынтегральная функция $F(x,y,y')$ не зависит явно от y. Первый интеграл уравнения Эйлера имеет вид (2.43). Составим программу для решения этой вариационной задачи. Вначале введем исходные данные. У нас будет первый интеграл уравнения Эйлера, поэтому ни сама функция y, ни ее вторая производная y'' нам не нужны, и мы их не описываем. Поставьте свою подынтегральную функцию и граничные условия.

```
clear all % очистили все
disp('Решаем задание 2.2') % заголовок задачи
syms x Dy % описали символические переменные
F=Dy^2+2*Dy*sinh(x)-5*x^2; % подынтегральная функция
x1=-1; % граничные условия
y1=2;
x2=1;
y2=3;
disp('Исходные данные:')
fprintf('Подынтегральная функция F(x,y'')=%s\n',char(F))
fprintf('Граничное условие слева: y(%d)=%d\n',x1,y1)
fprintf('Граничное условие справа: y(%d)=%d\n',x2,y2)
Решаем задание 2.2
Исходные данные:
Подынтегральная функция F(x,y')=Dy^2+2*Dy*sinh(x)-5*x^2
Граничное условие слева: y(-1)=2
Граничное условие справа: y(1)=3
```

Строим первый интеграл и решаем полученное дифференциальное уравнение. Названия констант `C1` и `C2` используются в команде `dsolve`, поэтому при составлении 1-го интеграла уравнения Эйлера обозначим константу `C`. Все использованные здесь функции и операторы MATLAB были описаны ранее, в *задании 1*.

```
dFdy1=simple(diff(F,Dy)); % Fy'
deqEuler=[char(dFdy1) '=C']; % составили уравнение
disp('Первый интеграл уравнения Эйлера:')
fprintf('%s\n',deqEuler)
Sol=dsolve(deqEuler,'x'); % решаем уравнение Эйлера
if length(Sol)~=1 % решений нет или более одного
  error('Нет решений или более одного решения!');
else
  disp('Общее решение уравнения Эйлера:')
  fprintf('y(x)=%s\n',char(Sol))
end
Первый интеграл уравнения Эйлера:
2*Dy+2*sinh(x)=C
Общее решение уравнения Эйлера:
y(x)=-cosh(x)+1/2*C*x+C1
```

В переменной `Sol` получено общее решение, произвольные постоянные обозначены `C` и `C1`. Найдем их. Для этого подставим в `Sol` граничные точки x_1 и x_2. Приравняем полученные выражения соответственно y_1 и y_2. Тем самым мы сформируем систему уравнений.

```
SolLeft=subs(Sol,x,sym(x1)); % подставили x1
SolRight=subs(Sol,x,sym(x2)); % подставили x2
EqLeft=[char(SolLeft) '=' char(sym(y1))]; % =y1
EqRight=[char(SolRight) '=' char(sym(y2))]; % =y2
disp('Уравнения для граничных условий:')
fprintf('%s\n',EqLeft,EqRight)
Уравнения для граничных условий:
-cosh(1)-1/2*C+C1=2
-cosh(1)+1/2*C+C1=3
```

Решим полученную систему — найдем произвольные постоянные `C` и `C1`. Подставим их в решение `Sol`. Ограничим решение 14 знаками. Напечатаем уравнение найденной экстремали.

```
Con=solve(EqLeft,EqRight,'C,C1'); % решаем
C=Con.C; % присваиваем полученные решения
C1=Con.C1; % символическим переменным C и C1
Sol22=vpa(eval(Sol),14); % подставили C1, C
disp('Уравнение экстремали:')
fprintf('y(x)=%s\n',char(Sol22))
Уравнение экстремали:
y(x)=-1.*cosh(x)+.50000000000000*x+4.0430806348152
```

Дальнейшие действия не отличаются от описанных в *задании 1*. Рисуем график (рис. 2.6) и вычисляем $F_{y'y'}$, которая нужна для проверки достаточных условий экстремума по признаку Лежандра.

```
xpl=linspace(x1,x2); % задали массив абсцисс
y22=subs(Sol22,x,xpl); % вычислили ординаты
figure % фигура
plot(xpl,y22,'-r') % рисуем график красной линией
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 2.2') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
Leg=diff(dFdy1,Dy); % Fy'y'
disp ('Достаточное условие Лежандра:')
fprintf('Fy''y''=%s\n',char(Leg))
Достаточное условие Лежандра:
Fy'y'=2
```

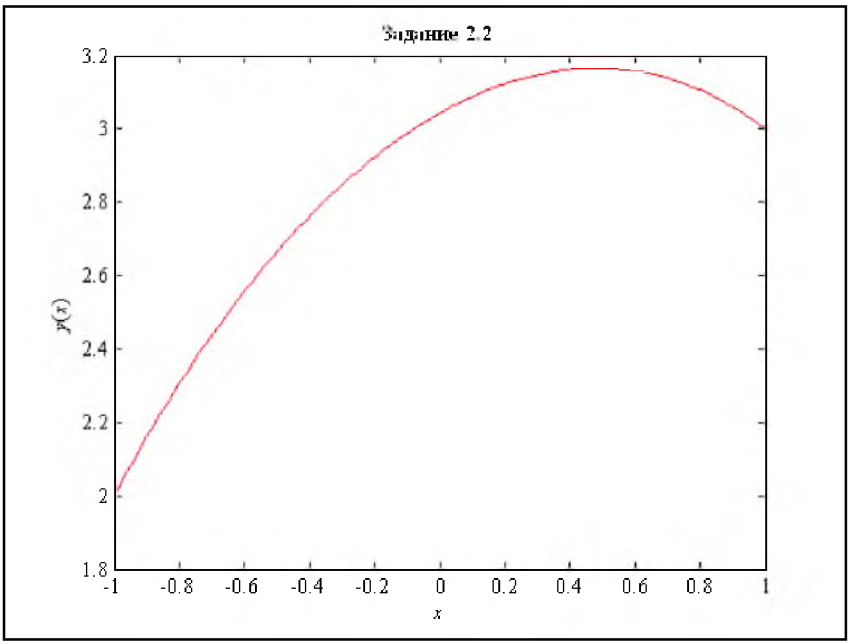


Рис. 2.6. График экстремали в задании 2.2

Проанализируйте достаточное условие Лежандра. Достигается ли экстремум на вашей экстремали? Если да, то какой?

2.4.3. Задание 3

Решить задачу о брахистохроне, соединяющей точки $M_1(0, 0)$ и $M_2(2, 1)$.

Мы уже решили эту задачу аналитически *(см. пример 2.8)*. Нам осталось найти значение константы C_1 и параметра в конечной точке t_2 из решения системы уравнений (2.84). Составим программу для решения этого примера. Вначале введем исходные данные задачи. Подставьте свою правую точку.

```
clear all % очистили все
disp('Решаем задание 2.3') % выводим заголовок задачи
x2=2;
y2=1;
fprintf('Правая точка: y(%d)=%d\n',x2,y2)
Решаем задание 2.3
Правая точка: y(2)=1
```

Составляем систему уравнений (2.84). Левую часть каждого уравнения мы задаем сразу в виде строки. В правой части переводим числовые переменные `x2` и `y2` в их строковые представления с помощью функции `num2str`. Ранее мы использовали конструкцию `char(sym(y2))`. Оба варианта работают правильно — вы можете это проверить. Решаем полученную систему уравнений аналитически. Печатаем решения.

```
eq1=['C1*(t2-sin(t2))=' num2str(x2)];
eq2=['C1*(1-cos(t2))=' num2str(y2)];
fprintf('Система уравнений:\n%s\n%s\n',eq1,eq2)
Sol=solve(eq1,eq2,'C1,t2');
C1=eval(Sol.C1);
t2=eval(Sol.t2);
disp('Найденное решение:')
fprintf('Константа C1=%10.5f\n',C1)
fprintf('Параметр t2=%10.5f\n',t2)
disp('Уравнения брахистохроны:')
fprintf('x(t)=%10.5f(t-sin(t))\n',C1)
fprintf('y(t)=%10.5f(1-cos(t))\n',C1)
Система уравнений:
C1*(t2-sin(t2))=2
C1*(1-cos(t2))=1
Найденное решение:
Константа C1=   0.51720
Параметр t2=   3.50837
Уравнения брахистохроны:
x(t)=   0.51720(t-sin(t))
y(t)=   0.51720(1-cos(t))
```

Рисуем график (рис. 2.7) полученной брахистохроны. Выбираем начало координат в левом верхнем углу с помощью команды `axis`. Задаем границы по оси *Ox*, чтобы график занимал все место на рисунке. Выравниваем масштабы по осям координат: брахистохрона должна выглядеть неискаженной. Надписываем заголовок и метки осей.

```
t=linspace(0,t2); % задали массив параметров
x=C1*(t-sin(t)); % вычислили абсциссы
y=C1*(1-cos(t)); % вычислили ординаты
figure % фигура
plot(x,y) % рисуем график
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
axis ij % установили 0 в левом верхнем углу
xlim([0 x2]) % установили пределы по оси 0x
da=daspect;
da(1:2)=min(da(1:2));
daspect(da); % одинаковый масштаб
title ('\bfЗадание 2.3 - задача о брахистохроне')
xlabel ('\itx\rm(\itt\rm)') % метка оси OX
ylabel ('\ity\rm(\itt\rm)') % метка оси OY
```

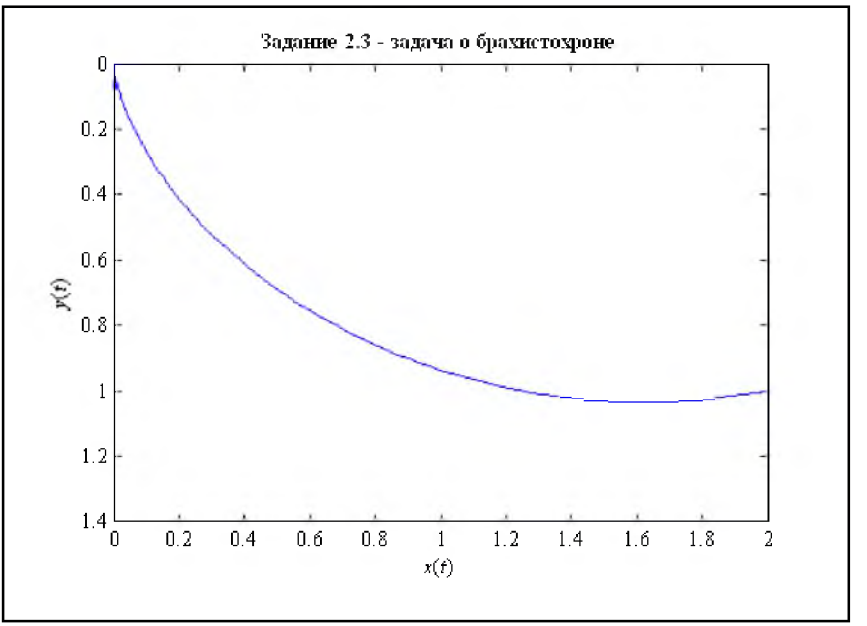
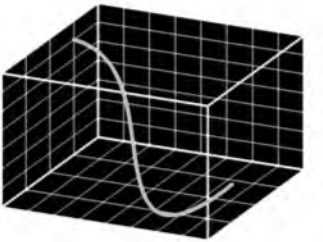


Рис. 2.7. Брахистохрона в задании 2.3

2.5. Задание

Для своего варианта функционалов 1, 2, 3 найти экстремали, построить их графики и исследовать на выполнение достаточных условий экстремума. Варианты заданий находятся на компакт-диске.

ГЛАВА 3



Функционалы, зависящие от нескольких функций

3.1. Система дифференциальных уравнений Эйлера

В предыдущей *главе 2* мы рассмотрели функционал (2.1), зависящий от одной функции y и ее производной y'. Пусть теперь функционал зависит не от одной, а от нескольких функций одной переменной и их первых производных:

$$J\left(y_1, y_2, \ldots, y_n\right) = \int_{x_1}^{x_2} F\left(x, y_1, y_2, \ldots, y_n, y_1', y_2', \ldots, y_n'\right) dx \to \text{extr} . \qquad (3.1)$$

Экстремаль функционала должна удовлетворять в общем случае $2n$ граничным условиям:

$$\begin{cases} y_1\left(x_1\right) = y_{11}; \\ y_1\left(x_2\right) = y_{12}; \end{cases} \quad \begin{cases} y_2\left(x_1\right) = y_{21}; \\ y_2\left(x_2\right) = y_{22}; \end{cases} \quad \ldots \quad \begin{cases} y_n\left(x_1\right) = y_{n1}; \\ y_n\left(x_2\right) = y_{n2}. \end{cases} \qquad (3.2)$$

Ранее мы вывели необходимое условие экстремума любого функционала: равенство нулю его вариации, вычисленной на экстремали. Для функционала (3.1), зависящего от нескольких функций, оно записывается так:

$$\delta J(y_{10}, y_{20}, \ldots, y_{n0}) = 0, \qquad (3.3)$$

где $\{y_{10}(x), y_{20}(x), \ldots, y_{n0}(x)\}$ — экстремали. Воспользуемся *выводами 1.1 и 1.3*, которые мы сформулировали в *главе 1*. В нашем случае они дают следующее:

✔ Если достигается экстремум некоторого функционала на множестве варьируемых функций $\{y_{10}(x), y_{20}(x), \ldots, y_{n0}(x)\}$, то будет достигаться

экстремум и на более узком классе функций: когда варьируется только одна из них, а остальные зафиксированы на экстремалях.

Поэтому мы можем в функционале (3.1) зафиксировать на экстремалях все функции, кроме одной $y_i(x)$. Получаем функционал, зависящий от одной функции $y_i(x)$. На экстремали $y_{i0}(x)$ она должна удовлетворять уравнению Эйлера (2.9). Остальные функции входят в это уравнение как параметры: они зафиксированы на экстремалях.

Повторяя эти рассуждения для каждой функции, приходим к выводу, что для нахождения экстремалей нужно решить систему n дифференциальных уравнений Эйлера

$$\begin{cases} F_{y_i} - \dfrac{dF_{y_i'}}{dx} = 0; \\ i = [1, n], \end{cases} \tag{3.4}$$

дополненную $2n$ граничными условиями (3.2).

Данную вариационную задачу при $n = 2$ можно интерпретировать геометрически. Обозначим функции через $y(x)$ и $z(x)$. Тогда задача сводится к отысканию линии, проходящей через заданные две точки $M_1(x_1, y_1, z_1)$ и $M_2(x_2, y_2, z_2)$ и доставляющей экстремум функционалу (3.1). На рис. 3.1 жирной линией показана пространственная кривая — экстремаль, а тонкими — допустимые функции. Все допустимые функции удовлетворяют граничным условиям: проходят через точки $M_1(x_1, y_1, z_1)$ и $M_2(x_2, y_2, z_2)$.

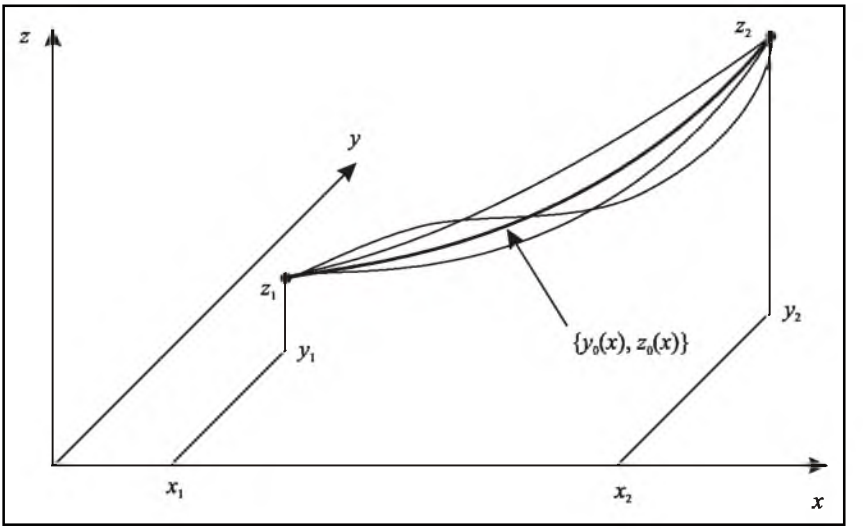


Рис. 3.1. Экстремаль и допустимые функции

ПРИМЕР 3.1. Найти экстремум функционала, зависящего от двух функций:

$$J(y,z)=\int_0^{\frac{\pi}{2}}\left(y'^2+z'^2+2yz\right)dx\to\text{extr} \tag{3.5}$$

при граничных условиях

$$\begin{cases} y(0)=0; \\ y\left(\dfrac{\pi}{2}\right)=1; \end{cases} \quad \begin{cases} z(0)=0; \\ z\left(\dfrac{\pi}{2}\right)=-1. \end{cases} \tag{3.6}$$

Составляем систему дифференциальных уравнений Эйлера (3.1):

$$\begin{cases} F_y-\dfrac{dF_{y'}}{dx}=0; \\ F_z-\dfrac{dF_{z'}}{dx}=0; \end{cases} \quad \begin{cases} 2z-\dfrac{d}{dx}(2y')=0; \\ 2y-\dfrac{d}{dx}(2z')=0; \end{cases} \quad \begin{cases} 2z-2y''=0; \\ 2y-2z''=0; \end{cases} \quad \begin{cases} y''-z=0; \\ z''-y=0. \end{cases} \tag{3.7}$$

Решаем полученную систему дифференциальных уравнений 2-го порядка путем сведения к одному дифференциальному уравнению. Дважды дифференцируем 1-е уравнение и исключаем z'' с помощью 2-го уравнения:

$$y^{IV}-y=0. \tag{3.8}$$

Составляем характеристическое уравнение и находим его корни:

$$\lambda^4-1=0; \quad \lambda_1=1; \quad \lambda_2=-1; \quad \lambda_{3,4}=\pm i. \tag{3.9}$$

В соответствии с найденными корнями записываем общее решение дифференциального уравнения (3.8):

$$y(x)=C_1\text{ch}x+C_2\text{sh}x+C_3\cos x+C_4\sin x. \tag{3.10}$$

Теперь из 1-го уравнения системы (3.7) находим $z(x)$:

$$z(x)=C_1\text{ch}x+C_2\text{sh}x-C_3\cos x-C_4\sin x. \tag{3.11}$$

Осталось найти произвольные постоянные из граничных условий (3.6). Подставляем найденные решения (3.10) и (3.11) в (3.6):

$$\begin{cases} y(0)=C_1+C_3=0; \\ y\left(\dfrac{\pi}{2}\right)=C_1\,\text{ch}\dfrac{\pi}{2}+C_2\,\text{sh}\dfrac{\pi}{2}+C_4=1; \\ z(0)=C_1-C_3=0; \\ z\left(\dfrac{\pi}{2}\right)=C_1\,\text{ch}\dfrac{\pi}{2}+C_2\,\text{sh}\dfrac{\pi}{2}-C_4=-1. \end{cases} \tag{3.12}$$

Решая совместно 1-е и 3-е уравнения (например, складывая и вычитая), получим единственное решение: $C_1 = 0$; $C_3 = 0$. Аналогично 2-е и 4-е уравнения дают также единственное решение: $C_2 = 0$; $C_4 = 1$. Таким образом, решением вариационной задачи (3.5—3.6) является экстремаль:

$$\begin{cases} y(x) = \sin x; \\ z(x) = -\sin x, \end{cases} \tag{3.13}$$

и это решение — единственное. □

3.2. Вопросы для самопроверки

1. Какую вариационную задачу мы решаем?
2. Как выводится система дифференциальных уравнений Эйлера?
3. Выведите систему дифференциальных уравнений Эйлера непосредственно, по аналогии с формулами (2.6—2.9).
4. Какие вы знаете методы решения систем дифференциальных уравнений?
5. Всегда ли вариационная задача (3.1—3.2) будет иметь решение? Всегда ли это решение будет единственным? От чего это зависит?

3.3. Пример выполнения задания

Найти экстремаль функционала, зависящего от двух функций, при заданных граничных условиях:

$$J(y,z) = \int_{-2}^{2} \left(y'^2 + z'^2 + 2yz \right) dx \to \text{extr}; \quad \begin{cases} y(-2) = 1; \\ y(2) = 0; \end{cases} \quad \begin{cases} z(-2) = 0; \\ z(2) = 2, \end{cases} \tag{3.14}$$

и построить график экстремали в виде двух функций $y(x)$, $z(x)$ и в виде пространственной кривой. Вычислить значение функционала на экстремали и какой-либо другой допустимой кривой.

Применим MATLAB для решения данной задачи. При написании будем использовать фрагменты программы решения *задания 2.1*. Вначале опишем необходимые данные и введем их.

```
clear all
disp('Решаем задание 3') % выводим заголовок задачи
syms x y z Dy D2y Dz D2z % описали переменные
F=Dy^2+Dz^2+2*y*z; % подынтегральная функция
x1=-2; % вводим граничные условия
```

```
y1=1;
z1=0;
x2=2;
y2=0;
z2=2;
disp('Исходные данные:')
fprintf('Подынтегральная функция F(x,y,y'',z,z'')=%s\n',char(F))
disp('Граничные условия слева:')
fprintf('y(%d)=%d;   z(%d)=%d\n',x1,y1,x1,z1)
disp('Граничные условия справа:')
fprintf('y(%d)=%d;   z(%d)=%d\n',x2,y2,x2,z2)
```

```
Решаем задание 3
Исходные данные:
Подынтегральная функция F(x,y,y',z,z')=Dy^2+Dz^2+2*y*z
Граничные условия слева:
y(-2)=1;   z(-2)=0
Граничные условия справа:
y(2)=0;   z(2)=2
```

Находим частные производные F_y, F_z, $F_{y'}$ и $F_{z'}$, и вычисляем полные производные $dF_{y'}/dx$ и $dF_{z'}/dx$ по правилу (2.87). Из этих величин формируем систему дифференциальных уравнений Эйлера.

```
dFdy=diff(F,y);
dFdy1=diff(F,Dy);
d_dFdy1_dx=diff(dFdy1,x);
d_dFdy1_dy=diff(dFdy1,y);
d_dFdy1_dy1=diff(dFdy1,Dy);
d_dFdy1_dz=diff(dFdy1,z);
d_dFdy1_dz1=diff(dFdy1,Dz);
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+...
  d_dFdy1_dy1*D2y+d_dFdy1_dz*Dz+d_dFdy1_dz1*D2z;
dFdz=diff(F,z);
dFdz1=diff(F,Dz);
d_dFdz1_dx=diff(dFdz1,x);
d_dFdz1_dy=diff(dFdz1,y);
d_dFdz1_dy1=diff(dFdz1,Dy);
d_dFdz1_dz=diff(dFdz1,z);
d_dFdz1_dz1=diff(dFdz1,Dz);
dFz1dx=d_dFdz1_dx+d_dFdz1_dy*Dy+...
  d_dFdz1_dy1*D2y+d_dFdz1_dz*Dz+d_dFdz1_dz1*D2z;
EulerY=simple(dFdy-dFy1dx);
EulerZ=simple(dFdz-dFz1dx);
dEuY=[char(EulerY) '=0']; % уравнение Y
```

```
dEuZ=[char(EulerZ) '=0']; % уравнение Z
fprintf('Система уравнений Эйлера:\n%s\n%s\n',dEuY,dEuZ)
```
```
Система уравнений Эйлера:
2*z-2*D2y=0
2*y-2*D2z=0
```

Решаем систему уравнений Эйлера, проверяем существование и единственность решения, печатаем аналитические решения.

```
Sol=dsolve(dEuY,dEuZ,'x'); % решаем
if length(Sol)~=1 % нет решений или более одного решения
  error('Нет решений или более одного решения!');
else
  SolY=Sol.y;
  SolZ=Sol.z;
  disp('Общее решение системы уравнений Эйлера:')
  fprintf('y(x)=%s\nz(x)=%s\n',char(SolY),char(SolZ))
end
```
```
Общее решение системы уравнений Эйлера:
y(x)=1/4*C1*exp(-x)+1/4*C1*exp(x)+1/2*C1*cos(x)+1/2*C2*sin(x)+
1/4*C2*exp(x)-1/4*C2*exp(-x)-1/2*C3*cos(x)+1/4*C3*exp(x)+1/4*C3*exp(-x)-
1/2*C4*sin(x)+1/4*C4*exp(x)-1/4*C4*exp(-x)
z(x)=-1/2*C1*cos(x)+1/4*C1*exp(x)+1/4*C1*exp(-x)-1/2*C2*sin(x)+
1/4*C2*exp(x)-1/4*C2*exp(-x)+1/4*C3*exp(-x)+1/4*C3*exp(x)+1/2*C3*cos(x)+
1/2*C4*sin(x)+1/4*C4*exp(x)-1/4*C4*exp(-x)
```

Подставляем в полученное решение граничные условия, формируем систему уравнений для нахождения произвольных постоянных. Печатаем ее. При подстановке сразу ограничиваем точность 14 знаками.

```
SolLY=subs(SolY,x,x1); % x1 в y
SolLZ=subs(SolZ,x,x1); % x1 в z
SolRY=subs(SolY,x,x2); % x2 в y
SolRZ=subs(SolZ,x,x2); % x2 в z
EqLY=[char(vpa(SolLY,14)) '=' char(sym(y1))];
EqLZ=[char(vpa(SolLZ,14)) '=' char(sym(z1))];
EqRY=[char(vpa(SolRY,14)) '=' char(sym(y2))];
EqRZ=[char(vpa(SolRZ,14)) '=' char(sym(z2))];
disp('Граничные условия:')
fprintf('%s\n',EqLY,EqLZ,EqRY,EqRZ)
```
```
Граничные условия:
1.6730244272683*C1-2.2680789173363*C2+
2.0891712638155*C3-1.3587814905107*C4=1
2.0891712638155*C1-1.3587814905107*C2+
1.6730244272683*C3-2.2680789173363*C4=0
```

```
1.6730244272683*C1+2.2680789173364*C2+
2.0891712638154*C3+1.3587814905107*C4=0
2.0891712638154*C1+1.3587814905107*C2+
1.6730244272683*C3+2.2680789173364*C4=2
```

Решаем полученную систему уравнений и находим из ее решения произвольные постоянные. Найденные константы подставляем в решение. Печатаем полученные уравнения экстремали.

```
Con=solve(EqLY,EqLZ,EqRY,EqRZ,'C1,C2,C3,C4');
C1=Con.C1;
C2=Con.C2;
C3=Con.C3;
C4=Con.C4;
Sol3Y=vpa(eval(SolY),14);
Sol3Z=vpa(eval(SolZ),14);
disp('Уравнения экстремали:')
fprintf('y(x)=%s\nz(x)=%s\n',char(Sol3Y),char(Sol3Z))
Уравнения экстремали:
y(x)=.65210765216302e-1*exp(-x)+.13414090640925*exp(x)+
.60074949043049*cos(x)-.82481262772102*sin(x)
z(x)=-.60074949043049*cos(x)+.13414090640925*exp(x)+
.65210765216302e-1*exp(-x)+.82481262772102*sin(x)
```

Вычисляем значения функционала (3.14) на найденной экстремали и для сравнения на прямой, соединяющей точки M_1 и M_2. Для этого подставляем в подынтегральную функцию F аналитические выражения для этих линий и их производных, упрощаем, а затем интегрируем и вычисляем значение интеграла. Печатаем результаты.

```
Fextr=simple(subs(F,{y,z,Dy,Dz},...
  {Sol3Y,Sol3Z,diff(Sol3Y,x),diff(Sol3Z,x)}));
Jextr=eval(int(Fextr,x,x1,x2))
ylin=(x-x1)*(y2-y1)/(x2-x1)+y1;
zlin=(x-x1)*(z2-z1)/(x2-x1)+z1;
Flin=simple(subs(F,{y,z,Dy,Dz},...
  {ylin,zlin,diff(ylin,x),diff(zlin,x)}));
Jlin=eval(int(Flin,x,x1,x2))
Jextr =
   1.94492120385672
Jlin =
   3.91666666666667
```

Задаем значения аргументов и вычисляем функции. Вначале рисуем на одном рисунке графики функций: $y(x)$ красной сплошной линией и $z(x)$ синей штри-

ховой — это рис. 3.2. Затем на другом рисунке (рис. 3.3) изображаем трехмерный график полученной пространственной линии. Выбираем точку просмотра. Показываем сетку и ограничивающий параллелепипед (контур).

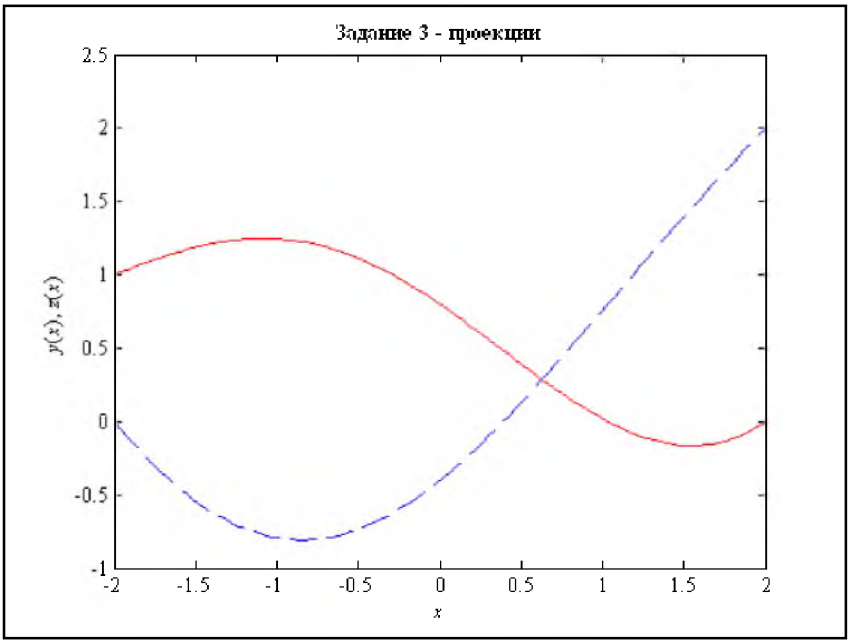


Рис. 3.2. Графики функций $y(x)$ и $z(x)$ в задании 3

```
xpl=linspace(x1,x2); % массив абсцисс
y3=subs(Sol3Y,x,xpl); % ординаты
z3=subs(Sol3Z,x,xpl); % аппликаты
figure % фигура
plot(xpl,y3,'-r',xpl,z3,'--b') % рисуем график
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 3 - проекции') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm), \itz\rm(\itx\rm)')
figure % фигура
plot3(xpl,y3,z3,'-r') % рисуем 3D-график
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 3 - трехмерный график') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
```

```
zlabel('\itz\rm(\itx\rm)') % метка оси OZ
view(205,30) % выбрали точку просмотра
grid on % показали сетку
box on % показали внешний контур
```

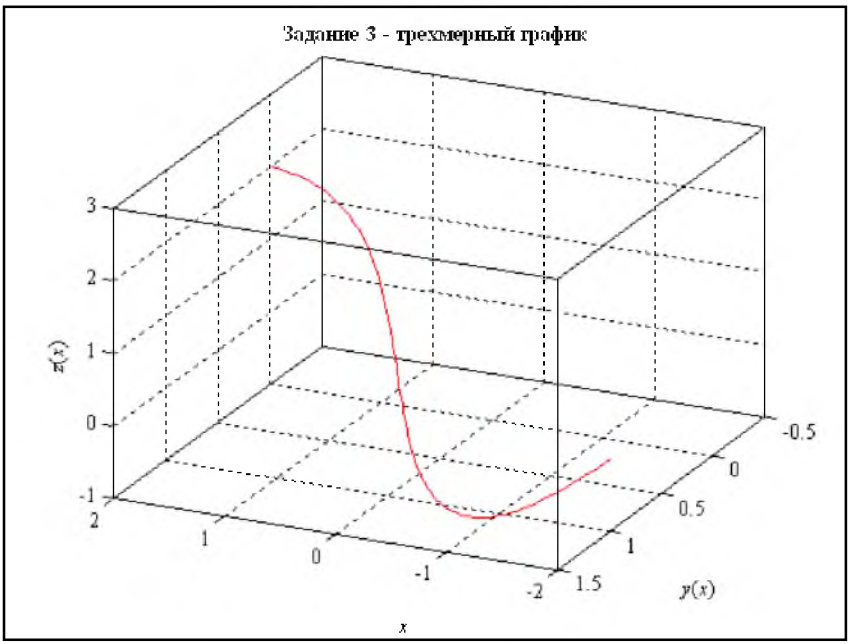
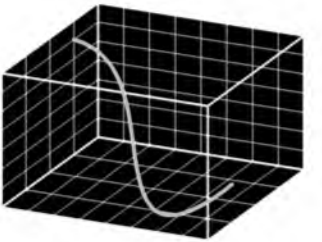


Рис. 3.3. График пространственной кривой в задании 3

3.4. Задание

Для своего варианта функционала найти экстремаль и построить ее графики: проекции и трехмерный. Вычислить значение функционала на экстремали и какой-либо другой допустимой функции. Варианты заданий имеются на компакт-диске.

ГЛАВА 4



Функционалы, зависящие от производных высших порядков

4.1. Дифференциальное уравнение Эйлера — Пуассона

Рассмотрим теперь функционал, зависящий от функции одной переменной и ее производных 1-го и 2-го порядка

$$J(y)=\int_{x_1}^{x_2} F(x,y,y',y'')\,dx \to \text{extr} \tag{4.1}$$

с граничными условиями вида

$$\begin{cases} y(x_1)=y_1; \\ y(x_2)=y_2; \end{cases} \quad \begin{cases} y'(x_1)=y_1'; \\ y'(x_2)=y_2'. \end{cases} \tag{4.2}$$

Посмотрите внимательно на формулу (4.2): в отличие от *главы 2*, здесь для искомой функции заданы не только ее значения на концах интервала, но и первые производные в граничных точках. Это значит, что класс допустимых функций здесь более узкий, чем на рис. 2.1: нужно, чтобы все допустимые функции имели на концах отрезка $[x_1, x_2]$ не только заданные значения, но и имели заданный угол наклона касательной в этих точках. На рис. 4.1 показаны экстремаль (жирная линия), допустимые функции (тонкие сплошные линии) и несколько недопустимых функций (штриховые линии). В недопустимых функциях какое-либо из граничных условий (4.2) нарушается.

Как и для других задач, необходимым условием экстремума функционала (4.1) является равенство нулю его вариации, вычисленной на экстремали $y_0(x)$: $\delta J(y_0)=0$. Для вывода вариации вначале запишем приращение нашего

функционала на экстремали y_0. Оно вызывается вариациями функции $y_0(x)$, ее первой и второй производных $y'_0(x)$ и $y''_0(x)$:

$$\Delta J\left(y_0\right)=\int_{x_1}^{x_2}\left(F\left(x,y_0+\delta y_0,y_0'+\delta y_0',y_0''+\delta y_0''\right)-F\left(x,y_0,y_0',y_0''\right)\right)dx. \quad (4.3)$$

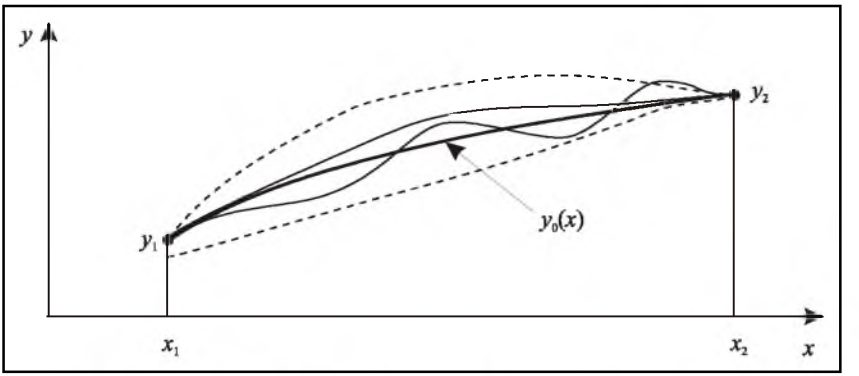


Рис. 4.1. Экстремаль $y_0(x)$, допустимые и недопустимые функции

Воспользуемся первым способом вычисления вариации функционала: вычислим δJ как линейную часть его приращения. Разложим первое слагаемое формулы (4.3) в ряд Тейлора в окрестности экстремали и удержим только линейные члены. Это разложение отличается от формул (2.6—2.7) наличием дополнительного слагаемого $F_{y''}\delta y''$ и тем, что функция F и частные производные от нее зависят еще и от аргумента y''_0. Получим вариацию $\delta J(y_0)$:

$$\delta J\left(y_0\right)=\int_{x_1}^{x_2}\left(F_y\,\delta y+F_{y'}\,\delta y'+F_{y''}\,\delta y''\right)dx\,. \quad (4.4)$$

Теперь слагаемое, содержащее 1-ю производную, проинтегрируем по частям один раз, а слагаемое, содержащее 2-ю производную, — 2 раза. При интегрировании по частям учтем, что в силу граничных условий на концах интервала $\delta y(x_1)=\delta y(x_2)=\delta y'(x_1)=\delta y'(x_2)=0$. Второе слагаемое интегрируем, как в формуле (2.8):

$$\int_{x_1}^{x_2}F_{y'}\,\delta y'\,dx=\left(F_{y'}\,\delta y\right)\Big|_{x_1}^{x_2}-\int_{x_1}^{x_2}\frac{dF_{y'}}{dx}\delta y\,dx=-\int_{x_1}^{x_2}\frac{dF_{y'}}{dx}\delta y\,dx\,. \quad (4.5)$$

Третье слагаемое дважды интегрируем по частям:

$$\int_{x_1}^{x_2} F_{y''}\,\delta y''\,dx = \left(F_{y''}\,\delta y'\right)\Big|_{x_1}^{x_2} - \int_{x_1}^{x_2} \frac{dF_{y''}}{dx}\delta y'\,dx = -\int_{x_1}^{x_2} \frac{dF_{y''}}{dx}\delta y'\,dx =$$
$$= -\left(\frac{dF_{y''}}{dx}\delta y\right)\Big|_{x_1}^{x_2} + \int_{x_1}^{x_2} \frac{d^2F_{y''}}{dx^2}\delta y\,dx = \int_{x_1}^{x_2} \frac{d^2F_{y''}}{dx^2}\delta y\,dx\,. \tag{4.6}$$

В итоге из (4.4) получаем:

$$\delta J(y_0) = \int_{x_1}^{x_2}\left(F_y - \frac{dF_{y'}}{dx} + \frac{d^2F_{y''}}{dx^2}\right)\delta y\,dx = 0. \tag{4.7}$$

В силу произвольности вариации функции $\delta y(x)$ по основной лемме вариационного исчисления первый сомножитель под интегралом должен равняться нулю. Таким образом, экстремаль должна удовлетворять уравнению

$$F_y - \frac{dF_{y'}}{dx} + \frac{d^2F_{y''}}{dx^2} = 0. \tag{4.8}$$

Определение 4.1. Уравнение (4.8) называется *дифференциальным уравнением Эйлера — Пуассона* (Simeon-Denis Poisson, 1781–1840, рис. 4.2). □

Рис. 4.2. С.-Д. Пуассон

Оно является в общем случае уравнением 4-го порядка: функция F и ее частные производные зависят от x, y, y' и y'', поэтому вторая полная производная по x будет уже включать 4-ю производную. Дополняется это уравнение четырьмя граничными условиями (4.2).

Выведем теперь уравнение Эйлера — Пуассона, если функционал зависит от функции и ее производных до n-го порядка включительно:

$$J(y)=\int_{x_1}^{x_2} F\left(x,y,y',y'',y''',\ldots,y^{(n)}\right)dx\to \text{extr} \tag{4.9}$$

и дополняется $2n$ граничными условиями вида:

$$\begin{cases} y(x_1)=y_1; \\ y(x_2)=y_2; \end{cases} \quad \begin{cases} y'(x_1)=y_1'; \\ y'(x_2)=y_2'; \end{cases} \quad \ldots \quad \begin{cases} y^{(n-1)}(x_1)=y_1^{(n-1)}; \\ y^{(n-1)}(x_2)=y_2^{(n-1)}. \end{cases} \tag{4.10}$$

Здесь в граничных точках должны быть заданы значения функции и ее производных до $(n-1)$-го порядка включительно.

Уравнение Эйлера — Пуассона для задачи (4.9—4.10) выводится аналогично. После разложения приращения функционала в ряд Тейлора и удержания линейных слагаемых линейная часть приращения функционала будет иметь вид:

$$\delta J(y_0)=\int_{x_1}^{x_2}\left(F_y\,\delta y+F_{y'}\,\delta y'+F_{y''}\,\delta y''+F_{y'''}\delta y'''+\ldots+F_{y^{(n)}}\delta y^{(n)}\right)dx\,. \tag{4.11}$$

Теперь, как и при выводе формул (4.5—4.8), интегрируем по частям: первое слагаемое — один раз, второе — два, ..., n-е — n раз. Все внеинтегральные слагаемые в силу граничных условий (4.10) обратятся в 0, и мы по основной лемме вариационного исчисления получим уравнение Эйлера — Пуассона:

$$F_y-\frac{dF_{y'}}{dx}+\frac{d^2F_{y''}}{dx^2}-\frac{d^3F_{y'''}}{dx^3}+\ldots+(-1)^n\frac{d^nF_{y^{(n)}}}{dx^n}=0. \tag{4.12}$$

Это уравнение порядка $2n$, оно дополняется $2n$ граничными условиями (4.10): значения искомой функции и ее производных до $(n-1)$-го порядка включительно на концах интервала x_1 и x_2 должны равняться заданным величинам.

ПРИМЕР 4.1. Найти экстремаль функционала

$$J(y)=\int_0^{\frac{\pi}{2}}\left(y''^2-y^2+x^2\right)dx \tag{4.13}$$

при заданных граничных условиях

$$\begin{cases} y(0)=1; \\ y\left(\dfrac{\pi}{2}\right)=0; \end{cases} \quad \begin{cases} y'(0)=0; \\ y'\left(\dfrac{\pi}{2}\right)=-1. \end{cases} \tag{4.14}$$

Составляем дифференциальное уравнение Эйлера — Пуассона вида (4.8):

$$F_y - \frac{dF_{y'}}{dx} + \frac{d^2F_{y''}}{dx^2} = -2y - \frac{d}{dx}(0) + \frac{d^2}{dx^2}(2y'') = -2y + 2y^{IV} = 0. \qquad (4.15)$$

Упрощаем его:

$$y^{IV} - y = 0. \qquad (4.16)$$

Составляем характеристическое уравнение и находим его корни:

$$\lambda^4 - 1 = 0; \quad \lambda_1 = 1; \quad \lambda_2 = -1; \quad \lambda_{3,4} = \pm i. \qquad (4.17)$$

В соответствии с найденными корнями записываем общее решение дифференциального уравнения (4.16):

$$y(x) = C_1\mathrm{ch}x + C_2\mathrm{sh}x + C_3\cos x + C_4\sin x. \qquad (4.18)$$

Находим произвольные постоянные из граничных условий (4.14). Подставляем их в решение (4.18) и его производную:

$$\begin{cases} y(0) = C_1 + C_3 = 1; \\ y\left(\dfrac{\pi}{2}\right) = C_1\,\mathrm{ch}\dfrac{\pi}{2} + C_2\,\mathrm{sh}\dfrac{\pi}{2} + C_4 = 0; \\ y'(0) = C_2 + C_4 = 0; \\ y'\left(\dfrac{\pi}{2}\right) = C_1\,\mathrm{sh}\dfrac{\pi}{2} + C_2\,\mathrm{ch}\dfrac{\pi}{2} - C_3 = -1. \end{cases} \qquad (4.19)$$

Из 1-го и 3-го уравнений выражаем C_3 и C_4:

$$\begin{cases} C_3 = 1 - C_1; \\ C_4 = -C_2. \end{cases} \qquad (4.20)$$

Подставляем их во 2-е и 4-е уравнения:

$$\begin{cases} C_1\,\mathrm{ch}\dfrac{\pi}{2} + C_2\,\mathrm{sh}\dfrac{\pi}{2} - C_2 = 0; \\ C_1\,\mathrm{sh}\dfrac{\pi}{2} + C_2\,\mathrm{ch}\dfrac{\pi}{2} - 1 + C_1 = -1. \end{cases} \qquad (4.21)$$

После очевидных упрощений имеем однородную систему линейных алгебраических уравнений относительно C_1 и C_2:

$$\begin{cases} C_1\,\mathrm{ch}\dfrac{\pi}{2} + C_2\left(\mathrm{sh}\dfrac{\pi}{2} - 1\right) = 0; \\ C_1\left(\mathrm{sh}\dfrac{\pi}{2} + 1\right) + C_2\,\mathrm{ch}\dfrac{\pi}{2} = 0, \end{cases} \qquad (4.22)$$

главный определитель которой

$$\Delta = \begin{vmatrix} \operatorname{ch}\frac{\pi}{2} & \operatorname{sh}\frac{\pi}{2}-1 \\ \operatorname{sh}\frac{\pi}{2}+1 & \operatorname{ch}\frac{\pi}{2} \end{vmatrix} = \operatorname{ch}^2\frac{\pi}{2} - \left(\operatorname{sh}^2\frac{\pi}{2}-1\right) = 2 \neq 0, \tag{4.23}$$

поэтому система (4.22) имеет единственное решение: $C_1 = 0$; $C_2 = 0$. Отсюда находим из (4.20): $C_3 = 1$; $C_4 = 0$; уравнение экстремали будет:

$$y(x) = \cos x, \tag{4.24}$$

и это решение — единственное. □

4.2. Вопросы для самопроверки

1. Какую вариационную задачу мы решаем?
2. Каким является класс допустимых функций в данной задаче: более узким или более широким, чем в задаче (2.1)?
3. Почему обращаются в нуль внеинтегральные слагаемые в выводе (4.5—4.7)?
4. Какие вы знаете методы решения дифференциальных уравнений высших порядков?
5. Всегда ли вариационные задачи (4.1—4.2) и (4.9—4.10) будут иметь решение? Всегда ли это решение будет единственным? От чего это зависит?

4.3. Пример выполнения задания

Найти экстремаль функционала при заданных граничных условиях:

$$J(y) = \int_{-1}^{1} \left(y''^2 - 2y'^2 + 4yy' + y^2 - 2y\sin x\right) dx; \quad \begin{cases} y(-1) = 1; \\ y'(-1) = -1; \end{cases} \quad \begin{cases} y(1) = 2; \\ y'(1) = -1. \end{cases} \tag{4.25}$$

Применим для решения задачи MATLAB. Вначале вводим исходные данные.

```
clear all
disp('Решаем задание 4') % выводим заголовок задачи
syms x y Dy D2y D3y D4y % описали переменные
F=D2y^2-2*Dy^2+4*y*Dy+y^2-2*y*sin(x);
```

```
x1=-1;
y1=1;
Dy1=-1;
x2=1;
y2=2;
Dy2=-1;
disp('Исходные данные:')
disp('Подынтегральная функция:')
fprintf('F(x,y,y'',y'''')=%s\n',char(F))
disp('Граничные условия слева:')
fprintf('y(%d)=%d;   y''(%d)=%d\n',x1,y1,x1,Dy1)
disp('Граничные условия справа:')
fprintf('y(%d)=%d;   y''(%d)=%d\n',x2,y2,x2,Dy2)
Решаем задание 4
Исходные данные:
Подынтегральная функция:
F(x,y,y',y'')=D2y^2-2*Dy^2+4*y*Dy+y^2-2*y*sin(x)
Граничные условия слева:
y(-1)=1;   y'(-1)=-1
Граничные условия справа:
y(1)=2;   y'(1)=-1
```

Для вывода дифференциального уравнения Эйлера — Пуассона нам надо сформировать полные производные $dF_{y'}/dx$ и $d^2F_{y''}/dx^2$. Формируем их с использованием формулы (2.87). Вначале находим частные производные F_y, $F_{y'}$ и $F_{y''}$, а затем строим по ним $dF_{y'}/dx$.

```
dFdy=diff(F,y);
dFdy1=diff(F,Dy);
dFdy2=diff(F,D2y);
d_dFdy1_dx=diff(dFdy1,x);
d_dFdy1_dy=diff(dFdy1,y);
d_dFdy1_dy1=diff(dFdy1,Dy);
d_dFdy1_dy2=diff(dFdy1,D2y);
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+d_dFdy1_dy1*D2y+d_dFdy1_dy2*D3y
dFy1dx =
4*Dy-4*D2y
```

Далее находим по такому же принципу $d^2F_{y''}/dx^2$.

```
d_dFdy2_dx=diff(dFdy2,x);
d_dFdy2_dy=diff(dFdy2,y);
d_dFdy2_dy1=diff(dFdy2,Dy);
d_dFdy2_dy2=diff(dFdy2,D2y);
dFy2dx=d_dFdy2_dx+d_dFdy2_dy*Dy+d_dFdy2_dy1*D2y+d_dFdy2_dy2*D3y;
```

```
d_dFdy2dx_dx=diff(dFy2dx,x);
d_dFdy2dx_dy=diff(dFy2dx,y);
d_dFdy2dx_dy1=diff(dFy2dx,Dy); % d((dFy'')/dx)/dy'
d_dFdy2dx_dy2=diff(dFy2dx,D2y); % d((dFy'')/dx)/dy''
d_dFdy2dx_dy3=diff(dFy2dx,D3y); % d((dFy'')/dx)/dy'''
d2Fy2dx2=d_dFdy2dx_dx+d_dFdy2dx_dy*Dy+d_dFdy2dx_dy1*D2y;
d2Fy2dx2=d2Fy2dx2+d_dFdy2dx_dy2*D3y+d_dFdy2dx_dy3*D4y
d2Fy2dx2 =
2*D4y
```

Формируем из этих данных уравнение Эйлера — Пуассона.

```
Euler=simple(dFdy-dFy1dx+d2Fy2dx2);
deqEuler=[char(Euler) '=0']; % составили уравнение
fprintf('Уравнение Эйлера-Пуассона:\n%s\n',deqEuler)
Уравнение Эйлера-Пуассона:
2*y-2*sin(x)+4*D2y+2*D4y=0
```

Находим общее решение, проверяем существование и единственность.

```
Sol=dsolve(deqEuler,'x'); % решаем уравнение
if length(Sol)~=1 % нет решений или более одного решения
  error('Нет решений или более одного решения!');
else
  disp('Общее решение уравнения Эйлера-Пуассона:')
  fprintf('y(x)=%s\n',char(Sol))
end
Общее решение уравнения Эйлера-Пуассона:
y(x)=-1/8*sin(x)*x^2-1/4*cos(x)*x+1/4*sin(x)+
C1*sin(x)+C2*cos(x)+C3*sin(x)*x+C4*cos(x)*x
```

Вычисляем производную от полученного решения — она необходима для формирования граничных условий. Вычисляем произвольные постоянные и находим частное решение.

```
dydx=diff(Sol,x); % нашли производную
slY=subs(Sol,x,x1); % подставили x1 в y(x)
slDY=subs(dydx,x,x1); % подставили x1 в y'(x)
srY=subs(Sol,x,x2); % подставили x2 в y(x)
srDY=subs(dydx,x,x2); % подставили x2 в y'(x)
elY=[char(vpa(slY,14)) '=' char(sym(y1))];
elDY=[char(vpa(slDY,14)) '=' char(sym(Dy1))];
erY=[char(vpa(srY,14)) '=' char(sym(y2))];
erDY=[char(vpa(srDY,14)) '=' char(sym(Dy2))];
disp('Граничные условия:')
fprintf('%s\n',elY,elDY,erY,erDY)
```

```
Con=solve(elY,elDY,erY,erDY,'C1,C2,C3,C4');
C1=Con.C1;
C2=Con.C2;
C3=Con.C3;
C4=Con.C4;
Sol4=vpa(eval(Sol),14); % подставили C1-C4
disp('Уравнение экстремали:')
fprintf('y(x)=%s\n',char(Sol4))
```

```
Граничные условия:
.2989170336605e-1-.84147098480790*C1+
.54030230586814*C2+.84147098480790*C3-.54030230586814*C4=1
-.67537788233518e-1+.54030230586814*C1+
.84147098480790*C2-1.381773290676O*C3-.30116867893976*C4=-1
-.2989170336605e-1+.84147098480790*C1+
.54030230586814*C2+.84147098480790*C3+.54030230586814*C4=2
-.67537788233518e-1+.54030230586814*C1-
.84147098480790*C2+1.381773290676O*C3-.30116867893976*C4=-1
Уравнение экстремали:
y(x)=-.12500000000000*sin(x)*x^2+
1.7137646983880*cos(x)*x-.38119810539482*sin(x)+
1.4248525550552*cos(x)+.86770535427108*sin(x)*x
```

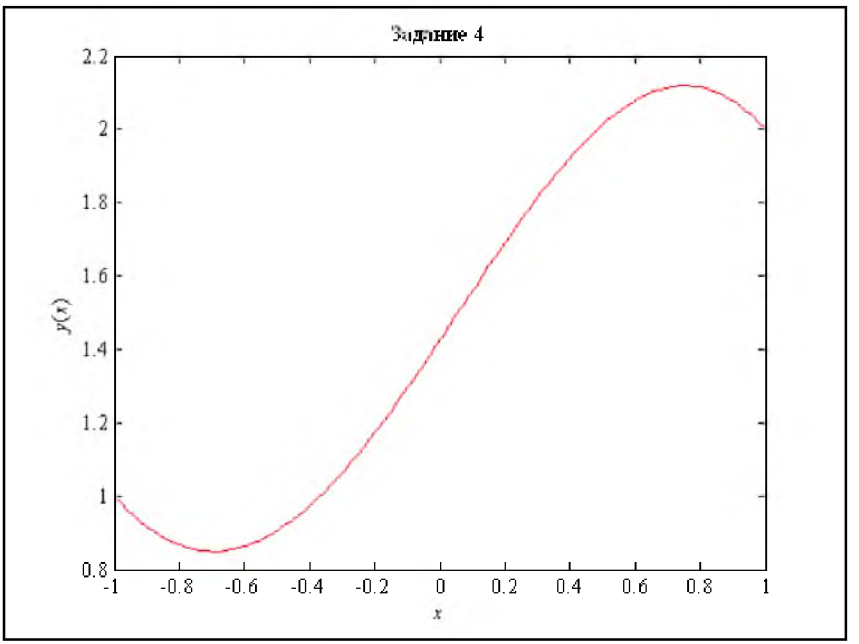


Рис. 4.3. Экстремаль в задании 4

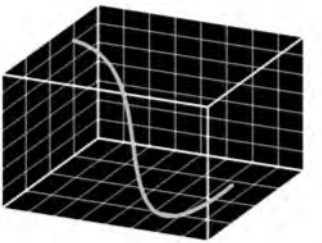
Задаем массив аргументов, вычисляем функцию и рисуем ее. Результат показан на рис. 4.3.

```
xpl=linspace(x1,x2); % массив аргументов
y4=subs(Sol4,x,xpl); % вычислили функцию
figure % фигура
plot(xpl,y4,'-r') % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 4') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
```

4.4. Задание

Для своего варианта функционала найти экстремаль и построить ее график. Найти значение функционала на экстремали и какой-либо близкой допустимой функции. Для этой последней задачи дописать соответствующий фрагмент программы самостоятельно. Посмотрите в *главах 2 и 3*, как это сделать. Варианты заданий есть на компакт-диске.

ГЛАВА 5



Функционалы, зависящие от функции нескольких переменных

5.1. Дифференциальное уравнение Эйлера — Остроградского

Исследуем на экстремум функционал, зависящий от функции двух переменных и ее частных производных 1-го порядка:

$$J\left(z\left(x,y\right)\right)=\iint\limits_{D}F\left(x,y,z,\frac{\partial z}{\partial x},\frac{\partial z}{\partial y}\right)dS\to\text{extr} \tag{5.1}$$

с заданными условиями на контуре C — границе области D:

$$z(x,y)|_{C}=z_{C}(x,\,y). \tag{5.2}$$

Будем далее обозначать $p=\partial z/\partial x$, $q=\partial z/\partial y$. Необходимым условием экстремума функционала является равенство нулю его вариации, вычисленной на экстремали $z_0(x,y)$: $\delta J(z_0)=0$. Найдем эту вариацию как линейную часть приращения функционала. Она вызывается вариациями функций z, p и q, причем на контуре C: $\delta z=0$. После разложения функции $F(x,y,z,p,q)$ в ряд Тейлора в окрестности экстремали и удержания линейных членов вариация функционала на экстремали имеет вид

$$\delta J\left(z\right)=\iint\limits_{D}\left(F_{z}\delta z+F_{p}\delta p+F_{q}\delta q\right)dS. \tag{5.3}$$

В *главах 2 и 4* мы преобразовывали все слагаемые, кроме первого, с помощью интегрирования по частям. Здесь этот прием применить не удается, т. к. у нас двойной интеграл. Однако мы можем использовать формулу Грина, дающую тот же результат.

Заметим прежде всего, что

$$\frac{\partial}{\partial x}\left(F_p\delta z\right)=\frac{\partial F_p}{\partial x}\delta z+F_p\delta p;\quad \frac{\partial}{\partial y}\left(F_q\delta z\right)=\frac{\partial F_q}{\partial y}\delta z+F_q\delta q. \tag{5.4}$$

Здесь $\partial F_p/\partial x$ и $\partial F_q/\partial y$ — так называемые "полные частные производные", т. е. частные производные, вычисляемые при условии, что в дифференцируемой функции сделаны подстановки: $z=z(x,y)$, $p=p(x,y)$, $q=q(x,y)$. Подставив (5.4) в (5.3), получим:

$$\delta J(z)=\iint\limits_D\left(\frac{\partial}{\partial x}\left(F_p\delta z\right)+\frac{\partial}{\partial y}\left(F_q\delta z\right)+F_z\delta z-\frac{\partial F_p}{\partial x}\delta z-\frac{\partial F_q}{\partial y}\delta z\right)dS. \tag{5.5}$$

Вычислим интеграл от первых двух слагаемых по формуле Грина, положив $Q(x,y)=F_p\delta z$, $P(x,y)=-F_q\delta z$.

$$\begin{aligned}&\iint\limits_D\left(\frac{\partial}{\partial x}\left(F_p\delta z\right)+\frac{\partial}{\partial y}\left(F_q\delta z\right)\right)dS=\iint\limits_D\left(\frac{\partial Q}{\partial x}-\frac{\partial P}{\partial y}\right)dS=\\&=\oint\limits_C P\,dx+Q\,dy=\oint\limits_C F_p\delta z\,dy-F_q\delta z\,dx=0,\end{aligned} \tag{5.6}$$

т. к. на контуре C — границе области D: $\delta z=0$. Таким образом, в интеграле (5.5) остаются только три последних слагаемых:

$$\delta J(z)=\iint\limits_D\left(F_z-\frac{\partial F_p}{\partial x}-\frac{\partial F_q}{\partial y}\right)\delta z\,dS. \tag{5.7}$$

Необходимое условие экстремума — равенство нулю этой вариации. В силу произвольности вариации функции $\delta z(x,y)$ по основной лемме вариационного исчисления должен быть равен нулю множитель при δz в подынтегральной функции:

$$F_z-\frac{\partial F_p}{\partial x}-\frac{\partial F_q}{\partial y}=0. \tag{5.8}$$

Определение 5.1. Уравнение (5.8) называется *дифференциальным уравнением Эйлера — Остроградского* (см. рис. 1.11). □

Это дифференциальное уравнение в частных производных, оно дополняется граничным условием (5.2).

Замечание 5.1. Если функционал зависит от функции n переменных $z(x_1, x_2, ..., x_n)$ и ее первых частных производных, то уравнение Эйлера — Остроградского будет иметь вид

$$F_z - \sum_{k=1}^{n} \frac{\partial F_{p_k}}{\partial x_k} = 0, \tag{5.9}$$

где $p_k = \partial z / \partial x_k$. При выводе этого уравнения используются многомерные аналоги формулы Грина: формулы Стокса и Остроградского — Гаусса. □

Замечание 5.2. Если функционал зависит от функции двух переменных $z(x, y)$ и ее частных производных до n-го порядка включительно, то уравнение Эйлера — Остроградского — Пуассона будет иметь вид

$$F_z - \frac{\partial F_{z_x}}{\partial x} - \frac{\partial F_{z_y}}{\partial y} + \frac{\partial^2 F_{z_{xx}}}{\partial x^2} + \frac{\partial^2 F_{z_{xy}}}{\partial x \partial y} + \frac{\partial^2 F_{z_{yy}}}{\partial y^2} - \quad \square$$
$$-\ldots + (-1)^n \frac{\partial^n F_{z_{yy\ldots y}}}{\partial y^n} = 0. \tag{5.10}$$

Замечание 5.3. И наконец, если функционал будет зависеть от нескольких функций нескольких переменных (и их частных производных), то мы будем уже иметь систему дифференциальных уравнений Эйлера — Остроградского (и, возможно, Пуассона). □

ПРИМЕР 5.1. Найти экстремаль функционала:

$$J\left(z(x,y)\right) = \iint\limits_{(D)} \left(\left(\frac{\partial z}{\partial x}\right)^2 + \left(\frac{\partial z}{\partial y}\right)^2 + 2zx \sin\frac{\pi y}{b} \right) dS \tag{5.11}$$

в прямоугольной области $x \in [0, a]$; $y \in [0, b]$, показанной на рис. 5.1.

Граничные условия: на правой стороне $x = a$:

$$z(x,y)\big|_{x=a} = \sin\frac{\pi y}{b}, \tag{5.12}$$

на остальных сторонах $z = 0$.

Выведем вначале уравнение Эйлера — Остроградского вида (5.8):

$$F_z - \frac{\partial F_p}{\partial x} - \frac{\partial F_q}{\partial y} = 2x \sin\frac{\pi y}{b} - 2\frac{\partial}{\partial x}\left(\frac{\partial z}{\partial x}\right) - 2\frac{\partial}{\partial y}\left(\frac{\partial z}{\partial y}\right) =$$
$$= 2x \sin\frac{\pi y}{b} - 2\frac{\partial^2 z}{\partial x^2} - 2\frac{\partial^2 z}{\partial y^2} = 0, \tag{5.13}$$

или, после сокращения на 2:

$$\frac{\partial^2 z}{\partial x^2}+\frac{\partial^2 z}{\partial y^2}=x\sin\frac{\pi y}{b}. \tag{5.14}$$

Граничные условия по переменной y однородные, поэтому будем искать решение в виде ряда Фурье по собственным функциям $Y_n(y)$, которые равны:

$$Y_n(y)=\sin\frac{n\pi y}{b}. \tag{5.15}$$

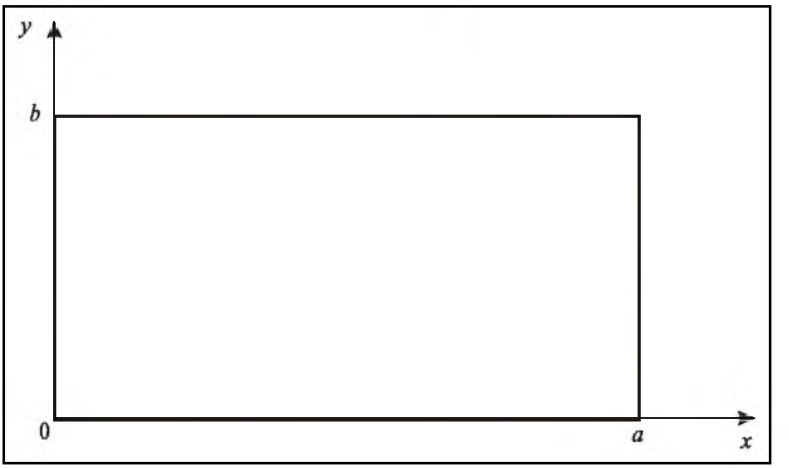


Рис. 5.1. Область решения примера 5.1

Ищем решение в виде ряда:

$$u(x,y)=\sum_{n=1}^{\infty}X_n(x)\sin\frac{n\pi y}{b}. \tag{5.16}$$

Это решение удовлетворяет граничным условиям на нижней и верхней сторонах: при $y=0$ и $y=b$. Для нахождения функций $X_n(x)$ подставим решение (5.16) в уравнение (5.14):

$$\begin{aligned}&\sum_{n=1}^{\infty}X_n''(x)\sin\frac{n\pi y}{b}-\sum_{n=1}^{\infty}\frac{n^2\pi^2}{b^2}X_n(x)\sin\frac{n\pi y}{b}=\\&=\sum_{n=1}^{\infty}\left(X_n''(x)-\frac{n^2\pi^2}{b^2}X_n(x)\right)\sin\frac{n\pi y}{b}=x\sin\frac{\pi y}{b}.\end{aligned} \tag{5.17}$$

Левая часть (5.17) — это разложение в ряд Фурье по $Y_n(y)$. Разложим в такой же ряд и правую часть. Собственно, она уже разложена: в этом ряде присут-

ствует только первый член ($n=1$), а коэффициенты при остальных гармониках равны нулю. Мы знаем, что два ряда Фурье тождественно равны друг другу тогда и только тогда, когда равны все их коэффициенты. Поэтому из (5.17) получаем бесконечную систему дифференциальных уравнений для функций $X_n(x)$:

$$\begin{cases} X_1''(x) - \dfrac{\pi^2}{b^2} X_1(x) = x; \\ X_n''(x) - \dfrac{n^2\pi^2}{b^2} X_n(x) = 0; \quad n > 1. \end{cases} \tag{5.18}$$

Граничные условия для (5.18) получим, раскладывая граничные условия на левой и правой сторонах в ряд Фурье по $Y_n(y)$. На левой стороне $x=0$ мы имеем $z=0$, коэффициенты разложения этой функции — нулевые:

$$X_n(0) = 0 \qquad \forall n. \tag{5.19}$$

На правой стороне $x=a$ у нас есть граничное условие (5.12). Подставляем в него решение (5.16):

$$\sum_{n=1}^{\infty} X_n(a) \sin\frac{n\pi y}{b} = \sin\frac{\pi y}{b}. \tag{5.20}$$

Это возможно тогда и только тогда, когда

$$\begin{cases} X_1(a) = 1; \\ X_n(a) = 0; \quad n > 1. \end{cases} \tag{5.21}$$

Решаем систему дифференциальных уравнений (5.18) при граничных условиях (5.19) и (5.21). При $n>1$:

$$X_n''(x) - \frac{n^2\pi^2}{b^2} X_n(x) = 0; \quad X_n(x) = C_1 \operatorname{ch}\frac{n\pi x}{b} + C_2 \operatorname{sh}\frac{n\pi x}{b}. \tag{5.22}$$

Подставляем граничные условия:

$$\begin{cases} X_n(0) = C_1 = 0; \\ X_n(a) = C_2 \operatorname{sh}\dfrac{n\pi a}{b} = 0. \end{cases} \tag{5.23}$$

Во втором уравнении второй множитель (гиперболический синус) не равен нулю, поэтому $C_2=0$. Таким образом, $\forall n>1$: $X_n(x)=0$. Найдем теперь $X_1(x)$. Дифференциальное уравнение для него — это 1-е уравнение системы (5.18). Вспомните, как решаются такие уравнения: нужно взять сумму общего реше-

ния соответствующего однородного уравнения (вида (5.22)) и частного решения неоднородного уравнения. Проделайте эти выкладки самостоятельно. В результате получим:

$$X_1(x) = C_1 \operatorname{ch}\frac{\pi x}{b} + C_2 \operatorname{sh}\frac{\pi x}{b} - \frac{b^2 x}{\pi^2}. \tag{5.24}$$

Произвольные постоянные C_1 и C_2 найдем из граничных условий (5.19) и (5.21):

$$\begin{cases} X_1(0) = C_1 = 0; \\ X_1(a) = C_2 \operatorname{sh}\dfrac{\pi a}{b} - \dfrac{b^2 a}{\pi^2} = 1. \end{cases} \tag{5.25}$$

Отсюда находим

$$C_2 = \frac{1 + \dfrac{b^2 a}{\pi^2}}{\operatorname{sh}\dfrac{\pi a}{b}} = \frac{\pi^2 + b^2 a}{\pi^2 \operatorname{sh}\dfrac{\pi a}{b}}. \tag{5.26}$$

Решение нашей задачи — это одна первая гармоника ряда (5.16):

$$u(x,y) = \left(\frac{\pi^2 + b^2 a}{\pi^2 \operatorname{sh}\dfrac{\pi a}{b}} \operatorname{sh}\frac{\pi x}{b} - \frac{b^2 x}{\pi^2} \right) \sin\frac{\pi y}{b}. \tag{5.27}$$

На рис. 5.2 показано, как выглядит график этой функции (мы здесь взяли $a = 1$, $b = 2$).

```
clear all % очистили память
a=1; % задали размеры
b=2;
x=linspace(0,a,40);
y=linspace(0,b,80);
[X,Y]=meshgrid(x,y); % сетка
U=((pi^2+b^2*a)/(pi^2*sinh(pi*a/b))*sinh(pi*X/b)-...
  b^2*X/pi^2).*sin(pi*Y/b); % вычисляем функцию
surf(X,Y,U) % рисуем поверхность
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
da=daspect; % текущие масштабы осей
da(1:2)=min(da(1:2)); % одинаковые масштабы
daspect(da); % установили одинаковые масштабы
```

```
title('\bfПример 5.1')
xlabel('\itx') % ось OX
ylabel('\ity') % ось OY
zlabel('\itu\rm(\itx\rm,\ity\rm)') % ось OZ
```

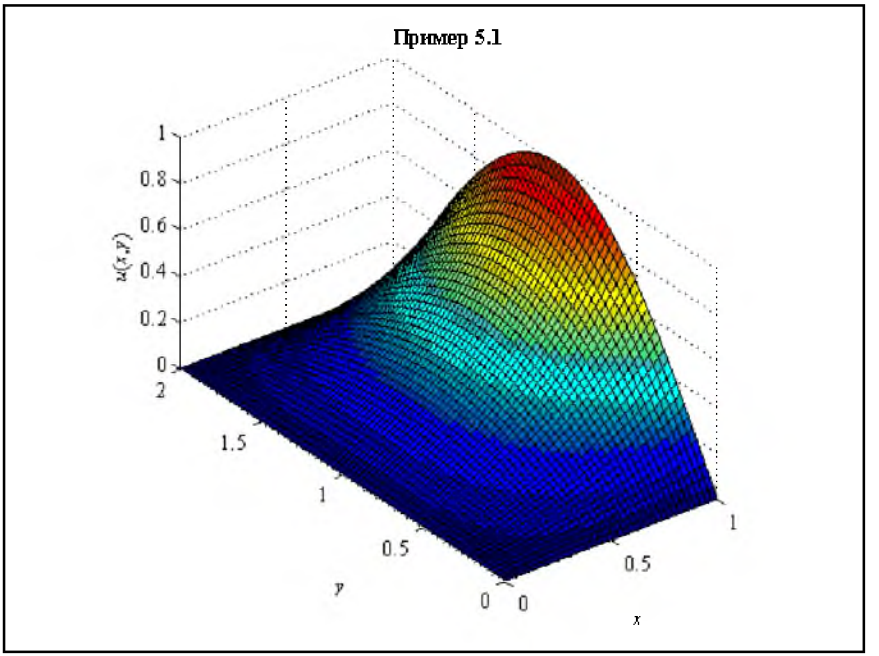


Рис. 5.2. Решение примера 5.1

5.2. Вопросы для самопроверки

1. Какую вариационную задачу мы решаем?
2. Как выводится дифференциальное уравнение Эйлера — Остроградского?
3. Где используется в выводе дифференциального уравнения Эйлера — Остроградского основная лемма вариационного исчисления?
4. Почему мы не можем использовать формулу интегрирования по частям? Чем мы ее заменяем?
5. Обязательно ли будет достигаться экстремум функционала на решении дифференциального уравнения Эйлера — Остроградского?
6. Какие вы знаете методы решения дифференциальных уравнений в частных производных?

7. Всегда ли решение вариационной задачи будет единственным? От чего это зависит?
8. Выведите систему дифференциальных уравнений Эйлера — Остроградского для функционала, зависящего от нескольких функций нескольких переменных.

5.3. Пример выполнения задания

Найти экстремум функционала

$$J(z)=\iint\limits_D\left(z_x^2+2z_y^2+10yz\left(\sin x+\frac{x^2}{5}\right)\right)dS\to\text{extr} \tag{5.28}$$

в области D, которая представляет квадрат со стороной 0,4 м, скругленный по верхнему краю дугой окружности радиуса 0,4 м, с добавкой полукруга слева и с вырезанной частью эллиптического очертания справа. Центр полукруга находится посередине левой стороны, а центр эллипса — на расстоянии 0,1 м наружу от середины правой стороны. Эллипс с полуосями 0,2 м и 0,1 м повернут на 30° против часовой стрелки. Начало координат выберем в центре полукруга (рис. 5.3).

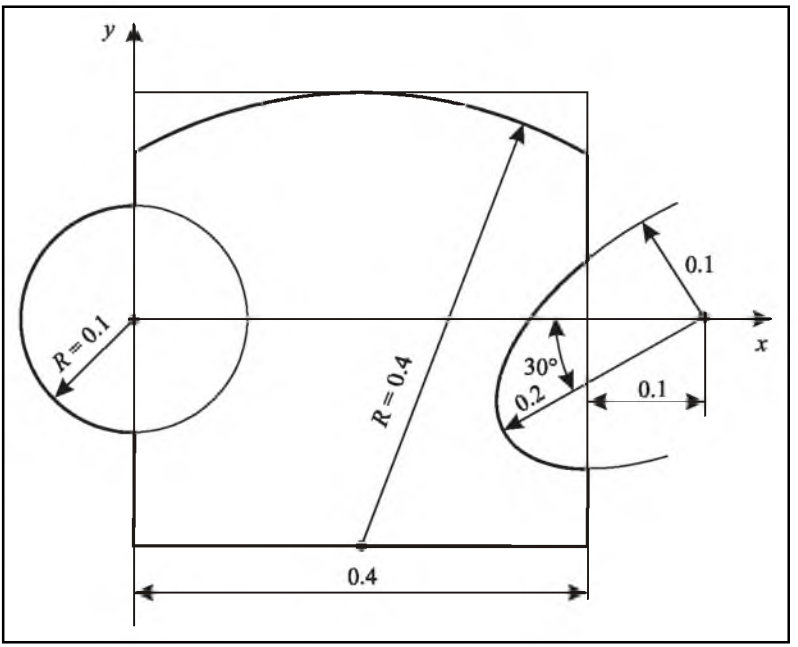


Рис. 5.3. Область решения задания 5

Граничные условия: на нижней стороне u изменяется по параболическому закону в зависимости от x с максимальным значением -10^{-4} м посередине стороны, на остальных сторонах $u=0$.

Составим программу для решения данной задачи. Вначале очистим рабочую область от предыдущих задач. Опишем необходимые переменные и введем исходные данные. Нам будут нужны символические переменные для аргументов `x` и `y`, функции `z`, первых и вторых частных производных `Dzx`, `Dzy`, `D2zx2`, `D2zxy` и `D2zy2`. Вводим подынтегральную функцию `F` (символическое выражение), граничное условие `zc` там, где оно не равно нулю (также символическое выражение), и строковую переменную `bc`, показывающую, где именно граничное условие отлично от нуля. Не забудьте, что при вычислениях мы имеем дело с округленными значениями. Поэтому пишите условие `bc` так, чтобы захватить нужную вам сторону.

```
clear all
disp('Решаем задание 5')
syms x y z Dzx Dzy D2zx2 D2zxy D2zy2
F=Dzx^2+2*Dzy^2+10*y*z*(sin(x)+x^2/5);
zc=x*(x-0.4)/400; % граничное условие
bc='y<-0.1999'; % участок границы, где задана zc
disp('Исходные данные:')
disp('Подынтегральная функция:')
fprintf('F=%s\n',char(F))
disp('Граничное условие:')
fprintf('при %s: z=%s;\n',bc,char(zc))
disp('на остальных участках z=0.')
```

```
Решаем задание 5
Исходные данные:
Подынтегральная функция:
F=Dzx^2+2*Dzy^2+10*y*z*(sin(x)+1/5*x^2)
Граничное условие:
при y<-0.1999: z=1/400*x*(x-2/5);
на остальных участках z=0.
```

Найдем частные производные F_z, F_p и F_q. Сформируем из них полные частные производные $\partial F_p/\partial x$ и $\partial F_q/\partial y$. При их формировании учитываем, что $z=z(x,y)$, $p=p(x,y)$, $q=q(x,y)$. Используем формулу (2.87). Формируем уравнение Эйлера — Остроградского. Как мы увидим дальше, нам нужно будет иметь уравнение именно в виде (5.14), когда слева записаны частные производные и сама функция z, а справа — известные функции. Поэтому вычислим отдельно левую и правую части. Правую часть получим, когда в выражение $F_z-\partial F_p/\partial x-\partial F_q/\partial y$ подставим $z=0$, и вместо всех частных производных также подставим нуль. Тогда левая часть — это все остальное.

```
dFdz=diff(F,z);
dFdp=diff(F,Dzx);
dFdq=diff(F,Dzy);
d_dFdp_dx=diff(dFdp,x);
d_dFdp_dz=diff(dFdp,z);
d_dFdp_dp=diff(dFdp,Dzx);
d_dFdp_dq=diff(dFdp,Dzy);
dFpdx=d_dFdp_dx+d_dFdp_dz*Dzx+d_dFdp_dp*D2zx2+d_dFdp_dq*D2zxy;
d_dFdq_dy=diff(dFdq,y);
d_dFdq_dz=diff(dFdq,z);
d_dFdq_dp=diff(dFdq,Dzx);
d_dFdq_dq=diff(dFdq,Dzy);
dFqdy=d_dFdq_dy+d_dFdq_dz*Dzy+d_dFdq_dp*D2zxy+d_dFdq_dq*D2zy2;
Euler=simple(dFdz-dFpdx-dFqdy);
EuR=-subs(Euler,{z,D2zx2,D2zy2,D2zxy},{0,0,0,0});
EuL=Euler+EuR; % левая часть уравнения
deqEuler=[char(EuL) '=' char(EuR)]; % уравнение
disp('Уравнение Эйлера-Остроградского:')
fprintf('%s\n',deqEuler)
Уравнение Эйлера — Остроградского:
-2*D2zx2-4*D2zy2=-10*y*sin(x)-2*y*x^2
```

Для решения дифференциальных уравнений в частных производных в MATLAB есть специальный инструментарий — Partial Differential Equation Toolbox (PDE Toolbox) [54], в котором используется метод конечных элементов (МКЭ, the finite elements method — FEM). Для применения PDE нужно привести дифференциальное уравнение к виду

$$-\mathrm{div}(C\,\mathrm{grad}\,u)+au=f \tag{5.29}$$

и дополнить его граничными условиями Дирихле

$$hu=r \tag{5.30}$$

или Неймана

$$\mathrm{div}(C\,\mathrm{grad}\,u)+qu=g. \tag{5.31}$$

Здесь $u(x,y)$ — искомая функция, C — матрица 2×2, элементы которой являются коэффициентами при $\partial^2u/\partial x^2$, $\partial^2u/\partial x\partial y$, $\partial^2u/\partial y^2$; a — коэффициент при u; f — правая часть, h, r, q, g — заданные функции x, y. Величины C, a, f, h, r, q, g могут быть как постоянными, так и переменными. В последнем случае они должны вычисляться в центрах тяжести конечных элементов и задаваться как массивы.

Посмотрите на выведенное нами уравнение Эйлера — Остроградского: не зря мы приводили его к виду (5.29)!

Процесс решения дифференциального уравнения в частных производных при помощи PDE состоит из следующих этапов.

1. Задание геометрии (области решения) и построение сетки из конечных элементов.
2. Задание граничных условий (функций h, r, q, g).
3. Задание функций, входящих в дифференциальное уравнение (C, a, f).
4. Решение дифференциального уравнения.
5. Отображение результатов в виде графика.

Область решения в PDE строится из примитивов (простейших областей). Примитивами являются круг, многоугольник, прямоугольник и эллипс. Из них можно составлять сложные области, используя логические операции "+" ("или"), "*" ("и") и "–" ("и не"). Операции "+" и "*" имеют одинаковый приоритет, а операция "–" более низкий. Таким образом, чтобы задать геометрию области, нужно задать:

- простейшие области (примитивы);
- формулу для построения области из примитивов с помощью логических операций "+", "*", "–".

Примитивы задаются в виде матрицы. Число столбцов этой матрицы равно числу примитивов, каждый ее столбец содержит данные по одному примитиву. Будем обозначать эту матрицу идентификатором `gd`.

Рассмотрим, как задаются различные примитивы. Чтобы задать *круг*, нужно в соответствующий столбец матрицы `gd` занести числа в таком порядке:

- первое число — это 1 (признак круга);
- 2-е и 3-е числа — это x-я и y-я координаты центра;
- 4-е число — радиус круга.

Таким образом, положение круга на плоскости будет полностью определено. Аналогично задаются и другие примитивы.

Чтобы задать *многоугольник*, нужно в соответствующий столбец занести числа в таком порядке:

- первое число — это 2 (признак многоугольника);
- 2-е число — это n — число вершин;
- следующие n чисел содержат x-е координаты вершин;
- следующие n чисел — это y-е координаты вершин.

Прямоугольник задается так же, как и многоугольник, но первое число — это 3 (признак прямоугольника).

Для задания *эллипса* нужно в соответствующий столбец занести числа в таком порядке:

- первое число — 4 (признак эллипса);
- 2-е и 3-е числа — это x-я и y-я координаты центра;
- 4-е и 5-е числа — это длины полуосей;
- 6-е число — угол поворота эллипса (в радианах, положительный поворот — против часовой стрелки).

Если столбцы полученной матрицы `gd` имеют различную длину, нужно дополнить более короткие столбцы нулями.

Чтобы сформировать из этих примитивов область, нужно задать в программе формулу, с помощью которой такая область будет формироваться из столбцов матрицы `gd`. В этой формуле столбцы матрицы `gd` будут обозначены какими-либо буквами, поэтому нужно также задать соответствие между буквами в формуле и столбцами матрицы. Такое соответствие задается строкой, каждый символ которой соответствует одному столбцу. Можно, например, задать строку вида `'abcdefgh'` (или более длинную, если это необходимо). Тогда в формуле для построения области буквой `a` будет обозначен 1-й столбец (примитив), буквой `b` — 2-й и т. д.

Формирование области выполняется с помощью команды `decsg`. В результате работы этой процедуры возвращаются 2 массива, которые мы обозначим: `dl` и `bt`. Массив `dl` — это разложенная матрица геометрии (the decomposed geometry matrix). Она имеет не более 12 строк и n_e столбцов, где n_e — число элементарных участков границы. Элементарный участок границы — это часть какого-либо примитива с монотонным изменением x и y. Так, например, круглая или эллиптическая области имеют 4 элементарных участка границы, разделяемые самой правой, левой, верхней и нижней точками. Каждый столбец массива `dl` соответствует одному участку границы и содержит такие числа:

- 1-е число — это тип линии, которая образует данную границу:
  - 1 — окружность,
  - 2 — многоугольник,
  - 3 — прямоугольник,
  - 4 — эллипс.
- 2-е и 3-е числа — это начальная и конечная x-е координаты участка границы;
- 4-е и 5-е числа — это y-е координаты начальной и конечной точки участка границы;

- 6-е и 7-е числа — это номера примитивов, которые находятся слева и справа от данного участка границы. Эти номера берутся в соответствии с нумерацией примитивов в массиве `gd`, число 0 соответствует внешней границе. Остальные числа зависят от того, какая линия образует границу (т. е. от 1-го числа);
- для окружности 8-е и 9-е числа — это координаты центра окружности, 10-е число — радиус окружности;
- для прямолинейной границы ничего не задается (граница уже полностью определена);
- для эллиптической границы 8-е и 9-е числа содержат координаты центра эллипса, 10-е и 11-е — величины полуосей и 12-е — угол поворота.

В массиве `bt` возвращается булевская таблица соответствия примитивов и построенной области (более подробно о `bt` можно узнать по команде `help decsg`).

Внутренние границы удаляются командой `csgdel`. Эта команда возвращает новые разложенную матрицу геометрии и таблицу соответствия, с удаленными внутренними границами. С помощью команды `pdegplot` можно нарисовать полученную область.

Наша область образована следующими примитивами:

- квадрат со стороной 0,4;
- круг радиусом 0,4 с центром в точке $(0{,}2; -0{,}2)$; он отрезает от квадрата верхнюю часть;
- круг радиусом 0,1 с центром в начале координат; он добавляется к области;
- эллипс с центром в точке (0,5; 0), с полуосями 0,2; 0,1, повернутый на угол 30°; он вырезается из области.

Зададим эти примитивы в виде матрицы. В 1-м столбце запишем данные для прямоугольника, во 2-м и 3-м — для кругов, и в 4-м — для эллипса. Уравняем длины всех столбцов.

Затем сформируем из этих примитивов область. Обозначим столбцы матрицы буквами a, b, c, d. Тогда область может быть получена с помощью формулы $\Omega = a * b + c - d$. Действительно, из квадрата a кругом b вырезается область, затем к ней добавляется круг c, и из полученной области вырезается эллипс d. Напечатаем число элементарных участков границы. Изобразим полученную область графически (рис. 5.4). Выровняем масштабы по осям координат. Надпишем заголовок и метки осей.

```
gd=[2;4;0;0.4;0.4;0;-0.2;-0.2;0.2;0.2]; % многоугольник
gd=[gd,[1;0.2;-0.2;0.4;0;0;0;0;0;0]]; % верхняя окружность
```

```
gd=[gd,[1;0;0;0.1;0;0;0;0;0;0]]; % левая окружность
gd=[gd,[4;0.5;0;0.2;0.1;pi/6;0;0;0;0]]; % эллипс
ns='abcd'; % строка соответствия столбцов
fo='(a*b+c)-d'; % формула операций
[dl,bt]=decsg(gd,fo,ns); % формируем область
[dl1,bt1]=csgdel(dl,bt); % удалили внутренние границы
ne=size(dl1,2); % число участков границы
fprintf('Число участков границы ne=%d\n',ne)
figure % фигура
pdegplot(dl1) % нарисовали область
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
da=daspect; % текущие масштабы осей
da(1:2)=min(da(1:2)); % одинаковые масштабы
daspect(da); % установили одинаковые масштабы
title('\bfЗадание 5 - область решения')
xlabel('\itx') % ось OX
ylabel('\ity') % ось OY
```

```
Число участков границы ne=11
```

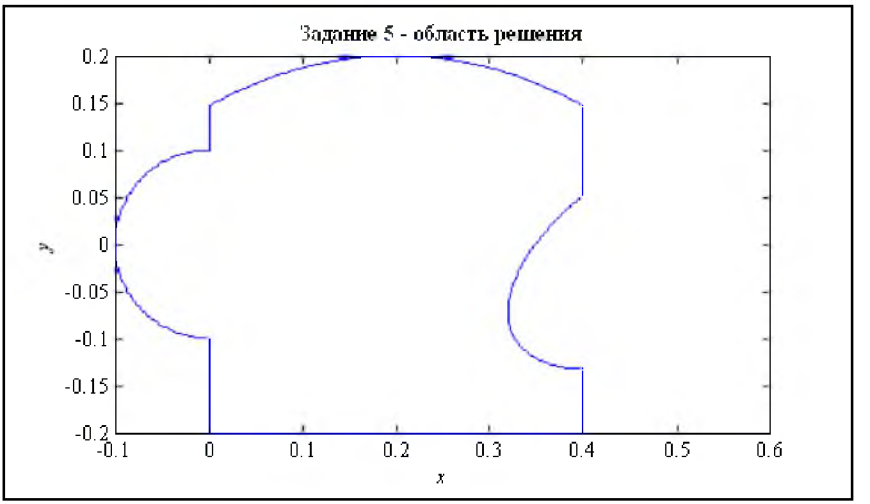


Рис. 5.4. Область решения в задании 5

Разбивка полученной области на треугольные конечные элементы осуществляется командой `initmesh`, которая возвращает 3 выходных параметра: `p`, `e` и `t`. Смысл их следующий:

- в переменной `p` возвращаются координаты узлов сформированной сетки. Массив `p` имеет размеры $2 \times n_p$, где n_p — число узлов.

В нем содержатся:

- 1-я строка — x-е координаты узлов;
- 2-я — y-е координаты;

❒ в переменной `t` возвращаются данные по треугольникам. Это массив размером $4 \times n_{el}$, где n_{el} — число элементов:

- первые 3 строки содержат номера узлов для каждого элемента в порядке обхода против часовой стрелки;
- 4-е число — это номер подобласти. Если мы удалим все внутренние границы, то у нас будет только одна подобласть, и все числа 4-й строки будут 1.

В переменной `e` возвращаются данные по граничным точкам сетки. Размер массива `e` $7 \times n_b$, где n_b — число участков границы в сетке МКЭ. В каждом столбце массива `e` содержатся данные по одной граничной линии. Числа этого столбца такие:

- 1-е и 2-е числа — это номера узлов (в том порядке, в котором они перечислены в массиве `p`);
- 3-е и 4-е числа содержат начальное и конечное значения параметра длины в начальной и конечной точках. Параметр длины точки — это отношение расстояния от начала участка границы до данной точки к общей длине участка границы;
- 5-е число содержит номер участка границы;
- 6-е и 7-е числа — это номера подобластей слева и справа от данной границы. Если мы удалим все внутренние границы, то из 6-го и 7-го чисел одно число будет равно 1, а другое — 0;

Заметим, что $n_b \neq n_e$: n_e — это число элементарных участков границы (дуг окружностей или эллипсов, отрезков прямых), а n_b — это число сторон конечных элементов, которые выходят на границу. Обычно $n_b > n_e$.

Полученную сетку МКЭ можно один или несколько раз измельчить при помощи команды `refinemesh`. Эту команду можно использовать, например, в цикле для достижения нужной точности вычислений.

Изобразить сетку разбиения можно командой `pdemesh`.

Сформируем треугольную сетку МКЭ и измельчим ее. Напечатаем количество узлов, элементов, граничных линий. Нарисуем полученную сетку (рис. 5.5). Выровняем масштабы по осям, надпишем заголовок, метки осей.

```
[p,e,t]=initmesh(dl1); % формируем FEM-сетку
[p,e,t]=refinemesh(dl1,p,e,t); % измельчаем
```

```
np=size(p,2); % число узлов
nel=size(t,2); % число элементов
nb=size(e,2); % число граничных линий
fprintf('Число узлов np=%d\n',np)
fprintf('Число элементов nel=%d\n',nel)
fprintf('Число граничных линий nb=%d\n',nb)
figure % фигура
pdemesh(p,e,t) % рисуем сетку
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
da=daspect; % текущие масштабы осей
da(1:2)=min(da(1:2)); % одинаковые масштабы
daspect(da); % установили одинаковые масштабы
title('\bfЗадание 5 - сетка МКЭ')
xlabel('\itx') % ось OX
ylabel('\ity') % ось OY
Число узлов np=506
Число элементов nel=924
Число граничных линий nb=86
```

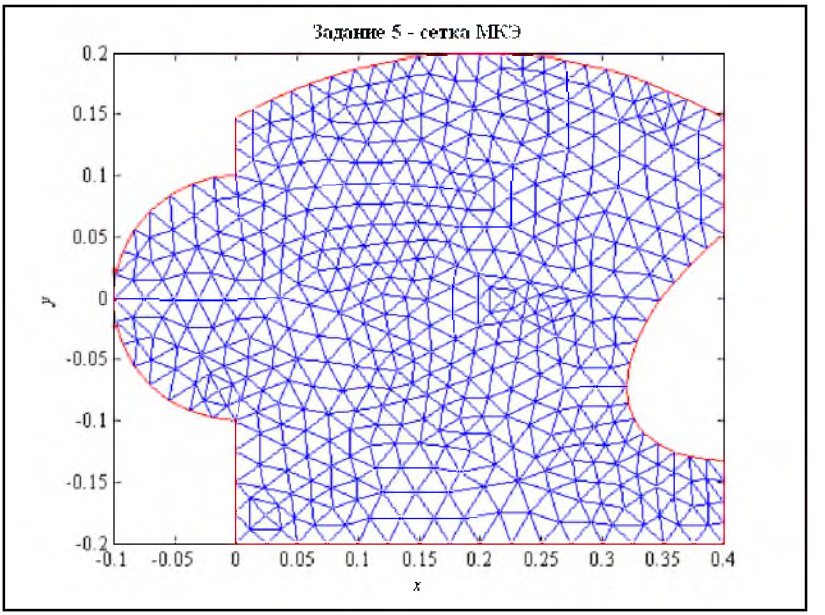


Рис. 5.5. Сетка МКЭ в задании 5

Следующий этап — это задание *граничных условий*. Граничные условия можно задать или в виде матрицы граничных условий (boundary condition matrix),

или в виде m-файла граничных условий (boundary M-file). Второй вариант проще. Файл граничных условий должен иметь такую структуру:

```
[q,g,h,r]=pdebound(p,e,u,time)
```

Входные параметры: `p`, `e` — данные по сетке разбиения, `u` — решение, `time` — время. Выходные параметры — матрицы граничных условий Дирихле (5.30) или Неймана (5.31). PDE Toolbox позволяет решать параболические и гиперболические уравнения (зависящие от времени), а также нелинейные задачи, поэтому граничные условия и параметры дифференциального уравнения могут зависеть также от времени и решения.

Для граничных условий Неймана выходные параметры `q` и `g` должны содержать значения параметров q и g в средних точках границ. Размер `q`: $N^2 \times n_b$, где N — число уравнений системы, а n_b — число сторон конечных элементов, выходящих на границу. Размер `g`: $N \times n_b$. В каждом из столбцов этих матриц должны возвращаться коэффициенты q и g в средних точках соответствующей границы. Для нескольких уравнений элементы q должны следовать по столбцам. Так, например, для 2-х уравнений в каждом столбце `q` нужно возвратить q_{11}, q_{21}, q_{12}, q_{22}. В случае граничных условий Дирихле в этих матрицах должны возвращаться нулевые значения.

Для граничных условий Дирихле должны формироваться выходные параметры `h` и `r`. Массив `h` имеет размеры $N^2 \times (2n_b)$ и содержит значения h во всех начальных точках каждой граничной линии и сразу за ними — в конечных точках граничных линий. Размер массива `r`: $N \times (2n_b)$, этот массив заполняется аналогично.

У нас заданы граничные условия Дирихле, поэтому сформируем файл для их вычисления. Число уравнений у нас $N = 1$, число точек граничных линий n_b находим из массива `e`. Формируем строки для записи в файл и записываем их. Файл размещаем в рабочем каталоге системы MATLAB. Имена всех файлов, которые мы будем записывать в каталоги MATLAB, мы будем начинать с префикса `My`. Файлов с такими именами в каталогах MATLAB нет, поэтому мы ничего не испортим. Файл граничных условий назовем `MyBound.m`. Это — обычный текстовый файл. Расширение `m` говорит о том, что MATLAB будет воспринимать его как свою функцию, причем имя функции должно совпадать с именем файла.

```
s{1}='function [q,g,h,r]=MyBound(p,e,u,time)';
s{2}='nb=size(e,2);'; % число граничных линий
s{3}='q=zeros(1,nb); g=zeros(1,nb);';
s{4}='h=ones(1,2*nb); r=zeros(1,2*nb);';
s{5}='x=[p(1,e(1,:)),p(1,e(2,:))];'; % столбец x
s{6}='y=[p(2,e(1,:)),p(2,e(2,:))];'; % столбец y
```

```
s{7}=['myb=find(' bc ');']; % номера нужных точек
s{8}='xb1=x(myb); yb1=y(myb);'; % нужные точки
zcf=subs(zc,{x,y},{sym('xb1'),sym('yb1')});
s{9}=['r(myb)=' vectorize(zcf) ';']; % вычислили
filename=fullfile(pwd,'MyBound.m'); % файл
disp(['Текст файла граничных условий ' filename ':'])
fprintf('%s\n',s{:})
fid=fopen(filename,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записываем в файл
fclose(fid); % закрываем файл
Текст файла граничных условий D:\Iglin\Matlab\MyBound.m:
function [q,g,h,r]=MyBound(p,e,u,time)
nb=size(e,2);
q=zeros(1,nb); g=zeros(1,nb);
h=ones(1,2*nb); r=zeros(1,2*nb);
x=[p(1,e(1,:)),p(1,e(2,:))];
y=[p(2,e(1,:)),p(2,e(2,:))];
myb=find(y<-0.1999);
xb1=x(myb); yb1=y(myb);
r(myb)=1./400.*xb1.*(xb1-2./5);
```

Давайте посмотрим, *какой* файл мы сформировали и *как* мы это сделали. Записываем в массив ячеек `s` нужные строки. Первая строка — это заголовок. Он должен соответствовать шаблону, приведенному выше. Во 2-й строке мы находим `nb` — число сторон конечных элементов, выходящих на границу. В 3-й и 4-й строках задаем нулевые массивы `q`, `g`, `r` и единичный массив `h` нужных размерностей для граничных условий Неймана и Дирихле. Массив из нулей задается функцией `zeros`, а массив из единиц — функцией `ones`. Граничные условия Неймана (5.31) мы больше трогать не будем, они так и останутся нулевыми. А вот в граничных условиях Дирихле массив `r` *не весь* должен быть нулевым: в некоторых граничных точках, определенных условием `bc`, элементы массива `r` должны вычисляться по формуле `zc`. Поэтому в 5-й и 6-й строках мы из всех узлов `p` выбираем граничные точки. В 7-й строке мы с помощью функции `find` находим номера тех точек, которые удовлетворяют условию `bc`. Эти номера записываются в массиве `myb`. В 8-й строке из всех граничных точек отбираются те, для которых условие `bc` выполняется. Координаты этих точек — в массивах `xb1` и `yb1`. Теперь в элементы массива `r` с номерами `myb` нужно записать результат вычисления по формуле `zc`. Мы сначала подставляем в формулу `zc` вместо аргументов `x` и `y` массивы `xb1` и `yb1`, переведенные в символические переменные, а затем в 9-й строке записываем соответствующую формулу. Формулу мы векторизуем с помощью функции `vectorize`, т. е. расставляем точки перед арифметическими операциями. Теперь вычисления будут проводиться над массивами поэлементно, а не по правилам матричной алгебры.

Итак, мы сформировали массив ячеек-строк `s`. Нужно теперь записать его в файл одного из каталогов, доступных системе MATLAB. Сформируем имя файла с помощью функции `fullfile`. Текущий каталог MATLAB получаем функцией `pwd`. Записываем содержимое массива `s` вначале в нашу область вывода, а затем — в файл. Перед записью открываем файл (функция `fopen`), а после записи — закрываем (функция `fclose`).

Следующий этап — *задание функций, входящих в дифференциальное уравнение*. Каждая из функций C, a, f может быть задана в следующих видах:

- в виде константы;
- в виде вектор-строки значений функции в центрах масс треугольников. Если для матрицы C задаются 2 строки, то подразумевается, что C — диагональная, и задаются c_{11} и c_{22}. Если для матрицы C задаются 4 строки, то подразумевается, что это элементы матрицы C в порядке c_{11}, c_{21}, c_{12}, c_{22};
- в виде текстового выражения, составленного по правилам MATLAB, по которому можно вычислить соответствующую функцию в центрах масс треугольников. Эта функция может зависеть от переменных `x`, `y`, `sd`, `u`, `ux`, `uy`, `t`. Смысл этих переменных: `t` — время (скаляр); остальные переменные — векторы-строки, представляющие значения в центрах масс треугольников: `x`, `y` — координаты центров масс; `sd` — номер подобласти; `u`, `ux`, `uy` — решение и его частные производные;
- в виде последовательности выражений, рассмотренных в предыдущем пункте. Эти выражения должны быть отделены друг от друга знаками `!`. Они воспринимаются как различные задания функции в разных подобластях. Этих выражений должно быть столько, сколько есть различных подобластей, т. е. `max(t(4,:))`, где `t` — массив, возвращаемый командой `initmesh`;
- в виде имени определенной пользователем функции MATLAB (т. е. m-файла), который представляет функцию, зависящую от аргументов `(p,t,u,time)`, где `p, t` — данные по сетке разбиения, `u` — решение, `time` — время.

Найдем C, a, f Элементы матрицы C — это коэффициенты при частных производных. При их вычислении учтем, что коэффициенты при $\partial^2 u/\partial x\partial y$ и $\partial^2 u/\partial y\partial x$ одинаковые. Число a — это коэффициент при u в дифференциальном уравнении. Параметр f вычисляем вначале в узлах, а затем в центрах тяжести конечных элементов.

```
a=eval(subs(EuL,{z,D2zx2,D2zxy,D2zy2},{1,0,0,0}));
c11=eval(subs(EuL,{z,D2zx2,D2zxy,D2zy2},{0,1,0,0}));
c12=eval(subs(EuL,{z,D2zx2,D2zxy,D2zy2},{0,0,1,0}))/2;
```

```
c22=eval(subs(EuL,{z,D2zx2,D2zxy,D2zy2},{0,0,0,1}));
c=-[c11;c12;c12;c22];
fprintf('Параметры: a=%d\n',a)
disp('C=')
fprintf('%d  %d\n',c)
if (simple(diff(EuR,x))==0), % x не входит
  fp=subs(EuR,y,p(2,:)); % подставили только y
elseif (simple(diff(EuR,y))==0), % y не входит
  fp=subs(EuR,x,p(1,:)); % подставили только x
else
  fp=subs(EuR,{x,y},{p(1,:),p(2,:)}); % f в узлах
end
f=(fp(t(1,:))+fp(t(2,:))+fp(t(3,:)))/3; % в центрах тяжести
Параметры: a=0
C=
2  0
0  4
```

Решаем наше дифференциальное уравнение с помощью функции `assempde`. Рисуем график решения командой `pdeplot` (рис. 5.6). Показываем конечно-элементную сетку и не показываем линейку соответствия цветов. Команда `axis` возвращает или устанавливает границы рисунка по осям. Мы получили текущие границы, выровняли масштаб по осям *Ox* и *Oy*, а затем восстановили старые границы. Надписываем оси. Выбираем палитру. Показываем сетку и контур. Осталось почистить за собой: убрать из текущего каталога записанный туда файл `MyBound.m`. Делаем это командой `delete`.

```
u=assempde('MyBound',p,e,t,c,a,f); % решили
figure % фигура
pdeplot(p,e,t,'xydata',u,'zdata',u,...
  'mesh','on','colorbar','off'); % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
v=axis; % границы осей
da=daspect; % масштаб осей
da(1:2)=min(da(1:2)); % min из двух
daspect(da) % выравниваем масштаб осей
axis(v); % оставили границы
colormap default % палитра по умолчанию
grid on % показали сетку
box on % показали внешний контур
title('\bfЗадание 5 - решение') % заголовок
xlabel('\itx') % метка оси OX
```

```
ylabel('\ity') % метка оси OY
zlabel('\itz\rm(\itx\rm,\ity\rm)') % метка оси OZ
delete(filename) % удалили файл граничных условий
```

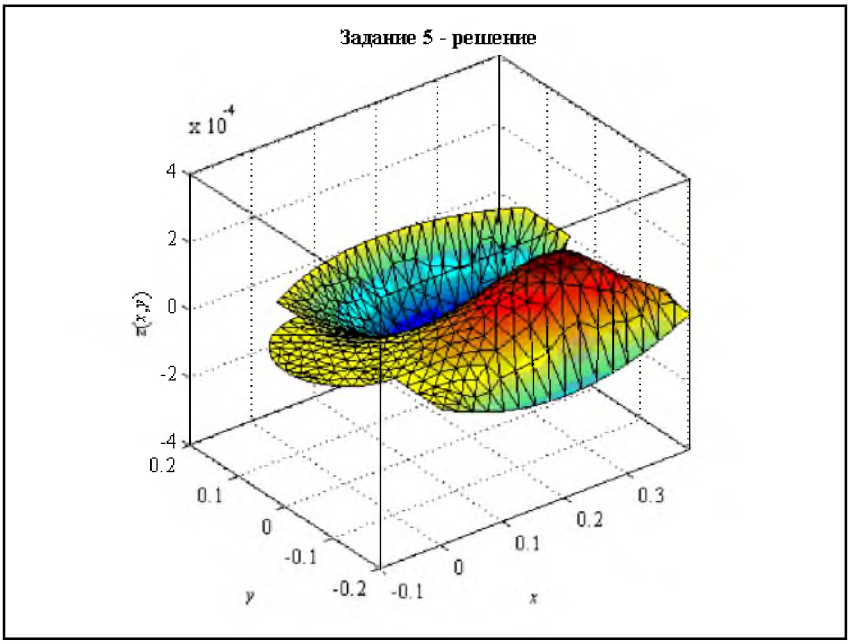
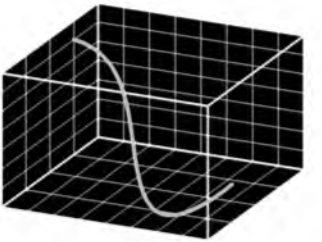


Рис. 5.6. Решение задания 5

5.4. Задание

Для своего варианта функционала выведите дифференциальное уравнение Эйлера — Остроградского и решите его МКЭ. Нарисуйте область решения, сетку МКЭ и трехмерную поверхность — график решения. Варианты заданий есть на компакт-диске.

ГЛАВА 6



Вариационная задача в параметрической форме

6.1. Когда это нужно?

В некоторых случаях решение вариационной задачи удобнее искать в параметрической форме. Чаще всего это происходит тогда, когда функция $y(x)$ неоднозначная. Например, на рис. 6.1 показано решение изопериметрической вариационной задачи: максимизировать площадь S, ограниченную кривой $y(x)$, которая соединяет точки $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$, и отрезком прямой M_1M_2, при дополнительном ограничении-равенстве на длину дуги $M_1M_2 = l = \text{const}$ (это задача Дидоны — *см. главу 14*).

Если длина дуги l не очень большая, то проблем нет: задачу можно решать и в явном виде, т. к. в этом случае функция $y(x)$ однозначная. Но если l значительно больше расстояния между точками M_1 и M_2, то решением этой задачи будет неоднозначная функция, которую удобнее записывать в параметрической форме:

$$\begin{cases} x = x(t); \\ y = y(t). \end{cases} \tag{6.1}$$

Тогда мы имеем вариационную задачу для двух функций, но уже однозначных: нужно максимизировать функционал:

$$S\big(x(t), y(t)\big) = \frac{1}{2}\int_{t_1}^{t_2}\left(x\dot{y} - y\dot{x}\right)dt \to \max \tag{6.2}$$

при заданных граничных условиях

$$\begin{cases} x(t_1) = x_1; \\ y(t_1) = y_1; \end{cases} \quad \begin{cases} x(t_2) = x_2; \\ y(t_2) = y_2, \end{cases} \tag{6.3}$$

и при дополнительном условии

$$L\big(x(t),\, y(t)\big)=\int_{t_1}^{t_2}\sqrt{\dot{x}^2\ \dot{x}+\dot{y}^2}\ dt=l=\text{const}\,. \tag{6.4}$$

Здесь точкой вверху обозначено дифференцирование по параметру t.

Задачи такого типа называются изопериметрическими, мы их будем решать дальше, в *главе 14*. Сейчас нам важен факт, что иногда такую задачу удобно решать именно в параметрической форме.

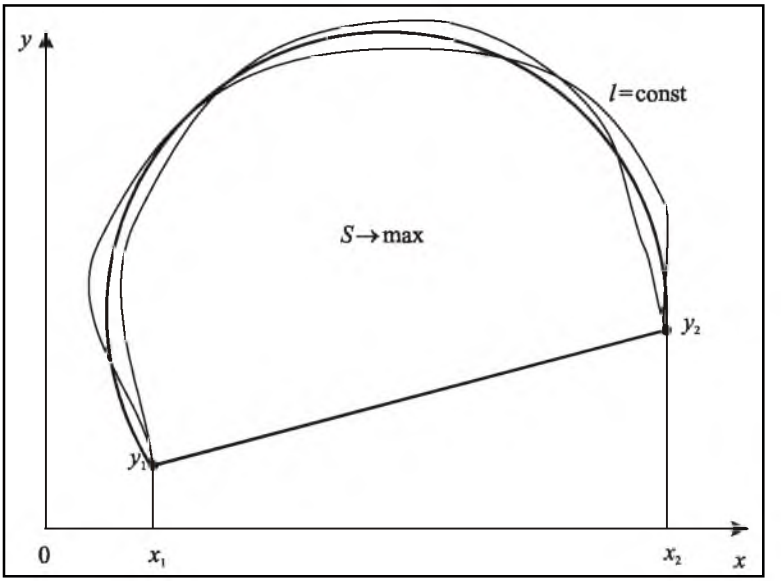


Рис. 6.1. Вариационная задача, которую удобно решать в параметрической форме

6.2. Переход к параметру в элементарной задаче вариационного исчисления

Рассмотрим элементарную задачу вариационного исчисления (2.1—2.2). Пусть по каким-то причинам нам удобно решать ее в параметрической форме. Перейдем к параметрическому заданию (6.1). Вспомните, как находится производная функции, заданной параметрически. Имеем:

$$J\big(x(t),\, y(t)\big)=\int_{t_1}^{t_2}F\left(x(t),\, y(t),\, \frac{\dot{y}(t)}{\dot{x}(t)}\right)\dot{x}(t)\,dt\to\text{extr}\,. \tag{6.5}$$

Заметим, что в полученном выражении подынтегральная функция F обладает двумя интересными свойствами:

1. Она не содержит t в явном виде.
2. Она является однородной функцией 1-го измерения по отношению к $\dot{x}(t)$ и $\dot{y}(t)$, т. е. одновременное умножение этих аргументов на постоянный множитель k приводит к умножению F на этот же множитель.

Таким образом, после перехода к параметру мы имеем не общий случай функционала, зависящего от двух функций (3.1), а только частный случай такой задачи с указанными выше свойствами. Если бы мы переходили к какому-либо другому параметру τ, то вид функционала остался бы таким же:

$$J\big(x(\tau), y(\tau)\big) = \int_{\tau_1}^{\tau_2} F\left(x(\tau), y(\tau), \frac{\dot{y}(\tau)}{\dot{x}(\tau)}\right)\dot{x}(\tau)\,d\tau \to \text{extr}\,. \qquad (6.6)$$

Имеет ли место обратное утверждение? Можно ли сказать, что любой функционал, зависящий от двух функций и обладающий указанными выше свойствами 1 и 2, будет давать параметрическое решение, т. е. решение, зависящее только от траектории движения, а не от конкретного закона движения по этой траектории? Оказывается, да!

■ **ТЕОРЕМА 6.1.** Если в функционале, зависящем от двух функций:

$$J\big(x(t), y(t)\big) = \int_{t_1}^{t_2} \Phi\left(t, x(t), y(t), \dot{x}(t), \dot{y}(t)\right) dt\,, \qquad (6.7)$$

подынтегральная функция F не зависит явно от t (нет первого аргумента) и является однородной функцией 1-го измерения относительно $\dot{x}(t)$ и $\dot{y}(t)$, то функционал J зависит лишь от вида кривой (6.1), а не от ее конкретного параметрического представления.

Доказательство. Перейдем в (6.7) к новому параметру τ путем замены

$$\tau = \varphi(t)\,;\quad \left(\dot{\varphi}(t) \neq 0\right) \;\Rightarrow\; \begin{cases} x = x(\tau)\,; \\ y = y(\tau)\,. \end{cases} \qquad (6.8)$$

Такая замена означает, что мы по той же самой кривой будем двигаться по другому закону (с другой скоростью). Подставим (6.8) в функционал (6.7). В (6.7) точкой вверху обозначено дифференцирование по t. При подстановке используем обычное правило дифференцирования сложной функции: сначала берем производную по τ и умножаем ее на производную от τ по t:

$$J\big(x(\tau),\, y(\tau)\big) = \int_{\tau_1}^{\tau_2} \Phi\left(x(\tau),\, y(\tau),\, \frac{dx}{d\tau}\dot{\varphi}(t),\, \frac{dy}{d\tau}\dot{\varphi}(t) \right) \frac{d\tau}{\dot{\varphi}(t)}. \qquad (6.9)$$

Подынтегральная функция F — однородная функция 1-го измерения относительно 3-го и 4-го своих аргументов, поэтому множитель $\dot{\varphi}(t)$ можно вынести за знак функции F и сократить. Таким образом, мы получим такой же вид, как и (6.7) — вариационная задача не изменилась. □

Из этой теоремы следует интересный факт. Если мы запишем систему дифференциальных уравнений Эйлера (3.4) для задачи (6.7):

$$\begin{cases} \Phi_x - \dfrac{d\Phi_{\dot{x}}}{dt} = 0; \\ \Phi_y - \dfrac{d\Phi_{\dot{y}}}{dt} = 0, \end{cases} \qquad (6.10)$$

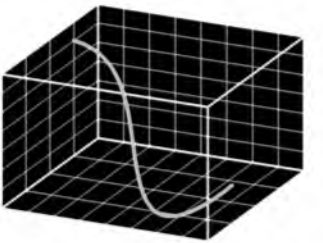
то эти уравнения не будут независимыми! Действительно, если бы они были независимыми, то мы чаще всего получали бы единственную кривую, удовлетворяющую этим уравнениям и граничным условиям вида (6.3) (хотя, если быть точным, то не всегда). Однако здесь мы точно знаем, что в силу доказанной выше теоремы системе (6.10) при любых граничных условиях будет удовлетворять бесконечное множество кривых, отличающихся одна от другой законом движения по одной и той же линии. Значит, одно из уравнений (6.10) является следствием другого. При решении можно взять одно из них и дополнить выбором закона движения по линии.

ПРИМЕР мы рассмотрим дальше, в *главе 14*, при решении задачи на условный экстремум.

6.3. Вопросы для самопроверки

1. Когда удобно решать вариационную задачу в параметрической форме?
2. Какими свойствами обладает подынтегральная функция в простейшей задаче вариационного исчисления после перехода к параметру?
3. Можно ли утверждать обратное?
4. Как формулируется и доказывается соответствующая теорема?
5. Являются ли уравнения системы уравнений Эйлера для задачи в параметрической форме независимыми? Почему?

ГЛАВА 7



Естественные граничные условия

7.1. Элементарная задача вариационного исчисления без граничного условия

Вернемся к элементарной задаче вариационного исчисления для функционала (2.1), зависящего от функции одной переменной и ее производной. Но в отличие от (2.2), будем считать, что граничное условие задано только на левом конце:

$$y(x_1)=y_1, \tag{7.1}$$

а на правом конце x_2 значение функции может быть произвольным, т. е. экстремум функционала (2.1) ищется на классе функций с закрепленным левым концом и свободным правым. Этот класс — более широкий, чем представленный на рис. 2.1. На рис. 7.1 показаны экстремаль (жирная линия), допустимые (тонкие сплошные) и недопустимые (штриховые линии) функции для данной вариационной задачи.

В допустимых функциях разрешается малое их изменение на полуотрезке $(x_1, x_2]$, т. е. может варьироваться также и ордината правой точки. А вот варьирование левой точки уже недопустимо.

Необходимым условием экстремума функционала является равенство нулю его вариации на экстремали: $\delta J=0$. Для вычисления вариации δJ повторим выкладки *главы 2*. Равенство (2.4) имеет место, но вместо двух граничных условий (2.5) имеем только одно:

$$\delta y(x_1)=0, \tag{7.2}$$

т. к. закреплена только левая точка. Следующие выражения (2.6—2.7) остаются в силе, а вот при интегрировании по частям из-за различия в граничных условиях появляется новое слагаемое:

$$\delta J(y_0) = \int_{x_1}^{x_2} F_y\,\delta y\,dx + \left(F_{y'}\,\delta y\right)\Big|_{x_1}^{x_2} - \int_{x_1}^{x_2} \frac{dF_{y'}}{dx}\delta y\,dx = \\ = \left(F_{y'}\,\delta y\right)\Big|_{x_2} + \int_{x_1}^{x_2}\left(F_y - \frac{dF_{y'}}{dx}\right)\delta y\,dx = 0\,. \tag{7.3}$$

Посмотрите на отличие от (2.8): в (7.3) внеинтегральное слагаемое не обратилось в 0, т. к. граничное на правом конце не задано.

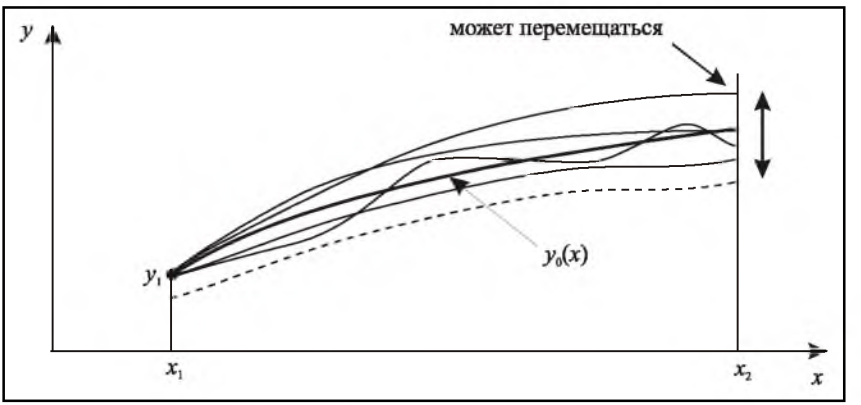


Рис. 7.1. Экстремаль, допустимые и недопустимые функции

Давайте проанализируем: когда же полученное выражение обратится в нуль при произвольных вариациях функции δy? Может ли это произойти за счет компенсации знаков в 1-м и 2-м слагаемых? Очевидно нет! Действительно, если мы будем варьировать функцию $y(x)$ так, что правый конец ее будет неподвижен: $\delta y(x_2) = 0$, то 1-е, внеинтегральное слагаемое в формуле (7.3) не будет изменяться, а 2-е — будет. А нам нужно, чтобы сумма обращалась в нуль *при любом* варьировании $y(x)$, в т. ч. и при таком, какое мы только что применили. Следовательно, (7.3) не может обратиться в 0 за счет компенсации знаков слагаемых. Значит, обязательно каждое из слагаемых в (7.3) должно обращаться в нуль по отдельности. В частности, 2-е слагаемое по основной лемме вариационного исчисления дает дифференциальное уравнение Эйлера (3.9), а 1-е слагаемое из-за произвольности $\delta y(x_2)$ дает недостающее граничное условие на правом конце:

$$F_{y'}(x_2) = 0. \tag{7.4}$$

Определение 7.1. Граничное условие вида (7.4) называется *естественным*. □

Его смысл следующий: если из всех экстремалей, удовлетворяющих заданному граничному условию слева (7.1) и различным граничным условиям

справа, выбрать функцию, доставляющую экстремум функционалу, то эта функция будет удовлетворять естественному граничному условию на правом конце (7.4).

К выводу естественного граничного условия можно подойти и путем таких рассуждений. Мы знаем, что если некоторое свойство выполняется для всех функций какого-либо класса, то оно тем более будет выполняться для всех функций из подкласса данного класса. В частности, если экстремум функционала достигается на классе функций с произвольным значением $y(x_2)$, то он тем более будет достигаться на более узком классе функций: с неподвижной правой точкой. Следовательно, экстремаль должна удовлетворять дифференциальному уравнению Эйлера (2.9). Отсюда следует, что в уравнении (7.3) второе слагаемое обращается в 0, а из 1-го слагаемого получаем естественное граничное условие (7.4).

Совершенно аналогично можно вывести естественное граничное условие на левом конце (если таковое отсутствует), а также оба.

ПРИМЕР 7.1. Решить *пример 2.1* при незаданном граничном условии на правом конце.

Общее решение имеет вид (2.16). Естественное граничное условие (7.4) по (2.12) дает:

$$y'(1) = 0. \tag{7.5}$$

Составляем систему уравнений для нахождения произвольных постоянных:

$$\begin{cases} y(0) = C_2 = 0; \\ y'(1) = 3 + C_1 = 0. \end{cases} \tag{7.6}$$

Отсюда $C_1 = -3$, и решение задачи:

$$y(x) = x^3 - 3x. \tag{7.7}$$

Проверим с помощью MATLAB, чему равен функционал (2.10) на кривой (7.7):

```
clear all % очистили все
syms x % описали символический аргумент
y=x^3-3*x; % символическая функция
Dy=diff(y,x); % производная
F=Dy^2+12*x*y; % подынтегральная функция
J=int(F,x,0,1); % функционал
Jm=eval(J) % посчитали значение
Jm =
  -4.80000000000000
```

Видно, что полученное значение меньше результата *примера 2.1*: там было $J_{\min}=4{,}2$. Это и не удивительно: раз мы ищем минимум на более широком классе функций, то он будет, по крайней мере, не больше, а, может быть, и меньше, чем минимум на более узком классе функций.

К этому результату можно прийти и другим путем. Возьмем все функции вида (2.16) (общее решение уравнения Эйлера), и выберем из них те, что удовлетворяют граничному условию на левом конце. Из 1-го уравнения (2.17) имеем: $C_2=0$; и любое решение уравнения Эйлера, проходящее через точку $M_1(0,0)$, имеет вид:

$$y(x)=x^3+C_1x. \tag{7.8}$$

Каким же нужно взять C_1, чтобы функционал (2.10) принял экстремальное значение? Решим эту задачу с помощью MATLAB:

```
clear all % очистили все
syms x C1 % описали символические переменные
y=x^3+C1*x; % символическая функция
Dy=diff(y,x); % производная по x
F=Dy^2+12*x*y; % подынтегральная функция
J=int(F,x,0,1); % функционал
DJ=diff(J,C1); % производная по C1
eq=char(DJ); % преобразовали в строку
C1=solve(eq) % решаем уравнение dJ/dC1=0
C1 =
-3
```

Получили тот же самый результат: $C_1=-3$. □

ПРИМЕР 7.2. Задача о брахистохроне с подвижным правым концом. Пусть в задаче о брахистохроне левая точка задана: $M_1(0,0)$, на правом конце x_2 ордината y_2 не задана. Требуется при этих условиях найти самую "брахистохронистую" брахистохрону: так подобрать y_2, чтобы время прохождения дистанции нашим лыжником (см. рис. 1.6) было минимальным.

Решение задачи о брахистохроне — циклоида (2.82) — после удовлетворения левому граничному условию (2.83) имеет вид:

$$\begin{cases} x=C_1\left(t-\sin t\right); \\ y=C_1\left(1-\cos t\right). \end{cases} \tag{7.9}$$

Константу C_1 найдем из естественного граничного условия (7.4). Выведем его. Продифференцируем подынтегральную функцию в функционале (2.74) по y':

$$F_{y'} = \frac{y'}{\sqrt{2gy}\sqrt{1+y'^2}} = 0\,. \tag{7.10}$$

Это выражение обращается в 0 при

$$y'(x_2) = 0. \tag{7.11}$$

Для функции, заданной параметрически

$$y' = \frac{\dot{y}}{\dot{x}} = \frac{\sin t}{1-\cos t} = 0, \tag{7.12}$$

что дает нетривиальное решение $t = \pi$. Иными словами, в данной задаче естественное граничное условие приводит к требованию: на правом конце касательная к брахистохроне должна быть горизонтальной. Именно такая кривая будет линией наискорейшего спуска из всех возможных кривых, соединяющих точку $M_1(0, 0)$ и точку с абсциссой x_2 и любой ординатой. Теперь из (7.9) находим C_1:

$$x_2 = C_1\pi; \quad \Rightarrow \quad C_1 = \frac{x_2}{\pi}; \tag{7.13}$$

и решение задачи имеет вид:

$$\begin{cases} x = \dfrac{x_2}{\pi}\left(t - \sin t\right); \\ y = \dfrac{x_2}{\pi}\left(1 - \cos t\right). \end{cases} \quad \square \tag{7.14}$$

7.2. Функционал, зависящий от нескольких функций, без граничного условия

Точно так же решается задача для функционала вида (3.1), зависящего от нескольких функций, если граничные условия заданы только на одном из концов (например, на левом). В этом случае недостающие граничные условия на правом конце выводятся аналогично и имеют вид

$$\begin{cases} F_{y_i'}\left(x_2\right) = 0\,; \\ i = \left[1, n\right]. \end{cases} \tag{7.15}$$

Для функционала, зависящего от двух функций, на рис. 7.2 показаны экстремаль (жирная линия) и допустимые функции (тонкие).

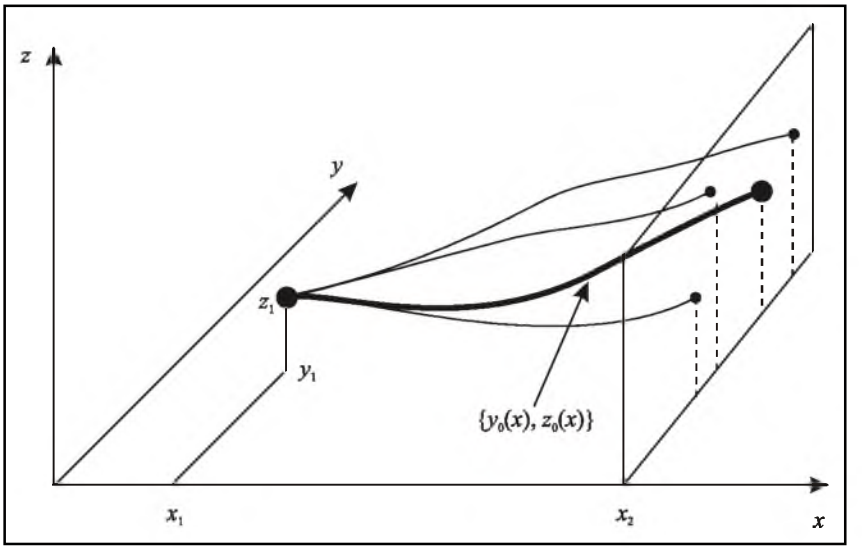


Рис. 7.2. Экстремаль и допустимые функции

Посмотрите: в допустимых функциях правый конец может скользить по плоскости $x = x_2$, а не быть зафиксированным в одной точке, как в *главе 3*.

ПРИМЕР 7.3. Решить *пример 3.1* с подвижным правым концом: найти экстремаль функционала (3.5), когда граничные условия (3.6) заданы только на левом конце интервала, при $x = 0$.

Общее решение системы уравнений Эйлера нами было найдено — это (3.10—3.11). Для нахождения произвольных постоянных выведем естественные граничные условия вида (7.15).

$$\begin{cases} F_{y'} = 2y' = 0\,; \\ F_{z'} = 2z' = 0\,. \end{cases} \tag{7.16}$$

Составляем систему уравнений для нахождения произвольных постоянных. В нее войдут 2 уравнения (3.12) — 1-е и 3-е, и уравнения (7.16).

$$\begin{cases} y(0) = C_1 + C_3 = 0; \\ y'\left(\dfrac{\pi}{2}\right) = C_1 \operatorname{sh}\dfrac{\pi}{2} + C_2 \operatorname{ch}\dfrac{\pi}{2} - C_3 = 0; \\ z(0) = C_1 - C_3 = 0; \\ z'\left(\dfrac{\pi}{2}\right) = C_1 \operatorname{sh}\dfrac{\pi}{2} + C_2 \operatorname{ch}\dfrac{\pi}{2} + C_3 = 0. \end{cases} \tag{7.17}$$

Из 1-го и 3-го уравнений находим единственные значения $C_1 = 0$ и $C_3 = 0$. Теперь 2-е (или 4-е) уравнение дает единственное решение $C_2 = 0$. Оставшаяся константа C_4 вообще в систему уравнений не входит — она произвольная. Таким образом решение нашей вариационной задачи:

$$\begin{cases} y(x) = C_4 \sin x; \\ z(x) = -C_4 \sin x. \end{cases} \tag{7.18}$$

Если вы внимательно посмотрите на это решение, то увидите, что (3.13) — это его частный случай, т. е. найденное ранее решение (3.13) также удовлетворяет естественным граничным условиям (7.15). Вычислите функционал (3.5) на решении (7.18) и убедитесь, что он вообще не зависит от C_4. Чему он равен? □

7.3. Вопросы для самопроверки

1. Какую вариационную задачу мы решаем?
2. Является ли класс допустимых функций в данной задаче более широким или более узким, чем в задачах, которые мы рассматривали в *главах 2—5*?
3. Где меньше минимальное значение функционала: в задаче с закрепленным концом или со свободным?
4. Выведите естественные граничные условия (7.15).
5. Выведите естественное граничное условие для функционала (4.1), зависящего от функции и ее производных до 2-го порядка включительно, если граничные условия вида (4.2) заданы только на левом конце интервала x_1, а на правом конце x_2 заданы:
   - только значение функции (вариант 1);
   - только значение производной (вариант 2);
   - вообще ничего не задано (вариант 3).

7.4. Примеры выполнения заданий

7.4.1. Задание 1

Найти экстремаль функционала, рассмотренного ранее в задании 1 *главы 2*, при том же самом граничном условии на левом конце и при незаданном граничном условии на правом конце. Сравнить решение с решением задания 1 *главы 2*.

Составим программу для решения этой задачи. Так как нам нужно сравнить решение с заданием 1 *главы 2*, воспользуемся программой для этого примера как заготовкой. Внесем в нее такие изменения:

- изменим заголовок задачи: теперь это задание 7.1, а не 2.1;
- уберем печати всех промежуточных результатов;
- добавим печать естественного граничного условия (7.4);
- вычислим значение функционала только на экстремали — решении задания 2.1 и обозначим его `J21`;
- на прямой M_1M_2 вычислять значение функционала не будем;
- не рисуем кривую — потом нарисуем все вместе.

```
clear all % очистили память
disp('Решаем задание 7.1') % выводим заголовок задачи
syms x y Dy D2y % описали символические переменные
F=x^2+y^2+Dy^2; % подынтегральная функция
x1=-1; % граничные условия
y1=1;
x2=1;
y2=2;
disp('Исходные данные:')
fprintf('Подынтегральная функция F(x,y,y'')=%s\n',char(F))
fprintf('Граничное условие слева: y(%d)=%d\n',x1,y1)
fprintf('Граничное условие справа: y(%d)=%d\n',x2,y2)
dFdy=diff(F,y); % вычислили Fy
dFdy1=diff(F,Dy); % вычислили Fy'
disp('Естественное граничное условие:')
fprintf('%s=0\n',char(dFdy1))
d_dFdy1_dx=diff(dFdy1,x); % Fy'x
d_dFdy1_dy=diff(dFdy1,y); % Fy'y
d_dFdy1_dy1=diff(dFdy1,Dy); % Fy'y'-условие Лежандра
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+d_dFdy1_dy1*D2y;
Euler=simple(dFdy-dFy1dx); % уравнение Эйлера
deqEuler=[char(Euler) '=0']; % добавили =0
Sol=dsolve(deqEuler,'x'); % решаем уравнение Эйлера
if length(Sol)~=1 % решений нет или более одного
  error('Нет решений нет или более одного решения!');
end
SolLeft=subs(Sol,x,x1); % подставили x1
SolRight=subs(Sol,x,x2); % подставили x2
EqLeft=[char(SolLeft) '=' char(sym(y1))]; % =y1
EqRight=[char(SolRight) '=' char(sym(y2))]; % =y2
Con=solve(EqLeft,EqRight,'C1,C2'); % решаем систему
```

```
C1=Con.C1; % присваиваем полученные решения
C2=Con.C2; % символическим константам C1 и C2
Sol21=vpa(eval(Sol),14); % подставили C1,C2
F21=subs(F,{y,Dy},{Sol21,diff(Sol21,x)});
J21=eval(int(F21,x,x1,x2))
xpl=linspace(x1,x2); % задаем массив абсцисс
y21=subs(Sol21,x,xpl); % вычислили ординаты
Решаем задание 7.1
Исходные данные:
Подынтегральная функция F(x,y,y')=x^2+y^2+Dy^2
Граничное условие слева: y(-1)=1
Граничное условие справа: y(1)=2
Естественное граничное условие:
2*Dy=0
J21 =
   4.75035801121746
```

Теперь переходим к решению нашего задания. Аналитическое решение дифференциального уравнения Эйлера получено — оно такое же, как и для задания 2.1. Для нахождения произвольных постоянных нам нужно сформировать систему уравнений. Граничное условие слева — такое же. Формируем естественное граничное условие на правом конце. Для этого в формулу (7.4) подставляем сначала y и y', а затем x_2 во все выражения вместо x.

```
dydx=diff(Sol,x); % dy/dx
disp('Производная от общего решения:')
fprintf('%s\n',char(dydx))
RightNat=subs(dFdy1,{y,Dy},{Sol,dydx}); % подставляем y, y'
SolRightNat=subs(RightNat,x,x2); % подставили x2
EqRightNat=[char(vpa(SolRightNat,14)) '=0'];
disp('Естественные граничные условия:')
fprintf('%s\n',EqLeft,EqRightNat)
Производная от общего решения:
C1*cosh(x)+C2*sinh(x)
Естественные граничные условия:
-C1*sinh(1)+C2*cosh(1)=1
3.0861612696304*C1+2.3504023872876*C2=0
```

Решаем систему уравнений `EqLeft` и `EqRightNat` — находим произвольные постоянные. Подставляем их в решение. Находим значение функционала на полученной экстремали.

```
Con=solve(EqLeft,EqRightNat,'C1,C2');
C1=Con.C1;
C2=Con.C2;
```

```
Sol71=vpa(eval(Sol),14); % подставили C1, C2
disp('Уравнение экстремали:')
fprintf('y(x)=%s\n',char(Sol71))
F71=subs(F,{y,Dy},{Sol71,diff(Sol71,x)});
J71=eval(int(F71,x,x1,x2))
Уравнение экстремали:
y(x)=-.31237109659900*sinh(x)+.41015427200459*cosh(x)
J71 =
   1.63069424674247
```

Получилось ли у вас `J71<J21`? Если нет, то почему? Проанализируйте поведение функционала (его величину) на кривых, близких к экстремалям — решениям заданий 2.1 и 7.1.

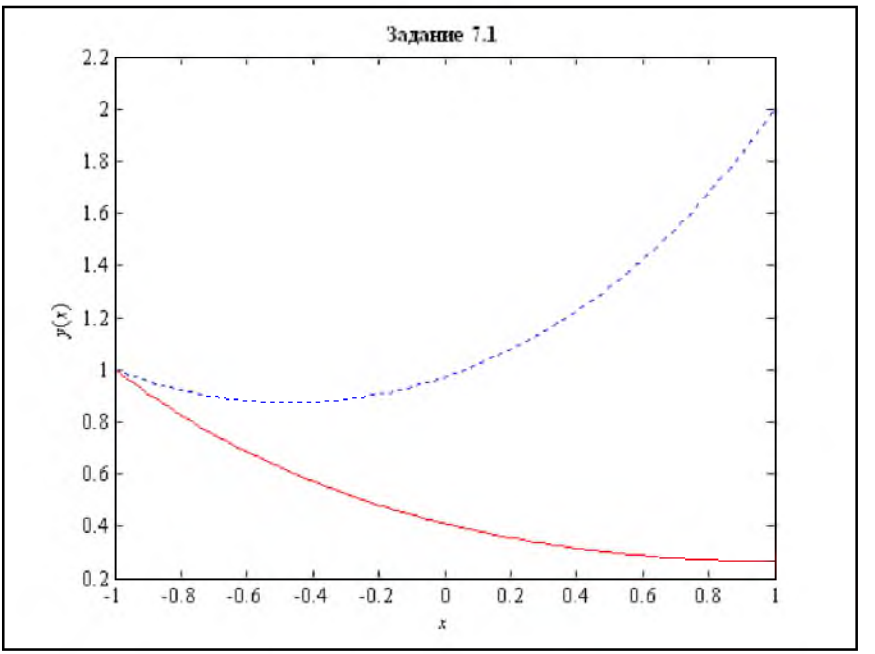


Рис. 7.3. Решение задания 7.1

Далее, как и в задании 2.1, заполняем таблицу и строим графики решений двух заданий на одном рисунке (рис. 7.3): красная сплошная линия — решение нашего задания 7.1, а синяя пунктирная — решение задания 2.1.

```
y71=subs(Sol71,x,xpl); % подставили x
figure % фигура
plot(xpl,y21,':b',xpl,y71,'-r') % 2 графика на 1 рисунке
```

```
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 7.1') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
```

7.4.2. Задание 2

Найти экстремаль функционала (3.14), рассмотренного ранее в *главе 3*, при тех же самых граничных условиях на левом конце и при незаданных граничных условиях на правом конце. Сравнить решение с решением задания из *главы 3*.

Программу для решения этого примера будем составлять на основе программы для задания из *главы 3*. Внесем в нее такие же изменения, как и в программу предыдущего задания 7.1. А именно: уберем все печати, кроме исходных данных и естественных граничных условий $F_{y'}$ и $F_{z'}$. Значение функционала вычислим только на экстремали — решении задания 3. Графики также пока не строим.

```
clear all
disp('Решаем задание 7.2') % выводим заголовок задачи
syms x y z Dy D2y Dz D2z % описали переменные
F=Dy^2+Dz^2+2*y*z;
x1=-2;
y1=1;
z1=0;
x2=2;
y2=0;
z2=2;
disp('Исходные данные:')
fprintf('F(x,y,y'',z,z'')=%s\n',char(F))
disp('Граничные условия слева:')
fprintf('y(%d)=%d;   z(%d)=%d\n',x1,y1,x1,z1)
disp('Граничные условия справа:')
fprintf('y(%d)=%d;   z(%d)=%d\n',x2,y2,x2,z2)
dFdy=diff(F,y);
dFdy1=diff(F,Dy);
d_dFdy1_dx=diff(dFdy1,x);
d_dFdy1_dy=diff(dFdy1,y);
d_dFdy1_dy1=diff(dFdy1,Dy);
d_dFdy1_dz=diff(dFdy1,z);
d_dFdy1_dz1=diff(dFdy1,Dz);
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+...
  d_dFdy1_dy1*D2y+d_dFdy1_dz*Dz+d_dFdy1_dz1*D2z;
```

```
dFdz=diff(F,z);
dFdz1=diff(F,Dz);
d_dFdz1_dx=diff(dFdz1,x);
d_dFdz1_dy=diff(dFdz1,y);
d_dFdz1_dy1=diff(dFdz1,Dy);
d_dFdz1_dz=diff(dFdz1,z);
d_dFdz1_dz1=diff(dFdz1,Dz);
dFz1dx=d_dFdz1_dx+d_dFdz1_dy*Dy+...
  d_dFdz1_dy1*D2y+d_dFdz1_dz*Dz+d_dFdz1_dz1*D2z;
disp('Естественные граничные условия на правом конце:')
fprintf('%s=0\n',char(dFdy1),char(dFdz1))
EulerY=simple(dFdy-dFy1dx);
EulerZ=simple(dFdz-dFz1dx);
dEuY=[char(EulerY) '=0']; % уравнение Y
dEuZ=[char(EulerZ) '=0']; % уравнение Z
Sol=dsolve(dEuY,dEuZ,'x'); % решаем
if length(Sol)~=1, % решений нет или более одного
  error('Решений нет или более одного!');
else
  SolY=Sol.y;
  SolZ=Sol.z;
end
SolLY=subs(SolY,x,x1); % x1 в y
SolLZ=subs(SolZ,x,x1); % x1 в z
SolRY=subs(SolY,x,x2); % x2 в y
SolRZ=subs(SolZ,x,x2); % x2 в z
EqLY=[char(vpa(SolLY,14)) '=' char(sym(y1))];
EqLZ=[char(vpa(SolLZ,14)) '=' char(sym(z1))];
EqRY=[char(vpa(SolRY,14)) '=' char(sym(y2))];
EqRZ=[char(vpa(SolRZ,14)) '=' char(sym(z2))];
Con=solve(EqLY,EqLZ,EqRY,EqRZ,'C1,C2,C3,C4');
C1=Con.C1;
C2=Con.C2;
C3=Con.C3;
C4=Con.C4;
Sol3Y=vpa(eval(SolY),14);
Sol3Z=vpa(eval(SolZ),14);
F3=simple(subs(F,{y,z,Dy,Dz},...
  {Sol3Y,Sol3Z,diff(Sol3Y,x),diff(Sol3Z,x)}));
J3=eval(int(F3,x,x1,x2)) % значение функционала
xpl=linspace(x1,x2); % массив абсцисс
y3=subs(Sol3Y,x,xpl); % вычислили ординаты
z3=subs(Sol3Z,x,xpl); % вычислили аппликаты
```

```
Решаем задание 7.2
Исходные данные:
F(x,y,y',z,z')=Dy^2+Dz^2+2*y*z
Граничные условия слева:
y(-2)=1;   z(-2)=0
Граничные условия справа:
y(2)=0;   z(2)=2
Естественные граничные условия на правом конце:
2*Dy=0
2*Dz=0
J4 =
   1.94492120385672
```

Формируем естественные граничные условия (7.15): подставляем в $F_{y'}$ и $F_{z'}$ вычисленные y, z, y', z', а затем $x=x_2$, и приравниваем полученные выражения нулю.

```
dydx=diff(SolY,x);
dzdx=diff(SolZ,x);
disp('Производные от общего решения:')
fprintf('y''(x)=%s\nz''(x)=%s\n',char(dydx),char(dzdx))
rnY=subs(dFdy1,{y,z,Dy,Dz},{SolY,SolZ,dydx,dzdx});
rnZ=subs(dFdz1,{y,z,Dy,Dz},{SolY,SolZ,dydx,dzdx});
SolRNY=subs(rnY,x,x2); % подставили x2
SolRNZ=subs(rnZ,x,x2);
EqRNY=[char(vpa(SolRNY,14)) '=0'];
EqRNZ=[char(vpa(SolRNZ,14)) '=0'];
disp('Естественные граничные условия:')
fprintf('%s\n',EqRNY,EqRNZ)
Производные от общего решения:
y'(x)=-1/4*C1*exp(-x)+1/4*C1*exp(x)-1/2*C1*sin(x)+1/4*C2*exp(-x)+
  1/4*C2*exp(x)+1/2*C2*cos(x)-1/4*C3*exp(-x)+1/4*C3*exp(x)+1/2*C3*sin(x)-
  1/2*C4*cos(x)+1/4*C4*exp(x)+1/4*C4*exp(-x)
z'(x)=-1/4*C1*exp(-x)+1/4*C1*exp(x)+1/2*C1*sin(x)-1/2*C2*cos(x)+
  1/4*C2*exp(x)+1/4*C2*exp(-x)-1/4*C3*exp(-x)+1/4*C3*exp(x)-
  1/2*C3*sin(x)+1/4*C4*exp(-x)+1/4*C4*exp(x)+1/2*C4*cos(x)
Естественные граничные условия:
2.7175629810214*C1+3.3460488545366*C2+
  4.5361578346728*C3+4.1783425276309*C4=0
4.5361578346728*C1+4.1783425276309*C2+
  2.7175629810214*C3+3.3460488545366*C4=0
```

Решаем полученную систему уравнений и подставляем значения найденных констант в $y(x)$ и $z(x)$. Печатаем найденные решения. Вычисляем значение функционала на экстремали.

```
Con=solve(EqLY,EqLZ,EqRNY,EqRNZ,'C1,C2,C3,C4');
C1=Con.C1;
C2=Con.C2;
C3=Con.C3;
C4=Con.C4;
Sol72Y=vpa(eval(SolY),14);
Sol72Z=vpa(eval(SolZ),14);
disp('Найдена экстремаль:')
fprintf('y(x)=%s\nz(x)=%s\n',char(Sol72Y),char(Sol72Z))
F72=simple(subs(F,{y,z,Dy,Dz},...
  {Sol72Y,Sol72Z,diff(Sol72Y,x),diff(Sol72Z,x)}));
J72=eval(int(F72,x,x1,x2))
Найдена экстремаль:
y(x)=.67644949265860e-1*exp(-1.*x)+.12389604634005e-2*exp(x)+
  .31832853810872*cos(x)-.69556054538131*sin(x)
z(x)=.67644949265860e-1*exp(-1.*x)+.12389604634005e-2*exp(x)-
  .31832853810872*cos(x)+.69556054538131*sin(x)
J72 =
  -0.07924599130544
```

Выполняется ли в вашем варианте J72<J3? Если нет, то проанализируйте поведение функционала на кривых, близких к экстремалям — решениям заданий 3 и 7.2.

Заполняем таблицу для построения графика. Строим на трех отдельных графиках результаты вычислений. Вначале (на рис. 7.4) строим двумерные графики функций $y(x)$: сплошной красной линией для нашего задания 7.2, а пунктирной синей — для задания из *главы 3*. Затем на новом рисунке (рис. 7.5) строим двумерные графики функций $z(x)$: также сплошной красной линией для нашего задания, а пунктирной синей — для задания *главы 3*. И, наконец, строим трехмерный график (рис. 7.6): сплошная красная линия — решение нашего задания, а пунктирная синяя — задание *главы 3*. Выбираем точку просмотра. Показываем сетку и контур.

```
y72=subs(Sol72Y,x,xpl);
z72=subs(Sol72Z,x,xpl);
figure % фигура
plot(xpl,y3,':b', xpl,y72,'-r')
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 7.2 - функции \rm\ity\rm(\itx\rm)')
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
figure % фигура
plot(xpl,z3,':b', xpl,z72,'-r')
```

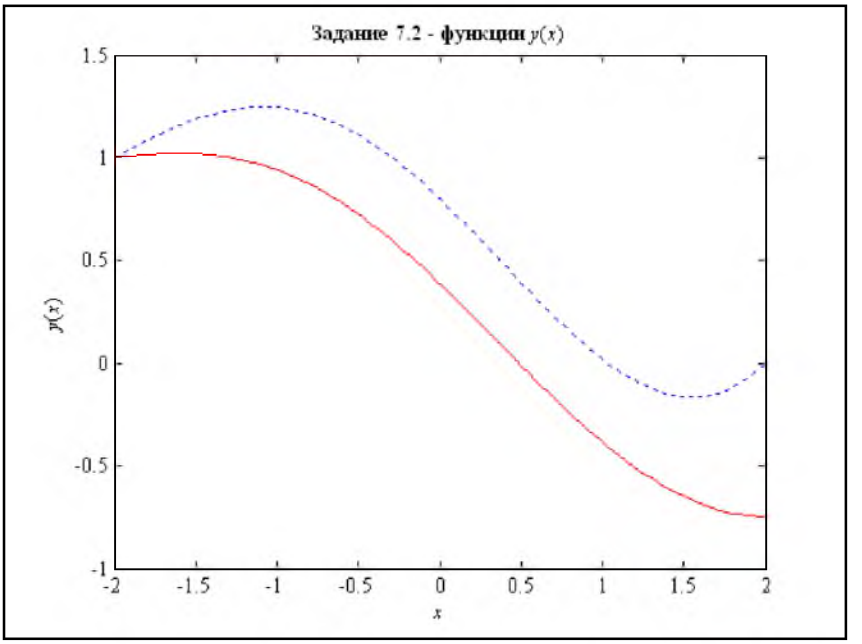


Рис. 7.4. Решение задания 7.2 — функции $y(x)$

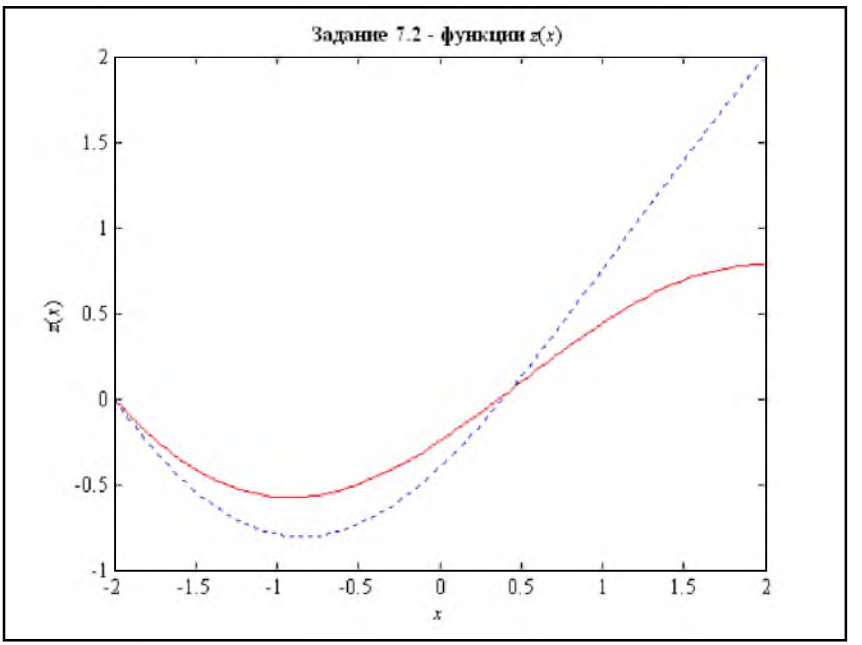


Рис. 7.5. Решение задания 7.2 — функции $z(x)$

```
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 7.2 - функции \rm\itz\rm(\itx\rm)')
xlabel('\itx') % метка оси OX
ylabel('\itz\rm(\itx\rm)') % метка оси OY
figure % фигура
plot3(xpl,y3,z3,':b',xpl,y72,z72,'-r')
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 7.2 - 3D-графики') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
zlabel('\itz\rm(\itx\rm)') % метка оси OZ
view(205,30) % точка просмотра
grid on % показали сетку
box on % показали внешний контур
```

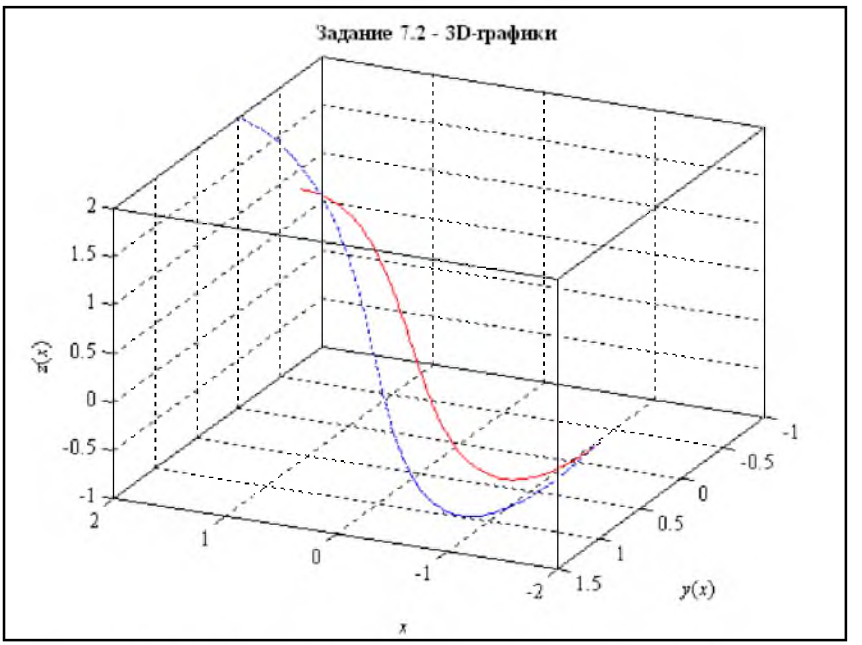


Рис. 7.6. Решение задания 7.2 — пространственные кривые

7.4.3. Задание 3

Решить задачу о брахистохроне, соединяющей точку $M_1(0,0)$ с точкой M_2, у которой задана абсцисса $x_2 = 2$ и не задана ордината y_2. Сравнить решение

полученной задачи с решением задачи при закрепленном правом конце $M_2(2, 1)$. Вычислить время прохождения дистанции лыжником, показанным на рис. 1.6, в том и другом случае. Нарисовать графики.

Мы уже решили эту задачу аналитически (7.14). Нам нужно оформить решение численно и сравнить его с решением задания 3 *главы 2*. Начнем с повторения старого примера: введем исходные данные и повторим решение задания 3 *главы 2*. Чтобы не путать обозначения, во всех переменных, относящихся к этому решению, поставим суффикс 23.

```
clear all % очистили все
disp('Решаем пример 7.3') % заголовок задачи
x2=2;
y2=1;
fprintf('Условие на правом конце: y(%d)=%d\n',x2,y2)
eq1=['C1*(t2-sin(t2))=' num2str(x2)];
eq2=['C1*(1-cos(t2))=' num2str(y2)];
Sol=solve(eq1,eq2,'C1,t2');
C1=eval(Sol.C1);
t2=eval(Sol.t2);
t23=linspace(0,t2); % задали массив параметров
x23=C1*(t23-sin(t23)); % вычислили абсциссы
y23=C1*(1-cos(t23)); % вычислили ординаты
Решаем пример 7.3
Условие на правом конце: y(2)=1
```

Теперь найдем решение нашей задачи: запрограммируем вычисления по формуле (7.14). Напечатаем найденное решение.

```
t73=linspace(0,pi);
x73=x2/pi*(t73-sin(t73));
y73=x2/pi*(1-cos(t73));
disp('Брахистохрона при подвижном правом конце:')
fprintf('x(t)=%12.6f*(t-sin(t))\n',x2/pi)
fprintf('y(t)=%12.6f*(1-cos(t))\n',x2/pi)
Брахистохрона при подвижном правом конце:
x(t)=    0.636620*(t-sin(t))
y(t)=    0.636620*(1-cos(t))
```

Вычислим значение функционала (2.74) для обоих вариантов. Решение у нас получено в параметрической форме, поэтому дифференциал длины дуги будет иметь несколько иной вид, чем при явном задании. Вспомните, как он записывается для функции, заданной параметрически. Описываем необходимые символические переменные, строим из них подынтегральную функцию. Вычисляем функционалы для обоих вариантов и печатаем их.

```
g=9.81; % ускорение силы тяжести
syms t R % описали символические переменные
x=R*(t-sin(t)); % символические уравнения
y=R*(1-cos(t)); % брахистохроны
F=simple(((diff(x,t)^2+diff(y,t)^2)/...
  (2*g*y))^0.5); % подынтегральная функция
F23=subs(F,R,C1); % функция для задания 2.3
J23=eval(int(F23,t,0,t2)); % время для задания 2.3
F73=subs(F,R,x2/pi); % функция для задания 7.3
J73=eval(int(F73,t,0,pi)); % время для задания 7.3
disp('Время прохождения дистанции:')
fprintf('T23=%12.8f c\nT73=%12.8f c\n',J23,J73)
Время прохождения дистанции:
T23=  0.80556383 c
T73=  0.80030482 c
```

Какое время оказалось меньше?

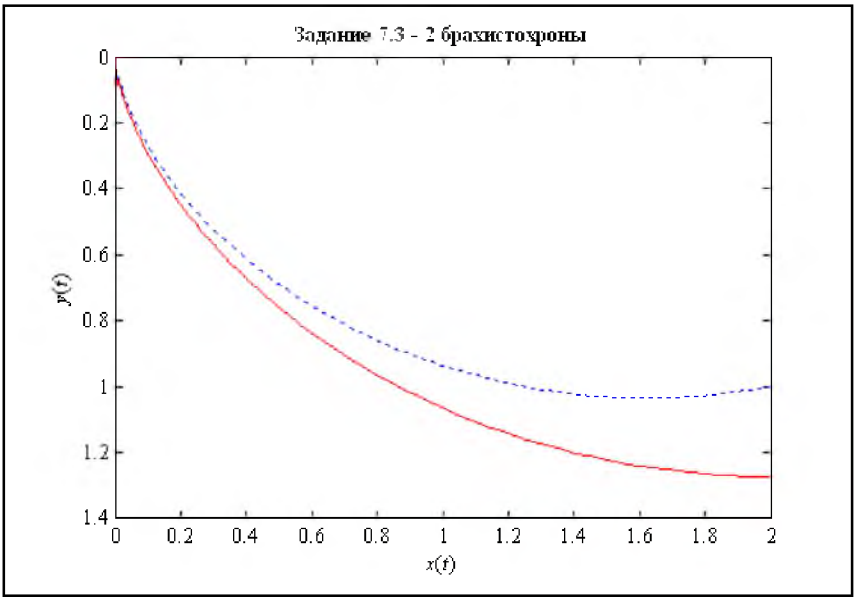


Рис. 7.7. Решение задания 7.3 — две брахистохроны

Теперь рисуем обе брахистохроны на одном графике (рис. 7.7). Как обычно, синей пунктирной линией рисуем решение старого *задания 2.3*, а красной сплошной — нашего 7.3.

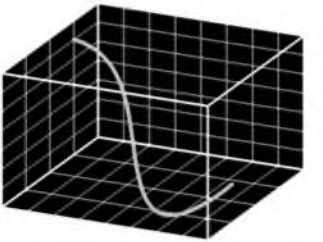
```
figure % фигура
plot(x23,y23,':b',x73,y73,'-r') % рисуем график
```

```
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
axis ij % установили 0 в левом верхнем углу
xlim([0 x2]) % установили пределы по оси 0x
da=daspect;
da(1:2)=min(da(1:2));
daspect(da); % выравниваем масштаб
title ('\bfЗадание 7.3 - 2 брахистохроны')
xlabel ('\itx\rm(\itt\rm)') % ось OX
ylabel ('\ity\rm(\itt\rm)') % ось OY
```

7.5. Задание

1. Для своего варианта *задания 1 главы 2* найти экстремаль, если y_2 не задано. Сравнить решение с решением *задания 1 главы 2*. Вычислить значение функционала на найденном решении.
2. Для своего варианта *задания главы 3* найти экстремаль, если y_2 и z_2 не заданы. Сравнить решение с решением *задания главы 3*. Вычислить значение функционала на найденном решении.
3. Для своего варианта брахистохроны из *главы 2* найти экстремаль, если y_2 не задано. Сравнить решение с решением *задания 3 главы 2*. Вычислить время прохождения дистанции в обоих случаях.

ГЛАВА 8



Условия трансверсальности

8.1. Условия трансверсальности в элементарной задаче вариационного исчисления

Вернемся вновь к элементарной задаче вариационного исчисления. Исследуем на экстремум функционал (2.1). Но пусть граничное условие вида (2.2) задано только на левом конце: $y(x_1)=y_1$; а на правом конце x_2 (неизвестном!) график функции проходит через точку $M_2(x_2, y_2)$, лежащую на заданной линии $y=\varphi(x)$. На рис. 8.1 показаны экстремаль (жирная линия) и допустимые функции (тонкие линии). Штриховой линией показан график функции $y=\varphi(x)$.

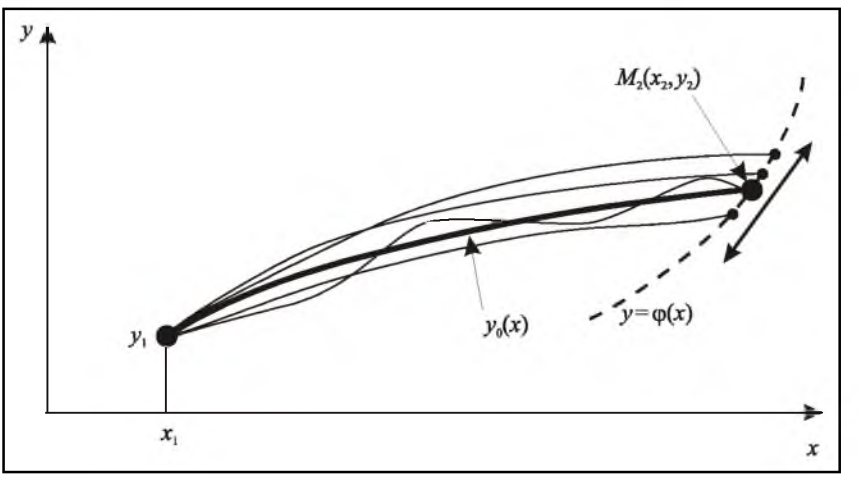


Рис. 8.1. Вариационная задача с подвижным правым концом

Это — обобщение задачи из предыдущей *главы 7*: здесь правая точка может двигаться не по вертикали, как на рис. 7.1, а по заданной линии.

Выведем необходимое условие экстремума для такого функционала. Дадим малое приращение (вариацию) функции на экстремали — δy — и вычислим приращение функционала $\Delta J(y_0)$ на этой экстремали. В отличие от (2.7), здесь приращение получает также и правая точка x_2, поэтому

$$\Delta J(y_0) = \int_{x_1}^{x_2+\delta x_2} F(x, y_0 + \delta y, y_0' + \delta y')\, dx - \int_{x_1}^{x_2} F(x, y_0, y_0')\, dx. \qquad (8.1)$$

Разобьем первый интеграл на 2 слагаемых: по отрезкам $[x_1, x_2]$ и $[x_2, x_2+\delta x_2]$:

$$\Delta J(y_0) = \int_{x_1}^{x_2} \left(F(x,\, y_0 + \delta y_0,\, y_0' + \delta y_0') - F(x,\, y_0,\, y_0')\right) dx + \\ + \int_{x_2}^{x_2+\delta x_2} F(x,\, y_0 + \delta y_0,\, y_0' + \delta y_0')\, dx\,. \qquad (8.2)$$

С первым интегралом мы уже несколько раз встречались. Применим для его преобразования первый способ вычисления вариации: разложим первое его слагаемое (с приращениями функций) в ряд Тейлора и удержим только линейные члены. Тем самым мы перейдем от приращения функционала $\Delta J(y_0)$ к линейной части этого приращения — вариации $\delta J(y_0)$. Таким образом, первое слагаемое (обозначим его $\delta J_1(y_0)$) будет иметь такой же вид, как и (7.3):

$$\delta J_1(y_0) = \left(F_{y'}\,\delta y\right)\Big|_{x_2} + \int_{x_1}^{x_2} \left(F_y - \frac{dF_{y'}}{dx}\right)\delta y\, dx\,. \qquad (8.3)$$

Для вычисления второго интеграла в (8.2) воспользуемся теоремой о среднем. По этой теореме интеграл на любом интервале равен подынтегральной функции в некоторой средней точке интервала, умноженной на ширину интервала. У нас интервал $[x_2, x_2 + \delta x_2]$ — малый, подынтегральная функция — непрерывная, поэтому с точностью до бесконечно малых величин высшего порядка малости (нам ведь нужна только линейная часть!) мы можем вычислить подынтегральную функцию в левой точке: при $x = x_2$. Более того, мы можем и подынтегральную функцию F вычислять на кривой без вариации, т. к. учет δy и $\delta y'$ приведет только лишь к появлению малых высшего порядка малости. Таким образом, с точностью до бесконечно малых величин высшего порядка малости можно считать, что:

$$\delta J_2(y_0) = F(x, y, y')\Big|_{x=x_2}\delta x_2\,. \qquad (8.4)$$

Теперь складываем (8.3) и (8.4) — получаем вариацию функционала для нашей задачи:

$$\delta J(y_0) = \left(F_{y'}\,\delta y\right)\Big|_{x_2} + F\big|_{x=x_2}\delta x_2 + \int_{x_1}^{x_2}\left(F_y - \frac{dF_{y'}}{dx}\right)\delta y\,dx\,. \tag{8.5}$$

Преобразуем первые 2 слагаемых. Из рис. 8.2 видно, что величина $\delta y\big|_{x=x_2}$ связана с δy_2 соотношением

$$\delta y\big|_{x=x_2} = \delta y_2 - y'(x_2)\,\delta x_2\,. \tag{8.6}$$

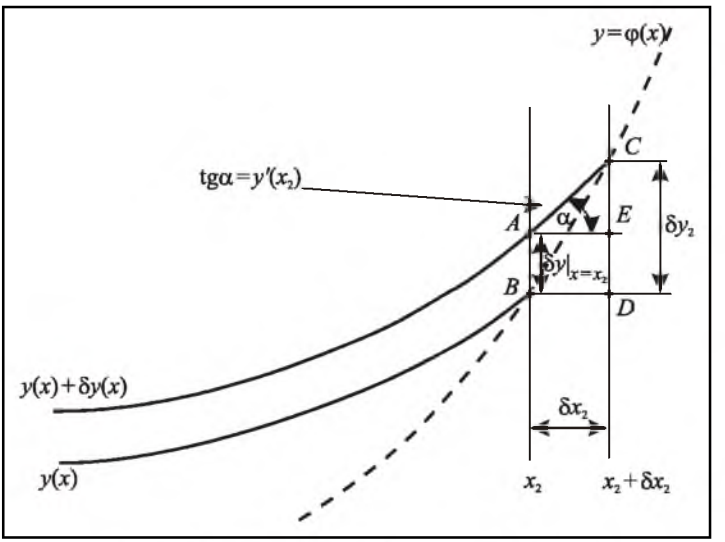


Рис. 8.2. Связь между $\delta y\big|_{x=x_2} = AB$ и $\delta y_2 = CD$

Поэтому в (8.5) первые 2 слагаемых объединяются:

$$\begin{aligned}&\left(F_{y'}\delta y\right)\Big|_{x_2} + F\big|_{x=x_2}\delta x_2 = F_{y'}\big|_{x=x_2}\left(\delta y_2 - y'(x_2)\,\delta x_2\right) + F\big|_{x=x_2}\delta x_2 = \\ &= \left(F + (\varphi' - y')F_{y'}\right)\Big|_{x=x_2}\delta x_2\,.\end{aligned} \tag{8.7}$$

Окончательно получаем выражение для вариации функционала на нашей экстремали:

$$\delta J(y_0) = \left(F + (\varphi' - y')F_{y'}\right)\Big|_{x=x_2}\delta x_2 + \int_{x_1}^{x_2}\left(F_y - \frac{dF_{y'}}{dx}\right)\delta y\,dx = 0. \tag{8.8}$$

Проанализируем полученное выражение так же, как мы это делали в *главе 7*. Может ли (8.8) обращаться в нуль за счет компенсации слагаемых? Если мы будем варьировать функцию $y(x)$ так, что правая точка не будет изменяться, то в (8.8) 1-е слагаемое не изменится, а 2-е будет меняться. Но нам нужно, чтобы (8.8) обращалось в нуль при любых $\delta y(x)$! Поэтому мы должны сделать вывод, что в (8.8) *обязательно каждое слагаемое должно обращаться в нуль в отдельности*. Или, как мы говорили в *главе 7*, если функционал достигает экстремума на классе функций с подвижной правой точкой, то он тем более должен достигать экстремума и на более узком классе функций: с неподвижной правой точкой.

Из равенства нулю 2-го слагаемого (8.8) по основной лемме вариационного исчисления следует, что подынтегральная функция должна удовлетворять дифференциальному уравнению Эйлера (2.9), а из равенства нулю 1-го слагаемого (8.8) получаем: т. к. δx_2 — произвольная, то должно обращаться в нуль выражение:

$$\left(F+\left(\varphi'-y'\right)F_{y'}\right)\Big|_{x=x_2}=0. \tag{8.9}$$

Определение 8.1. Выражение вида (8.9) называется *условием трансверсальности*. □

Оно фактически является недостающим граничным условием на правом конце. Смысл его такой: если двигать точку $M_2(x_2, y_2)$ по линии $y=\varphi(x)$ и по полученным двум точкам строить экстремали (решения уравнения Эйлера), то из всех экстремалей доставлять экстремум функционалу будет та, которая удовлетворяет условию трансверсальности (8.9).

Таким образом, для решения данной вариационной задачи нужно решить дифференциальное уравнение Эйлера, а затем найти произвольные постоянные C_1, C_2 и координаты точки $M_2(x_2, y_2)$ из решения системы 4-х нелинейных уравнений

$$\begin{cases} y\left(x_1,C_1,C_2\right)=y_1; \\ y\left(x_2,C_1,C_2\right)=y_2; \\ y_2=\varphi\left(x_2\right); \\ \left(F+\left(\varphi'-y'\right)F_{y'}\right)\Big|_{x=x_2}=0. \end{cases} \tag{8.10}$$

Но, конечно, можно найти решение этой вариационной задачи и по-другому. Если задать пробное значение x_2, то по нему можно найти $y_2=\varphi(x_2)$, а затем по известным граничным условиям построить решение уравнения Эйлера $y=y(x, C_1, C_2)$. На этой кривой функционал J принимает некоторое значение, которое, в конечном счете, является функцией x_2. Таким образом, вариацион-

ная задача сводится к исследованию на экстремум функции одной переменной $J(x_2)$. Если искать этот экстремум аналитически (находить производную и приравнивать ее нулю), то мы и получим условие трансверсальности (8.9). Если же решать эту задачу численно, то вместо системы 4-х уравнений (8.10) мы получаем экстремум функции одной переменной.

Аналогично можно вывести естественное граничное условие на левом конце (если таковое отсутствует), а также оба.

ПРИМЕР 8.1. Найти экстремаль функционала

$$J(y)=\int_0^{x_2}\frac{\sqrt{1+y'^2}}{y}dx \tag{8.11}$$

при граничных условиях

$$\begin{cases} y(0)=0; \\ y_2=x_2-5. \end{cases} \tag{8.12}$$

Здесь левая точка — неподвижная, а правая движется по заданной линии $y=x-5$. Решаем вначале дифференциальное уравнение Эйлера. Подынтегральная функция не зависит явно от x, поэтому записываем 1-й интеграл в виде (2.64):

$$\begin{gathered} F-y'F_{y'}=C_1;\quad \frac{\sqrt{1+y'^2}}{y}-y'\frac{y'}{y\sqrt{1+y'^2}}=C_1; \\ \frac{1+y'^2-y'^2}{y\sqrt{1+y'^2}}=C_1;\quad y\sqrt{1+y'^2}=C_1. \end{gathered} \tag{8.13}$$

Решаем полученное уравнение подстановкой $y'=-\operatorname{ctg} t$. Находим параметрическое выражение для $y(t)$:

$$y\frac{1}{\sin t}=C_1;\quad y(t)=C_1\sin t. \tag{8.14}$$

Затем вычисляем dx:

$$y'=\frac{dy}{dx}=-\operatorname{ctg} t;\quad dx=\frac{dy}{-\operatorname{ctg} t}=\frac{C_1\cos t dt}{-\operatorname{ctg} t}=-C_1\sin t dt; \tag{8.15}$$

и параметрическое выражение для $x(t)$ будет:

$$x(t)=C_1\cos t+C_2. \tag{8.16}$$

Исключая из (8.14) и (8.16) параметр t, получим:

$$(x-C_2)^2+y^2=C_1^2. \tag{8.17}$$

Таким образом, экстремали — это окружности с центрами на оси Ox. Для нахождения произвольных постоянных C_1, C_2 и координат правой точки x_2, y_2 записываем систему уравнений вида (8.10):

$$\begin{cases} C_2^2=C_1^2; \\ (x_2-C_2)^2+y_2^2=C_1^2; \\ y_2=x_2-5; \\ \left(F+(\varphi'-y')F_{y'}\right)\Big|_{x=x_2}=0. \end{cases} \tag{8.18}$$

Четвертое уравнение мы пока что не вывели, но сейчас выведем. Вначале упростим первые три. Из 1-го уравнения следует, что $C_2=\pm C_1$. По смыслу задачи $C_1=R>0$ — радиус окружности, и центр окружности также находится справа от 0, поэтому $C_2=R>0$. Следовательно, 1-е уравнение из (8.18) можно исключить, заменив в остальных $C_1=C_2=R$.

Выводим теперь необходимые данные для 4-го уравнения. Из правого граничного условия (8.12) имеем: $\varphi'=1$. Найдем y', дифференцируя неявно заданную функцию (8.17):

$$2(x-R)+2yy'=0; \quad y'=\frac{R-x}{y}. \tag{8.19}$$

Теперь выводим 4-е уравнение системы (8.18):

$$\begin{aligned} F+(\varphi'-y')F_{y'} &= \frac{\sqrt{1+y'^2}}{y}+(1-y')\frac{y'}{y\sqrt{1+y'^2}}= \\ =\frac{1+y'^2+y'-y'^2}{y\sqrt{1+y'^2}} &= \frac{1+y'}{y\sqrt{1+y'^2}}=0. \end{aligned} \tag{8.20}$$

Из четырех уравнений системы (8.18) остается три:

$$\begin{cases} (x_2-R)^2+y_2^2=R^2; \\ y_2=x_2-5; \\ y'(x_2)=\dfrac{R-x_2}{y_2}=-1. \end{cases} \tag{8.21}$$

Осталось решить полученную систему из 3-х уравнений с 3-мя неизвестными. Из 2-го и 3-го уравнений следует, что

$$x_2 - y_2 = R = 5. \tag{8.22}$$

Теперь подставляем $R=5$ и $y_2 = x_2 - 5$ в 1-е уравнение и находим x_2:

$$\left(x_2 - 5\right)^2 + \left(x_2 - 5\right)^2 = 25; \quad x_2 = 5 \pm \frac{5}{\sqrt{2}}. \tag{8.23}$$

Таким образом, имеем два решения. Оба они описываются уравнением окружности:

$$(x-5)^2 + y^2 = 25 \tag{8.24}$$

и представляют верхнюю и нижнюю ее части от $x_1 = 0$ до x_2 из (8.23). Графики полученных решений показаны на рис. 8.3. □

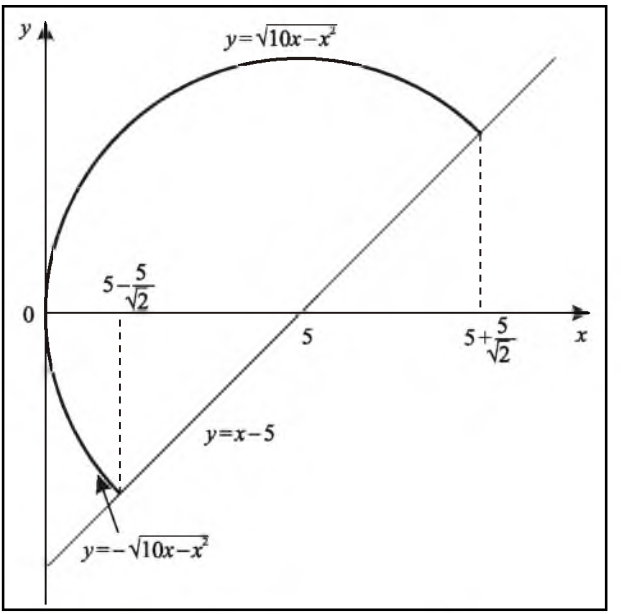


Рис. 8.3. Решение примера 8.1

8.2. Условия трансверсальности для функционала, зависящего от двух функций

Рассмотрим теперь функционал вида (3.1), но для упрощения выкладок пусть он зависит только от двух функций $y(x)$, $z(x)$.

$$J(y,z)=\int_{x_1}^{x_2} F\left(x,y,z,y',z'\right)\,dx \to \text{extr}. \tag{8.25}$$

Пусть для него задано только граничное условие при $x=x_1$, а при $x=x_2$ экстремаль (пространственная линия) проходит через точку $M_2(x_2,y_2,z_2)$, лежащую на заданной поверхности или линии, как показано на рис. 8.4.

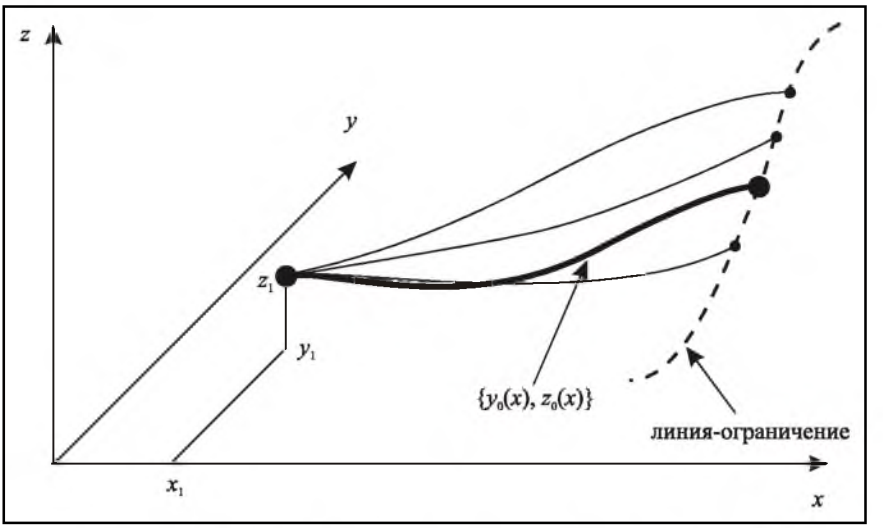


Рис. 8.4. Экстремаль, допустимые функции и линия-ограничение

И в этом случае поступим так же, как и ранее. Составим приращение функционала на нашей экстремали, и найдем его главную, линейную часть — вариацию:

$$\delta J=\int_{x_1}^{x_2+\delta x_2}\left(F\left(x,y_0+\delta y,z_0+\delta z,y_0'+\delta y',z_0'+\delta z'\right)-F\left(x,y_0,z_0,y_0',z_0'\right)\right)dx=$$

$$=\int_{x_1}^{x_2}\left(F_y\delta y+F_z\delta z+F_{y'}\delta y'+F_{z'}\delta z'\right)dx+ \tag{8.26}$$

$$+\int_{x_2}^{x_2+\delta x_2} F\left(x,y_0+\delta y,z_0+\delta z,y_0'+\delta y',z_0'+\delta z'\right)dx.$$

Первый интеграл в (8.26), как и в (8.3), вычисляем с помощью интегрирования по частям:

$$\delta J_1 = \left(F_{y'}\delta y + F_{z'}\delta z\right)\Big|_{x_2} - \int_{x_1}^{x_2}\left(\left(F_y - \frac{dF_{y'}}{dx}\right)\delta y + \left(F_z - \frac{dF_{z'}}{dx}\right)\delta z\right)dx. \qquad (8.27)$$

Второй интеграл в (8.26), как и в (8.4), вычисляем по теореме о среднем: берем подынтегральную функцию без приращений аргументов в точке x_2 и умножаем на малую ширину интервала. С точностью до бесконечно малых величин высшего порядка малости второй интеграл равен:

$$\delta J_2 = F\left(x, y, z, y', z'\right)\Big|_{x=x_2}\delta x_2 . \qquad (8.28)$$

Складываем (8.27) и (8.28) — получаем необходимое условие экстремума нашего функционала (8.25):

$$\begin{aligned}\delta J = &\left(F_{y'}\delta y + F_{z'}\delta z\right)\Big|_{x_2} + F\Big|_{x=x_2}\delta x_2 - \\ &-\int_{x_1}^{x_2}\left(\left(F_y - \frac{dF_{y'}}{dx}\right)\delta y + \left(F_z - \frac{dF_{z'}}{dx}\right)\delta z\right)dx = 0 .\end{aligned} \qquad (8.29)$$

Здесь справедлив тот же самый принцип, о котором мы уже говорили: если какая-либо кривая доставляет экстремум функционалу на классе функций с подвижной правой точкой, то она тем более будет доставлять ему экстремум на более узком классе функций: с неподвижной правой точкой. Следовательно, в (8.29) обязательно должен обращаться в нуль последний интеграл, и обязательно — сумма первых двух слагаемых. Из обращения в нуль интеграла в (8.29) следует (в силу основной леммы вариационного исчисления), что экстремаль должна удовлетворять системе дифференциальных уравнений Эйлера вида (3.4). А сумма первых двух слагаемых дает:

$$\left(F_{y'}\delta y + F_{z'}\delta z\right)\Big|_{x_2} + F\Big|_{x=x_2}\delta x_2 = 0 . \qquad (8.30)$$

Учитывая, что соотношение (8.6) имеет место и для функции $z(x)$:

$$\delta z\Big|_{x=x_2} = \delta z_2 - z'\left(x_2\right)\delta x_2 , \qquad (8.31)$$

запишем результат:

$$\left(F - y'F_{y'} - z'F_{z'}\right)\Big|_{x=x_2}\delta x_2 + F_{y'}\Big|_{x=x_2}\delta y_2 + F_{z'}\Big|_{x=x_2}\delta z_2 = 0. \qquad (8.32)$$

Отсюда можно вывести условия трансверсальности для различных случаев граничных условий на правом конце.

Пусть, например, правая точка $M_2(x_2, y_2, z_2)$ лежит на заданной поверхности $z = \varphi(x, y)$. В этом случае в выражении (8.32) δx_2 и δy_2 независимые, а δz_2 находится дифференцированием уравнения поверхности:

$$\delta z_2 = \varphi_x'(x_2, y_2)\delta x_2 + \varphi_y'(x_2, y_2)\delta y_2. \tag{8.33}$$

Подставим (8.33) в (8.32) и приравняем нулю коэффициенты при независимых δx_2 и δy_2. Получаем условия трансверсальности для этого случая:

$$\begin{cases} \left(F - y'F_{y'} + \left(\varphi_x' - z'\right)F_{z'}\right)\Big|_{x=x_2} = 0; \\ \left(F_{y'} + F_{z'}\varphi_y'\right)\Big|_{x=x_2} = 0. \end{cases} \tag{8.34}$$

Для нахождения 4-х произвольных постоянных и 3-х координат точки M_2 мы имеем систему семи нелинейных уравнений:

- ❒ два граничных условия на левом конце;
- ❒ два граничных условия на неизвестном правом конце;
- ❒ точка M_2 находится на поверхности $z = \varphi(x, y)$;
- ❒ два условия трансверсальности (8.34).

Смысл условий трансверсальности (8.34) следующий: если правую точку двигать по поверхности $z = \varphi(x, y)$, и по полученным двум точкам строить экстремаль, то из всех этих экстремалей самое экстремальное значение функционалу (8.25) доставит та пространственная кривая, для которой выполняются условия (8.34).

Поэтому вместо решения системы семи нелинейных уравнений (8.34) можно, как и в случае (8.10), исследовать на экстремум функционал J, зависящий от двух переменных x_2, y_2: по ним вначале находим $z_2 = \varphi(x_2, y_2)$; затем по известным теперь граничным условиям строим решения уравнений Эйлера $y(x, C_1, C_2, C_3, C_4)$ и $z(x, C_1, C_2, C_3, C_4)$ и вычисляем значение функционала J на этих функциях. Аналитическое исследование на экстремум функции $J(x_2, y_2)$ (приравнивание нулю частных производных) дает условия трансверсальности (8.34), а численное может быть проведено различными методами.

Пусть теперь правая точка $M_2(x_2, y_2, z_2)$ находится на заданной линии

$$\begin{cases} y = \varphi(x); \\ z = \psi(x). \end{cases} \tag{8.35}$$

В этом случае в (8.32) независимым будет только δx_2, а δy_2 и δz_2 находятся дифференцированием уравнений (8.35):

$$\delta y_2 = \varphi'(x_2)\delta x_2; \qquad \delta z_2 = \psi'(x_2)\delta x_2. \tag{8.36}$$

Подставим эти выражения в (8.32). Приравнивая нулю коэффициент при независимой δx_2, получим условие трансверсальности для данного вида граничных условий на правом конце:

$$\left(F + (\varphi' - y')F_{y'} + (\psi' - z')F_{z'}\right)\Big|_{x=x_2} = 0. \qquad (8.37)$$

Для нахождения 4-х произвольных постоянных и 3-х координат точки M_2 мы в этом случае будем иметь такую систему семи нелинейных уравнений:

- два граничных условия на левом конце;
- два граничных условия на неизвестном правом конце;
- точка M_2 находится на линии (8.35) (два уравнения);
- условие трансверсальности (8.37).

Смысл условия трансверсальности (8.37) следующий: если правую точку двигать по линии (8.35), и по полученным двум точкам строить экстремаль, то из всех этих экстремалей самое экстремальное значение функционалу (8.25) доставит та пространственная кривая, для которой выполняются условия (8.37).

8.3. Вопросы для самопроверки

1. Выведите условие трансверсальности для элементарной задачи вариационного исчисления (2.1), когда граничное условие задано только на правом конце, а левый конец движется по заданной плоской линии.
2. Выведите условия трансверсальности для элементарной задачи вариационного исчисления (2.1), когда оба конца движутся по заданным линиям.
3. Выведите условие трансверсальности для функционала вида (4.1), если граничные условия (4.2) заданы только на левом конце интервала x_1, а правый конец движется по заданной линии $y = \varphi(x)$, причем угол α между экстремалью и кривой $y = \varphi(x)$ в точке сопряжения x_2 задан (вариант *a*) или произвольный (вариант *b*).
4. Выведите условия трансверсальности для функционала (5.1), если граница области D может скользить по заданной поверхности.
5. Выведите из уравнения (8.32) условия трансверсальности, если правая точка вообще не задана. Сколько должно быть таких условий? Какие вариации в (8.32) будут произвольными?
6. Выведите из (8.32) естественные граничные условия вида (7.4) как частный случай.

7. Разработайте алгоритм численного решения задачи на экстремум функционала (8.25), когда правая точка движется по линии (8.35).
8. В какой задаче максимальное значение функционала больше (или минимальное меньше): при закрепленном правом конце или при его движении по заданной линии (поверхности)?

8.4. Примеры выполнения заданий

8.4.1. Задание 1

Найти экстремаль функционала, рассмотренного ранее в *задании 1 главы 2*. Граничное условие на левом конце то же, что и в *задании 1 главы 2*, а правый конец движется по заданной линии:

$$J(y)=\int_{-1}^{x_2}\left(x^2+y^2+y'^2\right)dx;\quad \begin{cases} y(-1)=1; \\ y_2=4-2x_2^2. \end{cases} \tag{8.38}$$

Сравнить решение с решением *задания 1 главы 2*. Вычислить значение функционала на обеих кривых.

Программу для решения этого примера будем составлять на основе программы для *задания 1 главы 2*. Оставим полностью решение этого задания, но печатать будем только исходные данные и результаты.

```
clear all % очистили все
disp('Решаем задание 8.1') % выводим заголовок задачи
syms x y Dy D2y % описали символические переменные
F=x^2+y^2+Dy^2; % вводим подынтегральную функцию
x1=-1;
y1=1;
x2=1;
y2=2;
phi=4-2*x^2; % условие на правом конце
disp('Исходные данные:')
fprintf('F(x,y,y'')=%s\n',char(F))
fprintf('Граничное условие слева: y(%d)=%d\n',x1,y1)
fprintf('Граничное условие справа: y(%d)=%d\n',x2,y2)
fprintf('Ограничение на правом конце: y=%s\n',char(phi))
dFdy=diff(F,y); % вычислили Fy
dFdy1=diff(F,Dy); % вычислили Fy'
d_dFdy1_dx=diff(dFdy1,x); % Fy'x
d_dFdy1_dy=diff(dFdy1,y); % Fy'y
d_dFdy1_dy1=diff(dFdy1,Dy); % Fy'y'-условие Лежандра
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+d_dFdy1_dy1*D2y;
```

```
Euler=simple(dFdy-dFy1dx); % уравнение Эйлера
deqEuler=[char(Euler) '=0']; % добавили =0
Sol=dsolve(deqEuler,'x'); % решаем уравнение Эйлера
if length(Sol)~=1 % решений нет или более одного решения
  error('Нет решений или более одного решения!');
end
SolLeft=subs(Sol,x,x1); % подставили x1
SolRight=subs(Sol,x,x2); % подставили x2
EqLeft=[char(SolLeft) '=' char(sym(y1))]; % =y1
EqRight=[char(SolRight) '=' char(sym(y2))]; % =y2
Con=solve(EqLeft,EqRight,'C1,C2'); % решаем систему
C1=Con.C1; % присваиваем полученные решения
C2=Con.C2; % символическим константам C1 и C2
Sol21=vpa(eval(Sol),14);
xpl=linspace(x1,x2); % задаем массив абсцисс
y21=subs(Sol21,x,xpl); % вычислили ординаты
disp('Уравнение экстремали:')
fprintf('y(x)=%s\n',char(Sol21))
F21=subs(F,{y,Dy},{Sol21,diff(Sol21,x)});
J21=eval(int(F21,x,x1,x2)) % значение функционала
Решаем задание 8.1
Исходные данные:
F(x,y,y')=x^2+y^2+Dy^2
Граничное условие слева: y(-1)=1
Граничное условие справа: y(1)=2
Ограничение на правом конце: y=4-2*x^2
Уравнение экстремали:
y(x)=.42545906411967*sinh(x)+.97208141049585*cosh(x)
J21 =
   4.75035801121746
```

Приступим теперь к решению нашего задания. Для нахождения произвольных постоянных C_1, C_2 и правой точки x_2, y_2 нужно решить систему нелинейных уравнений (8.10). Мы будем ее решать численно, с помощью функции `fsolve`. Ей нужно будет передать в качестве параметра имя функции, в которой вычисляются левые части уравнений (8.10). Построим такую функцию.

Вначале запишем заголовок функции и переобозначим переменные. Сформируем первые 3 уравнения. После этого выведем условие трансверсальности. Если записывать 4-е уравнение в одну строку, то эта строка получится очень длинной. Поэтому вычислим предварительно функции $y(x_2)$, $y'(x_2)$, $\varphi'(x_2)$, $F(x_2)$, $F_{y'}(x_2)$, а затем сформируем строки с левой частью 4-го уравнения. Записываем все в файл (в рабочем каталоге) и для контроля выводим в область вывода.

```
s{1}='function y = MyFunc(x)'; % заголовок
s{2}='C1=x(1); C2=x(2); xr=x(3); yr=x(4); y=x;';
s{3}=['y(1)=' char(SolLeft) '-' char(sym(y1)) ';'];
s{4}=['y(2)=' char(subs(Sol,x,sym('xr'))) '-yr;'];
s{5}=['y(3)=' char(subs(phi,x,sym('xr'))) '-yr;'];
yxr=subs(Sol,x,sym('xr')); % x<=xr
dydx=diff(Sol,x);
dydxxr=subs(dydx,x,sym('xr'));
dphidx=diff(phi,x);
dphidxxr=subs(dphidx,x,sym('xr'));
Fxr=subs(F,{x,y,Dy},{sym('xr'),sym('yxr'),sym('dydxxr') });
dFdy1xr=subs(dFdy1,{x,y,Dy},{sym('xr'),sym('yxr'),sym('dydxxr')});
s{6}=['yxr=' char(yxr) ';'];
s{7}=['dydxxr=' char(dydxxr) ';'];
s{8}=['Fxr=' char(Fxr) ';'];
s{9}=['dphidxxr=' char(dphidxxr) ';'];
s{10}=['dFdy1xr=' char(dFdy1xr) ';'];
s{11}='y(4)=Fxr+(dphidxxr-dydxxr)*dFdy1xr;';
filename=fullfile(pwd,'MyFunc.m'); % файл
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:});
fid=fopen(filename,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записали файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\MyFunc.m:
function y = MyFunc(x)
C1=x(1); C2=x(2); xr=x(3); yr=x(4); y=x;
y(1)=-C1*sinh(1)+C2*cosh(1)-1;
y(2)=C1*sinh(xr)+C2*cosh(xr)-yr;
y(3)=4-2*xr^2-yr;
yxr=C1*sinh(xr)+C2*cosh(xr);
dydxxr=C1*cosh(xr)+C2*sinh(xr);
Fxr=xr^2+yxr^2+dydxxr^2;
dphidxxr=-4*xr;
dFdy1xr=2*dydxxr;
y(4)=Fxr+(dphidxxr-dydxxr)*dFdy1xr;
```

В качестве начального приближения задаем те значения констант, которые получились из решения *задания 1 главы 2*, точку $M_2(x_2, y_2)$ также берем из условия этого задания. Задаем параметры процесса решения: точность и максимально допустимое число итераций. Решаем систему нелинейных уравнений, анализируем критерий окончания процесса.

```
xinit=[eval(C1);eval(C2);x2;y2]; % начальное приближение
options=optimset('fsolve'); % опции по умолчанию
```

```
options=optimset(options,'Display','off',...
  'MaxIter',1000,'TolX',1e-8); % изменили опции
[xzero,fval,exitflag]=fsolve('MyFunc',xinit,options);
if exitflag>0, % анализируем критерий окончания
  disp('Решение найдено');
elseif exitflag<0,
  disp('Решение не найдено - нет сходимости');
else
  disp(['Проведено максимально допустимое ' ...
        'количество итераций - сходимость медленная']);
end
Решение найдено
```

Формируем аналитическое решение. Печатаем его. Вычисляем и печатаем значение функционала.

```
C1=vpa(sym(xzero(1)),14);
C2=vpa(sym(xzero(2)),14);
x2n=xzero(3);
y2n=xzero(4);
Sol81=vpa(eval(Sol),14);
disp('Уравнение экстремали:')
fprintf('y(x)=%s\n',char(Sol81))
fprintf('Правый конец x2=%12.6f\n',x2n)
F81=subs(F,{y,Dy},{Sol81,diff(Sol81,x)});
J81=eval(int(F81,x,x1,x2n))
Уравнение экстремали:
y(x)=-.28799990398088*sinh(x)+.42871522987622*cosh(x)
Правый конец x2=    1.348332
J91 =
   2.16340838859935
```

Заполняем таблицу для построения графика функций $y(x)$ и $\varphi(x)$. Мы будем рисовать функцию $\varphi(x)$ на небольшом участке вблизи новой и старой точек x_2. Строим на одном рисунке (рис. 8.5) три графика: решение *задания 1 главы 2*, функцию $\varphi(x)$ и решение нашего задания. В конце уберем за собой — сотрем записанный ранее файл.

```
xphi=linspace(min(x2,x2n)-0.05,max(x2,x2n)+0.05);
yphi=subs(phi,x,xphi); % phi(x)
x81=linspace(x1,x2n); % абсциссы
y81=subs(Sol81,x,x81); % ординаты
figure % фигура
plot(xpl,y21,'--b',xphi,yphi,':g',x81,y81,'-r') % рисуем
```

```
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 8.1') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm), \phi(\itx\rm)') % метка оси OY
delete(filename) % удалили файл
```

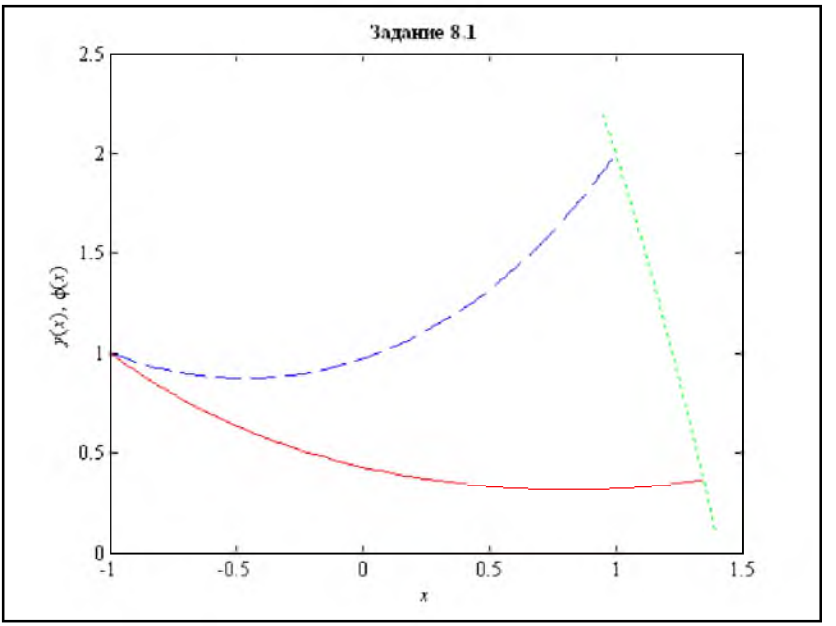


Рис. 8.5. Решение задания 8.1

В нашем варианте процесс решения системы нелинейных уравнений (8.10) успешно проработал и нашел решение. А как в вашем варианте? Если процесс плохо сходится, попробуйте вместо решения системы (8.10) воспользоваться алгоритмом, изложенным после вывода формулы (8.10). Или подберите другую функцию $y = \varphi(x)$, проходящую через точку $M_2(x_2, y_2)$.

8.4.2. Задание 2

Найти экстремаль функционала (3.14), рассмотренного ранее в *главе 3*, при тех же граничных условиях слева, но при условии, что правый конец находится на заданной поверхности.

$$J(y,z) = \int_{-2}^{x_2} \left(y'^2 + z'^2 + 2yz \right) dx; \quad \begin{cases} y(-2) = 1; \\ z(-2) = 0; \end{cases} \quad z_2 = 4 - \frac{x_2^2 + y_2^2}{2}. \qquad (8.39)$$

Сравнить решение с решением *задания главы 3*: вычислить значения функционалов для обеих задач.

Программу для решения этого примера будем составлять на основе программы для *задания главы 3* по принципам, изложенным в предыдущем задании. Оставим полностью решение *задания главы 3*, которое нам нужно для сравнения, но напечатаем только исходные данные и результаты. Мы находим решение, заполняем таблицу для построения графика, а график пока не строим: будем потом строить все вместе.

```
clear all
disp('Решаем задание 8.2') % выводим заголовок задачи
syms x y z Dy D2y Dz D2z % описали переменные
F=Dy^2+Dz^2+2*y*z; % вводим подынтегральную функцию
x1=-2; % вводим граничные условия
y1=1;
z1=0;
x2=2;
y2=0;
z2=2;
phi=4-(x^2+y^2)/2;
disp('Исходные данные:')
fprintf('F(x,y,y'',z,z'')=%s\n',char(F))
disp('Граничные условия слева:')
fprintf('y(%d)=%d;   z(%d)=%d\n',x1,y1,x1,z1)
disp('Граничные условия справа:')
fprintf('y(%d)=%d;   z(%d)=%d\n',x2,y2,x2,z2)
disp('Ограничивающая поверхность справа:')
fprintf('z(x,y)=%s\n',char(phi))
dFdy=diff(F,y);
dFdy1=diff(F,Dy);
d_dFdy1_dx=diff(dFdy1,x);
d_dFdy1_dy=diff(dFdy1,y);
d_dFdy1_dy1=diff(dFdy1,Dy);
d_dFdy1_dz=diff(dFdy1,z);
d_dFdy1_dz1=diff(dFdy1,Dz);
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+...
  d_dFdy1_dy1*D2y+d_dFdy1_dz*Dz+d_dFdy1_dz1*D2z;
dFdz=diff(F,z);
dFdz1=diff(F,Dz);
d_dFdz1_dx=diff(dFdz1,x);
d_dFdz1_dy=diff(dFdz1,y);
d_dFdz1_dy1=diff(dFdz1,Dy);
d_dFdz1_dz=diff(dFdz1,z);
```

```
d_dFdz1_dz1=diff(dFdz1,Dz);
dFz1dx=d_dFdz1_dx+d_dFdz1_dy*Dy+...
  d_dFdz1_dy1*D2y+d_dFdz1_dz*Dz+d_dFdz1_dz1*D2z;
EulerY=simple(dFdy-dFy1dx);
EulerZ=simple(dFdz-dFz1dx);
dEuY=[char(EulerY) '=0']; % уравнение Y
dEuZ=[char(EulerZ) '=0']; % уравнение Z
Sol=dsolve(dEuY,dEuZ,'x'); % решаем
if length(Sol)~=1 % решений нет или более одного
  error('Решений нет или более одного!');
else
  SolY=Sol.y;
  SolZ=Sol.z;
end
SolLY=subs(SolY,x,x1); % x1 в y
SolLZ=subs(SolZ,x,x1); % x1 в z
SolRY=subs(SolY,x,x2); % x2 в y
SolRZ=subs(SolZ,x,x2); % x2 в z
EqLY=[char(vpa(SolLY,14)) '=' char(sym(y1))];
EqLZ=[char(vpa(SolLZ,14)) '=' char(sym(z1))];
EqRY=[char(vpa(SolRY,14)) '=' char(sym(y2))];
EqRZ=[char(vpa(SolRZ,14)) '=' char(sym(z2))];
Con=solve(EqLY,EqLZ,EqRY,EqRZ,'C1,C2,C3,C4');
C1=Con.C1;
C2=Con.C2;
C3=Con.C3;
C4=Con.C4;
Sol3Y=vpa(eval(SolY),14);
Sol3Z=vpa(eval(SolZ),14);
disp('Уравнения экстремали:')
fprintf('y(x)=%s\nz(x)=%s\n',char(Sol3Y),char(Sol3Z))
F31=simple(subs(F,{y,z,Dy,Dz},...
  {Sol3Y,Sol3Z,diff(Sol3Y,x),diff(Sol3Z,x)}));
J31=eval(int(F31,x,x1,x2))
xpl=linspace(x1,x2); % массив абсцисс
y3=subs(Sol3Y,x,xpl); % вычислили ординаты
z3=subs(Sol3Z,x,xpl); % вычислили аппликаты
```

```
Решаем задание 8.2
Исходные данные:
F(x,y,y',z,z')=Dy^2+Dz^2+2*y*z
Граничные условия слева:
y(-2)=1;   z(-2)=0
Граничные условия справа:
y(2)=0;   z(2)=2
```

```
Ограничивающая поверхность справа:
z(x,y)=4-1/2*x^2-1/2*y^2
Уравнения экстремали:
y(x)=.65210765216302e-1*exp(-1.*x)+.13414090640925*exp(x)+
.60074949043049*cos(x)-.82481262772102*sin(x)
z(x)=.65210765216302e-1*exp(-1.*x)+.13414090640925*exp(x)-
.60074949043049*cos(x)+.82481262772102*sin(x)
J31 =
   1.94492120385672
```

Нам нужно решить систему семи нелинейных уравнений, перечисленных после вывода формулы (8.34), для нахождения 4-х произвольных постоянных и 3-х координат точки M_2. Сформируем строки, описывающие уравнения этой системы. Вначале запишем заголовок функции и переобозначим переменные. Граничные условия слева и справа дают первые 4 уравнения. Точка M_2 находится на поверхности $z=\varphi(x,y)$ — это 5-е уравнение. 6-е и 7-е уравнения дают условия трансверсальности (8.34). Сформируем их. Вначале вычислим вспомогательные функции $y(x_2)$, $z(x_2)$, $y'(x_2)$, $z'(x_2)$, $\varphi'_x(x_2)$, $\varphi'_y(x_2)$, $F(x_2)$, $F_{y'}(x_2)$, $F_{z'}(x_2)$. Переведем эти символические величины в строки и сформируем из них 6-е и 7-е уравнения. Запишем полученные строки в файл и в область вывода.

```
s{1}='function y = MyFunc(x)'; % заголовок
s{2}=['C1=x(1); C2=x(2); C3=x(3); C4=x(4);' ...
  'xr=x(5); yr=x(6); zr=x(7); y=x;'];
s{3}=['y(1)=' char(SolLY) '-' char(sym(y1)) ';'];
s{4}=['y(2)=' char(SolLZ) '-' char(sym(z1)) ';'];
s{5}=['y(3)=' char(subs(SolY,x,sym('xr'))) '-yr;'];
s{6}=['y(4)=' char(subs(SolZ,x,sym('xr'))) '-zr;'];
LeftEq5=subs(phi,{x,y},{sym('xr'),sym('yr')});
s{7}=['y(5)=' char(LeftEq5) '-zr;']; % уравнение 5
yxr=subs(SolY,x,sym('xr'));
zxr=subs(SolZ,x,sym('xr'));
dydx=diff(SolY,x);
dydxxr=subs(dydx,x,sym('xr'));
dzdx=diff(SolZ,x);
dzdxxr=subs(dzdx,x,sym('xr'));
dphidx=diff(phi,x);
dphidy=diff(phi,y);
dphidxxr=subs(dphidx,{x,y},{sym('xr'),sym('yxr')});
dphidyxr=subs(dphidy,{x,y},{sym('xr'),sym('yxr')});
Fxr=subs(F,{x,y,Dy,z,Dz},{sym('xr'),...
  sym('yxr'),sym('dydxxr'),sym('zxr'),sym('dzdxxr')});
```

```
dFdy1xr=subs(dFdy1,{x,y,Dy,z,Dz},{sym('xr'),...
  sym('yxr'),sym('dydxxr'),sym('zxr'),sym('dzdxxr')});
dFdz1xr=subs(dFdz1,{x,y,Dy,z,Dz},{sym('xr'),...
  sym('yxr'),sym('dydxxr'),sym('zxr'),sym('dzdxxr')});
s{8}=['yxr=' char(yxr) ';'];
s{9}=['zxr=' char(zxr) ';'];
s{10}=['dydxxr=' char(dydxxr) ';'];
s{11}=['dzdxxr=' char(dzdxxr) ';'];
s{12}=['Fxr=' char(Fxr) ';'];
s{13}=['dphidxxr=' char(dphidxxr) ';'];
s{14}=['dphidyxr=' char(dphidyxr) ';'];
s{15}=['dFdy1xr=' char(dFdy1xr) ';'];
s{16}=['dFdz1xr=' char(dFdz1xr) ';'];
s{17}=['y(6)=Fxr-dydxxr*dFdy1xr+'...
  '(dphidxxr-dzdxxr)*dFdz1xr;'];
s{18}='y(7)=dFdy1xr+dphidyxr*dFdz1xr;';
filename=fullfile(pwd,'MyFunc.m'); % имя файла
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:});
fid=fopen(filename,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записали в файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\MyFunc.m:
function y = MyFunc(x)
C1=x(1); C2=x(2); C3=x(3); C4=x(4);xr=x(5); yr=x(6); zr=x(7); y=x;
y(1)=1/4*C1*exp(2)+1/4*C1*exp(-2)+1/2*C1*cos(2)-1/4*C2*exp(2)+
1/4*C2*exp(-2)-1/2*C2*sin(2)+1/4*C3*exp(2)+1/4*C3*exp(-2)-
1/2*C3*cos(2)+1/2*C4*sin(2)+1/4*C4*exp(-2)-1/4*C4*exp(2)-1;
y(2)=1/4*C1*exp(2)+1/4*C1*exp(-2)-1/2*C1*cos(2)+1/2*C2*sin(2)+
1/4*C2*exp(-2)-1/4*C2*exp(2)+1/4*C3*exp(2)+1/4*C3*exp(-2)+
1/2*C3*cos(2)-1/4*C4*exp(2)+1/4*C4*exp(-2)-1/2*C4*sin(2)-0;
y(3)=1/4*C1*exp(-xr)+1/4*C1*exp(xr)+1/2*C1*cos(xr)-1/4*C2*exp(-xr)+
1/4*C2*exp(xr)+1/2*C2*sin(xr)+1/4*C3*exp(-xr)+1/4*C3*exp(xr)-
1/2*C3*cos(xr)-1/2*C4*sin(xr)+1/4*C4*exp(xr)-1/4*C4*exp(-xr)-yr;
y(4)=1/4*C1*exp(-xr)+1/4*C1*exp(xr)-1/2*C1*cos(xr)-1/2*C2*sin(xr)+
1/4*C2*exp(xr)-1/4*C2*exp(-xr)+1/4*C3*exp(-xr)+1/4*C3*exp(xr)+
1/2*C3*cos(xr)-1/4*C4*exp(-xr)+1/4*C4*exp(xr)+1/2*C4*sin(xr)-zr;
y(5)=4-1/2*xr^2-1/2*yr^2-zr;
yxr=1/4*C1*exp(-xr)+1/4*C1*exp(xr)+1/2*C1*cos(xr)-1/4*C2*exp(-xr)+
1/4*C2*exp(xr)+1/2*C2*sin(xr)+1/4*C3*exp(-xr)+1/4*C3*exp(xr)-
1/2*C3*cos(xr)-1/2*C4*sin(xr)+1/4*C4*exp(xr)-1/4*C4*exp(-xr);
zxr=1/4*C1*exp(-xr)+1/4*C1*exp(xr)-1/2*C1*cos(xr)-1/2*C2*sin(xr)+
1/4*C2*exp(xr)-1/4*C2*exp(-xr)+1/4*C3*exp(-xr)+1/4*C3*exp(xr)+
1/2*C3*cos(xr)-1/4*C4*exp(-xr)+1/4*C4*exp(xr)+1/2*C4*sin(xr);
```

```
dydxxr=-1/4*C1*exp(-xr)+1/4*C1*exp(xr)-1/2*C1*sin(xr)+1/4*C2*exp(-xr)+
1/4*C2*exp(xr)+1/2*C2*cos(xr)-1/4*C3*exp(-xr)+1/4*C3*exp(xr)+
1/2*C3*sin(xr)-1/2*C4*cos(xr)+1/4*C4*exp(xr)+1/4*C4*exp(-xr);
dzdxxr=-1/4*C1*exp(-xr)+1/4*C1*exp(xr)+1/2*C1*sin(xr)-1/2*C2*cos(xr)+
1/4*C2*exp(xr)+1/4*C2*exp(-xr)-1/4*C3*exp(-xr)+1/4*C3*exp(xr)-
1/2*C3*sin(xr)+1/4*C4*exp(-xr)+1/4*C4*exp(xr)+1/2*C4*cos(xr);
Fxr=dydxxr^2+dzdxxr^2+2*yxr*zxr;
dphidxxr=-xr;
dphidyxr=-yxr;
dFdy1xr=2*dydxxr;
dFdz1xr=2*dzdxxr;
y(6)=Fxr-dydxxr*dFdy1xr+(dphidxxr-dzdxxr)*dFdz1xr;
y(7)=dFdy1xr+dphidyxr*dFdz1xr;
```

Этот файл будет одним из параметров вызова функции `fsolve`, которая решает систему нелинейных уравнений. Другой параметр — начальное приближение. В качестве начальных значений констант и координат точки M_2 задаем данные из решения задания 3. Настраиваем параметры процесса решения. Решаем систему нелинейных уравнений. Печатаем критерий окончания.

```
xinit=[eval(C1);eval(C2);eval(C3);eval(C4);x2;y2;z2];
options=optimset('fsolve'); % опции по умолчанию
options=optimset(options,'Display','off',...
  'MaxIter',1000,'TolX',1e-8);
[xzero,fval,exitflag]=fsolve('MyFunc',xinit,options);
if exitflag>0,
  disp('Решение найдено');
elseif exitflag<0,
  disp('Решение не найдено - нет сходимости');
else
  disp(['Проведено максимально допустимое ' ...
        'количество итераций - сходимость медленная']);
end
```

```
Решение найдено
```

Подставляем найденные значения констант в решение — находим уравнения экстремали. Вычисляем значение функционала на экстремали.

```
C1=vpa(sym(xzero(1)),14);
C2=vpa(sym(xzero(2)),14);
C3=vpa(sym(xzero(3)),14);
C4=vpa(sym(xzero(4)),14);
x2n=xzero(5);
y2n=xzero(6);
z2n=xzero(7);
```

```
Sol82Y=vpa(eval(SolY),14);
Sol82Z=vpa(eval(SolZ),14);
disp('Уравнения экстремали:')
fprintf('y(x)=%s\nz(x)=%s\n',char(Sol82Y),char(Sol82Z))
fprintf('Правая точка: x2=%12.6f\n',x2n)
F82=simple(subs(F,{y,z,Dy,Dz},...
  {Sol82Y,Sol82Z,diff(Sol82Y,x),diff(Sol82Z,x)}));
J82=eval(int(F82,x,x1,x2n))
Уравнения экстремали:
y(x)=.67703062537165e-1*exp(-1.*x)-.19339166422550e-2*exp(x)+
.40918386066712*cos(x)-.73714117010392*sin(x)
z(x)=.67703062537165e-1*exp(-1.*x)-.19339166422550e-2*exp(x)-
.40918386066712*cos(x)+.73714117010392*sin(x)
Правая точка: x2=    2.401870
J82 =
  -0.60687741704535
```

Составляем таблицу для графика поверхности $z=\varphi(x,y)$. Чтобы не затенять чертеж, будем рисовать эту поверхность только вблизи точек M_2 (старой и новой). Затем заполняем таблицу значений найденного решения и рисуем график нашего задания и, для сравнения, *задания главы 3* (рис. 8.6).

```
[xi,yi]=meshgrid(linspace(min(x2,x2n)-0.1,max(x2,x2n)+0.1,10),...
  linspace(min(y2,y2n)-0.1,max(y2,y2n)+0.1,10));
z=subs(phi,{x,y},{xi,yi}); % вычислили phi(x,y) на сетке
x82=linspace(x1,x2n); % массив абсцисс
y82=subs(Sol82Y,x,x82);
z82=subs(Sol82Z,x,x82);
figure % фигура
plot3(xpl,y3,z3,'--b',xi,yi,z,':g',x82,y82,z82,'-r')
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 8.2') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
zlabel('\itz\rm(\itx\rm)') % метка оси OZ
view(205,30) % точка просмотра
grid on % показали сетку
box on % показали внешний контур
delete(filename) % удалили файл
```

Сошелся ли у вас процесс решения системы нелинейных уравнений? Если нет, то реализуйте алгоритм, описанный после вывода формулы (8.34), или подберите другую поверхность $z=\varphi(x,y)$. Поверхность должна проходить

через точку $M_2(x_2,y_2)$, которая задана в вашем варианте *задания главы 3* — тогда можно будет сравнивать решения (значения функционалов).

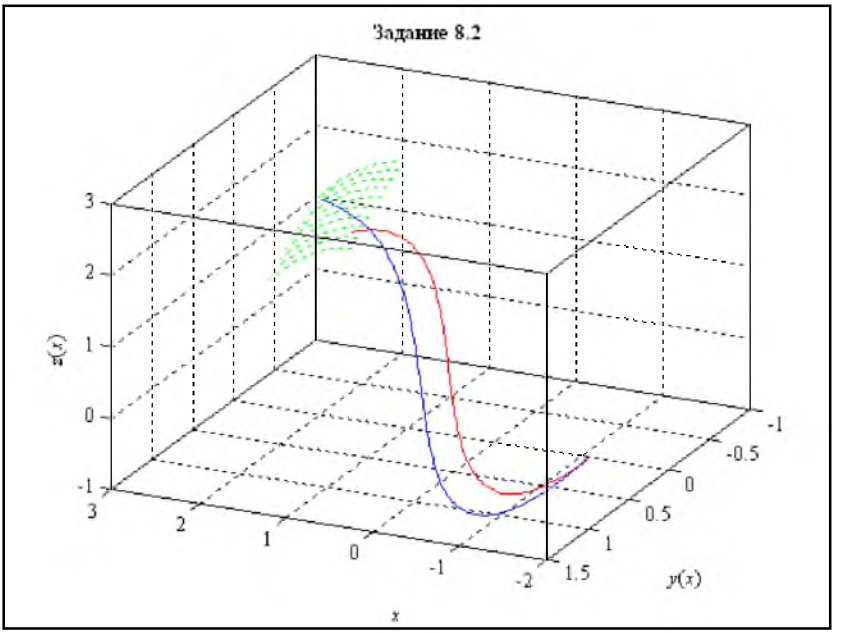


Рис. 8.6. Решение задания 8.2

8.4.3. Задание 3

Найти экстремаль функционала, рассмотренного ранее в *задании главы 3* и предыдущем примере, при условии, что правый конец экстремали находится на заданной линии.

$$J(y,z)=\int_{-2}^{x_2}\left(y'^2+z'^2+2yz\right)dx;\quad \begin{cases} y(-2)=1; \\ z(-2)=0; \end{cases} \quad \begin{cases} y_2=4-x_2^2; \\ z_2=6e^{x_2-2}-4. \end{cases} \tag{8.40}$$

Сравнить результат с решением *задания главы 3*. Посчитать значения функционалов на экстремалях для обоих случаев.

Напишем программу для этого задания на основе программы предыдущего задания. Изменятся исходные данные. Решение *задания главы 3* оставляем без изменения.

```
clear all
disp('Решаем задание 8.3') % выводим заголовок задачи
```

```
syms x y z Dy D2y Dz D2z % описали переменные
F=Dy^2+Dz^2+2*y*z; % вводим подынтегральную функцию
x1=-2; % вводим граничные условия
y1=1;
z1=0;
x2=2;
y2=0;
z2=2;
phi=4-x^2;
psi=6*exp(x-2)-4;
disp('Исходные данные:')
fprintf('F(x,y,y'',z,z'')=%s\n',char(F))
disp('Граничные условия слева:')
fprintf('y(%d)=%d;   z(%d)=%d\n',x1,y1,x1,z1)
disp('Граничные условия справа:')
fprintf('y(%d)=%d;   z(%d)=%d\n',x2,y2,x2,z2)
disp('Ограничивающая линия справа:')
fprintf('y(x)=%s\nz(x)=%s\n',char(phi),char(psi))
dFdy=diff(F,y);
dFdy1=diff(F,Dy);
d_dFdy1_dx=diff(dFdy1,x);
d_dFdy1_dy=diff(dFdy1,y);
d_dFdy1_dy1=diff(dFdy1,Dy);
d_dFdy1_dz=diff(dFdy1,z);
d_dFdy1_dz1=diff(dFdy1,Dz);
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+...
  d_dFdy1_dy1*D2y+d_dFdy1_dz*Dz+d_dFdy1_dz1*D2z;
dFdz=diff(F,z);
dFdz1=diff(F,Dz);
d_dFdz1_dx=diff(dFdz1,x);
d_dFdz1_dy=diff(dFdz1,y);
d_dFdz1_dy1=diff(dFdz1,Dy);
d_dFdz1_dz=diff(dFdz1,z);
d_dFdz1_dz1=diff(dFdz1,Dz);
dFz1dx=d_dFdz1_dx+d_dFdz1_dy*Dy+...
  d_dFdz1_dy1*D2y+d_dFdz1_dz*Dz+d_dFdz1_dz1*D2z;
EulerY=simple(dFdy-dFy1dx);
EulerZ=simple(dFdz-dFz1dx);
dEuY=[char(EulerY) '=0']; % уравнение Y
dEuZ=[char(EulerZ) '=0']; % уравнение Z
Sol=dsolve(dEuY,dEuZ,'x'); % решаем
if length(Sol)~=1 % решений нет или более одного
  error('Решений нет или более одного!');
```

```
else
  SolY=Sol.y;
  SolZ=Sol.z;
end
SolLY=subs(SolY,x,x1); % x1 в y
SolLZ=subs(SolZ,x,x1); % x1 в z
SolRY=subs(SolY,x,x2); % x2 в y
SolRZ=subs(SolZ,x,x2); % x2 в z
EqLY=[char(vpa(SolLY,14)) '=' char(sym(y1))];
EqLZ=[char(vpa(SolLZ,14)) '=' char(sym(z1))];
EqRY=[char(vpa(SolRY,14)) '=' char(sym(y2))];
EqRZ=[char(vpa(SolRZ,14)) '=' char(sym(z2))];
Con=solve(EqLY,EqLZ,EqRY,EqRZ,'C1,C2,C3,C4');
C1=Con.C1;
C2=Con.C2;
C3=Con.C3;
C4=Con.C4;
Sol3Y=vpa(eval(SolY),14);
Sol3Z=vpa(eval(SolZ),14);
disp('Уравнения экстремали:')
fprintf('y(x)=%s\nz(x)=%s\n',char(Sol3Y),char(Sol3Z))
F31=simple(subs(F,{y,z,Dy,Dz},...
  {Sol3Y,Sol3Z,diff(Sol3Y,x),diff(Sol3Z,x)}));
J31=eval(int(F31,x,x1,x2))
xpl=linspace(x1,x2); % массив абсцисс
y3=subs(Sol3Y,x,xpl); % вычислили ординаты
z3=subs(Sol3Z,x,xpl); % вычислили аппликаты
```

```
Решаем задание 8.3
Исходные данные:
F(x,y,y',z,z')=Dy^2+Dz^2+2*y*z
Граничные условия слева:
y(-2)=1;   z(-2)=0
Граничные условия справа:
y(2)=0;   z(2)=2
Ограничивающая линия справа:
y(x)=4-x^2
z(x)=6*exp(x-2)-4
Уравнения экстремали:
y(x)=.65210765216302e-1*exp(-1.*x)+.13414090640925*exp(x)+
.60074949043049*cos(x)-.82481262772102*sin(x)
z(x)=.65210765216302e-1*exp(-1.*x)+.13414090640925*exp(x)-
.60074949043049*cos(x)+.82481262772102*sin(x)
J31 =
   1.94492120385672
```

Заполняем файл, который вычисляет левые части системы нелинейных уравнений, описанных после вывода формулы (8.37). Вначале записываем заголовок и переобозначаем переменные. Первые 4 уравнения — это граничные условия на концах. Следующие 2 уравнения: точка M_2 находится на линии (8.35). 7-е уравнение — это условие трансверсальности (8.37). Формируем его. Записываем функцию в файл и область вывода.

```
s{1}='function y = MyFunc(x)'; % заголовок
s{2}=['C1=x(1); C2=x(2); C3=x(3); C4=x(4);' ...
  'xr=x(5); yr=x(6); zr=x(7); y=x;'];
s{3}=['y(1)=' char(SolLY) '-' char(sym(y1)) ';'];
s{4}=['y(2)=' char(SolLZ) '-' char(sym(z1)) ';'];
s{5}=['y(3)=' char(subs(SolY,x,sym('xr'))) '-yr;'];
s{6}=['y(4)=' char(subs(SolZ,x,sym('xr'))) '-zr;'];
s{7}=['y(5)=' char(subs(phi,x,sym('xr'))) '-yr;'];
s{8}=['y(6)=' char(subs(psi,x,sym('xr'))) '-zr;'];
yxr=subs(SolY,x,sym('xr'));
zxr=subs(SolZ,x,sym('xr'));
dydx=diff(SolY,x);
dydxxr=subs(dydx,x,sym('xr'));
dzdx=diff(SolZ,x);
dzdxxr=subs(dzdx,x,sym('xr'));
dphidx=diff(phi,x);
dphidxxr=subs(dphidx,x,sym('xr'));
dpsidx=diff(psi,x);
dpsidxxr=subs(dpsidx,x,sym('xr'));
Fxr=subs(F,{x,y,Dy,z,Dz},{sym('xr'),...
  sym('yxr'),sym('dydxxr'),sym('zxr'),sym('dzdxxr')});
dFdy1xr=subs(dFdy1,{x,y,Dy,z,Dz},{sym('xr'),...
  sym('yxr'),sym('dydxxr'),sym('zxr'),sym('dzdxxr')});
dFdz1xr=subs(dFdz1,{x,y,Dy,z,Dz},{sym('xr'),...
  sym('yxr'),sym('dydxxr'),sym('zxr'),sym('dzdxxr')});
s{9}=['yxr=' char(yxr) ';'];
s{10}=['zxr=' char(zxr) ';'];
s{11}=['dydxxr=' char(dydxxr) ';'];
s{12}=['dzdxxr=' char(dzdxxr) ';'];
s{13}=['Fxr=' char(Fxr) ';'];
s{14}=['dphidxxr=' char(dphidxxr) ';'];
s{15}=['dpsidxxr=' char(dpsidxxr) ';'];
s{16}=['dFdy1xr=' char(dFdy1xr) ';'];
s{17}=['dFdz1xr=' char(dFdz1xr) ';'];
s{18}=['y(7)=Fxr+(dphidxxr-dydxxr)*dFdy1xr+' ...
  '(dpsidxxr-dzdxxr)*dFdz1xr;'];
```

```
filename=fullfile(pwd,'MyFunc.m'); % имя файла
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:});
fid=fopen(filename,'w'); % открываем файл
fprintf(fid,'%s\n',s{:}); % записываем в файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\MyFunc.m:
function y = MyFunc(x)
C1=x(1); C2=x(2); C3=x(3); C4=x(4);
xr=x(5); yr=x(6); zr=x(7); y=x;
y(1)=1/4*C1*exp(2)+1/4*C1*exp(-2)+1/2*C1*cos(2)-1/4*C2*exp(2)+
1/4*C2*exp(-2)-1/2*C2*sin(2)+1/4*C3*exp(2)+1/4*C3*exp(-2)-
1/2*C3*cos(2)+1/2*C4*sin(2)+1/4*C4*exp(-2)-1/4*C4*exp(2)-1;
y(2)=1/4*C1*exp(2)+1/4*C1*exp(-2)-1/2*C1*cos(2)+1/2*C2*sin(2)+
1/4*C2*exp(-2)-1/4*C2*exp(2)+1/4*C3*exp(2)+1/4*C3*exp(-2)+
1/2*C3*cos(2)-1/4*C4*exp(2)+1/4*C4*exp(-2)-1/2*C4*sin(2)-0;
y(3)=1/4*C1*exp(-xr)+1/4*C1*exp(xr)+1/2*C1*cos(xr)-1/4*C2*exp(-xr)+
1/4*C2*exp(xr)+1/2*C2*sin(xr)+1/4*C3*exp(-xr)+1/4*C3*exp(xr)-
1/2*C3*cos(xr)-1/2*C4*sin(xr)+1/4*C4*exp(xr)-1/4*C4*exp(-xr)-yr;
y(4)=1/4*C1*exp(-xr)+1/4*C1*exp(xr)-1/2*C1*cos(xr)-1/2*C2*sin(xr)+
1/4*C2*exp(xr)-1/4*C2*exp(-xr)+1/4*C3*exp(-xr)+1/4*C3*exp(xr)+
1/2*C3*cos(xr)-1/4*C4*exp(-xr)+1/4*C4*exp(xr)+1/2*C4*sin(xr)-zr;
y(5)=4-xr^2-yr;
y(6)=6*exp(xr-2)-4-zr;
yxr=1/4*C1*exp(-xr)+1/4*C1*exp(xr)+1/2*C1*cos(xr)-1/4*C2*exp(-xr)+
1/4*C2*exp(xr)+1/2*C2*sin(xr)+1/4*C3*exp(-xr)+1/4*C3*exp(xr)-
1/2*C3*cos(xr)-1/2*C4*sin(xr)+1/4*C4*exp(xr)-1/4*C4*exp(-xr);
zxr=1/4*C1*exp(-xr)+1/4*C1*exp(xr)-1/2*C1*cos(xr)-1/2*C2*sin(xr)+
1/4*C2*exp(xr)-1/4*C2*exp(-xr)+1/4*C3*exp(-xr)+1/4*C3*exp(xr)+
1/2*C3*cos(xr)-1/4*C4*exp(-xr)+1/4*C4*exp(xr)+1/2*C4*sin(xr);
dydxxr=-1/4*C1*exp(-xr)+1/4*C1*exp(xr)-1/2*C1*sin(xr)+1/4*C2*exp(-xr)+
1/4*C2*exp(xr)+1/2*C2*cos(xr)-1/4*C3*exp(-xr)+1/4*C3*exp(xr)+
1/2*C3*sin(xr)-1/2*C4*cos(xr)+1/4*C4*exp(xr)+1/4*C4*exp(-xr);
dzdxxr=-1/4*C1*exp(-xr)+1/4*C1*exp(xr)+1/2*C1*sin(xr)-1/2*C2*cos(xr)+
1/4*C2*exp(xr)+1/4*C2*exp(-xr)-1/4*C3*exp(-xr)+1/4*C3*exp(xr)-
1/2*C3*sin(xr)+1/4*C4*exp(-xr)+1/4*C4*exp(xr)+1/2*C4*cos(xr);
Fxr=dydxxr^2+dzdxxr^2+2*yxr*zxr;
dphidxxr=-2*xr;
dpsidxxr=6*exp(xr-2);
dFdy1xr=2*dydxxr;
dFdz1xr=2*dzdxxr;
y(7)=Fxr+(dphidxxr-dydxxr)*dFdy1xr+(dpsidxxr-dzdxxr)*dFdz1xr;
```

Задаем начальное приближение, параметры процедуры решения и решаем полученную систему. Найденные значения констант подставляем в общее решение. Вычисляем значение функционала на полученной экстремали.

```
xinit=[eval(C1);eval(C2);eval(C3);eval(C4);x2;y2;z2];
options=optimset('fsolve'); % опции по умолчанию
options=optimset(options,'Display','off',...
  'MaxIter',1000,'TolX',1e-8); % изменили опции
[xzero,fval,exitflag]=fsolve('MyFunc',xinit,options);
if exitflag>0,
  disp('Решение найдено');
elseif exitflag<0,
  disp('Решение не найдено - нет сходимости');
else
  disp(['Проведено максимально допустимое ' ...
        'количество итераций - сходимость медленная']);
end
C1=vpa(sym(xzero(1)),14);
C2=vpa(sym(xzero(2)),14);
C3=vpa(sym(xzero(3)),14);
C4=vpa(sym(xzero(4)),14);
x2n=xzero(5);
y2n=xzero(6);
z2n=xzero(7);
Sol83Y=vpa(eval(SolY),14);
Sol83Z=vpa(eval(SolZ),14);
disp('Уравнения экстремали:')
fprintf('y(x)=%s\nz(x)=%s\n',char(Sol83Y),char(Sol83Z))
fprintf('Правая точка: x2=%12.6f\n',x2n)
F83=simple(subs(F,{y,z,Dy,Dz},...
  {Sol83Y,Sol83Z,diff(Sol83Y,x),diff(Sol83Z,x)}));
J83=eval(int(F83,x,x1,x2))
Решение найдено
Уравнения экстремали:
y(x)=.65204984200015e-1*exp(-1.*x)+.13445653920418*exp(x)+
.70517723513910e-1*cos(x)-.58214805402975*sin(x)
z(x)=.65204984200015e-1*exp(-1.*x)+.13445653920418*exp(x)-
.70517723513910e-1*cos(x)+.58214805402975*sin(x)
Правая точка: x2=    1.911380
J83 =
   1.93213435309514
```

Вычисляем таблицу значений функции (8.35) и решения нашей задачи, рисуем графики нашего решения и решения *задания главы 3* (рис. 8.7). Ограничивающую линию рисуем на небольшом участке. Удаляем файл.

```
xb=linspace(min(x2,x2n)-0.1,max(x2,x2n)+0.1);
yi=subs(phi,x,xb); % phi(x)
zi=subs(psi,x,xb); % psi(x)
x83=linspace(x1,x2n); % массив абсцисс
y83=subs(Sol83Y,x,x83);
z83=subs(Sol83Z,x,x83);
figure % фигура
plot3(xpl,y3,z3,'--b',xb,yi,zi,':g',x83,y83,z83,'-r')
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 8.3') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
zlabel('\itz\rm(\itx\rm)') % метка оси OZ
view(205,30) % точка просмотра
grid on % показали сетку
box on % показали внешний контур
delete(filename) % удалили файл
```

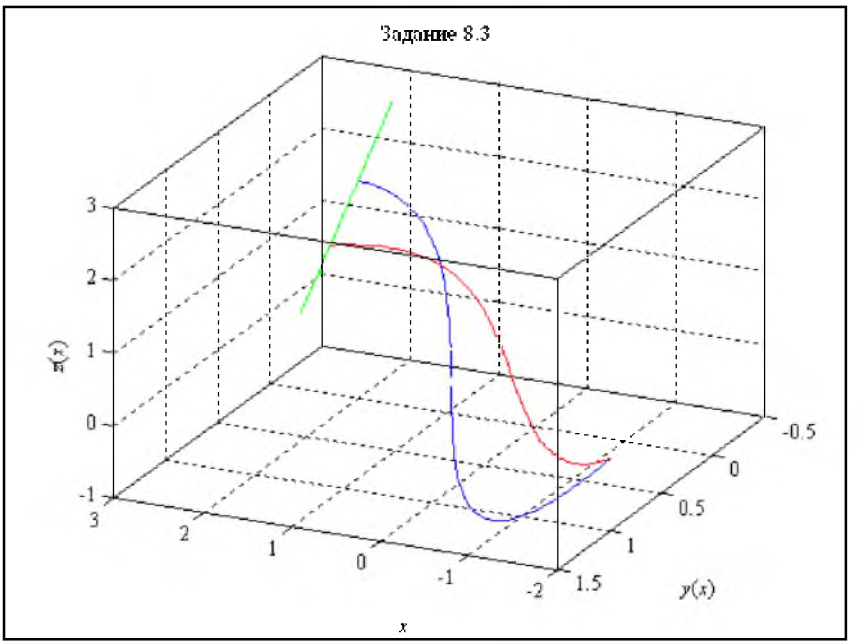


Рис. 8.7. Решение задания 8.3

Сошелся ли у вас процесс решения системы нелинейных уравнений? Если нет, то реализуйте алгоритм, разработанный вами (см. вопрос 7 для самопро-

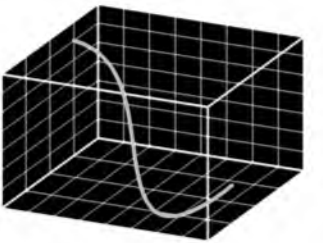
верки). Или попробуйте подобрать другую линию. Она должна проходить через точку $M_2(x_2, y_2)$, которая задана в вашем варианте *задания главы 3* — тогда можно будет сравнивать решения (значения функционалов).

8.5. Задание

1. Для своего варианта *задания 1 главы 2* найти экстремаль при условии, что правый конец движется по заданной линии. Сравнить решение с решением *задания 1 главы 2*. Если процесс численного решения системы нелинейных уравнений расходится, применить алгоритм, описанный после вывода системы (8.10), или подобрать другую линию.
2. Для своего варианта *задания главы 3* найти экстремаль при условии, что правый конец движется по заданной поверхности. Сравнить решение с решением *задания главы 3*. Если процесс численного решения системы нелинейных уравнений расходится, применить алгоритм, описанный после вывода уравнений (8.34), или подобрать другую поверхность.
3. Для своего варианта *задания главы 3* найти экстремаль при условии, что правый конец движется по заданной линии. Сравнить решение с решением *задания главы 3*. Если процесс численного решения системы нелинейных уравнений расходится, применить алгоритм, разработанный вами при ответе на 7-й вопрос (см. вопросы для самопроверки) или подобрать другую линию.

Уравнения линий и поверхностей для каждого варианта есть на диске.

ГЛАВА 9



Отражение экстремалей

9.1. Отражение экстремалей в элементарной задаче вариационного исчисления

Рассмотрим элементарную задачу вариационного исчисления для функционала (2.1), зависящего от функции одной переменной и ее производной. Пусть для этого функционала заданы граничные условия (2.2) и дополнительное требование: экстремаль $y(x)$ касается заданной кривой $y=\varphi(x)$ в неизвестной точке $M_0(x_0, y_0)$, где $x_0 \in [x_1, x_2]$. В точке $M_0(x_0, y_0)$ экстремаль не обязательно должна быть гладкой. На рис. 9.1 показаны экстремаль (жирная линия), допустимые функции (тонкие сплошные линии) и линия отражения (штриховая).

Выведем необходимые условия экстремума для этой вариационной задачи. При этом мы применим те рассуждения, которые уже несколько раз использовали. А именно: если функционал достигает экстремума на более широком классе функций, то он тем более будет достигать экстремума и на подклассе этого класса. Так, у нас функционал $J(y)$ достигает экстремума на интервале $[x_1, x_2]$, т. е. на классе функций, варьируемых на всем этом интервале (разумеется, при выполнении граничных условий (2.2) и условия касания кривой $y=\varphi(x)$ в точке x_0). Значит, он будет достигать экстремума и тогда, когда, например, на интервале $[x_1, x_0]$ функция будет варьироваться, а на интервале $[x_0, x_2]$ — нет (и наоборот). Иными словами, наш функционал достигает экстремума на каждом из интервалов $[x_1, x_0]$ и $[x_0, x_2]$ в отдельности. Следовательно, на каждом из этих интервалов экстремаль должна удовлетворять дифференциальному уравнению Эйлера (2.9). Обозначим решения этого уравнения на левом интервале $y^l(x)$, а на правом — $y^r(x)$. Каждое из них содержат по 2 произвольных постоянных: C_1^l, C_2^l и C_1^r, C_2^r соответственно. Кроме

того, неизвестны также 2 координаты точки касания x_0 и y_0. Для определения этих шести неизвестных мы имеем уравнения:

- граничное условие на левом конце $y^l(x_1)=y_1$;
- граничное условие на правом конце $y^r(x_2)=y_2$;
- $y^l(x)$ проходит через точку M_0: $y^l(x_0)=y_0$;
- $y^r(x)$ также проходит через точку M_0: $y^r(x_0)=y_0$;
- точка M_0 лежит на линии $y=\varphi(x)$: $y_0=\varphi(x_0)$.

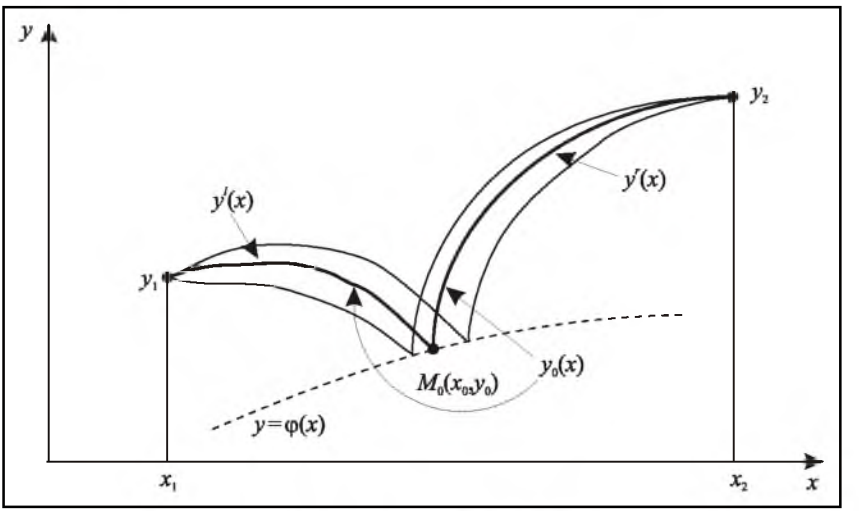


Рис. 9.1. Отражение экстремалей

Недостает еще одного уравнения. Смысл этого уравнения: где именно на кривой $y=\varphi(x)$ нужно взять точку x_0, чтобы построенные на участках $[x_1, x_0]$ и $[x_0, x_2]$ экстремали доставляли нашему функционалу экстремальное значение? Как всегда, мы получим недостающее уравнение из необходимого условия экстремума функционала: $\delta J=0$. Вариация функционала на каждом из двух участков δJ_1 и δJ_2 вызывается вариацией функции $y(x)$ (вместе с ее производной $y'(x)$) и вариацией точки δx_0. Для δJ_1 точка x_0 — правая. В *главе 8* было выведено условие трансверсальности (8.9) для функционала с подвижной правой границей. В нашей задаче это δJ_1:

$$\delta J_1 = \left(F + (\varphi' - y')F_{y'}\right)\Big|_{x=x_0-0} \delta x_0 . \qquad (9.1)$$

Мы здесь взяли левосторонний предел, т. к. предел справа будет другим: там будет другая функция $y(x)$. Аналогично вы вывели в *главе 8* условие транс-

версальности для задачи с подвижным левым концом (см. первый вопрос для самопроверки в *главе 8*):

$$\delta J_2 = -\left(F + (\varphi' - y')F_{y'}\right)\Big|_{x=x_0+0} \delta x_0. \tag{9.2}$$

Приравнивая сумму этих вариаций нулю и учитывая, что δx_0 — произвольная, получим недостающее 6-е уравнение:

$$\left(F + (\varphi' - y')F_{y'}\right)\Big|_{x=x_0-0} = \left(F + (\varphi' - y')F_{y'}\right)\Big|_{x=x_0+0}. \tag{9.3}$$

Уравнение (9.3) является условием трансверсальности для задачи отражения. Его смысл следующий: если точку x_0 двигать по линии $y=\varphi(x)$ и находить пару экстремалей (решений уравнения Эйлера) $y^l(x)$ и $y^r(x)$, удовлетворяющих граничным условиям, то доставлять экстремум функционалу будет та пара функций, которая удовлетворяет условию трансверсальности (9.3). При таком подходе функционал становится функцией одной переменной x_0. Необходимое условие экстремума, записанное в аналитическом виде, и дает условие трансверсальности (9.3). Но, конечно же, находить минимум функции одной переменной можно и численно.

ПРИМЕР 9.1. Отражение света. Пусть свет распространяется в среде с переменной скоростью $v(x,y)$. Он идет из точки $M_1(x_1,y_1)$ в точку $M_2(x_2,y_2)$ с отражением от линии $y=\varphi(x)$ так, как это показано на рис. 9.1. Требуется исследовать поведение луча в точке отражения $M_0(x_0,y_0)$.

Согласно принципу Ферма свет распространяется по такой траектории $y(x)$, по которой время его движения минимально. Следовательно, задача сводится к минимизации функционала $T(y(x))$ — времени движения по траектории $y(x)$, при граничных условиях $M_1(x_1,y_1)$, $M_2(x_2,y_2)$ и дополнительном условии: отражении экстремали в точке $M_0(x_0,y_0)$. Выведем этот функционал. Скорость $v(x,y)$ — это производная от пути по времени, поэтому имеем:

$$v(x,y) = \frac{dl}{dt} = \frac{\sqrt{1+y'^2}\,dx}{dt}; \quad dt = \frac{\sqrt{1+y'^2}}{v(x,y)}dx; \quad T(y) = \int_{x_1}^{x_2} \frac{\sqrt{1+y'^2}}{v(x,y)}dx. \tag{9.4}$$

Составим условие трансверсальности (9.3). Вначале упростим выражение $F + (\varphi' - y')F_{y'}$:

$$\begin{aligned} F + (\varphi' - y')F_{y'} &= \frac{\sqrt{1+y'^2}}{v(x,y)} + (\varphi' - y')\frac{y'}{v(x,y)\sqrt{1+y'^2}} = \\ \frac{1+y'^2 + (\varphi' - y')y'}{v(x,y)\sqrt{1+y'^2}} &= \frac{1+\varphi' y'}{v(x,y)\sqrt{1+y'^2}}. \end{aligned} \tag{9.5}$$

Поскольку в точке отражения $M_0(x_0, y_0)$ скорость света одинакова и при $x \to x_0 - 0$, и при $x \to x_0 + 0$, то условие трансверсальности (9.3) будет иметь вид

$$\left.\frac{1+\varphi' y'}{\sqrt{1+y'^2}}\right|_{x_0-0} = \left.\frac{1+\varphi' y'}{\sqrt{1+y'^2}}\right|_{x_0+0}. \tag{9.6}$$

Введем в рассмотрение углы:

- α — угол наклона касательной кривой $y=\varphi(x)$ к оси Ox в точке x_0; $\operatorname{tg}\alpha = \varphi'(x_0)$;
- β_- — угол наклона экстремали к оси Ox в точке x_0 слева; $\operatorname{tg}\beta_- = y'(x_0 - 0)$;
- β_+ — угол наклона экстремали к оси Ox в точке x_0 справа; $\operatorname{tg}\beta_+ = y'(x_0 + 0)$;
- i_- — угол падения;
- i_+ — угол отражения.

Эти углы показаны на рис. 9.2. Соотношения между ними:

$$i_- = \beta_- - \alpha - \frac{\pi}{2}; \quad i_+ = \frac{\pi}{2} - \beta_+ + \alpha\,. \tag{9.7}$$

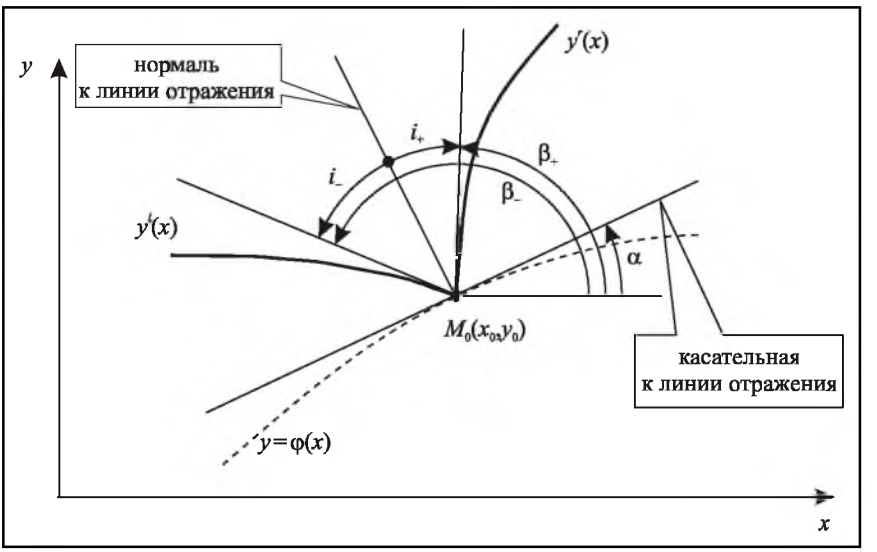


Рис. 9.2. К задаче об отражении света

В этих обозначениях условие трансверсальности, которое у нас уже имеет вид (9.6), записывается так:

$$\frac{1+\operatorname{tg}\alpha\operatorname{tg}\beta_-}{\sqrt{1+\operatorname{tg}^2\beta_-}}=\frac{1+\operatorname{tg}\alpha\operatorname{tg}\beta_+}{\sqrt{1+\operatorname{tg}^2\beta_+}}. \tag{9.8}$$

Далее — обычные тригонометрические преобразования. Учитывая, что угол β_- — тупой, имеем:

$$\begin{gathered}
-\cos\beta_-\left(1+\operatorname{tg}\alpha\operatorname{tg}\beta_-\right)=\cos\beta_+\left(1+\operatorname{tg}\alpha\operatorname{tg}\beta_+\right);\\
-\cos\beta_- -\frac{\sin\alpha}{\cos\alpha}\sin\beta_-=\cos\beta_+ +\frac{\sin\alpha}{\cos\alpha}\sin\beta_+;\\
-\cos\alpha\cos\beta_- -\sin\alpha\sin\beta_-=\cos\alpha\cos\beta_+ +\sin\alpha\sin\beta_+;\\
-\cos\left(\beta_- -\alpha\right)=\cos\left(\beta_+ -\alpha\right);\\
-\cos\left(i_- +\frac{\pi}{2}\right)=\cos\left(\frac{\pi}{2}-i_+\right);\\
\sin i_-=\sin i_+;
\end{gathered} \tag{9.9}$$

и окончательно — угол падения равен углу отражения:

$$i_-=i_+. \tag{9.10}$$

Вывод: если при любой скорости распространения света $v(x, y)$ свет приходит из точки $M_1(x_1, y_1)$ в точку $M_2(x_2, y_2)$ путем отражения в точке $M_0(x_0, y_0)$, то в этой точке угол падения равен углу отражения (см. также задачу о преломлении света в *главе 10*). □

9.2. Вопросы для самопроверки

1. Пусть на нашей экстремали достигается минимум. На какой функции значение функционала будет меньше: на кривой без отражения или с отражением?
2. Выведите условия трансверсальности для задачи отражения функционала вида (3.1), зависящего от двух функций, при отражении от заданной поверхности $z=\varphi(x,y)$.
3. Выведите условия трансверсальности для задачи отражения функционала вида (3.1), зависящего от двух функций, при отражении от заданной линии $y=\varphi(x)$, $z=\psi(x)$.

9.3. Пример выполнения задания

Найти экстремаль функционала, рассмотренного ранее в *задании 1 главы 2*, при тех же граничных условиях и при дополнительном требовании: экстре-

маль касается заданной линии в точке $M_0(x_0, y_0)$. Сравнить решение с решением *задания 1 главы 2*: вычислить значения функционалов на обеих экстремалях:

$$J(y) = \int_{-1}^{1} \left(x^2 + y^2 + y'^2\right) dx; \quad \begin{cases} y(-1) = 1; \\ y(1) = 2; \end{cases} \quad y_0 = 0{,}8 - 0{,}2x_0^2. \qquad (9.11)$$

Для составления программы воспользуемся программой для решения задания 8.1, а также другими программами, составленными ранее. Как и в предыдущих задачах, вначале найдем решение *задания 1 главы 2*.

```
clear all % очистили все
disp('Решаем задание 9') % выводим заголовок задачи
syms x y Dy D2y % описали символические переменные
F=x^2+y^2+Dy^2; % подынтегральная функция
x1=-1; % вводим граничные условия
y1=1;
x2=1;
y2=2;
phi=0.8-0.2*x^2; % линия отражения
disp('Исходные данные:')
fprintf('F(x,y,y'')=%s\n',char(F))
fprintf('Граничное условие слева: y(%d)=%d\n',x1,y1)
fprintf('Граничное условие справа: y(%d)=%d\n',x2,y2)
fprintf('Линия отражения: y=%s\n',char(phi))
dFdy=diff(F,y); % вычислили Fy
dFdy1=diff(F,Dy); % вычислили Fy'
d_dFdy1_dx=diff(dFdy1,x); % Fy'x
d_dFdy1_dy=diff(dFdy1,y); % Fy'y
d_dFdy1_dy1=diff(dFdy1,Dy); % Fy'y'-условие Лежандра
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+d_dFdy1_dy1*D2y;
Euler=simple(dFdy-dFy1dx); % уравнение Эйлера
deqEuler=[char(Euler) '=0']; % добавили =0
Sol=dsolve(deqEuler,'x'); % решаем уравнение Эйлера
if length(Sol)~=1, % решений нет или более одного
  error('Нет решений или более одного решения!');
end
SolLeft=subs(Sol,x,x1); % подставили x1
SolRight=subs(Sol,x,x2); % подставили x2
EqLeft=[char(SolLeft) '=' char(sym(y1))]; % =y1
EqRight=[char(SolRight) '=' char(sym(y2))]; % =y2
Con=solve(EqLeft,EqRight,'C1,C2'); % решаем систему
C1=Con.C1; % присваиваем полученные решения
C2=Con.C2; % символическим константам C1 и C2
```

```
Sol21=vpa(eval(Sol),14); % подставили C1, C2
xpl=linspace(x1,x2); % задаем массив абсцисс
y21=subs(Sol21,x,xpl); % вычислили ординаты
disp('Уравнение экстремали:')
fprintf('y(x)=%s\n',char(Sol21))
F21=subs(F,{y,Dy},{Sol21,diff(Sol21,x)});
J21=eval(int(F21,x,x1,x2)) % значение функционала
Решаем задание 9
Исходные данные:
F(x,y,y')=x^2+y^2+Dy^2
Граничное условие слева: y(-1)=1
Граничное условие справа: y(1)=2
Линия отражения: y=4/5-1/5*x^2
Уравнение экстремали:
y(x)=.42545906411967*sinh(x)+.97208141049585*cosh(x)
J21 =
   4.75035801121746
```

Начинаем формирование файла для вычисления левых частей системы шести нелинейных уравнений, описанных в начале главы. Записываем заголовок функции. Присваиваем аргументы нужным параметрам. Чтобы записать первые 4 уравнения, необходимо в аналитическое решение подставить нужные константы. Поэтому восстанавливаем символические константы и находим решения на левом и правом участках в нужных точках. Формируем первые 5 уравнений системы. Вычисляем необходимые функции для условия трансверсальности (9.3). Формируем строки для 6-го уравнения. Записываем функции в файл и в область вывода.

```
s{1}='function y = MyFunc(x)'; % заголовок
s{2}=['C1l=x(1); C2l=x(2); C1r=x(3); C2r=x(4); '...
  'x0=x(5); y0=x(6); y=x;'];
syms C1 C2 C1l C2l C1r C2r % символические константы
yl=subs(Sol,{C1,C2},{C1l,C2l});
ylx1=subs(yl,x,x1);
ylx0=subs(yl,x,sym('x0'));
yr=subs(Sol,{C1,C2},{C1r,C2r});
yrx0=subs(yr,x,sym('x0'));
yrx2=subs(yr,x,x2);
phix0=subs(phi,x,sym('x0'));
s{3}=['y(1)=' char(ylx1) '-' char(sym(y1)) ';'];
s{4}=['y(2)=' char(ylx0) '-y0;'];
s{5}=['y(3)=' char(yrx0) '-y0;'];
s{6}=['y(4)=' char(yrx2) '-' char(sym(y2)) ';'];
s{7}=['y(5)=' char(phix0) '-y0;'];
```

```
dyl=diff(yl,x);
dyr=diff(yr,x);
dphi=diff(phi,x);
dyl0=subs(dyl,x,sym('x0'));
dyr0=subs(dyr,x,sym('x0'));
dphi0=subs(dphi,x,sym('x0'));
Fl0=subs(F,{x,y,Dy},{sym('x0'),sym('ylx0'),sym('dyl0')});
Fr0=subs(F,{x,y,Dy},{sym('x0'),sym('yrx0'),sym('dyr0')});
dFl0=subs(dFdy1,{x,y,Dy},{sym('x0'),sym('ylx0'),sym('dyl0')});
dFr0=subs(dFdy1,{x,y,Dy},{sym('x0'),sym('yrx0'),sym('dyr0')});
s{8}=['ylx0=' char(ylx0) ';'];
s{9}=['yrx0=' char(yrx0) ';'];
s{10}=['dyl0=' char(dyl0) ';'];
s{11}=['dyr0=' char(dyr0) ';'];
s{12}=['dphi0=' char(dphi0) ';'];
s{13}=['Fl0=' char(Fl0) ';'];
s{14}=['Fr0=' char(Fr0) ';'];
s{15}=['dFl0=' char(dFl0) ';'];
s{16}=['dFr0=' char(dFr0) ';'];
s{17}=['y(6)=(Fl0+(dphi0-dyl0)*dFl0)-(Fr0+(dphi0-dyr0)*dFr0);'];
filename=fullfile(pwd,'MyFunc.m'); % имя файла
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:});
fid=fopen(filename,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записали файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\MyFunc.m:
function y = MyFunc(x)
C1l=x(1); C2l=x(2); C1r=x(3); C2r=x(4); x0=x(5); y0=x(6); y=x;
y(1)=-C1l*sinh(1)+C2l*cosh(1)-1;
y(2)=C1l*sinh(x0)+C2l*cosh(x0)-y0;
y(3)=C1r*sinh(x0)+C2r*cosh(x0)-y0;
y(4)=C1r*sinh(1)+C2r*cosh(1)-2;
y(5)=4/5-1/5*x0^2-y0;
ylx0=C1l*sinh(x0)+C2l*cosh(x0);
yrx0=C1r*sinh(x0)+C2r*cosh(x0);
dyl0=C1l*cosh(x0)+C2l*sinh(x0);
dyr0=C1r*cosh(x0)+C2r*sinh(x0);
dphi0=-2/5*x0;
Fl0=x0^2+ylx0^2+dyl0^2;
Fr0=x0^2+yrx0^2+dyr0^2;
dFl0=2*dyl0;
dFr0=2*dyr0;
y(6)=(Fl0+(dphi0-dyl0)*dFl0)-(Fr0+(dphi0-dyr0)*dFr0);
```

Для быстрой сходимости численного метода решения системы нелинейных уравнений нужно удачно задать начальное приближение. Зададим x_0 посередине интервала $[x_1, x_2]$, а $y_0 = \varphi(x_0)$. Начальные значения произвольных постоянных C_1^l, C_2^l и C_1^r, C_2^r найдем из условия, что линии $y^l(x)$ и $y^r(x)$ являются экстремалями каждая на своем участке. Тем самым пять из шести уравнений будут удовлетворены. Запишем на каждом участке систему уравнений — граничных условий и решим ее. Сформируем начальное приближение.

```
ix0=(x1+x2)/2; % середина интервала
iy0=eval(subs(phi,x,sym(ix0)));
syms C1left C2left C1right C2right
yleft=subs(Sol,{C1,C2},{C1left,C2left});
ylx1=subs(yleft,x,sym(x1));
ylx0=subs(yleft,x,sym(ix0));
Eq1=[char(ylx1) '=' char(sym(y1))];
Eq2=[char(ylx0) '=' char(sym(iy0))];
Con=solve(Eq1,Eq2,'C1left,C2left');
C1left=Con.C1left;
C2left=Con.C2left;
yright=subs(Sol,{C1,C2},{C1right,C2right});
yrx2=subs(yright,x,sym(x2));
yrx0=subs(yright,x,sym(ix0));
Eq1=[char(yrx2) '=' char(sym(y2))];
Eq2=[char(yrx0) '=' char(sym(iy0))];
Con=solve(Eq1,Eq2,'C1right,C2right');
C1right=Con.C1right;
C2right=Con.C2right;
xinit=[eval(C1left);eval(C2left);...
       eval(C1right);eval(C2right);ix0;iy0];
disp('Начальное приближение:')
fprintf(['C1l=%12.6f\nC2l=%12.6f\nC1r=%12.6f\n' ...
  'C2r=%12.6f\nx0=%12.6f\ny0=%12.6f\n'],xinit);
Начальное приближение:
C1l=    0.199510
C2l=    0.800000
C1r=    0.651408
C2r=    0.800000
x0=    0.000000
y0=    0.800000
```

Задаем параметры процесса решения: точность и максимально допустимое число итераций. Решаем систему нелинейных уравнений, анализируем критерий окончания процесса. Печатаем решения. Вычисляем значение функционала.

```
options=optimset('fsolve'); % опции по умолчанию
options=optimset(options,'Display','off',...
  'MaxIter',1000,'TolX',1e-8);
[xzero,fval,exitflag]=fsolve('MyFunc',xinit,options);
if exitflag>0,
  disp('Решение найдено');
elseif exitflag<0,
  disp('Решение не найдено - нет сходимости');
else
  disp(['Проведено максимально допустимое ' ...
        'количество итераций - сходимость медленная']);
end
C1l=vpa(sym(xzero(1)),14);
C2l=vpa(sym(xzero(2)),14);
C1r=vpa(sym(xzero(3)),14);
C2r=vpa(sym(xzero(4)),14);
x0=xzero(5);
y0=xzero(6);
yl=vpa(eval(yl),14);
yr=vpa(eval(yr),14);
disp('Уравнения экстремали:')
fprintf('Yleft(x)=%s\nYright(x)=%s\n',char(yl),char(yr))
fprintf('Точка отражения: x0=%12.6f\n',x0)
F9l=subs(F,{y,Dy},{yl,diff(yl,x)});
F9r=subs(F,{y,Dy},{yr,diff(yr,x)});
J9=eval(int(F9l,x,x1,x0))+eval(int(F9r,x,x0,x2))
Решение найдено
Уравнения экстремали:
Yleft(x)=.21208214123796*sinh(x)+.80957479301323*cosh(x)
Yright(x)=.52020756636393*sinh(x)+.89992150490061*cosh(x)
Точка отражения: x0=   -0.302079
J9 =
   4.78115252754638
```

Заполняем таблицы для графиков функций $y=\varphi(x)$ и решения задачи. Строим графики нашего решения и решения *задания 1 главы 2*. Он показан на рис. 9.3. Функцию $y=\varphi(x)$ строим на небольшом участке слева и справа от точки отражения. Удаляем созданный файл.

```
xi=linspace(x0-0.2,x0+0.2); % аргументы для phi(x)
yi=subs(phi,x,xi); % вычислили phi(x)
xl=xpl(find(xpl<x0)); % аргументы на левом участке
xr=xpl(find(xpl>=x0)); % аргументы на правом участке
y9=[subs(yl,x,xl),subs(yr,x,xr)];
```

```
plot(xpl,y21,'--b',xi,yi,':g',xpl,y9,'-r') % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 9') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm), \phi(\itx\rm)') % метка оси OY
delete(filename) % удалили файл
```

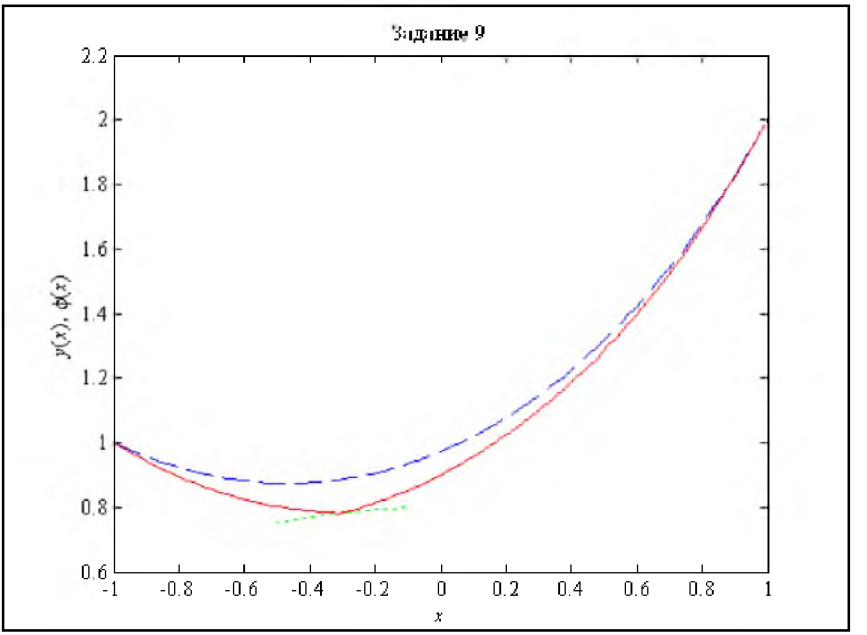


Рис. 9.3. Решение задания 9

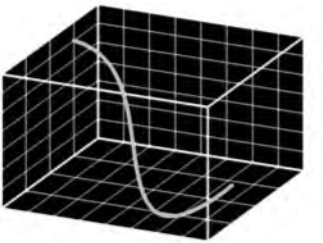
В нашем варианте процесс решения системы нелинейных уравнений нашел решение. А как в вашем варианте? Если процесс плохо сходится, попробуйте вместо решения системы воспользоваться алгоритмом, изложенным после формулы (9.3). Или подберите другую функцию $y = \varphi(x)$, проходящую через точку $M_0(x_0, y_0)$.

Проанализируйте: почему в нашем задании значение функционала получилось больше, чем в *задании 1 главы 2*?

9.4. Задание

Для своего варианта *задания 1 главы 2* найти экстремаль при дополнительном условии, что эта экстремаль касается заданной линии. Уравнения линий отражения есть на компакт-диске — приложении к книге.

ГЛАВА 10



Преломление экстремалей

10.1. Преломление экстремалей в элементарной задаче

Эта задача очень похожа на задачу отражения экстремалей, рассмотренную нами в предыдущей *главе 9*. Чем же они отличаются? Пусть кривая $y=\varphi(x)$ разделяет плоскость xOy на две области, и точки $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$ расположены по разные стороны от этой кривой (рис. 10.1).

Рассмотрим элементарную задачу вариационного исчисления для функционала:

$$J(y)=\int_{x_1}^{x_0} F^l(x,y,y')\,dx+\int_{x_0}^{x_2} F^r(x,y,y')\,dx. \qquad (10.1)$$

Он отличается от функционала, рассмотренного в *главе 9* тем, что в нем подынтегральные функции слева и справа от точки $M_0(x_0, y_0)$ — разные. И все! На рис. 10.1 показаны экстремаль (жирная линия), допустимые функции (тонкие сплошные линии) и линия преломления (штриховая).

Вывод необходимого условия экстремума будет очень похожим на вывод *главы 9*. Раз наш функционал (10.1) достигает экстремума на всем интервале $[x_1, x_2]$, т. е. при варьировании функций $y(x)$ на обоих участках, то он будет достигать экстремума и на более узком классе функций: когда варьируется функция только на одном из интервалов: $[x_1, x_0]$ или $[x_0, x_2]$. При этом можно еще и точку x_0 считать неподвижной, т. е. дополнительно сузить класс допустимых функций. Поэтому каждая из кривых $y^l(x)$, $y^r(x)$ является решением уравнения Эйлера для своей функции F и содержит по 2 произвольные постоянные: C_1^l, C_2^l и C_1^r, C_2^r. Для нахождения этих 4-х констант и 2-х координат точки преломления $M_0(x_0, y_0)$ у нас есть уравнения:

- граничное условие на левом конце $y^l(x_1)=y_1$;
- граничное условие на правом конце $y^r(x_2)=y_2$;
- $y^l(x)$ проходит через точку M_0: $y^l(x_0)=y_0$;
- $y^r(x)$ также проходит через M_0: $y^r(x_0)=y_0$;
- точка M_0 лежит на линии $y=\varphi(x)$: $y_0=\varphi(x_0)$.

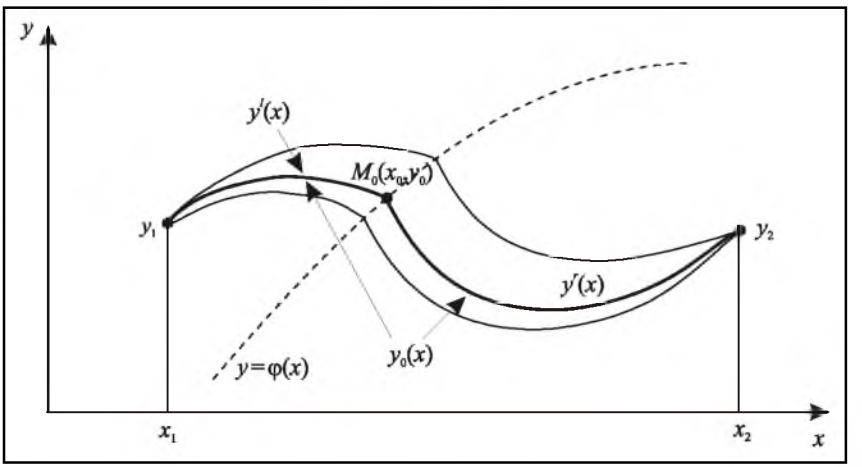


Рис. 10.1. Преломление экстремалей

Недостающее шестое уравнение получим из необходимого условия экстремума функционала: $\delta J=0$. Оно отличается от (9.3) только тем, что в левую и правую части равенства входят разные функции F:

$$\left(F^l+(\varphi'-y')F^l_{y'}\right)\Big|_{x=x_0-0}=\left(F^r+(\varphi'-y')F^r_{y'}\right)\Big|_{x=x_0+0}. \qquad (10.2)$$

Уравнение (10.2) является условием трансверсальности для задачи преломления. Его смысл такой: если точку x_0 двигать по линии $y=\varphi(x)$ и находить пару экстремалей (решений уравнения Эйлера) $y^l(x)$ и $y^r(x)$, удовлетворяющих граничным условиям, то доставлять экстремум функционалу будет та пара функций, для которой выполняется условие трансверсальности (10.2). Как и в задаче отражения, при таком подходе функционал становится функцией одной переменной x_0. Необходимое условие экстремума, записанное в аналитическом виде, дает условие трансверсальности (10.2), численное решение сводится к нахождению экстремума функции одной переменной.

ПРИМЕР 10.1. Преломление света. Пусть свет распространяется в среде, разделенной на 2 части линией раздела $y=\varphi(x)$. С левой стороны от этой ли-

нии скорость света $v_1(x, y)$, а с правой — $v_2(x, y)$. Свет идет из точки $M_1(x_1, y_1)$ в точку $M_2(x_2, y_2)$ с преломлением на линии $y=\varphi(x)$ так, как это показано на рис. 10.1. Требуется исследовать поведение луча в точке преломления $M_0(x_0, y_0)$.

Воспользуемся опять принципом Ферма: свет распространяется по такой траектории $y(x)$, по которой время его движения минимально. Следовательно, задача сводится к минимизации функционала $T(y(x))$ — времени движения по всей траектории $y(x)$: от точки $M_1(x_1, y_1)$ до $M_2(x_2, y_2)$ при преломлении экстремали в точке $M_0(x_0, y_0)$. По аналогии с (9.4) мы можем записать наш функционал:

$$T(y)=\int_{x_1}^{x_0}\frac{\sqrt{1+y'^2}}{v_1(x,y)}\,dx+\int_{x_0}^{x_2}\frac{\sqrt{1+y'^2}}{v_2(x,y)}\,dx\,. \tag{10.3}$$

Разумеется, в каждом из интегралов будет своя функция: $y^l(x)$ и $y^r(x)$. Выражение для условия трансверсальности нами было выведено (9.5):

$$F+(\varphi'-y')F_{y'}=\frac{1+\varphi'y'}{v(x,y)\sqrt{1+y'^2}}. \tag{10.4}$$

В отличие от задачи отражения, здесь скорости света в точке $M_0(x_0, y_0)$ при $x\to x_0-0$ и при $x\to x_0+0$ — разные (10.2), поэтому условие трансверсальности будет иметь вид

$$\left.\frac{1+\varphi'y'}{v_1(x,y)\sqrt{1+y'^2}}\right|_{x_0-0}=\left.\frac{1+\varphi'y'}{v_2(x,y)\sqrt{1+y'^2}}\right|_{x_0+0}. \tag{10.5}$$

Введем в рассмотрение те же углы, что и в предыдущей главе:

- α — угол наклона касательной к кривой $y=\varphi(x)$ к оси Ox в точке x_0; $\operatorname{tg}\alpha=\varphi'(x_0)$;
- β_- — угол наклона экстремали к оси Ox в точке x_0 слева; $\operatorname{tg}\beta_-=(y^l(x_0-0))'$;
- β_+ — угол ее наклона к оси Ox в точке x_0 справа; $\operatorname{tg}\beta_+=(y^r(x_0+0))'$;
- i_- — угол падения;
- i_+ — угол преломления.

Эти углы показаны на рис. 10.2. Учитывая, что угол β_+ — отрицательный, запишем соотношения между ними:

$$i_-=\beta_--\alpha-\frac{\pi}{2};\quad i_+=\frac{\pi}{2}-\alpha+\beta_+\,. \tag{10.6}$$

В этих обозначениях из (10.5) имеем:

$$\frac{1+\operatorname{tg}\alpha\operatorname{tg}\beta_-}{v_1(x,y)\sqrt{1+\operatorname{tg}^2\beta_-}}=\frac{1+\operatorname{tg}\alpha\operatorname{tg}\beta_+}{v_2(x,y)\sqrt{1+\operatorname{tg}^2\beta_+}}. \tag{10.7}$$

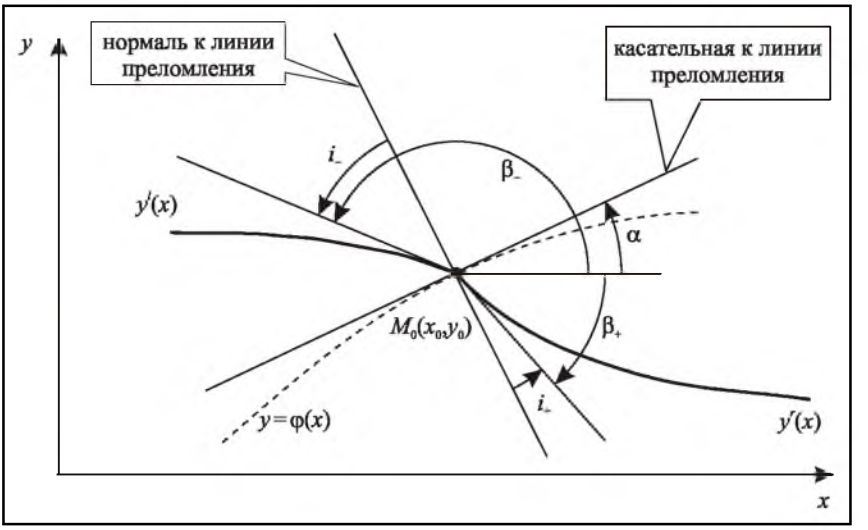


Рис. 10.2. К задаче о преломлении света

Теперь, как и в *главе 9*, проводим тригонометрические преобразования. Поскольку угол β_- — тупой, имеем:

$$\begin{gathered}
-\frac{\cos\beta_-\left(1+\operatorname{tg}\alpha\operatorname{tg}\beta_-\right)}{v_1(x,y)}=\frac{\cos\beta_+\left(1+\operatorname{tg}\alpha\operatorname{tg}\beta_+\right)}{v_2(x,y)};\\
-\frac{\cos\alpha\cos\beta_-+\sin\alpha\sin\beta_-}{v_1(x,y)}=\frac{\cos\alpha\cos\beta_++\sin\alpha\sin\beta_+}{v_2(x,y)};\\
-\frac{\cos(\beta_--\alpha)}{v_1(x,y)}=\frac{\cos(\beta_+-\alpha)}{v_2(x,y)};\\
-\frac{\cos\left(i_-+\dfrac{\pi}{2}\right)}{v_1(x,y)}=\frac{\cos\left(i_+-\dfrac{\pi}{2}\right)}{v_2(x,y)};\\
\frac{\sin i_-}{v_1(x,y)}=\frac{\sin i_+}{v_2(x,y)}.
\end{gathered} \tag{10.8}$$

Окончательно из (10.8) получаем классический закон преломления света:

$$\frac{\sin i_-}{\sin i_+} = \frac{v_1(x,y)}{v_2(x,y)}, \tag{10.9}$$

т. е. отношение синуса угла падения к синусу угла преломления равно отношению скоростей света в точке преломления или, что то же самое, отношению коэффициентов преломления на границе раздела двух сред. Вывод: если свет приходит из точки $M_1(x_1, y_1)$ в точку $M_2(x_2, y_2)$ путем преломления в точке $M_0(x_0, y_0)$, то в этой точке отношение синуса угла падения к синусу угла преломления равно отношению скоростей света в точке преломления (см. также задачу об отражении света в *главе 9*). □

10.2. Вопрос для самопроверки

Выведите условие трансверсальности в задаче преломления для функционала (3.1), зависящего от двух функций, когда экстремаль преломляется на заданной поверхности.

10.3. Пример выполнения задания

Найти экстремаль функционала

$$J(y) = \int_{-1}^{x_0} \left(x^2 + y^2 + y'^2\right) dx + \int_{x_0}^{1} \left(y'^2 + 2y' \operatorname{sh} x - 5x^2\right) dx;$$
$$\begin{cases} y(-1) = 1; \\ y(1) = 3; \end{cases} \quad y_0 = 4 - 2e^{x_0}. \tag{10.10}$$

Если вы внимательно посмотрите на этот пример, то увидите, что слева от неизвестной точки преломления мы взяли подынтегральную функцию и граничное условие из *задания 1 главы 2*, а справа — из *задания 2* той же главы. Будем составлять программу для решения нашего примера на основе программы для *задания главы 9* с использованием других программ. Описываем необходимые переменные и вводим исходные данные.

```
clear all % очистили все
disp('Решаем задание 10') % выводим заголовок задачи
syms x y Dy D2y % описали символические переменные
Fl=x^2+y^2+Dy^2;
x1=-1;
y1=1;
```

```
Fr=Dy^2+2*Dy*sinh(x)-5*x^2;
x2=1;
y2=3;
phi=4-2*exp(x); % линия преломления
disp('Исходные данные на левом участке:')
fprintf('Fleft=%s\n',char(Fl))
fprintf('Граничное условие слева: y(%d)=%d\n',x1,y1)
disp('Исходные данные на правом участке:')
fprintf('Fright=%s\n',char(Fr))
fprintf('Граничное условие справа: y(%d)=%d\n',x2,y2)
fprintf('Линия преломления: phi=%s\n',char(phi))
Решаем задание 10
Исходные данные на левом участке:
Fleft=x^2+y^2+Dy^2
Граничное условие слева: y(-1)=1
Исходные данные на правом участке:
Fright=Dy^2+2*Dy*sinh(x)-5*x^2
Граничное условие справа: y(1)=3
Линия преломления: phi=4-2*exp(x)
```

Составляем и решаем дифференциальное уравнение Эйлера на левом участке. Для этого воспользуемся фрагментом программы для *задания 1 главы 2*, в котором во всех переменных добавим суффикс `l`, т. к. решение ищется на левом участке.

```
dFldy=diff(Fl,y);
dFldy1=diff(Fl,Dy);
d_dFldy1_dx=diff(dFldy1,x); % d(dFl/dy')/dx
d_dFldy1_dy=diff(dFldy1,y); % d(dFl/dy')/dy
d_dFldy1_dy1=diff(dFldy1,Dy); % d(dFl/dy')/dy'
dFly1dx=d_dFldy1_dx+d_dFldy1_dy*Dy+d_dFldy1_dy1*D2y;
EulerL=simple(dFldy-dFly1dx);
deqEulerL=[char(EulerL) '=0']; % уравнение
SolL=dsolve(deqEulerL,'x'); % решаем уравнение Эйлера
if length(SolL)~=1, % решений нет или более одного
  error('Нет решений или более одного решения!');
end
disp('Решение на левом участке:')
fprintf('Yleft=%s\n',char(SolL))
Решение на левом участке:
Yleft=C1*sinh(x)+C2*cosh(x)
```

Для формирования первого интеграла уравнения Эйлера на правом участке используем фрагмент программы для *задания 2 главы 2*, но во всех переменных добавим суффикс `r`.

```
dFrdy1=simple(diff(Fr,Dy)); % dFr/dy'
deqEulerR=[char(dFrdy1) '=C']; % уравнение
SolR=dsolve(deqEulerR,'x'); % решаем
if length(SolR)~=1, % решений нет или более одного
  error('Нет решений или более одного решения!');
end
disp('Решение на правом участке:')
fprintf('Yright=%s\n',char(SolR))
Решение на правом участке:
Yright=-cosh(x)+1/2*C*x+C1
```

Начинаем формирование системы нелинейных уравнений (см. начало главы) для определения произвольных постоянных и координат точки преломления. Записываем заголовок функции, переобозначаем переменные. Описываем необходимые символические переменные. Находим аналитические выражения для решений на обоих интервалах в необходимых точках. Первые 4 уравнения — это граничные условия. Пятое уравнение — точка лежит на кривой $y=\varphi(x)$. Далее формируем условие трансверсальности (10.2). Для этого вычисляем входящие в (10.2) функции в нужных точках, переводим в строки 9 вспомогательных функций и записываем само уравнение. Записываем систему уравнений в файл и в область вывода.

```
s{1}='function y = MyFunc(x)'; % заголовок
s{2}=['C1l=x(1); C2l=x(2); C1r=x(3); C2r=x(4);' ...
  'x0=x(5); y0=x(6); y=x;'];
syms C C1 C2 C1l C2l C1r C2r % описали переменные
yl=subs(SolL,{C1,C2},{C1l,C2l});
ylx1=subs(yl,x,x1);
ylx0=subs(yl,x,sym('x0'));
yr=subs(SolR,{C1,C},{C1r,C2r});
yrx0=subs(yr,x,sym('x0'));
yrx2=subs(yr,x,x2);
phix0=subs(phi,x,sym('x0'));
s{3}=['y(1)=' char(ylx1) '-(' char(sym(yl)) ');'];
s{4}=['y(2)=' char(ylx0) '-y0;'];
s{5}=['y(3)=' char(yrx0) '-y0;'];
s{6}=['y(4)=' char(yrx2) '-(' char(sym(y2)) ');'];
s{7}=['y(5)=' char(phix0) '-y0;'];
dyldx=diff(yl,x);
dyrdx=diff(yr,x);
dphidx=diff(phi,x);
dphi0=subs(dphidx,x,sym('x0'));
dyl0=subs(dyldx,x,sym('x0'));
dyr0=subs(dyrdx,x,sym('x0'));
```

```
Fl0=subs(Fl,{x,y,Dy},{sym('x0'),sym('ylx0'),sym('dyl0')});
Fr0=subs(Fr,{x,y,Dy},{sym('x0'),sym('yrx0'),sym('dyr0')});
dFl0=subs(dFldy1,{x,y,Dy},{sym('x0'),sym('ylx0'),sym('dyl0')});
dFr0=subs(dFrdy1,{x,y,Dy},{sym('x0'),sym('yrx0'),sym('dyr0')});
s{8}=['ylx0=' char(ylx0) ';'];
s{9}=['yrx0=' char(yrx0) ';'];
s{10}=['dyl0=' char(dyl0) ';'];
s{11}=['dyr0=' char(dyr0) ';'];
s{12}=['dphi0=' char(dphi0) ';'];
s{13}=['Fl0=' char(Fl0) ';'];
s{14}=['Fr0=' char(Fr0) ';'];
s{15}=['dFl0=' char(dFl0) ';'];
s{16}=['dFr0=' char(dFr0) ';'];
s{17}=['y(6)=(Fl0+(dphi0-dyl0)*dFl0)-(Fr0+(dphi0-dyr0)*dFr0);'];
filename=fullfile(pwd,'MyFunc.m'); % имя файла
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:});
fid=fopen(filename,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записали файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\MyFunc.m:
function y = MyFunc(x)
C1l=x(1); C2l=x(2); C1r=x(3); C2r=x(4);
x0=x(5); y0=x(6); y=x;
y(1)=-C1l*sinh(1)+C2l*cosh(1)-(1);
y(2)=C1l*sinh(x0)+C2l*cosh(x0)-y0;
y(3)=-cosh(x0)+1/2*C2r*x0+C1r-y0;
y(4)=-cosh(1)+1/2*C2r+C1r-(3);
y(5)=4-2*exp(x0)-y0;
ylx0=C1l*sinh(x0)+C2l*cosh(x0);
yrx0=-cosh(x0)+1/2*C2r*x0+C1r;
dyl0=C1l*cosh(x0)+C2l*sinh(x0);
dyr0=-sinh(x0)+1/2*C2r;
dphi0=-2*exp(x0);
Fl0=x0^2+ylx0^2+dyl0^2;
Fr0=dyr0^2+2*dyr0*sinh(x0)-5*x0^2;
dFl0=2*dyl0;
dFr0=2*dyr0+2*sinh(x0);
y(6)=(Fl0+(dphi0-dyl0)*dFl0)-(Fr0+(dphi0-dyr0)*dFr0);
```

Зададим начальное приближение для процедуры решения системы нелинейных уравнений. Начальную точку $M_0(x_0,\ y_0)$ берем посередине интервала $[x_1, x_2]$ на линии $y=\varphi(x)$. Начальные значения произвольных постоянных C_1^l,

C_2^l и C_1^r, C_2^r находим из условий, что каждая из кривых $y^l(x)$, $y^r(x)$ является экстремалью на своем участке. Запишем на каждом участке систему уравнений — граничных условий и решим ее. Напечатаем полученное начальное приближение.

```
ix0=(x1+x2)/2; % середина интервала
iy0=eval(subs(phi,x,sym(ix0)));
syms C1left C2left C1right C2right
yleft=subs(SolL,{C1,C2},{C1left,C2left});
yleftx1=subs(yleft,x,x1);
yleftx0=subs(yleft,x,ix0);
Eq1=[char(yleftx1) '=' char(sym(y1))];
Eq2=[char(yleftx0) '=' char(sym(iy0))];
Con=solve(Eq1,Eq2,'C1left,C2left'); % решаем
C1left=Con.C1left;
C2left=Con.C2left;
yright=subs(SolR,{C1,C},{C1right,C2right});
yrightx2=subs(yright,x,x2);
yrightx0=subs(yright,x,ix0);
Eq1=[char(yrightx2) '=' char(sym(y2))];
Eq2=[char(yrightx0) '=' char(sym(iy0))];
Con=solve(Eq1,Eq2,'C1right,C2right'); % решаем
C1right=Con.C1right;
C2right=Con.C2right;
xinit=[eval(C1left);eval(C2left);...
  eval(C1right);eval(C2right);ix0;iy0];
disp('Начальное приближение:')
fprintf(['C1l=%12.6f\nC2l=%12.6f\nC1r=%12.6f\n' ...
  'C2r=%12.6f\nx0=%12.6f\ny0=%12.6f\n'],xinit);
```

```
Начальное приближение:
C1l=    1.775152
C2l=    2.000000
C1r=    3.000000
C2r=    3.086161
x0=     0.000000
y0=     2.000000
```

Задаем параметры процесса решения: точность и максимально допустимое число итераций. Решаем систему нелинейных уравнений, анализируем критерий окончания процесса. Находим и печатаем аналитические решения на интервалах.

```
options=optimset('fsolve'); % опции по умолчанию
options=optimset(options,'Display','off',...
  'MaxIter',1000,'TolX',1e-8);
```

```
[xzero,fval,exitflag]=fsolve('MyFunc',xinit,options);
if exitflag>0,
  disp('Решение найдено');
elseif exitflag<0,
  disp('Решение не найдено - нет сходимости');
else
  disp(['Проведено максимально допустимое ' ...
        'количество итераций - сходимость медленная']);
end
C1l=vpa(sym(xzero(1)),14);
C2l=vpa(sym(xzero(2)),14);
C1r=vpa(sym(xzero(3)),14);
C2r=vpa(sym(xzero(4)),14);
x0=xzero(5);
y0=xzero(6);
yl=vpa(eval(yl),14);
yr=vpa(eval(yr),14);
disp('Уравнения экстремали:')
fprintf('Yleft(x)=%s\nYright(x)=%s\n',char(yl),char(yr))
fprintf('Точка преломления x0=%12.6f\n',x0)
Решение найдено
Уравнения экстремали:
Yleft(x)=2.1489817259098*sinh(x)+2.2847061973741*cosh(x)
Yright(x)=-1.*cosh(x)+1.3099032147113*x+3.2331774201031
Точка преломления x0=   -0.071178
```

Заполняем таблицы для построения графиков функции $y=\varphi(x)$ и решения. Строим графики (см. рис. 10.3). Функцию $\varphi(x)$ строим на небольшом участке слева и справа от точки преломления. Удаляем созданный файл.

```
xpl=linspace(x1,x2); % массив абсцисс
xi=linspace(x0-0.2,x0+0.2); % абсциссы phi(x)
yi=subs(phi,x,xi); % phi(x)
xl=xpl(find(xpl<x0)); % абсциссы на левом участке
xr=xpl(find(xpl>=x0)); % абсциссы на правом участке
y10=[subs(yl,x,xl),subs(yr,x,xr)];
plot(xi,yi,':g',xpl,y10,'-r') % рисуем график
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 10') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm), \phi(\itx\rm)')
delete(filename) % удалили файл
```

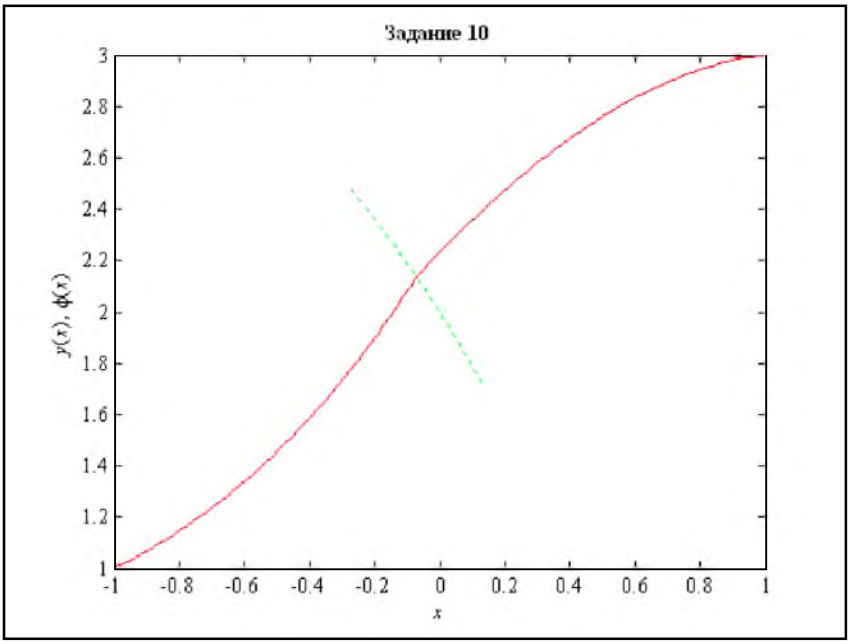


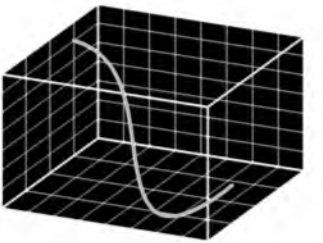
Рис. 10.3. Решение задания 10

В нашем варианте процесс решения системы нелинейных уравнений нашел решение. А как у вас? Если процесс плохо сходится, попробуйте вместо решения системы шести нелинейных уравнений найти экстремум функции одной переменной, варьируя положение x_0, как описано в начале главы. Или подберите другую функцию $y = \varphi(x)$.

10.4. Задание

Решить задачу преломления экстремалей для функционала вида (10.1). Подынтегральную функцию на участке $[x_1, x_0]$ и граничное условие на левом конце $M_1(x_1, y_1)$ взять из своего варианта *задания 1 главы 2*, а функцию на правом участке $[x_0, x_2]$ и правое граничное условие $M_2(x_2, y_2)$ — из своего варианта *задания 2 главы 2*. Линия преломления для каждого варианта есть на диске.

ГЛАВА 11



Экстремали с угловыми точками

11.1. Откуда берутся угловые точки?

Экстремали с точками излома встречаются не только в задачах отражения и преломления экстремалей. Точки излома могут появляться также и в обычных задачах вариационного исчисления без дополнительных условий. Давайте рассмотрим пример.

ПРИМЕР 11.1. Решить вариационную задачу:

$$J(y)=\int_0^2 y'^2\left(1-y'\right)^2\,dx;\quad \begin{cases} y(0)=0; \\ y(2)=1. \end{cases} \tag{11.1}$$

Подынтегральная функция $F(x,y,y')=y'^2(1-y')^2\geq 0$; поэтому на любой линии $J(y)\geq 0$. Если на какой-либо линии окажется, что $J(y)=0$, то понятно, что это минимум. Рассмотрим функцию

$$y(x)=\begin{cases} x; & 0\leq x\leq 1; \\ 1; & 1<x\leq 2, \end{cases} \tag{11.2}$$

график которой показан на рис. 11.1 жирной линией. На этой функции $J(y)=0$, т. е. она доставляет минимум нашему функционалу, а значит, является экстремалью.

Но функция (11.2) — не единственная экстремаль функционала (11.1). Такой же минимум $J(y)=0$ доставляют и другие функции, состоящие из участков прямых $y=x+C$ и $y=C$. Они показаны на рис. 11.1 тонкими линиями (сплошными и штриховыми). Все эти линии непрерывные, но с изломами.

Если взять гладкие кривые, то на них $J(y)$ можно как угодно близко приблизить к нулю, но минимальное, нулевое значение достигается только на лома-

ных линиях. Действительно, на кривых участках y' — переменная, значит $y' \neq 0$ и $y' \neq 1$. Поэтому $F(x, y, y') = y'^2(1-y')^2 > 0$; и $J(y) > 0$. □

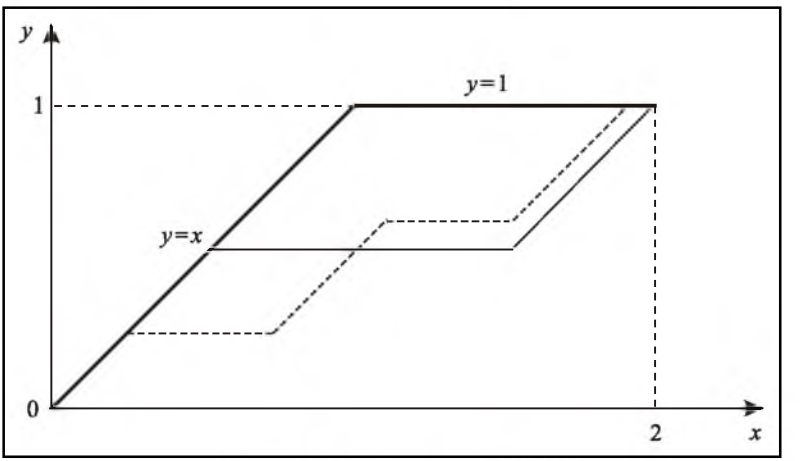


Рис. 11.1. Решение примера 11.1

Выведем условия, при которых возможно существование экстремалей с угловыми точками. Рассмотрим элементарную задачу вариационного исчисления (2.1—2.2). Пусть на некоторой непрерывной кривой с изломами $y(x)$ достигается экстремум. Очевидно, что тогда и на каждом гладком участке также будет достигаться экстремум. Действительно, мы ведь можем рассмотреть более узкий класс функций: когда все участки, кроме одного — фиксированные, а только один варьируется. На этом более узком классе функций также достигается экстремум. Следовательно, каждый из гладких участков обязательно удовлетворяет уравнению Эйлера (2.9).

Будем считать для упрощения, что есть только одна точка излома, как это показано на рис. 11.2 (здесь показаны экстремаль — жирная линия и близкие кривые — тонкие). Если это не так, то применим наши рассуждения к каждой угловой точке.

Таким образом, мы имеем задачу вариационного исчисления для функционала

$$J(y) = \int_{x_1}^{x_0} F(x, y, y')\, dx + \int_{x_0}^{x_2} F(x, y, y')\, dx \qquad (11.3)$$

при заданных граничных условиях вида (2.2) на обоих концах интервала $[x_1, x_2]$. Обе линии: $y^l(x)$ и $y^r(x)$ являются решениями уравнения Эйлера (2.9). Каждая из них содержит по 2 произвольные постоянные (всего 4). Для их на-

хождения у нас есть 2 граничных условия вида (2.2). Еще 2 недостающие уравнения мы получим из необходимого условия экстремума: $\delta J = 0$. Обозначим вариации функционала на участках соответственно δJ_1 и δJ_2. Каждая из этих вариаций вызывается вариацией функции δy вместе с ее производной $\delta y'$ и вариацией точки M_0: δx_0 и δy_0, причем δx_0 и δy_0 — обе независимые (две независимые величины дадут два условия). В *главе 8* мы выводили подобную вариацию. Если воспользоваться средней частью формулы (8.7), заменив в ней нижний индекс 2 на 0, то это и будет δJ_1: в наших обозначениях эта вариация выглядит так:

$$\delta J_1 = \left(F - y'F_{y'}\right)\Big|_{x_0-0} \delta x_0 + F_{y'}\Big|_{x_0-0} \delta y_0 . \qquad (11.4)$$

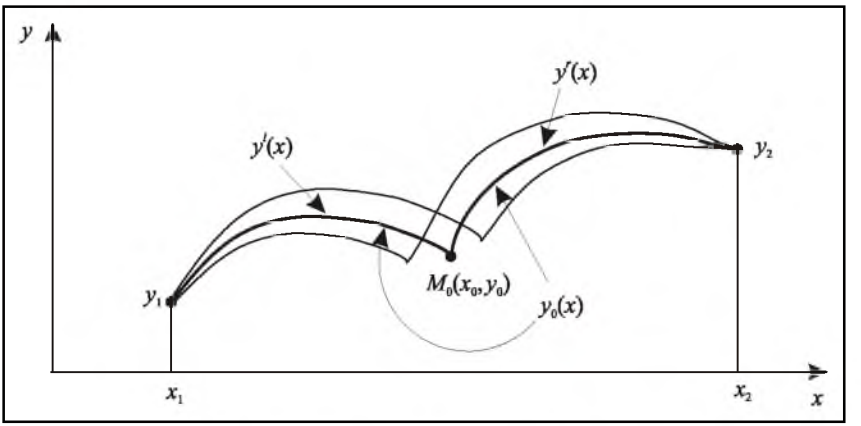


Рис. 11.2. Экстремаль и допустимые функции

Вариация δJ_2 отличается от δJ_1 тем, что M_0 будет не правой, а левой точкой. Вы ведь выводили условие трансверсальности для подвижной точки слева? Вот как оно выглядит:

$$\delta J_2 = -\left(F - y'F_{y'}\right)\Big|_{x_0+0} \delta x_0 - F_{y'}\Big|_{x_0+0} \delta y_0 . \qquad (11.5)$$

Складывая (11.4) и (11.5), получим:

$$\delta J = \left(\left(F - y'F_{y'}\right)\Big|_{x_0-0} - \left(F - y'F_{y'}\right)\Big|_{x_0+0}\right)\delta x_0 + \left(F_{y'}\Big|_{x_0-0} - F_{y'}\Big|_{x_0+0}\right)\delta y_0 = 0. \qquad (11.6)$$

Здесь δx_0 и δy_0 — независимые, поэтому имеем 2 недостающих условия для нахождения произвольных постоянных:

$$\begin{cases} \left(F - y'F_{y'}\right)\Big|_{x_0-0} = \left(F - y'F_{y'}\right)\Big|_{x_0+0}; \\ F_{y'}\Big|_{x_0-0} = F_{y'}\Big|_{x_0+0}. \end{cases} \tag{11.7}$$

Если нам удастся найти какие-либо точки M_0, удовлетворяющие условиям (11.7) при разных значениях y' слева и справа, то они могут быть точками излома экстремали. Если же условия (11.7) выполняются только при одинаковых y' слева и справа, то экстремаль в этой точке может быть только гладкой.

ПРИМЕР 11.2. Найти экстремали с изломами:

$$J(y) = \int_0^a \left(y'^2 - y^2\right) dx; \quad a > 0. \tag{11.8}$$

Проверяем 2-е условие из (11.7):

$$F_{y'}\Big|_{x_0-0} = F_{y'}\Big|_{x_0+0}; \quad 2y'\Big|_{x_0-0} = 2y'\Big|_{x_0+0}. \tag{11.9}$$

Получили условие гладкости кривой. Изломов нет, экстремалями могут быть только гладкие кривые. □

ПРИМЕР 11.1 (продолжение). Исследуем подробнее пример 11.1. Функционал (11.1) зависит только от y'. Согласно *разделу 2.2.4* экстремалями могут быть только прямые:

$$y = C_1 x + C_2. \tag{11.10}$$

Выведем условия (11.7). Найдем вначале:

$$F_{y'} = 2y'(1-y')^2 + y'^2 2(1-y')(-1) = 2y'(1-y')(1-y'-y') = 2y'(1-y')(1-2y');$$

$$F - y'F_{y'} = y'^2(1-y')^2 - 2y'(1-y')(1-2y') = y'^2(1-y')(1-y'-2(1-2y')) =$$
$$= y'^2(1-y')(-1+3y') = -y'^2(1-y')(1-3y').$$

Теперь записываем условия (11.7):

$$\begin{cases} y'^2(1-y')(1-3y')\Big|_{x_0-0} = y'^2(1-y')(1-3y')\Big|_{x_0+0}; \\ y'(1-y')(1-2y')\Big|_{x_0-0} = y'(1-y')(1-2y')\Big|_{x_0+0}. \end{cases} \tag{11.11}$$

Возможны ли здесь решения вида $y'(x_0-0) \neq y'(x_0+0)$? Решим эту систему с помощью MATLAB. Обозначим $y'(x_0-0) = a$; $y'(x_0+0) = b$. Составим строки-уравнения системы, решим систему. Напечатаем решения.

```
clear all % очистили все
eq1='a^2*(1-a)*(1-3*a)=b^2*(1-b)*(1-3*b)';
eq2='a*(1-a)*(1-2*a)=b*(1-b)*(1-2*b)';
sol=solve(eq1,eq2,'a,b'); % решаем систему
n=length(sol.a); % количество решений
for k=1:n,
  fprintf('a(%d)=%s\n',k,char(sol.a(k)));
  fprintf('b(%d)=%s\n',k,char(sol.b(k)));
end
a(1)=b
b(1)=b
a(2)=1
b(2)=0
a(3)=0
b(3)=1
```

Посмотрите: первое решение — тривиальное: $a=b$. А вот второе и третье — это то, что нам нужно: $\{a=1; b=0\}$ и $\{a=0; b=1\}$. То есть нетривиальные решения системы (11.11) — это:

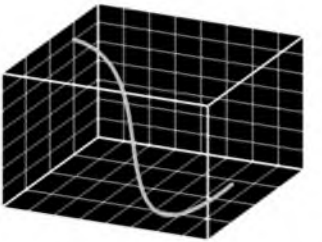
$$\begin{cases} y'(x_0-0)=0; \\ y'(x_0+0)=1; \end{cases} \qquad \begin{cases} y'(x_0-0)=1; \\ y'(x_0+0)=0. \end{cases} \tag{11.12}$$

Таким образом, ломаные экстремали могут быть только отрезками прямых вида $y=x+C$ и $y=C$.

11.2. Вопросы для самопроверки

1. Какую вариационную задачу мы решаем?
2. Выведите условия существования экстремалей с изломами для функционала (3.1), зависящего от двух функций.

ГЛАВА 12



Односторонние вариации

12.1. Запрет на пребывание экстремали в заданной области

Рассмотрим элементарную задачу вариационного исчисления для функционала (2.1) при граничных условиях (2.2) и при дополнительном условии: допустимая кривая не должна попасть в область D, ограниченную линией L: $y=\varphi(x)$ (рис. 12.1). Будем предполагать, что решение задачи нетривиальное, т. е. если область D не запретная, то экстремаль проходит через нее, как показано тонкой линией.

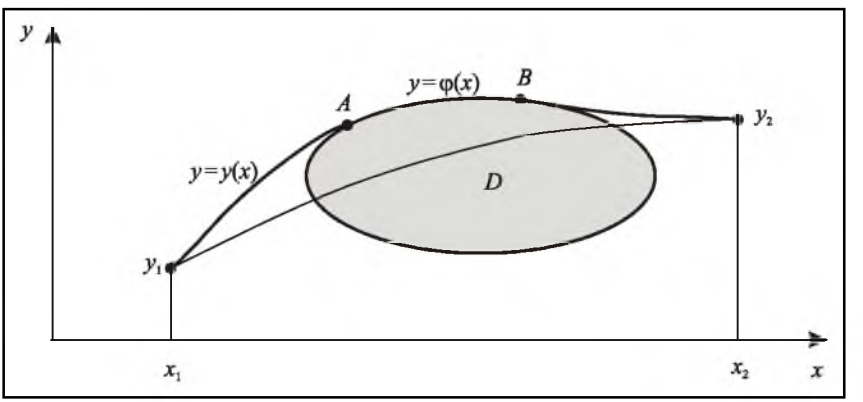


Рис. 12.1. Односторонние вариации

Для этой задачи экстремаль состоит из дуг, лежащих вне D, и частей L. При этом на границе области D (дуга AB) возможны только *односторонние вариации* функции $y(x)$: в сторону вне области D. Для упрощения выкладок рас-

смотрим случай, когда имеется только одна точка сопряжения M_0 (рис. 12.2). Допустимыми в этой задаче являются функции, которые от точки $M_1(x_1, y_1)$ до M_0 проходят вне области D, а на участке $[x_0, x_2]$ совпадают с границей D.

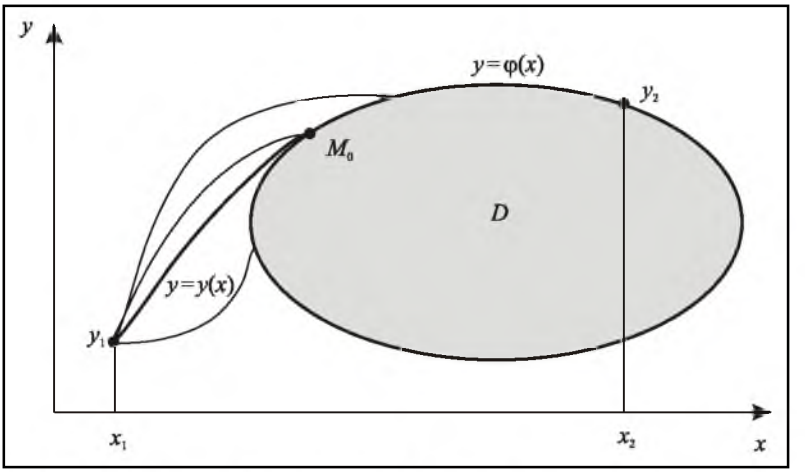


Рис. 12.2. Экстремаль и допустимые функции: одна точка сопряжения

Как обычно, применим тот принцип, который мы уже раньше использовали: если функционал достигает экстремума на классе функций с подвижной точкой x_0, то он тем более будет достигать экстремума на более узком классе функций: когда точка x_0 неподвижна. Следовательно, на интервале $[x_1, x_0]$ искомая экстремаль должна удовлетворять дифференциальному уравнению Эйлера (2.9). В решение этого уравнения входят 2 произвольные константы C_1 и C_2. Кроме того, неизвестны 2 координаты точки касания M_0. Для нахождения этих четырех величин у нас есть уравнения:

- граничное условие на левом конце: $y(x_1, C_1, C_2) = y_1$;
- граничное условие в неизвестной точке M_0: $y(x_0, C_1, C_2) = y_0$;
- точка M_0 находится на линии L: $y_0 = \varphi(x_0)$.

Недостающее 4-е уравнение (условие трансверсальности) выведем из необходимого условия экстремума функционала: $\delta J = 0$. Наш функционал состоит из суммы двух слагаемых вида (2.1): на интервалах $[x_1, x_0]$ и $[x_0, x_2]$:

$$J\big(y(x)\big) = \int_{x_1}^{x_0} F\big(x, y, y'\big)\,dx + \int_{x_0}^{x_2} F\big(x, y, y'\big)\,dx\,. \qquad (12.1)$$

Вариация функционала на первом участке $[x_1, x_0]$ вызывается вариацией функции $y(x)$ вместе с ее производной и вариацией правой точки M_0. На вто-

ром же участке варьируется только положение левой точки x_0, а функция $y(x)$ здесь не варьируется: мы находимся на линии $y=\varphi(x)$ — границе области D. Вычислим вариации функционала δJ_1 на участке M_1M_0 и δJ_2 на участке M_0M_2. Первую из этих величин δJ_1 мы находили ранее при выводе условий трансверсальности в *главе 8*. Это формула (8.9), в которой вместо x_2 стоит x_0.

$$\delta J_1 = \left(F + (\varphi' - y')F_{y'}\right)\Big|_{x=x_0} \delta x_0 \ . \tag{12.2}$$

Наша задача отличается от рассмотренной в *разделе 8.1* тем, что в ней есть еще второй участок — кусочек дуги линии L. На этом втором участке функция не варьируется, варьируется только положение точки x_0. Поэтому

$$\delta J_2 = \int\limits_{x_0+\delta x_0}^{x_2} F(x,y,y')\,dx - \int\limits_{x_0}^{x_2} F(x,y,y')\,dx = -\int\limits_{x_0}^{x_0+\delta x_0} F(x,\varphi(x),\varphi'(x))\,dx \ . \tag{12.3}$$

Интервал интегрирования здесь — малый, поэтому по теореме о среднем с точностью до бесконечно малых величин высшего порядка малости имеем:

$$\delta J_2 = -F(x,\varphi(x),\varphi'(x))\Big|_{x=x_0} \delta x_0 \ . \tag{12.4}$$

Сложим (12.2) и (12.4) — найдем вариацию нашего функционала. Так как δx_0 произвольная, то множитель при δx_0 равен 0:

$$\left(F(x,y,y') - F(x,\varphi,\varphi') + (\varphi' - y')F_{y'}\right)\Big|_{x=x_0} = 0 \ . \tag{12.5}$$

Формально мы уже получили условие трансверсальности для нашей задачи, но его можно упростить. Разложим первое слагаемое в (12.5) в ряд Тейлора по 3-му аргументу в окрестности φ', и удержим малые только 1-го порядка. Это можно делать, т. к. мы находим вариацию функционала — линейную часть его приращения. Имеем:

$$F(x,y,y') = F(x,y,\varphi') + F_{y'}(x,y,\varphi')(y' - \varphi'). \tag{12.6}$$

Подставим (12.6) в (12.5) и учтем, что в точке сопряжения x_0 выполняется условие $y(x)=\varphi(x)$:

$$\left(F_{y'}(x,y,\varphi')(y' - \varphi') - (y' - \varphi')F_{y'}(x,y,y')\right)\Big|_{x=x_0} = 0 \ . \tag{12.7}$$

Применим еще раз теорему о среднем к выражению (12.7). С точностью до бесконечно малых высшего порядка

$$\left((y' - \varphi')^2 F_{y'y'}(x,y,y')\right)\Big|_{x=x_0} = 0 \ . \tag{12.8}$$

Если экстремум на данной экстремали действительно достигается, то выполняется достаточное условие Лежандра *(см. главу 13)*, и $F_{y'y'} \neq 0$. Тогда из (12.8) получаем условие сопряжения

$$y'(x_0) = \varphi'(x_0). \tag{12.9}$$

Условие сопряжения (12.9) является условием трансверсальности для нашей задачи. Его смысл следующий. Если на кривой $y = \varphi(x)$ выбирать различные точки x_0, по полученным точкам x_0 строить экстремали на участке $[x_1, x_0]$ и добавлять к ним участки дуг функции $\varphi(x)$ на интервале $[x_0, x_2]$, то доставлять экстремум функционалу (12.1) будет та линия, для которой выполняется условие гладкости в точке сопряжения (12.9).

Таким образом, для нахождения двух произвольных констант, входящих в выражение для экстремали, и координат точки сопряжения $M_0(x_0, y_0)$ у нас есть 4 уравнения:

$$\begin{cases} y(x_1, C_1, C_2) = y_1; \\ y(x_0, C_1, C_2) = y_0; \\ y_0 = \varphi(x_0); \\ y'(x_0) = \varphi'(x_0). \end{cases} \tag{12.10}$$

Другой путь решения данной задачи — вместо решения системы 4-х уравнений (12.10) минимизировать функцию одной переменной x_0. Она строится следующим образом:

1. По x_0 находим $y_0 = \varphi(x_0)$.
2. Берем решение уравнения Эйлера для нашего функционала и подставляем в него граничные условия $M_1(x_1, y_1)$ и $M_0(x_0, y_0)$.
3. Вычисляем значение функционала (2.1) на найденной кривой в интервале M_1M_0 и прибавляем к нему значение функционала на функции $y = \varphi(x)$ в интервале M_0M_2.

Далее минимизируем полученный функционал, который теперь является функцией одной переменной x_0. Аналитическое выражение для необходимого условия экстремума дает нам условие трансверсальности (12.9).

ПРИМЕР 12.1. Решить вариационную задачу:

$$J(y) = \int_0^{10} \sqrt{1 + y'^2}\, dx; \quad \begin{cases} y(0) = 0; \\ y(10) = 5; \end{cases} \quad (x-10)^2 + y^2 \geq 25. \tag{12.11}$$

Подынтегральная функция в данном функционале зависит только от y', поэтому экстремалями могут быть только прямые (2.60). Если бы не было огра-

ничения-запрета на нахождение экстремали внутри круга, то очевидное решение — прямая $y=x/2$. Она удовлетворяет граничным условиям при $x=0$ и $x=10$, и доставляет минимум нашему функционалу. Но из-за наличия дополнительного ограничения нужно решить систему (12.10). Найдем вначале производную от линии $y=\varphi(x)$, которой является в нашем случае окружность $(x-10)^2+y^2=25$:

$$2(x-10)+2yy'=0; \quad y'=\varphi'=\frac{10-x}{y}. \tag{12.12}$$

Теперь составляем систему вида (12.10):

$$\begin{cases} C_2=0; \\ C_1x_0+C_2=y_0; \\ (x_0-10)^2+y_0^2=25; \\ C_1=\dfrac{10-x_0}{y_0}. \end{cases} \tag{12.13}$$

Исключаем $C_2=0$:

$$\begin{cases} y_0=C_1x_0; \\ C_1^2x_0=10-x_0; \\ (x_0-10)^2+C_1^2x_0^2=25. \end{cases} \tag{12.14}$$

Решаем 3-е уравнение системы (12.14) после подстановки в него 2-го — находим x_0:

$$(x_0-10)^2+(10-x_0)\,x_0=25; \quad -10x_0=-75; \quad x_0=7{,}5. \tag{12.15}$$

Теперь из 2-го уравнения (12.14) находим C_1, а из 1-го — y_0:

$$7{,}5C_1^2=2{,}5; \quad C_1^2=\frac{1}{3}; \quad C_1=\pm\frac{1}{\sqrt{3}}; \quad y_0=\pm\frac{7{,}5}{\sqrt{3}}. \tag{12.16}$$

Из граничного условия на правом конце имеем решение:

$$y=\frac{x}{\sqrt{3}}. \tag{12.17}$$

Оно показано на рис. 12.3. Показана также экстремаль для задачи без запретной области. □

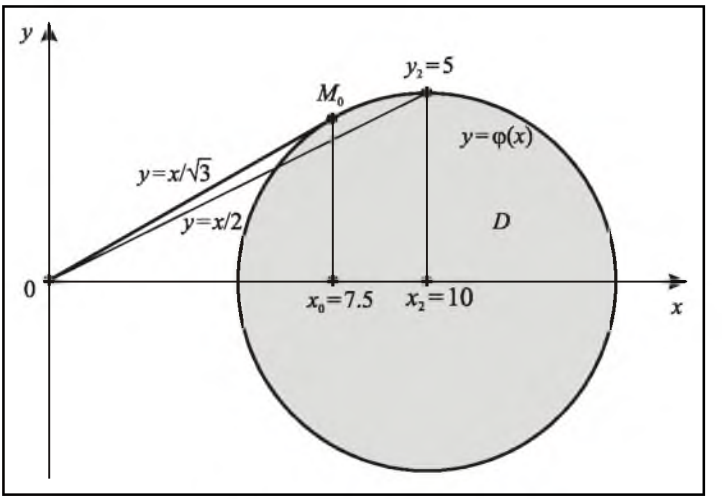


Рис. 12.3. Решение примера 12.1

12.2. Вопрос для самопроверки

Выведите условия сопряжения функционала (3.1), зависящего от двух функций, когда в пространстве задана поверхность, ограничивающая некоторую запретную область.

12.3. Пример выполнения задания

Найти экстремаль функционала, рассмотренного в *задании 1 главы 2*, при дополнительном ограничении: запрете на вхождение экстремали в заданную область:

$$J(y)=\int_{-1}^{1}\left(x^2+y^2+y'^2\right)dx;\quad \begin{cases} y(-1)=1; \\ y(1)=2; \end{cases}\quad y\geq 2-(x-1)^2. \qquad (12.18)$$

Сравнить решение с решением *задания 1 главы 2*. Вычислить значения функционалов на обеих функциях. Построить графики.

Составим программу для решения данного примера на основе созданных ранее программ. Описываем необходимые переменные и вводим исходные данные. Оставляем решение *задания 1 главы 2*.

```
clear all % очистили память
disp('Решаем задание 12')
syms x y Dy D2y % описали символические переменные
```

```
F=x^2+y^2+Dy^2;
x1=-1;
y1=1;
x2=1;
y2=2;
phi=2-(x-1)^2; % граница запретной области
disp('Исходные данные:')
fprintf('F(x,y,y'')=%s\n',char(F))
fprintf('Граничное условие слева: y(%d)=%d\n',x1,y1)
fprintf('Граничное условие справа: y(%d)=%d\n',x2,y2)
fprintf('Граница запретной области: y=%s\n',char(phi))
dFdy=diff(F,y); % вычислили Fy
dFdy1=diff(F,Dy); % вычислили Fy'
d_dFdy1_dx=diff(dFdy1,x); % Fy'x
d_dFdy1_dy=diff(dFdy1,y); % Fy'y
d_dFdy1_dy1=diff(dFdy1,Dy); % Fy'y'-условие Лежандра
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+d_dFdy1_dy1*D2y;
Euler=simple(dFdy-dFy1dx);
deqEuler=[char(Euler) '=0']; % добавили =0
Sol=dsolve(deqEuler,'x'); % решаем уравнение Эйлера
if length(Sol)~=1 % решений нет или более одного
  error('Нет решений или более одного решения!');
end
SolLeft=subs(Sol,x,x1); % подставили x1
SolRight=subs(Sol,x,x2); % подставили x2
EqLeft=[char(SolLeft) '=' char(sym(y1))]; % =y1
EqRight=[char(SolRight) '=' char(sym(y2))]; % =y2
Con=solve(EqLeft,EqRight,'C1,C2'); % решаем систему уравнений
C1=Con.C1; % присваиваем полученные решения
C2=Con.C2; % символическим константам C1 и C2
Sol21=vpa(eval(Sol),14); % подставляем C1,C2, вычисляем с 14 зн.
xpl=linspace(x1,x2); % задаем массив абсцисс
y21=subs(Sol21,x,xpl); % вычислили ординаты
Решаем задание 12
Исходные данные:
F(x,y,y')=x^2+y^2+Dy^2
Граничное условие слева: y(-1)=1
Граничное условие справа: y(1)=2
Граница запретной области: y=2-(x-1)^2
```

Сохраняем численные значения найденных произвольных постоянных. Они будут нам нужны как начальное приближение при решении системы нелинейных уравнений вида (12.10). Формируем эту систему. Записываем ее в файл и в область вывода.

```
C1e=eval(C1);
C2e=eval(C2);
s{1}='function y = MyFunc(x)'; % заголовок
s{2}='C1=x(1); C2=x(2); x0=x(3); y0=x(4); y=x;';
syms C1 C2 % описали символические константы
yx1=subs(Sol,x,x1);
yx0=subs(Sol,x,sym('x0'));
dydx=diff(Sol,x);
dphidx=diff(phi,x);
dydxx0=subs(dydx,x,sym('x0'));
phix0=subs(phi,x,sym('x0'));
dphidxx0=subs(dphidx,x,sym('x0'));
s{3}=['y(1)=' char(yx1) '-(' char(sym(y1)) ');'];
s{4}=['y(2)=' char(yx0) '-y0;']; % уравнение 2
s{5}=['y(3)=' char(phix0) '-y0;']; % уравнение 3
s{6}=['dydxx0=', char(dydxx0), ';'];
s{7}=['dphidxx0=', char(dphidxx0), ';'];
s{8}='y(4)=dydxx0-dphidxx0;';
filename=fullfile(pwd,'MyFunc.m');
disp(['Текст файла ' filename ':']) % файл
fprintf('%s\n',s{:})
fid=fopen(filename,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записали в файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\MyFunc.m:
function y = MyFunc(x)
C1=x(1); C2=x(2); x0=x(3); y0=x(4); y=x;
y(1)=-C1*sinh(1)+C2*cosh(1)-(1);
y(2)=C1*sinh(x0)+C2*cosh(x0)-y0;
y(3)=2-(x0-1)^2-y0;
dydxx0=C1*cosh(x0)+C2*sinh(x0);
dphidxx0=-2*x0+2;
y(4)=dydxx0-dphidxx0;
```

В качестве начального приближения для произвольных постоянных берем их значения из решения *задания 1 главы 2*. Начальное приближение для точки M_0 — точка M_2. Решаем полученную систему нелинейных уравнений. Печатаем аналитическое решение.

```
xinit=[C1e;C2e;x2;y2]; % начальное приближение
options=optimset('fsolve'); % опции по умолчанию
options=optimset(options,'Display','off',...
  'MaxIter',1000,'TolX',1e-8);
[xzero,fval,exitflag]=fsolve('MyFunc',xinit,options);
```

```
if exitflag>0,
  disp('Решение найдено');
elseif exitflag<0,
  disp('Решение не найдено - нет сходимости');
else
  disp(['Проведено максимально допустимое число ' ...
        'итераций - сходимость медленная']);
end
C1=vpa(sym(xzero(1)),14);
C2=vpa(sym(xzero(2)),14);
x0=xzero(3);
y0=xzero(4);
Sol12=vpa(eval(Sol),14); % аналитическое решение
disp('Уравнение экстремали:')
fprintf('y=%s\n',char(Sol12))
fprintf('Точка касания: x0=%12.6f\n',x0)
Решение найдено
Уравнение экстремали:
y=.76309770719858*sinh(x)+1.2292250278898*cosh(x)
Точка касания: x0=    0.364112
```

Вычисляем и печатаем значения функционалов на обеих кривых. Заполняем таблицы для функции $\varphi(x)$ и для решения. Строим графики: решения нашего примера, решения *задания 1 главы 2* (для сравнения) и функции $y=\varphi(x)$. Результат показан на рис. 12.4.

```
F21=subs(F,{y,Dy},{Sol21,diff(Sol21,x)});
J21=eval(int(F21,x,x1,x2))
F12left=subs(F,{y,Dy},{Sol12,diff(Sol12,x)});
F12right=subs(F,{y,Dy},{phi,diff(phi,x)});
J12=eval(int(F12left,x,x1,x0))+eval(int(F12right,x,x0,x2))
xi=linspace(x0-0.4,x2); % аргументы для phi(x)
yi=subs(phi,x,xi); % phi(x)
xl=xpl(find(xpl<x0)); % аргументы слева от точки касания
xr=xpl(find(xpl>=x0)); % аргументы справа от точки касания
y12=[subs(Sol12,x,xl),subs(phi,x,xr)];
plot(xpl,y21,'--b',xi,yi,':g',xpl,y12,'-r',x0,y0,'mo')
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 12') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm), \phi(\itx\rm)') % метка оси OY
delete(filename) % удалили файл
```

```
J21 =
   4.75035801121746
J12 =
   5.52738105357257
```

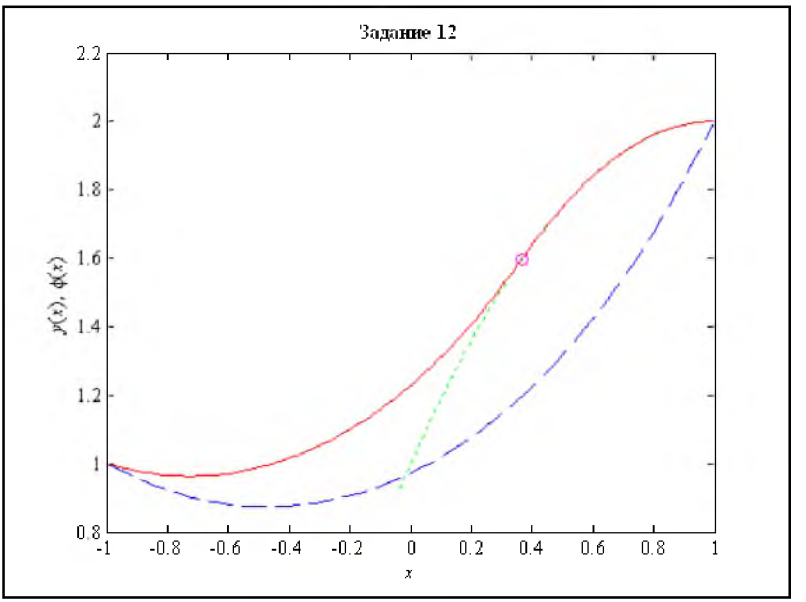


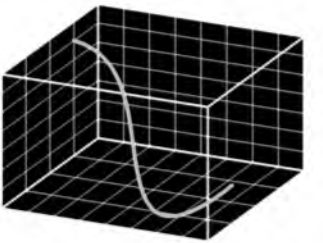
Рис. 12.4. Решение задания 12

В нашем варианте процесс решения системы нелинейных уравнений (12.10) успешно проработал и нашел решение. А как в вашем варианте? Если процесс плохо сходится, попробуйте вместо решения системы (12.10) воспользоваться алгоритмом, изложенным после формулы (12.10). Или подберите другую функцию $y = \varphi(x)$, проходящую через точку $M_2(x_2, y_2)$.

12.4. Задание

Решить *задание 1 главы 2* при дополнительном условии, что экстремаль не может заходить в запрещенную область D. Уравнение границы области для каждого варианта имеется на диске. Сравнить результат с решением *задания 1 главы 2*.

ГЛАВА 13



Достаточные условия экстремума

13.1. Собственное и центральное поле

Ну, вот мы, наконец, добрались и до вопросов, связанных с достаточными условиями экстремума функционалов. Мы ограничимся только простейшим случаем: элементарной задачей (2.1—2.2). Пусть такая задача решена, т. е. найдена экстремаль, и нам надо проверить: действительно ли на этой экстремали достигается экстремум функционала $J(y)$, и если да, то какой (т. е. минимум или максимум, сильный или слабый, а если слабый, то какого порядка).

Рассмотрение этого вопроса мы начнем с некоторых определений.

Определение 13.1. Пусть задано однопараметрическое семейство кривых $y=y(x, C)$. Говорят, что это семейство образует *поле* (или *собственное поле*) в некоторой области D, если через любую точку $M_0(x_0, y_0) \in D$ проходит единственная кривая семейства. □

Математически это записывается так:

$$\forall (x_0, y_0) \in D \quad \exists! C_0 : y(x_0, C_0) = y_0, \tag{13.1}$$

т. е. для любой точки $M_0(x_0, y_0)$ из области D существует единственное значение $C=C_0$, такое, что кривая семейства $y=y(x, C_0)$ проходит через точку $M_0(x_0, y_0)$.

Определение 13.2. Угловой коэффициент касательной к линиям семейства в любой точке поля $p(x, y)$ называется *функцией наклона поля.* □

ПРИМЕР 13.1. Пусть задана область D — круг $x^2+y^2 \le 1$. Рассмотрим в этой области семейство кривых: $y=x+C$. Данное семейство образует поле (рис. 13.1). Функция наклона поля в любой точке $p(x, y)=1$, т. к. все линии семейства — прямые. □

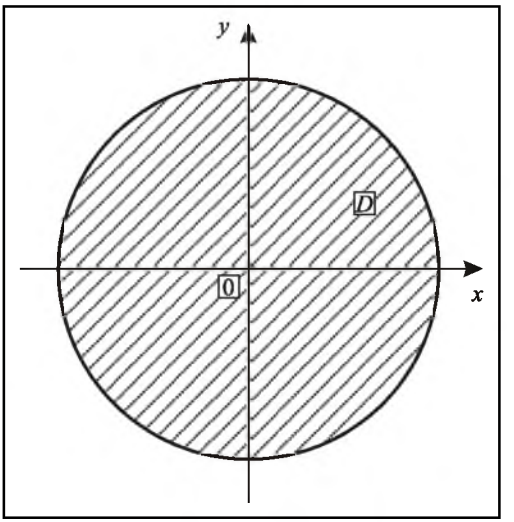


Рис. 13.1. Собственное поле в примере 13.1

ПРИМЕР 13.2. В той же области D: $x^2+y^2\le 1$ задано 1-параметрическое семейство линий $y=(x-C)^2-1$. Они поля не образуют (рис. 13.2), т. к. через каждую точку области D проходит не одна, а две линии семейства. □

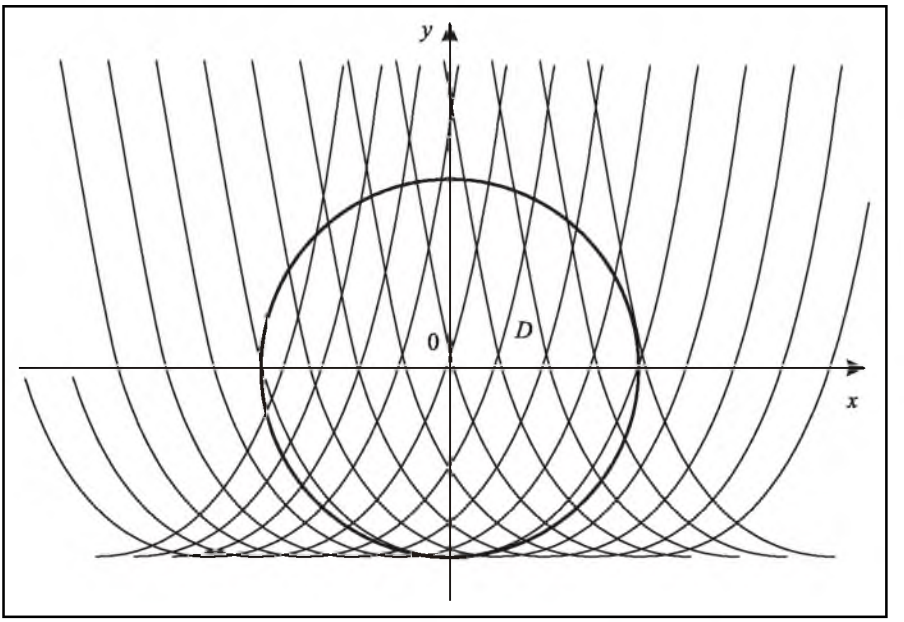


Рис. 13.2. Эти линии из примера 13.2 поля не образуют

Следующее определение, с которым мы познакомимся, — это центральное поле.

Определение 13.3. Пусть все линии семейства $y=y(x,C)$ проходят через некоторую точку $M_0(x_0,y_0)\in D$. Очевидно, такое семейство не образует поля в D. Но, если кривые семейства $y=y(x,C)$ полностью покрывают всю область D_1 и нигде больше, кроме точки $M_0(x_0,y_0)$, не пересекаются, то такое семейство называется *центральным полем*. □

В центральном поле, в отличие от собственного, допускается пересечение всех линий семейства, но только в одной точке.

ПРИМЕР 13.3. В той же самой области D: $x^2+y^2\le 1$ семейство линий $y=Cx$, дополненное вертикальной прямой $x=0$, образует центральное поле (рис. 13.3). Линии этого семейства полностью заполняют область D и пересекаются все в единственной точке $O(0,0)$. Функция наклона этого поля в каждой точке, кроме точек с нулевой абсциссой, равна: $p(x,y)=y/x$. На оси ординат функция наклона поля не определена. □

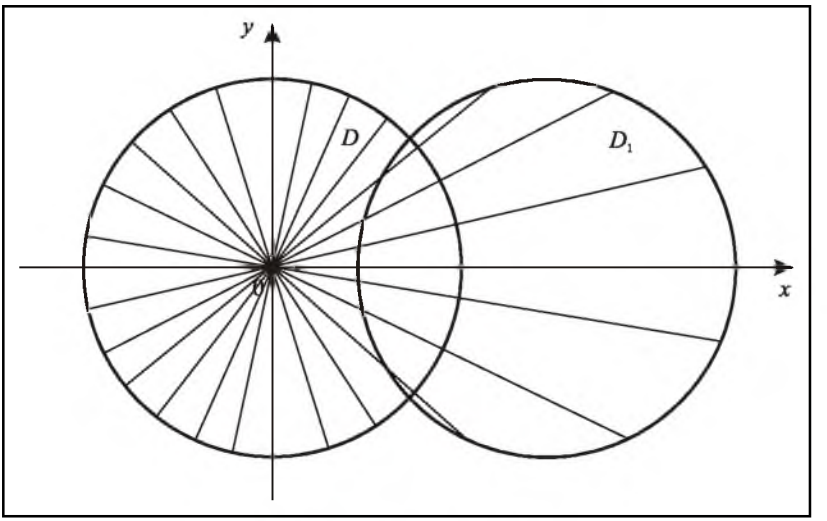


Рис. 13.3. Центральное поле в примере 13.3

Заметим, что, если рассмотреть это же семейство в области D_1: $(x-1{,}5)^2+y^2\le 1$, показанной на рис. 13.3, то в этой области рассматриваемое семейство кривых образует уже собственное поле, т. к. центр семейства теперь не попадает в область D_1.

13.2. Поле экстремалей

Перейдем теперь от любого поля к полю экстремалей. Рассмотрим элементарную задачу вариационного исчисления (2.1—2.2). Нам известно, что решение дифференциального уравнения Эйлера (2.9) — это двухпараметрическое семейство кривых, т. к. в него входят 2 произвольные постоянные: $y=y(x, C_1, C_2)$. Пусть мы одну из них каким-то образом зафиксировали, например, удовлетворили одному из граничных условий (2.2). Тогда у нас останется однопараметрическое семейство кривых $y=y(x, C)$, и каждая из кривых этого семейства является экстремалью, т. е. удовлетворяет дифференциальному уравнению Эйлера (2.9). Это семейство может образовывать (или не образовывать) поле (собственное или центральное) в некоторой области D.

Определение 13.4. *Полем экстремалей* задачи (2.1—2.2) называется собственное или центральное поле $y=y(x, C)$, такое, что:

- $\forall C$ кривая $y=y(x, C)$ является экстремалью, т. е. решением дифференциального уравнения Эйлера (2.9);
- $\forall$ граничных условий (2.2) $\exists! C_0$, такое, что кривая $y=y(x, C_0)$ удовлетворяет (2.2);
- кривая $y=y(x, C_0)$ не лежит на границе области D, в которой семейство $y=y(x, C)$ образует поле.

В этом случае говорят, что *экстремаль* (решение задачи (2.1—2.2)) *включена в поле экстремалей.* □

На рис. 13.4 показаны собственное поле экстремалей (*а*) и центральное поле (*б*).

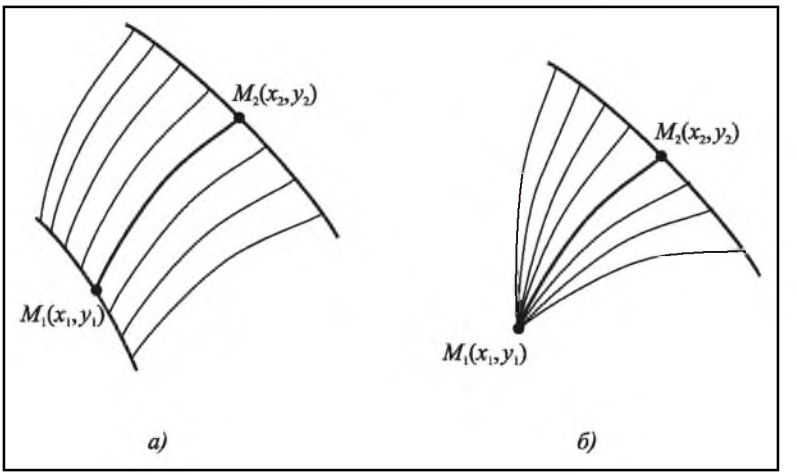


Рис. 13.4. Поле экстремалей: *а*) собственное; *б*) центральное

ПРИМЕР 13.4. Построить поле экстремалей для вариационной задачи:

$$J(y)=\int_0^a \left(y'^2-y^2\right)dx; \quad y(0)=0; \quad y(a)=b, \tag{13.2}$$

где $a>0$.

Дифференциальное уравнение Эйлера имеет вид:

$$y''+y=0. \tag{13.3}$$

Его общее решение:

$$y=C_1\cos x+C_2\sin x. \tag{13.4}$$

Удовлетворив граничному условию на левом конце, получим

$$y(0)=C_1=0; \;\Rightarrow\; y(x)=C_2\sin x. \tag{13.5}$$

Имеем однопараметрическое семейство кривых, показанное на рис. 13.5.

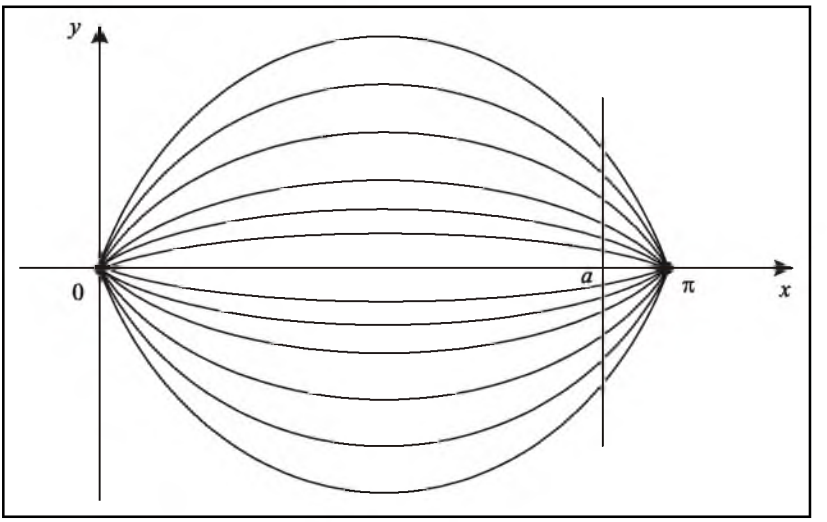


Рис. 13.5. Решение примера 13.4

При $a<\pi$ это семейство образует центральное поле в области $x\in[0,a]$ (бесконечная вертикальная полоса) с центром в точке $O(0,0)$: через любую точку $M_2(a,b)$ проходит только одна экстремаль, и любая экстремаль — внутренняя линия области. Если же $a\geq\pi$, то семейство (13.5) поля не образует: появляется второй узел в точке $(\pi,0)$. □

13.3. Функция Вейерштрасса

Выведем достаточные условия экстремума для элементарной задачи вариационного исчисления (2.1—2.2). Пусть найдена экстремаль L, уравнение которой $y=y(x)$, и она включена в некоторое собственное или центральное поле экстремалей с функцией наклона $p(x,y)$. Мы будем рассматривать только такие экстремали. На практике чаще всего такое поле можно построить. Какие же условия должны выполняться, чтобы мы могли точно сказать: на кривой C достигается, например, минимум? Очевидно, минимум будет достигаться (см. определение минимума в *главе 1*), если на близких допустимых кривых $L_1 \neq L$ приращение функционала будет строго положительным:

$$\Delta J(y) = \int_{L_1} F(x,y,y')dx - \int_{L} F(x,y,y')dx > 0, \tag{13.6}$$

и порядок минимума (сильный или слабый) определяется классом, в котором кривые L и L_1 близкие. На рис. 13.6 жирной линией показана экстремаль L, включенная в центральное поле экстремалей (тонкие сплошные линии), и допустимые кривые (штриховые линии).

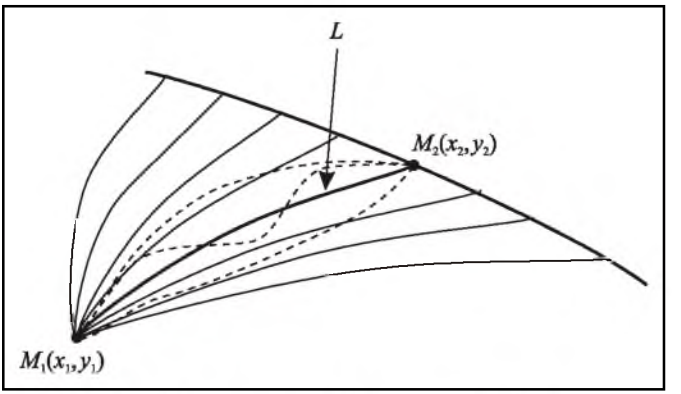


Рис. 13.6. Экстремаль L и близкие функции

Преобразуем формулу (13.6). Пока что она неудобна для дальнейших выкладок: в ней есть разность интегралов от одной и той же функции, но по разным путям интегрирования. Нам было бы удобнее иметь разность интегралов от разных функций, но по одному и тому же пути. Попытаемся привести (13.6) к такому виду. Для этого рассмотрим вспомогательный функционал

$$V(y) = \int_{L_1} \left(F(x,y,p) + (y'-p)F_p(x,y,p) \right) dx, \tag{13.7}$$

где $p(x,y)$ — функция наклона поля. Этот функционал обладает очень интересными свойствами.

Во-первых, если линия интегрирования L_1 совпадает с экстремалью L, то он обращается в функционал (2.1). Действительно, на линии L функция наклона поля $p(x,y)$ совпадает с углом наклона касательной к экстремали y', и

$$V(y)\Big|_L = \int_L F(x,y,y')\,dx = J(y). \tag{13.8}$$

А во-вторых, функционал (13.7) вообще не зависит от линии интегрирования. Для доказательства этого факта перепишем (13.7) в виде криволинейного интеграла 2-го рода:

$$\begin{aligned} V(y) &= \int_{L_1}\left(F(x,y,p) - pF_p(x,y,p) + \frac{dy}{dx}F_p(x,y,p)\right)dx = \\ &= \int_{L_1}\Big(F(x,y,p) - pF_p(x,y,p)\Big)\,dx + F_p(x,y,p)\,dy. \end{aligned} \tag{13.9}$$

Теперь обозначим

$$\begin{cases} P(x,y) = F(x,y,p) - pF_p(x,y,p); \\ Q(x,y) = F_p(x,y,p); \end{cases} \tag{13.10}$$

и проверим выполнение необходимого и достаточного условия независимости криволинейного интеграла 2-го рода от линии интегрирования:

$$\begin{aligned} &\frac{\partial Q}{\partial x} = F_{xp} + F_{pp}p_x \\ &\frac{\partial P}{\partial y} = F_y + F_p p_y - p_y F_p - p\Big(F_{yp} + F_{pp}p_y\Big) = F_y - pF_{yp} - pp_y F_{pp}; \\ &\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} = F_{xp} + F_{pp}p_x - F_y + pF_{yp} + pp_y F_{pp} = \\ &= \Big(F_{xp} + pF_{yp} + F_{pp}\big(p_x + pp_y\big)\Big) - F_y. \end{aligned} \tag{13.11}$$

Любая точка кривой L_1 лежит на какой-либо экстремали, и там $p(x,y) = y'$, поэтому

$$\begin{aligned} &\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} = \left(F_{xp} + F_{yp} + F_{pp}\left(p_x + p_y\frac{dy}{dx}\right)\right) - F_y = \\ &= \left(F_{xp} + pF_{yp} + F_{pp}\frac{dp}{dx}\right) - F_y = \frac{dF_p}{dx} - F_y = \frac{dF_{y'}}{dx} - F_y = 0, \end{aligned} \tag{13.12}$$

т. к. в любой точке любой экстремали поля удовлетворяется дифференциальное уравнение Эйлера (2.9).

Что же нам дает этот вспомогательный функционал (13.7)? Посмотрите: с его помощью мы теперь можем легко преобразовать приращение функционала (13.6). В (13.6) — разность двух функционалов по разным путям, но от одного и того же выражения. Мы же его преобразуем в разность интегралов по одному и тому же пути, но от разных выражений. Для этого вместо 2-го слагаемого в (13.6) подставим равный ему функционал (13.7):

$$\Delta J(y) = \int_{L_1} F(x,y,y')dx - \int_{L_1} \Big(F(x,y,p) + (y'-p)F_p(x,y,p)\Big)dx = \\ = \int_{L_1} \Big(F(x,y,y') - F(x,y,p) - (y'-p)F_p(x,y,p)\Big)dx. \tag{13.13}$$

Определение 13.5. Подынтегральная функция в выражении (13.13) называется *функцией Вейерштрасса* (Karl Theodor Wilhelm Weierstrass, 1815—1897, рис. 13.7) и обозначается:

$$E(x,y,p,y') = F(x,y,y') - (y'-p)F_p(x,y,p). \ \square \tag{13.14}$$

Через функцию Вейерштрасса $E(x,y,p,y')$ выражается приращение функционала:

$$\Delta J(y) = \int_{x_1}^{x_2} E(x,y,p,y')\,dx. \tag{13.15}$$

Теперь, используя функцию Вейерштрасса, мы можем перейти к выводу достаточных условий экстремума.

Рис. 13.7. К. Вейерштрасс

13.4. Достаточные условия Вейерштрасса экстремума функционала

Будем считать, что $x_2 > x_1$. Тогда, если на экстремали $y(x)$ достигается минимум, то на близких кривых приращение функционала (13.15) будет строго больше нуля, а значит, и функция Вейерштрасса $E(x, y, p, y')$ должна быть положительной (для максимума — отрицательной). Если речь идет о нестрогом экстремуме, то соответствующие неравенства могут быть нестрогими. В зависимости от класса кривых, на котором выполняются данные условия, мы можем говорить о сильном или слабом экстремуме. Так, если близкие кривые — такие, в которых значения функций отличаются мало, а значения производных могут отличаться сильно, то мы говорим о сильном экстремуме. Если же класс функций — более узкий: и значения функции, и значения производной отличаются мало, то мы можем говорить лишь о слабом экстремуме.

Таким образом, мы доказали следующие условия.

Достаточные условия Вейерштрасса (13.1) сильного минимума. Функция $y(x)$ доставляет сильный минимум функционалу $J(y)$, если:

- она является решением дифференциального уравнения Эйлера (2.9) при граничных условиях (2.2), т. е. если выполняются необходимые условия экстремума;
- ее можно включить в поле экстремалей;
- $\forall (x, y)$, близких к экстремали, и $\forall y'$ (не обязательно близких к $p(x, y)$) функция Вейерштрасса $E(x, y, p, y') \geq 0$ (или > 0 для строгого сильного минимума). □

Достаточные условия Вейерштрасса (13.2) слабого минимума. Функция $y(x)$ доставляет слабый минимум функционалу $J(y)$, если:

- то же самое: она является решением дифференциального уравнения Эйлера (2.9) при граничных условиях (2.2), т. е. если выполняются необходимые условия экстремума;
- то же самое: ее можно включить в поле экстремалей;
- $\forall (x, y)$, близких к экстремали, и $\forall y'$, близких к $p(x, y)$, функция Вейерштрасса $E(x, y, p, y') \geq 0$ (или > 0 для строгого слабого минимума). □

В достаточных условиях максимума нужно знак в третьем требовании заменить на противоположный.

ПРИМЕР 13.5. Найти экстремали заданного функционала и исследовать их на выполнение достаточных условий Вейерштрасса:

$$J(y)=\int_0^a y'^3\,dx;\quad \begin{cases} y(0)=0; & a>0; \\ y(a)=b; & b>0. \end{cases} \tag{13.16}$$

Подынтегральная функция зависит только от y', поэтому решениями являются прямые:

$$y=C_1x+C_2. \tag{13.17}$$

Имеем двухпараметрическое семейство линий. Подставляем вначале граничное условие на левом конце — пытаемся построить поле экстремалей:

$$y(0)=C_2=0;\ \Rightarrow\ y(x)=C_1x. \tag{13.18}$$

Получили центральное поле экстремалей. Константу C_1 находим из граничного условия на правом конце:

$$C_1a=b;\quad C_1=\frac{b}{a}>0;\quad y(x)=\frac{bx}{a}. \tag{13.19}$$

Таким образом, 1-е и 2-е условия выполнены: получена линия, удовлетворяющая дифференциальному уравнению Эйлера, граничным условиям и включенная в центральное поле экстремалей. Это поле и включенная в него экстремаль показаны на рис. 13.8, функция наклона поля $p(x,y)=y/x$.

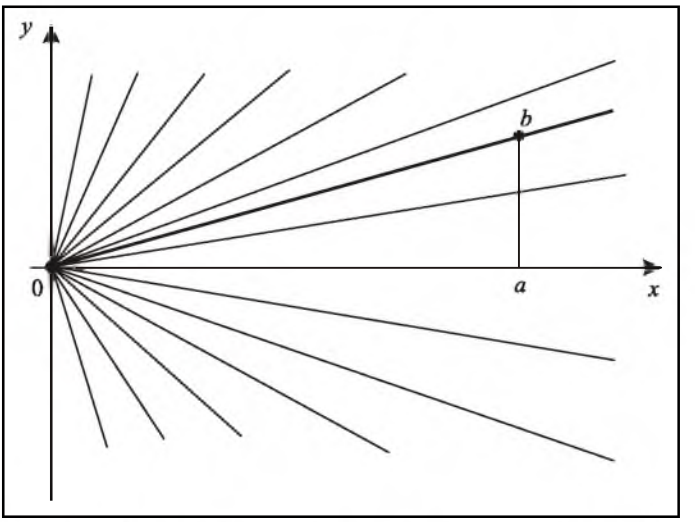


Рис. 13.8. Центральное поле экстремалей в примерах 13.5 и 13.6

Осталось выяснить знак функции Вейерштрасса $E(x,y,p,y')$. Вычисляем по формуле (13.14):

$$
\begin{aligned}
&F\left(x,y,y'\right)=y'^3;\\
&F\left(x,y,p\right)=p^3;\\
&F_p\left(x,y,p\right)=3p^2;\\
&E\left(x,y,p,y'\right)=y'^3-p^3-\left(y'-p\right)3p^2=\left(y'-p\right)\left(y'^2+y'p-2p^2\right)=\\
&=\left(y'-p\right)\left(y'-p\right)\left(y'+2p\right)=\left(y'-p\right)^2\left(y'+2p\right).
\end{aligned}
\qquad (13.20)
$$

Как мы видим, функция Вейерштрасса $E(x,y,p,y')$ вообще не зависит от x, y, а зависит только от аргументов y' и p. Первый сомножитель — неотрицательный. Выясним знак 2-го сомножителя на кривых, близких к экстремали (13.19). На этой экстремали функция поля $p(x,y)=b/a$. Если y' близкая к b/a: $y'=b/a+\varepsilon$, где ε — малая величина, то

$$
y'+2p=\frac{b}{a}+\varepsilon+2\frac{b}{a}=3\frac{b}{a}+\varepsilon>0. \qquad (13.21)
$$

Функция Вейерштрасса $E(x,y,p,y')\geq 0$ на кривых, достаточно близких к экстремали по значениям производной, следовательно, слабый минимум точно достигается. Проверим, достигается ли сильный минимум. Если взять y' не близкой к b/a, то слагаемое $(y'+2p)$ может уже изменить свой знак. Так, если, например, взять $y'=-3b/a$, то $y'+2p=-b/a<0$. Поэтому сильный минимум здесь не достигается.

Ответ. Экстремаль (13.19) доставляет функционалу (13.16) только слабый минимум, но не сильный. □

ПРИМЕР 13.6. Найти экстремали функционала и исследовать их на выполнение достаточных условий Вейерштрасса:

$$
J\left(y\right)=\int_0^a\left(6y'^2-y'^4+yy'\right)dx;\quad
\begin{cases}
y(0)=0; & a>0;\\
y(a)=b; & b>0.
\end{cases}
\qquad (13.22)
$$

Здесь подынтегральная функция не зависит явно от x — записываем первый интеграл уравнения Эйлера (2.64):

$$
\begin{aligned}
&6y'^2-y'^4+yy'-y'\left(12y'-4y'^3+y\right)=C_1;\\
&3y'^4-6y'^2-C_1=0.
\end{aligned}
\qquad (13.23)
$$

Получили биквадратное уравнение относительно y'. Его решения — это какие-то числа (константы), зависящие от C_1. Переобозначив решения (обозначив их снова C_1), найдем экстремали, которые будут прямыми:

$$
y'=C_1;\quad\Rightarrow\quad y(x)=C_1x+C_2. \qquad (13.24)
$$

Решение совпадает с решением предыдущего примера, и граничные условия — те же. Поэтому экстремалью будет прямая (13.19), и она также включается в поле экстремалей (13.19), показанное на рис. 13.8. Вычислим функцию Вейерштрасса (13.14) с помощью MATLAB.

```
clear all % очистили память
syms y Dy p % описали символические переменные
F=6*Dy^2-Dy^4+y*Dy % задали F(x,y,y')
Fxyp=subs(F,Dy,p) % вычислили F(x,y,p)
Fp=diff(Fxyp,p) % вычислили dF(x,y,p)/dp
E=simple(F-Fxyp-(Dy-p)*Fp) % вычислили E(x,y,p,y')
F =
6*Dy^2-Dy^4+y*Dy
Fxyp =
6*p^2-p^4+y*p
Fp =
12*p-4*p^3+y
E =
-(3*p^2-6+2*Dy*p+Dy^2)*(-Dy+p)^2
```

Очевидно знак $E(x, y, p, y')$ определяется знаком 1-го сомножителя. Обозначим его E_1 и запишем в виде:

$$E_1 = -y'^2 - 2py' + (6 - 3p^2). \tag{13.25}$$

Нам предстоит выяснить: при каких y' (и насколько близких к b/a) эта величина $E_1 \geq 0$? Будем рассматривать (13.25) как квадратный трехчлен относительно y'. Он обращается в 0 при выполнении условия:

$$y' = -p \pm \sqrt{6 - 2p^2}\ . \tag{13.26}$$

Если p таково, что в (13.26) будет 2 действительных различных корня, то это означает, что при разных y' величина E_1 будет иметь разный знак, и сильного экстремума быть уже не может. Если же при некоторых значениях p в выражении (13.26) будут комплексные корни, то знак E_1 будет постоянный при любых y', и, как нетрудно заметить, этот знак отрицательный. Поэтому при тех p, при которых оба корня (13.26) — комплексные, будет сильный максимум. Комплексные корни (13.26) будут при таких значениях p:

$$6 - 2p^2 < 0\,; \quad p^2 > 3\,; \quad p > \sqrt{3}\ . \tag{13.27}$$

При кратном корне также будет $E_1 = -p < 0$, а значит, сильный максимум.

Исследуем теперь знак E_1 при действительных различных корнях (13.26), т. е. при $p^2 < 3$. Мы уже знаем, что в этом случае при значительном изменении y' знак E_1 не сохраняется, т. е. сильного экстремума нет. А как насчет слабого

экстремума? Может быть, знак E_1 сохраняется при малом отклонении y' от $p = b/a$? Проверим: положим $y' = p + \varepsilon$, где ε — малое. Тогда

$$\begin{aligned} E_1 &= -(p+\varepsilon)^2 - 2p(p+\varepsilon) + 6 - 3p^2 = \\ &= -p^2 - 2p\varepsilon - \varepsilon^2 - 2p^2 - 2p\varepsilon + 6 - 3p^2 = 6 - 6p^2 - 4p\varepsilon - \varepsilon^2 = \\ &= (\text{с точностью до малых высшего порядка}) = 6 - 6p^2 - 4p\varepsilon. \end{aligned} \qquad (13.28)$$

При малых ε эта величина будет положительной при $6 - 6p^2 > 0$, т. е. при $p < 1$ (p у нас положительное). Значит, при $p < 1$ достигается слабый минимум. Если же $p > 1$, то при малых ε знак E_1 будет отрицательный — достигается слабый максимум. Области различных экстремумов для этого примера показаны на рис. 13.9. □

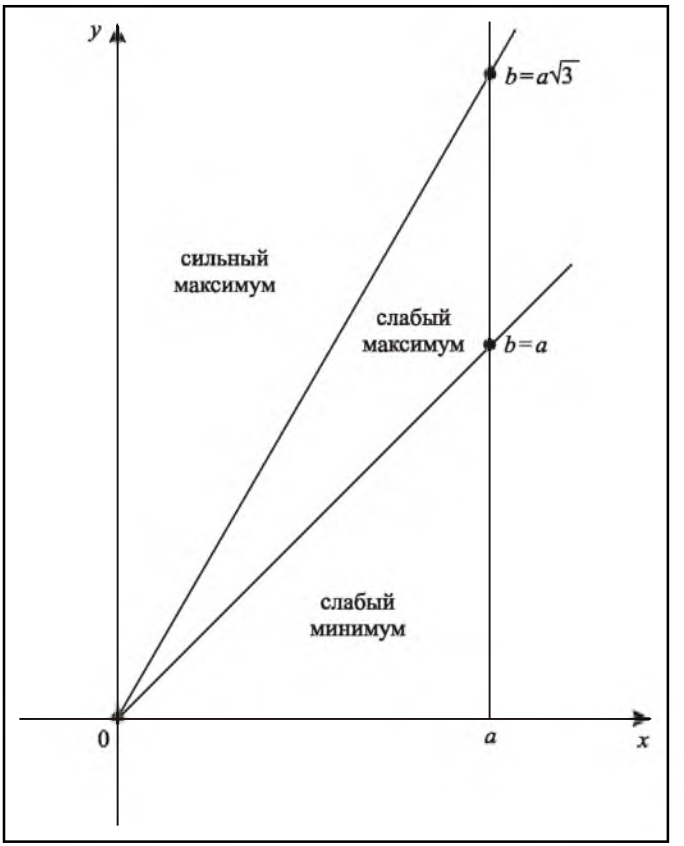


Рис. 13.9. Различные экстремумы в примере 13.6

13.5. Достаточные условия Лежандра экстремума функционала

Рассмотрим еще одно достаточное условие, более простое, чем условие Вейерштрасса. Вспомним, какие достаточные условия экстремума мы имели для функции одной переменной $y(x)$.

Во-первых, у нас было самое общее условие: приращение функции в окрестности точки экстремума должно сохранять свой знак. Для функционалов аналогичное условие — это достаточное условие Вейерштрасса.

Рис. 13.10. А. М. Лежандр

А во-вторых, для функции одной переменной мы имели еще одно, более простое достаточное условие: если функция — дважды дифференцируемая, то, если в стационарной точке $y''>0$, то в этой точке — минимум, а если $y''<0$ — то максимум. Аналогом этого условия для функционалов и будет *достаточное условие Лежандра* (Adrien Marie Legendre, 1752—1833, рис. 13.10). Для его вывода разложим подынтегральную функцию нашей задачи (2.1) $F(x,y,y')$ по формуле Тейлора по 3-му аргументу y' в окрестности значения $y'=p$. При этом ограничимся только формулой 1-го порядка:

$$F(x,y,y') = F(x,y,p) + \frac{F_p(x,y,p)}{1!}(y'-p) + \frac{F_{pp}(x,y,q)}{2!}(y'-p)^2. \qquad (13.29)$$

Здесь q — средняя "точка" между y' и p, а последнее слагаемое — остаточный член в форме Лагранжа. Если сравнить это выражение с функцией Вейер-

штрасса (13.14), то мы видим, что функция Вейерштрасса равна фактически этому остаточному члену:

$$E(x,y,p,y') = F(x,y,y') - F(x,y,p) - (y'-p)F_p(x,y,p) = \\ = \frac{(y'-p)^2}{2} F_{pp}(x,y,q)\,. \tag{13.30}$$

На функциях, близких к экстремали, первый множитель всегда положителен, поэтому знак функции Вейерштрасса определяется знаком выражения $F_{pp}(x,y,q)$. Если вблизи экстремали эта функция существует и непрерывна, мы можем исследовать знак выражения $F_{y'y'}$. Если $F_{y'y'}>0$ на всех функциях, близких к экстремали в смысле близости 0-го порядка (т. е. мало отличающихся от y по ординатам и как угодно отличающихся по производной), то на нашей экстремали функция Вейерштрасса положительна, а значит, достигается сильный минимум. Если же $F_{y'y'}>0$ на всех функциях, близких к экстремали в смысле близости 1-го порядка (т. е. мало отличающихся от y по ординатам и мало отличающихся по производной), то на экстремали достигается только слабый минимум. Аналогично формулируется достаточное условие максимума (сильного или слабого). Именно в этом (проверке знака $F_{y'y'}$) и состоит достаточное условие Лежандра.

ПРИМЕР 13.7. Задача о брахистохроне. Проверим подынтегральную функцию функционала (2.74):

$$F = \frac{1}{\sqrt{2g}} \frac{\sqrt{1+y'^2}}{\sqrt{y}}; \\ F_{y'} = \frac{1}{\sqrt{2gy}} \frac{y'}{\sqrt{1+y'^2}}; \tag{13.31} \\ F_{y'y'} = \frac{1}{\sqrt{2gy}} \frac{\sqrt{1+y'^2} - y' \dfrac{y'}{\sqrt{1+y'^2}}}{1+y'^2} = \frac{1}{\sqrt{2gy}} \frac{1}{(1+y'^2)^{3/2}}.$$

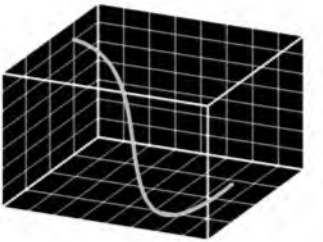
Выражение $F_{y'y'}>0$ для всех y', не обязательно близких к p, следовательно, в данной задаче достигается сильный минимум. □

13.6. Вопросы для самопроверки

1. Чем вообще необходимые условия экстремума отличаются от достаточных?

2. Что такое однопараметрическое семейство кривых? Как его можно получить из общего решения дифференциального уравнения Эйлера?
3. Что такое собственное поле? Центральное поле? Чем они отличаются?
4. Как получается поле экстремалей?
5. Для чего нам понадобилось вводить вспомогательный функционал $V(y)$ при выводе $\Delta J(y)$?
6. Что такое функция Вейерштрасса? Зачем мы ее вводим?
7. Как формулируется достаточный признак Вейерштрасса?
8. Как формулируется достаточный признак Лежандра? Как он выводится?
9. Выведите достаточные условия экстремума для функционалов, рассмотренных в *главах 3—5*.

ГЛАВА 14



Условный экстремум функционалов

14.1. Вариационная задача для функционала с голономными ограничениями

Рассмотрим функционал вида (3.1), зависящий от n функций, с граничными условиями (3.2), при дополнительном условии, что эти n функций связаны между собой m ограничениями-равенствами ($m < n$):

$$\begin{cases} \varphi_j\left(y_1, y_2, \ldots, y_n\right) = 0\ ; \\ j = \overline{1,m}. \end{cases} \tag{14.1}$$

Будем предполагать, что условия (14.1) независимы между собой и не противоречат граничным условиям (3.2).

Связи вида (14.1) называются *голономными* (конечными, недифференциальными). Бывают еще и *неголономные* (дифференциальные) связи, в которые входят не только сами функции y_i, но и их производные. О них мы поговорим дальше.

Как же можно решить задачу (3.1, 3.2, 14.1)? Простейший способ — это попытаться решить нелинейную систему (14.1) относительно каких-либо m переменных (выразить их через остальные $n-m$ функций), и исключить их затем из функционала (3.1). Тем самым мы упростим вариационную задачу: вместо функционала, зависящего от n функций, получим функционал, зависящий только лишь от $(n-m)$ функций.

Но, как и для функции n переменных при ограничениях-равенствах, здесь удобнее применить *метод неопределенных множителей Лагранжа* (Joseph Louis Lagrange, 1736—1813, рис. 14.1). Рассмотрим его вывод. Во многом он будет похож на вывод этого метода для функции n переменных.

Рис. 14.1. Ж. Л. Лагранж

Итак, как обычно, исходим из необходимого условия экстремума функционала: его вариация на экстремали равна нулю. Для функционала (3.1) его вариация вызывается вариациями всех функций δy_i и их производных $\delta y_i'$. В соответствии с 1-м способом вычисляем вариацию путем разложения подынтегральной функции (3.1) в ряд Тейлора с удержанием только линейных членов:

$$\delta J(y_1, y_2, \ldots, y_n) = \int_{x_1}^{x_2} \left(F_{y_1} \delta y_1 + \ldots + F_{y_n} \delta y_n + F_{y_1'} \delta y_1' + \ldots + F_{y_n'} \delta y_n' \right) dx = 0 . \quad (14.2)$$

Далее интегрируем по частям, как мы делали в (2.8). Внеинтегральные слагаемые обращаются в нуль в силу граничных условий (3.2), и остается:

$$\delta J(y_1, y_2, \ldots, y_n) = \int_{x_1}^{x_2} \sum_{i=1}^{n} \left(F_{y_i} - \frac{dF_{y_i'}}{dx} \right) \delta y_i \, dx = 0 . \quad (14.3)$$

Если бы все δy_i были бы независимыми, можно было бы приравнять нулю каждую скобочку, и мы бы получили систему дифференциальных уравнений Эйлера (3.4) (в *главе 3* мы вывели ее из других соображений). Но здесь мы не можем этого сделать: вариации функций δy_i *не являются независимыми*: они связаны продифференцированными соотношениями (14.1):

$$\begin{cases} \sum_{i=1}^{n} \dfrac{\partial \varphi_j}{\partial y_i} \delta y_i = 0 ; \\ j = \overline{1, m} . \end{cases} \quad (14.4)$$

Здесь мы опять можем пойти по пути исключения зависимых переменных: можно решить систему (14.4) относительно зависимых вариаций. Эта система — линейная, и решается легко (мы предполагаем, что уравнения (14.1) независимые, поэтому ранг матрицы коэффициентов системы (14.4) равен m). Далее подставляем полученные m вариаций, выраженные через остальные $n-m$, в (14.3), и приравниваем нулю коэффициенты при независимых $n-m$ вариациях δy_i. Но мы снова поступим по-другому: умножим каждое из уравнений (14.4) на произвольную функцию $\lambda_j(x)$ (равенство нулю сохранится), проинтегрируем по интервалу $[x_1, x_2]$ (опять равенство нулю сохранится), сложим их между собой и прибавим к (14.3) (снова получим равенство нулю). Вот что будет после перегруппировки слагаемых:

$$\int_{x_1}^{x_2} \sum_{i=1}^{n} \left(F_{y_i} - \frac{dF_{y_i'}}{dx} + \sum_{j=1}^{m} \lambda_j(x) \frac{\partial \varphi_j}{\partial y_i} \right) \delta y_i \, dx = 0 . \tag{14.5}$$

Здесь из n вариаций δy_i независимыми будут только $n-m$ штук, а остальные m — зависимые. Какие из них считать независимыми, а какие — зависимыми, не имеет значения: все равно мы получим, как сейчас увидим, одинаковый результат. Пусть мы определились в этом вопросе: выбрали какие-то $n-m$ вариаций в качестве независимых. За счет выбора m функций $\lambda_j(x)$ мы всегда можем добиться того, чтобы скобочки при m зависимых δy_i в выражении (14.5) обратились в нуль: нужно просто решить соответствующую систему линейных уравнений, а мы предположили, что она решается. После этой процедуры в (14.5) остаются только независимые δy_i, а значит, множители при них (те же скобочки) также обращаются в нуль. Таким образом, имеем систему n модифицированных уравнений Эйлера:

$$\begin{cases} F_{y_i} - \dfrac{dF_{y_i'}}{dx} + \displaystyle\sum_{j=1}^{m} \lambda_j(x) \frac{\partial \varphi_j}{\partial y_i} = 0 ; \\ i = \overline{1,n} ; \end{cases} \tag{14.6}$$

дополненную m уравнениями связи (14.1). Этих $n+m$ уравнений достаточно для нахождения n искомых функций $y_i(x)$ и m вспомогательных функций $\lambda_j(x)$, которые играют роль неопределенных множителей Лагранжа в нашей задаче.

Конечно же, как и в классическом методе неопределенных множителей Лагранжа, мы введем в рассмотрение функцию Лагранжа:

$$\begin{aligned} &L(x, y_1, y_2, \ldots, y_n, y_1', y_2', \ldots, y_n', \lambda_1, \lambda_2, \ldots, \lambda_m) = \\ &= F(x, y_1, y_2, \ldots, y_n, y_1', y_2', \ldots, y_n') + \sum_{j=1}^{m} \lambda_j(x) \varphi_j(y_1, y_2, \ldots, y_n). \end{aligned} \tag{14.7}$$

Если мы построим дифференциальные уравнения Эйлера для функции L, то увидим, что уравнения для y_i совпадают с (14.6), а уравнения для $\lambda_j(x)$ — с (14.1):

$$\begin{aligned} L_{y_i} - \frac{dL_{y_i'}}{dx} &= F_{y_i} - \frac{dF_{y_i'}}{dx} + \sum_{j=1}^{m} \lambda_j(x) \frac{\partial \varphi_j}{\partial y_i} = 0; \\ L_{\lambda_j} - \frac{dL_{\lambda_j'}}{dx} &= \varphi_j(y_1, y_2, \ldots, y_n) = 0. \end{aligned} \tag{14.8}$$

Значит, задача фактически свелась к исследованию на обычный безусловный экстремум функционала с подынтегральной функцией (14.7):

$$\begin{aligned} &J^*(y_1, y_2, \ldots, y_n, y_1', y_2', \ldots, y_n', \lambda_1, \lambda_2, \ldots, \lambda_m) = \\ &= \int_{x_1}^{x_2} \left(F(x, y_1, y_2, \ldots, y_n, y_1', y_2', \ldots, y_n') + \sum_{j=1}^{m} \lambda_j(x) \varphi_j(y_1, y_2, \ldots, y_n) \right) dx. \end{aligned} \tag{14.9}$$

Этот функционал зависит от $n+m$ функций, но дополняется граничными условиями (4.2) только для n функций $y_i(x)$. Для функций Лагранжа $\lambda_j(x)$ граничные условия не нужны, т. к. уравнения Эйлера для них получаются алгебраическими (14.1).

ПРИМЕР 14.1. Найти кратчайшее расстояние между точками $A(R, 0, 0)$ и $B(0, R, H)$ на цилиндрической поверхности $x^2 + y^2 = R^2$ (рис. 14.2).

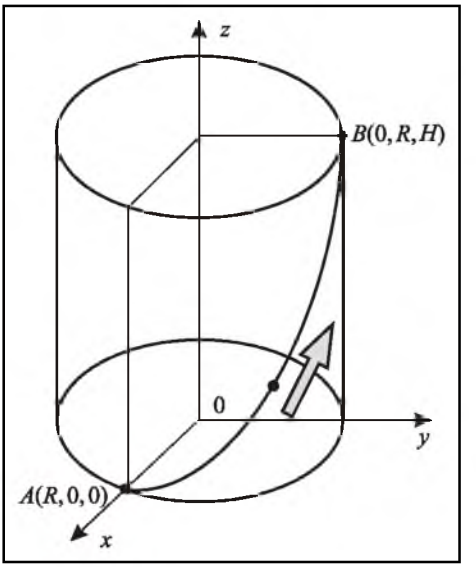


Рис. 14.2. Пример 14.1: муха ползет из точки A в точку B по кратчайшему пути

Здесь удобнее в качестве аргумента выбрать z, а искомые функции обозначить $x(z)$ и $y(z)$. Тогда не возникает многозначности в этих функциях, и задача формулируется следующим образом:

$$\begin{gathered} l(x,y)=\int_0^H \sqrt{1+x'^2+y'^2}\,dz \to \min; \\ \begin{cases} x(0)=R; \\ y(0)=0; \end{cases} \quad \begin{cases} x(H)=0; \\ y(H)=R; \end{cases} \\ \varphi(x,y)=x^2+y^2-R^2=0. \end{gathered} \tag{14.10}$$

Составляем функцию Лагранжа:

$$L(x,y,\lambda)=\sqrt{1+x'^2+y'^2}+\lambda(z)\left(x^2+y^2-R^2\right). \tag{14.11}$$

Система дифференциальных уравнений Эйлера имеет вид:

$$\begin{cases} L_x-\dfrac{dL_{x'}}{dz}=2\lambda x-\dfrac{d}{dz}\left(\dfrac{x'}{\sqrt{1+x'^2+y'^2}}\right)=0: \\ L_y-\dfrac{dL_{y'}}{dz}=2\lambda y-\dfrac{d}{dz}\left(\dfrac{y'}{\sqrt{1+x'^2+y'^2}}\right)=0; \\ L_\lambda-\dfrac{dL_{\lambda'}}{dz}=x^2+y^2-R^2=0. \end{cases} \tag{14.12}$$

Вычислим отдельно полную производную в 1-м уравнении (во 2-м выражение будет симметричным):

$$\begin{gathered} \frac{d}{dz}\left(\frac{x'}{\sqrt{1+x'^2+y'^2}}\right)=\frac{x''\sqrt{1+x'^2+y'^2}-x'\dfrac{x'x''+y'y''}{\sqrt{1+x'^2+y'^2}}}{1+x'^2+y'^2}= \\ =\frac{x''\left(1+x'^2+y'^2\right)-x'\left(x'x''+y'y''\right)}{\left(1+x'^2+y'^2\right)^{3/2}}=\frac{x''+x''y'^2-x'y'y''}{\left(1+x'^2+y'^2\right)^{3/2}}. \end{gathered} \tag{14.13}$$

Подставим (14.13) и симметричное ему выражение в (14.12):

$$\begin{cases} 2\lambda x - \dfrac{x''\left(1+y'^2\right) - x'y'y''}{\left(1+x'^2+y'^2\right)^{3/2}} = 0; \\ 2\lambda y - \dfrac{y''\left(1+x'^2\right) - x'y'x''}{\left(1+x'^2+y'^2\right)^{3/2}} = 0; \\ x^2 + y^2 - R^2 = 0. \end{cases} \tag{14.14}$$

Теперь нам предстоит решить полученную систему нелинейных дифференциальных уравнений. Будем стараться сохранить симметрию при ее решении. Исключим из 1-го и 2-го уравнений λ:

$$\begin{cases} \dfrac{x''\left(1+y'^2\right) - x'y'y''}{2x\left(1+x'^2+y'^2\right)^{3/2}} = \dfrac{y''\left(1+x'^2\right) - x'y'x''}{2y\left(1+x'^2+y'^2\right)^{3/2}}; \\ x^2 + y^2 - R^2 = 0. \end{cases} \tag{14.15}$$

Введем параметр $t(z)$ вдоль оси z следующим образом:

$$\begin{cases} x = R\cos t; \\ y = R\sin t. \end{cases} \tag{14.16}$$

Это можно сделать, т. к. у нас есть 2-е уравнение (14.15). Теперь оно удовлетворяется, а в 1-м уравнении перейдем к параметру t. Дифференцируя (14.16), найдем:

$$\begin{cases} x' = -R\sin t \cdot t'; \\ y' = R\cos t \cdot t'; \end{cases} \quad \begin{cases} x'' = -R\cos t \cdot t'^2 - R\sin t \cdot t''; \\ y'' = -R\sin t \cdot t'^2 + R\cos t \cdot t''. \end{cases} \tag{14.17}$$

Подставим полученные выражения в 1-е уравнение (14.15). Числитель левой части:

$$\begin{aligned} &x''\left(1+y'^2\right) - x'y'y'' = \left(-R\cos t \cdot t'^2 - R\sin t \cdot t''\right)\left(1 + R^2\cos^2 t \cdot t'^2\right) + \\ &+R^2 \sin t\cos t \cdot t'^2\left(-R\sin t \cdot t'^2 + R\cos t \cdot t''\right); \end{aligned} \tag{14.18}$$

числитель правой части:

$$y''\left(1+x'^2\right)-x'y'x''=\left(-R\sin t\cdot t'^2+R\cos t\cdot t''\right)\left(1+R^2\sin^2 t\cdot t'^2\right)+$$
$$+R^2\sin t\cos t\cdot t'^2\left(-R\cos t\cdot t'^2-R\sin t\cdot t''\right). \tag{14.19}$$

Решаем 1-е уравнение (14.15) как пропорцию, сокращая на одинаковые множители в знаменателе и числителе:

$$\left(-t'^2\cos t-t''\sin t\right)\left(1+R^2t'^2\cos^2 t\right)\sin t+$$
$$+Rt'^2\sin^2 t\cos t\left(-Rt'^2\sin t+Rt''\cos t\right)=$$
$$=\left(-t'^2\sin t+t''\cos t\right)\left(1+R^2t'^2\sin^2 t\right)\cos t+ \tag{14.20}$$
$$+Rt'^2\sin t\cos^2 t\left(-Rt'^2\cos t-Rt''\sin t\right).$$

Раскрываем скобки и упрощаем:

$$-t'^2\sin t\cos t-t''\sin^2 t-R^2t'^4\sin t\cos^3 t-R^2t'^2t''\sin^2 t\cos^2 t+$$
$$+R^2t'^2t''\sin^2 t\cos^2 t-R^2t'^4\sin^3 t\cos t=$$
$$=-t'^2\sin t\cos t+t''\cos^2 t-R^2t'^4\sin^3 t\cos t+R^2t''t'^2\sin^2 t\cos^2 t- \tag{14.21}$$
$$-R^2t'^4\sin t\cos^3 t-R^2t'^2t''\sin^2 t\cos^2 t\ .$$

Приводим подобные и переносим все в одну сторону:

$$t''=0. \tag{14.22}$$

Решение этого уравнения:

$$t(z)=C_1z+C_2. \tag{14.23}$$

Подставляем его в (14.16):

$$\begin{cases}x=R\cos\left(C_1z+C_2\right);\\ y=R\sin\left(C_1z+C_2\right).\end{cases} \tag{14.24}$$

Граничные условия при $z=0$ дают:

$$\begin{cases}x(0)=R\cos C_2=R;\\ y(0)=R\sin C_2=0;\end{cases} \tag{14.25}$$

что соответствует $C_2=0$. Теперь подставляем граничные условия при $z=H$:

$$\begin{cases}x(H)=R\cos C_1H=0;\\ y(H)=R\sin C_1H=R;\end{cases} \tag{14.26}$$

а это дает $C_1H=\pi/2$, т. е. $C_1=\pi/2H$. Окончательное решение вариационной задачи (14.10) — это винтовая линия:

$$\begin{cases} x(z)=R\cos\dfrac{\pi z}{2H}; \\ y(z)=R\sin\dfrac{\pi z}{2H}. \end{cases} \quad \square \tag{14.27}$$

14.2. Вариационная задача с неголономными ограничениями

Пусть теперь вместо голономных ограничений (14.1) имеются неголономные, т. е. такие, в левые части которых входят, кроме функций, и производные

$$\begin{cases} \varphi_j\left(y_1,y_2,\ldots,y_n,y_1',y_2',\ldots,y_n'\right)=0; \\ j=\overline{1,m}. \end{cases} \tag{14.28}$$

Что изменится в этом случае? Очевидно, (14.2) и (14.3) остаются в силе. Но при дифференцировании ограничений вместо (14.4) будем иметь

$$\begin{cases} \sum\limits_{i=1}^{n}\dfrac{\partial\varphi_j}{\partial y_i}\delta y_i+\sum\limits_{i=1}^{n}\dfrac{\partial\varphi_j}{\partial y_i'}\delta y_i'=0; \\ j=\overline{1,m}. \end{cases} \tag{14.29}$$

В (14.5) появится новое слагаемое:

$$\int\limits_{x_1}^{x_2}\left(\sum_{i=1}^{n}\left(F_{y_i}-\frac{dF_{y_i'}}{dx}+\sum_{j=1}^{m}\lambda_j(x)\frac{\partial\varphi_j}{\partial y_i}\right)\delta y_i+\sum_{j=1}^{m}\lambda_j(x)\frac{\partial\varphi_j}{\partial y_i'}\delta y_j'\right)dx=0. \tag{14.30}$$

Его можно также проинтегрировать по частям, как мы это делали при выводе (14.3). Получим:

$$\int\limits_{x_1}^{x_2}\sum_{i=1}^{n}\left(F_{y_i}-\frac{dF_{y_i'}}{dx}+\sum_{j=1}^{m}\left(\lambda_j(x)\frac{\partial\varphi_j}{\partial y_i}-\frac{d}{dx}\left(\lambda_j(x)\frac{\partial\varphi_j}{\partial y_i'}\right)\right)\right)\delta y_i\,dx=0. \tag{14.31}$$

Теперь те же самые рассуждения, что мы проводили после получения формулы (14.5), приводят вместо (14.6) к уравнениям

$$\begin{cases} F_{y_i} - \dfrac{dF_{y_i'}}{dx} + \displaystyle\sum_{j=1}^{m}\left(\lambda_j(x)\frac{\partial\varphi_j}{\partial y_i} - \frac{d}{dx}\left(\lambda_j(x)\frac{\partial\varphi_j}{\partial y_i'}\right)\right) = 0; \\ i = \overline{1,n}. \end{cases} \tag{14.32}$$

Введя в рассмотрение функцию Лагранжа (14.7), где все φ_j зависят уже, конечно, и от производных, получим, что дифференциальные уравнения Эйлера для нее будут как раз давать (14.32) и ограничения (14.28): вместо (14.8) у нас будет:

$$\begin{cases} L_{y_i} - \dfrac{dL_{y_i'}}{dx} = F_{y_i} - \dfrac{dF_{y_i'}}{dx} + \displaystyle\sum_{j=1}^{m}\left(\lambda_j(x)\frac{\partial\varphi_j}{\partial y_i} - \frac{d}{dx}\left(\lambda_j(x)\frac{\partial\varphi_j}{\partial y_i'}\right)\right) = 0; \\ L_{\lambda_j} - \dfrac{dL_{\lambda_j'}}{dx} = \varphi_j\left(y_1, y_2, \ldots, y_n, y_1', y_2', \ldots, y_n'\right) = 0. \end{cases} \tag{14.33}$$

Значит, мы и в этом случае можем ввести в рассмотрение вспомогательный функционал (14.9) и исследовать его на обычный экстремум. Конечно, в этом случае в (14.9) все функции φ_j зависят не только от функций y_i, но и от их производных. А в остальном задача при неголономных ограничениях решается так же, как и при голономных.

14.3. Изопериметрические задачи

Определение 14.1. *Изопериметрической задачей в узком смысле слова* называется задача исследования на экстремум функционала для одной или двух функций одной переменной (т. е. для плоской или пространственной линии) при ограничении-равенстве на длину этой линии. □

Типичными примерами изопериметрических задач в узком смысле слова являются задача Дидоны и задача о цепной линии *(см. главу 1)*.

Определение 14.2. *Изопериметрической задачей в широком смысле слова* называется вариационная задача вида (3.1—3.2) при дополнительных так называемых "изопериметрических" условиях: ограничениях-равенствах вида

$$\begin{cases} \displaystyle\int_{x_1}^{x_2} F_j\left(x, y_1, y_2, \ldots, y_n, y_1', y_2', \ldots, y_n'\right) dx = l_j = \text{const}; \\ j = \overline{1,m}. \end{cases} \quad \square \tag{14.34}$$

Здесь число ограничений-равенств m может быть и меньше, и равно, и даже больше числа функций n. Так, например, в задаче Дидоны мы имеем одну функцию и одно ограничение-равенство, т. е. $m=n$.

Рассмотрим, как можно решить изопериметрическую задачу с помощью метода неопределенных множителей Лагранжа. Для этого нам нужно попытаться как-то перейти от ограничений в виде (14.34) к голономным ограничениям вида (14.1) или неголономным вида (14.28). Поступим следующим образом. Введем в рассмотрение новые, дополнительные переменные z_j с помощью соотношений:

$$\begin{cases} z_j(x)=\int\limits_{x_1}^{x} F_j\left(x,y_1,y_2,\ldots,y_n,y_1',y_2',\ldots,y_n'\right)dx; \\ j=\overline{1,m}\,. \end{cases} \tag{14.35}$$

Тогда для функций z_j имеем дифференциальные соотношения:

$$\begin{cases} F_j\left(x,y_1,y_2,\ldots,y_n,y_1',y_2',\ldots,y_n'\right)-z_j'=0\,; \\ j=\overline{1,m}; \end{cases} \tag{14.36}$$

и граничные условия:

$$\begin{cases} z_j(x_1)=0\,; \\ z_j(x_2)=l_j\,; \\ j=\overline{1,m}. \end{cases} \tag{14.37}$$

Теперь мы можем считать, что наш функционал (4.1) зависит не только от n функций y_i и их производных, но и от m функций z_j и их производных (хотя эти z_j и не входят в функционал явно). Но для функций z_j у нас есть граничные условия (14.37) и неголономные ограничения (14.36)! Тем самым мы свели изопериметрическую задачу к задаче на условный экстремум. Применяем теперь метод неопределенных множителей Лагранжа. Записываем вспомогательный функционал вида (14.9), зависящий от $(n+m)$ функций и m множителей Лагранжа:

$$\begin{aligned} &J^*\left(y_1,\ldots,y_n,z_1,\ldots,z_m,y_1',\ldots,y_n',z_1',\ldots,z_m',\lambda_1,\ldots,\lambda_m\right)= \\ &=\int\limits_{x_1}^{x_2} F^*\left(y_1,\ldots,y_n,z_1,\ldots,z_m,y_1',\ldots,y_n',z_1',\ldots,z_m',\lambda_1,\ldots,\lambda_m\right)dx\,, \end{aligned} \tag{14.38}$$

где

$$\begin{aligned} &F^*\left(y_1,\ldots,y_n,z_1,\ldots,z_m,y_1',\ldots,y_n',z_1',\ldots,z_m',\lambda_1,\ldots,\lambda_m\right)= \\ &=F\left(x,y_1,\ldots,y_n,y_1',\ldots,y_n'\right)+\sum_{j=1}^{m}\lambda_j(x)\left(F_j\left(x,y_1,\ldots,y_n,y_1',\ldots,y_n'\right)-z_j'\right). \end{aligned} \tag{14.39}$$

Составляем уравнения Эйлера. Запишем вначале уравнения для переменных z_j:

$$\begin{cases} F^*_{z_j} - \dfrac{dF^*_{z'_j}}{dx} = \dfrac{d\lambda_j}{dx} = 0 ; \\ j = \overline{1,m}. \end{cases} \tag{14.40}$$

Оказывается, в изопериметрической задаче все множители Лагранжа постоянны:

$$\begin{cases} \lambda_j = \text{const} ; \\ j = \overline{1,m}. \end{cases} \tag{14.41}$$

Это несколько упрощает дальнейшие выкладки. Записываем теперь уравнения Эйлера для переменных y_i, уже с учетом (14.41):

$$\begin{cases} F^*_{y_i} - \dfrac{dF^*_{y'_i}}{dx} = F_{y_i} + \sum\limits_{j=1}^{m} \lambda_j \dfrac{\partial F_j}{\partial y_i} - \dfrac{d}{dx}\left(F_{y'_i} + \sum\limits_{j=1}^{m} \lambda_j \dfrac{\partial F_j}{\partial y'_i} \right) = 0 ; \\ i = \overline{1,n}. \end{cases} \tag{14.42}$$

Третья группа уравнений Эйлера — для переменных λ_j — дает ограничения (14.36).

Из (14.42) мы видим, что вспомогательный функционал может быть построен значительно проще, чем (14.38—14.39): он не будет зависеть от переменных z_j, и множители Лагранжа будут постоянными:

$$\begin{aligned} &J^{**}\left(y_1,\ldots,y_n,y'_1,\ldots,y'_n,\lambda_1,\ldots,\lambda_m\right) = \\ &= \int\limits_{x_1}^{x_2} \left(F\left(x,y_1,\ldots,y_n,y'_1,\ldots,y'_n\right) + \sum_{j=1}^{m} \lambda_j F_j\left(x,y_1,\ldots,y_n,y'_1,\ldots,y'_n\right) \right) dx . \end{aligned} \tag{14.43}$$

Таким образом, для получения необходимых условий экстремума в изопериметрической задаче нужно составить вспомогательный функционал (14.43), где $\forall\lambda_j = \text{const}$, записать для него систему уравнений Эйлера и решить ее. Произвольные постоянные (их $2n$ штук) и m множителей Лагранжа λ_j находятся из $2n$ граничных условий (4.2) и m условий изопериметричности (14.35).

Заметим, что система уравнений Эйлера не изменится, если функционал (14.43) умножить на постоянный множитель μ_0:

$$\mu_0 J^{**} = \int\limits_{x_1}^{x_2} \sum_{j=0}^{m} \mu_j F_j \, dx \to \text{extr} , \tag{14.44}$$

где $F_0 = F$; $\mu_j = \lambda_j \mu_0$. Но теперь задача приобрела симметричную форму! И какую из F_j теперь считать целевой функцией, а какие — ограничениями, не имеет значения. Отсюда следует очень важный *принцип двойственности теории оптимизации*: решение задачи на экстремум функционала A при ограничении на функционал B — такое же, как и решение задачи на экстремум функционала B при ограничении на функционал A.

Более подробное исследование вопросов теории двойственности не входит в нашу задачу.

ПРИМЕР 14.2. Задача о цепной линии *(см. главу 1)*. Математически эта задача ставится так: для функции $y(x)$, проходящей через 2 заданные точки $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$, минимизировать потенциальную энергию (y-ю координату центра тяжести) при условии, что длина линии постоянна и больше длины отрезка M_1M_2. Полистайте свои старые конспекты по математическому анализу и вспомните, как находится центр тяжести плоской однородной линии. Вот какую задачу мы будем решать:

$$\begin{aligned} &y_c = \int_{x_1}^{x_2} y\sqrt{1+y'^2}\,dx \to \min; \quad \begin{cases} y(x_1) = y_1; \\ y(x_2) = y_2; \end{cases} \\ &\int_{x_1}^{x_2} \sqrt{1+y'^2}\,dx = l > \left|\vec{M_1M_2}\right|. \end{aligned} \tag{14.45}$$

Составляем вспомогательную функцию вида (14.43) для нашего функционала:

$$F = y\sqrt{1+y'^2}; \quad F_1 = \sqrt{1+y'^2}; \quad F^* = F + \lambda F_1 = (y+\lambda)\sqrt{1+y'^2}. \tag{14.46}$$

Для нее нам теперь нужно записать уравнение Эйлера. Эта функция не зависит явно от x, поэтому записываем 1-й интеграл в виде (2.64):

$$F^* - y'F^*_{y'} = (y+\lambda)\sqrt{1+y'^2} - \frac{(y+\lambda)y'^2}{\sqrt{1+y'^2}} = C_1. \tag{14.47}$$

Умножаем на корень в знаменателе и упрощаем:

$$y + \lambda = C_1\sqrt{1+y'^2}. \tag{14.48}$$

Решаем полученное уравнение с помощью гиперболической подстановки (вспомните основную формулу гиперболической геометрии: $\mathrm{ch}^2 x - \mathrm{sh}^2 x = 1$):

$$y' = \mathrm{sh}t; \quad y = C_1\mathrm{ch}t - \lambda. \tag{14.49}$$

Мы выразили через параметр t переменную y, далее находим x:

$$dy = C_1 \operatorname{sh} t dt; \quad dx = \frac{dy}{y'} = C_1 dt; \quad x = C_1 t + C_2. \tag{14.50}$$

Исключаем параметр t из полученного решения:

$$t = \frac{x - C_2}{C_1}; \quad y = C_1 \operatorname{ch} \frac{x - C_2}{C_1} - \lambda. \tag{14.51}$$

Решение нашей задачи — гиперболический косинус, который так и называется: *цепная линия*. Произвольные постоянные C_1, C_2 и неопределенный множитель Лагранжа λ находим из граничных условий и условия изопериметричности (постоянства длины):

$$\begin{cases} y(x_1) = y_1; \\ y(x_2) = y_2; \\ \int\limits_{x_1}^{x_2} \sqrt{1 + y'^2}\, dx = l. \end{cases} \tag{14.52}$$

Имеем систему 3-х нелинейных уравнений:

$$\begin{cases} C_1 \operatorname{ch} \dfrac{x_1 - C_2}{C_1} - \lambda = y_1; \\ C_1 \operatorname{ch} \dfrac{x_2 - C_2}{C_1} - \lambda = y_2; \\ C_1 \left(\operatorname{sh} \dfrac{x_2 - C_2}{C_1} - \operatorname{sh} \dfrac{x_1 - C_2}{C_1} \right) = l. \end{cases} \tag{14.53}$$

Ее можно решить численно и подставить полученные значения C_1, C_2 и λ в решение (14.51). □

ПРИМЕР 14.3. Задача Дидоны. Требуется максимизировать площадь между кривой, соединяющей точки $M_1(0, 0)$ и $M_2(x_2, 0)$, и осью Ox при условии, что длина дуги линии постоянна и больше длины отрезка M_1M_2. В одном варианте абсцисса правой точки x_2 задана, в другом — правая точка может свободно перемещаться вдоль оси Ox. Мы решим оба варианта задачи. Вот ее математическая формулировка:

$$S = \int\limits_0^{x_2} y\, dx \to \max; \quad \begin{cases} y(0) = 0; \\ y(x_2) = 0; \end{cases} \quad \int\limits_0^{x_2} \sqrt{1 + y'^2}\, dx = l > x_2. \tag{14.54}$$

Второй вариант (подвижная правая точка скользит вдоль оси Ox) отличается от первого (неподвижная правая точка) только граничными условиями. Дифференциальное уравнение Эйлера будет одинаковым в обоих случаях.

Составляем подынтегральную функцию для вспомогательного функционала вида (14.43):

$$F = y; \quad F_1 = \sqrt{1 + y'^2}; \quad F^* = F + \lambda F_1 = y + \lambda\sqrt{1 + y'^2}. \tag{14.55}$$

Как и в задаче о цепной линии, здесь подынтегральная функция не зависит явно от x, поэтому записываем 1-й интеграл уравнения Эйлера вида (3.64):

$$F^* - y'F^*_{y'} = y + \lambda\sqrt{1 + y'^2} - y'\lambda\frac{y'}{\sqrt{1 + y'^2}} = C_1. \tag{14.56}$$

Умножаем на знаменатель (он никогда не равен нулю) и упрощаем:

$$y\sqrt{1 + y'^2} + \lambda\left(1 + y'^2\right) - \lambda y'^2 = C_1\sqrt{1 + y'^2}; \quad \left(y - C_1\right)\sqrt{1 + y'^2} = -\lambda. \tag{14.57}$$

Здесь удобнее ввести параметр путем тригонометрической подстановки:

$$y' = \operatorname{ctg} t; \quad \frac{y - C_1}{\sin t} = -\lambda; \quad y = C_1 - \lambda\sin t. \tag{14.58}$$

Находим x:

$$dy = -\lambda\cos t dt; \quad dx = \frac{dy}{y'} = -\lambda\sin t dt; \quad x = C_2 + \lambda\cos t. \tag{14.59}$$

Исключаем параметр t из последних уравнений (14.58—14.59). Получаем уравнение окружности:

$$(x - C_2)^2 + (y - C_1)^2 = \lambda^2. \tag{14.60}$$

Значит, решением задачи Дидоны является дуга окружности. Ее центр и радиус мы найдем из двух граничных условий и условия изопериметричности.

Вариант 1. Правая точка задана: $M_2(x_2, 0)$. В этом случае имеем систему 3-х уравнений с 3-мя неизвестными, которую удобнее решать в параметрической форме, т. к. явное уравнение $y(x)$ может быть неоднозначным (рис. 14.3).

Введем в рассмотрение углы: α_1 (начальный) и α_2 (конечный, он отрицательный, т. к. отсчитывается по часовой стрелке). Центральный угол, охватываемый дугой, равен:

$$\alpha = \alpha_1 - \alpha_2. \tag{14.61}$$

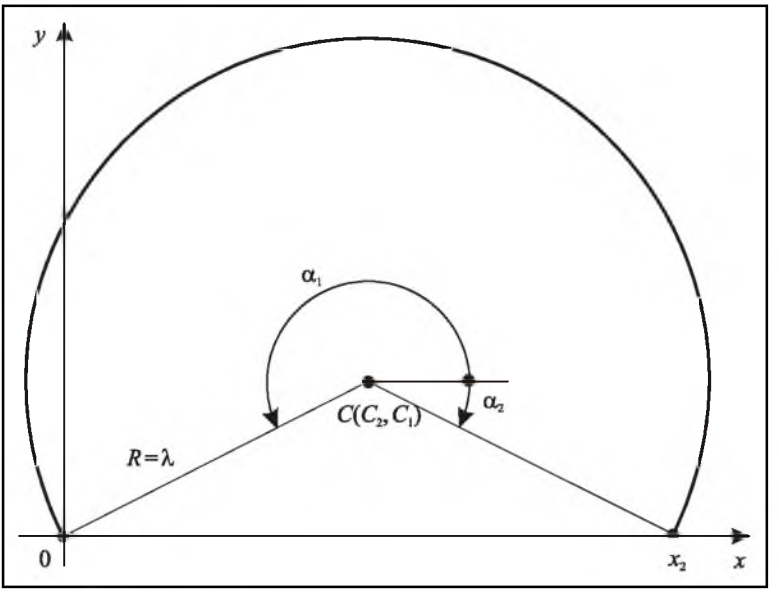


Рис. 14.3. Задача Дидоны — вариант 1, выполненный с помощью MATLAB

Получаем систему 2-х уравнений для нахождения неизвестных R и α:

$$\begin{cases} R\alpha = l; \\ 2R\sin\dfrac{\alpha}{2} = x_2. \end{cases} \tag{14.62}$$

Ее можно решить численно, а затем найти:

$$\begin{cases} C_2 = \dfrac{x_2}{2}; \\ C_1 = \begin{cases} \sqrt{R^2 - C_2^2}; & \alpha \geq \pi; \\ -\sqrt{R^2 - C_2^2}; & \alpha < \pi. \end{cases} \end{cases} \tag{14.63}$$

Вариант 2. Правая точка $M_2(x_2, 0)$ скользит вдоль оси Ox, т. е. x_2 не задано. Здесь нам придется воспользоваться условием трансверсальности (8.9). Мы движемся по горизонтальной прямой, поэтому $\varphi'(x) = 0$, и это условие имеет вид:

$$\left(F - y'F_{y'}\right)\Big|_{x=x_2} = 0. \tag{14.64}$$

Полученное выражение совпадает с нашим 1-м интегралом уравнения Эйлера (14.56) при $C_1 = 0$. Значит, и решением его будет $C_1 = 0$: центр окружности

в этом случае лежит на оси Ox, и решением задачи Дидоны является полуокружность. □

14.4. Вопросы для самопроверки

1. Какую вариационную задачу мы решаем?
2. Чем отличаются голономные ограничения от неголономных? Чем отличается решение задачи для одного и другого случая?
3. Когда неопределенные множители Лагранжа будут функциями от x, а когда постоянными?
4. Какая задача называется изопериметрической в узком смысле слова? В широком смысле слова?
5. Какая кривая является решением задачи о цепной линии?
6. Какая кривая является решением задачи Дидоны?
7. В каком случае при решении задачи Дидоны охватываемая площадь будет больше: когда правая точка фиксирована или когда она может скользить вдоль оси Ox?

14.5. Примеры выполнения заданий

14.5.1. Задание 1

Найти экстремаль функционала, который мы уже исследовали в *главе 2 (задание 1)*, но с дополнительным условием:

$$J(y)=\int_{-1}^{1}\left(x^2+y^2+y'^2\right)dx;\quad \begin{cases} y(-1)=1; \\ y(1)=2; \end{cases}\quad \int_{x_1}^{x_2} y\,dx=2. \tag{14.65}$$

Сравнить решение с решением *задания 1 главы 2*. Вычислить значения функционалов на обеих кривых. Построить графики.

Составим программу для этого примера на основе программы для *задания 1 главы 2* с использованием других программ. Вначале описываем необходимые переменные и вводим исходные данные. Повторяем полностью решение *задания 1 главы 2*.

```
clear all % очистили память
disp('Решаем задание 14.1')
syms x y Dy D2y lambda
F=x^2+y^2+Dy^2; % подынтегральная функция
```

```
x1=-1; % вводим граничные условия
y1=1;
x2=1;
y2=2;
F1=y;
J1=2;
disp('Исходные данные:')
fprintf('F(x,y,y'')=%s\n',char(F))
fprintf('Граничное условие слева: y(%d)=%d\n',x1,y1)
fprintf('Граничное условие справа: y(%d)=%d\n',x2,y2)
disp('Условие изопериметричности:')
fprintf('Int(%s,''x'',%d,%d)=%d\n',char(F1),x1,x2,J1)
dFdy=diff(F,y); % вычислили Fy
dFdy1=diff(F,Dy); % вычислили Fy'
d_dFdy1_dx=diff(dFdy1,x); % Fy'x
d_dFdy1_dy=diff(dFdy1,y); % Fy'y
d_dFdy1_dy1=diff(dFdy1,Dy); % Fy'y'-условие Лежандра
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+d_dFdy1_dy1*D2y;
Euler=simple(dFdy-dFy1dx);
deqEuler=[char(Euler) '=0']; % добавили =0
Sol=dsolve(deqEuler,'x'); % решаем уравнение Эйлера
if length(Sol)~=1 % нет решений или более одного
  error('Нет решений или более одного решения!');
end
SolLeft=subs(Sol,x,x1); % подставили x1
SolRight=subs(Sol,x,x2); % подставили x2
EqLeft=[char(SolLeft) '=' char(sym(y1))]; % =y1
EqRight=[char(SolRight) '=' char(sym(y2))]; % =y2
Con=solve(EqLeft,EqRight,'C1,C2'); % решаем систему
C1=Con.C1; % присваиваем полученные решения
C2=Con.C2; % символическим константам C1 и C2
Sol21=vpa(eval(Sol),14); % подставляем C1,C2, вычисляем с 14 зн.
xpl=linspace(x1,x2); % задаем массив абсцисс
y21=subs(Sol21,x,xpl); % вычислили ординаты
Решаем задание 14.1
Исходные данные:
F(x,y,y')=x^2+y^2+Dy^2
Граничное условие слева: y(-1)=1
Граничное условие справа: y(1)=2
Условие изопериметричности:
Int(y,'x',-1,1)=2
```

Приступаем к решению нашего примера. Формируем вспомогательный функционал вида (14.43), записываем для него дифференциальное уравнение Эйлера и решаем его.

```
L=F+lambda*F1; % модифицированный функционал
fprintf('L(x,y,y'',lambda)=%s\n',char(L))
dLdy=diff(L,y);
dLdy1=diff(L,Dy);
d_dLdy1_dx=diff(dLdy1,x);
d_dLdy1_dy=diff(dLdy1,y);
d_dLdy1_dy1=diff(dLdy1,Dy); % d(dL/dy')/dy'
dLy1dx=d_dLdy1_dx+d_dLdy1_dy*Dy+d_dLdy1_dy1*D2y;
EulerL=simple(dLdy-dLy1dx);
deqEulerL=[char(EulerL) '=0']; % составили уравнение
fprintf('Уравнение Эйлера:\n%s\n',deqEulerL)
SolL=dsolve(deqEulerL,'x'); % решаем уравнение Эйлера
if length(SolL)~=1 % нет решений или более одного
  error('Нет решений или более одного решения!');
else
  disp('Общее решение:')
  fprintf('y(x)=%s\n',char(SolL))
end
```

```
L(x,y,y',lambda)=x^2+y^2+Dy^2+lambda*y
Уравнение Эйлера:
2*y+lambda-2*D2y=0
Общее решение:
y(x)=-1/2*lambda+C1*sinh(x)+C2*cosh(x)
```

Мы получили общее решение дифференциального уравнения, в которое входят 2 произвольные постоянные и неопределенный множитель Лагранжа. Для их нахождения у нас есть 2 граничных условия и условие изопериметричности. Вычисляем левую часть условия изопериметричности.

```
dydx=diff(SolL,x); % dy/dx
F1_y=subs(F1,{y,Dy},{SolL,dydx});
intF1=vpa(int(F1_y,x,x1,x2),14); % вычислили интеграл
disp('Левая часть условия изопериметричности:')
disp(char(intF1))
```

```
Левая часть условия изопериметричности:
-1.*lambda+2.3504023872876*C2
```

Формируем систему уравнений и решаем ее — находим произвольные постоянные и множитель Лагранжа. В данных примерах система получается линейной, и мы решаем ее с помощью команды `solve`. Печатаем решения.

```
SolLleft=vpa(subs(SolL,x,x1),14);
SolLright=vpa(subs(SolL,x,x2),14);
LeftL=[char(SolLleft) '=' char(sym(y1))];
RightL=[char(SolLright) '=' char(sym(y2))];
```

```
intF1J1=[char(intF1) '=' char(sym(J1))];
disp('Система уравнений относительно C1, C2, lambda')
fprintf('%s\n',LeftL,RightL,intF1J1)
ConL=solve(LeftL,RightL,intF1J1,'C1,C2,lambda');
C1=vpa(ConL.C1,14);
C2=vpa(ConL.C2,14);
lambda=vpa(ConL.lambda,14); % множитель Лагранжа
Sol141=vpa(eval(SolL),14); % аналитическое решение
disp('Уравнение экстремали:')
fprintf('y(x)=%s\n',char(Sol141))
disp('Множитель Лагранжа')
fprintf('lambda=%s\n',char(lambda))
Система уравнений относительно C1, C2, lambda
-.50000000000000*lambda-1.1752011936438*C1+
  1.5430806348152*C2=1
-.50000000000000*lambda+1.1752011936438*C1+
  1.5430806348152*C2=2
-1.*lambda+2.3504023872876*C2=2
Уравнение экстремали:
y(x)=-.59726402473285+.42545906411966*sinh(x)+
  1.3591409142297*cosh(x)
Множитель Лагранжа
lambda=1.1945280494657
```

Вычисляем значения функционалов на обоих решениях. Заполняем таблицу и строим графики: сплошной линией — *задания 14.1*, а штриховой — *задания 2.1* (для сравнения). Результат показан на рис. 14.4.

```
F21=subs(F,{y,Dy},{Sol21,diff(Sol21,x)});
J21=eval(int(F21,x,x1,x2))
F141=subs(F,{y,Dy},{Sol141,diff(Sol141,x)});
J141=eval(int(F141,x,x1,x2))
y141=subs(Sol141,x,xpl); % вычислили функцию
plot(xpl,y21,'--b', xpl,y141,'-r') % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 14.1') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
J21 =
   4.75035801121746
J141 =
   4.92044833414949
```

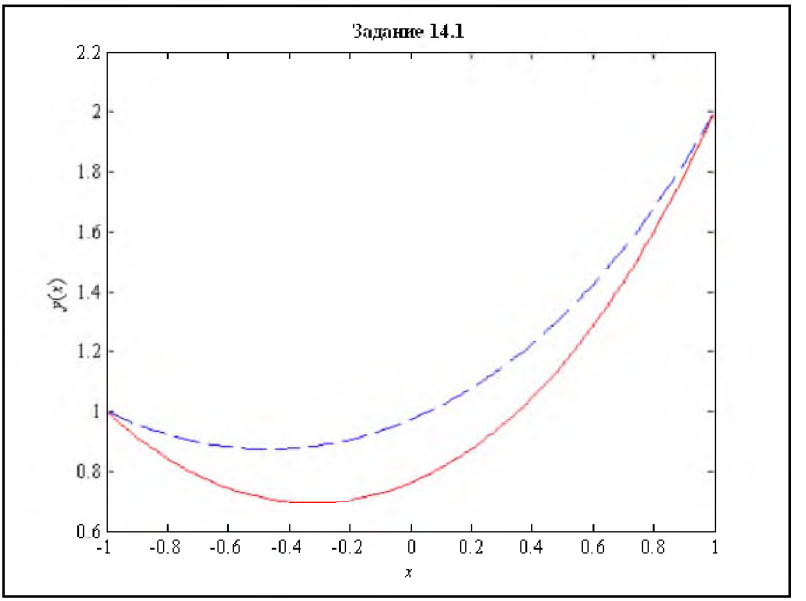


Рис. 14.4. Решение задания 14.1, выполненное с помощью MATLAB

14.5.2. Задание 2

Решить задачу о цепной линии длиной $l = 8$, соединяющей точки $M_1(-2, 1)$ и $M_2(3, 3)$. Построить график.

Мы уже решили эту задачу в общем виде, и знаем, что ее решением является цепная линия. Нам осталось найти произвольные постоянные и неопределенный множитель Лагранжа из системы нелинейных уравнений (14.53). Вводим вначале исходные данные.

```
clear all % очистили память
disp('Решаем задание 14.2') % выводим заголовок задачи
x1=-2;
y1=1;
x2=3;
y2=3;
l=8;
fprintf('Левая точка: y(%d)=%d\n',x1,y1)
fprintf('Правая точка: y(%d)=%d\n',x2,y2)
fprintf('Длина цепи = %d\n',l)
Решаем задание 14.2
Левая точка: y(-2)=1
Правая точка: y(3)=3
Длина цепи = 8
```

Систему (14.53) мы будем решать численно, с помощью команды `fsolve`. Для упрощения мы вычли из 2-го уравнения системы (14.53) 1-е, и оставили только систему двух уравнений с двумя неизвестными. Заполняем файл с функцией, которая вычисляет левые части нашей системы.

```
s{1}='function y = MyFunc(x)'; % заголовок
s{2}='C1=x(1); C2=x(2); y=x;';
s{3}=['y(1)=C1*(cosh((' char(sym(x2)) ...
  '-C2)/C1)-cosh((' char(sym(x1)) ...
  '-C2)/C1))-(' char(sym(y2)) ')+(' char(sym(y1)) ');'];
s{4}=['y(2)=C1*(sinh((' char(sym(x2)) ...
  '-C2)/C1)-sinh((' char(sym(x1)) ...
  '-C2)/C1))-(' char(sym(1)) ');'];
filename=fullfile(pwd,'MyFunc.m'); % имя файла
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:});
fid=fopen(filename,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записали в файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\MyFunc.m:
function y = MyFunc(x)
C1=x(1); C2=x(2); y=x;
y(1)=C1*(cosh((3-C2)/C1)-cosh((-2-C2)/C1))-(3)+(1);
y(2)=C1*(sinh((3-C2)/C1)-sinh((-2-C2)/C1))-(8);
```

Задаем начальное приближение, решаем систему уравнений и печатаем полученное решение.

```
xinit=[0.5;(x1+x2)/2]; % начальное приближение
options=optimset('fsolve'); % опции по умолчанию
options=optimset(options,'Display','off',...
  'MaxIter',1000,'TolX',1e-8);
[xzero,fval,exitflag]=fsolve('MyFunc',xinit,options);
if exitflag>0,
  disp('Решение найдено');
elseif exitflag<0,
  disp('Решение не найдено - нет сходимости');
else
  disp(['Проведено максимально допустимое число ' ...
        'итераций - сходимость медленная']);
end
C1=xzero(1);
C2=xzero(2);
lambda=C1*cosh((x1-C2)/C1)-y1;
```

```
disp('Решение системы:')
fprintf('C1=%12.6f\nC2=%12.6f\nlambda=%12.6f\n',C1,C2,lambda)
disp('Уравнение цепной линии:')
fprintf('y(x)=%12.6f*ch((x%+12.6f)/%12.6f)%+12.6f',...
   C1,C2,C1,lambda)
Решение найдено
Решение системы:
C1=     1.478832
C2=     0.122287
lambda=     2.281675
Уравнение цепной линии:
y(x)=1.478832*ch((x+0.122287)/1.478832)+2.281675
```

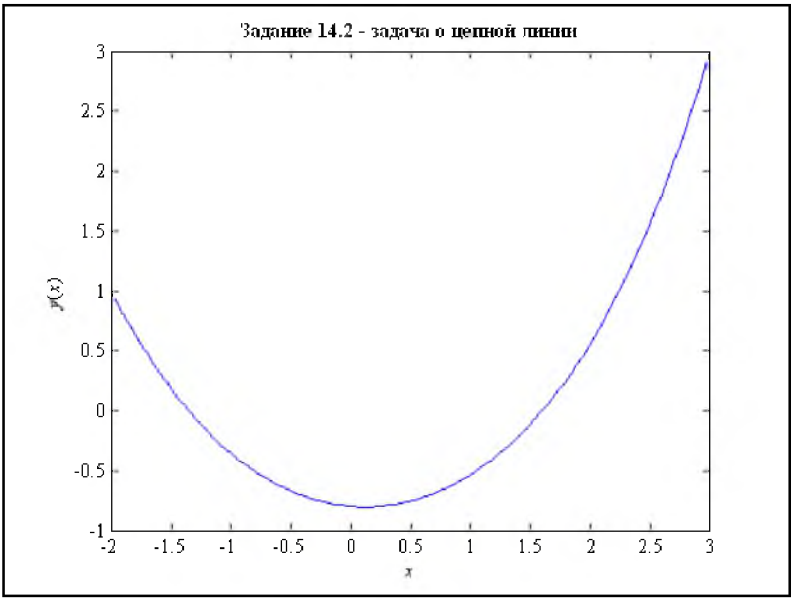


Рис. 14.5. Решение задания 14.2 — задачи о цепной линии, выполненное с помощью MATLAB

Строим график полученной цепной линии (рис. 14.5). Выравниваем коэффициенты масштаба по осям координат, чтобы фигура выглядела неискаженной. Подписываем заголовок и метки координатных осей. Удаляем ненужный теперь файл с записанной туда функцией.

```
x142=linspace(x1,x2); % задали массив абсцисс
y142=C1*cosh((x142-C2)/C1)-lambda; % ординаты
plot(x142,y142) % рисуем график
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
```

```
xlim([x1 x2]) % установили пределы по оси 0x
da=daspect;
da(1:2)=min(da(1:2));
daspect(da); % выравниваем масштаб
title ('\bfЗадание 14.2 - задача о цепной линии')
xlabel ('\itx') % метка оси OX
ylabel ('\ity\rm(\itx\rm)') % метка оси OY
delete(filename)
```

Как вычислить значение функционала на этой линии? Допишите этот фрагмент программы самостоятельно.

14.5.3. Задание 3

Решить задачу Дидоны для линии длиной $l=4$, соединяющей точки $M_1(0,0)$ и $M_2(3,0)$. Рассмотреть также случай, когда правая точка движется вдоль оси Ox. Построить на одном графике обе кривые. Вычислить охватываемую площадь в обоих случаях.

Эту задачу мы также уже решили теоретически. Осталось найти значения произвольных постоянных и неопределенный множитель Лагранжа.

Вводим исходные данные.

```
clear all % очистили память
disp('Решаем задание 14.3') % выводим заголовок задачи
x2=3;
l=4;
fprintf('Правая точка x2=%d\n',x2)
fprintf('Длина линии l=%d\n',l)
Решаем задание 14.3
Правая точка x2=3
Длина линии l=4
```

Заполняем файл с функцией, которая вычисляет левые части системы нелинейных уравнений (14.62).

```
s{1}='function y = MyFunc(x)'; % заголовок
s{2}='R=x(1); alpha=x(2); y=x;';
s{3}=['y(1)=R*alpha-(' char(sym(l)) ');'];
s{4}=['y(2)=2*R*sin(alpha/2)-(' char(sym(x2)) ');'];
filename=fullfile(pwd,'MyFunc.m'); % файл
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:});
fid=fopen(filename,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записали файл
fclose(fid); % закрываем файл
```

```
Текст файла D:\Iglin\Matlab\MyFunc.m:
function y = MyFunc(x)
R=x(1); alpha=x(2); y=x;
y(1)=R*alpha-(4);
y(2)=2*R*sin(alpha/2)-(3);
```

Задаем начальное приближение и решаем систему уравнений (14.62). Печатаем полученное решение.

```
xinit=[x2/2;pi]; % начальное приближение
options=optimset('fsolve'); % опции по умолчанию
options=optimset(options,'Display','off',...
  'MaxIter',1000,'TolX',1e-8);
[xzero,fval,exitflag]=fsolve('MyFunc',xinit,options);
if exitflag>0,
  disp('Решение найдено');
elseif exitflag<0,
  disp('Решение не найдено - нет сходимости');
else
  disp(['Проведено максимально допустимое число ' ...
        'итераций - сходимость медленная']);
end
R=xzero(1);
alpha=xzero(2);
disp('Решение системы:')
fprintf('R=%12.6f\nalpha=%12.6f\n',R,alpha)
x0=x2/2;
y0=(R^2-x0^2)^0.5;
if alpha<pi,
  y0=-y0;
end
disp('Центр окружности:')
fprintf('x0=%12.6f\ny0=%12.6f\n',x0,y0)
t1=(pi-alpha)/2;
t2=(pi+alpha)/2;
disp('Граничные значения параметра:')
fprintf('t1=%12.6f\nt2=%12.6f\n',t1,t2)
Решение найдено
Решение системы:
R=    1.567769
alpha=    2.551396
Центр окружности:
x0=    1.500000
y0=   -0.455960
```

```
Граничные значения параметра:
t1=    0.295098
t2=    2.846494
```

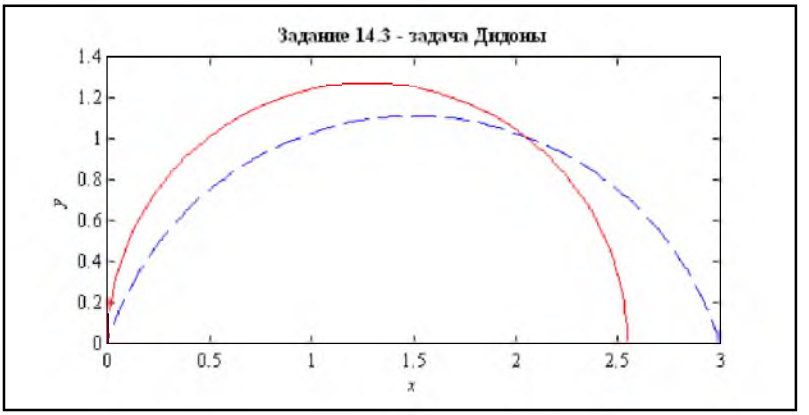


Рис. 14.6. Решение задания 14.3 — задачи Дидоны, выполненное с помощью MATLAB

Находим решение для линии с подвижным правым концом, вычисляем охватываемые площади и строим графики полученных функций (рис. 14.6). Выравниваем масштабы по осям координат, чтобы окружности выглядели окружностями, а не эллипсами. Стираем ненужный теперь файл.

```
t=linspace(t1,t2); % задаем параметр
x143=x0+R*cos(t); % абсциссы
y143=y0+R*sin(t); % ординаты
x2n=2*l/pi; % задача с подвижным концом
Rn=x2n/2; % радиус в этой же задаче
tn=linspace(0,pi); % задаем параметр
x143n=Rn+Rn*cos(tn); % абсциссы
y143n=Rn*sin(tn); % ординаты
S=R^2*alpha/2+x2*y0/2; % площадь
Sm=pi*Rn^2/2; % max площадь
fprintf('Площадь при фиксированном x2: S=%12.6f\n',S)
fprintf('Площадь при подвижном x2: S=%12.6f\n',Sm)
plot(x143,y143,'--b',x143n,y143n,'-r') % график
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
yl=ylim; % нашли пределы по оси 0y
yl(1)=0; % нижний предел 0
da=daspect;
da(1:2)=min(da(1:2));
```

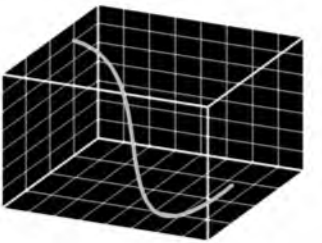
```
daspect(da); % выравниваем масштаб
ylim(yl); % установили пределы по оси Oy
title ('\bfЗадание 14.3 - задача Дидоны')
xlabel ('\itx') % метка оси OX
ylabel ('\ity') % метка оси OY
delete(filename)
Площадь при фиксированном x2: S=    2.451598
Площадь при подвижном x2: S=    2.546479
```

14.6. Задание

1. Решить *задание 1 главы 2* при заданном изопериметрическом условии. Сравнить результат с решением *задания 1 главы 2*.
2. Решить задачу о цепной линии заданной длины l, соединяющей заданные точки $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$.
3. Решить задачу Дидоны для линии заданной длины l, соединяющей точки $M_1(0, 0)$ и $M_2(x_2, 0)$. Найти также решение, если правая точка может скользить вдоль оси Ox. Вычислить охватываемые площади в обоих случаях.

Варианты исходных данных есть на диске.

ГЛАВА 15



Метод начальных параметров

15.1. Метод стрельбы, начальных параметров, матричной прогонки

Мы переходим к рассмотрению численных методов решения задач вариационного исчисления. Ведь дифференциальное уравнение Эйлера (или Эйлера — Пуассона, или Эйлера — Остроградского, или их систему) не всегда просто решить аналитически. Эти уравнения могут быть очень сложными или вообще не интегрироваться в конечном виде. В этом случае их приходится решать численно.

Ранее *(см. главы 8—10, 12—13)* нам уже приходилось сталкиваться с численным решением нелинейной системы уравнений, но не дифференциальных, а конечных. Дифференциальные уравнения Эйлера мы почти всегда решали аналитически (исключение составляет уравнение Эйлера — Остроградского для двумерной области сложной формы, которое мы решали в *главе 5* методом конечных элементов). В этой главе мы рассмотрим один из методов численного решения системы *обыкновенных* дифференциальных уравнений с граничными условиями, который называется в разных приложениях или методом начальных параметров, или методом стрельбы, или методом матричной прогонки.

Давайте вспомним, какие обыкновенные дифференциальные уравнения или их системы нам встречались. Во-первых, в *главе 2* мы вывели дифференциальное уравнение Эйлера (2.9), которое является уравнением 2-го порядка и дополняется двумя граничными условиями (2.2). Далее, в *главе 3* мы получили систему n дифференциальных уравнений Эйлера (3.4), дополненную $2n$ граничными условиями (3.2). Эти уравнения также являются уравнениями 2-го порядка. И наконец, в *главе 4* было получено дифференциальное уравне-

ние Эйлера — Пуассона (4.8), которое является уравнением 4-го порядка и дополняется 4-мя граничными условиями (4.2). Граничные условия у нас во всех случаях имеют одинаковый вид: искомая функция или ее производная какого-либо порядка на одном из концов интервала равна заданной величине.

Из теории обыкновенных дифференциальных уравнений нам известно, что любое уравнение n-го порядка, разрешенное относительно высшей производной, всегда можно свести к системе n нормальных дифференциальных уравнений 1-го порядка: для этого нужно ввести новые функции по правилу:

$$y_1 = y;\quad y_2 = y';\quad y_3 = y'';\quad \ldots;\quad y_n = y^{(n-1)}. \tag{15.1}$$

После такой замены получим:

$$\frac{d\mathbf{Y}}{dx} = \mathbf{F}(x, \mathbf{Y}), \tag{15.2}$$

где $\mathrm{Y}(x)$ — вектор искомых функций, $\mathrm{F}(x,\mathrm{Y})$ — известная вектор-функция, зависящая от x и $\mathrm{Y}(x)$. Она обычно называется *правыми частями* системы (15.2). Конечно же, не только одно уравнение, но и систему n уравнений 2-го порядка можно этой заменой свести к системе $2n$ нормальных уравнений 1-го порядка. Далее буквой n будем обозначать суммарный порядок уравнений системы, т. е. количество уравнений в (15.2). Например, если у нас есть система трех дифференциальных уравнений Эйлера 2-го порядка, то мы получим $n = 6$.

Во всех рассматриваемых случаях система (15.2) дополняется n граничными условиями: каждая из функций $y_i(x)$ задана либо на левом конце интервала x_1, либо на правом x_2, либо на обоих, либо нигде не задана, но общее число граничных условий равно n. В наших задачах n — четное, и половина граничных условий задана на левом конце, а половина — на правом, но для метода начальных параметров это не обязательно. Главное, чтобы задача была определена: должно быть задано n граничных условий, а сколько из них на левом конце и сколько на правом — не имеет значения.

Если все граничные условия заданы на левом конце, то это будут уже не *краевая*, а *начальная* задача или задача Коши: $\mathrm{Y}(x_1) = \mathrm{Y}_1$, и такую систему можно решить с помощью стандартного программного обеспечения, например, методов Рунге — Кутта, Ньютона — Котеса, Адамса и др. Мы предполагаем, что у нас имеется программное обеспечение для решения начальной задачи для нормальной системы дифференциальных уравнений. В MATLAB для численного решения начальной задачи имеются прекрасные функции (они называются *решатели обыкновенных дифференциальных уравнений*). Вот имена некоторых из них: `ode23`, `ode113`, `ode45` (`ode` — это сокращение от ordinal differential equation: обыкновенное дифференциальное уравнение). Для

их работы нужно задать вид системы (15.2), т. е. построить и записать m-файл: функцию, которая вычисляет правые части системы (15.2) по заданным значениям x и $Y(x)$. Имя этого файла и передается решателю в качестве параметра. Ну и, конечно, нужно также задать начальные условия $Y(x_1) = Y_1$ и те точки x, в которых надо получить решение. Пример решения такой задачи есть в *приложении 1*.

Рассмотрим теперь случай, когда в системе (15.2) часть координат вектора Y задана на левом конце x_1, а часть — на правом x_2. Пусть, для определенности, на правом конце заданы m первых координат вектора Y, а на левом конце — $n - m$ любых (не обязательно последних). Этого всегда можно добиться перенумерацией переменных. Если бы нам удалось найти недостающие координаты вектора Y на левом конце, то можно было бы решить начальную задачу. Будем рассматривать m недостающих координат Y в начальной точке x_1 как неизвестные $t_1, t_2, ..., t_m$. Для их нахождения имеется m уравнений — граничные условия на правом конце x_2. Таким образом, задача нахождения неизвестных начальных условий сводится к решению системы из m в общем случае нелинейных конечных уравнений. Эти недостающие параметры (координаты вектора Y в начальной точке x_1) называются *начальными*, а метод, который мы рассматриваем, — *метод начальных параметров*. Решив эту систему, найдем недостающие начальные условия, и теперь можно решить начальную задачу стандартными решателями. Пример численного решения системы нелинейных конечных уравнений есть в *приложении 1*.

Рассмотрим геометрическую интерпретацию метода начальных параметров для $n = 2$, $m = 1$. Пусть имеется дифференциальное уравнение Эйлера 2-го порядка вида (2.9), и в соответствии с (15.1) мы обозначили $y_1 = y$; $y_2 = y'$. Граничные условия у нас: $y(x_1) = y_1(x_1) = y_1^0$; $y(x_2) = y_1(x_2) = y_2^0$. Производная в начальной точке (угол наклона экстремали) не задана, зато задано значение самой функции в конечной точке x_2. Задавая различные значения для недостающего начального условия $y'(x_1)$, будем получать различные $y(x_2)$. Чтобы получить нужное значение y_2^0, необходимо угадать угол наклона экстремали в начальной точке. Это напоминает процесс стрельбы из артиллерийского орудия. Поэтому метод начальных параметров называется также *методом стрельбы* (рис. 15.1).

В MATLAB для решения систем нелинейных конечных уравнений можно использовать команду `fsolve`. Ей нужно передать в качестве параметра имя функции (m-файла), которая по заданным значениям аргументов (неизвестные m начальных параметров $t_1, t_2, ..., t_m$) вычисляет значения функций, которые должны равняться нулю. У нас это — значения вычисленных m граничных условий на правом конце минус их действительные, заданные значения. Для нахождения этих функций при заданных $t_1, t_2, ..., t_m$ нужно интегрировать

систему дифференциальных уравнений с помощью одного из решателей ode, в которых вызывается другая составленная нами m-функция, вычисляющая правые части системы (15.2).

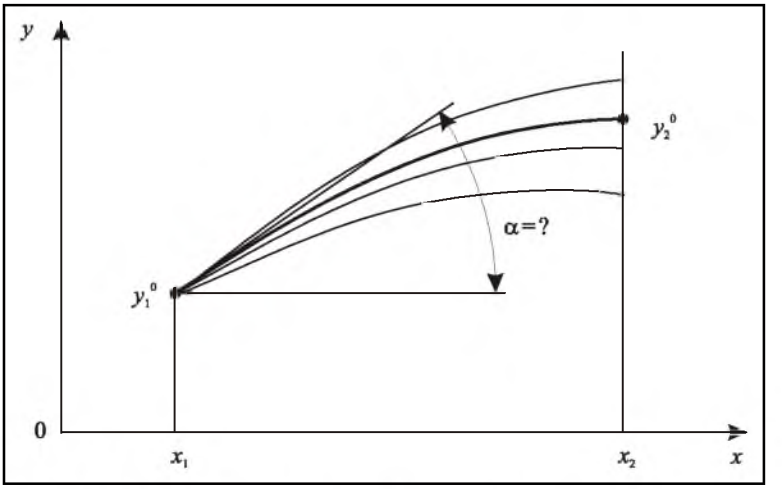


Рис. 15.1. Геометрическая интерпретация метода начальных параметров

Таким образом, мы получаем примерно такую же "матрешечную" конструкцию, как та, что показана на рис. 1.16, только не из пяти, а из трех матрешек. Самая внутренняя матрешка — это файл, в котором по заданным значениям x и $Y(x)$ вычисляются правые части нормальной системы дифференциальных уравнений (15.2) — функции $F(Y, x)$.

Следующая по вложенности матрешка — это файл, в котором по заданным пробным значениям неизвестных начальных параметров $t_1, t_2, ..., t_m$ вычисляются значения функций, которые должны быть равны нулю. Так, для $n=2$, $m=1$ (см. рис. 15.1) по одному заданному начальному условию $y'(x_1)=\operatorname{tg}\alpha$ вычисляется одна величина $y(x_2)-y_2^0$, которая должна быть равна нулю. Для этого численно интегрируется система дифференциальных уравнений (используется предыдущий файл!) при заданных теперь уже начальных условиях.

И наконец, наружная матрешка — это основная программа, в которой решается система нелинейных уравнений (используется предыдущий файл!), определяются неизвестные начальные условия и интегрируется система дифференциальных уравнений (15.2) при этих начальных условиях. При этом мы наверняка "попадем" в нужные граничные условия на правом конце, т. к. именно из этих условий попадания мы и находили неизвестные начальные условия.

Однако в некоторых случаях можно упростить задачу и не составлять три m-файла, а ограничиться двумя. В частности, из теории дифференциальных уравнений известно, что, если исходная система дифференциальных уравнений (15.2) *линейная* (даже с переменными коэффициентами), то и вектор решений в любой точке $Y(x_2)$ также будет *линейно* зависеть от начальных условий:

$$Y(x_2) = AY(x_1) + b. \tag{15.3}$$

Поэтому и система уравнений для нахождения начальных условий также будет *линейной*. Для ее построения оставим в (15.3) нужные строки и столбцы. Мы предполагаем, что неизвестные начальные условия — это какие-либо m координат вектора $Y(x_1)$, а заданы первые m координат вектора $Y(x_2)$, поэтому и оставим в системе (15.3) только m первых строк, соответствующих граничным условиям при $x = x_2$, и только нужные m столбцов, соответствующих неизвестным начальным параметрам:

$$\begin{cases} a_{1i_1} t_1 + a_{1i_2} t_2 + \cdots + a_{1i_m} t_m + b_1 = y_1(x_2); \\ a_{2i_1} t_1 + a_{2i_2} t_2 + \cdots + a_{2i_m} t_m + b_2 = y_2(x_2); \\ \cdots \\ a_{mi_1} t_1 + a_{mi_2} t_2 + \cdots + a_{mi_m} t_m + b_m = y_m(x_2). \end{cases} \tag{15.4}$$

Здесь i_1, i_2, ..., i_m — номера столбцов, соответствующие неизвестным начальным условиям на левом конце. Сами неизвестные обозначены t_1, t_2, ..., t_m. Чтобы найти компоненты матрицы A и координаты вектора b, нужно решить начальную задачу $m + 1$ раз с такими вариантами неизвестных начальных условий:

- $t = \{1, 0, ..., 0\}$ — для формирования $\{a_{1i_1} + b_1,\ a_{2i_1} + b_2,\ \ldots,\ a_{mi_1} + b_m\}$;
- $t = \{0, 1, ..., 0\}$ — для формирования $\{a_{1i_2} + b_1,\ a_{2i_2} + b_2,\ \ldots,\ a_{mi_2} + b_m\}$;
- ...
- $t = \{0, 0, ..., 1\}$ — для формирования $\{a_{1i_m} + b_1,\ a_{2i_m} + b_2,\ \ldots,\ a_{mi_m} + b_m\}$;

а также

- $t = \{0, 0, ..., 0\}$ — для формирования $\{b_1,\ b_2,\ \ldots,\ b_m\}$.

После этого, решая *линейную* систему (15.4), найдем недостающие начальные условия.

Для нахождения столбцов матрицы A и координаты вектора b нам пришлось несколько раз решить систему линейных дифференциальных уравнений (15.2) при различных вариантах начальных условий (на жаргоне программистов и

расчетчиков это называется "прогнать систему"). Поэтому метод начальных параметров иногда называют также методом *матричной прогонки*.

15.2. Вопросы для самопроверки

1. Какие дифференциальные уравнения можно решать методом начальных параметров: обыкновенные или в частных производных?
2. Как свести дифференциальное уравнение n-го порядка к нормальной системе уравнений? Каким должно быть это уравнение?
3. Как свести систему дифференциальных уравнений n-го порядка к нормальной системе уравнений? Какой должна быть эта система?
4. Что нужно задать для работы решателей нормальных систем дифференциальных уравнений?
5. Как находятся недостающие начальные условия в методе начальных параметров?
6. Почему задача упрощается, если система дифференциальных уравнений — линейная?
7. Откуда появились названия "метод начальных параметров", "метод стрельбы", "метод матричной прогонки"?

15.3. Примеры выполнения заданий

15.3.1. Задание 1

Решить методом начальных параметров *задание 1 главы 2*. Сравнить решение с аналитическим. Построить графики.

Используем программу *задания 1 главы 2* для составления этой программы. Вводим исходные данные. Вначале для сравнения решаем задачу аналитически: повторяем решение *задания 1 главы 2* и заполняем таблицу.

```
clear all % очистили память
disp('Решаем задание 15.1')
syms x y Dy D2y
nnp=10; % число интервалов интегрирования
F=x^2+y^2+Dy^2; % подынтегральная функция
x1=-1; % вводим граничные условия
y1=1;
x2=1;
y2=2;
```

```
disp('Исходные данные:')
fprintf('Подынтегральная функция F(x,y,y'')=%s\n',char(F))
fprintf('Граничное условие слева: y(%d)=%d\n',x1,y1)
fprintf('Граничное условие справа: y(%d)=%d\n',x2,y2)
dFdy=diff(F,y); % вычислили Fy
dFdy1=diff(F,Dy); % вычислили Fy'
d_dFdy1_dx=diff(dFdy1,x); % Fy'x
d_dFdy1_dy=diff(dFdy1,y); % Fy'y
d_dFdy1_dy1=diff(dFdy1,Dy); % Fy'y'-условие Лежандра
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+d_dFdy1_dy1*D2y;
Euler=simple(dFdy-dFy1dx); % уравнение Эйлера
deqEuler=[char(Euler) '=0']; % добавили =0
Sol=dsolve(deqEuler,'x'); % решаем уравнение Эйлера
if length(Sol)~=1, % решений нет или более одного
  error('Нет решений или более одного решения!');
end
SolLeft=subs(Sol,x,x1); % подставили x1
SolRight=subs(Sol,x,x2); % подставили x2
EqLeft=[char(SolLeft) '=' char(sym(y1))]; % =y1
EqRight=[char(SolRight) '=' char(sym(y2))]; % =y2
Con=solve(EqLeft,EqRight,'C1,C2'); % решаем систему
C1=Con.C1; % присваиваем полученные решения
C2=Con.C2; % символическим константам C1 и C2
Sol21=vpa(eval(Sol),14); % подставили C1, C2
xpl=linspace(x1,x2); % задаем массив абсцисс
y21=subs(Sol21,x,xpl); % вычислили ординаты
Решаем задание 15.1
Исходные данные:
Подынтегральная функция F(x,y,y')=x^2+y^2+Dy^2
Граничное условие слева: y(-1)=1
Граничное условие справа: y(1)=2
```

Для применения решателей систем дифференциальных уравнений приводим уравнение Эйлера 2-го порядка к системе 2-х дифференциальных уравнений 1-го порядка путем замены

$$\begin{cases} y_1 = y; \\ y_2 = y'. \end{cases} \tag{15.5}$$

Для этого решаем дифференциальное уравнение Эйлера относительно y'' и формируем правые части системы дифференциальных уравнений. Записываем их в файл.

```
f2=solve(deqEuler,'D2y'); % решаем относительно y''
f21=subs(f2,{y,Dy},{sym('y(1)'),sym('y(2)')});
```

```
s{1}='function dydx = MyRightPart(x,y)';
s{2}='dydx=zeros(2,1);';
s{3}='dydx(1)=y(2);';
s{4}=['dydx(2)=' char(f21) ';'];
filename=fullfile(pwd,'MyRightPart.m'); % файл
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:})
fid=fopen(filename,'w'); % открываем файл
fprintf(fid,'%s\n',s{:}); % записываем файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\ MyRightPart.m:
function dydx = MyRightPart(x,y)
dydx=zeros(2,1);
dydx(1)=y(2);
dydx(2)=(y(1));
```

Мы сформировали систему дифференциальных уравнений

$$\begin{cases} y_1' = y_2; \\ y_2' = y_1 \end{cases} \tag{15.6}$$

с граничными условиями:

$$\begin{cases} y_1(x_1) = 1; \\ y_1(x_2) = 2, \end{cases} \tag{15.7}$$

где в нашем случае $x_1 = -1$; $x_2 = 1$. Если бы было известно начальное условие $y_2(x_1)$, то эту систему можно было бы решить с помощью стандартных решателей. Обозначим $y_2(x_1)$ как неизвестную величину: $t = y_2(x_1)$. Присвоив ей какое-либо пробное значение, можно решить систему дифференциальных уравнений и найти функцию $f = y_1(x_2) - 2$. Очевидно, f можно рассматривать как функцию от t, т. е. нужно решить уравнение $f(t) = 0$. В нашем случае, когда исходная система дифференциальных уравнений *линейная*, уравнение относительно t также будет *линейным*, т. е. функция $f(t)$ имеет структуру $f(t) = at + b$. Чтобы построить эту функцию, нужно решить 2 начальные задачи: для $t = 0$ и $t = 1$. Решаем их.

```
xr=linspace(x1,x2,nnp+1); % готовим численное решение
y0=[y1,0]; % решаем СДУ при y'(x1)=0;
[xx,YY]=ode45('MyRightPart',xr,y0);
yend0=YY(nnp+1,1)-y2;
fprintf('При y''(x1)=0: y(x2)=%12.6f\n',yend0)
y0=[y1,1]; % решаем СДУ при y'(x1)=1;
[xx,YY]=ode45('MyRightPart',xr,y0);
```

```
yend1=YY(nnp+1,1)-y2;
fprintf('При y''(x1)=1: y(x2)=%12.6f\n',yend1)
При y'(x1)=0: y(x2)=     1.762196
При y'(x1)=1: y(x2)=     5.389057
```

Система линейных уравнений (15.4) в данном случае состоит из одного уравнения. Для нахождения неизвестного $t = y_2(x_1)$ проводим линейную интерполяцию.

```
y0=[y1,yend0/(yend0-yend1)];
disp('Начальные условия:')
fprintf('y(x1)=%12.6f\ny''(x1)=%12.6f\n',y0)
Начальные условия:
y(x1)=     1.000000
y'(x1)=   -0.485874
```

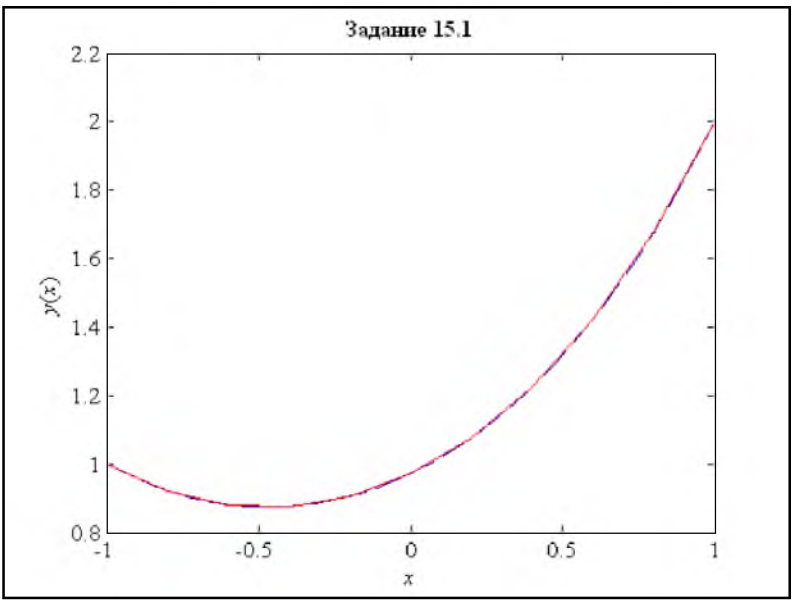


Рис. 15.2. Решение задания 15.1, выполненное с помощью MATLAB

Решаем систему дифференциальных уравнений при найденных действительных начальных условиях. Строим графики (рис. 15.2): сплошной красной линией — численного решения, а штриховой синей — аналитического.

```
[xx,YY]=ode45('MyRightPart',xr,y0); % решаем
figure
plot(xpl,y21,'--b', xr,YY(:,1),'-r') % рисуем
```

```
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',12)
title('\bfЗадание 15.1') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
delete(filename) % удалили ненужный файл
```

Можно ли обойтись без решения системы дифференциальных уравнений (первый оператор в этом фрагменте программы)? Можно, если воспользоваться полученными ранее решениями при $t = 0$ и $t = 1$. Нужно просто взять их линейную комбинацию с подходящими коэффициентами. Найдите это решение самостоятельно.

15.3.2. Задание 2

Решить методом начальных параметров *задание главы 3*. Сравнить решение с аналитическим. Построить графики.

Программу для этого задания напишем на основе программы для *задания главы 3* с использованием программы для *задания 15.1*. Находим аналитическое решение *задания главы 3*. Заполняем таблицу.

```
clear all % очистили память
disp('Решаем задание 15.2')
syms x y z Dy D2y Dz D2z % описали переменные
nnp=10; % число интервалов интегрирования
F=Dy^2+Dz^2+2*y*z; % подынтегральная функция
x1=-2; % вводим граничные условия
y1=1;
z1=0;
x2=2;
y2=0;
z2=2;
disp('Исходные данные:')
fprintf('Подынтегральная функция F(x,y,y'',z,z'')=%s\n',char(F))
disp('Граничные условия слева:')
fprintf('y(%d)=%d;   z(%d)=%d\n',x1,y1,x1,z1)
disp('Граничные условия справа:')
fprintf('y(%d)=%d;   z(%d)=%d\n',x2,y2,x2,z2)
dFdy=diff(F,y);
dFdy1=diff(F,Dy);
d_dFdy1_dx=diff(dFdy1,x);
d_dFdy1_dy=diff(dFdy1,y);
```

```
d_dFdy1_dy1=diff(dFdy1,Dy);
d_dFdy1_dz=diff(dFdy1,z);
d_dFdy1_dz1=diff(dFdy1,Dz);
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+...
  d_dFdy1_dy1*D2y+d_dFdy1_dz*Dz+d_dFdy1_dz1*D2z;
dFdz=diff(F,z);
dFdz1=diff(F,Dz);
d_dFdz1_dx=diff(dFdz1,x);
d_dFdz1_dy=diff(dFdz1,y);
d_dFdz1_dy1=diff(dFdz1,Dy);
d_dFdz1_dz=diff(dFdz1,z);
d_dFdz1_dz1=diff(dFdz1,Dz);
dFz1dx=d_dFdz1_dx+d_dFdz1_dy*Dy+...
  d_dFdz1_dy1*D2y+d_dFdz1_dz*Dz+d_dFdz1_dz1*D2z;
EulerY=simple(dFdy-dFy1dx);
EulerZ=simple(dFdz-dFz1dx);
dEuY=[char(EulerY) '=0']; % уравнение Y
dEuZ=[char(EulerZ) '=0']; % уравнение Z
Sol=dsolve(dEuY,dEuZ,'x'); % решаем
if length(Sol)~=1 % нет решений или более одного решения
  error('Нет решений или более одного решения!');
else
  SolY=Sol.y;
  SolZ=Sol.z;
end
SolLY=subs(SolY,x,x1); % x1 в y
SolLZ=subs(SolZ,x,x1); % x1 в z
SolRY=subs(SolY,x,x2); % x2 в y
SolRZ=subs(SolZ,x,x2); % x2 в z
EqLY=[char(vpa(SolLY,14)) '=' char(sym(y1))];
EqLZ=[char(vpa(SolLZ,14)) '=' char(sym(z1))];
EqRY=[char(vpa(SolRY,14)) '=' char(sym(y2))];
EqRZ=[char(vpa(SolRZ,14)) '=' char(sym(z2))];
Con=solve(EqLY,EqLZ,EqRY,EqRZ,'C1,C2,C3,C4');
C1=Con.C1;
C2=Con.C2;
C3=Con.C3;
C4=Con.C4;
Sol3Y=vpa(eval(SolY),14);
Sol3Z=vpa(eval(SolZ),14);
xpl=linspace(x1,x2); % массив абсцисс
y3=subs(Sol3Y,x,xpl); % ординаты
z3=subs(Sol3Z,x,xpl); % аппликаты
```

```
Решаем задание 15.2
Исходные данные:
Подынтегральная функция F(x,y,y',z,z')=Dy^2+Dz^2+2*y*z
Граничные условия слева:
y(-2)=1;   z(-2)=0
Граничные условия справа:
y(2)=0;    z(2)=2
```

Сведем систему двух дифференциальных уравнений Эйлера 2-го порядка к системе четырех нормальных дифференциальных уравнений 1-го порядка вида (15.2). Для этого решим уравнения Эйлера относительно y'', z''. Подставим в оба уравнения y, z, y', z'. Сформируем правые части для системы дифференциальных уравнений и запишем их в файл.

```
f2yz=solve(dEuY,dEuZ,'D2y,D2z');
f2=subs(f2yz.D2y,{y,Dy,z,Dz},...
  {sym('y(1)'),sym('y(2)'),sym('y(3)'),sym('y(4)')});
f4=subs(f2yz.D2z,{y,Dy,z,Dz},...
  {sym('y(1)'),sym('y(2)'),sym('y(3)'),sym('y(4)')});
s{1}='function dydx = MyRightPart(x,y)';
s{2}='dydx=zeros(4,1);';
s{3}='dydx(1)=y(2);';
s{4}=['dydx(2)=' char(f2) ';'];
s{5}='dydx(3)=y(4);';
s{6}=['dydx(4)=' char(f4) ';'];
filename=fullfile(pwd,'MyRightPart.m');
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:})
fid=fopen(filename,'w');
fprintf(fid,'%s\n',s{:});
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\MyRightPart.m:
function dydx = MyRightPart(x,y)
dydx=zeros(4,1);
dydx(1)=y(2);
dydx(2)=(y(3));
dydx(3)=y(4);
dydx(4)=(y(1));
```

Применим метод начальных параметров для решения задачи. Неизвестные у нас обозначены

$$y_1 = y;\quad y_2 = y';\quad y_3 = z;\quad y_4 = z'. \tag{15.8}$$

В начальной точке x_1 неизвестны $y_2(x_1)=t_1$ и $y_4(x_1)=t_2$. Найдем их из решения системы двух уравнений

$$\begin{cases} f_1(t_1,t_2)=y(x_2)-y_2=0; \\ f_2(t_1,t_2)=z(x_2)-z_2=0. \end{cases} \tag{15.9}$$

Система дифференциальных уравнений Эйлера у нас является линейной, поэтому и система уравнений (15.9) также будет линейной:

$$\begin{cases} f_1(t_1,t_2)=a_{11}t_1+a_{12}t_2+b_1=0; \\ f_2(t_1,t_2)=a_{21}t_1+a_{22}t_2+b_2=0. \end{cases} \tag{15.10}$$

Найдем коэффициенты этой системы и правые части.

```
xr=linspace(x1,x2,nnp+1); % точки
A=zeros(2,2); % матрица системы уравнений
b=zeros(2,1); % вектор правых частей
y0=[y1;0;z1;0]; % решаем СДУ при y'(x1)=0; z'(x1)=0;
[xx,YY]=ode45('MyRightPart',xr,y0);
b=YY(nnp+1,[1 3])'-[y2;z2];
y0=[y1;1;z1;0]; % решаем СДУ при y'(x1)=1; z'(x1)=0;
[xx,YY]=ode45('MyRightPart',xr,y0);
A(:,1)=YY(nnp+1,[1 3])'-[y2;z2]-b;
y0=[y1;0;z1;1]; % решаем СДУ при y'(x1)=0; z'(x1)=1;
[xx,YY]=ode45('MyRightPart',xr,y0);
A(:,2)=YY(nnp+1,[1 3])'-[y2;z2]-b;
disp('Получена система уравнений:')
fprintf('%12.6f *y''(x1)%+12.6f *z''(x1)=%12.6f\n',[A,b]')
```

```
Получена система уравнений:
   13.266624 *y'(x1)  +14.023425 *z'(x1)=   13.327366
   14.023425 *y'(x1)  +13.266624 *z'(x1)=   11.980999
```

Решаем систему линейных уравнений (15.10), находим недостающие начальные условия. Печатаем их.

```
yz0=-A\b; % нашли начальные параметры
y0=[y1;yz0(1);z1;yz0(2)]; % начальные условия
disp('Начальные условия:')
fprintf('y(x1) =%12.6f\ny''(x1)=%12.6f\n',y0(1:2))
fprintf('z(x1) =%12.6f\nz''(x1)=%12.6f\n',y0(3:4))
```

```
Начальные условия:
y(x1) =    1.000000
y'(x1)=    0.425820
z(x1) =    0.000000
z'(x1)=   -1.353205
```

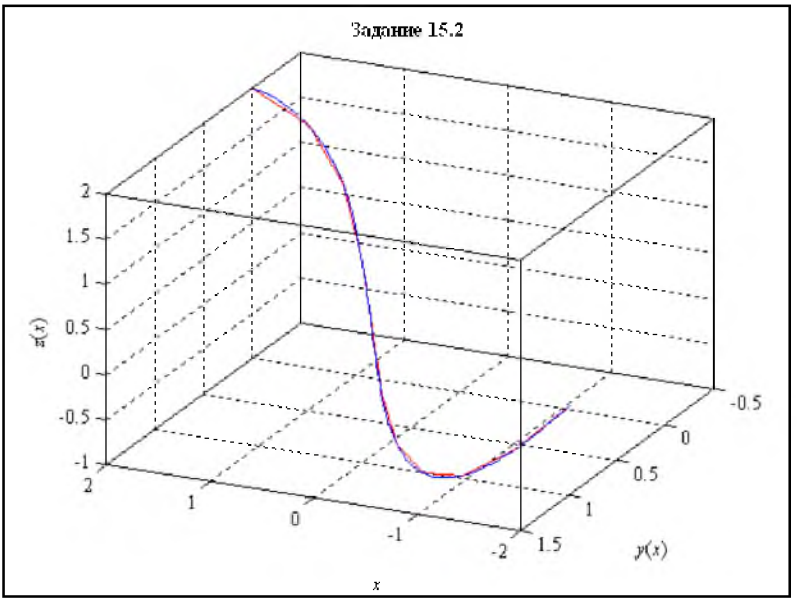


Рис. 15.3. Решение задания 15.2

Для этих начальных условий решаем систему дифференциальных уравнений вида (15.2). Строим график полученного решения (рис. 15.3). Для сравнения строим на этом рисунке и график аналитического решения. Выбираем точку просмотра. Показываем масштабную сетку и ограничивающий контур. Удаляем ненужный файл с функцией вычисления правых частей системы дифференциальных уравнений.

```
[xx,YY]=ode45('MyRightPart',xr,y0); % решаем
figure % фигура
plot3(xpl,y3,z3,'--b',xr,YY(:,1),YY(:,3),'-r') % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 15.2') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
zlabel('\itz\rm(\itx\rm)') % метка оси OZ
view(205,30) % точка просмотра
grid on % показали сетку
box on % показали внешний контур
delete(filename) % удалили файл
```

Вместо решения системы дифференциальных уравнений при истинных значениях начальных условий (первый оператор в этом фрагменте программы)

можно взять линейную комбинацию найденных ранее решений. Напишите этот фрагмент самостоятельно.

15.3.3. Задание 3

Решить методом начальных параметров *задание главы 4*. Сравнить решение с аналитическим. Построить графики.

Уравнение Эйлера — Пуассона (4.8) является уравнением 4-го порядка с 2-мя граничными условиями на левом конце и 2-мя на правом. Сведем его к нормальной системе 4-х уравнений 1-го порядка заменой:

$$\begin{cases} y_1 = y; \\ y_2 = y'; \\ y_3 = y''; \\ y_4 = y'''; \end{cases} \quad \begin{cases} y_1(x_1) = y_{10}; \\ y_2(x_1) = y'_{10}; \\ y_1(x_2) = y_{20}; \\ y_2(x_2) = y'_{20}. \end{cases} \tag{15.11}$$

Неизвестные начальные условия $t_1 = y_3(x_1)$ и $t_2 = y_4(x_1)$ найдем из решения системы уравнений:

$$\begin{cases} f_1(t_1, t_2) = y_1(x_2) - y_{20} = 0; \\ f_2(t_1, t_2) = y_2(x_2) - y'_{20} = 0. \end{cases} \tag{15.12}$$

Так как уравнение (4.8) линейное, то и система (15.12) будет линейной. Решая ее, найдем начальные условия, а затем и решение уравнения Эйлера.

Для составления программы используем программы для *заданий главы 4 и раздела 15.2*. Вначале повторяем решение примера *главы 4*.

```
clear all % очистили память
disp('Решаем задание 15.3')
syms x y Dy D2y D3y D4y % описали переменные
nnp=10; % число интервалов интегрирования
F=D2y^2-2*Dy^2+4*y*Dy+y^2-2*y*sin(x);
x1=-1;
y1=1;
Dy1=-1;
x2=1;
y2=2;
Dy2=-1;
disp('Исходные данные:')
disp('Подынтегральная функция:')
fprintf('F(x,y,y'',y'''')=%s\n',char(F))
```

```
disp('Граничные условия слева:')
fprintf('y(%d)=%d;   y''(%d)=%d\n',x1,y1,x1,Dy1)
disp('Граничные условия справа:')
fprintf('y(%d)=%d;   y''(%d)=%d\n',x2,y2,x2,Dy2)
dFdy=diff(F,y);
dFdy1=diff(F,Dy);
dFdy2=diff(F,D2y);
d_dFdy1_dx=diff(dFdy1,x);
d_dFdy1_dy=diff(dFdy1,y);
d_dFdy1_dy1=diff(dFdy1,Dy);
d_dFdy1_dy2=diff(dFdy1,D2y);
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+d_dFdy1_dy1*D2y+d_dFdy1_dy2*D3y;
d_dFdy2_dx=diff(dFdy2,x);
d_dFdy2_dy=diff(dFdy2,y);
d_dFdy2_dy1=diff(dFdy2,Dy);
d_dFdy2_dy2=diff(dFdy2,D2y);
dFy2dx=d_dFdy2_dx+d_dFdy2_dy*Dy+d_dFdy2_dy1*D2y+d_dFdy2_dy2*D3y;
d_dFdy2dx_dx=diff(dFy2dx,x);
d_dFdy2dx_dy=diff(dFy2dx,y);
d_dFdy2dx_dy1=diff(dFy2dx,Dy); % d((dFy'')/dx)/dy'
d_dFdy2dx_dy2=diff(dFy2dx,D2y); % d((dFy'')/dx)/dy''
d_dFdy2dx_dy3=diff(dFy2dx,D3y); % d((dFy'')/dx)/dy'''
d2Fy2dx2=d_dFdy2dx_dx+d_dFdy2dx_dy*Dy+d_dFdy2dx_dy1*D2y;
d2Fy2dx2=d2Fy2dx2+d_dFdy2dx_dy2*D3y+d_dFdy2dx_dy3*D4y;
Euler=simple(dFdy-dFy1dx+d2Fy2dx2);
deqEuler=[char(Euler) '=0']; % составили уравнение
Sol=dsolve(deqEuler,'x'); % решаем уравнение
if length(Sol)~=1 % нет решений или более одного решения
  error('Нет решений или более одного решения!');
end
dydx=diff(Sol,x); % нашли производную
slY=subs(Sol,x,x1); % подставили x1 в y(x)
slDY=subs(dydx,x,x1); % подставили x1 в y'(x)
srY=subs(Sol,x,x2); % подставили x2 в y(x)
srDY=subs(dydx,x,x2); % подставили x2 в y'(x)
elY=[char(vpa(slY,14)) '=' char(sym(y1))];
elDY=[char(vpa(slDY,14)) '=' char(sym(Dy1))];
erY=[char(vpa(srY,14)) '=' char(sym(y2))];
erDY=[char(vpa(srDY,14)) '=' char(sym(Dy2))];
Con=solve(elY,elDY,erY,erDY,'C1,C2,C3,C4');
C1=Con.C1;
C2=Con.C2;
C3=Con.C3;
C4=Con.C4;
```

```
Sol4=vpa(eval(Sol),14); % подставили C1-C4
xpl=linspace(x1,x2); % массив аргументов
y4=subs(Sol4,x,xpl); % вычислили функцию
Решаем задание 15.3
Исходные данные:
Подынтегральная функция:
F(x,y,y',y'')=D2y^2-2*Dy^2+4*y*Dy+y^2-2*y*sin(x)
Граничные условия слева:
y(-1)=1;   y'(-1)=-1
Граничные условия справа:
y(1)=2;   y'(1)=-1
```

Разрешаем уравнение Эйлера относительно y^{IV} и подставляем в него обозначения (15.11). Формируем правые части для системы дифференциальных уравнений, к которой сводится уравнение Эйлера. Записываем их в файл.

```
f4=solve(deqEuler,'D4y'); % находим D4y
f41=subs(f4,{y,Dy,D2y,D3y},...
  {sym('y(1)'),sym('y(2)'),sym('y(3)'),sym('y(4)')});
s{1}='function dydx = MyRightPart(x,y)';
s{2}='dydx=zeros(4,1);';
s{3}='dydx(1)=y(2);';
s{4}='dydx(2)=y(3);';
s{5}='dydx(3)=y(4);';
s{6}=['dydx(4)=' char(f41) ';'];
filename=fullfile(pwd,'MyRightPart.m');
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:})
fid=fopen(filename,'w');
fprintf(fid,'%s\n',s{:});
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\MyRightPart.m:
function dydx = MyRightPart(x,y)
dydx=zeros(4,1);
dydx(1)=y(2);
dydx(2)=y(3);
dydx(3)=y(4);
dydx(4)=-y(1)+sin(x)-2*y(3);
```

Формируем коэффициенты и свободные члены системы линейных уравнений (15.4, 15.12). Для этого 3 раза решаем начальную задачу при значениях неизвестных начальных параметров: $\{t_1, t_2\} = \{0, 0\}$; $\{t_1, t_2\} = \{1, 0\}$; и $\{t_1, t_2\} = \{0, 1\}$. Решаем систему (15.12) — находим неизвестные начальные параметры. Решаем начальную задачу при этих значениях начальных параметров.

Рисуем график численного решения и, для сравнения, аналитического. Он показан на рис. 15.4.

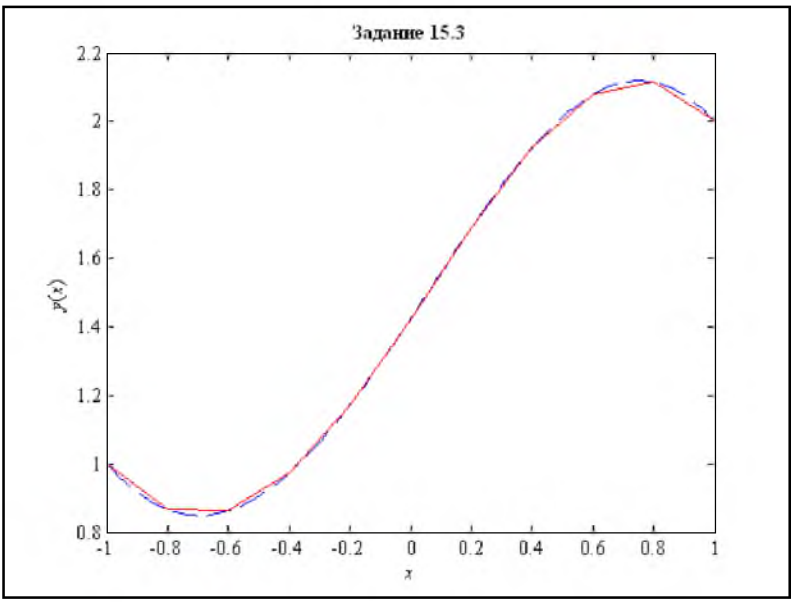


Рис. 15.4. Решение задания 15.3

```
xr=linspace(x1,x2,nnp+1); % точки
A=zeros(2,2);
b=zeros(2,1);
y0=[y1;Dy1;0;0]; % начальные условия (0,0)
[xx,YY]=ode45('MyRightPart',xr,y0); % решаем СДУ
b=YY(nnp+1,[1 2])'-[y2;Dy2]; % правые части
y0=[y1;Dy1;1;0]; % начальные условия (1,0)
[xx,YY]=ode45('MyRightPart',xr,y0); % решаем СДУ
A(:,1)=YY(nnp+1,[1 2])'-[y2;Dy2]-b; % 1-й столбец A
y0=[y1;Dy1;0;1]; % начальные условия (0,1)
[xx,YY]=ode45('MyRightPart',xr,y0); % решаем СДУ
A(:,2)=YY(nnp+1,[1 2])'-[y2;Dy2]-b; % 2-й столбец A
disp('Получена система уравнений:')
fprintf('%12.6f *y''''(x1)%+12.6f *y''''''(x1)=%12.6f\n',[A,b]')
yz0=-A\b; % нашли начальные параметры
y0=[y1;Dy1;yz0]; % истинные начальные условия
disp('Начальные условия:')
fprintf('y(x1)=%12.6f\ny''(x1)=%12.6f\n',y0(1:2))
fprintf('y''''(x1)=%12.6f\ny''''''(x1)=%12.6f\n',y0(3:4))
```

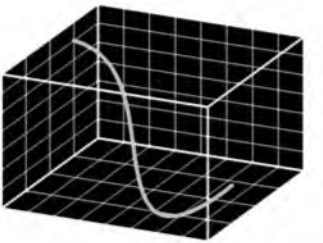
```
[xx,YY]=ode45('MyRightPart',xr,y0); % решаем
figure % фигура
plot(xpl,y4,'--b',xr,YY(:,1),'-r') % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 15.3') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
delete(filename) % удалили файл
Получена система уравнений:
    0.909297 *y''(x1)   +0.870796 *y'''(x1)=   -3.546096
    0.038501 *y''(x1)   +0.909297 *y'''(x1)=   -0.694461
Начальные условия:
y(x1)=    1.000000
y'(x1)=   -1.000000
y''(x1)=    3.302330
y'''(x1)=    0.623907
```

Как и в предыдущих заданиях, решение системы дифференциальных уравнений при найденных начальных условиях можно заменить линейной комбинацией решений при пробных значениях начальных условий: $\{t_1, t_2\} = \{0, 0\}$; $\{t_1, t_2\} = \{1, 0\}$ и $\{t_1, t_2\} = \{0, 1\}$. Запрограммируйте сами этот вариант решения.

15.4. Задание

1. Решить *задание 1 главы 2* методом начальных параметров. Сравнить решение с аналитическим.
2. Решить *задание главы 3* методом начальных параметров. Сравнить решение с аналитическим.
3. Решить *задание главы 4* методом начальных параметров. Сравнить решение с аналитическим.

ГЛАВА 16



Метод конечных разностей (МКР)

16.1. МКР для вариационных задач с обыкновенными дифференциальными уравнениями

Изучим еще один численный метод решения задач вариационного исчисления — метод конечных разностей (МКР). Вначале рассмотрим применение этого метода к одномерным задачам: когда искомая функция (функции) зависят только от одной переменной. Такими были задачи, рассмотренные в *главах 2*, *3*, *4*: соответствующие дифференциальные уравнения Эйлера были обыкновенными. Применение МКР к дифференциальным уравнениям в частных производных будет рассмотрено в следующих разделах.

Основная идея МКР — замена производных отношением малых конечных приращений функции и аргумента:

$$y' = \frac{dy}{dx} \approx \frac{\Delta y}{\Delta x}. \qquad (16.1)$$

Фактически это означает, что в МКР экстремаль аппроксимируется кусочно-линейной функцией. Варьируются ординаты $y(x)$ в заданных точках, и функционал становится функцией этих неизвестных ординат $y_1, y_2, \ldots$.

Рассмотрим применение МКР к элементарной задаче вариационного исчисления для функционала (2.1—2.2). Разобьем интервал $[x_1, x_2]$ на n участков одинаковой длины $\Delta x = (x_2 - x_1)/n$, как показано на рис. 16.1.

Будем обозначать точки деления x_k. То есть граничные точки в этом разделе в дальнейшем будут обозначаться не x_1 и x_2, а x_0 и x_n. Функционал J равен сумме функционалов на отдельных участках J_k:

$$J = \sum_{k=1}^{n} J_k = \sum_{k=1}^{n} \int_{x_{k-1}}^{x_k} F(x, y, y')\, dx\,. \tag{16.2}$$

Так как участки малые, применим для вычисления интегралов формулу касательных, согласно которой каждый из J_k равен подынтегральной функции, вычисленной в средней точке и умноженной на ширину интервала:

$$J_k = \int_{x_{k-1}}^{x_k} F(x, y, y')\, dx \approx F\left(\frac{x_{k-1} + x_k}{2}, \frac{y_{k-1} + y_k}{2}, \frac{y_k - y_{k-1}}{\Delta x}\right) \Delta x\,. \tag{16.3}$$

Функционал (16.2) будет теперь функцией неизвестных ординат $y_1, y_2, ..., y_{n-1}$:

$$J(y_1, y_2, ..., y_{n-1}) = \sum_{k=1}^{n} J_k \approx \Delta x \sum_{k=1}^{n} F\left(\frac{x_{k-1} + x_k}{2}, \frac{y_{k-1} + y_k}{2}, \frac{y_k - y_{k-1}}{\Delta x}\right). \tag{16.4}$$

Значения y_0 и y_n известны — это заданные граничные условия. Для нахождения экстремума продифференцируем J по переменным $y_1, y_2, ..., y_{n-1}$, приравняем производные нулю и решим полученную систему уравнений.

При вычислении $\partial J/\partial y_k$ учтем, что от y_k зависят только 2 слагаемых формулы (16.4): J_k и J_{k+1}. В оба слагаемых наша переменная y_k входит во 2-й и 3-й аргументы. Применяем обычное правило дифференцирования сложной функции. Находим вначале $\partial J_k/\partial y_k$:

$$\begin{aligned}\frac{\partial J_k}{\partial y_k} = {} & F_y\left(\frac{x_{k-1} + x_k}{2}, \frac{y_{k-1} + y_k}{2}, \frac{y_k - y_{k-1}}{\Delta x}\right)\frac{\Delta x}{2} + \\ & + F_{y'}\left(\frac{x_{k-1} + x_k}{2}, \frac{y_{k-1} + y_k}{2}, \frac{y_k - y_{k-1}}{\Delta x}\right).\end{aligned} \tag{16.5}$$

Теперь вычисляем $\partial J_{k+1}/\partial y_k$:

$$\begin{aligned}\frac{\partial J_{k+1}}{\partial y_k} = {} & F_y\left(\frac{x_k + x_{k+1}}{2}, \frac{y_k + y_{k+1}}{2}, \frac{y_{k+1} - y_k}{\Delta x}\right)\frac{\Delta x}{2} - \\ & - F_{y'}\left(\frac{x_k + x_{k+1}}{2}, \frac{y_k + y_{k+1}}{2}, \frac{y_{k+1} - y_k}{\Delta x}\right).\end{aligned} \tag{16.6}$$

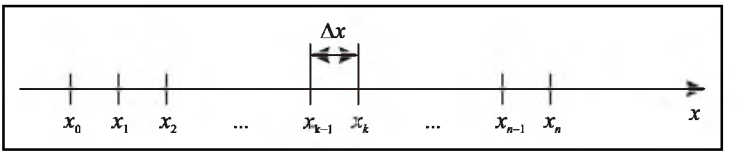


Рис. 16.1. Разбиение интервала интегрирования на участки

Складываем и сокращаем на Δx. Получаем систему уравнений, которая своей структурой напоминает уравнение Эйлера (2.9):

$$\begin{cases} \frac{1}{2}\left(F_y\left(\frac{x_{k-1}+x_k}{2}, \frac{y_{k-1}+y_k}{2}, \frac{y_k-y_{k-1}}{\Delta x} \right) + \right. \\ \left. + F_y\left(\frac{x_k+x_{k+1}}{2}, \frac{y_k+y_{k+1}}{2}, \frac{y_{k+1}-y_k}{\Delta x} \right) \right) - \\ -\frac{1}{\Delta x}\left(F_{y'}\left(\frac{x_k+x_{k+1}}{2}, \frac{y_k+y_{k+1}}{2}, \frac{y_{k+1}-y_k}{\Delta x} \right) - \right. \\ \left. - F_{y'}\left(\frac{x_{k-1}+x_k}{2}, \frac{y_{k-1}+y_k}{2}, \frac{y_k-y_{k-1}}{\Delta x} \right) \right) = 0; \\ k \in [1, n-1]. \end{cases} \tag{16.7}$$

Вместо F_y, которая фигурирует в уравнении Эйлера (2.9), в уравнении (16.7) — полусумма значений F_y на интервалах, примыкающих к точке x_k. Вместо dF_y/dx — отношение разности величин F_y справа и слева от точки x_k к ширине интервала Δx. Таким образом, (16.7) можно рассматривать как конечно-разностный аналог уравнения Эйлера (2.9).

Если, как в наших примерах, подынтегральная функция $F(x, y, y')$ является квадратичной относительно аргументов y и y', то ее частные производные по этим аргументам будут линейными и система уравнений (16.7) становится линейной.

Заметим, что не обязательно выводить систему уравнений МКР из условия экстремума функционала. Можно непосредственно подставить в уравнение Эйлера конечно-разностные аппроксимации производных: результат будет тем же.

16.2. МКР для вариационной задачи в частных производных: прямоугольная сетка

В этом случае для аппроксимации частных производных в МКР применяются различные схемы (в теории МКР они называются шаблонами). Наиболее простым и понятным является шаблон, показанный на рис. 16.2. В нем для аппроксимации первых частных производных используется "центральная" конечная разность: полусумма "правой" и "левой" разностей:

$$\frac{\partial z}{\partial x} \approx \frac{1}{2}\left(\frac{z_{ik} - z_{i-1,k}}{\Delta x} + \frac{z_{i+1,k} - z_{ik}}{\Delta x}\right) = \frac{z_{i+1,k} - z_{i-1,k}}{2\Delta x};$$
$$\frac{\partial z}{\partial y} \approx \frac{1}{2}\left(\frac{z_{ik} - z_{i,k-1}}{\Delta y} + \frac{z_{i,k+1} - z_{ik}}{\Delta y}\right) = \frac{z_{i,k+1} - z_{i,k-1}}{2\Delta y}; \quad (16.8)$$

а для аппроксимации вторых частных производных — выражения:

$$\frac{\partial^2 z}{\partial x^2} \approx \frac{1}{\Delta x}\left(\frac{z_{i+1,k} - z_{ik}}{\Delta x} - \frac{z_{ik} - z_{i-1,k}}{\Delta x}\right) = \frac{z_{i+1,k} - 2z_{ik} + z_{i-1,k}}{(\Delta x)^2};$$
$$\frac{\partial^2 z}{\partial x \partial y} \approx \frac{1}{2\Delta y}\left(\frac{z_{i+1,k+1} - z_{i-1,k+1}}{2\Delta x} - \frac{z_{i+1,k-1} - z_{i-1,k-1}}{2\Delta x}\right) = \quad (16.9)$$
$$= \frac{z_{i+1,k+1} - z_{i-1,k+1} - z_{i+1,k-1} + z_{i-1,k-1}}{4\Delta x \Delta y};$$
$$\frac{\partial^2 z}{\partial y^2} \approx \frac{1}{\Delta y}\left(\frac{z_{i,k+1} - z_{ik}}{\Delta y} - \frac{z_{ik} - z_{i,k-1}}{\Delta y}\right) = \frac{z_{i,k+1} - 2z_{ik} + z_{i,k-1}}{(\Delta y)^2}.$$

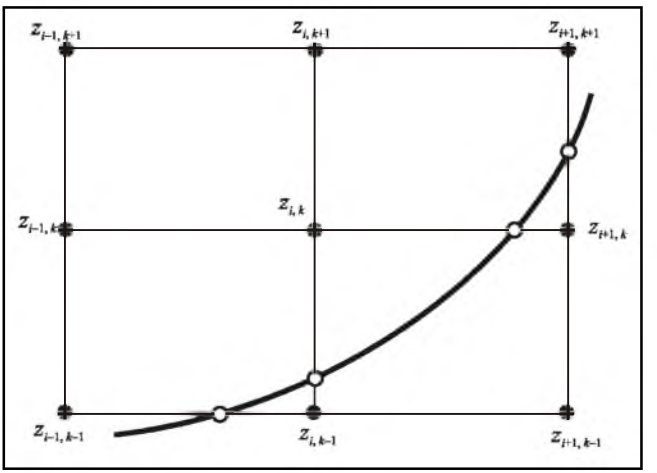


Рис. 16.2. Шаблон для аппроксимации частных производных

Теперь полученные выражения можно подставить в функционал (5.1). Он станет функцией неизвестных z_{ik}. Дифференцируя функционал по этим неизвестным z_{ik} и приравнивая производные нулю, получим систему уравнений для нахождения неизвестных z_{ik}. Проделайте это самостоятельно и убедитесь, что полученные уравнения будут конечно-разностным аналогом уравнения Эйлера — Остроградского (5.8). Поэтому можно поступить проще: выраже-

ния для частных производных (16.8—16.9) подставить не в функционал (5.1), а непосредственно в уравнение Эйлера — Остроградского (5.8). При этом также получится система уравнений для нахождения неизвестных z_{ik}.

Здесь основную трудность представляет учет граничных условий. Ведь линия границы не обязательно проходит через узлы: она может проходить так, как показано на рис. 16.2 жирной линией. Для учета граничных условий применяются различные приемы, описанные, в частности, в [24, 29]. Мы рассмотрим только самый простой прием, который заключается в следующем. Назовем *внутренней* точкой области такой узел, для которого весь его шаблон находится в области. Для этих точек может быть записано уравнение МКР с учетом соотношений (16.8—16.9). Точки, не являющиеся внутренними, но находящиеся в области решения, считаются *граничными*. Например, точка z_{ik} на рис. 16.2 является граничной, т. к. в ее окружении 3 точки не входят в область решения: это $z_{i,k-1}$, $z_{i+1,k-1}$, $z_{i+1,k}$. Эти точки используются для учета граничных условий. В простейшем случае в каждой граничной точке значение искомой функции принимается равным значению функции на границе. Точнее граничные условия можно учесть с помощью линейной интерполяции в граничных точках, показанных на рис. 16.2 незакрашенными кружками.

Более сложные схемы учета граничных условий рассмотрены в [24, 29].

16.3. МКР для вариационной задачи в частных производных: треугольная сетка

В MATLAB есть возможность построить треугольную сетку для любой плоской области. Поэтому рассмотрим применение МКР для такого разбиения. Обозначим узлы одного из треугольников $M_1(x_1, y_1)$, $M_2(x_2, y_2)$, $M_3(x_3, y_3)$. Они перенумерованы против часовой стрелки, как показано на рис. 16.3.

Неизвестными являются аппликаты точек: z_1, z_2, z_3. Чтобы вывести формулы для частных производных, запишем уравнение заштрихованной плоскости, которой аппроксимируется неизвестная функция $z = z(x, y)$:

$$\begin{vmatrix} x - x_1 & y - y_1 & z - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} = 0. \qquad (16.10)$$

Это — известное из курса аналитической геометрии уравнение плоскости, проходящей через 3 точки. Если теперь раскрыть определитель и решить полученное уравнение относительно z, то коэффициенты при x и y будут аппроксимациями для $\partial z/\partial x$ и $\partial z/\partial y$. Раскрываем определитель по элементам 1-й строки:

$$\begin{vmatrix} y_2 - y_1 & z_2 - z_1 \\ y_3 - y_1 & z_3 - z_1 \end{vmatrix} (x - x_1) + \begin{vmatrix} z_2 - z_1 & x_2 - x_1 \\ z_3 - z_1 & x_3 - x_1 \end{vmatrix} (y - y_1) + \\ + \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix} (z - z_1) = 0 . \tag{16.11}$$

Решаем относительно z. Определитель в последнем слагаемом — это удвоенная площадь элемента, поэтому:

$$z = z_1 + \frac{1}{2S} \begin{vmatrix} z_2 - z_1 & y_2 - y_1 \\ z_3 - z_1 & y_3 - y_1 \end{vmatrix} (x - x_1) + \frac{1}{2S} \begin{vmatrix} x_2 - x_1 & z_2 - z_1 \\ x_3 - x_1 & z_3 - z_1 \end{vmatrix} (y - y_1) . \tag{16.12}$$

Функция $z(x, y)$ — линейная. Множители перед скобками — это и есть приближения для нужных нам частных производных. После раскрытия определителей и преобразований они имеют вид:

$$\frac{\partial z}{\partial x} \approx \frac{1}{2S} \left(z_1 (y_2 - y_3) + z_2 (y_3 - y_1) + z_3 (y_1 - y_2) \right); \\ \frac{\partial z}{\partial y} \approx \frac{1}{2S} \left(z_1 (x_3 - x_2) + z_2 (x_1 - x_3) + z_3 (x_2 - x_1) \right). \tag{16.13}$$

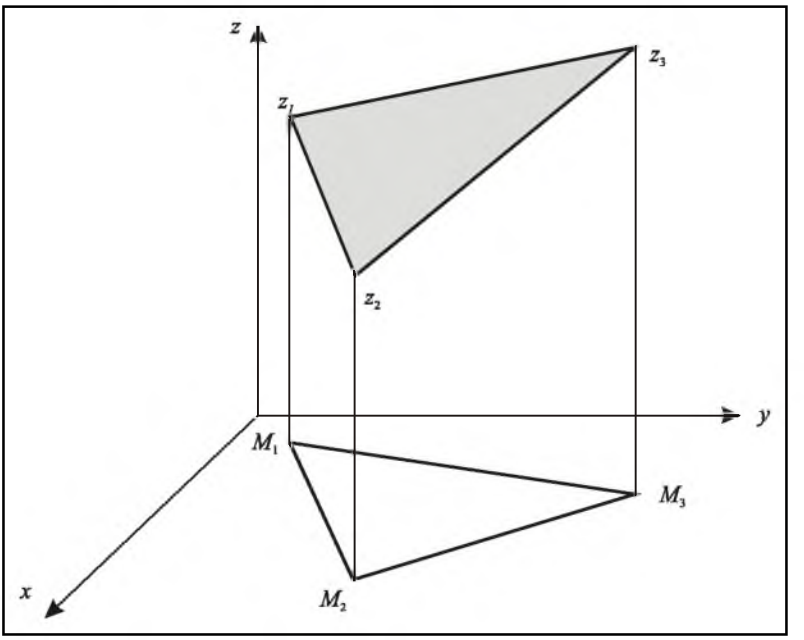


Рис. 16.3. Треугольный элемент

Частные производные являются постоянными величинами во всей треугольной области — линейными комбинациями неизвестных значений в узлах z_1, z_2, z_3.

Для вычисления интеграла (5.1) мы воспользуемся построенной треугольной сеткой. Каждая область — малая, поэтому применим простейшую 1-точечную схему интегрирования по ней: вычислим подынтегральную функцию F в центре тяжести каждого треугольника, умножим на площадь и сложим. Получим:

$$J \approx \sum_{k=1}^{n_e} F\left(\frac{x_{1k}+x_{2k}+x_{3k}}{3}, \frac{y_{1k}+y_{2k}+y_{3k}}{3}, \frac{z_{1k}+z_{2k}+z_{3k}}{3}, \right.$$
$$\frac{1}{2S_k}\left(z_{1k}\left(y_{2k}-y_{3k}\right)+z_{2k}\left(y_{3k}-y_{1k}\right)+z_{3k}\left(y_{1k}-y_{2k}\right)\right), \qquad (16.14)$$
$$\left.\frac{1}{2S_k}\left(z_{1k}\left(x_{3k}-x_{2k}\right)+z_{2k}\left(x_{1k}-x_{3k}\right)+z_{3k}\left(x_{2k}-x_{1k}\right)\right)\right) S_k \ .$$

Здесь n_e — общее число треугольных элементов, второй индекс k — номер элемента.

Мы получили выражение для функционала в виде функции неизвестных значений z в узлах разбиения. Если узлы какого-либо элемента выходят на границу, то соответствующее значение z_{1k}, z_{2k} или z_{3k} известно: оно определяется граничным условием (5.2) в нужной точке.

Дифференцируем выражение (16.14) по каждой из неизвестных z_i, $i \in [1, n_p]$, где n_p — число узлов. Каждая переменная z_i входит не во все n_e слагаемых суммы (16.14), а только в те, элементы которых имеют одним из своих узлов узел номер i. Поэтому при вычислении $\partial J/\partial z_i$ суммирование нужно производить только по тем треугольникам k, которые включают i-й узел. Будем обозначать это так: $\forall k \ni i$. Имеем:

$$\frac{\partial J}{\partial z_i} \approx \sum_{\forall k \ni i} \left(\frac{S_k}{3} F_z(\ldots) + \frac{y_{123k}}{2} F_p(\ldots) + \frac{x_{123k}}{2} F_q(\ldots) \right), \qquad (16.15)$$

где F_z, F_p, F_q — частные производные от функции F по ее 3-, 4- и 5-му аргументам. Многоточия в скобках — те же значения аргументов, что и в формуле (16.14); y_{123k} — величина $(y_{2k}-y_{3k})$, $(y_{3k}-y_{1k})$ или $(y_{1k}-y_{2k})$ в зависимости от того, является ли узел номер i для элемента номер k соответственно первым, вторым или третьим; x_{123k} — величина $(x_{3k}-x_{2k})$, $(x_{1k}-x_{3k})$ или $(x_{2k}-x_{1k})$ в зависимости от того, является ли узел номер i для элемента номер k соответственно первым, вторым или третьим.

Производные (16.15) вычисляются во всех внутренних узлах. Если при этом в аргументах, обозначенных (...), участвуют аппликаты граничных точек, то

им присваиваем значения, определяемые граничными условиями. Таким образом, получаем систему уравнений для неизвестных z_i. Если подынтегральная функция квадратично зависит от z, p, q (именно такие функционалы мы рассматривали в *главе 5*), то система уравнений (16.15) получается линейной:

$$\mathrm{K}z = \mathrm{f}. \tag{16.16}$$

Построение матрицы K и вектора f можно упростить, если заполнять их не строками, а блоками. Для этого вначале K и f обнуляют. Затем просматривают каждый элемент номер k; определяют, какие узлы (k_1, k_2, k_3) ему соответствуют, и добавляют к нужным элементам K и f соответствующие слагаемые из формулы (16.15). Этот процесс показан на рис. 16.4: для каждого k заполняются 9 элементов матрицы K и 3 координаты f, выделенные на рис. 16.4 черным цветом.

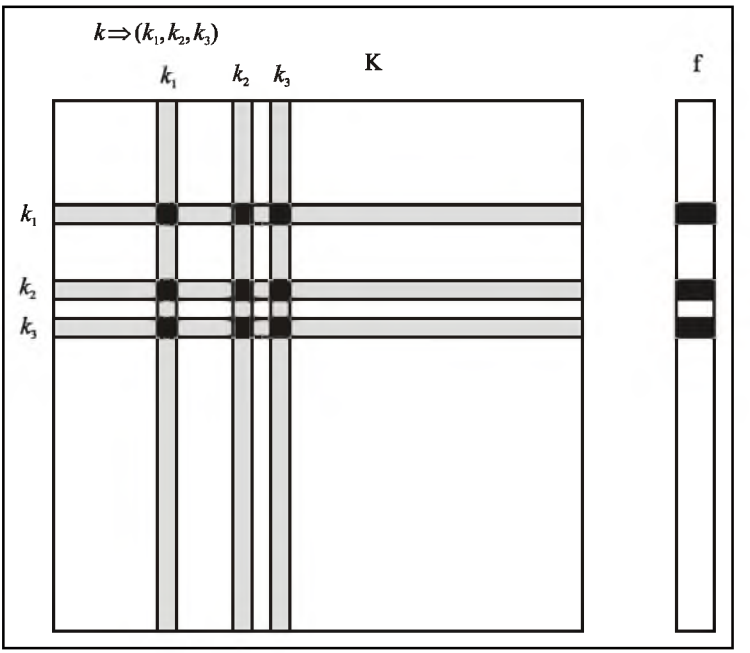


Рис. 16.4. Заполнение матрицы K и вектора f

Другой упрощающий прием связан с учетом граничных условий. Вначале K и f заполняют для всех узлов: и внутренних, и граничных. При этом система (16.16) получается вырожденной. Для устранения вырожденности и учета граничных условий будем искать решение z в виде:

$$z = u + v, \tag{16.17}$$

где u — известный вектор, у которого "граничные" координаты (т. е. координаты, соответствующие узлам на границе) взяты из граничных условий, а "внутренние" равны нулю; v — вектор, у которого "граничные" координаты равны нулю, а "внутренние" неизвестны (подлежат определению). Тогда для v имеем систему уравнений:

$$Kv = f - Ku, \tag{16.18}$$

в которой "граничные" координаты v должны быть равны нулю. Проще всего этого добиться не исключением соответствующих строк и столбцов, а добавлением больших чисел к диагональным элементам матрицы K, соответствующим "граничным" узлам. Тогда при решении системы (16.18) мы автоматически получим нужные координаты близкими к нулю.

16.4. Вопросы для самопроверки

1. В чем заключается метод конечных разностей?
2. Почему в уравнения (16.7) не входит частная производная F_x?
3. Выведите конечно-разностный аналог дифференциального уравнения Эйлера — Пуассона (4.8).
4. Сколько точек из окружения z_{ik} участвует в конечно-разностном уравнении, если уравнение Эйлера — Остроградского содержит $\partial^2 z/\partial x \partial y$? не содержит $\partial^2 z/\partial x \partial y$?
5. Как аппроксимировать вторые частные производные на треугольной сетке?

16.5. Примеры выполнения заданий

16.5.1. Задание 1

Решить методом конечных разностей *задание 1 главы 2*. Сравнить решение с аналитическим. Построить графики.

На основе программы для *задания 1 главы 2* составим программу для данного примера. Вводим исходные данные. Задаем дополнительно число интервалов разбиения `nf`. Решаем вначале *задание 1 главы 2*.

```
clear all % очистили память
disp('Решаем задание 16.1')
syms x y Dy D2y
nf=10; % число интервалов
F=x^2+y^2+Dy^2; % подынтегральная функция
x1=-1; % вводим граничные условия
y1=1;
```

```
x2=1;
y2=2;
disp('Исходные данные:')
fprintf('Подынтегральная функция F(x,y,y'')=%s\n',char(F))
fprintf('Граничное условие слева: y(%d)=%d\n',x1,y1)
fprintf('Граничное условие справа: y(%d)=%d\n',x2,y2)
dFdy=diff(F,y); % вычислили Fy
dFdy1=diff(F,Dy); % вычислили Fy'
d_dFdy1_dx=diff(dFdy1,x); % Fy'x
d_dFdy1_dy=diff(dFdy1,y); % Fy'y
d_dFdy1_dy1=diff(dFdy1,Dy); % Fy'y'-условие Лежандра
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+d_dFdy1_dy1*D2y;
Euler=simple(dFdy-dFy1dx); % уравнение Эйлера
deqEuler=[char(Euler) '=0']; % добавили =0
Sol=dsolve(deqEuler,'x'); % решаем уравнение Эйлера
if length(Sol)~=1 % решений нет или более одного
  error('Нет решений или более одного решения!');
end
SolLeft=subs(Sol,x,x1); % подставили x1
SolRight=subs(Sol,x,x2); % подставили x2
EqLeft=[char(SolLeft) '=' char(sym(y1))]; % =y1
EqRight=[char(SolRight) '=' char(sym(y2))]; % =y2
Con=solve(EqLeft,EqRight,'C1,C2'); % решаем систему
C1=Con.C1; % присваиваем полученные решения
C2=Con.C2; % символическим константам C1 и C2
Sol21=vpa(eval(Sol),14); % подставили C1,C2
xpl=linspace(x1,x2); % задаем массив абсцисс
y21=subs(Sol21,x,xpl); % вычислили ординаты
Решаем задание 16.1
Исходные данные:
Подынтегральная функция F(x,y,y')=x^2+y^2+Dy^2
Граничное условие слева: y(-1)=1
Граничное условие справа: y(1)=2
```

Формируем конечно-разностное уравнение вида (16.7). Для этого описываем необходимые символические переменные: x_{k-1}, x_k, x_{k+1}, y_{k-1}, y_k, y_{k+1}. Вычисляем символические частные производные F_y и $F_{y'}$ в нужных точках: предыдущей, текущей и следующей. Выводим уравнение (16.7) и печатаем его.

```
syms xkm1 xk xkp1 ykm1 yk ykp1
dx=sym((x2-x1)/nf);
dFdyP=subs(dFdy,{x,y,Dy},{(xkm1+xk)/2,(ykm1+yk)/2,(yk-ykm1)/dx});
dFdyN=subs(dFdy,{x,y,Dy},{(xk+xkp1)/2,(yk+ykp1)/2,(ykp1-yk)/dx});
dFdy1P=subs(dFdy1,{x,y,Dy},...
  {(xkm1+xk)/2,(ykm1+yk)/2,(yk-ykm1)/dx});
```

```
dFdy1N=subs(dFdy1,{x,y,Dy},...
  {(xk+xkp1)/2,(yk+ykp1)/2,(ykp1-yk)/dx});
EqF=simple((dFdyN+dFdyP)/2-(dFdy1N-dFdy1P)/dx);
fprintf('Шаг сетки разбиения dx=%s\n',char(dx))
fprintf('Конечно-разностное уравнение:\n%s=0\n',char(EqF))
Шаг сетки разбиения dx=1/5
Конечно-разностное уравнение:
101*yk-99/2*ykp1-99/2*ykm1=0
```

Это уравнение в общем случае зависит от x_{k-1}, x_k, x_{k+1}, y_{k-1}, y_k, y_{k+1}. Сформируем из него трехдиагональную систему уравнений. Коэффициенты при неизвестных этой системы — это коэффициенты при y_{k-1}, y_k, y_{k+1}; а правые части — свободные члены. Заполним матрицу коэффициентов и вектор правых частей.

```
neq=nf-1; % количество уравнений
fprintf('Количество уравнений neq=%d\n',neq)
edx=eval(dx); % шаг сетки
x171=linspace(x1,x2,nf+1); % абсциссы
xkC=x171(2:end-1)'; % текущие точки
xkP=xkC-edx; % предыдущие точки
xkN=xkC+edx; % последующие точки
ew=subs(EqF,{ykm1,yk,ykp1},{0,0,0}); % свободные члены
e0=subs(ew,{xkm1,xk,xkp1},{xkP,xkC,xkN});
e0=ones(neq,1).*e0; % если нужно, сделали вектором
ew=subs(EqF,{ykm1,yk,ykp1},{1,0,0});
ekP=subs(ew,{xkm1,xk,xkp1},{xkP,xkC,xkN})-e0;
ew=subs(EqF,{ykm1,yk,ykp1},{0,1,0});
ekC=subs(ew,{xkm1,xk,xkp1},{xkP,xkC,xkN})-e0;
ew=subs(EqF,{ykm1,yk,ykp1},{0,0,1});
ekN=subs(ew,{xkm1,xk,xkp1},{xkP,xkC,xkN})-e0;
b=-e0; % правые части
A=diag(ekC)+diag(ekN(1:neq-1),1)+diag(ekP(2:neq),-1);
b(1)=b(1)-ekP(1)*y1; % учли граничные условия
b(neq)=b(neq)-ekN(1)*y2;
Количество уравнений neq=9
```

Решаем полученную систему и строим графики (рис. 16.5): сплошной линией — численного решения и, для сравнения, штриховой — аналитического.

```
y=double(A\b); % решаем систему
y161=[y1;y;y2]; % решение
figure
plot(xpl,y21,'--b', x171,y161,'-r') % рисуем
```

```
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 16.1') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
```

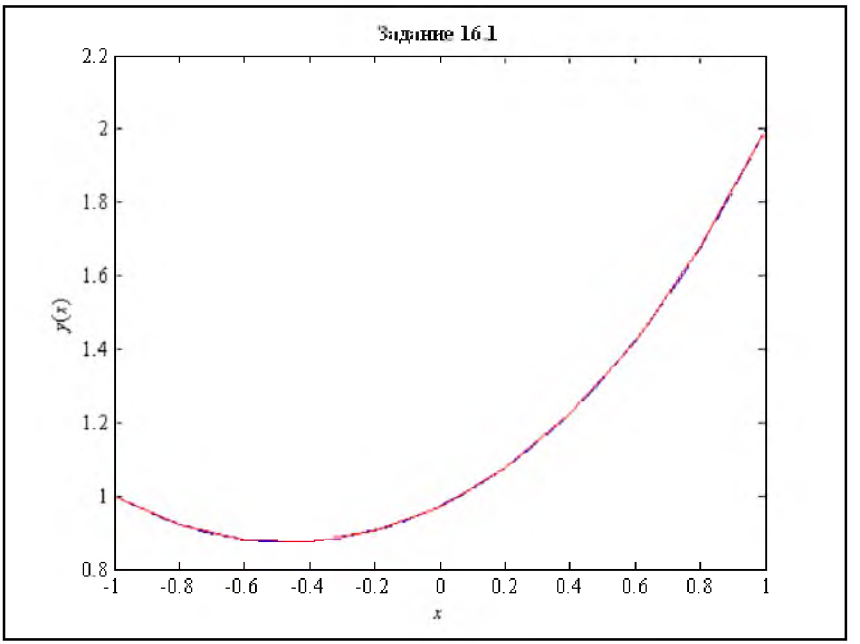


Рис. 16.5. Решение задания 16.1

16.5.2. Задание 2

Решить МКР задание *главы 5*. Визуально сравнить с решением МКЭ. Построить график.

Программу для этого примера напишем на основе программы для задания *главы 5* и теории, изложенной в *разделе 16.3*. Вначале введем исходные данные.

```
clear all % очистили память
disp('Решаем задание 16.2')
syms x y z Dzx Dzy
F=Dzx^2+2*Dzy^2+10*y*z*(sin(x)+x^2/5);
zc=x*(x-0.4)/400; % граничное условие
bc='y<-0.1999'; % участок границы
```

```
disp('Исходные данные:')
disp('Подынтегральная функция:')
fprintf('F(x,y,y,dz/dx,dz/dy)=%s\n',char(F))
disp('Граничное условие:')
fprintf('при %s: z=%s;\n',bc,char(zc))
disp('на остальных участках z=0.')
```

```
Решаем задание 16.2
Исходные данные:
Подынтегральная функция:
F(x,y,y,dz/dx,dz/dy)=Dzx^2+2*Dzy^2+10*y*z*(sin(x)+1/5*x^2)
Граничное условие:
при y<-0.1999: z=1/400*x*(x-2/5);
на остальных участках z=0.
```

Нам нужно сформировать систему уравнений (16.15—16.16). Вычисляем для нее частные производные F_z, F_p, F_q. В выражении для F_z будут слагаемые, содержащие в качестве множителя z, и слагаемые без z. Первые нам нужно оставить в левой части системы (16.15); по ним мы будем формировать матрицу K. Вторые слагаемые мы перенесем направо и будем использовать для построения вектора f.

```
dFdz=diff(F,z);
dFdp=diff(F,Dzx);
dFdq=diff(F,Dzy);
EqR=-subs(dFdz,z,0); % правая часть
fz=eval((dFdz+EqR)/z); % коэффициент при z - число
fdFdp=eval(dFdp/Dzx); % коэффициент при dFdp - число
fdFdq=eval(dFdq/Dzy); % коэффициент при dFdq - число
fprintf('Формула для правой части: f=%s\n',char(EqR));
fprintf(['В левой части:\nКоэффициент при z = %d\n'...
  'Коэффициент при dz/dx = %d\n'...
  'Коэффициент при dz/dy =  = %d\n'],fz,fdFdp,fdFdq);
```

```
Формула для правой части: f=-10*y*(sin(x)+1/5*x^2)
В левой части:
Коэффициент при z = 0
Коэффициент при dz/dx = 2
Коэффициент при dz/dy =  = 4
```

Сформируем область и разобьем ее на треугольные элементы. Для этого возьмем фрагмент из программы для *задания главы 5*. Здесь удалено только рисование эскиза области и сетки разбиения.

```
gd=[2;4;0;0.4;0.4;0;-0.2;-0.2;0.2;0.2]; % многоугольник
gd=[gd,[1;0.2;-0.2;0.4;0;0;0;0;0;0]]; % окружность
```

```
gd=[gd,[1;0;0;0.1;0;0;0;0;0;0]]; % окружность
gd=[gd,[4;0.5;0;0.2;0.1;pi/6;0;0;0;0]]; % эллипс
ns='abcd'; % строка
fo='(a*b+c)-d'; % формула операций
[dl,bt]=decsg(gd,fo,ns); % формируем область
[dl1,bt1]=csgdel(dl,bt); % удалили внутренние границы
ne=size(dl1,2); % число участков границы
fprintf('Число участков границы ne=%d\n',ne)
[p,e,t]=initmesh(dl1); % формируем FEM-сетку
[p,e,t]=refinemesh(dl1,p,e,t); % измельчаем
np=size(p,2); % число узлов
nel=size(t,2); % число элементов
nb=size(e,2); % число граничных линий
fprintf('Число узлов np=%d\n',np)
fprintf('Число элементов nel=%d\n',nel)
fprintf('Число граничных линий nb=%d\n',nb)
Число участков границы ne=11
Число узлов np=506
Число элементов nel=924
Число граничных линий nb=86
```

Продолжаем готовить данные для системы уравнений (16.15—16.16). Находим координаты вершин каждого треугольника. Вычисляем площади всех треугольников S_k, величины y_{123k}, x_{123k} и другие вспомогательные величины.

```
xk=reshape(p(1,t(1:3,:)),3,nel)'; % координаты треугольников
yk=reshape(p(2,t(1:3,:)),3,nel)';
Sk=((xk(:,2)-xk(:,1)).*(yk(:,3)-yk(:,1))-...
  (xk(:,3)-xk(:,1)).*(yk(:,2)-yk(:,1)))/2; % площади
y123k=[yk(:,2)-yk(:,3),yk(:,3)-yk(:,1),yk(:,1)-yk(:,2)];
x123k=[xk(:,3)-xk(:,2),xk(:,1)-xk(:,3),xk(:,2)-xk(:,1)];
dEqRdx=diff(EqR,x); % ищем частные производные
dEqRdy=diff(EqR,y);
if (dEqRdx==0)&(dEqRdy==0), % не зависит ни от x, ни от y
  fel=zeros(nel,1); % нулевые правые части
elseif (dEqRdx==0), % подставляем только y
  fel=subs(EqR,y,mean(yk,2));
elseif (dEqRdy==0), % подставляем только x
  fel=subs(EqR,x,mean(xk,2));
else % подставляем и x, и y
  fel=subs(EqR,{x,y},{mean(xk,2),mean(yk,2)});
end
disp(['Вычислили площади, y123, x123, '...
  'функцию f в элементах']);
Вычислили площади, y123, x123, функцию f в элементах
```

Заполняем матрицу K и вектор f поблочно, как показано на рис. 16.4. Просматриваем каждый треугольный элемент. Для него находим номера узлов и заполняем K и f в соответствии с (16.15). Из-за большого размера матрицу K строим как разреженную.

```
K=sparse(np,np); % матрица K - разреженная
f=zeros(np,1);
Sk3=Sk/3;
Sk9=Sk/9;
fdFdp4=fdFdp/4;
fdFdq4=fdFdq/4;
for k=1:nel, % просматриваем каждый треугольник
  k123=t(1:3,k); % узлы треугольника
  K(k123,k123)=K(k123,k123)+Sk9(k)*fz; % из Fz
  for ik=1:3, % из Fp и Fq
    f(k123(ik))=f(k123(ik))+fel(k)*Sk3(k);
    for jk=1:3,
      K(k123(ik),k123(jk))=K(k123(ik),k123(jk))+...
        (fdFdp4*y123k(k,ik)*y123k(k,jk)+...
        fdFdq4*x123k(k,ik)*x123k(k,jk))/Sk(k);
    end
  end
end
disp('Заполнили матрицу K и вектор f');
```

```
Заполнили матрицу K и вектор f
```

Строим вектор u граничных условий, участвующий в формулах (16.17—16.18). Для этого находим узлы, удовлетворяющие заданному условию, и вычисляем u в этих узлах по заданной формуле.

```
xbor=p(1,:); % координаты узлов
ybor=p(2,:);
bcbor=char(subs(bc,{x,y},{sym('xbor'),sym('ybor')}));
fprintf('Ищем точки, удовлетворяющие условию: %s\n',bcbor);
numbor=eval(['find(' bcbor ');']); % номера узлов
fprintf('Таких точек оказалось %d\n',length(numbor));
u=zeros(np,1); % для граничных условий
dzcdx=diff(zc,x);
dzcdy=diff(zc,y);
if (dzcdx==0)&(dzcdy==0), % не зависит ни от x, ни от y
elseif (dzcdx==0), % подставляем только y
  u(numbor)=subs(zc,y,p(2,numbor));
elseif (dzcdy==0), % подставляем только x
  u(numbor)=subs(zc,x,p(1,numbor));
```

```
else % подставляем и x, и y
  u(number)=subs(zc,{x,y},{p(1,number),p(2,number)});
end
disp('Построили вектор u граничных условий');
Ищем точки, удовлетворяющие условию: ybor < -.1999
Таких точек оказалось 17
Построили вектор u граничных условий
```

Ищем вторую составляющую решения — функцию v. Чтобы "граничные" координаты v получились близкими к нулевым, делаем соответствующие диагональные элементы матрицы K большими. Складываем u и v — находим решение. Строим его график (рис. 16.6). Выравниваем масштабы по осям *x* и *y*, чтобы область выглядела неискаженной. Выбираем красочную палитру по умолчанию. Показываем сетку и внешний контур. Надписываем график и метки осей.

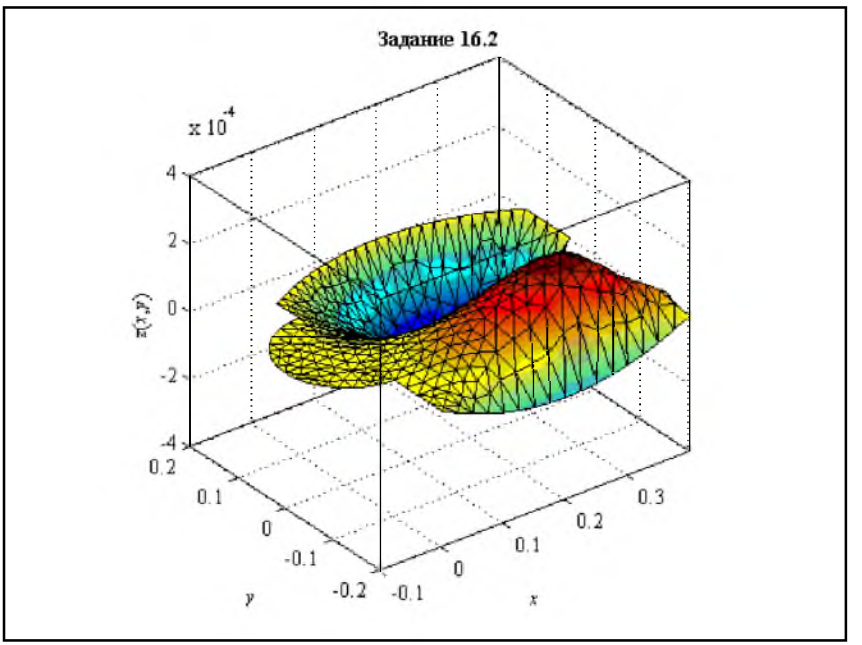


Рис. 16.6. Решение задания 16.2

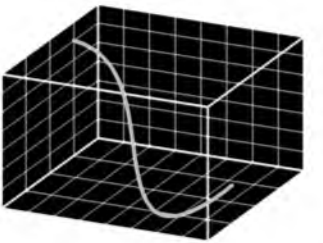
```
f1=f-K*u; % правые части для v
borp=unique(reshape(e(1:2,:),2*nb,1)); % узлы на границе
for k=1:length(borp), % добавим в диагональные элементы
  K(borp(k),borp(k))=1e20; % большие числа
end
```

```
v=K\f1;
uv=u+v;
figure % новая фигура
pdeplot(p,e,t,'xydata',uv,'zdata',uv,...
  'mesh','on','colorbar','off'); % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
v1=axis; % границы осей
da=daspect; % масштаб осей
da(1:2)=min(da(1:2)); % min из двух
daspect(da) % выравниваем масштаб осей
axis(v1); % оставили границы
colormap default % палитра по умолчанию
grid on % показали сетку
box on % показали внешний контур
title('\bfЗадание 16.2') % заголовок
xlabel('\itx')
ylabel('\ity')
zlabel('\itz\rm(\itx\rm,\ity\rm)') % метки осей
```

16.6. Задание

1. Решить *задание 1 главы 2* методом конечных разностей. Сравнить решение с аналитическим.
2. Решить задание *главы 5* методом конечных разностей. Сравнить решение МКР с решением МКЭ.

ГЛАВА 17



Метод Ритца

17.1. Применение метода Ритца к одномерным задачам

Рассмотрим еще один метод решения вариационных задач — *метод Ритца* (Walter Ritz, 1878—1909, рис. 17.1). В этом методе решение вариационной задачи ищется в виде линейной комбинации известных, заданных заранее функций (они называются базисными). После подстановки такой линейной комбинации в функционал J он становится функцией неизвестных коэффициентов этой комбинации. Коэффициенты подбираются так, чтобы функционал принимал экстремальное значение. Таким образом, вариационная задача сводится к задаче исследования на экстремум функции нескольких переменных.

По сути дела, в методе Ритца мы сужаем класс допустимых функций: вместо множества непрерывных и дифференцируемых функций, проходящих через граничные точки (см. рис. 2.1), мы ограничиваемся только всевозможными линейными комбинациями базисных функций. Конечно же, эти линейные комбинации должны быть допустимыми: они должны удовлетворять граничным условиям при любых значениях коэффициентов. Если граничные условия однородные (нулевые), то это легко сделать: нужно выбрать такие базисные функции, чтобы все они также удовлетворяли однородным граничным условиям. Если же граничные условия неоднородные, то можно применить следующий прием. Будем искать решение в виде линейной комбинации линейно-независимых базисных функций $\varphi_k(x)$, удовлетворяющих однородным граничным условиям:

$$\varphi_k(x_1) = 0; \quad \varphi_k(x_2) = 0; \quad k = [1, n]; \tag{17.1}$$

но прибавим к этой комбинации какую-либо функцию $\varphi_0(x)$, удовлетворяющую заданным граничным условиям:

$$y(x) = \varphi_0(x) + \sum_{k=1}^{n} \alpha_k \varphi_k(x). \tag{17.2}$$

Коэффициент при $\varphi_0(x)$ не варьируется, он всегда равен 1. Таким образом, полученная линейная комбинация при любых значениях варьируемых параметров будет удовлетворять заданным граничным условиям.

Рис. 17.1. У. Ритц

В остальном выбор базисных функций произвольный, лишь бы они были линейно-независимыми. Конечно, желательно при их выборе использовать различные дополнительные соображения, например, предполагаемый вид решения. Тогда для достижения нужной точности понадобится меньше самих базисных функций.

17.2. Метод Ритца в применении к двумерным задачам

При решении двумерных задач общие соображения — те же. Мы ищем решение в виде (17.2), но все базисные функции будут уже зависеть от двух переменных: $\varphi_k(x, y)$. Функция $\varphi_0(x, y)$ должна удовлетворять граничным условиям на контуре L — границе области D:

$$\varphi_0(x, y)|_L = u_0(x, y), \tag{17.3}$$

а остальные $\varphi_k(x, y)$ — однородным граничным условиям:

$$\varphi_k(x, y)|_L = 0. \tag{17.4}$$

Если контур L — простой (прямоугольник, эллипс и т. п.), то построение системы базисных функций не представляет особых трудностей. Если же L, как у нас, сложный, то это становится нетривиальной задачей. Один из возможных подходов к ее решению — применение R-функций [39]. Теория R-функций разработана академиком НАН Украины Владимиром Логвиновичем Рвачевым (1926 г. рожд., рис. 17.2), и эти функции носят его имя.

Рис. 17.2. В. Л. Рвачев

Основная идея этого подхода — найти такую функцию $\omega(x, y)$ (хотя бы одну), которая удовлетворяла бы однородным граничным условиям (17.4), а внутри области D была бы строго положительной. Тогда всевозможные базисные функции $\varphi_k(x, y)$ можно построить как произведения этой $\omega(x, y)$ на полиномы (или другие линейно-независимые функции). Для построения функции $\omega(x, y)$ нужно вначале записать логическое уравнение области в виде

$$L_1 b_1 L_2 b_2 \ldots b_{m-1} L_m. \tag{17.5}$$

Здесь L_k — логические выражения вида

$$\omega_k(x, y) \geq 0 \tag{17.6}$$

(неравенства, связывающие координаты x, y и описывающие различные участки границы), а b_k — одна из логических операций: $\cap$ ("и"), $\cup$ ("или"),

$\neg$ ("не"), и в выражении (17.5) можно расставлять скобки в соответствии с обычными правилами арифметики. После того как логическое выражение (17.5) построено (в теории R-функций оно называется предикатным уравнением), нетрудно уже построить и саму функцию $\omega(x, y)$. Для этого все логические операции $\cap$ заменяются функцией

$$x \cap y \Rightarrow x + y - \sqrt{x^2 + y^2}, \tag{17.7}$$

операции $\cup$ — функцией

$$x \cup y \Rightarrow x + y + \sqrt{x^2 + y^2}, \tag{17.8}$$

операция $\neg$ — функцией

$$\neg x \Rightarrow -x, \tag{17.9}$$

а вместо x и y берутся левые части соответствующих выражений (17.6).

Функции (17.7—17.9) и называются R-функциями. С их помощью можно построить необходимую нам $\omega(x, y)$. Вид (17.7—17.9) — не единственный. Существуют и другие системы R-функций.

Для построения $\varphi_0(x, y)$ в [39] предлагается "формула склейки". Если граница области L состоит из m участков $\omega_k(x, y)$, на которых искомая функция равна соответственно $u_k(x, y)$, то можно взять

$$\varphi_0(x, y) = \frac{\sum_{k=1}^{m} \frac{u_k(x, y)}{\omega_k(x, y)}}{\sum_{k=1}^{m} \frac{1}{\omega_k(x, y)}}. \tag{17.10}$$

Полученное выражение понимается в предельном смысле, т. е. возникающие неопределенности нужно раскрывать. Однако возможны и другие подходы к построению $\varphi_0(x, y)$.

17.3. МКР + метод Ритца $\Rightarrow$ МКЭ

Вернемся к методу конечных разностей, рассмотренному нами в *главе 16*. Для одномерных задач мы аппроксимировали искомую функцию $y(x)$ кусочно-линейной ломаной с неизвестными ординатами y_k в узлах. Такой подход можно считать разновидностью метода Ритца, если выбирать базисные функции $\varphi_k(x)$ в виде:

$$\varphi_k(x) = \begin{cases} 0; & x \notin [x_{k-1}, x_{k+1}]; \\ \dfrac{x - x_{k-1}}{x_k - x_{k-1}}; & x \in [x_{k-1}, x_k]; \\ \dfrac{x - x_{k+1}}{x_k - x_{k+1}}; & x \in [x_k, x_{k+1}], \end{cases} \tag{17.11}$$

а функцию $\varphi_0(x)$ — в виде:

$$\varphi_0(x) = \begin{cases} y_1 \dfrac{x - x_1}{x_0 - x_1}; & x < x_1; \\ 0; & x \in [x_1, x_{n-1}]; \\ y_2 \dfrac{x - x_n}{x_{n-1} - x_n}; & x > x_{n-1}, \end{cases} \tag{17.12}$$

где y_1, y_2 — заданные граничные условия. Эти функции показаны на рис. 17.3 и 17.4.

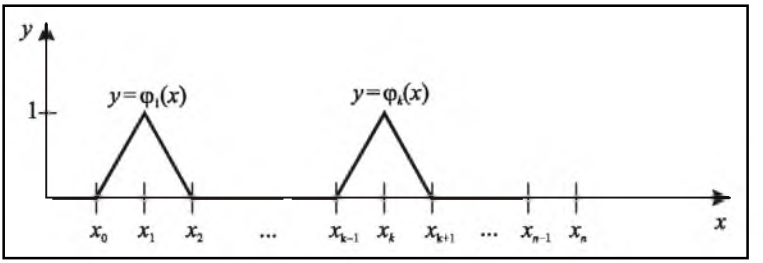


Рис. 17.3. Базисные функции $\varphi_k(x)$ МКР

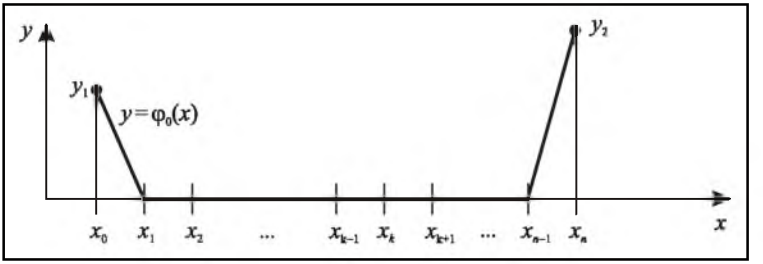


Рис. 17.4. Базисная функция $\varphi_0(x)$ МКР

Базисные функции $\varphi_k(x)$ — кусочно-линейные, они равны 1 в одной точке x_k и 0 в остальных узлах. Их линейная комбинация (с добавлением $\varphi_0(x)$) будет

кусочно-линейной, а коэффициенты этой комбинации как раз будут равны неизвестным значениям y_k.

Аналогичную интерпретацию имеет МКР и для двумерных задач на треугольной сетке. Если рассматривать его как метод Ритца, то здесь базисные функции $\varphi_k(x, y)$ имеют форму шатра или индейского вигвама: они равны 1 в одном k-м узле и 0 во всех остальных узлах, как показано на рис. 17.5.

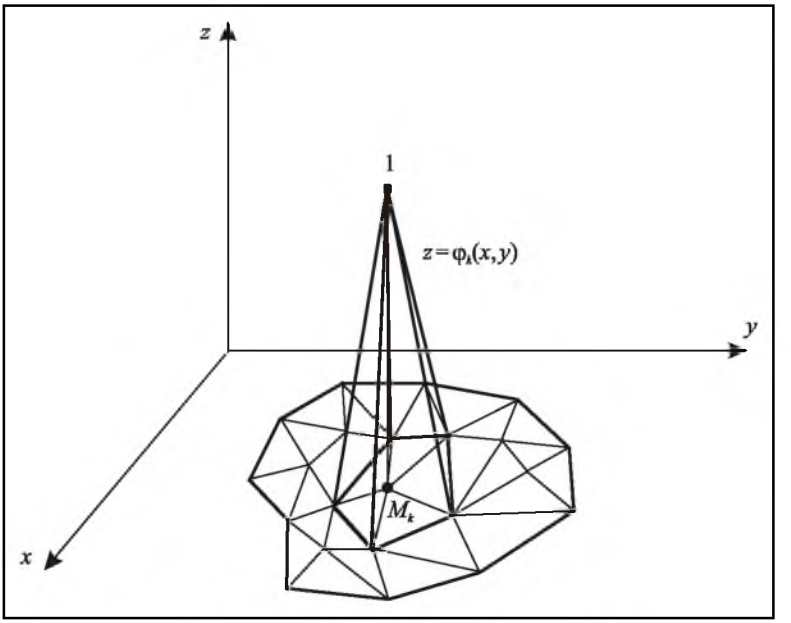


Рис. 17.5. Базисная функция $\varphi_k(x, y)$ МКР

Функция $\varphi_0(x, y)$ здесь похожа на корыто или крутой берег озера: ее значения в граничных узлах определяются граничным условием (5.2), а во всех внутренних узлах она равна нулю. Ее примерный вид — на рис. 17.6.

Развитие этих идей привело к появлению метода конечных элементов (МКЭ). По своей сути МКЭ — это метод Ритца с очень большим числом базисных функций, каждая из которых отлична от нуля только в небольшой области — конечном элементе. С другой стороны, мы видим, что метод конечных разностей (в наших задачах) можно рассматривать как частный случай МКЭ. Поэтому МКЭ иногда называют вариационно-разностным методом.

В нашу задачу не входит подробное изучение МКЭ. Некоторые первоначальные сведения о нем можно почерпнуть, например, из [17, 40, 54].

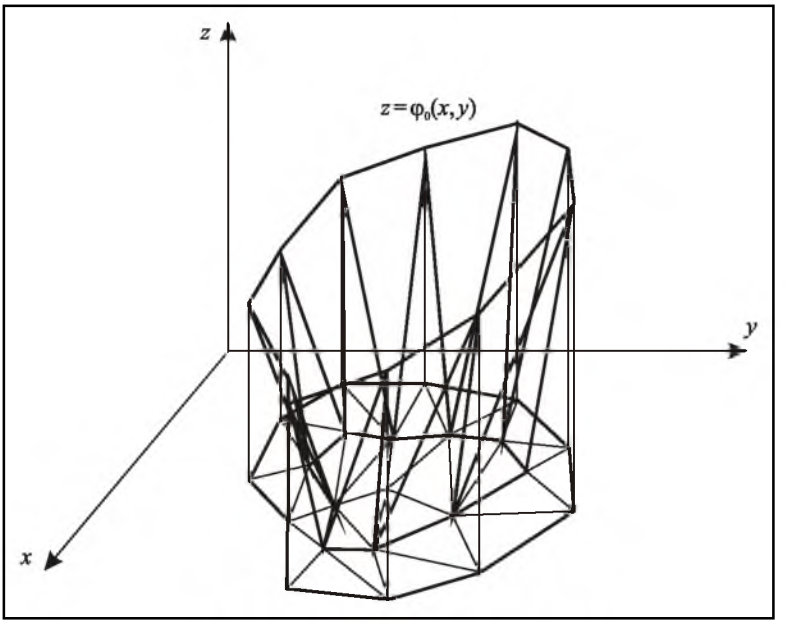


Рис. 17.6. Базисная функция $\varphi_0(x, y)$ МКР

17.4. Вопросы для самопроверки

1. В чем заключается метод Ритца?
2. В чем основная трудность применения метода Ритца к двумерным задачам?
3. Как строятся базисные функции $\varphi_k(x, y)$ в двумерной области?
4. Как строится функция $\varphi_0(x, y)$ в двумерной области?
5. Возможно ли применение метода R-функций к 3-мерным задачам?
6. Почему МКЭ называется вариационно-разностным?

17.5. Примеры выполнения заданий

17.5.1. Задание 1

Решить методом Ритца *задание 1 главы 2*. Взять в качестве базисных функций несколько полуволн синуса. Сравнить решение с аналитическим. Построить графики.

Будем искать решение в виде (17.2). Выберем $\varphi_0(x)$ в виде прямой, соединяющей граничные точки $M_1(x_1, y_1)$ и $M_2(x_2, y_2)$:

$$\varphi_0(x) = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1). \tag{17.13}$$

Базисные функции — это полуволны синуса:

$$\varphi_k(x) = \sin k\pi \frac{x - x_1}{x_2 - x_1}. \tag{17.14}$$

Составим программу для этого примера на основе программы для *задания 1 главы 2*. Вводим исходные данные. Решаем вначале *задание 1 главы 2*.

```
clear all % очистили память
disp('Решаем задание 17.1')
syms x y Dy D2y % описали символические переменные
F=x^2+y^2+Dy^2; % подынтегральная функция
x1=-1; % вводим граничные условия
y1=1;
x2=1;
y2=2;
disp('Исходные данные:')
fprintf('Подынтегральная функция F(x,y,y'')=%s\n',char(F))
fprintf('Граничное условие слева: y(%d)=%d\n',x1,y1)
fprintf('Граничное условие справа: y(%d)=%d\n',x2,y2)
dFdy=diff(F,y); % вычислили Fy
dFdy1=diff(F,Dy); % вычислили Fy'
d_dFdy1_dx=diff(dFdy1,x); % Fy'x
d_dFdy1_dy=diff(dFdy1,y); % Fy'y
d_dFdy1_dy1=diff(dFdy1,Dy); % Fy'y'- условие Лежандра
dFy1dx=d_dFdy1_dx+d_dFdy1_dy*Dy+d_dFdy1_dy1*D2y;
Euler=simple(dFdy-dFy1dx); % уравнение Эйлера
deqEuler=[char(Euler) '=0']; % добавили =0
Sol=dsolve(deqEuler,'x'); % решаем уравнение Эйлера
if length(Sol)~=1 % решений нет или более одного
  error('Нет решений или более одного решения!');
end
SolLeft=subs(Sol,x,x1); % подставили x1
SolRight=subs(Sol,x,x2); % подставили x2
EqLeft=[char(SolLeft) '=' char(sym(y1))]; % =y1
EqRight=[char(SolRight) '=' char(sym(y2))]; % =y2
Con=solve(EqLeft,EqRight,'C1,C2'); % решаем систему
C1=Con.C1; % присваиваем полученные решения
C2=Con.C2; % символическим константам C1 и C2
```

```
Sol21=vpa(eval(Sol),14); % подставили C1,C2
xpl=linspace(x1,x2); % задаем массив абсцисс
y21=subs(Sol21,x,xpl); % вычислили ординаты
Решаем задание 17.1
Исходные данные:
Подынтегральная функция F(x,y,y')=x^2+y^2+Dy^2
Граничное условие слева: y(-1)=1
Граничное условие справа: y(1)=2
```

Выберем необходимое количество базисных функций `nf`. Построим их и напечатаем.

```
nf=6; % задаем число базисных функций
fprintf('Количество базисных функций nf=%d\n',nf)
f0=y1+(y2-y1)*(x-x1)/(x2-x1); % функция phi0
fprintf('Выбрали базисные функции:\nfi0=%s\n',char(f0));
for k=1:nf, % строим базисные функции и печатаем их
  f{k}=sin(k*pi*(x-x1)/(x2-x1)); % строим
  fprintf('fi%d=%s\n',k,char(f{k})); % печатаем
end
Количество базисных функций nf=6
Выбрали базисные функции:
fi0=3/2+1/2*x
fi1=sin(1/2*pi*(x+1))
fi2=sin(pi*(x+1))
fi3=sin(3/2*pi*(x+1))
fi4=sin(2*pi*(x+1))
fi5=sin(5/2*pi*(x+1))
fi6=sin(3*pi*(x+1))
```

Сформируем решение в виде (17.2). Найдем от него производную, вычислим подынтегральную функцию $F(x,y,y')$ и функционал J. Он будет функцией коэффициентов α_k. Для исследования его на экстремум найдем производные и приравняем их нулю. Напечатаем полученную систему уравнений.

```
yr=f0; % начинаем формировать решение
for k=1:nf,
  a{k}=sym(sprintf('a%d',k)); % описали символическую переменную
  yr=yr+a{k}*f{k}; % добавили в решение
end
Dyr=diff(yr,x); % продифференцировали
Fr=subs(F,{y,Dy},{yr,Dyr}); % подставили
Jr=int(Fr,x,x1,x2); % проинтегрировали
for k=1:nf,
  eq{k}=char(diff(Jr,a{k})); % дифференцируем
end
```

```
disp('Получена система уравнений:')
fprintf('%s=0\n',eq{:});
```
```
Получена система уравнений:
1/60*(720+30*a1*pi^3+120*a1*pi)/pi=0
1/60*(-120+120*a2*pi^3+120*a2*pi)/pi=0
1/60*(240+120*a3*pi+270*a3*pi^3)/pi=0
1/60*(-60+120*a4*pi+480*a4*pi^3)/pi=0
1/60*(144+750*a5*pi^3+120*a5*pi)/pi=0
1/60*(-40+120*a6*pi+1080*a6*pi^3)/pi=0
```

Для решения полученной системы уравнений применим команду `solve`. Ее синтаксис требует явного перечисления всех уравнений. Сформируем строку с соответствующей командой.

```
com='solve('; % формируем команду решения
for k=1:nf, % добавили уравнение
  com=[com sprintf('eq{%d},',k)];
end
com=[com ''''']; % добавили '
for k=1:nf, % добавили переменные
  com=[com sprintf('a%d,',k)];
end
com=[com(1:end-1) ''')']; % закончили команду
fprintf('Команда решения системы уравнений:\n%s\n',com);
```
```
Команда решения системы уравнений:
solve(eq{1},eq{2},eq{3},eq{4},eq{5},eq{6},'a1,a2,a3,a4,a5,a6')
```

Решаем полученную систему уравнений. Печатаем найденные значения коэффициентов α_k.

```
SolSys=eval(com); % решаем систему
disp('Найдены коэффициенты:')
for k=1:nf,
  com=sprintf('SolSys.a%d',k); % команда выборки
  ff=eval(com); % выбрали нужное поле
  ar(k)=eval(ff); % перевели в число
  fprintf('a%d=%s=%15.10f\n',k,char(ff),ar(k));
end
```
```
Найдены коэффициенты:
a1=-24/pi/(4+pi^2)=  -0.5508042658
a2=1/pi/(1+pi^2)=   0.0292844040
a3=-8/pi/(4+9*pi^2)=  -0.0274326916
a4=1/2/pi/(1+4*pi^2)=   0.0039318470
a5=-24/5/pi/(4+25*pi^2)=  -0.0060935103
a6=1/3/pi/(1+9*pi^2)=   0.0011812034
```

Обратите внимание: с ростом номера k коэффициенты в среднем убывают по абсолютной величине. Это говорит об удачном подборе базисных функций. Подставляем α_k в решение. Строим на одном графике аналитическое решение и решение, полученное методом Ритца (рис. 17.7).

```
yrr=subs(yr,a,ar); % подставили константы
y171=subs(yrr,x,xpl); % вычислили ординаты
figure
plot(xpl,y21,'--b', xpl,y171,'-r') % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЗадание 17.1') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity\rm(\itx\rm)') % метка оси OY
```

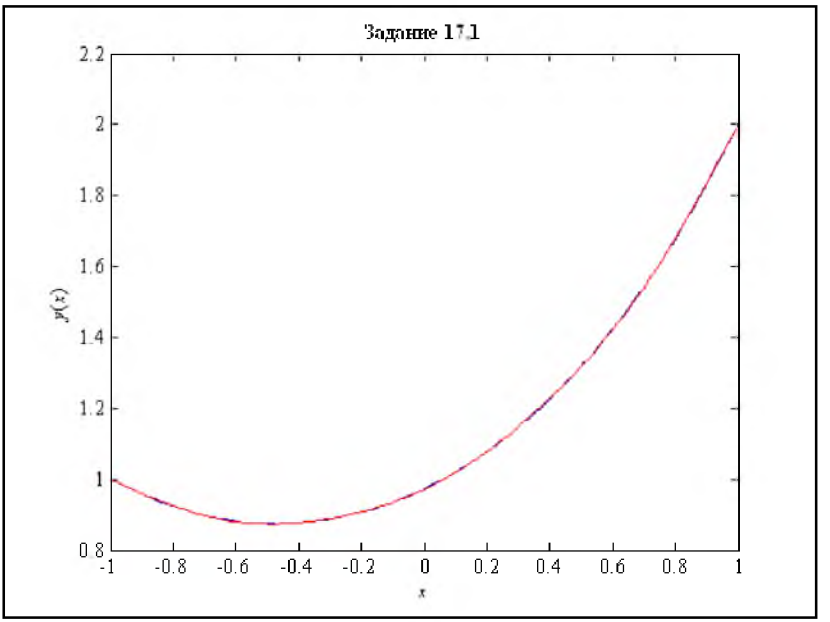


Рис. 17.7. Решение задания 17.1, выполненное с помощью MATLAB

17.5.2. Задание 2

Решить методом Ритца *задание главы 5*. Взять в качестве базисных функций многочлены до 2-й степени включительно, умноженные на функцию $\omega(x, y)$. Сравнить с решением МКР *(задание 2 главы 16)* и МКЭ *(задание главы 5)*. Удачно ли выбраны базисные функции?

Начнем с ввода исходных данных, которые мы возьмем из своего варианта *задания главы 5*.

```
clear all % очистили память
disp('Решаем задание 17.2')
syms x y z Dzx Dzy D2zx2 D2zxy D2zy2
F=Dzx^2+2*Dzy^2+10*y*z*(sin(x)+x^2/5);
zc=x*(x-0.4)/400; % граничное условие
bc='y<-0.1999'; % участок границы
disp('Исходные данные:')
fprintf('F(x,y,y,dz/dx,dz/dy)=%s\n',char(F))
disp('Граничное условие:')
fprintf('при %s: z=%s;\n',bc,char(zc))
disp('на остальных участках z=0.')
Решаем задание 17.2
Исходные данные:
F(x,y,y,dz/dx,dz/dy)=Dzx^2+2*Dzy^2+10*y*z*(sin(x)+1/5*x^2)
Граничное условие:
при y<-0.1999: z=1/400*x*(x-2/5);
на остальных участках z=0.
```

Нам нужно будет строить сложную область из простых с помощью R-функций (17.7—17.9). Построим такие функции для операций $\cap$ и $\cup$. Операция $\neg$ очень простая, и для нее R-функцию составлять не будем. Функции составляем таким образом, чтобы их можно было вызвать для любого количества аргументов. Записываем обе функции в файлы.

```
s{1}='function z=R_And(varargin)';  % заголовок
s{2}='z1=varargin{1};'; % берем 1-й аргумент
s{3}='if nargin>1,';
s{4}='  for k=2:nargin,';
s{5}='    z1=z1+varargin{k}-(z1.^2+varargin{k}.^2).^0.5;';
s{6}='  end';
s{7}='end';
s{8}='z=z1;';
filenameAnd=fullfile(pwd,'R_And.m');
disp(['Текст файла R-функции AND ' filenameAnd ':'])
fprintf('%s\n',s{:})
fid=fopen(filenameAnd,'w');
fprintf(fid,'%s\n',s{:});
fclose(fid); % закрываем файл
s{1}='function z=R_Or(varargin)'; % заголовок
s{5}='    z1=z1+varargin{k}+(z1.^2+varargin{k}.^2).^0.5;';
filenameOr=fullfile(pwd,'R_Or.m');
disp(['Текст файла R-функции OR ' filenameOr ':'])
```

```
fprintf('%s\n',s{:})
fid=fopen(filenameOr,'w');
fprintf(fid,'%s\n',s{:});
fclose(fid); % закрываем файл
Текст файла R-функции AND D:\Iglin\Matlab\R_And.m:
function z=R_And(varargin)
z1=varargin{1};
if nargin>1,
  for k=2:nargin,
    z1=z1+varargin{k}-(z1.^2+varargin{k}.^2).^0.5;
  end
end
z=z1;
Текст файла R-функции OR D:\Iglin\Matlab\R_Or.m:
function z=R_Or(varargin)
z1=varargin{1};
if nargin>1,
  for k=2:nargin,
    z1=z1+varargin{k}+(z1.^2+varargin{k}.^2).^0.5;
  end
end
z=z1;
```

Зададим уравнения участков границы $\omega_k(x, y)$. Знак выбираем так, чтобы при замене знака равенства на знак ">" мы перешли внутрь области решения.

Рассмотрим, как записываются выражения для левых частей неравенств различных областей. Проще всего записывается выражение для полуплоскости. Нужно записать уравнение соответствующей прямой ($Ax+By+C=0$) и при необходимости умножить его на -1, чтобы выбрать нужную полуплоскость.

Внутренность круга радиуса R с центром в точке (x_0, y_0) нужно задавать в виде:

$$R^2-\left(x-x_0\right)^2-\left(y-y_0\right)^2>0\,; \tag{17.15}$$

а внутренность эллипса с полуосями a и b — в виде:

$$1-\frac{x^2}{a^2}-\frac{y^2}{b^2}>0\,. \tag{17.16}$$

Если эллипс повернут на угол α радиан против часовой стрелки, то уравнение его внутренней части имеет вид

$$1-\frac{\left(x\cos\alpha+y\sin\alpha\right)^2}{a^2}-\frac{\left(-x\sin\alpha+y\cos\alpha\right)^2}{b^2}>0\,, \tag{17.17}$$

а если центр его еще и смещен в точку (x_0, y_0), то

$$1-\frac{\big((x-x_0)\cos\alpha+(y-y_0)\sin\alpha\big)^2}{a^2}-\\ -\frac{\big(-(x-x_0)\sin\alpha+(y-y_0)\cos\alpha\big)^2}{b^2}>0\,. \tag{17.18}$$

Зададим левые части уравнений участков границы. Сформируем из них функцию $\omega(x, y)$ и напечатаем ее. Выражение для этой функции — очень длинное, поэтому при печати разобьем его на отдельные строки. Вычислим также производные от функции $\omega(x, y)$ (они будут еще длиннее).

```
o(1)=x;
o(2)=y+0.2;
o(3)=0.4-x;
o(4)=0.4^2-(x-0.2)^2-(y+0.2)^2;
o(5)=1-((x-0.5)*cos(pi/6)+y*sin(pi/6))^2/0.2^2-...
  (-(x-0.5)*sin(pi/6)+y*cos(pi/6))^2/0.1^2;
o(6)=0.1^2-x^2-y^2;
m=length(o); % число участков границы
disp('Линии границ:')
for k=1:m,
  fprintf('o(%d):  %s=0\n',k,char(o(k)))
end
omega=R_And(R_Or(R_And(o(1),o(2),o(3),o(4)),o(6)),-o(5));
domegadx=diff(omega,x);
domegady=diff(omega,y);
chom=char(omega); % перевели omega в строку
lom=length(chom); % число символов
nl=70; % длина одной строки
nf=floor(lom/nl); % число строк по nl символов
ost=lom-nf*nl; % остаток при делении на nl
disp('omega(x,y)=')
for kk=1:nf, % печатаем полные строки
   fprintf('%s\n',chom((kk-1)*nl+1:kk*nl))
end
if ost>0, % есть остаток
   fprintf('%s\n',chom(nf*nl+1:end))
end
Линии границ:
o(1):  x=0
o(2):  y+1/5=0
o(3):  2/5-x=0
```

```
o(4):  4/25-(x-1/5)^2-(y+1/5)^2=0
o(5):  1-25*(1/2*(x-1/2)*3^(1/2)+1/2*y)^2-
       100*(-1/2*x+1/4+1/2*y*3^(1/2))^2=0
o(6):  1/100-x^2-y^2=0
omega(x,y)=
y-23/100-(x^2+(y+1/5)^2)^(1/2)-((x+y+1/5-(x^2+(y+1/5)^2)^(1/2))^2+(2/5
-x)^2)^(1/2)-(x-1/5)^2-(y+1/5)^2-((y+3/5-(x^2+(y+1/5)^2)^(1/2)-((x+y+1
/5-(x^2+(y+1/5)^2)^(1/2))^2+(2/5-x)^2)^(1/2))^2+(4/25-(x-1/5)^2-(y+1/5
)^2)^2)^(1/2)-x^2-y^2+((y+19/25-(x^2+(y+1/5)^2)^(1/2)-((x+y+1/5-(x^2+(
y+1/5)^2)^(1/2))^2+(2/5-x)^2)^(1/2)-(x-1/5)^2-(y+1/5)^2-((y+3/5-(x^2+(
y+1/5)^2)^(1/2)-((x+y+1/5-(x^2+(y+1/5)^2)^(1/2))^2+(2/5-x)^2)^(1/2))^2
+(4/25-(x-1/5)^2-(y+1/5)^2)^2)^(1/2))^2+(1/100-x^2-y^2)^2)^(1/2)+25*(1
/2*(x-1/2)*3^(1/2)+1/2*y)^2+100*(-1/2*x+1/4+1/2*y*3^(1/2))^2-((y+77/10
0-(x^2+(y+1/5)^2)^(1/2)-((x+y+1/5-(x^2+(y+1/5)^2)^(1/2))^2+(2/5-x)^2)^
(1/2)-(x-1/5)^2-(y+1/5)^2-((y+3/5-(x^2+(y+1/5)^2)^(1/2)-((x+y+1/5-(x^2
+(y+1/5)^2)^(1/2))^2+(2/5-x)^2)^(1/2))^2+(4/25-(x-1/5)^2-(y+1/5)^2)^2)
^(1/2)-x^2-y^2+((y+19/25-(x^2+(y+1/5)^2)^(1/2)-((x+y+1/5-(x^2+(y+1/5)^
2)^(1/2))^2+(2/5-x)^2)^(1/2)-(x-1/5)^2-(y+1/5)^2-((y+3/5-(x^2+(y+1/5)^
2)^(1/2)-((x+y+1/5-(x^2+(y+1/5)^2)^(1/2))^2+(2/5-x)^2)^(1/2))^2+(4/25-
(x-1/5)^2-(y+1/5)^2)^2)^(1/2))^2+(1/100-x^2-y^2)^2)^(1/2))^2+(-1+25*(1
/2*(x-1/2)*3^(1/2)+1/2*y)^2+100*(-1/2*x+1/4+1/2*y*3^(1/2))^2)^2)^(1/2)
```

Далее нам нужно будет подставлять базисные функции и производные от них в подынтегральную функцию $F(x, y, z, z_x, z_y)$ и интегрировать по сложной области. Такое интегрирование мы будем проводить численно, по 1-точечной схеме: разобьем область решения на малые участки, в каждом из них вычислим $F(x, y, z, z_x, z_y)$ в центре тяжести, умножим на площадь участка и сложим. Разбивку на малые участки можно взять из МКЭ. Поэтому возьмем из своего варианта задания *главы 5* разбивку на треугольные конечные элементы. Проверим правильность построения функции $\omega(x, y)$: построим на одном графике область решения, полученную в *главе 5*, и значения $\omega(x, y)$ в нашей области. Если функция $\omega(x, y)$ построена правильно, то синие кружочки, которые мы строим там, где $\omega(x, y) > 0$, будут заполнять всю внутреннюю часть области решения, и только ее.

```
gd=[2;4;0;0.4;0.4;0;-0.2;-0.2;0.2;0.2]; % многоугольник
gd=[gd,[1;0.2;-0.2;0.4;0;0;0;0;0;0]]; % окружность
gd=[gd,[1;0;0;0.1;0;0;0;0;0;0]]; % окружность
gd=[gd,[4;0.5;0;0.2;0.1;pi/6;0;0;0;0]]; % эллипс
ns='abcd'; % строка соответствия
fo='(a*b+c)-d'; % формула операций
[dl,bt]=decsg(gd,fo,ns); % формируем область
[dl1,bt1]=csgdel(dl,bt); % удалили внутренние границы
[p,e,t]=initmesh(dl1); % формируем FEM-сетку
[p,e,t]=refinemesh(dl1,p,e,t); % измельчаем
```

```
p1=p(1,:); % x-е координаты узлов
p2=p(2,:); % y-е координаты узлов
xmin=min(p1);
xmax=max(p1); % границы
ymin=min(p2);
ymax=max(p2);
xi=linspace(xmin,xmax,40);
yk=linspace(ymin,ymax,40);
[XI,YK]=meshgrid(xi,yk); % построили сетку
ZIK=subs(omega,{x,y},{XI,YK}); % нашли omega(x,y)
ip=find(ZIK>=0); % номера точек, в которых omega(x,y)>=0
figure % новая фигура
pdegplot(dl1) % нарисовали область
hold on % рисуем на этом же графике
plot(XI(ip),YK(ip),'o') % точки, где omega(x,y)>=0
hold off
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
da=daspect; % текущие масштабы осей
da(1:2)=min(da(1:2)); % одинаковые масштабы
daspect(da); % установили одинаковые масштабы
title('\bfЗадание 17.2 - область решения')
xlabel('\itx') % ось OX
ylabel('\ity') % ось OY
```

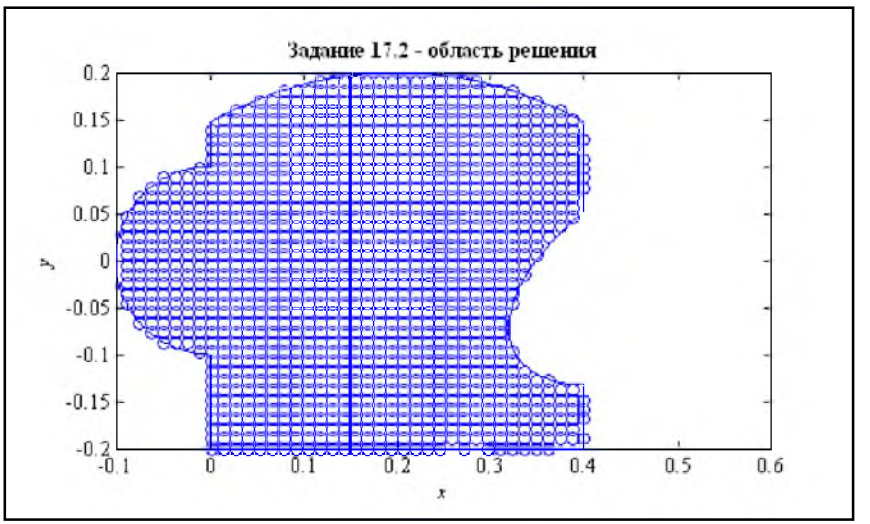


Рис. 17.8. Область решения в задании 17.2

В наших задачах функционал (5.1) имеет структуру:

$$J = \iint_D \left(c_x z_x^2 + c_y z_y^2 + a z^2 + f z \right) dS . \qquad (17.19)$$

Найдем параметры подынтегральной функции $F(x, y, z, z_x, z_y)$: коэффициенты при z, z^2, при квадратах частных производных.

```
cx=eval(subs(F,{Dzx,Dzy,z},{1,0,0})); % при Dzx^2
cy=eval(subs(F,{Dzx,Dzy,z},{0,1,0})); % при Dzy^2
dFdz=diff(F,z);
f=subs(dFdz,z,0); % коэффициент при z
a=eval((subs(dFdz,z,1)-f)/2); % коэффициент при z^2
disp('Параметры задачи:')
fprintf('Коэффициент при Dzx^2 cx=%d\n',cx)
fprintf('Коэффициент при Dzy^2 cy=%d\n',cy)
fprintf('Коэффициент при z      f=%s\n',char(f))
fprintf('Коэффициент при z^2    a=%d\n',a)
```

```
Параметры задачи:
Коэффициент при Dzx^2 cx=1
Коэффициент при Dzy^2 cy=2
Коэффициент при z      f=10*y*(sin(x)+1/5*x^2)
Коэффициент при z^2    a=0
```

Начинаем подготовку к численному интегрированию по области решения. Находим координаты центров тяжести треугольников, вычисляем $\omega(x, y)$ и ее частные производные в этих точках. Центр тяжести треугольника — это среднее арифметическое координат его вершин. Находим также площади всех треугольников по формуле

$$S = \frac{1}{2} \begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{vmatrix} . \qquad (17.20)$$

Если точки $M_1(x_1, y_1)$, $M_2(x_2, y_2)$ и $M_3(x_3, y_3)$ пронумерованы против часовой стрелки (а именно так функция `initmesh` нумерует узлы), то модуль в этой формуле не нужен.

```
xt=[p1(t(1,:));p1(t(2,:));p1(t(3,:))]; % x-е координаты
yt=[p2(t(1,:));p2(t(2,:));p2(t(3,:))]; % y-е координаты
xci=mean(xt);
yci=mean(yt); % центры тяжести треугольников
Fi=((xt(2,:)-xt(1,:)).*(yt(3,:)-yt(1,:))-...
  (xt(3,:)-xt(1,:)).*(yt(2,:)-yt(1,:)))/2; % площади
omc=subs(omega,{x,y},{xci,yci}); % omega в ц.т.
```

```
domcx=subs(domegadx,{x,y},{xci,yci}); % domegadx в ц.т.
domcy=subs(domegady,{x,y},{xci,yci}); % domegady в ц.т.
disp(['Вычислили omega(x,y) и ее производные '...
  'в центрах тяжести треугольников'])
```

```
Вычислили omega(x,y) и ее производные в центрах тяжести треугольников
```

Строим функцию $\varphi_0(x,y)$, удовлетворяющую граничным условиям. Мы не будем использовать формулу склейки (17.10), а поступим проще. У нас есть разбивка на треугольники. Вот мы и возьмем $\varphi_0(x,y)$ в следующем виде: в граничных узлах — какую нужно, а во всех внутренних узлах — нулевую, как показано на рис. 17.4. Эта функция будет состоять из треугольных кусочков плоскостей. Напомним, что уравнение плоскости, проходящей через 3 точки $M_1(x_1,y_1)$, $M_2(x_2,y_2)$ и $M_3(x_3,y_3)$, имеет вид (16.10). Частные производные записываются в виде (16.13). Вычисляем $\varphi_0(x,y)$ и частные производные от нее в центрах тяжести треугольников.

```
phi0=zeros(size(p1)); % вначале все 0-е
bcp=subs(bc,{x,y},{sym('p1'),sym('p2')}); % в условие
myb=eval(['find(' char(bcp) ')']); % нужные номера
if diff(zc,x)==0, % не зависит от x
  phi0(myb)=subs(zc,y,p2(myb)); % подставляем только y
elseif diff(zc,y)==0, % не зависит от y
  phi0(myb)=subs(zc,x,p1(myb)); % подставляем только x
else
  phi0(myb)=subs(zc,{x,y},{p1(myb),p2(myb)}); % подставляем x и y
end
phi0c=(phi0(t(1,:))+phi0(t(2,:))+phi0(t(3,:)))/3;
zt=[phi0(t(1,:));phi0(t(2,:));phi0(t(3,:))]; % z вершин
dphi0cx=((zt(2,:)-zt(1,:)).*(yt(3,:)-yt(1,:))-...
  (zt(3,:)-zt(1,:)).*(yt(2,:)-yt(1,:)))./(2*Fi);
dphi0cy=((xt(2,:)-xt(1,:)).*(zt(3,:)-zt(1,:))-...
  (xt(3,:)-xt(1,:)).*(zt(2,:)-zt(1,:)))./(2*Fi);
disp(['Вычислили phi0(x,y) и ее производные '...
  'в центрах тяжести треугольников'])
```

```
Вычислили phi0(x,y) и ее производные в центрах тяжести треугольников
```

Строим теперь базисные функции в виде произведения $\omega(x, y)$ на полиномы до 2-го порядка включительно. Вычисляем базисные функции и производные от них в центрах тяжести наших треугольников.

```
xcim=xci-xmin;
ycim=yci-ymin;
om(1,:)=omc;
```

```
om(2,:)=omc.*xcim;
om(3,:)=omc.*ycim;
om(4,:)=omc.*xcim.^2;
om(5,:)=omc.*xcim.*ycim;
om(6,:)=omc.*ycim.^2;
om(7,:)=phi0c;
domx(1,:)=domcx;
domx(2,:)=domcx.*xcim+omc;
domx(3,:)=domcx.*ycim;
domx(4,:)=xcim.*(domcx.*xcim+2*omc);
domx(5,:)=ycim.*(domcx.*xcim+omc);
domx(6,:)=domcx.*ycim.^2;
domx(7,:)=dphi0cx;
domy(1,:)=domcy;
domy(2,:)=domcy.*xcim;
domy(3,:)=domcy.*ycim+omc;
domy(4,:)=domcy.*xcim.^2;
domy(5,:)=xcim.*(domcy.*ycim+omc);
domy(6,:)=ycim.*(domcy.*ycim+2*omc);
domy(7,:)=dphi0cy;
disp(['Вычислили базисные функции и их производные '...
  'в центрах тяжести треугольников'])
```

```
Вычислили базисные функции и их производные
  в центрах тяжести треугольников
```

Разберемся теперь, как будет происходить интегрирование по области решения D и вычисление частных производных от полученного функционала по коэффициентам α_k. Наш функционал имеет структуру (17.19). Подставим в него вид нашего решения (17.2) (все базисные функции, конечно, будут зависеть от x и y).

$$J = \iint\limits_D \left(c_x \left(\left(\sum_{k=1}^{n} \alpha_k \frac{\partial \varphi_k}{\partial x} \right)^2 + 2\frac{\partial \varphi_0}{\partial x} \sum_{k=1}^{n} \alpha_k \frac{\partial \varphi_k}{\partial x} + \left(\frac{\partial \varphi_0}{\partial x} \right)^2 \right) + \right.$$

$$+ c_y \left(\left(\sum_{k=1}^{n} \alpha_k \frac{\partial \varphi_k}{\partial y} \right)^2 + 2\frac{\partial \varphi_0}{\partial y} \sum_{k=1}^{n} \alpha_k \frac{\partial \varphi_k}{\partial y} + \left(\frac{\partial \varphi_0}{\partial y} \right)^2 \right) + \qquad (17.21)$$

$$\left. + a \left(\left(\sum_{k=1}^{n} \alpha_k \varphi_k \right)^2 + 2\varphi_0 \sum_{k=1}^{n} \alpha_k \varphi_k + \varphi_0^2 \right) + f \left(\sum_{k=1}^{n} \alpha_k \varphi_k + \varphi_0 \right) \right) dS \,.$$

Квадрат каждой суммы — это двойная сумма:

$$J = \iint_D \left(c_x \left(\sum_{i=1}^{n}\sum_{k=1}^{n} \alpha_i \alpha_k \frac{\partial \varphi_i}{\partial x}\frac{\partial \varphi_k}{\partial x} + 2\frac{\partial \varphi_0}{\partial x}\sum_{k=1}^{n}\alpha_k \frac{\partial \varphi_k}{\partial x} + \left(\frac{\partial \varphi_0}{\partial x}\right)^2 \right) + \right.$$
$$+ c_y \left(\sum_{i=1}^{n}\sum_{k=1}^{n} \alpha_i \alpha_k \frac{\partial \varphi_i}{\partial y}\frac{\partial \varphi_k}{\partial y} + 2\frac{\partial \varphi_0}{\partial y}\sum_{k=1}^{n}\alpha_k \frac{\partial \varphi_k}{\partial y} + \left(\frac{\partial \varphi_0}{\partial y}\right)^2 \right) + \qquad (17.22)$$
$$\left. + a\left(\sum_{i=1}^{n}\sum_{k=1}^{n} \alpha_i \alpha_k \varphi_i \varphi_k + 2\varphi_0 \sum_{k=1}^{n}\alpha_k \varphi_k + \varphi_0^2 \right) + f\left(\sum_{k=1}^{n}\alpha_k \varphi_k + \varphi_0 \right) \right) dS .$$

Дифференцируем по α_k. После сокращения на 2 получим:

$$\frac{\partial J}{\partial \alpha_k} = \iint_D \left(c_x \left(\sum_{i=1}^{n} \alpha_i \frac{\partial \varphi_i}{\partial x}\frac{\partial \varphi_k}{\partial x} + \frac{\partial \varphi_k}{\partial x}\frac{\partial \varphi_0}{\partial x} \right) + \right.$$
$$\left. + c_y \left(\sum_{i=1}^{n} \alpha_i \frac{\partial \varphi_i}{\partial y}\frac{\partial \varphi_k}{\partial y} + \frac{\partial \varphi_k}{\partial y}\frac{\partial \varphi_0}{\partial y} \right) + a\left(\sum_{i=1}^{n} \alpha_i \varphi_i \varphi_k + \varphi_0 \varphi_k \right) + \frac{f}{2}\varphi_k \right) dS = 0 . \qquad (17.23)$$

Мы получили систему линейных алгебраических уравнений. Коэффициенты при неизвестных:

$$a_{ik} = \iint_D \left(c_x \frac{\partial \varphi_i}{\partial x}\frac{\partial \varphi_k}{\partial x} + c_y \frac{\partial \varphi_i}{\partial y}\frac{\partial \varphi_k}{\partial y} + a\varphi_i \varphi_k \right) dS ; \qquad (17.24)$$

а правые части

$$b_k = -\iint_D \left(c_x \frac{\partial \varphi_0}{\partial x}\frac{\partial \varphi_k}{\partial x} + c_y \frac{\partial \varphi_0}{\partial y}\frac{\partial \varphi_k}{\partial y} + a\varphi_0 \varphi_k + \frac{f}{2}\varphi_k \right) dS . \qquad (17.25)$$

Вычисляем коэффициенты нашей системы уравнений и правые части по (17.24—17.25). Интегрирование по области D выполняем численно, по 1-точечной схеме: вычисляем соответствующие функции в центрах тяжести наших треугольников, умножаем на площади треугольников и суммируем. Решаем полученную систему уравнений и печатаем решения.

```
A=zeros(6);
b=zeros(6,1);
fc=subs(f,{x,y},{xci,yci})/2;
for ii=1:6,
  for kk=ii:6,
    A(ii,kk)=sum((cx*domx(ii,:).*domx(kk,:)+...
      cy*domy(ii,:).*domy(kk,:)+a*om(ii,:).*om(kk,:)).*Fi);
  end
```

```
  b(ii)=-sum(fc.*om(ii,:).*Fi)-sum((cx*domx(7,:).*domx(ii,:)+...
    cy*domy(7,:).*domy(ii,:)+a*om(7,:).*om(ii,:)).*Fi);
end
A1=A+(triu(A,1))'; % симметризовали
alpha=[A1\b;1]; % находим коэффициенты, последний =1
disp('Коэффициенты:')
fprintf('alpha(%d)=%14.10f\n',[[1:7];alpha'])
```

```
Коэффициенты:
alpha(1)= -0.0056773337
alpha(2)=  0.0390428305
alpha(3)=  0.0347633938
alpha(4)= -0.0232625486
alpha(5)= -0.1596577514
alpha(6)= -0.0435888636
```

Вычисляем все базисные функции в узлах сетки, находим решение и рисуем его (рис. 17.9).

```
pomc=subs(omega,{x,y},{p1,p2}); % omega в узлах
pom(1,:)=pomc;
pom(2,:)=pomc.*(p1-xmin);
pom(3,:)=pomc.*(p2-ymin);
pom(4,:)=pomc.*(p1-xmin).^2;
pom(5,:)=pomc.*(p1-xmin).*(p2-ymin);
pom(6,:)=pomc.*(p2-ymin).^2;
pom(7,:)=phi0; % phi0 в узлах
w=alpha'*pom; % решение
figure % новая фигура
pdeplot(p,e,t,'xydata',w,'zdata',w,...
  'mesh','on','colorbar','off'); % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
v=axis; % границы осей
da=daspect; % масштаб осей
da(1:2)=min(da(1:2)); % min из двух
daspect(da) % выравниваем масштаб осей
axis(v); % оставили границы
colormap default % палитра по умолчанию
grid on % показали сетку
box on % показали внешний контур
title('\bfЗадание 17.2 - решение') % заголовок
xlabel('\itx')
ylabel('\ity')
zlabel('\itz\rm(\itx\rm,\ity\rm)') % метки осей
```

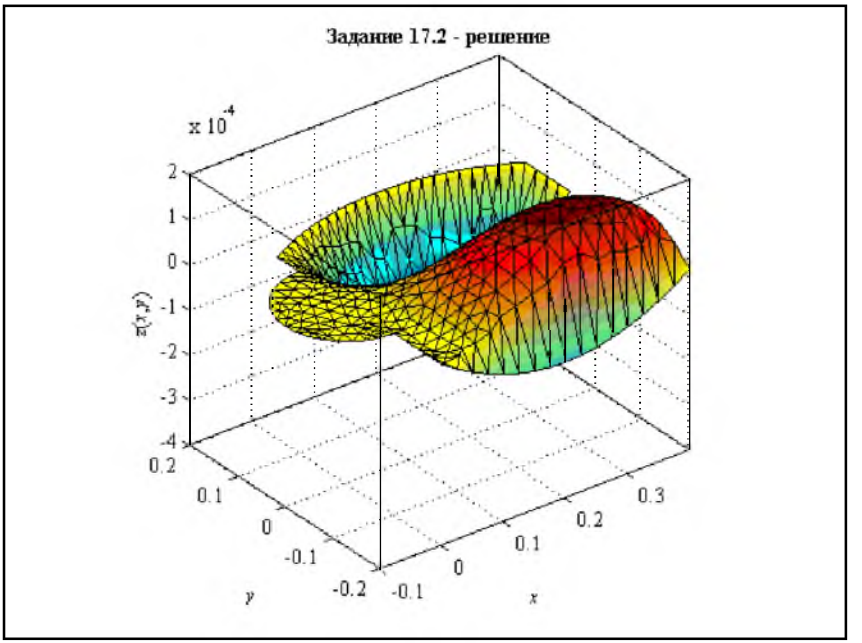
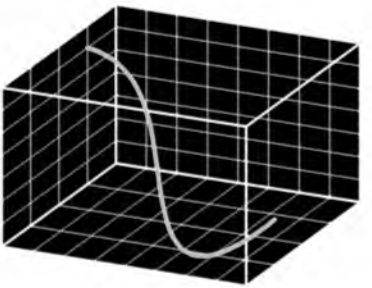


Рис. 17.9. Решение задания 17.2, выполненное с помощью MATLAB

17.6. Задание

1. Решить *задание 1 главы 2* методом Ритца. Сравнить решение с аналитическим.
2. Решить *задание главы 5* методом Ритца. Сравнить решение с решением, полученным МКР и МКЭ. Удачно ли выбраны базисные функции?

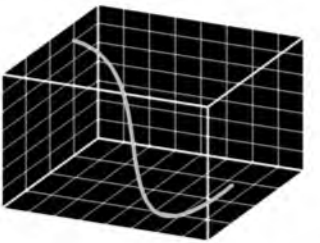


ЧАСТЬ II

Математическая статистика

Глава 18. Основы выборочного метода
Глава 19. Доверительные оценки параметров распределения
Глава 20. Оценки генеральных параметров распределения
Глава 21. Анализ закона распределения
Глава 22. Сравнение выборок
Глава 23. Дисперсионный анализ
Глава 24. Метод наименьших квадратов
Глава 25. Корреляционный анализ
Глава 26. Генерация вариантов заданий
Глава 27. Функции пакета Statistics Toolbox

ГЛАВА 18



Основы выборочного метода

Мы переходим к *части II* книги — элементам математической статистики. При решении задач будем стараться в максимальной степени использовать MATLAB. В нем есть специальный инструментарий [57] для решения задач статистики.

18.1. Генеральная совокупность и выборка

По своей сути математическая статистика — это применение понятий, методов и результатов теории вероятностей к обработке экспериментальных данных. Мы должны на основе данных исследований делать научно обоснованные выводы. Рассмотрим примеры.

ПРИМЕР 18.1. На заводе выпущена партия из N изделий, имеющих определенный срок эксплуатации. Из нее отобрано n изделий ($n << N$), для которых найдено время их работы до отказа: t_1, t_2, ..., t_n. Можно ли эти результаты перенести на все N изделий? □

ПРИМЕР 18.2. Каждый день в магазине фиксируется количество зашедших в него посетителей. Пусть в 1-й день в магазин зашло x_1 человек, во 2-й — x_2, ..., в n-й — x_n. Какие выводы мы можем сделать по этим данным о случайной величине X — количестве посетителей за 1 день? □

ПРИМЕР 18.3. Физик-экспериментатор измеряет интервал времени между двумя последовательными распадами частиц. Всего он измерил n значений этой величины: t_1, t_2, ..., t_n. Что можно сказать о случайной величине T — интервале времени между двумя последовательными распадами? □

В 1-м примере можно, например, провести исследование всей партии из N изделий. Но, во-первых, эта партия может быть очень большой, и исследование влетит в копеечку. А во-вторых, контроль может быть разрушающим, и

тогда исследовать всю партию вообще нет смысла. Иначе мы уподобимся Ходже Насреддину, которого жена послала за спичками, но предупредила: "Обязательно проверь, чтобы они были сухими". И наш герой проверил их все!

А вот во 2-м и 3-м примерах мы даже теоретически не можем исследовать всех покупателей или все элементарные частицы. Поэтому мы должны *выбрать* из исследуемой величины данные, исследовать их, а результаты обобщить на исходную случайную величину. В математической статистике исходная исследуемая случайная величина называется генеральной совокупностью, а полученный из нее набор экспериментальных данных — выборочной совокупностью, или выборкой.

Определение 18.1. *Генеральной совокупностью* называется совокупность всех возможных значений исследуемой случайной величины X, т. е. сама эта величина. □

Определение 18.2. *Выборочной совокупностью или выборкой* называется совокупность тех значений случайной величины X, которые есть в нашем распоряжении для исследования. □

Определение 18.3. *Выборочным методом исследования* называется исследование выборки и перенесение его результатов на генеральную совокупность. □

Первый вопрос, который возникает из этих определений: можно ли вообще применять выборочный метод? Насколько обоснованным является перенесение результатов выборки на генеральную совокупность? Чтобы ответить на этот вопрос, введем в рассмотрение еще некоторые принципы и определения.

Пусть исследуется случайная величина X с функцией распределения $F(x)$. Из нее взята выборка объема n: $x_1, x_2, \ldots, x_n$. Будем считать, что эти числа упорядочены по возрастанию: $x_1 \le x_2 \le \ldots \le x_n$. Построим по этим числам функцию, которую назовем *выборочной функцией распределения* $F^*(x)$:

$$F^*(x) = \begin{cases} 0; & x < x_1; \\ \dfrac{1}{n}; & x_1 \le x < x_2; \\ \dfrac{2}{n}; & x_2 \le x < x_3; \\ \ldots & \ldots \\ \dfrac{n-1}{n}; & x_{n-1} \le x < x_n; \\ 1; & x \ge x_n. \end{cases} \qquad (18.1)$$

(Здесь и далее все результаты, относящиеся к выборке, будем обозначать звездочкой вверху.)

Выборочная функция распределения $F^*(x)$ состоит из n ступенек высотой $1/n$, проведенных в точках с абсциссами x_i, $i \in [1, n]$. Если среди x_i есть одинаковые, то формула (18.1) очевидным образом модифицируется: несколько ступенек сливаются в одну.

Определение 18.4. *Выборочной функцией распределения* случайной величины X, соответствующей упорядоченной выборке $x_1, x_2, ..., x_n$, называется функция (18.1). □

Почему мы строим выборочную функцию распределения именно по формуле (18.1), а не по какой-либо другой? Ответ на этот вопрос дает *принцип максимума правдоподобия* (в дальнейшем ПМП), который отражает многовековой опыт человечества и выглядит так:

> ✔ На практике чаще всего происходят события с максимальными вероятностями.

ПРИМЕР 18.4. Сколько бы мы ни бросали монету на гладкий стол, она почти всегда ляжет плашмя и почти никогда не станет на ребро. □

Давайте теперь применим ПМП к нашей задаче — построению выборочной функции распределения. Пусть мы ничего не знаем о законе распределения исходной величины X, а имеем только упорядоченную (по возрастанию) выборку из нее: $x_1, x_2, ..., x_n$. Согласно ПМП мы должны считать, что величина X именно такая, какой она оказалась на практике: ведь на практике должны проявляться события с максимальными вероятностями! На практике по одному разу проявились значения $x_1, x_2, ..., x_n$, и не было никаких других значений. Поэтому и мы должны считать, что величина X — дискретная, n-значная, с возможными значениями $x_1, x_2, ..., x_n$ (теми, что проявились на практике!) и одинаковыми вероятностями их появления $p_i = 1/n$. А для такой величины функция распределения как раз и имеет вид (18.1). Процесс построения $F^*(x)$ показан на рис. 18.1.

Как ведет себя $F^*(x)$ с ростом объема выборки? Оказывается, что каким бы ни было распределение исходной величины X, с увеличением n выборочная функция распределения $F^*(x)$ сходится по вероятности к функции распределения генеральной совокупности $F(x)$ (которая так и называется — генеральная функция распределения).

■ **ТЕОРЕМА 18.1 Гливенко — Кантелли.** Для любого непрерывного распределения при $n \to \infty$ максимальная по модулю разность между выборочной

функцией распределения $F^*(x)$ и генеральной $F(x)$ имеет предел по вероятности, равный нулю:

$$\lim_{n\to\infty} \max_{\forall x\in R} \left| F^*(x) - F(x) \right| = 0, \tag{18.2}$$

где под пределом понимается предел по вероятности.

Доказательство можно найти, например, на сайте [46]. □

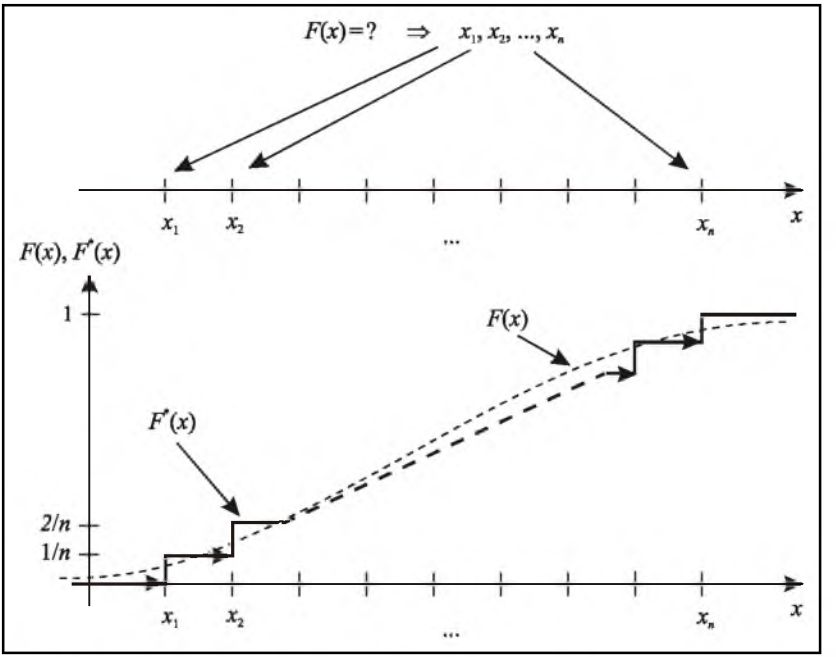


Рис. 18.1. Применение ПМП к построению выборочной функции распределения

Далее, в *главе 21* мы рассмотрим еще одну теорему — Колмогорова, которая уточняет теорему Гливенко — Кантелли. Теорема Колмогорова устанавливает скорость сходимости к нулю максимальной по модулю разности между выборочной и генеральной функциями распределения. Там же мы научимся строить $F^*(x)$ средствами MATLAB.

Смысл теоремы Гливенко — Кантелли такой. Если мы будем увеличивать объем выборки n, то $F^*(x)$ будет иметь все больше и больше ступенек, а высота каждой ступеньки будет все меньше и меньше. В пределе выборочная функция распределения $F^*(x)$ будет сглаживаться и стремиться к генеральной функции распределения $F(x)$, которая показана на рис. 18.1 тонкой штриховой линией.

Эта теорема дает нам в руки мощный инструмент исследования — выборочный метод. Мы можем быть уверены, что, увеличивая объем выборки, будем делать все более и более точные выводы о поведении генеральной совокупности. Ведь через функцию распределения можно вычислить любые другие характеристики случайной величины: математическое ожидание, дисперсию и пр. Нужно только следить, чтобы выборка была действительно случайной и независимой. Статистики говорят, что выборка должна быть *репрезентативной* или *представительной*, т. е. представлять все возможные значения генеральной совокупности и примерно в том же соотношении, что в исходной величине X.

ПРИМЕР 18.5. Нельзя из полученных значений x_1, x_2, ..., x_n отобрать несколько первых (минимальных) или несколько средних, т. к. в первом случае исказится математическое ожидание, а во втором — дисперсия. □

ПРИМЕР 18.6. Во время предвыборных социологических исследований социологи одного кандидата проводят опросы в районе студенческих общежитий, другого — у заводских проходных, третьего — в пригородных электричках, а четвертого — по телефону. В результате каждый из кандидатов, полагаясь на данные своих политтехнологов, уверен в своей победе и не предпринимает никаких усилий для завоевания сторонников в иной социальной среде. □

18.2. Оценки и требования к ним

По генеральной функции распределения $F(x)$ мы определяем числовые параметры распределения: математическое ожидание m_x, дисперсию D_x и др.

Определение 18.5. *Генеральными параметрами* называются числовые параметры генеральной совокупности. □

Мы так и будем говорить: генеральное математическое ожидание (среднее), генеральный эксцесс и т. д.

Определение 18.6. *Выборочными параметрами* называются числовые параметры выборки. □

В некоторых книгах выборочные параметры называются *точечными оценками*, т. к. каждый из них представляет из себя одно число (одну точку). Будем их обозначать, как и выборочную функцию распределения, звездочкой вверху, т. е. m_x^* — выборочное математическое ожидание, σ_x^* — выборочное среднеквадратичное отклонение и т. д. Мы их находим из выборочной функции распределения $F^*(x)$, т. е. в конечном счете они зависят от экспериментальных данных x_1, x_2, ..., x_n и объема выборки n.

Определение 18.7. Выборочные параметры называются *оценками* соответствующих генеральных параметров. □

Вычислив, например, D_x^*, мы тем самым оцениваем с какой-то степенью точности D_x. Поэтому формулы для вычисления выборочных параметров через x_1, x_2, ..., x_n, n должны быть "правильными", чтобы мы действительно получали то, что нужно. Сейчас мы с помощью выборочного метода выясним, что такое "правильная" формула. Рассмотрим задачу в общем виде. Пусть есть некоторый генеральный параметр b, и мы хотим по выборке x_1, x_2, ..., x_n найти его "правильную" оценку b^*. Какой должна быть формула для ее вычисления? Обозначим эту формулу буквой φ:

$$b^* = \varphi(x_1, x_2, ..., x_n). \tag{18.3}$$

Объем выборки n явно здесь не записан, но он автоматически определяется количеством аргументов. Каким же требованиям должна удовлетворять формула φ? У нас все x_i — независимые, и каждое из них является реализацией генеральной совокупности X. Это эквивалентно тому, что каждое x_i является единственной реализацией случайной величины X_i, а все X_i независимые и имеют тот же закон распределения, что и X. Такой переход позволяет считать выборочный параметр b^* реализацией случайной величины B^*, которая является функцией n случайных величин X_1, X_2, ..., X_n, и вычисляется по той же формуле φ:

$$B^* = \varphi(X_1, X_2, ..., X_n), \tag{18.4}$$

где все X_i независимы и имеют одинаковое распределение, такое же, как у исходной величины X.

Закон распределения B^* зависит:

- от законов распределения X_i, т. е. от закона распределения X;
- от количества величин n;
- от вида функции φ.

Отсюда мы можем вывести требования к виду функции φ. Сформулируем их.

Требование 18.1. Оно вытекает из теоремы Гливенко — Кантелли. Мы должны быть уверены, что при $n \to \infty$ случайная величина B^* сходится по вероятности к детерминированной величине b, т. е. к тому генеральному параметру, который мы оцениваем:

$$\lim_{n\to\infty} B^* = b. \tag{18.5}$$

Предел здесь — по вероятности. Это требование называется *состоятельностью*. □

На рис. 18.2 показано, как должна вести себя плотность распределения $f(b^*)$ случайной величины B^* с увеличением n: график $f(b^*)$ должен "сплющиваться" по горизонтали и растягиваться по вертикали, все более и более приближаясь к δ-функции. А центр распределения должен при этом приближаться к b.

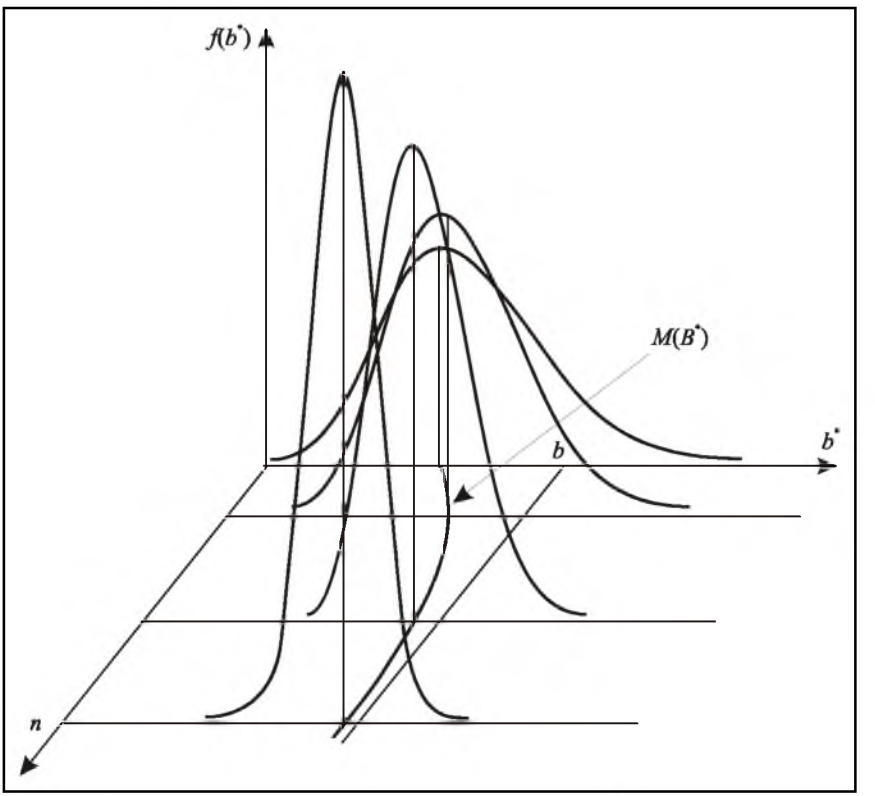


Рис. 18.2. Состоятельность оценки b^*

Можно показать, что условие (18.5) эквивалентно выполнению двух условий:

$$\begin{cases} \lim\limits_{n\to\infty} M\left(B^*\right) = b; \\ \lim\limits_{n\to\infty} D\left(B^*\right) = 0; \end{cases} \tag{18.6}$$

где пределы — обычные (не по вероятности). На рис. 18.2 мы как раз и наблюдаем, что с ростом n математическое ожидание B^* приближается к b, а дисперсия B^* уменьшается. Для доказательства этого факта вначале сформу-

лируем и докажем одну важную лемму, которой мы будем пользоваться и в дальнейшем.

■ **ЛЕММА 18.1 — неравенство Чебышева** (Пафнутий Львович Чебышев, 1821—1894, рис. 18.3). Вероятность того, что случайная величина X будет отклоняться от своего математического ожидания m_x на величину, больше или равную $\eta > 0$, не превышает дисперсии этой величины D_x, деленной на η^2:

$$\mathbf{P}\left(\left|X - m_x\right| \geq \eta\right) < \frac{D_x}{\eta^2}. \tag{18.7}$$

Рис. 18.3. П. Л. Чебышев

Доказательство (для непрерывной случайной величины). Запишем выражение для дисперсии и разобьем весь интервал интегрирования $(-\infty, +\infty)$ на 3 участка:

$$D_x = \int_{-\infty}^{+\infty} (x - m_x)^2 f(x)dx = \int_{-\infty}^{m_x-\eta} (x - m_x)^2 f(x)dx + \\ + \int_{m_x-\eta}^{m_x+\eta} (x - m_x)^2 f(x)dx + \int_{m_x+\eta}^{+\infty} (x - m_x)^2 f(x)dx. \tag{18.8}$$

Здесь $f(x)$ — плотность распределения величины X. Каждый из интегралов в правой части — неотрицательный. Поэтому, если отбросить среднее слагаемое, получим неравенство:

$$\int_{-\infty}^{m_x-\eta} (x - m_x)^2 f(x)dx + \int_{m_x+\eta}^{+\infty} (x - m_x)^2 f(x)dx \leq D_x. \tag{18.9}$$

На оставшихся интервалах интегрирования $|x-m_x|\ge\eta$. Поэтому если заменить первый множитель под интегралом меньшей величиной η^2, то неравенство только усилится:

$$\eta^2\left(\int\limits_{-\infty}^{m_x-\eta} f(x)dx+\int\limits_{m_x+\eta}^{+\infty} f(x)dx\right)\le D_x. \qquad (18.10)$$

Но теперь каждый из интегралов в скобках — вероятность попадания в соответствующий интервал. Отсюда получаем (18.7). □

Теперь, используя неравенство Чебышева, сформулируем следующую теорему.

■ **ТЕОРЕМА 18.2.** Для состоятельности оценки B^* (18.5) необходимо и достаточно выполнение условий (18.6).

Доказательство. *Достаточность.* Пусть условия (18.6) выполняются. Докажем, что для любых сколь угодно малых заданных наперед положительных чисел ε и δ существует такой объем выборки N, начиная с которого вероятность отклонения случайной величины B^* от числа b на величину, большую или равную ε, меньше δ:

$$\mathbf{P}\left(\left|B^*-b\right|\ge\varepsilon\right)<\delta. \qquad (18.11)$$

Неравенство (18.11) — это и есть предел по вероятности (18.5), сформулированный в терминах теории пределов. Для доказательства (18.11) зададим сколь угодно малые положительные числа ε и δ. Вычислим по ним $\eta=\dfrac{2\varepsilon}{3}$, $\varepsilon_1=\dfrac{\eta}{2}=\dfrac{\varepsilon}{3}$ и $\varepsilon_2=\dfrac{4\varepsilon^2\delta}{9}$. Соотношение между ε, η и ε_1 показано на рис. 18.4.

Из 1-го предела (18.6) следует, что для выбранного нами ε_1 существует такой объем выборки N_1, начиная с которого $|M(B^*)-b|\le\varepsilon_1$. А из 2-го предела (18.6) следует, что для нашего ε_2 существует такой объем выборки N_2, начиная с которого $D(B^*)\le\varepsilon_2$. Выберем теперь $N=\max(N_1,N_2)$ и рассмотрим поведение случайной величины B^* для $n>N$. По неравенству Чебышева

$$\mathbf{P}\left(\left|B^*-M\left(B^*\right)\right|\ge\eta\right)<\frac{D\left(B^*\right)}{\eta^2}\le\frac{\varepsilon_2}{\eta^2}=\delta. \qquad (18.12)$$

Но, благодаря удачно выбранным ε, η и ε_1, область $|B^*-b|\ge\varepsilon$ является подобластью области $|B^*-M(B^*)|\ge\eta$, поэтому вероятность попадания туда также меньше δ.

Необходимость докажите самостоятельно. □

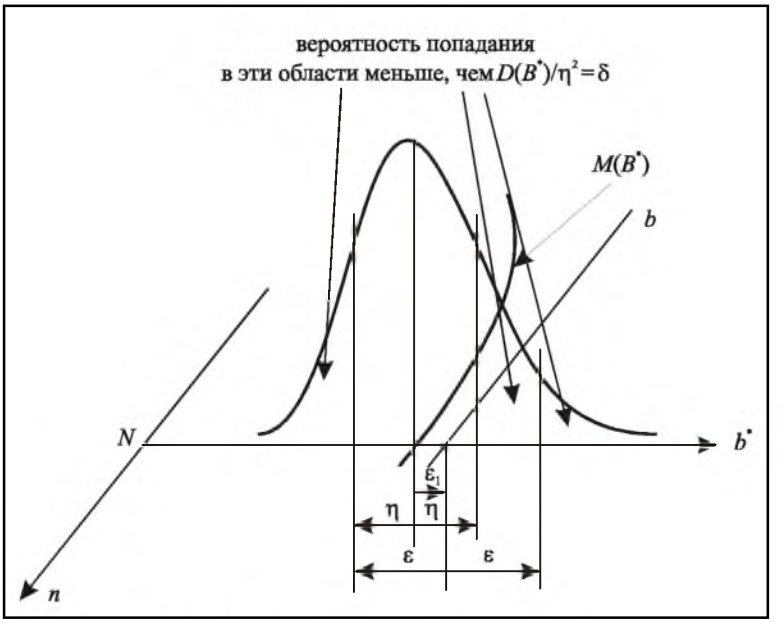


Рис. 18.4. К доказательству теоремы 18.2

Состоятельность является обязательным требованием. Если оно не выполняется, то вид функции φ подобран неправильно. В этом случае нужно попытаться подобрать другую функцию φ, такую, чтобы вычисленная по (18.4) B^* удовлетворяла (18.6).

Требование 18.2. На практике объем выборки n ограничен, и хорошо было бы, если бы вместо $M(B^*) \to b$ (первый предел в (18.6)) мы имели:

$$M(B^*) = b. \tag{18.13}$$

Тогда мы бы не допускали при измерениях систематических ошибок, а все погрешности носили бы случайный характер. Это требование называется *несмещенностью*. □ Оно показано на рис. 18.5. С ростом n график $f(b^*)$ не только "сплющивается" по вертикали, но и не сдвигается по горизонтали.

Несмещенность — более жесткое требование, чем состоятельность, и не всегда удается ему удовлетворить. Но по мере возможности мы будем строить именно несмещенные оценки.

Требование 18.3. Из всех состоятельных и несмещенных оценок нужно выбирать ту, у которой при каждом n минимальная дисперсия. Такое требование называется *эффективностью*. □ Оно проиллюстрировано на рис. 18.6. Оценка, плотность распределения которой показана сплошной ли-

нией, более эффективна, чем оценка, плотность распределения которой изображена штриховой линией.

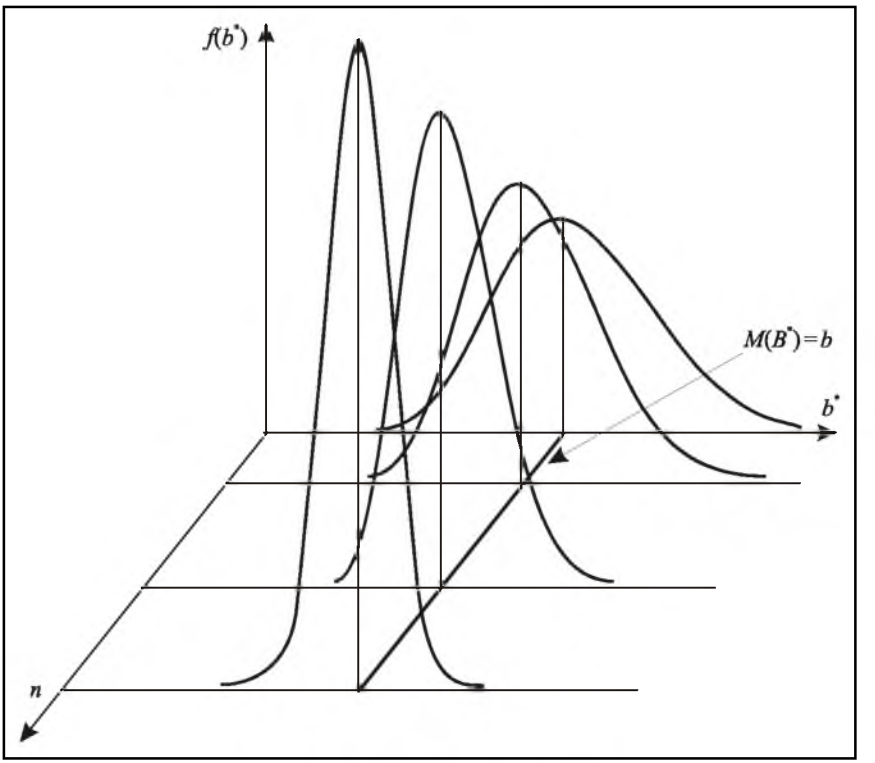


Рис. 18.5. Несмещенность оценки b^*

Требование эффективности — еще более жесткое, чем несмещенность. Может вообще так оказаться, что при одних n эффективна одна оценка, а при других — другая. Но если уж у нас будет возможность выбирать из нескольких оценок (формул φ) одну, то мы будем выбирать более эффективную.

Рассмотрев общие требования к оценкам, выведем теперь оценки конкретных числовых параметров распределения. Попутно мы начнем выполнять индивидуальное расчетное задание по теме "Обработка массива данных". Введем исходные данные. Будем предполагать, что все измерения x_i записаны в обычном текстовом формате в виде прямоугольной таблицы чисел. Для разделения целой и дробной частей используется десятичная точка. Между числами оставлено хотя бы по одному пробелу. При необходимости можно перед числом поставить знак плюс или минус. Допускается также экспоненциальная форма записи чисел, т. е. числа вида –1,52345E–0002 или

–1.52345e–0002, что нужно понимать как $-1{,}52345\times10^{-2}$. Эту операцию вы должны проделать до обработки данных. Можно использовать обычные текстовые редакторы, например, Notepad или Aditor. Несколько образцов правильного оформления файлов исходных данных имеется в электронной версии книги, размещенной на диске. Для начала их можно использовать как тестовые примеры, а потом взять свои.

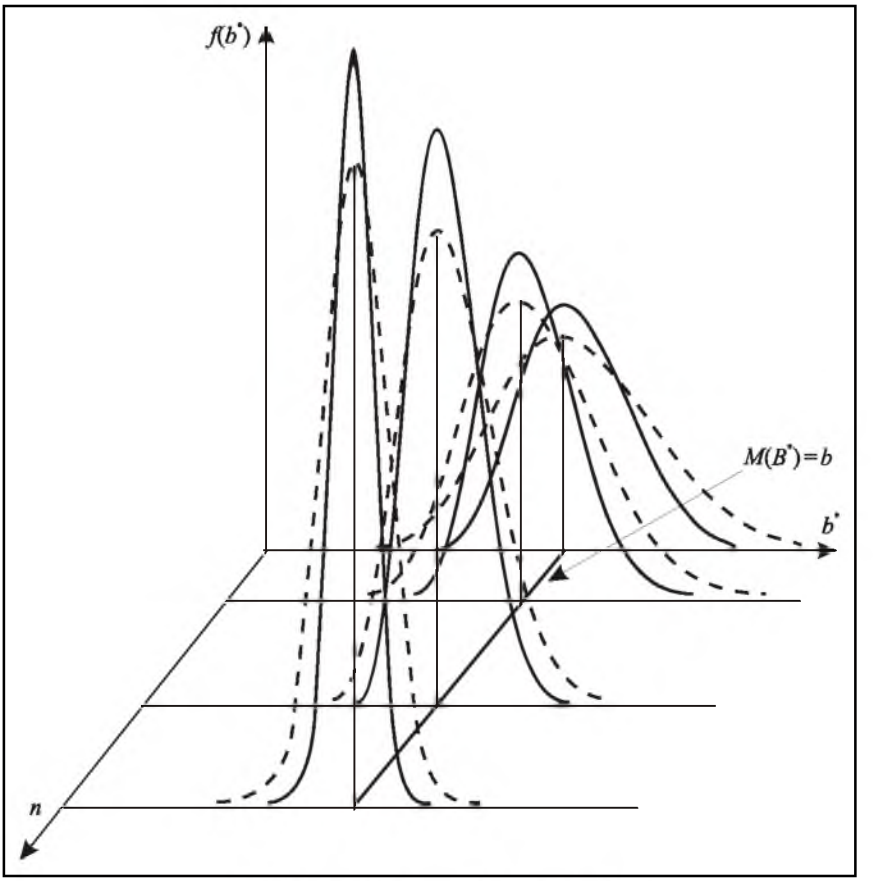


Рис. 18.6. Эффективность оценки b^*

В нижеприведенной области ввода идентификатором `sf` обозначена строка с полным именем файла данных. Поставьте здесь имя своего файла. Введем данные в программу и обозначим их идентификатором `x`. После ввода переформатируем данные в вектор-столбец. Рассортируем данные в порядке возрастания, найдем минимальное x_{min} и максимальное x_{max} значения. Определим количество данных n.

```
clear all % очистили рабочую область
sf='D:\Iglin\Matlab\ContData\xrayl.txt'; % ИМЯ ФАЙЛА
x=load(sf); % вводим ИД
x=sort(x(:)); % переформатировали столбец и рассортировали
n=length(x); % количество данных
xmin=x(1); % минимальное значение
xmax=x(n); % максимальное значение
fprintf('Обрабатываем файл %s\n',sf)
fprintf('Объем выборки n=%d\n',n)
fprintf('xmin=%14.7f\n',xmin)
fprintf('xmax=%14.7f\n',xmax)
Обрабатываем файл D:\Iglin\Matlab\ContData\xrayl.txt
Объем выборки n=200
xmin=     0.0281344
xmax=     5.4383751
```

18.3. Оценка математического ожидания

Пусть проведено n измерений одной случайной величины X. Мы предполагаем, что при измерениях избежали грубых и систематических ошибок, поэтому разброс экспериментальных данных x_1, x_2, ..., x_n вызывается только случайными ошибками. По этим данным требуется найти оценку математического ожидания m_x, т. е. выборочное среднее m_x^*. По какой формуле его нужно вычислять?

Применим рассуждения, изложенные в *разделе 18.2*, к нашей задаче. Согласно ПМП мы должны считать, что случайная величина X — дискретная, n-значная и принимает свои возможные значения x_1, x_2, ..., x_n с одинаковыми вероятностями $p_i = 1/n$. Математическое ожидание такой величины равно:

$$m_x^* = \sum_{i=1}^{n} x_i p_i = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad (18.14)$$

т. е. должно вычисляться как среднее арифметическое. Проверим этот результат на состоятельность и несмещенность (если бы были и другие формулы, мы бы еще и выбрали наиболее эффективную). В соответствии с (18.3—18.4) считаем, что m_x^* — реализация случайной величины M_x^*, которая вычисляется по той же формуле:

$$M_x^* = \frac{1}{n}\sum_{i=1}^{n} X_i, \qquad (18.15)$$

но X_i — уже не экспериментальные данные, а независимые случайные величины, имеющие тот же закон распределения, что и X. Найдем параметры случайной величины M_x^*.

Ее математическое ожидание:

$$M\left(M_x^*\right)=M\left(\frac{1}{n}\sum_{i=1}^{n}X_i\right)=\frac{1}{n}\sum_{i=1}^{n}M\left(X_i\right)=\frac{1}{n}\sum_{i=1}^{n}m_x=m_x, \qquad (18.16)$$

т. е. оценка математического ожидания — *несмещенная*. При вычислении $M(M_x^*)$ мы использовали известные свойства математического ожидания: выносили постоянный множитель за знак операции вычисления математического ожидания и расписывали математическое ожидание суммы случайных величин как сумму математических ожиданий слагаемых. Найдем теперь дисперсию M_x^*:

$$D\left(M_x^*\right)=D\left(\frac{1}{n}\sum_{i=1}^{n}X_i\right)=\frac{1}{n^2}D\left(\sum_{i=1}^{n}X_i\right)=\frac{1}{n^2}\sum_{i=1}^{n}D_x=\frac{D_x n}{n^2}=\frac{D_x}{n}. \qquad (18.17)$$

Здесь по свойствам дисперсии постоянный множитель выносится в квадрате, а дисперсия суммы *независимых* случайных величин равна сумме дисперсий. Проверяем (18.6):

$$\begin{cases}\lim\limits_{n\to\infty}M\left(M_x^*\right)=\lim\limits_{n\to\infty}m_x=m_x;\\ \lim\limits_{n\to\infty}D\left(M_x^*\right)=\lim\limits_{n\to\infty}\dfrac{D_x}{n}=0.\end{cases} \qquad (18.18)$$

Оба условия выполняются, поэтому оценка (18.14) является не только несмещенной, но и *состоятельной*. Вот почему независимо от закона распределения генеральной совокупности X математическое ожидание выборки можно вычислять по формуле (18.14). В MATLAB вычисление среднего арифметического реализуется функцией `mean`. Вычислим математическое ожидание нашей выборки.

```
Mx=mean(x); % математическое ожидание
fprintf('Выборочное математическое ожидание Mx=%14.7f\n',Mx)
Выборочное математическое ожидание Mx=      1.9607780
```

18.4. Оценка дисперсии

Постановка задачи та же: есть n независимых измерений случайной величины X без грубых и систематических ошибок. По ним требуется найти выборочную дисперсию D_x^*. Как ее вычислять?

Как и в предыдущем *разделе 18.3*, применим ПМП. Считаем, что случайная величина X — дискретная, n-значная и принимает свои возможные значения

x_1, x_2, ..., x_n с одинаковыми вероятностями $p_i = 1/n$. Для такой величины дисперсия вычисляется по формуле:

$$\Delta_x^* = \sum_{i=1}^{n} \left(x_i - m_x^*\right)^2 p_i = \frac{1}{n} \sum_{i=1}^{n} \left(x_i - m_x^*\right)^2, \tag{18.19}$$

где m_x^* находится по (18.14). Здесь мы обозначили дисперсию не латинской буквой D, а греческой Δ, т. к., как мы увидим дальше, это — не совсем то, что нам нужно. Кроме того, мы здесь вынуждены обозначать большой буквой и реализацию случайной величины (18.3), и ее саму (18.4). В математической статистике часто так поступают.

Проверим оценку (18.19) на состоятельность и несмещенность. Переходим от реализаций к случайным величинам:

$$\Delta_x^* = \frac{1}{n} \sum_{i=1}^{n} \left(X_i - M_x^*\right)^2. \tag{18.20}$$

Преобразуем вначале выражение (18.20). Центрируем оба слагаемых в скобках:

$$\Delta_x^* = \frac{1}{n} \sum_{i=1}^{n} \left(\left(X_i - m_x\right) - \left(M_x^* - m_x\right)\right)^2. \tag{18.21}$$

Раскроем квадрат разности:

$$\Delta_x^* = \frac{1}{n} \left(\sum_{i=1}^{n} \left(X_i - m_x\right)^2 - 2\left(M_x^* - m_x\right) \sum_{i=1}^{n} \left(X_i - m_x\right) + n\left(M_x^* - m_x\right)^2 \right). \tag{18.22}$$

Сумма в среднем слагаемом преобразуется:

$$\sum_{i=1}^{n} \left(X_i - m_x\right) = \sum_{i=1}^{n} X_i - n m_x = n M_x^* - n m_x = n\left(M_x^* - m_x\right), \tag{18.23}$$

и после подстановки (18.23) в (18.22) имеем:

$$\Delta_x^* = \frac{1}{n} \left(\sum_{i=1}^{n} \left(X_i - m_x\right)^2 - n\left(M_x^* - m_x\right)^2 \right) = \frac{1}{n} \sum_{i=1}^{n} \left(X_i - m_x\right)^2 - \left(M_x^* - m_x\right)^2. \tag{18.24}$$

Найдем математическое ожидание случайной величины Δ_x^*:

$$\begin{aligned} M\left(\Delta_x^*\right) &= \frac{1}{n} \sum_{i=1}^{n} M\left(\left(X_i - m_x\right)^2\right) - M\left(\left(M_x^* - m_x\right)^2\right) = \\ &= \frac{n D_x}{n} - D\left(M_x^*\right) = D_x - \frac{D_x}{n} = \frac{n-1}{n} D_x. \end{aligned} \tag{18.25}$$

Здесь во второй строке использовано соотношение (18.17). Мы видим, что (18.13) не выполняется, т. е. оценка (18.20) является смещенной. Чтобы избавиться от смещения, нужно компенсировать множитель $(n-1)/n$, и вместо (18.19) взять другую оценку D_x^*:

$$D_x^* = \frac{n}{n-1}\Delta_x^* = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - m_x^*\right)^2. \qquad (18.26)$$

Эта оценка будет уже несмещенной:

$$\lim_{n\to\infty} D_x^* = \frac{n}{n-1}\lim_{n\to\infty}\Delta_x^* = \frac{n}{n-1}\lim_{n\to\infty}\left(\frac{n-1}{n}D_x\right) = D_x. \qquad (18.27)$$

Определение 18.8. Число в знаменателе выборочной дисперсии:

$$f = n-1$$

называется *числом степеней свободы* выборки из n элементов. □

Смысл числа степеней свободы следующий. У нас есть n экспериментальных данных. Если мы хотим оставить неизменной выборочную дисперсию D_x^*, то мы свободно можем варьировать не всеми n данными, а только $(n-1)$ из них, а n-е находится из формулы для выборочного математического ожидания m_x^*. Иными словами, на n переменных x_i наложено одно ограничение: вычисление m_x^*. Это — общий принцип: во всех задачах статистики в знаменателе выборочной дисперсии стоит число степеней свободы, равное объему выборки минус число ограничений.

Для проверки состоятельности оценки D_x^* нужно вычислить ее дисперсию $D(D_x^*)$ и убедиться, что она стремится к нулю при $n\to\infty$. Вывод $D(D_x^*)$ очень громоздкий, приведем лишь окончательный результат:

$$D\left(D_x^*\right) = \frac{D_x^2}{n-1}, \qquad (18.28)$$

что свидетельствует о ее состоятельности.

В MATLAB есть функция `var` для вычисления выборочной дисперсии и функция `std` для выборочного среднеквадратичного отклонения — квадратного корня из дисперсии:

$$\sigma_x^* = \sqrt{D_x^*}. \qquad (18.29)$$

Вычислим эти параметры для нашей выборки.

```
f=n-1; % число степеней свободы
Dx=var(x); % дисперсия
Sx=std(x); % среднеквадратичное отклонение
fprintf('Число степеней свободы выборки f=%d\n',f)
fprintf('Дисперсия Dx=%14.7f\n',Dx)
fprintf('Среднеквадратичное отклонение Sx=%14.7f\n',Sx)
Число степеней свободы выборки f=199
Дисперсия Dx=     1.0493907
Среднеквадратичное отклонение Sx=     1.0243977
```

18.5. Другие выборочные параметры

По такому же принципу можно построить формулы для вычисления выборочных асимметрии и эксцесса. Если мы хотим получить несмещенные и состоятельные оценки, то при их выводе нужно исходить из того, что выборочные начальные моменты любого порядка k:

$$M_x^{k*} = \frac{1}{n}\sum_{i=1}^{n} x_i^k \tag{18.30}$$

являются состоятельными и несмещенными оценками соответствующих генеральных моментов. Отсюда можно вывести оценки центральных моментов 3-го и 4-го порядков, а затем — асимметрии и эксцесса. Но полученные таким путем формулы довольно громоздкие и редко используются на практике. Обычно применяют более простые формулы, которые являются немного смещенными, но состоятельными. Так, в MATLAB для вычисления асимметрии есть функции `skewness`, в которой реализована формула:

$$a_x^* = \frac{\sqrt{n}}{\sqrt{(n-1)^3}\left(\sigma_x^*\right)^3}\sum_{i=1}^{n}\left(x_i - m_x^*\right)^3. \tag{18.31}$$

Эксцесс в системе MATLAB можно вычислить с помощью функции `kurtosis`, но потом из найденного значения нужно еще вычесть число 3, чтобы получить тот эксцесс, который обычно определяется в курсе теории вероятностей и который для нормального распределения равен нулю. В итоге получаем:

$$e_x^* = \frac{n}{(n-1)^2\left(\sigma_x^*\right)^4}\sum_{i=1}^{n}\left(x_i - m_x^*\right)^4 - 3, \tag{18.32}$$

и все, кроме –3, вычисляет функция `kurtosis`. Математические ожидания соответствующих случайных величин лишь приближенно равны генеральным асимметрии и эксцессу:

$$M(A_x^*) \approx a_x; \tag{18.33}$$

$$M(E_x^*) \approx e_x; \tag{18.34}$$

а дисперсии вычисляются по формулам:

$$D\left(A_x^*\right) = \frac{6(n-1)}{(n+1)(n+3)}; \tag{18.35}$$

$$D\left(E_x^*\right) = \frac{24n(n-2)(n-3)}{(n+1)^2(n+3)(n+5)}. \tag{18.36}$$

Вывод этих формул еще более громоздкий, чем (18.25), (18.28), и мы его опускаем. Но видно, эти оценки — состоятельные и почти несмещенные.

Вычислим асимметрию, эксцесс, медиану и размах выборочного распределения. Медиана равна среднему элементу упорядоченной выборки (если в выборке нечетное число элементов) или полусумме средних элементов (если их число четно). Для вычисления медианы существует функция `median`. Размах — это разность между максимальным и минимальным элементами выборки, он вычисляется с помощью функции `range`.

```
Ax=skewness(x); % асимметрия
Ex=kurtosis(x)-3; % эксцесс
Medx=median(x); % медиана
Rx=range(x); % размах выборки
fprintf('Асимметрия Ax=%14.7f\n',Ax)
fprintf('Эксцесс Ex=%14.7f\n',Ex)
fprintf('Медиана Medx=%14.7f\n',Medx)
fprintf('Размах Rx=%14.7f\n',Rx)
Асимметрия Ax=     0.6041663
Эксцесс Ex=     0.0857646
Медиана Medx=     1.8904831
Размах Rx=     5.4102408
```

18.6. Методика текущих измерений

Математическое ожидание (или среднее) и дисперсия характеризуют две стороны проводимого экспериментального исследования. Среднее характеризует свойство объекта исследований, а дисперсия — точность применяемой методики. Из теоремы Гливенко — Кантелли следует, что чем больше число испытаний n, тем выше точность измерений.

На практике не всегда есть возможность провести большое число испытаний. Однако часто в руках исследователя есть результаты, полученные в других

лабораториях (статьи, книги, отчеты и т. п.). Эти полученные ранее результаты могут интересовать нас с двух точек зрения.

1. Над одним и тем же объектом работают различные лаборатории. Различные методики приводят к различным дисперсиям, а один объект исследования — к одному среднему.
2. Для исследования различных объектов применяется одна и та же методика. В этом случае математические ожидания разные, а дисперсия — одинаковая.

Оказывается, изменение одного из параметров не мешает использовать *все* измерения для оценки другого, если этот другой неизменен. Нужно только уметь грамотно использовать полученные ранее результаты для увеличения точности своих измерений.

Определение 18.9. Схема учета полученных ранее результатов для повышения точности своих исследований называется *методикой текущих измерений*. □

Рассмотрим применение этой методики к перечисленным выше двум задачам. При решении 1-й задачи — вычислении среднего — различиями в дисперсиях можно просто пренебречь, т. е. можно собрать все измерения (и свои, и полученные ранее) в одну выборку и вычислить для нее выборочные параметры. Повышение точности при этом происходит за счет увеличения общего объема выборки.

ПРИМЕР 18.7. В статье опубликованы результаты исследования некоторого объекта X. В ней приведены: объем выборки $n_1 = 20$, выборочные среднее $m_1^* = 8{,}6$ и дисперсия $D_1^* = 6{,}8$. Наша лаборатория тоже занимается исследованием этого объекта. Но исследования стоят дорого, да и оборудование у нас похуже, поэтому мы смогли провести только $n_2 = 12$ опытов и получили $m_2^* = 8{,}4$ и $D_2^* = 8{,}4$. Среднее — примерно такое же, а дисперсия у нас выше из-за износа оборудования. Но методика текущих измерений позволяет объединить все опыты в одну выборку объема $n = n_1 + n_2 = 32$. Среднее всей выборки вычисляется по очевидной формуле:

$$m_x^* = \frac{n_1 m_1^* + n_2 m_2^*}{n_1 + n_2} = 8{,}525, \qquad (18.37)$$

а общее число степеней свободы $f = n - 1 = 31$. Теперь мы можем считать, что провели $n = 32$ опыта, среднее $m_x^* = 8{,}525$, но дисперсия наша по-прежнему $D_2^* = 8{,}4$. Тем не менее, увеличение общего числа степеней свободы позволяет давать более точные оценки для генерального среднего *(см. главу 20)*. □

При решении 2-й задачи, т. е. при оценке дисперсии, мы уже не можем объединить все данные в одну выборку, т. к. они соответствуют разным объектам. Тем не менее, построить средневзвешенную дисперсию можно. Нужно только учесть, что при вычислении выборочной дисперсии (18.26) мы делили сумму квадратов отклонений не на число опытов n, а на число степеней свободы $f=n-1$. Поэтому в отличие от (18.37) средневзвешенная дисперсия вычисляется так:

$$D_x^* = \frac{f_1 D_1^* + f_2 D_2^*}{f_1 + f_2}, \qquad (18.38)$$

а общее число степеней свободы $f=f_1+f_2=n_1+n_2-2$.

ПРИМЕР 18.8. В нашей лаборатории установили дорогостоящее оборудование, и мы смогли провести только $n_1=20$ опытов над объектом X. Наши результаты: $m_1^*=8{,}6$; $D_1^*=6{,}8$. В библиотеке была найдена статья, где эта же методика применяется для исследования другого объекта Y. Авторы этой статьи провели $n_2=18$ опытов и получили $m_2^*=15{,}4$; $D_2^*=6{,}4$. Несмотря на то, что в статье описан другой объект исследования, ее результаты по методике текущих измерений можно применить для оценки выборочной дисперсии. Вычисления по формуле (18.38) дают значение $D_x^*\approx 6{,}611$ с общим числом степеней свободы $f=f_1+f_2=n_1+n_2-2=36$. Теперь мы можем считать, что провели 38 опытов, получили $m_1^*=8{,}6$; $D_x^*\approx 6{,}611$; и у выборки 36 степеней свободы. Увеличение n и f позволяет дать более точные оценки для генеральной дисперсии *(см. главу 20)*. □

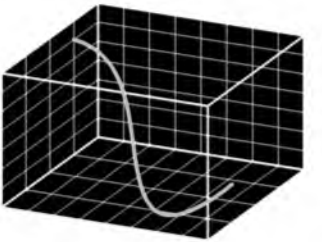
Разумеется, при использовании методики текущих измерений нужно соблюдать общепринятые этические нормы и авторские права: в своих публикациях обязательно давать ссылки на те источники, откуда взяты результаты.

18.7. Вопросы для самопроверки

1. Что называется генеральной совокупностью? Выборкой? Объем чего больше?
2. Приведите примеры разрушающего и неразрушающего исследования выборки.
3. В чем заключается выборочный метод?
4. Как вычисляется выборочная функция распределения? Почему именно так?
5. Найдите в [46] или другой литературе доказательство теоремы Гливенко — Кантелли. Используется ли там неравенство Чебышева?

6. Как правильно организовать экзит-пол на выборах, чтобы его результаты совпали с истинными результатами выборов?
7. Что означают требования состоятельности, несмещенности и эффективности?
8. Докажите необходимость выполнения условий (18.6) в теореме 18.2.
9. Представьте себе, что выборочное математическое ожидание вычисляется не как среднее арифметическое (18.14), а как среднее геометрическое или среднее гармоническое. Будут ли эти оценки состоятельными? Несмещенными? Более или менее эффективными, чем (18.14)?
10. Найдите в литературе доказательство состоятельности и несмещенности оценки (18.30).
11. Выведите из (18.30) состоятельные и несмещенные оценки асимметрии и эксцесса. Насколько полученные формулы сложнее (18.31) и (18.32)?
12. Найдите в литературе вывод формул (18.28), (18.35) и (18.36).
13. Сравните медиану с математическим ожиданием. Сильно ли они отличаются?
14. Как выглядят формулы (18.37—18.38) методики текущих измерений, если наши исследования дополняются результатами не из одной статьи, а из двух или нескольких?

ГЛАВА 19



Доверительные оценки параметров распределения

В предыдущей *главе 18* мы научились определять выборочные параметры распределения, которые являются оценками соответствующих генеральных параметров. Такие оценки будут приближенными в силу случайности выборки. В этой главе мы рассмотрим некоторые понятия, с помощью которых сможем судить, насколько точно выборочные параметры совпадают с генеральными.

19.1. Квантили

Определение 19.1. *Квантилем* x_p распределения случайной величины X с функцией распределения $F(x)$ называется решение уравнения:

$$F(x_p)=p. \square \tag{19.1}$$

Квантиль — это аргумент, соответствующий данному значению функции распределения. Процесс вычисления квантиля показан на рис. 19.1.

Чтобы получить таблицу квантилей какого-либо распределения, нужно взять таблицу значений функции этого распределения и поменять местами столбцы аргументов и функций. Поэтому иногда квантили называют обратной функцией распределения. Другое название квантилей — критические значения соответствующего распределения. Это название связано с понятием доверительного интервала, о чем мы поговорим дальше в этой главе.

Индекс внизу у квантиля соответствует уровню вероятности. Так, $x_{0.05}$ — это 5%-ный квантиль, $x_{0.99}$ — 99%-ный, $x_{0.5}$ — медиана распределения (теоретическая, не выборочная!)

В MATLAB квантили различных распределений вычисляются с помощью функций `*inv`, где `*` — название вида распределения. Вообще, как правило,

функции для различных законов распределения в MATLAB ходят "колонна-ми по 7":

- `*cdf` — вычисление функции распределения (сокращение от cumulative distribution function — функция распределения);
- `*pdf` — вычисление плотности распределения (probability density function — плотность распределения);
- `*inv` — вычисление квантиля распределения (inverse cumulative distribution function — обратная функция распределения);
- `*rnd` — генератор случайных чисел (random number generator — генератор случайных чисел);
- `*stat` — вычисление статистических характеристик (математического ожидания и дисперсии) соответствующего теоретического распределения;
- `*fit` — вычисление параметров теоретического распределения по выборке с помощью ПМП;
- `*like` — вычисление функции правдоподобия.

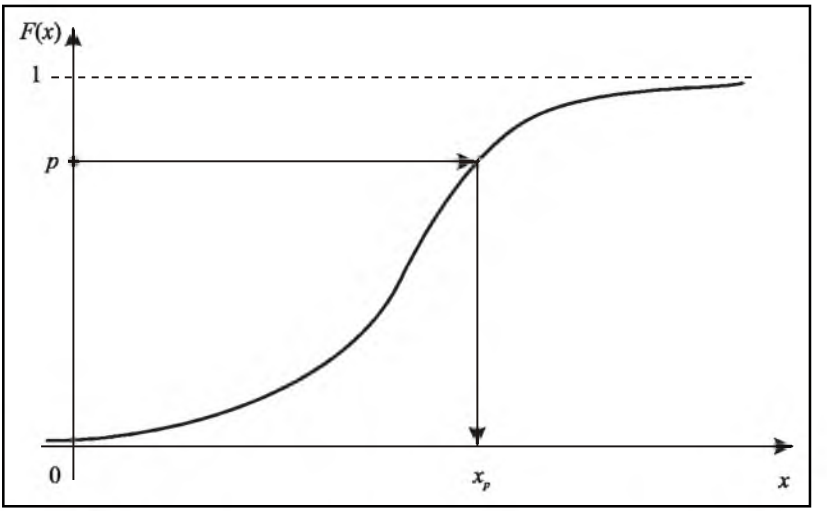


Рис. 19.1. Вычисление квантиля x_p распределения $F(x)$ по заданной вероятности p

Но не для всякого распределения реализованы все 7 функций. Наберите в командном окне MATLAB `help stats` и посмотрите, для каких распределений какие функции имеются. Полный список функций инструментария [57] приведен в *главе 27*.

19.2. Доверительный интервал и доверительная вероятность

Ранее мы научились по выборке $x_1, x_2, ..., x_n$ находить различные выборочные параметры, которые в общем виде мы обозначили b^* (18.3). В соответствии с выборочным методом считаем, что b^* является реализацией случайной величины B^*. Но, даже если оценка B^* состоятельная и несмещенная, ее реализация b^* не равна в точности генеральному параметру b, т. к. сказывается влияние случайностей. Возникает вопрос: насколько точно выборочный параметр b^* оценивает генеральный b? Мы можем ответить на этот вопрос только в вероятностной постановке. Например: какова вероятность, что b^* будет отличаться от b не более чем на заданную величину? Или наоборот: в какой интервал вокруг b^* попадает b с заданной вероятностью? Ответы на эти вопросы приводят нас к понятиям доверительного интервала и доверительной вероятности.

Пусть мы знаем закон распределения генеральной совокупности X, например, ее функцию распределения $F(x)$ или плотность распределения $f(x)$. Тогда по (18.4) мы можем определить закон распределения случайной величины B^*, например, $F(b^*)$ или $f(b^*)$. Зная его, можно найти вероятность того, что случайная величина B^* отличается от своего математического ожидания b на величину, не превышающую ε:

$$\mathbf{P}\left(\left|B^* - b\right| \le \varepsilon\right) = \int_{b-\varepsilon}^{b+\varepsilon} f\left(b^*\right) db^* = F\left(b+\varepsilon\right) - F\left(b-\varepsilon\right). \qquad (19.2)$$

Соответствующая площадь под графиком плотности распределения выборочного параметра $f(b^*)$ показана на рис. 19.2.

В левой части формулы (19.2) — детерминированный интервал $[b-\varepsilon, b+\varepsilon]$ для случайной величины B^*:

$$b-\varepsilon \le B^* \le b+\varepsilon. \qquad (19.3)$$

Перейдем от него к интервалу со случайными границами для детерминированной величины b — нашего генерального параметра. Для этого решим неравенства (19.2) относительно b:

$$B^*-\varepsilon \le b \le B^*+\varepsilon. \qquad (19.4)$$

Поскольку (19.4) — это следствие (19.3), то вместо (19.2) можем записать:

$$\mathbf{P}\left(b \in \left[B^* - \varepsilon;\ B^* + \varepsilon\right]\right) = \int_{b-\varepsilon}^{b+\varepsilon} f\left(b^*\right) db^* = F\left(b+\varepsilon\right) - F\left(b-\varepsilon\right). \qquad (19.5)$$

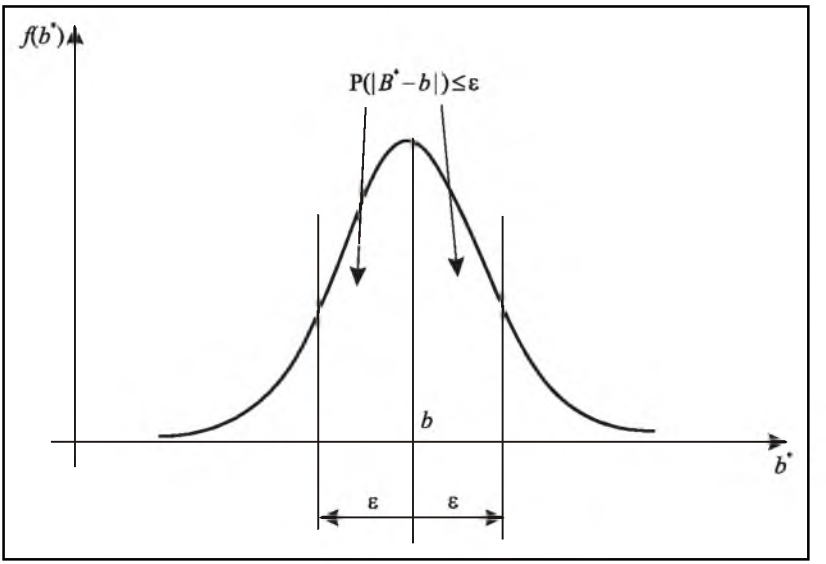


Рис. 19.2. Вероятность попадания выборочного параметра b^* в ε-окрестность генерального параметра b

Эта простая операция: переход от детерминированного интервала для случайной величины к случайному интервалу для детерминированной величины дает нам понятие доверительного интервала.

Определение 19.2. *Доверительным интервалом* называется интервал со случайными границами для детерминированной величины — генерального параметра (19.4). □

Определение 19.3. *Доверительной вероятностью* называется вероятность попадания генерального параметра в его доверительный интервал. □

На практике у нас есть только конкретное значение выборочного параметра b^* — реализация случайной величины B^*. Поэтому вместо случайного интервала (19.4) мы получим только его реализацию $[b^* - \varepsilon, b^* + \varepsilon]$. Но мы можем быть уверены, что неизвестный генеральный параметр b попадет в этот интервал с вероятностью (19.5).

Доверительный интервал и доверительная вероятность связаны между собой. Зная одну из этих величин, всегда можно вычислить другую. Вычисление доверительной вероятности по доверительному интервалу выполняется по формуле (19.5) и показано на рис. 19.2.

ПРИМЕР 19.1. Генеральная совокупность X имеет нормальное распределение с известной дисперсией $D_x = 4$. Проведено одно измерение: $x_1 = 2{,}7$. Какова вероятность того, что неизвестное математическое ожидание m_x отличает-

ся от этого значения не более чем на 0,5 в каждую сторону, т. е. лежит в пределах [2,2; 3,2]?

По общей схеме выборочного метода считаем, что x_1 — реализация случайной величины X_1, имеющей то же распределение, что и X, т. е. нормальное с неизвестным математическим ожиданием m_x и известной дисперсией $D_x = 4$. Центрируем (вычитаем математическое ожидание): $X_1 - m_x$ является нормальной с нулевым математическим ожиданием и той же дисперсией $D_x = 4$. Нам нужно найти вероятность того, что эта величина лежит в пределах [–0,5; 0,5]. Это и будет доверительная вероятность. Вычисляем с помощью MATLAB:

```
normcdf(0.5,0,2)-normcdf(-0.5,0,2)
ans =
   0.19741265136585
```

Ответ. Искомая доверительная вероятность равна ≈19,74 %. □

Конечно, доверительный интервал может быть и несимметричным относительно b. Так, в примере 19.1 можно найти доверительную вероятность для доверительного интервала [2; 3] или [0; 5].

Рассмотрим теперь обратную задачу. Пусть задана доверительная вероятность p, а нужно найти доверительный интервал, в который попадает генеральный параметр b с вероятностью p. Эта задача решается неоднозначно. Посмотрите на рис. 19.3. Мы ведь можем двигать правую и левую границы доверительного интервала так, что площадь криволинейной трапеции, ограниченной этими боковыми сторонами, все время остается равной p!

В качестве решения можно взять два любых квантиля распределения величины B^*, для которых уровень вероятности отличается на p. Выбрав любое число $r \in [0; 1-p]$, получим ответ в виде доверительного интервала $\left[b_r^*, b_{r+p}^* \right]$ (затененная область на рис. 19.3). При изменении r доверительный интервал сдвигается, и ширина его в общем случае также изменяется. Неизменной останется только площадь p под кривой графика плотности распределения внутри доверительного интервала. Как же выбрать "наилучшее" решение? Интуиция подсказывает, что нужно выбирать доверительный интервал посередине. Если распределение B^* симметричное, то и квантили $\left[b_r^*, b_{r+p}^* \right]$ нужно взять симметрично относительно b. Если же, как на рис. 19.3, график плотности распределения $f(b^*)$ несимметричен, то под симметричным решением будем понимать отбрасывание слева и справа кусков одинаковой площади. Общая отрезаемая площадь равна $1-p$.

Определение 19.4. Величина

$$q = 1 - p, \tag{19.6}$$

где p — доверительная вероятность, называется *уровнем значимости*. □

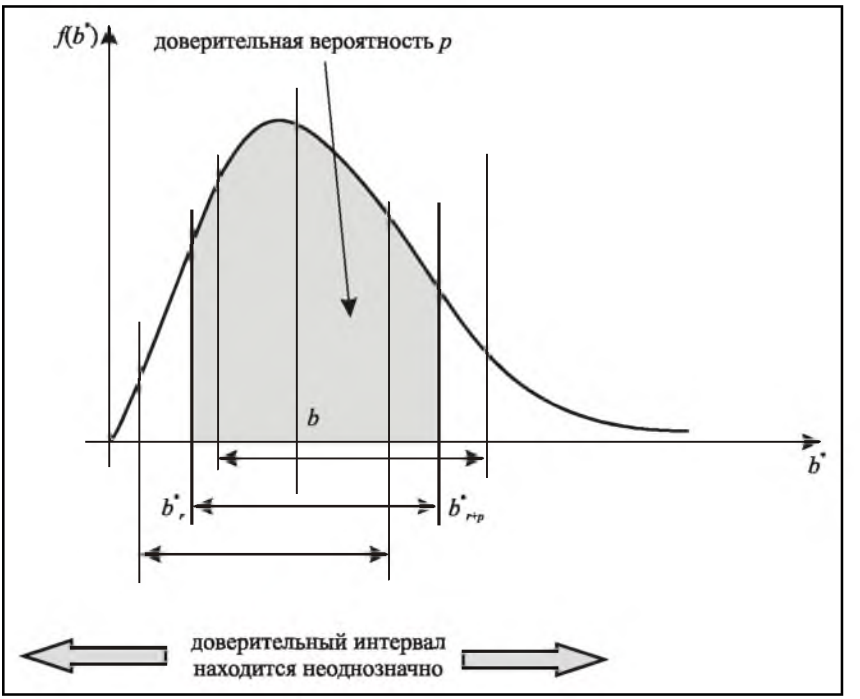


Рис. 19.3. Нахождение доверительного интервала по заданной доверительной вероятности p

Это название возникло следующим образом. Если задать достаточно большую доверительную вероятность p (близкую к 1), то q будет малым. По ПМП попадание вне доверительного интервала почти невозможно. Поэтому, если такое произошло, мы должны считать, что это не случайное событие, а *значимое*, т. е. что-то означающее. Что именно — см. далее в *разделе 19.4*. А мы возвращаемся к построению симметричного доверительного интервала. Для симметрии нужно отрезать слева и справа куски одинаковой площади $q/2$, и тогда решением будет доверительный интервал $\left[b^*_{\frac{q}{2}}, b^*_{1-\frac{q}{2}} \right]$.

ПРИМЕР 19.2. Условие то же, что и в примере 19.1: генеральная совокупность X имеет нормальное распределение с известной дисперсией $D_x = 4$. Проведено одно измерение: $x_1 = 2{,}7$. Для доверительных вероятностей $p = 0{,}9$, 0,99 и 0,999 найти доверительный интервал.

Доверительный интервал — симметричный относительно $x_1 = 2{,}7$. Его ширина определяется квантилями нормального распределения с дисперсией $D_x = 4$, соответствующим уровням вероятности $q/2$ и $1-q/2$, где $q = 1-p$.

Считаем:

```
p=[0.9;0.99;0.999];
q=1-p;
norminv([q/2,1-q/2],2.7,2)
ans =
  -0.58970725390295   5.98970725390295
  -2.45165860709780   7.85165860709781
  -3.88105346298379   9.28105346298381
```

Ответ. Первая строка — это доверительный интервал для $p = 0{,}9$, 2-я — для $p = 0{,}99$, 3-я — для $p = 0{,}999$. Видно, что с увеличением доверительной вероятности доверительный интервал расширяется. Это соответствует увеличению затененной площади на рис. 19.3. □

С помощью этой же функции можно проверить правильность решения примера 19.1:

```
norminv([(1-0.19741265136585)/2 (1+0.19741265136585)/2],2.7,2)
ans =
   2.19999999999999   3.20000000000001
```

ПРИМЕР 19.3. Найти доверительные вероятности, соответствующие правилам 1σ, 2σ и 3σ. Иными словами: случайная величина X имеет нормальное распределение с математическим ожиданием m_x и дисперсией D_x. Среднеквадратичное отклонение $\sigma_x = \sqrt{D_x}$. Проведено 1 измерение x_1. Найти вероятность того, что оно отличается от m_x не более, чем на σ_x, $2\sigma_x$, $3\sigma_x$.

Перейдем к стандартной нормальной величине с помощью линейного преобразования:

$$U = \frac{X - m_x}{\sigma_x}. \tag{19.7}$$

Величина U имеет нормальное распределение с $m_u = 0$ и $\sigma_u = 1$. Теперь задача заключается в нахождении вероятности ее попадания в интервалы [–1; 1], [–2; 2] и [–3; 3]. Находим эти вероятности:

```
u=[1:3]';
p=normcdf(u,0,1)-normcdf(-u,0,1)
p =
   0.68268949213709
   0.95449973610364
   0.99730020393674
```

Ответ. Правилам 1σ, 2σ и 3σ соответствуют доверительные вероятности 68,27 %, 95,45 % и 99,73 % соответственно. □

19.3. Абсолютная и практическая достоверность

Из каких соображений назначается доверительная вероятность p в практических расчетах? Например, в массовом производстве нельзя назначать вероятность безотказной работы $p = 0{,}9$, т. к. мы не можем себе позволить 10% брака. Лучше всего было бы назначить доверительную вероятность $p = 1$, но ей соответствует бесконечный доверительный интервал. На практике же мы всегда задаем конечный доверительный интервал, поэтому вынуждены задавать $p < 1$. Получаем два противоречивых требования. С одной стороны, нужно увеличить доверительную вероятность p, а с другой — задать доверительный интервал поуже. Это противоречие связано с другим: между абсолютной и практической достоверностью.

Определение 19.5. Событие A называется *абсолютно достоверным*, если $P(A) = 1$. □

Определение 19.6. Событие A называется *практически достоверным*, если оно практически всегда происходит на практике. □

Для практически достоверного события вероятность его наступления $p < 1$, но она настолько близка к 1, что для всех проведенных ранее и проводимых в будущем испытаний, общее число которых обозначим n, должно выполняться $p^n \approx 1$.

ПРИМЕР 19.4. Теоретически $2 \times 2 = 4$. Это — абсолютная истина. Если же мы будем считать 2×2 на практике, то почти всегда также получим 4, за исключением возможных ошибок, сбоев и поломок компьютеров, вероятность которых очень мала. Поэтому $2 \times 2 = 4$ — и практически достоверное событие. □

Появление на практике практически достоверных событий — это еще одна формулировка ПМП (принцип практической достоверности), и именно этим принципом мы руководствуемся при назначении доверительной вероятности на практике. Доверительную вероятность p нужно назначать так, чтобы для любого реального числа испытаний n число p^n мало отличалось от 1. Применительно к нашим задачам это позволяет ограничиться доверительной вероятностью $p < 1$ и конечным доверительным интервалом.

Из принципа практической достоверности следует принцип практической невозможности (еще одна формулировка ПМП): события с очень малыми вероятностями в практических приложениях можно считать невозможными. В нашем случае: выход случайной величины за границы доверительного интервала практически невозможен.

19.4. Проверка статистических гипотез

Определение 19.7. *Статистической гипотезой* называется любое предположение о законе распределения генеральной совокупности или его параметрах. □

Так, мы можем выдвигать предположение о величине m_x, D_x или о характере функций $F(x)$, $f(x)$. Все это — статистические гипотезы. Другие гипотезы будут рассмотрены далее в примерах этой главы. Выясним, как принцип практической достоверности (или невозможности) может быть применен для проверки статистических гипотез.

Пусть из генеральной совокупности X получена выборка $x_1, x_2, ..., x_n$. По ней найден некоторый выборочный параметр b^* (18.3), а затем по заданной доверительной вероятности p определен доверительный интервал для генерального параметра b:

$$b \in \left[b^*_{\frac{q}{2}}, b^*_{1-\frac{q}{2}} \right]. \tag{19.8}$$

Мы продолжаем исследования дальше: увеличиваем объем выборки n, пересчитываем b^*, и вдруг оказывается, что новое b^* не попадает в доверительный интервал (19.8), как показано на рис. 19.4 сплошными стрелками. Что же происходит? Здесь могут быть 2 вывода.

1. Сказывается влияние случайностей. Ведь доверительный интервал (19.8) — не абсолютно достоверный, а лишь практически. Однако при таком допущении мы должны отказаться от принципа практической достоверности.

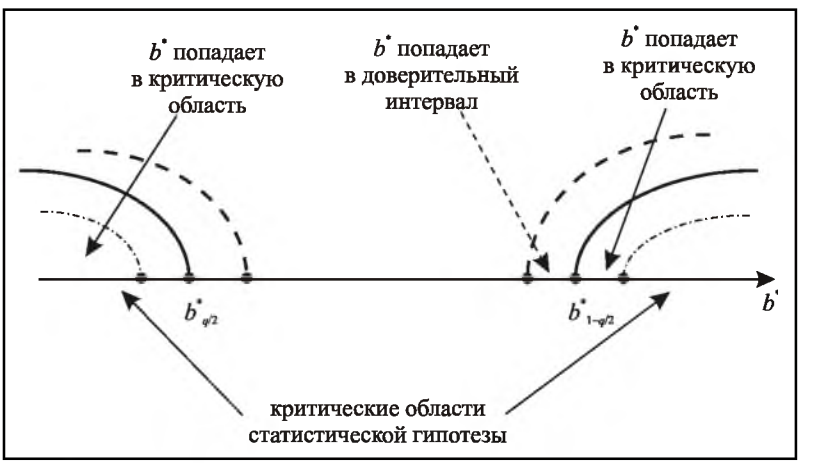


Рис. 19.4. Доверительный интервал и критические области статистической гипотезы

2. Принятая нами статистическая гипотеза неверна, и мы считали b^* по неправильной формуле (18.3), вид которой должен быть другим. Тогда и доверительный интервал (19.8) будет другим, и уточненное b^* уже попадет в него.

Очевидно, исходя из принципа практической достоверности, мы должны принять 2-е предположение. Таким образом, принцип практической достоверности (или невозможности) является одним из критериев проверки статистических гипотез. Если в процессе исследований проявляется такое значение выборочного параметра, которое в силу принципа практической невозможности проявиться не может, следует считать, что появление такого события — не случайное явление. Говорят, что такое событие *значимо*, а использование принципа практической невозможности для доказательства неслучайности появления события с малой вероятностью называется *принципом значимости*.

Заметим, что принципы практической достоверности, практической невозможности и значимости — это просто различные формулировки ПМП. Вот их формулировки.

Принцип 19.1 (практической достоверности). Практически достоверные события происходят почти всегда. В частности, выборочный параметр практически всегда попадает в свой доверительный интервал (19.8). □

Принцип 19.2 (практической невозможности). Практически невозможные события не происходят почти никогда. В частности, выборочный параметр практически никогда не выйдет из своего доверительного интервала (19.8). □

Принцип 19.3 (значимости). Если произошло практически невозможное, по нашему предположению, событие, то наше предположение неверно. В частности, если выборочный параметр выходит из своего доверительного интервала (19.8), то это значит, что доверительный интервал найден неправильно. □

Определение 19.8. Области, не попадающие в доверительный интервал, называются *критическими областями* статистической гипотезы, а границы доверительного интервала — *критическими числами (значениями) гипотезы*. □

Попадание уточненного b^* в критические области (сплошные стрелки на рис. 19.4) свидетельствует о том, что нужно отвергнуть статистическую гипотезу. Наоборот, попадание уточненного b^* в доверительный интервал (штриховая стрелка на том же рисунке) говорит о том, что у нас нет оснований отвергать статистическую гипотезу, хотя она, может быть, и не верна. Принимая то или иное решение (принять или отвергнуть гипотезу), мы можем допустить такие ошибки:

1. Попали в критическую область, но близко к границе. Нужно отвергнуть гипотезу, хотя она, может быть, верна. Если мы уменьшим уровень значимости q, как показано на рис. 19.4 тонкими пунктирными линиями, то b^* уже попадет в доверительный интервал.
2. Попали в доверительный интервал, но опять-таки близко к границе. Нужно принять гипотезу, а она, может быть, неверна. Если увеличить q, то доверительный интервал сузится (штриховые линии на рис. 19.4), и b^* попадет в критическую область.

Вероятность 1-й ошибки мала. Она не превышает уровня значимости q. Чтобы ее еще уменьшить, нужно уменьшить q, тем самым расширяя доверительный интервал. И если уж мы и при расширенном доверительном интервале попадем в критическую область гипотезы, то можно ее смело отвергать: она почти наверняка неверна. Но при таком подходе возрастает вероятность 2-й ошибки: если мы попали в доверительный интервал, то это не значит, что гипотеза верна. Чтобы застраховаться от 2-й ошибки, нужно, наоборот, увеличивать q и сужать доверительный интервал. И тогда, если мы попадаем даже в этот узкий интервал, то гипотеза почти наверняка верна. Зато теперь при попадании в критическую область мы не можем быть уверены, что отвергаемая гипотеза ошибочна. Чтобы уменьшить 1-ю ошибку, надо уменьшить q, и тогда см. начало этого абзаца.

Как же решить эту проблему с "перетягиванием одеяла"? Все зависит от того, какую цель мы перед собой ставим. Если мы хотим максимально застраховаться от ошибки отвергнуть правильную гипотезу, нужно уменьшить q (говорят: сделать уровень значимости более мягким). Тогда, если мы попадаем в критическую область, то гипотеза почти наверняка неверна. И наоборот, если нам нужно в максимальной степени застраховаться от принятия неправильной гипотезы, нужно назначать более жесткий (высокий) уровень значимости q. Если мы теперь попадем в узкий доверительный интервал, то статистическая гипотеза почти наверняка верна.

19.5. Односторонние и двухсторонние критерии

Чтобы понять, о чем пойдет речь, рассмотрим 2 примера статистических гипотез.

ПРИМЕР 19.5. Есть 2 генеральные совокупности: X и Y. Из них взяты выборки объемами n_x и n_y соответственно. Статистическая гипотеза: $m_x = m_y$. □

ПРИМЕР 19.6. Есть 2 генеральные совокупности: X и Y. Из них взяты выборки объемами n_x и n_y соответственно. Статистическая гипотеза: $F(x) = F(y)$. □

В примере 19.5 можно ввести в рассмотрение случайную величину $M_x^* - M_y^*$, и тогда при справедливости гипотезы доверительный интервал для этой величины будет охватывать 0. Если реальное значение $m_x^* - m_y^*$ попадает в этот интервал, то мы принимаем гипотезу о нуле (она так и называется: 0-гипотеза), а если нет — отвергаем. При этом, отвергая 0-гипотезу, мы автоматически принимаем одну из двух альтернативных гипотез. Если мы попадаем в левую критическую область, то должны считать, что $m_x < m_y$, а если в правую — то $m_x > m_y$.

Иная ситуация в примере 19.6. Здесь нам нужно сравнивать не числа, а функции $F^*(x)$ и $F^*(y)$ по какому-либо критерию: максимальной по модулю разности, интеграла от квадрата разности или др. Полученная величина b^* может быть большой или маленькой. Если она малая, то можно принять гипотезу, а если большая, то $F(x) \neq F(y)$. В отличие от предыдущего примера, здесь всего одна альтернативная гипотеза. Поэтому критическая область, которая ей соответствует, также будет одна, как показано на рис. 19.5. Значит, весь уровень значимости q не делится пополам между двумя альтернативными гипотезами, а припадает на одну. В данном примере область альтернативной гипотезы — справа, она соответствует большим значениям b^*. Если $b^* \le b^*_{1-q}$, то принимаем нашу гипотезу $F(x) = F(y)$, а если $b^* > b^*_{1-q}$, то $F(x) \neq F(y)$.

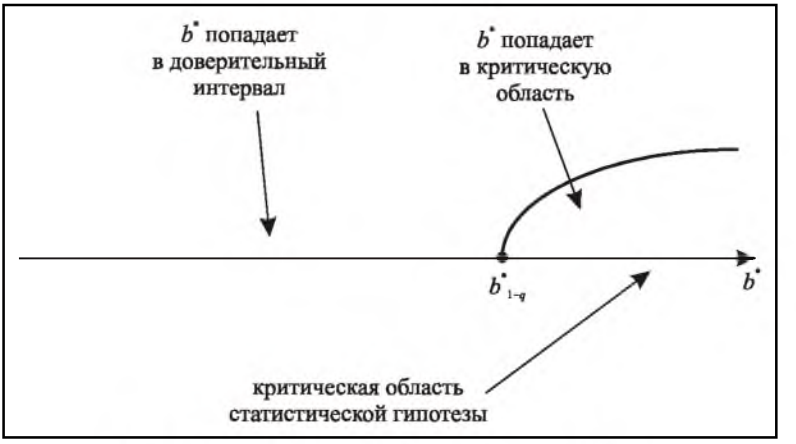


Рис. 19.5. Односторонний (правосторонний) критерий

Определение 19.9. Критерий проверки статистической гипотезы называется *двухсторонним*, если 0-гипотезе противостоят 2 альтернативные, и *односторонним* — если одна. □

Односторонние критерии возникают не только при сравнении функций, но иногда и при сравнении числовых параметров. Они появляются везде, где одна из альтернативных гипотез теоретически невозможна.

ПРИМЕР 19.7. По старой методике исследована генеральная совокупность X: из нее получена выборка объемом n_x. Затем методика была усовершенствована, и по ней была исследована генеральная совокупность Y: получена выборка объемом n_y. Действительно ли мы усовершенствовали методику?

Если методика действительно улучшена, то точность результатов должна быть выше. Значит, дисперсия должна уменьшиться: $D_y < D_x$. Здесь мы заранее предполагаем, что гипотеза $D_y > D_x$ теоретически невозможна. Поэтому 0-гипотезе $D_y = D_x$ противостоит только одна альтернативная: $D_y < D_x$. Рассматриваем случайную величину: отношение выборочных дисперсий $\frac{D_y^*}{D_x^*}$ (при сравнении дисперсий обычно рассматривают не разность, а отношение — см. *главу 22*). Если эта величина находится вблизи единицы или слишком большая, то нужно принять 0-гипотезу и считать, что никакого улучшения методики не было. Если же эта величина мала (меньше соответствующего квантиля), то нужно отвергнуть 0-гипотезу $D_y = D_x$ и принять тем самым альтернативную: $D_y < D_x$. Эта ситуация показана на рис. 19.6. □

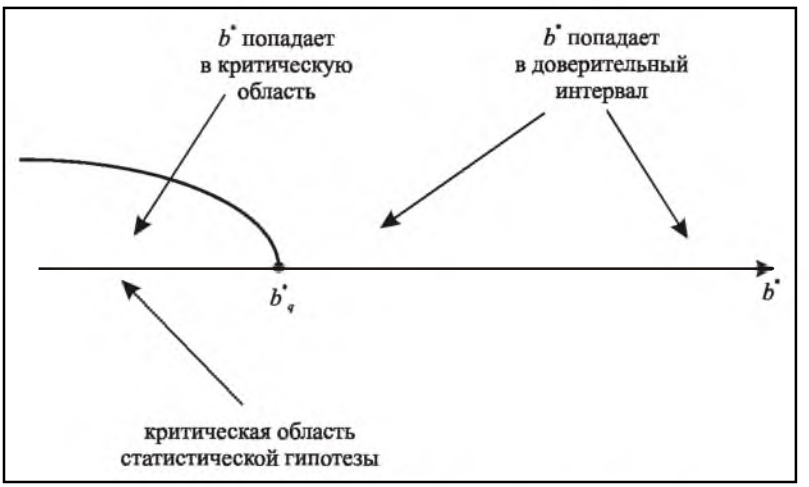


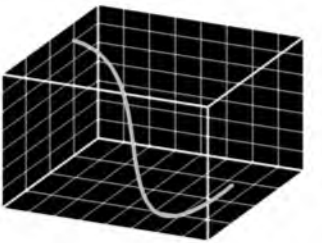
Рис. 19.6. Односторонний (левосторонний) критерий

Здесь критическим значением, разделяющим 0-гипотезу и альтернативную, является "левый" квантиль b_q^*. Весь уровень значимости q не делится пополам, а приходится на одну альтернативную гипотезу, область которой располагается слева от области 0-гипотезы.

19.6. Вопросы для самопроверки

1. Что называется квантилем?
2. Найдите с помощью MATLAB квантиль распределения Рэлея с параметром $\sigma = 2{,}5$, соответствующий вероятности $p = 0{,}9$.
3. В примере 19.1 найдите доверительную вероятность попадания m_x в интервал [0; 10].
4. Вы поставили чайник на плиту и включили газовую горелку достаточной мощности. Является ли закипание воды в чайнике абсолютно достоверным событием или только практически достоверным?
5. Есть 2 партии часов: одна — швейцарские, другая — китайские. Сравнивается точность их хода. Что здесь является статистической гипотезой? Является ли критерий ее проверки односторонним или двухсторонним?

ГЛАВА 20



Оценки генеральных параметров распределения

Применим сведения из предыдущей *главы 19* для нахождения доверительных интервалов различных числовых параметров генеральной совокупности. Эти интервалы иногда называют интервальными оценками.

20.1. Оценка генерального математического ожидания

Пусть генеральная совокупность X имеет нормальное распределение. Из нее получена выборка: $x_1, x_2, ..., x_n$. Далее по формуле (18.14) найдено выборочное математическое ожидание m_x^*. Требуется найти доверительный интервал для генерального математического ожидания m_x с заданной доверительной вероятностью p.

Если дисперсия генеральной совокупности D_x известна, то эта задача может быть решена так. По общей схеме выборочного метода m_x^* — реализация случайной величины M_x^*, которая вычисляется по формуле (18.15). В ней все X_i независимы и имеют нормальное распределение с параметрами m_x, D_x. Сумма независимых нормальных величин имеет нормальное распределение, умножение на константу также оставляет распределение нормальным. Поэтому случайная величина M_x^* имеет нормальное распределение. Его параметры — это (18.16—18.17). Переходим к стандартному нормальному распределению:

$$U = \frac{M_x^* - m_x}{\sqrt{\frac{D_x}{n}}} = \frac{M_x^* - m_x}{\sigma_x}\sqrt{n}. \qquad (20.1)$$

На уровне значимости $q=1-p$ эта величина лежит в пределах:

$$-u_{1-\frac{q}{2}} \le \frac{M_x^* - m_x}{\sigma_x}\sqrt{n} \le u_{1-\frac{q}{2}}. \qquad (20.2)$$

Здесь используется свойство симметрии квантилей стандартного нормального распределения: $u_{\frac{q}{2}} = -u_{1-\frac{q}{2}}$. Умножаем все 3 части неравенства на σ_x и делим на $\sqrt{n}$:

$$-\frac{\sigma_x u_{1-\frac{q}{2}}}{\sqrt{n}} \le M_x^* - m_x \le \frac{\sigma_x u_{1-\frac{q}{2}}}{\sqrt{n}}. \qquad (20.3)$$

Теперь переходим от детерминированного интервала для случайной величины M_x^* к случайному интервалу для детерминированной величины m_x, как мы это делали в общем виде (19.3—19.4):

$$M_x^* - \frac{\sigma_x u_{1-\frac{q}{2}}}{\sqrt{n}} \le m_x \le M_x^* + \frac{\sigma_x u_{1-\frac{q}{2}}}{\sqrt{n}}. \qquad (20.4)$$

Получили доверительный интервал для неизвестного математического ожидания m_x, если генеральная совокупность имеет нормальное распределение и ее дисперсия известна. На практике у нас есть только m_x^* — реализация M_x^*, поэтому вместо (20.4) имеем реализацию доверительного интервала:

$$m_x^* - \frac{\sigma_x u_{1-\frac{q}{2}}}{\sqrt{n}} \le m_x \le m_x^* + \frac{\sigma_x u_{1-\frac{q}{2}}}{\sqrt{n}}. \qquad (20.5)$$

Мы можем утверждать, что с доверительной вероятностью p неизвестное генеральное среднее m_x попадает в доверительный интервал (20.5).

Все это годится, если дисперсия генеральной совокупности D_x известна. Но обычно это не так. Как правило, у нас есть только сама выборка $x_1, x_2, \ldots, x_n$, а по ней мы D_x никак не сможем найти. По выборке мы можем найти только выборочную дисперсию D_x^* (18.26). Обозначим случайную величину, реализацией которой является D_x^*, также D_x^*:

$$D_x^* = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - M_x^*\right)^2, \qquad (20.6)$$

где M_x^* находится по (18.15).

Если квадратный корень из этой величины $\sigma_x^* = \sqrt{D_x^*}$ подставить в формулу (20.1) вместо σ_x, то полученная случайная величина:

$$T = \frac{M_x^* - m_x}{\sigma_x^*}\sqrt{n} \tag{20.7}$$

уже не будет иметь стандартное нормальное распределение! Впервые подробно исследовал распределение величины T Уильям Сили Госсет (William Sealey Gosset, 1876—1937, рис. 20.1). Он опубликовал эту работу под псевдонимом Студент (Student), поэтому распределение величины (20.7) вошло в историю как t-распределение Стьюдента.

Рис. 20.1. У. С. Госсет

Определение 20.1. Распределение случайной величины (20.7) называется *t-распределением Стьюдента.* □

Оно зависит как от параметра, от объема выборки n или от числа степеней свободы $f = n - 1$. При $n \to \infty$ t-распределение Стьюдента переходит в стандартное нормальное. В MATLAB есть функции `tcdf`, `tpdf`, `tinv`, `trnd`, `tstat` для работы с t-распределением. Нарисуем графики плотности t-распределения Стьюдента при нескольких значениях f. Для сравнения здесь же (рис. 20.2) нарисуем плотность стандартного нормального распределения.

```
t=linspace(-3,3)'; % аргументы
v=[2 5 10 20 30 60 120]; % степени свободы
for k=1:length(v)
  Y(:,k)=tpdf(t,v(k)); % плотность t-распределения
end
```

```
Y=[Y,normpdf(t,0,1)]; % добавили плотность норм. распр.
figure
plot(t,Y)
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfГрафик плотности \rm\itt\rm\bf-распределения Стьюдента')
xlabel('\itt') % метка оси OX
ylabel('\itf\rm(\itt\rm)') % метка оси OY
```

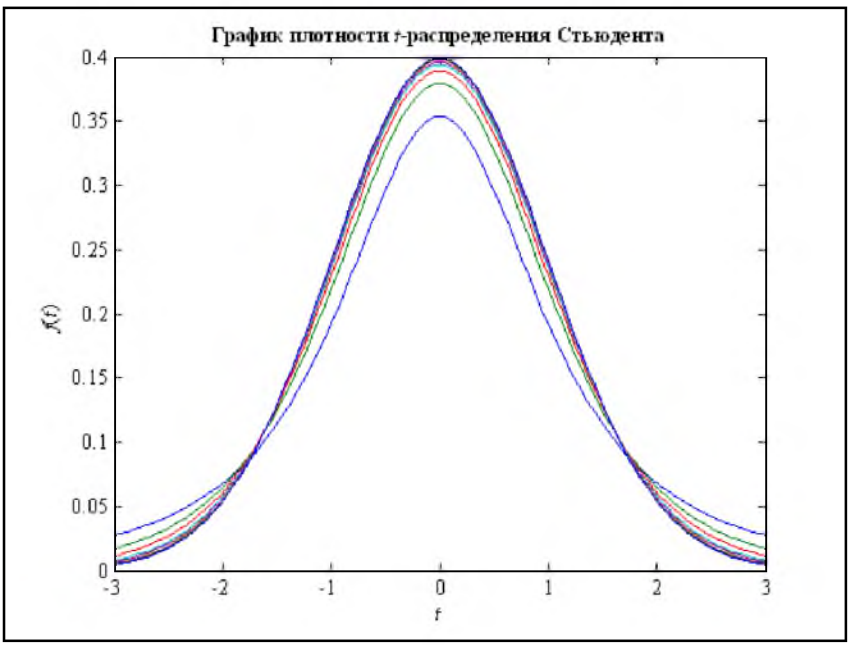


Рис. 20.2. График плотности *t*-распределения Стьюдента, выполненный с помощью MATLAB

Видно, что график $f(t)$ симметричен относительно вертикальной прямой $t=0$ и похож на график плотности нормального распределения, но проходит более полого. Выражения для функции и плотности *t*-распределения Стьюдента можно найти в [1, 27].

Теперь точно так же, как мы от (20.1) перешли к (20.5), перейдем от (20.7) к доверительному интервалу для m_x:

$$m_x^* - \frac{\sigma_x^* t_{1-\frac{q}{2}}(f)}{\sqrt{n}} \le m_x \le m_x^* + \frac{\sigma_x^* t_{1-\frac{q}{2}}(f)}{\sqrt{n}}. \qquad (20.8)$$

По сравнению с (20.5) изменения заключаются в том, что вместо генерального среднеквадратичного отклонения σ_x появилось выборочное σ_x^*, а вместо квантилей стандартного нормального распределения — квантили t-распределения Стьюдента, которые зависят от числа степеней свободы f.

Заметим, что в (20.8) m_x^* и σ_x^* не обязательно должны вычисляться по одной и той же выборке. Поэтому мы можем применить методику текущих измерений *(см. раздел 18.6)* для более точной оценки.

ПРИМЕР 20.1 (продолжение примера 18.7). Если мы используем только свои результаты: $n_2 = 12$, $m_2^* = 8.4$, $D_2^* = 8{,}4$, то по формуле (20.8) на уровне значимости $q = 0{,}1$ получим доверительный интервал [6,897; 9,9]. Если же использовать текущие измерения и взять $n = 32$, $m_x^* = 8{,}525$, $D_2^* = 8{,}4$, то на том же уровне значимости получим значительно более узкий доверительный интервал [7,656; 9,394]. Основное влияние на сужение интервала оказал знаменатель $\sqrt{n}$. □

Продолжим выполнение ИДЗ "Обработка массива данных". Зададим несколько значений доверительных вероятностей в виде вектор-столбца p. Вычислим по ним уровни значимости $q = 1 - p$. Для этих уровней значимости найдем доверительные интервалы для генерального математического ожидания m_x.

```
p=[0.9;0.95;0.99;0.999]; % задаем доверительные вероятности
q=1-p; % уровни значимости
t=tinv(1-q/2,f); % квантили Стьюдента с f степенями свободы
Mxd=[p,Mx-Sx*t/n^0.5,Mx+Sx*t/n^0.5]'; % формула (20.8)
disp(['Доверительные интервалы для генерального '...
  'математического ожидания'])
fprintf('p=%8.4f:   %9.6f<=mx<=%9.6f\n',Mxd)
```

```
Доверительные интервалы для генерального математического ожидания
p=  0.9000:    1.841074<=mx<= 2.080482
p=  0.9500:    1.817938<=mx<= 2.103618
p=  0.9900:    1.772390<=mx<= 2.149166
p=  0.9990:    1.718836<=mx<= 2.202720
```

Конечно, мы не знаем, является ли наше распределение нормальным. Но обычно формулу (20.8) применяют и для неизвестного распределения, оговаривая при этом, что полученные результаты — лишь приближенные.

20.2. Оценка генеральной дисперсии

Исходные данные те же: из генеральной совокупности X с нормальным распределением получена выборка $x_1, x_2, \ldots, x_n$. По ней найдена выборочная дис-

персия (18.26). Требуется найти доверительный интервал для генеральной дисперсии D_x с заданной доверительной вероятностью p.

Решаем эту задачу так же, как и для математического ожидания. По общей схеме выборочного метода считаем выборочную дисперсию (18.26) реализацией случайной величины

$$D_x^* = \frac{1}{f}\sum_{i=1}^{n}\left(X_i - M_x^*\right)^2, \tag{20.9}$$

где случайная величина M_x^* вычисляется по формуле (18.15). Закон распределения этой величины зависит от объема выборки n и генеральной дисперсии D_x каждой X_i. Есть еще один параметр: математическое ожидание m_x каждой X_i, но из-за вычитания в скобках зависимость от m_x пропадает. Теперь по аналогии с (20.1) или (20.7) нужно перейти к стандартной величине. Но если там для центрирования мы вычитали из случайной величины ее математическое ожидание, то здесь удобнее делить D_x^* на D_x, которая является в данном случае математическим ожиданием D_x^*. При вычитании мы получали нулевое математическое ожидание, а при делении получим единичное, и останется зависимость только от одного параметра f. Полученная случайная величина D_x^*/D_x с точностью до множителя f совпадает с другой величиной:

$$\chi^2 = \frac{1}{D_x}\sum_{i=1}^{n}\left(X_i - M_x^*\right)^2, \tag{20.10}$$

которую ввел в рассмотрение Карл Пирсон (Karl Pearson, 1857—1936, рис. 20.3).

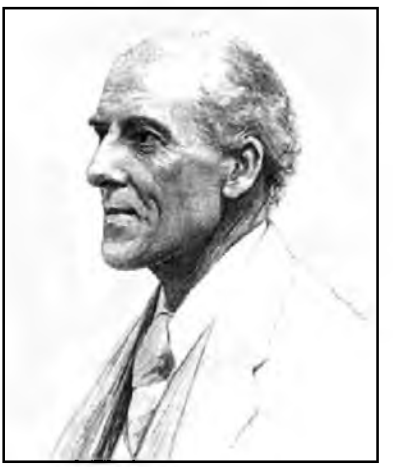


Рис. 20.3. К. Пирсон

Определение 20.2. Распределение случайной величины (20.10) называется χ^2-*распределением Пирсона.* □

Вообще-то Пирсон выводил эту величину из других соображений, но, как оказалось, она прекрасно подходит для наших целей: оценки генеральной дисперсии. Вот его определение.

Определение 20.3. χ^2-*распределением Пирсона* с f степенями свободы называется распределение случайной величины, равной сумме f квадратов независимых стандартных нормальных величин:

$$\chi^2 = \sum_{i=1}^{f} U_i^2, \tag{20.11}$$

где все U_i независимы и имеют нормальное распределение с нулевым математическим ожиданием и единичным среднеквадратичным отклонением. □

Можно показать эквивалентность определений 20.2 и 20.3, что мы оставляем читателю для самостоятельной работы.

Распределение величины χ^2 зависит от одного параметра — числа степеней свободы f, причем $M(\chi^2)=f$ Вот как выглядят (рис. 20.4) графики плотности χ^2-распределения Пирсона для различных значений f

```
chi2=linspace(0,20)'; % аргументы
v=[2 3 5 10 20]; % степени свободы
clear Y
for k=1:length(v)
  Y(:,k)=chi2pdf(chi2,v(k)); % плотность chi2-распределения
end
figure
plot(chi2,Y)
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfГрафик плотности \rm\chi^2\bf-распределения Пирсона')
xlabel('\chi^2') % метка оси OX
ylabel('\itf\rm(\chi^2)') % метка оси OY
```

Это распределение — несимметричное. Максимальное значение плотности χ^2-распределения Пирсона не совпадает с математическим ожиданием и достигается при $\chi^2=f-2$.

Вернемся к нахождению доверительного интервала для генеральной дисперсии D_x. Сравнивая (20.9) и (20.10), можно записать:

$$\chi^2 = \frac{f D_x^*}{D_x}. \tag{20.12}$$

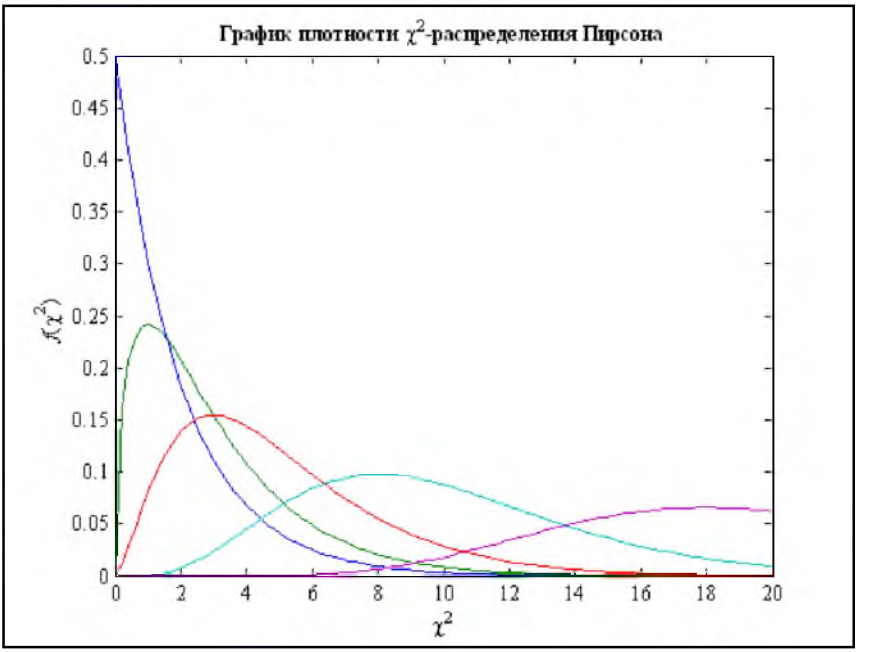


Рис. 20.4. График плотности χ^2-распределения Пирсона, выполненный с помощью MATLAB

Величина χ^2 на уровне значимости $q = 1 - p$ попадает в квантильные границы:

$$\chi^2_{\frac{q}{2}} \le \frac{f\, D_x^*}{D_x} \le \chi^2_{1-\frac{q}{2}}. \qquad (20.13)$$

Решая (20.13) относительно D_x, получим доверительный интервал:

$$\frac{f\, D_x^*}{\chi^2_{1-\frac{q}{2}}} \le D_x \le \frac{f\, D_x^*}{\chi^2_{\frac{q}{2}}}. \qquad (20.14)$$

Здесь уже D_x^* — не случайная величина (20.9), а ее реализация (18.26), которую мы также обозначаем большой буквой.

Как и в случае с математическим ожиданием, при использовании этой формулы мы можем применить методику текущих измерений *(см. раздел 18.6)*.

Пример 20.2 (продолжение примера 18.8). Если использовать только свои $n_1 = 20$ опытов, по которым получено $D_1^* = 6{,}8$, то по формуле (20.14) на уровне значимости $q = 0{,}1$ получим доверительный интервал для генеральной дисперсии [4,286; 12,77]. Привлечение текущих измерений позволяет взять

$n=38$, $D_x^* \approx 6{,}611$, $f=36$. С этими данными имеем такой доверительный интервал: [4,667; 10,228]. Эта оценка — более точная, чем в первом случае. □

Продолжим выполнение задания по обработке массива данных. Найдем доверительные интервалы для D_x в нашей выборке.

```
chi2l=chi2inv(1-q/2,f);
chi2r=chi2inv(q/2,f); % квантили chi2-распределения Пирсона
Dxd=[p,f*Dx./chi2l,f*Dx./chi2r]';  % формула (20.14)
disp('Доверительные интервалы для генеральной дисперсии')
fprintf('p=%8.4f:   %9.6f<=Dx<=%9.6f\n',Dxd)
Доверительные интервалы для генеральной дисперсии
p=  0.9000:    0.896600<=Dx<= 1.247774
p=  0.9500:    0.870266<=Dx<= 1.290451
p=  0.9900:    0.821723<=Dx<= 1.379592
p=  0.9990:    0.769853<=Dx<= 1.493501
```

Здесь справедливо то же замечание, что и для математического ожидания. Если закон распределения генеральной совокупности не нормальный или заранее не известен, то полученные доверительные интервалы можно считать лишь приближенными.

20.3. Оценка других генеральных параметров

К сожалению, для выборочных асимметрии и эксцесса (18.31—18.32) законы распределения соответствующих случайных величин изучены недостаточно. У нас есть только их математические ожидания (18.33—18.34) и дисперсии (18.35—18.36), но нет ни функций, ни плотностей распределения. Поэтому мы можем дать лишь очень приближенные оценки для доверительных интервалов с помощью неравенства Чебышева (18.7). Вероятность больших отклонений (правая часть неравенства Чебышева) равна уровню значимости:

$$\frac{D_x}{\eta^2} = q. \tag{20.15}$$

Отсюда находим полуширину доверительного интервала η, соответствующую уровню значимости q:

$$\eta = \frac{\sigma_x}{\sqrt{q}}. \tag{20.16}$$

На уровне значимости q случайная величина X будет находиться в пределах:

$$-\frac{\sigma_x}{\sqrt{q}} \le X - m_x \le \frac{\sigma_x}{\sqrt{q}}, \tag{20.17}$$

откуда находим доверительный интервал для математического ожидания:

$$X - \frac{\sigma_x}{\sqrt{q}} \le m_x \le X + \frac{\sigma_x}{\sqrt{q}}. \tag{20.18}$$

Эта формула дает верхнюю оценку (самый широкий интервал) и годится для любых видов распределений. Ее есть смысл применять, только если закон распределения X неизвестен. Если же в нашем распоряжении есть функция или плотность распределения X, то нужно попытаться найти более точные оценки, как это сделал У. С. Госсет для математического ожидания нормального распределения. У нас же законы распределения величин A_x^* и E_x^* неизвестны, поэтому применяем (20.18):

$$a_x^* - \sqrt{\frac{D\left(A_x^*\right)}{q}} \le a_x \le a_x^* + \sqrt{\frac{D\left(A_x^*\right)}{q}}; \tag{20.19}$$

$$e_x^* - \sqrt{\frac{D\left(E_x^*\right)}{q}} \le e_x \le e_x^* + \sqrt{\frac{D\left(E_x^*\right)}{q}}. \tag{20.20}$$

Находим доверительные интервалы для генеральных асимметрии и эксцесса в нашем варианте данных:

```
Da=6*(n-1)/(n+1)/(n+3); % дисперсия Ax
De=24*n*(n-2)*(n-3)/(n+1)^2/(n+3)/(n+5); % дисперсия Ex
fprintf('Da=%10.5f\nDe=%10.5f\n',Da,De)
Axd=[p,Ax-(Da./q).^0.5,Ax+(Da./q).^0.5]';  % формула (20.19)
disp('Доверительные интервалы для генеральной асимметрии')
fprintf('p=%8.4f:   %9.6f<=ax<=%9.6f\n',Axd)
Exd=[p,Ex-(De./q).^0.5,Ex+(De./q).^0.5]';  % формула (20.20)
disp('Доверительные интервалы для генерального эксцесса')
fprintf('p=%8.4f:   %9.6f<=ex<=%9.6f\n',Exd)
Da=   0.02926
De=   0.11136
Доверительные интервалы для генеральной асимметрии
p=  0.9000:    0.063218<=ax<= 1.145115
p=  0.9500:   -0.160851<=ax<= 1.369183
p=  0.9900:   -1.106464<=ax<= 2.314796
p=  0.9990:   -4.805321<=ax<= 6.013654
```

```
Доверительные интервалы для генерального эксцесса
p=  0.9000:   -0.969510<=ex<= 1.141039
p=  0.9500:   -1.406618<=ex<= 1.578148
p=  0.9900:   -3.251305<=ex<= 3.422835
p=  0.9990:   -10.466977<=ex<=10.638507
```

В отличие от доверительных интервалов для m_x и D_x, эти результаты годятся для любых распределений. Но уж очень широкие интервалы получаются!

20.4. Выявление промахов

Лучше всего грубые ошибки измерений (промахи) выявить и исключить из рассмотрения в самом начале обработки данных. Самый надежный способ здесь — тщательная проверка условий испытаний, их однородности. Однако на практике подобный анализ не всегда возможен, особенно когда исследователь имеет дело с готовым цифровым материалом. В этом случае нужно попытаться выявить промахи по самой выборке. Например, на выборке, показанной на рис. 20.5, *а*, отмеченный элемент — почти наверняка промах. Ситуация на рис. 20.5, *б* менее определенная. Имеем ли мы дело с промахом или это большое случайное отклонение? В первом случае нужно исключить это измерение, а во втором этого нельзя делать, т. к. исказятся результаты.

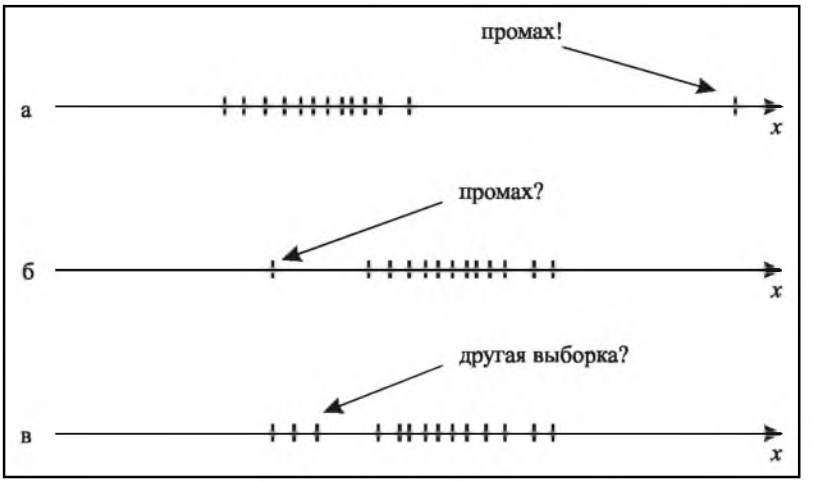


Рис. 20.5. Выявление промахов

Как же отличить промах от большой случайной ошибки? Ответ на этот вопрос дает *принцип 19.2* практической невозможности. Предположим, что распределение экспериментальных результатов нормальное. Тогда любой про-

мах приводит к нарушению нормальности выборки, и это нарушение можно выявить.

Рассмотрим следующую задачу. Имеется выборка $x_1, x_2, ..., x_n$. На уровне значимости q требуется оценить, совместимы ли элементы выборки с гипотезой о том, что они взяты из одной и той же генеральной совокупности с нормальным распределением.

Если сомнение вызывают несколько элементов выборки, как показано на рис. 20.5, *в*, нужно выделить их в отдельную выборку и сравнить 2 выборки *(см. главу 22)*. Но обычно подозрение вызывают 1 или 2 крайних элемента выборки. Рассмотрим, как можно проверить их на совместимость с остальной выборкой.

Если известны математическое ожидание и дисперсия (или среднеквадратичное отклонение) генеральной совокупности m_x, D_x, σ_x, то можно утверждать, что на уровне значимости q одно измерение x должно лежать в квантильных границах:

$$-\sigma_x u_{1-\frac{q}{2}} \le x - m_x \le \sigma_x u_{1-\frac{q}{2}}. \qquad (20.21)$$

Поэтому, если проведено одно измерение и оно не укладывается в (20.21), то его нужно считать ошибочным (промахом). Но с увеличением объема испытаний вероятность больших отклонений возрастает. Доверительному интервалу (20.21) соответствует доверительная вероятность в одном испытании, равная $p=1-q$. Для n независимых испытаний тогда доверительная вероятность, соответствующая интервалу (20.21), будет равна $p^n=(1-q)^n \approx 1-nq$ (при малых q). Значит, при n испытаниях доверительному интервалу (20.21) соответствует уровень значимости nq. Чтобы получить доверительный интервал для уровня значимости q, нужно переобозначить $nq \to q$. Получаем:

$$-\sigma_x u_{1-\frac{q}{2n}} \le x_i - m_x \le \sigma_x u_{1-\frac{q}{2n}}. \qquad (20.22)$$

Любое x_i нужно считать ошибочным на уровне значимости q, если оно противоречит (20.22).

На практике обычно генеральные параметры неизвестны, а есть лишь выборочные m_x^*, D_x^*, σ_x^*. Поэтому, если мы из любого x_i вычтем m_x^*, а полученную величину разделим на σ_x^*, то мы получим вовсе не стандартное нормальное распределение U!

Определение 20.4. Пусть генеральная совокупность X имеет нормальное распределение. Возьмем выборку из n значений $x_1, x_2, ..., x_n$. Вычислим по ним m_x^*, D_x^*, σ_x^*. Найдем среди x_i крайний элемент: такой, для которого

$|x_i - m_x^*| \to \max$. Обозначим этот элемент x_b. Возьмем случайные величины X_b, M_x^* и σ_x^*, реализациями которых являются соответственно x_b, m_x^* и σ_x^*. Вычислим случайную величину:

$$\tau = \frac{\left|X_b - M_x^*\right|}{\sigma_x^*}. \qquad (20.23)$$

Распределение величины τ называется τ-*распределением максимального относительного отклонения*. □

Оно зависит от одного параметра n — объема выборки. Таблицы его квантилей есть, например, в [38]. Если выборочное значение τ не очень большое:

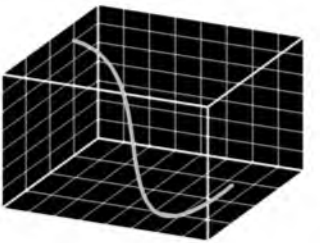
$$\frac{\left|x_b - m_x^*\right|}{\sigma_x^*} \le \tau_{1-q}(n), \qquad (20.24)$$

то крайний элемент можно считать совместимым с выборкой на уровне значимости q. Нарушение (20.24) требует отбросить x_b как промах. Здесь через $\tau_{1-q}(n)$ обозначен "правый" квантиль τ-распределения для объема выборки n, соответствующий уровню значимости q. Мы здесь используем односторонний критерий, т. к. малым значениям τ соответствует совместимость крайнего элемента с выборкой.

20.5. Вопросы для самопроверки

1. Найдите в [1, 27] выражения для плотности и функции t-распределения Стьюдента и χ^2-распределения Пирсона.
2. Докажите эквивалентность двух определений χ^2-распределения Пирсона: 20.2 и 20.3.
3. Попробуйте вывести законы распределения случайных величин A_x^* и E_x^*. Если вам это удастся сделать, то, возможно, соответствующие распределения будут носить ваше имя.
4. Найдите в [51] или запрограммируйте в соответствии с [38] процедуру для проверки крайнего элемента выборки. Проверьте свои данные.

ГЛАВА 21



Анализ закона распределения

В предыдущей *главе 20* мы находили числовые параметры распределения. Сейчас мы рассмотрим, как по выборке найти закон распределения, т. е. функцию $F(x)$ или плотность $f(x)$ распределения.

21.1. Простейший критерий проверки основной гипотезы

Определение 21.1. *Основной гипотезой* называется гипотеза о нормальном распределении генеральной совокупности X. □

У нормальной величины генеральные асимметрия и эксцесс равны нулю. Поэтому выборочные асимметрия a_x^* и эксцесс e_x^* должны не очень сильно отличаться от нуля. А что такое "не очень сильно" — мы уже знаем: найденные в *главе 20* доверительные интервалы для a_x и e_x должны перекрывать (включать в себя) нуль. Если это выполняется, то у нас нет оснований отвергать основную гипотезу, а если нет — ее нужно отвергнуть. Для нашего варианта данных доверительный интервал для a_x на самом "жестком" уровне значимости $q = 0{,}1$ не включает 0, поэтому у нас есть основания отвергнуть основную гипотезу: уж очень большой оказалась выборочная асимметрия a_x^*. Примерно такой же критерий (тест Бера — Жарка) реализован в MATLAB в виде функции `jbtest`.

Этот критерий — очень приближенный. К тому же он дает ответ только на один вопрос: нормальное распределение или нет. Поэтому он обычно используется на начальных этапах исследования распределения. Сейчас мы изучим более точные критерии, которые годятся для проверки не только основной гипотезы, но и гипотез о других распределениях. В этих критериях мы будем проверять, насколько хорошо наша выборка *согласуется* с предполагаемым

теоретическим распределением, поэтому эти критерии называются критериями согласия.

Определение 21.2. *Критерием согласия* называется критерий проверки правильности подбора теоретического распределения на его соответствие выборке. □

Чтобы применить критерии согласия, нужно вначале подобрать теоретическое распределение. Рассмотрим, как это делается.

21.2. Подбор теоретического распределения и его параметров

Подбор теоретического распределения состоит из следующих этапов:

1. Подбор вида распределения (т. е. закона).
2. Подбор параметров распределения (т. е. чисел, входящих в выражение для функции и плотности распределения).
3. Проверка правильности подбора.

В этом разделе мы подберем вид теоретического распределения и его параметры (пп. 1 и 2 из приведенного списка). В следующих разделах мы проверим правильность подбора с помощью критериев согласия Колмогорова и Пирсона.

21.2.1. Вид теоретического распределения

Различные законы распределения отличаются видом графиков $F(x)$ и $f(x)$. Из математического анализа мы знаем, что при интегрировании функции сглаживаются, а при дифференцировании, наоборот, их особенности проявляются все сильнее. Поэтому график плотности распределения $f(x)$ несет гораздо больше информации о виде распределения, чем график $F(x)$. Это несколько усложняет нашу задачу: ведь выборочную функцию распределения мы умеем строить (18.1), а выборочную плотность — пока еще нет. Давайте учиться.

По определению плотность распределения $f(x)$ — это предел отношения вероятности попадания в малый интервал к ширине этого интервала, когда ширина стремится к нулю. Для выборки выборочная вероятность попадания в некоторый интервал — это отношение числа попаданий в интервал n_j к общему числу попаданий n. Если ее разделить на ширину интервала h, то при малых h мы и получим выборочную плотность распределения:

$$f^*(x) = \frac{n_j}{nh}. \qquad (21.1)$$

Здесь мы не сможем использовать x_i поодиночке, их придется группировать по участкам. Поэтому вначале весь интервал изменения данных $[x_{min}, x_{max}]$ нужно разбить на участки одинаковой длины. Сколько участков взять? Есть несколько подходов к определению числа участков разбиения k. Один из них — это использование формулы Стэрджесса:

$$k = \lfloor 1 + 3{,}322 \lg n \rfloor, \tag{21.2}$$

где n — объем выборки, а $\lfloor ... \rfloor$ — операция округления до ближайшего целого. Другой подход состоит в следующем. С одной стороны, число участков разбиения должно быть как можно больше, а с другой стороны, в каждый из этих участков должно попадать как можно больше значений x_i. Компромисс между этими требованиями приводит к тому, что обычно выбирают число участков k для построения гистограммы как ближайшее целое к корню квадратному из n:

$$k = \lfloor \sqrt{n} \rfloor. \tag{21.3}$$

После разбиения $[x_{min}, x_{max}]$ на k участков подсчитываем число попаданий в каждый из них n_j.

Определение 21.3. Столбиковая диаграмма числа попаданий в каждый участок n_j называется *гистограммой*. □

Из (21.1) следует, что гистограмма с точностью до множителя nh совпадает с графиком выборочной плотности распределения $f^*(x)$. Разделив ординаты гистограммы на nh, мы получим график $f^*(x)$.

Для построения гистограммы в MATLAB имеется функция `hist`. Она автоматически разбивает интервал изменения выборки на нужное количество участков, подсчитывает n_j и строит график.

Продолжим выполнение задания "Обработка массива данных". В нижеприведенной области ввода первая строка — это определение числа участков k. Сейчас здесь стоит $\lfloor \sqrt{n} \rfloor$. Если вы хотите использовать формулу Стэрджесса, измените эту строку. Определим ширину каждого интервала h (идентификатор `d` в программе). Построим гистограмму распределения (рис. 21.1).

```
k=round(n^0.5); % число интервалов для построения гистограммы
d=(xmax-xmin)/k; % ширина каждого интервала
del=(xmax-xmin)/20; % добавки влево и вправо
xl=xmin-del;
xr=xmax+del; % границы интервала для построения графиков
fprintf('Число интервалов k=%d\n',k)
fprintf('Ширина интервала h=%14.7f\n',d)
```

```
figure % создаем новую фигуру
hist(x,k) % построили гистограмму
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfГистограмма') % заголовок
xlim([xl xr]) % границы по оси OX
xlabel('\itx_{j}') % метка оси x
ylabel('\itn_{j}') % метка оси y
Число интервалов k=14
Ширина интервала h=     0.3864458
```

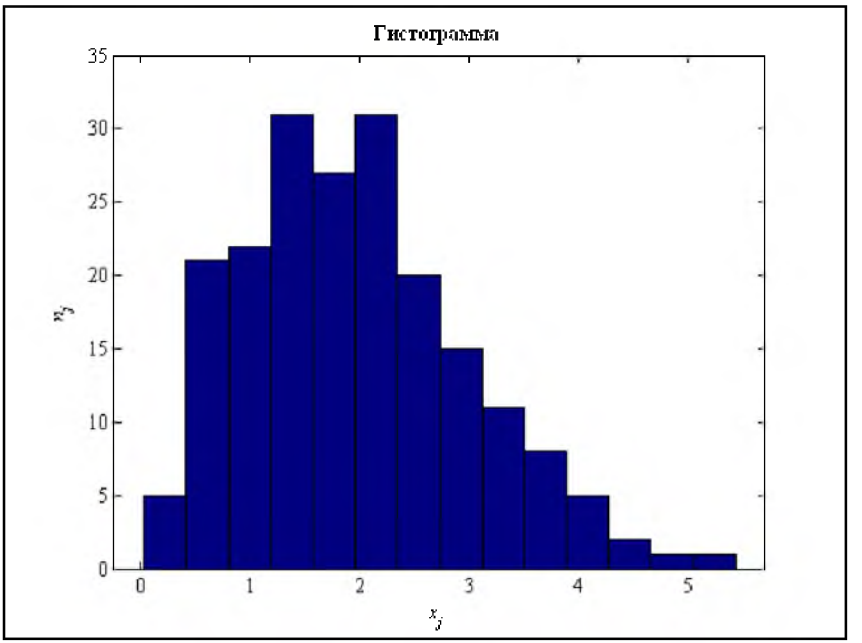


Рис. 21.1. Гистограмма распределения, выполненная с помощью MATLAB

По виду гистограммы подбирается теоретический закон распределения. Для этого смотрим, на какую плотность распределения похожа гистограмма и выбираем соответствующий закон. В этом задании выбор небольшой. Мы рассматриваем только 4 из наиболее часто встречающихся в приложениях законов распределения:

1. Нормальное.
2. Показательное (экспоненциальное).
3. Равномерное.
4. Рэлеевское.

Нарисуем с помощью MATLAB графики соответствующих плотностей распределения. Они показаны на рис. 21.2—21.5. Здесь для вычисления $f(x)$ используется функция `pdf`, которая находит плотность любого из имеющихся в MATLAB видов распределений. Можно использовать и другой вариант: вычислять каждую плотность распределения с помощью своей функции: `normpdf`, `exppdf` и т. д.

```
tdistr={'norm','exp','unif','rayl'}; % названия
pardistr=[[2 1];[2,0];[0 4];[1 0]]; % параметры
ndistr=length(tdistr); % количество распределений
xpl=[-1:0.01:5]'; % абсциссы для графиков
for idistr=1:ndistr, % заполняем и строим графики
  ypdf=pdf(tdistr{idistr},xpl,...
    pardistr(idistr,1),pardistr(idistr,2)); % ординаты
  figure % новая фигура
  plot(xpl,ypdf); % рисуем
  set(get(gcf,'CurrentAxes'),...
    'FontName','Times New Roman Cyr','FontSize',10)
  title(['\bfПлотность распределения ' tdistr{idistr}])
end
```

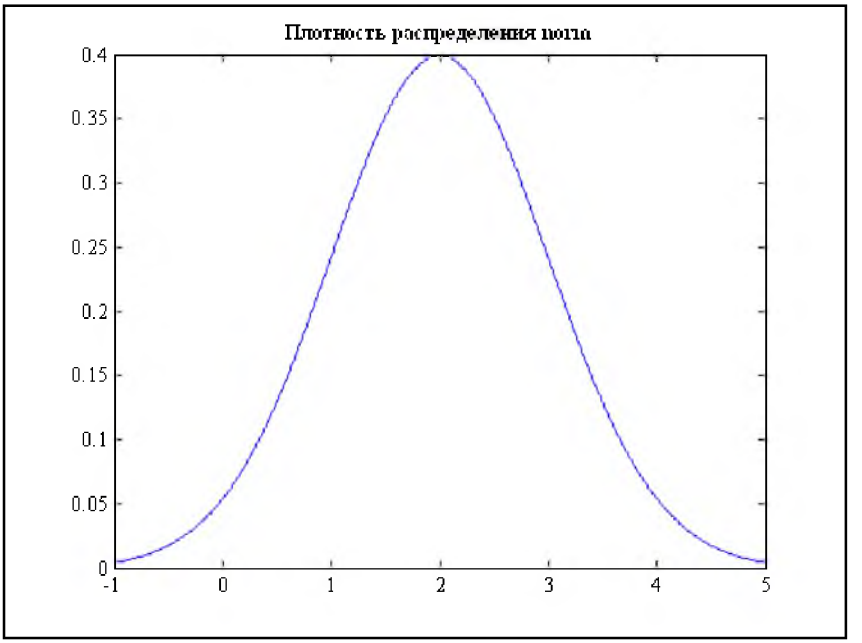


Рис. 21.2. График плотности нормального распределения, выполненный с помощью MATLAB

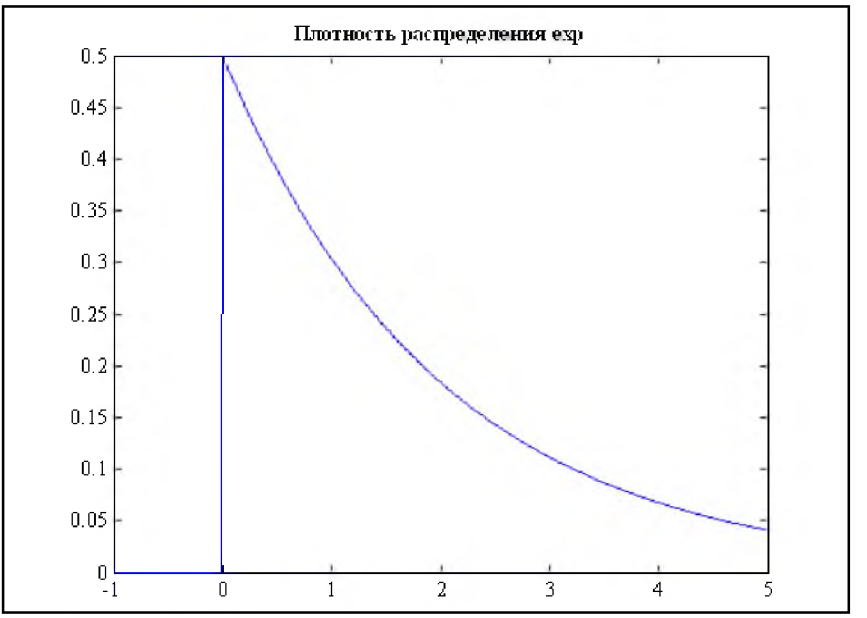


Рис. 21.3. График плотности экспоненциального распределения, выполненный с помощью MATLAB

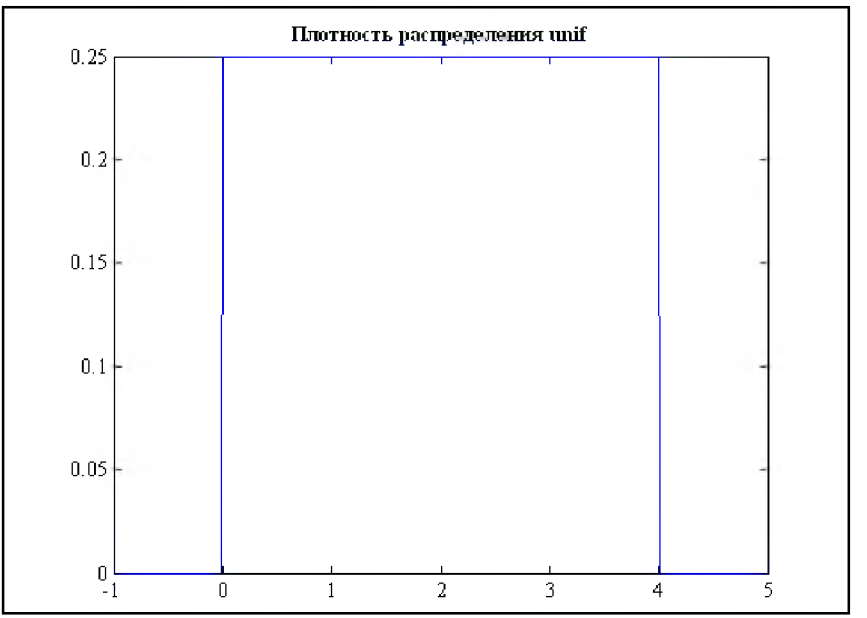


Рис. 21.4. График плотности равномерного распределения, выполненный с помощью MATLAB

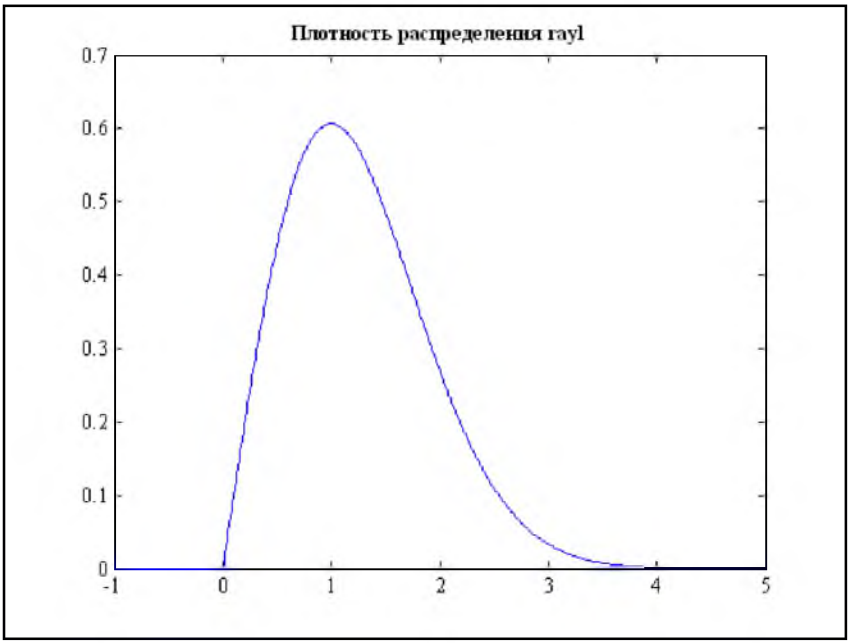


Рис. 21.5. График плотности рэлеевского распределения, выполненный с помощью MATLAB

Плотность нормального распределения — колоколообразная кривая, симметричная относительно некоторой вертикальной оси, но она может быть смещена по горизонтали относительно оси *Oy*. Значения *x* могут быть разного знака. Выражение для плотности нормального распределения имеет вид:

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-m)^2}{2\sigma^2}}, \tag{21.4}$$

а функция распределения:

$$F(x)=\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x}e^{-\frac{(t-m)^2}{2\sigma^2}}dt=\Phi\left(\frac{x-m}{\sigma}\right)+0.5, \tag{21.5}$$

где $\Phi(u)$ — интеграл Лапласа, для которого есть таблицы. Если считать функцию нормального распределения вручную, то удобно пользоваться таблицами интеграла Лапласа, которые есть в любом учебнике по теории вероятностей. При использовании MATLAB в этом нет необходимости: там есть функции `normpdf` и `normcdf`, а также функции `pdf` и `cdf`, в которых первый параметр (название распределения) должен иметь значение `'norm'`. В выражение для плотности и функции нормального распределения входят 2 парамет-

ра: m и σ, поэтому нормальное распределение является *двухпараметрическим*. По нормальному закону обычно распределена ошибка наблюдений, если на результат эксперимента влияет много мелких независимых факторов.

Плотность показательного распределения отлична от нуля только для неотрицательных значений x. В нуле она принимает максимальное значение, равное α. С ростом x она убывает, оставаясь вогнутой и асимптотически приближаясь к 0. Выражение для плотности показательного распределения:

$$f(x)=\begin{cases}0; & x<0;\\ \alpha e^{-\alpha x}; & x\geq 0;\end{cases} \tag{21.6}$$

а для функции распределения:

$$F(x)=\begin{cases}0; & x<0;\\ 1-e^{-\alpha x}; & x\geq 0.\end{cases} \tag{21.7}$$

Показательное распределение является *однопараметрическим*: функция и плотность его зависят от одного параметра α. По показательному закону распределен интервал времени между однотипными случайными событиями: вызовами на АТС, заказами в фирму, страховыми случаями. Для вычисления плотности и функции показательного распределения в MATLAB есть функции `exppdf` и `expcdf`, а также функции `pdf` и `cdf` с первым параметром `'exp'`. Обратите внимание: в MATLAB параметр показательного распределения — это величина, обратная α в формулах (21.6—21.7).

Плотность равномерного распределения отлична от нуля только в заданном интервале $[a, b]$, и принимает в этом интервале постоянное значение:

$$f(x)=\begin{cases}\dfrac{1}{b-a}; & x\in[a,b];\\ 0; & x\notin[a,b].\end{cases} \tag{21.8}$$

Функция равномерного распределения левее точки a равна нулю, правее b — единице, а в интервале $[a, b]$ изменяется по линейному закону:

$$F(x)=\begin{cases}0; & x<a;\\ \dfrac{x-a}{b-a}; & x\in[a,b];\\ 1; & x>b.\end{cases} \tag{21.9}$$

Равномерное распределение — *двухпараметрическое*, т. к. в выражения для $F(x)$ и $f(x)$ входят 2 параметра: a и b. По равномерному закону распределены

ошибка округления и фаза случайных колебаний. В MATLAB плотность и функция равномерного распределения могут быть посчитаны с помощью функций `unifpdf` и `unifcdf`, а также с помощью функций `pdf` и `cdf` с первым параметром `'unif'`.

Плотность рэлеевского распределения отлична от нуля только для неотрицательных значений x. От нуля она выпуклая и возрастает до некоторого максимального значения. Далее с ростом x она убывает, оставаясь выпуклой. Затем становится вогнутой, продолжая убывать, и асимптотически приближается к 0. Выражение для плотности рэлеевского распределения имеет вид:

$$f(x)=\begin{cases}0; & x<0;\\ \dfrac{x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}}; & x\geq 0.\end{cases} \tag{21.10}$$

Функция рэлеевского распределения:

$$f(x)=\begin{cases}0; & x<0;\\ 1-e^{-\frac{x^2}{2\sigma^2}}; & x\geq 0.\end{cases} \tag{21.11}$$

Это распределение *однопараметрическое*: оно зависит от одного параметра σ. По рэлеевскому закону распределено расстояние от точки попадания в мишень до ее центра. Вычисление плотности и функции рэлеевского распределения в MATLAB реализовано с помощью функций `raylpdf`, `raylcdf` или функций `pdf`, `cdf` с первым параметром `'rayl'`.

Посмотрите на свою гистограмму и выберите подходящее распределение. Нетрудно заметить, что в печатном варианте книги гистограмма более всего напоминает рэлеевское распределение. В электронном варианте (на диске) информация обновляется в зависимости от файла обрабатываемых данных.

Разумеется, реальная жизнь гораздо богаче этих примеров. На практике могут встретиться и другие виды распределений (β, χ^2, логнормальное, Вейбулла и т. д.). Многие из них реализованы в MATLAB *(см. главу 27)*, но иногда приходится писать свои функции.

Графики некоторых плотностей распределения похожи между собой, поэтому иногда вид гистограммы позволяет выбрать сразу несколько законов. Если есть какие-либо теоретические соображения предпочесть одно распределение другому, можно их использовать. Если нет — нужно проверить все подходящие законы, а затем выбрать тот, для которого критерии согласия дают лучшие результаты.

21.2.2. Параметры теоретического распределения

Итак, мы определились с видом распределения. Идем дальше. В выражения для теоретической функции распределения $F(x)$ и теоретической плотности распределения $f(x)$ входят различные числовые параметры. В зависимости от их количества распределения бывают 1-параметрические, 2-, 3- и т. д. Так, в рассмотренных выше примерах показательное и рэлеевское распределения — 1-параметрические, а нормальное и равномерное — 2-параметрические. Для определения этих параметров можно применить или ПМП, или метод моментов.

Применение ПМП заключается в следующем. Окружим каждое измерение x_i малой ε-окрестностью. Найдем вероятность того, что случайная величина X_i попадет в этот интервал. Здесь мы используем общую схему выборочного метода и вместо X пишем X_i. Поскольку ε — малое, эта вероятность равна плотности распределения, вычисленной в этой точке x_i, умноженной на ширину интервала 2ε:

$$\mathbf{P}\left(X_i \in \left[x_i - \varepsilon,\, x_i + \varepsilon\right]\right) = p_i = f\left(x_i\right)2\varepsilon. \tag{21.12}$$

Вероятность одновременного попадания во все интервалы в силу независимости испытаний равна произведению p_i:

$$\mathbf{P}\left(\forall X_i \in \left[x_i - \varepsilon,\, x_i + \varepsilon\right]\right) = \prod_{i=1}^{n} p_i = \left(2\varepsilon\right)^n \prod_{i=1}^{n} f\left(x_i\right). \tag{21.13}$$

Все эти события проявились на практике, поэтому нам нужно максимизировать (21.13). Обычно ее делят на постоянный множитель $(2\varepsilon)^n$, при этом получается функция, которая называется *функцией правдоподобия*:

$$L = \prod_{i=1}^{n} f\left(x_i\right) \to \max. \tag{21.14}$$

Она зависит от параметров, которые входят в выражение для $f(x)$. Параметры распределения подбираются так, чтобы максимизировать функцию правдоподобия (21.14).

ПРИМЕР 21.1. Предположим, что по выборке $x_1, x_2, \ldots, x_n$ мы подбираем показательное распределение, плотность которого (21.6) зависит от одного параметра α. Составляем выражение для функции правдоподобия:

$$L(\alpha) = \prod_{i=1}^{n} \left(\alpha e^{-\alpha x_i}\right) = \alpha^n e^{-\alpha \sum_{i=1}^{n} x_i} \to \max. \tag{21.15}$$

Для нахождения максимума вычисляем производную по α и приравниваем ее нулю.

$$\frac{\partial L(\alpha)}{\partial \alpha} = n\alpha^{n-1} e^{-\alpha \sum_{i=1}^{n} x_i} - \alpha^n e^{-\alpha \sum_{i=1}^{n} x_i} \sum_{i=1}^{n} x_i = 0. \qquad (21.16)$$

Сокращаем на экспоненту, которая больше нуля, и на множитель α^{n-1}, который также больше нуля, т. к. параметр показательного распределения $\alpha > 0$. Остается:

$$n - \alpha \sum_{i=1}^{n} x_i = 0, \qquad (21.17)$$

т. е. значение параметра α, доставляющее максимум функции правдоподобия (21.15):

$$\alpha_{\max} = \frac{n}{\sum_{i=1}^{n} x_i}. \qquad (21.18)$$

Нетрудно показать, что это значение действительно доставляет функции правдоподобия максимум, а не минимум. Это читатель может проделать самостоятельно. □

Более простым является метод моментов. В нем параметры, входящие в выражения для $F(x)$ и $f(x)$, подбираются так, чтобы вычисленные по этим параметрам математическое ожидание (для 1-параметрических законов) или математическое ожидание и дисперсия (для 2-параметрических законов) совпали с выборочными. Для 3-параметрических законов распределения должны совпадать среднее, дисперсия и асимметрия и т. д.

ПРИМЕР 21.2. В выражениях для плотности и функции нормального распределения (21.4—21.5) параметры m и σ являются математическим ожиданием и среднеквадратичным отклонением. Поэтому, если мы остановились на нормальном распределении, то берем их равными, соответственно, выборочным математическому ожиданию и среднеквадратичному отклонению:

$$m = m_x^*; \quad \sigma = \sigma_x^*. \qquad (21.19)$$

Математическое ожидание показательного распределения есть величина, обратная его параметру α. Поэтому, если мы выбрали показательное распределение, параметр α находим:

$$\alpha = \frac{1}{m_x^*}, \qquad (21.20)$$

что совпадает с оценкой по ПМП (21.18).

Из выражений для m_x и σ_x равномерного распределения находим его параметры a и b:

$$a = m_x^* - \sigma_x^*\sqrt{3}; \quad a = m_x^* + \sigma_x^*\sqrt{3}. \tag{21.21}$$

Параметр σ рэлеевского распределения также находится из выражения для m_x:

$$\sigma = m_x^*\sqrt{\frac{2}{\pi}}. \ \square \tag{21.22}$$

В системе MATLAB вычисление параметров теоретического распределения с помощью ПМП реализовано в функциях `*fit` или `mle`. Подбор с помощью метода моментов не реализован. Найдем параметры теоретического распределения (для всех четырех вариантов) по ПМП и методу моментов.

```
s={'нормальное распределение'; 'показательное распределение'; ...
  'равномерное распределение'; 'рэлеевское распределение'};
disp('Параметры по ПМП:')
[mx,sx]=normfit(x); % параметры нормального распределения
lam=1/expfit(x); % параметр показательного распределения
[a,b]=unifit(x); % параметры равномерного распределения
sig=raylfit(x); % параметр рэлеевского распределения
fprintf(['%s: m=%12.7f;  sigma=%12.7f\n'],s{1},mx,sx)
fprintf('%s: alpha=%12.7f\n',s{2},lam)
fprintf('%s: a=%12.7f;  b=%12.7f\n',s{3},a,b)
fprintf('%s: sigma=%12.7f\n',s{4},sig)
disp('Параметры по методу моментов:')
mx=Mx;
sx=Sx; % параметры нормального распределения
lam=abs(1/Mx); % параметр показательного распределения
a=Mx-Sx*3^0.5;
b=Mx+Sx*3^0.5; % параметры равномерного распределения
sig=abs(Mx)*(2/pi)^0.5; % параметр рэлеевского распределения
fprintf(['%s: m=%12.7f;  sigma=%12.7f\n'],s{1},mx,sx)
fprintf('%s: alpha=%12.7f\n',s{2},lam)
fprintf('%s: a=%12.7f;  b=%12.7f\n',s{3},a,b)
fprintf('%s: sigma=%12.7f\n',s{4},sig)
Параметры по ПМП:
нормальное распределение: m=   1.9607780;  sigma=   1.0243977
показательное распределение: alpha=   0.5100017
равномерное распределение: a=   0.0281344;  b=   5.4383751
рэлеевское распределение: sigma=   1.5634567
Параметры по методу моментов:
нормальное распределение: m=   1.9607780;  sigma=   1.0243977
```

```
показательное распределение: alpha=   0.5100017
равномерное распределение: a=   0.1864691;  b=   3.7350868
рэлеевское распределение: sigma=   1.5644745
```

Здесь параметры по методу моментов вычислены позже, поэтому именно они будут использоваться в дальнейших расчетах (они записываются в одни и те же переменные). Если вы хотите использовать параметры, найденные по ПМП, поменяйте местами соответствующие блоки в программе.

Построим на одном графике теоретическую и эмпирическую плотности распределения. Эмпирическая плотность распределения — это та же гистограмма, у которой масштаб по оси ординат изменен таким образом, чтобы площадь под кривой стала равна 1. Для этого все метки по оси ординат в гистограмме нужно разделить на nh, где n — число экспериментальных данных, а h — ширина интервала при построении гистограммы. Теоретическую плотность распределения строим по одной из формул (21.4), (21.6), (21.8) или (21.10). Параметры для них у нас вычислены. Эмпирическую плотность распределения нарисуем красной линией, а предполагаемую теоретическую — линией одного из цветов: синего, зеленого, сиреневого или черного. Результат показан на рис. 21.6.

```
[nj,xm]=hist(x,k); % число попаданий и середины интервалов
delta=xm(2)-xm(1); % ширина интервала
clear xfv fv xft ft % очистили массивы для f(x)
xfv=[xm-delta/2;xm+delta/2]; % абсциссы для эмпирической f(x)
xfv=reshape(xfv,prod(size(xfv)),1); % преобразовали в столбец
xfv=[xl;xfv(1);xfv;xfv(end);xr]; % добавили крайние
fv=nj/(n*delta); % значения эмпирической f(x) в виде 1 строки
fv=[fv;fv]; % 2 строки
fv=[0;0;reshape(fv,prod(size(fv)),1);0;0]; % + крайние, 1 столбец
xft=linspace(xl,xr,1000)'; % абсциссы для теоретической f(x)
ft=[normpdf(xft,mx,sx),exppdf(xft,1/lam),...
    unifpdf(xft,a,b),raylpdf(xft,sig)];
col='bgmk'; % цвета для построения графиков
figure
plot(xfv,fv,'-r', xft,ft(:,1),col(1), xft,ft(:,2),col(2), ...
  xft,ft(:,3),col(3), xft,ft(:,4),col(4))  % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfПлотности распределения')
xlim([xl xr]), ylim([0 1.4*max(fv)]) % границы рисунка по осям
xlabel('\itx') % метка оси x
ylabel('\itf\rm(\itx\rm)') % метка оси y
```

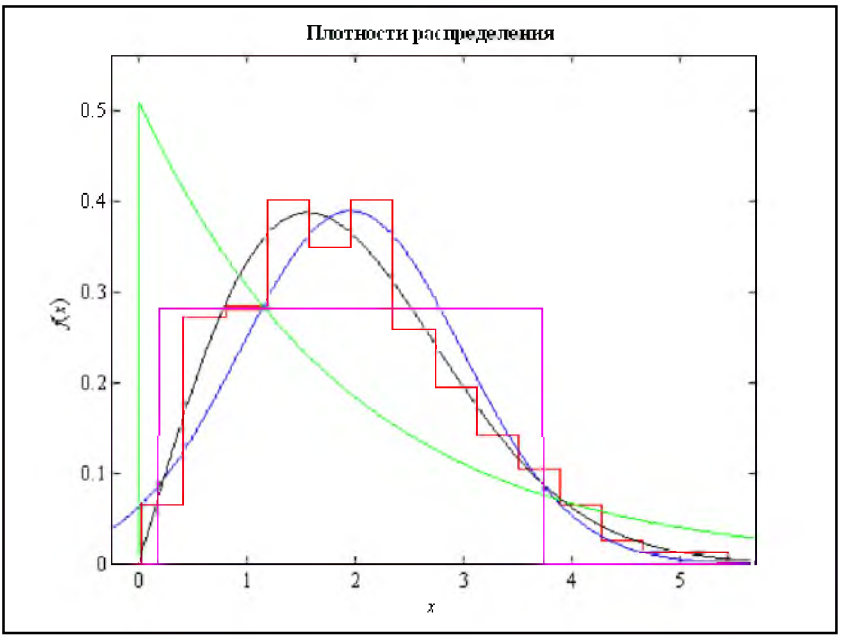


Рис. 21.6. Теоретическая и выборочная плотности распределения, выполненные с помощью MATLAB

На этом графике в одном масштабе нарисованы эмпирическая плотность распределения и теоретическая. Какое теоретическое распределение лучше всего согласуется с эмпирическим: нормальное, показательное, равномерное или рэлеевское?

21.3. Критерии согласия

Итак, мы подобрали вид теоретического распределения и его параметры. Следующий этап — это проверка правильности подбора. Мы должны выяснить, насколько хорошо подобранное теоретическое распределение согласуется с нашими данными. Закон распределения полностью определяется выражением для функции или плотности распределения. Поэтому применение критериев согласия сводится к сравнению двух функций: $F(x)$ и $F^*(x)$ или $f(x)$ и $f^*(x)$ на всем интервале возможных значений x. Если эти функции отличаются не слишком сильно, то статистическую гипотезу о правильности подбора теоретического распределения (0-гипотезу) можно принять. Если же различие значительно, то нужно отвергнуть 0-гипотезу и принять альтернативную: распределение подобрано неправильно.

Как же сравнить две функции? Можно найти максимальную (по ординатам) разность между ними и взять ее модуль. Другой вариант — посчитать пло-

щадь между кривыми (интеграл от модуля разности) или интеграл от квадрата разности. Любая из этих величин может служить мерой расхождения двух функций. Два наиболее широко распространенных критерия согласия как раз и используют эти величины. Первая из них участвует в критерии согласия Колмогорова, а вторая — в критерии согласия Пирсона.

21.3.1. Критерий согласия Колмогорова

В этом критерии сравниваются между собой выборочная функция распределения (18.1) и подобранная теоретическая $F(x)$. Сравнивается максимальная по модулю разность между ними. С точки зрения выборочного метода $F^*(x)$ является случайной функцией, т. к. от реализации к реализации ее вид меняется. Взяв другую выборку того же объема n, мы получим в общем случае другие x_i, и, следовательно, другую $F^*(x)$. Отсюда следует, что максимальная по модулю разность между выборочной и генеральной функциями распределения:

$$D = \max_{\forall x \in R} \left| F^*(x) - F(x) \right| \tag{21.23}$$

является случайной величиной. Теорема Гливенко — Кантелли (18.2) утверждает, что эта величина с ростом объема выборки сходится по вероятности к нулю. Андрей Николаевич Колмогоров (1903—1987, рис. 21.7) уточнил этот результат: он выяснил, как именно D сходится к нулю. Он рассмотрел случайную величину

$$\Lambda = D\sqrt{n} \tag{21.24}$$

и нашел ее закон распределения. Как оказалось, при достаточно больших n он вообще не зависит от закона распределения генеральной совокупности X.

Рис. 21.7. А. Н. Колмогоров

■ **ТЕОРЕМА 21.1 (Колмогорова).** Для любого непрерывного закона распределения генеральной совокупности X функция распределения случайной величины Λ (21.24) при достаточно больших n имеет вид:

$$F(\lambda) = \sum_{k=-\infty}^{+\infty} (-1)^k e^{-2k^2\lambda^2}. \qquad (21.25)$$

Доказательство есть в [46]. □

Определение 21.4. Распределение случайной величины Λ называется *распределением Колмогорова.* □

Эта теорема дает нам возможность проверить правильность подбора теоретического распределения. Если опытные данные x_i действительно взяты из генеральной совокупности с функцией распределения $F(x)$, то вычисленная по (21.23—21.24) реализация λ случайной величины Λ на уровне значимости q должна лежать в квантильных границах распределения Колмогорова (21.25). При этом если λ малое (выходит за "левый" квантиль, см. рис. 19.4), то это — тоже область 0-гипотезы: теоретическое распределение еще лучше согласуется с опытными данными. А вот если λ — большое, то мы должны отвергнуть 0-гипотезу. То есть, здесь мы должны использовать односторонний критерий (см. рис. 19.5): 0-гипотезу можно принять, если выполняется условие:

$$\lambda \le \lambda_{1-q}. \qquad (21.26)$$

Таким образом, для применения критерия согласия Колмогорова (он еще называется критерием Колмогорова — Смирнова) мы должны найти максимальную по модулю разность между выборочной и теоретической функциями распределения D (21.23), вычислить по ней λ — реализацию случайной величины Λ (21.24) и проверить выполнение условия (21.26).

При решении задачи "вручную" квантили распределения Колмогорова берутся из таблиц. В MATLAB проверка по критерию согласия Колмогорова может быть проведена с помощью функции `kstest`. Проверим, какое из распределений лучше всего подходит по критерию согласия Колмогорова. Проверим все четыре типа и выберем тот, для которого критическое значение q максимальное. Нарисуем на одном графике выборочную функцию распределения $F^*(x)$ и наиболее подходящую теоретическую. График выборочной функции распределения рисует функция `cdfplot`. Результат — на рис. 21.8.

```
param=[[mx sx];[lam 0];[a b];[sig 0]]; % параметры распределений
qq=[]; % критические уровни значимости
for idistr=1:ndistr, % критерий Колмогорова
  [hkolm,pkolm,kskolm,cvkolm]=...
    kstest(x,[x cdf(tdistr{idistr},x,...
    param(idistr,1),param(idistr,2))],0.1,0);
```

```
  qq=[qq pkolm]; % критические уровни значимости
end
[maxqq,bdistr]=max(qq); % выбрали лучшее распределение
fprintf(['Лучше всего подходит %s;\nкритический уровень '...
  'значимости для него = %8.5f\n'],s{bdistr},maxqq);
figure
cdfplot(x); % эмпирическая функция распределения
xpl=linspace(xl,xr,500); % для графика F(x)
ypl=cdf(tdistr{bdistr},xpl,param(bdistr,1),param(bdistr,2));
hold on % для рисования на этом же графике
plot(xpl,ypl,'r'); % дорисовали F(x)
hold off
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title(['\bfПодобрано ' s{bdistr}])
xlabel('\itx') % метка оси x
ylabel('\itf\rm(\itx\rm)') % метка оси y
```

```
Лучше всего подходит рэлеевское распределение;
критический уровень значимости для него =  0.99105
```

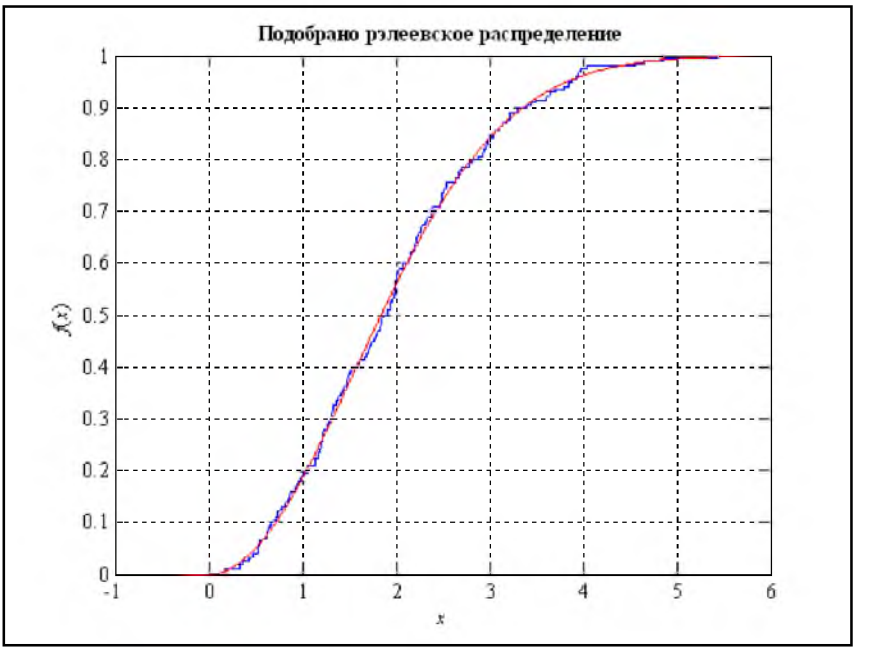


Рис. 21.8. Теоретическая и эмпирическая функции распределения, выполненные с помощью MATLAB

Найденный критический уровень значимости — это то значение q, при котором неравенство (21.26) обращается в равенство. Видно, что даже при очень жестком уровне значимости $q = 0.99$ неравенство (21.26) все еще будет выполняться. В соответствии с *разделом 19.8* мы просто обязаны принять гипотезу о рэлеевском распределении генеральной совокупности, т. к. если мы ее отбросим, то допустим 99%-ную ошибку!

21.3.2. Критерий согласия Пирсона

В критерии согласия Пирсона сравниваются между собой теоретические и эмпирические числа попаданий в интервалы. Интервалы могут быть любыми, не обязательно одинаковой длины, лишь бы теоретическое число попаданий в каждый из них было не менее 5. Удобно, в частности, взять те интервалы, по которым была построена гистограмма. Эмпирические числа попаданий в них n_j у нас есть из гистограммы. Каждое n_j мы сравниваем с теоретическим числом попаданий в этот интервал np_j, где p_j — вероятность попадания величины X в j-й интервал:

$$p_j = \int_{a_j}^{b_j} f(x)\,dx = F(b_j) - F(a_j), \tag{21.27}$$

a_j, b_j — границы j-го интервала. Карл Пирсон показал, что, если все $np_j \geq 5$, суммарная квадратичная относительная разность между теоретическим и практическим числом попаданий в каждый интервал:

$$\chi^2 = \sum_{j=1}^{k} \frac{(n_j - np_j)^2}{np_j} \tag{21.28}$$

имеет приближенно χ^2-распределение Пирсона с $k - m$ степенями свободы, где m — число ограничений, равное числу параметров выбранного распределения плюс 1. Так, для 2-параметрических распределений $m = 3$, а для 1-параметрических — 2.

Если эмпирические и теоретические числа попаданий взяты из одной генеральной совокупности, то величина (21.28) действительно должна иметь χ^2-распределение и ее опытное значение должно находиться в соответствующих квантильных границах. Как и в критерии согласия Колмогорова, слишком малое значение (21.28) оставляет нас в области 0-гипотезы. А вот выход за "правый" квантиль говорит о значительном расхождении между теорией и практикой и требует отвергнуть 0-гипотезу. Поэтому применяем 1-сторонний критерий. Теоретическое распределение можно считать подобранным верно

на уровне значимости q, если величина (21.28) будет не очень большой: должно выполняться условие:

$$\sum_{j=1}^{k}\frac{\left(n_j - np_j\right)^2}{np_j} \le \chi^2_{1-q}\left(k-m\right). \qquad (21.29)$$

Построим таблицу результатов, в которую занесем: номера интервалов (1-й столбец), границы интервалов a_j и b_j (2-й и 3-й столбцы), вероятность попадания в интервал p_j (4-й столбец), теоретическое число попаданий np_j (5-й столбец) и практическое число попаданий n_j (6-й столбец).

Границы интервалов и практическое число попаданий возьмем из гистограммы. Теоретическая вероятность попадания в j-й интервал подсчитывается по формуле (21.27).

```
clear Tabl % очистили таблицу результатов
Tabl(:,1)=[1:k]'; % номера интервалов
Tabl(:,2)=xm'-delta/2; % левые границы интервалов
Tabl(:,3)=xm'+delta/2; % правые границы интервалов
Tabl(1,2)=-inf; % теоретическое начало 1-го интервала
Tabl(k,3)=inf; % теоретический конец последнего интервала
Tabl(:,4)=nj'; % опытные числа попаданий
bor=[Tabl(:,2);Tabl(end,3)]; % все границы интервалов
pro=cdf(tdistr{bdistr},bor,param(bdistr,1),param(bdistr,2));
Tabl(:,5)=pro(2:end)-pro(1:end-1); % вероятности попаданий pj
Tabl(:,6)=n*Tabl(:,5); % теоретическое число попаданий npj
disp('Сводная таблица результатов')
fprintf(' j       aj          bj')
fprintf('       nj       pj          npj\n')
fprintf('%2.0f%12.5f%12.5f%6.0f%12.5f%12.5f\n',Tabl')
```

```
Сводная таблица результатов
 j       aj          bj       nj       pj          npj
 1        -Inf     0.41458     5     0.03450     6.90046
 2     0.41458     0.80103    21     0.08835    17.66948
 3     0.80103     1.18747    22     0.12743    25.48640
 4     1.18747     1.57392    31     0.14685    29.36971
 5     1.57392     1.96036    27     0.14678    29.35603
 6     1.96036     2.34681    31     0.13147    26.29349
 7     2.34681     2.73325    20     0.10725    21.44965
 8     2.73325     3.11970    15     0.08043    16.08630
 9     3.11970     3.50615    11     0.05578    11.15544
10     3.50615     3.89259     8     0.03591     7.18128
11     3.89259     4.27904     5     0.02152     4.30314
```

```
12     4.27904     4.66548     2     0.01202     2.40492
13     4.66548     5.05193     1     0.00628     1.25543
14     5.05193         Inf     1     0.00544     1.08828
```

Если распределение подобрано верно, то числа из 4-го и 6-го столбцов не должны сильно отличаться. Проверим выполнение условия $\forall np_j \geq 5$ и объединим те интервалы, в которых $np_j < 5$. Перестроим таблицу и добавим в нее еще один, 7-й столбец — слагаемые левой части формулы (21.29): $(n_j - np_j)^2/np_j$. Подсчитаем сумму элементов последнего столбца, т. е. левую часть формулы (21.29). Она называется статистикой Пирсона. Сравним ее с квантилем χ^2-распределения Пирсона на заданном уровне значимости и сделаем вывод.

```
qz=0.3; % выбрали уровень значимости
ResTabl=Tabl(1,1:6); % взяли первую строку
for k1=2:k, % берем остальные строки таблицы
  if ResTabl(end,6)<5, % предыдущее npj<5 - будем суммировать
    ResTabl(end,3)=Tabl(k1,3); % новая правая граница интервала
    ResTabl(end,4:6)=ResTabl(end,4:6)+Tabl(k1,4:6); % суммируем
  else % предыдущее npj>=5 - будем дописывать строку
    ResTabl=[ResTabl;Tabl(k1,1:6)]; % дописываем строку
  end
end
if ResTabl(end,6)<5, % последнее npj<5
  ResTabl(end-1,3)=ResTabl(end,3); % новая правая граница
  ResTabl(end-1,4:6)=ResTabl(end-1,4:6)+ResTabl(end,4:6);
  ResTabl=ResTabl(1:end-1,:); % отбросили последнюю строку
end
kn=size(ResTabl,1); % число объединенных интервалов
ResTabl(:,1)=[1:kn]'; % новые номера интервалов
ResTabl(:,7)=(ResTabl(:,4)-ResTabl(:,6)).^2./ResTabl(:,6);
disp('Сгруппированная сводная таблица результатов')
fprintf(' j       aj          bj')
fprintf('       nj       pj          npj     ')
fprintf(['(nj-npj)^2/npj\n'])
fprintf('%2.0f%12.5f%12.5f%6.0f%12.5f%12.5f%12.5f\n',ResTabl')
hi2=sum(ResTabl(:,7)); % сумма элементов последнего столбца
fprintf(['Статистика Пирсона chi2=%10.5f\n'],hi2)
m=[3,2,3,2]; % число ограничений
fprintf('Задаем уровень значимости q=%5.4f\n',qz)
chi2qz=chi2inv(1-qz,kn-m(bdistr)); % квантиль
fprintf(['Квантиль chi2-распределения Пирсона '...
  'chi2(1-q)=%10.5f\n'],chi2qz)
if hi2<=chi2qz,
  disp('Распределение подобрано верно, т. к. chi2<=chi2(1-q)')
```

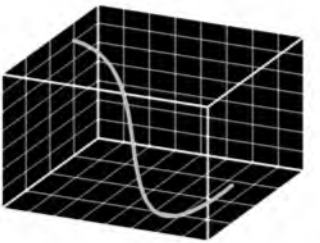
```
else
  disp('Распределение подобрано неверно, т. к. chi2>chi2(1-q)')
end
```

```
Сгруппированная сводная таблица результатов
 j       aj          bj        nj       pj          npj      (nj-npj)^2/npj
 1        -Inf     0.41458      5     0.03450     6.90046      0.52341
 2     0.41458     0.80103     21     0.08835    17.66948      0.62777
 3     0.80103     1.18747     22     0.12743    25.48640      0.47692
 4     1.18747     1.57392     31     0.14685    29.36971      0.09050
 5     1.57392     1.96036     27     0.14678    29.35603      0.18909
 6     1.96036     2.34681     31     0.13147    26.29349      0.84246
 7     2.34681     2.73325     20     0.10725    21.44965      0.09797
 8     2.73325     3.11970     15     0.08043    16.08630      0.07336
 9     3.11970     3.50615     11     0.05578    11.15544      0.00217
10     3.50615     3.89259      8     0.03591     7.18128      0.09334
11     3.89259         Inf      9     0.04526     9.05176      0.00030
Статистика Пирсона chi2=   3.01728
Задаем уровень значимости q=0.3000
Квантиль chi2-распределения Пирсона chi2(1-q)=  10.65637
Распределение подобрано верно, т. к. chi2<=chi2(1-q)
```

21.4. Вопросы для самопроверки

1. В чем состоит простейший критерий проверки 0-гипотезы?
2. Проверка по простейшему критерию показала, что 0-гипотезу нельзя отвергнуть. Значит ли, что распределение X — нормальное? Почему?
3. Как строится гистограмма? Когда количество интервалов больше: при применении формулы (21.2) или (21.3)?
4. Как находятся по ПМП параметры нормального распределения? равномерного? рэлеевского?
5. Каким классам функций *(см. главу 1)* соответствует сравнение двух функций в критериях согласия Колмогорова и Пирсона?
6. Найдите в [46] доказательство теоремы Колмогорова. Где в ходе доказательства исчезает зависимость от закона распределения X?
7. Найдите в литературе или Интернете доказательство критерия согласия Пирсона. Для любых ли распределений оно годится?

ГЛАВА 22



Сравнение выборок

В этой главе мы рассмотрим не одну, а две или более выборок и задачи статистики для них. Будем предполагать, что исходные генеральные совокупности независимы и имеют нормальное распределение и опыты в каждой выборке также независимы.

22.1. Сравнение двух дисперсий

Имеются 2 генеральные совокупности: X_1 и X_2. Из каждой взята выборка объемом n_1 и n_2 соответственно. По формулам (18.14) вычислены выборочные математические ожидания m_1^* и m_2^*, а затем по (18.26) — выборочные дисперсии D_1^* и D_2^*. Требуется проверить 0-гипотезу о равенстве генеральных дисперсий:

$$D_1 = D_2. \tag{22.1}$$

В соответствии с общей схемой выборочного метода считаем, что найденные по (18.26) выборочные дисперсии D_1^* и D_2^* — это реализации соответствующих случайных величин, которые мы обозначим теми же буквами: D_1^* и D_2^*. Они вычисляются по (20.9) и зависят как от параметров, от чисел степеней свободы f_1, f_2 и генеральных дисперсий D_1, D_2 соответственно. Рассмотрим случайную величину:

$$F = \frac{D_1^*}{D_1} : \frac{D_2^*}{D_2} = \frac{D_1^* D_2}{D_2^* D_1}. \tag{22.2}$$

В распределении величины F исчезла (сократилась) зависимость от параметров D_1 и D_2, а осталась только от f_1, f_2. Впервые эту величину рассмотрел Рональд Элмер Фишер (Ronald Aylmer Fisher, 1890—1962, рис. 22.1), поэтому распределение величины F носит его имя.

Рис. 22.1. Р. Э. Фишер

Определение 22.1. Распределение случайной величины (22.2) называется *F-распределением Фишера.* □

Сравнивая формулы (20.9) и (20.10), можно получить другое выражение для величины F:

$$F = \frac{\chi^2(f_1)}{f_1} : \frac{\chi^2(f_2)}{f_2} = \frac{\chi^2(f_1) f_2}{\chi^2(f_2) f_1}. \tag{22.3}$$

Нарисуем на одном рисунке (рис. 22.2) несколько графиков плотностей F-распределения при различных числах степеней свободы.

```
clear all % очистили рабочую область
F=linspace(0,4)'; % аргументы
v1=[2 3 5 10 20]; % степени свободы
v2=[3 5 8 15 30];
for k=1:length(v1)
  Y(:,k)=fpdf(F,v1(k),v2(k)); % плотность F-распределения
end
figure
plot(F,Y)
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfГрафик плотности \rm\itF\rm\bf-распределения Фишера')
xlabel('\itF') % метка оси OX
ylabel('\itf\rm(\itF\rm)') % метка оси OY
```

Рассмотрим, как применяется F-распределение Фишера для проверки 0-гипотезы (22.1).

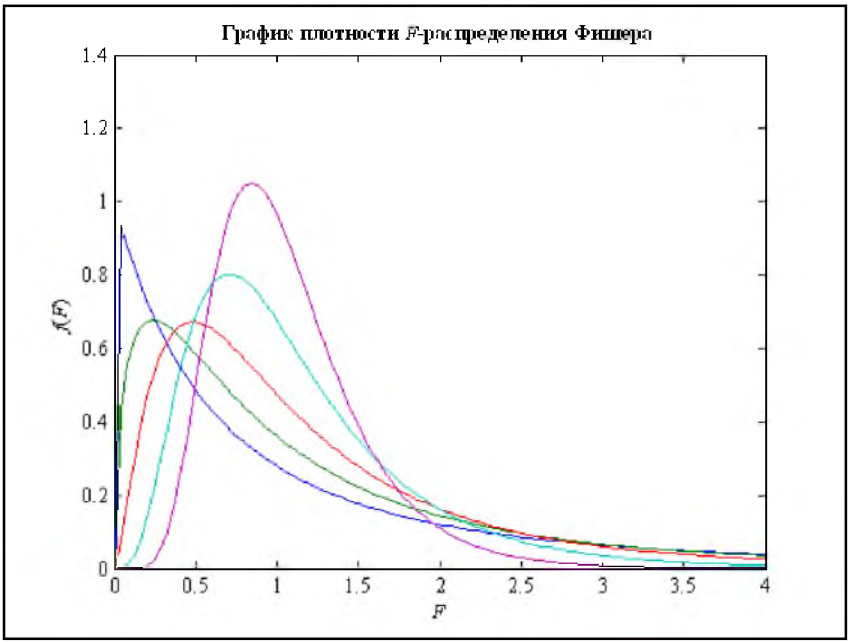


Рис. 22.2. Плотность F-распределения Фишера, выполненная с помощью MATLAB

Определение 22.2. Критерий проверки гипотезы о равенстве (или заданном отношении) двух генеральных дисперсий называется *F-критерием Фишера*. □

Если 0-гипотеза имеет место, то из (22.1—22.2) следует, что отношение выборочных дисперсий должно иметь F-распределение:

$$F = \frac{D_1^*}{D_2^*}. \tag{22.4}$$

Значит, реализация этой величины, т. е. вычисленное на практике отношение выборочных дисперсий, должно на уровне значимости q попадать в квантильные границы. Если альтернативой 0-гипотезе являются две гипотезы: $D_1 < D_2$ и $D_1 > D_2$, то доверительный интервал для 0-гипотезы будет 2-сторонним, как показано на рис. 19.4:

$$F_{\frac{q}{2}}(f_1, f_2) \le \frac{D_1^*}{D_2^*} \le F_{1-\frac{q}{2}}(f_1, f_2). \tag{22.5}$$

Если реальное отношение выборочных дисперсий попадает в доверительный интервал (22.5), то можно принять 0-гипотезу. Выход за левую границу ин-

тервала соответствует альтернативной гипотезе $D_1 < D_2$ (D_1^*/D_2^* мало), а за правую — гипотезе $D_1 > D_2$ (D_1^*/D_2^* велико).

Если заранее известно, что одна из альтернативных гипотез теоретически невозможна, то нужно применить односторонний критерий. Пусть, например, гипотеза $D_1 < D_2$ теоретически невозможна *(см. пример 19.7)*. В этом случае, если вдруг окажется, что D_1^*/D_2^* мало, то мы должны это считать случайностью и принять 0-гипотезу. Эта ситуация показана на рис. 19.5, а доверительный интервал для 0-гипотезы имеет вид:

$$\frac{D_1^*}{D_2^*} \le F_{1-q}(f_1, f_2). \tag{22.6}$$

Аналогично записывается доверительный интервал для проверки 0-гипотезы, если невозможна альтернативная гипотеза $D_1 > D_2$. Запишите его самостоятельно. Иллюстрация для этого случая — рис. 19.6.

Начнем выполнять ИДЗ по сравнению выборок. Пусть исходные данные для сравнения двух выборок хранятся в текстовом файле в виде двух столбцов одинаковой длины. Введем их. Найдем объем каждой выборки, число степеней свободы. Посчитаем средние и дисперсии.

```
clear all
sf='D:\Iglin\Matlab\ContData\comp2.txt';
x=load(sf); % вводим ИД - 2 столбца
n=size(x,1);
f=n-1;
fprintf('Объемы выборок: n=%d;\n',n);
fprintf('число степеней свободы каждой выборки f=%d.\n',f);
mx=mean(x); % средние столбцов
Dx=var(x); % дисперсии столбцов
fprintf('Математические ожидания: m1=%8.5f;  m2=%8.5f\n',mx);
fprintf('Дисперсии: D1=%8.5f;  D2=%8.5f\n',Dx);
Объемы выборок: n=100;
число степеней свободы каждой выборки f=99.
Математические ожидания: m1= 2.45819;  m2= 0.67622
Дисперсии: D1=31.16085;  D2=31.53027
```

Зададим уровень значимости. Посчитаем отношение D_1^*/D_2^* и квантили F-распределения Фишера. Проверим выполнение (22.5) и сделаем соответствующий вывод.

```
q=0.1; % уровень значимости
fprintf('Выбран уровень значимости q=%4.2f\n',q);
F=Dx(1)/Dx(2);
```

```
Fbord=finv([q/2 1-q/2],f,f); % границы доверительного интервала
fprintf('F=D1/D2=%12.8f\n',F);
fprintf(['Доверительный интервал для 0-гипотезы: '...
  '%12.8f<=D1/D2<=%12.8f.\n'],Fbord);
if F<Fbord(1),
  disp('Принимаем альтернативную гипотезу D1<D2.');
elseif F>Fbord(2),
  disp('Принимаем альтернативную гипотезу D1>D2.');
else
  disp('Принимаем 0-гипотезу D1=D2.');
end
Выбран уровень значимости q=0.10
F=D1/D2=  0.98828349
Доверительный интервал для 0-гипотезы:   0.71732859<=D1/D2<=  1.39406126.
Принимаем 0-гипотезу D1=D2.
```

22.2. Сравнение двух средних

Исходная информация та же, что и в предыдущем разделе. Имеются 2 генеральные совокупности: X_1 и X_2. Из каждой взята выборка объемом n_1 и n_2 соответственно. По формулам (18.14) вычислены выборочные математические ожидания m_1^* и m_2^*, а затем по (18.26) — выборочные дисперсии D_1^* и D_2^*. Но проверить теперь нужно 0-гипотезу о равенстве генеральных математических ожиданий:

$$m_1 = m_2. \tag{22.7}$$

Если генеральные дисперсии D_1 и D_2 известны, то эту задачу можно решить так. Рассмотрим случайные величины M_1^* и M_2^*, реализациями которых являются m_1^* и m_2^*. Они вычисляются по (18.15) и имеют параметры (18.16—18.17). Мы предполагаем, что генеральные совокупности нормальные, поэтому M_1^* и M_2^* также имеют нормальное распределение, т. к. они — суммы независимых нормальных величин, умноженные на константу. Введем теперь в рассмотрение случайную величину:

$$\Delta = M_1^* - M_2^*, \tag{22.8}$$

реализацией которой является разность между выборочными средними $\delta = m_1^* - m_2^*$. Распределение случайной величины Δ также нормальное (разность нормальных величин) с параметрами:

$$M(\Delta) = m_1 - m_2, \tag{22.9}$$

$$D(\Delta) = D\left(M_1^*\right) + D\left(M_2^*\right) = \frac{D_1}{n_1} + \frac{D_2}{n_2}. \tag{22.10}$$

Повторив выкладки (20.1)—(20.5), получим доверительный интервал для $m_1 - m_2$:

$$\left(m_1^* - m_2^*\right) - \sqrt{\frac{D_1}{n_1} + \frac{D_2}{n_2}}\, u_{1-\frac{q}{2}} \le m_1 - m_2 \le \left(m_1^* - m_2^*\right) + \sqrt{\frac{D_1}{n_1} + \frac{D_2}{n_2}}\, u_{1-\frac{q}{2}}. \quad (22.11)$$

Для проверки 0-гипотезы считаем, что (22.7) имеет место. Подставляем (22.7) в (22.11) и решаем (22.11) относительно $m_1^* - m_2^*$:

$$-\sqrt{\frac{D_1}{n_1} + \frac{D_2}{n_2}}\, u_{1-\frac{q}{2}} \le m_1^* - m_2^* \le \sqrt{\frac{D_1}{n_1} + \frac{D_2}{n_2}}\, u_{1-\frac{q}{2}}. \quad (22.12)$$

Если (22.12) выполняется, мы должны принять 0-гипотезу. Нарушение левого или правого неравенства требует отвергнуть 0-гипотезу и принять одну из альтернативных: $m_1 < m_2$ или $m_1 > m_2$ соответственно.

Оценка (22.12) годится, если заранее известны генеральные дисперсии D_1 и D_2. На практике это почти всегда не так. Обычно по выборкам мы можем найти только выборочные дисперсии D_1^* и D_2^*. Сравним их по F-критерию Фишера. Здесь возможны 2 варианта.

Вариант 1 — различие между D_1^* и D_2^* незначимо на заданном уровне значимости q. В этом случае можно считать, что D_1^* и D_2^* являются оценками одной и той же генеральной дисперсии D_x, более точную оценку которой D_x^* можно вычислить по методике текущих измерений (18.38). В этом случае величина Δ (22.8) имеет то же самое математическое ожидание (22.9), а дисперсия по (22.10) равна:

$$D(\Delta) = D_x\left(\frac{1}{n_1} + \frac{1}{n_2}\right). \quad (22.13)$$

Вместо (22.12) имеем:

$$-\sigma_x\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}\, u_{1-\frac{q}{2}} \le m_1^* - m_2^* \le \sigma_x\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}\, u_{1-\frac{q}{2}}. \quad (22.14)$$

По *определению 20.1* t-распределения Стьюдента, если мы в эту формулу вместо генерального среднеквадратичного отклонения σ_x подставим выборочное σ_x^*, то квантили стандартного нормального распределения нужно заменить квантилями t-распределения Стьюдента с числом степеней свободы $f = f_1 + f_2$:

$$-\sigma_x^*\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}\, t_{1-\frac{q}{2}}(f) \le m_1^* - m_2^* \le \sigma_x^*\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}\, t_{1-\frac{q}{2}}(f). \quad (22.15)$$

Получили доверительный интервал для 0-гипотезы (22.7), если генеральные дисперсии неизвестны, а выборочные сравнимы (т. е. мало отличаются). Примерные значения экспериментальных данных и соответствующие плотности распределения показаны на рис. 22.3. Слева — случай, когда 0-гипотеза справедлива, а справа — случай альтернативной гипотезы.

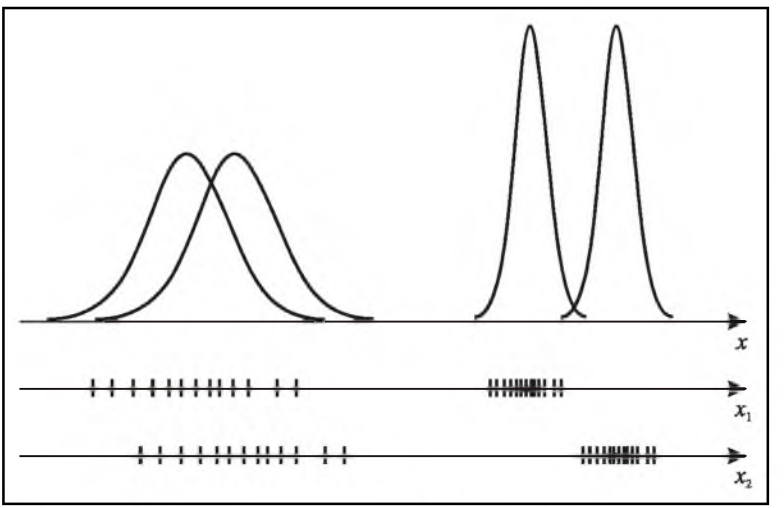


Рис. 22.3. Сравнение двух средних при одинаковых дисперсиях

Вариант 2 — различие между D_1^* и D_2^* признано значимым на уровне значимости q. В этом случае точных критериев сравнения нет. Рассмотрим без доказательства приближенный критерий, изложенный в [38]. Он состоит в следующем. Вначале вычисляем вспомогательные величины:

$$v_1 = \frac{D_1^*}{n_1}; \quad v_2 = \frac{D_2^*}{n_2}. \tag{22.16}$$

Далее находим величину τ:

$$\tau = \frac{v_1 t_{1-\frac{q}{2}}(f_1) + v_2 t_{1-\frac{q}{2}}(f_2)}{\sqrt{v_1 + v_2}}. \tag{22.17}$$

Сравнивая τ с левыми и правыми частями неравенств (22.12, 22.14—2.15), видим, что τ — это некое "средневзвешенное" из этих выражений. Сюда входят и обе выборочные дисперсии, и объемы выборок, и квантили t-распре-

деления Стьюдента для обеих степеней свободы, причем входят в нужном соотношении. Поэтому критерий проверки 0-гипотезы выглядит теперь так:

$$-\tau \le m_1^* - m_2^* \le \tau . \qquad (22.18)$$

Примерный разброс данных и графики плотностей распределения показаны на рис. 22.4: слева для случая 0-гипотезы, справа — для альтернативной.

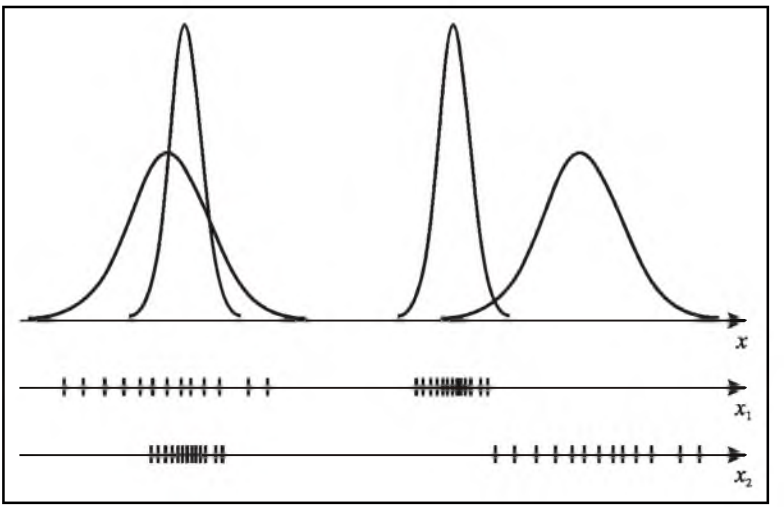


Рис. 22.4. Сравнение двух средних при различных дисперсиях

В MATLAB сравнивает два средних при любых выборочных дисперсиях функция `ttest2`. Сравним математические ожидания наших выборок.

```
[hm,pm,cm] = ttest2(x(:,1),x(:,2),q); % сравнение средних
fprintf(['Доверительный интервал: '...
  '%12.8f<=m1-m2<=%12.8f.\n'],cm);
if all(cm<0), % весь доверительный интервал <0
  disp('Принимаем альтернативную гипотезу m1<m2.');
elseif all(cm>0), % весь доверительный интервал >0
  disp('Принимаем альтернативную гипотезу m1>m2.');
else
  disp('Принимаем 0-гипотезу m1=m2.');
end
Доверительный интервал:   0.47349071<=m1-m2<=  3.09045040.
Принимаем альтернативную гипотезу m1>m2.
```

22.3. Сравнение нескольких дисперсий

Постановка задачи такая. Есть k генеральных совокупностей: $X_1, X_2, ..., X_k$. Из каждой взята выборка объемом $n_1, n_2, ..., n_k$ соответственно. По этим выборкам с помощью формул (18.14) вычислены выборочные математические ожидания $m_1^*, m_2^*, ..., m_k^*$, а затем по (18.26) — выборочные дисперсии $D_1^*, D_2^*, ..., D_k^*$. Выдвинута 0-гипотеза о равенстве всех генеральных дисперсий:

$$D_1 = D_2 = ... = D_k, \tag{22.19}$$

и требуется построить критерий ее проверки.

Заметьте: мы здесь не задаемся вопросом, какая дисперсия больше, а какая меньше и насколько. Мы решаем более простую задачу: выясняем, одинаковые дисперсии или разные. То есть альтернативная гипотеза здесь одна — генеральные дисперсии разные. У нас уже есть некоторый опыт решения таких задач: нужно построить функцию, которая характеризовала бы меру различия выборочных дисперсий D_i^*. Если эта мера окажется большой, то 0-гипотезу нужно отвергнуть, а если малой — принять. Критерий проверки, очевидно, будет односторонним, как показано на рис. 19.5.

Первое, что приходит на ум,— это сравнить по F-критерию Фишера максимальную и минимальную выборочные дисперсии из нашего набора. Если различие между ними выявится незначимым, то и их, и все промежуточные можно считать оценками одной и той же генеральной дисперсии, т. е. принять 0-гипотезу. Но если крайние дисперсии отличаются значительно, то вопрос остается открытым. Ведь если мы рассортируем D_i^* в порядке возрастания и начнем сравнивать их попарно, то может оказаться, что соседние дисперсии отличаются незначительно, хотя крайние различаются значимо. Поэтому нужны такие критерии, которые учитывают все D_i^* одновременно. И такие критерии есть. Мы рассмотрим без доказательства два из них.

22.3.1. Критерий Бартлета

Его проще всего описать по шагам, как рецепт. Вот он.

1. Вычисляем средневзвешенную выборочную дисперсию и число ее степеней свободы:

$$D_x^* = \frac{1}{f}\sum_{j=1}^{k} f_j D_j^*; \tag{22.20}$$

$$f = \sum_{j=1}^{k} f_j. \tag{22.21}$$

2. Вычисляем две вспомогательные величины:

$$B = f \ln D_x^* - \sum_{j=1}^{k} f_j \ln D_j^*; \tag{22.22}$$

$$C = 1 + \frac{1}{3(k-1)}\left(\sum_{j=1}^{k}\frac{1}{f_j} - \frac{1}{f}\right). \tag{22.23}$$

3. Находим их отношение:

$$\chi^2 = \frac{B}{C}. \tag{22.24}$$

Бартлет (Bartlett) доказал, что в случае справедливости 0-гипотезы (22.19) полученное отношение имеет приближенно χ^2-распределение Пирсона с $k-1$ степенями свободы, если все $f_j \geq 5$. Поэтому критерием проверки 0-гипотезы является не очень большая величина B/C: именно она в критерии Бартлета и характеризует меру различия D_i^*. Если выполняется условие:

$$\frac{B}{C} \leq \chi^2_{1-q}(k-1), \tag{22.25}$$

то на уровне значимости q можно принять 0-гипотезу. Нарушение (22.25) требует отвергнуть ее и принять единственную альтернативную гипотезу: генеральные дисперсии разные.

Проверка по критерию Бартлета реализована в MATLAB в виде функции `barttest`. Продолжим выполнение ИДЗ по сравнению выборок. Введем из файла данные по нескольким выборкам. Найдем количество выборок, их объемы, числа степеней свободы. Посчитаем средние и дисперсии. Зададим уровень значимости. Проверим 0-гипотезу (22.19) по критерию Бартлета и сделаем вывод.

```
sf='D:\Iglin\Matlab\ContData\compk.txt';
x=load(sf); % вводим ИД - k столбцов
[n,k]=size(x); % объем каждой выборки и их количество
f=n-1;
fprintf('Число выборок: k=%d;\nобъемы выборок: n=%d;\n',k,n);
fprintf('число степеней свободы каждой выборки f=%d.\n',f);
mx=mean(x); % средние столбцов
Dx=var(x); % дисперсии столбцов
disp('Математические ожидания:')
fprintf('mx(%d)=%8.5f\n',[1:k;mx]);
disp('Дисперсии:')
fprintf('Dx(%d)=%8.5f\n',[1:k;Dx]);
```

```
q=0.1; % уровень значимости
fprintf('Выбрали уровень значимости q=%5.3f\n',q);
ndim=barttest(x,q); % тест по критерию Бартлета
if isnan(ndim) | (ndim==1), % делаем вывод
  disp('Принимаем 0-гипотезу.')
else
  disp('Отвергаем 0-гипотезу.')
end
Число выборок: k=6;
объемы выборок: n=100;
число степеней свободы каждой выборки f=99.
Математические ожидания:
mx(1)= 1.29542
mx(2)= 1.89533
mx(3)= 2.03699
mx(4)= 1.34814
mx(5)= 2.56326
mx(6)= 2.12714
Дисперсии:
Dx(1)=28.42349
Dx(2)=25.46456
Dx(3)=24.38347
Dx(4)=35.82064
Dx(5)=48.25061
Dx(6)=26.16001
Выбрали уровень значимости q=0.100
Отвергаем 0-гипотезу.
```

22.3.2. Критерий Кохрана

Этот критерий более точный, чем критерий Бартлета, но область его применения более узкая. Он может быть применен только для выборок одинакового объема. Для применения этого критерия вычисляется величина g, которая является отношением максимальной выборочной дисперсии к сумме всех остальных:

$$g = \frac{D^*_{\max}}{\sum_{j=1}^{k} D^*_j - D^*_{\max}}. \qquad (22.26)$$

Кохран (Cochran) вывел распределение случайной величины G, реализацией которой является (22.26) в предположении справедливости 0-гипотезы. Оно так и называется: G-распределение Кохрана. Это распределение зависит от

двух параметров: количества выборок k и числа степеней свободы каждой выборки f. На уровне значимости q 0-гипотезу можно принять, если выполняется неравенство:

$$g \le g_{1-q}, \tag{22.27}$$

где g_{1-q} — квантиль G-распределения Кохрана.

В стандартном MATLAB и его расширении [57] нет проверки по критерию Кохрана, но пользователи MATLAB разработали соответствующие функции и разместили их на сайте [51] на странице "Теория вероятностей и статистика". Их можно свободно переписать оттуда и осуществить по ним проверку данных.

22.4. Сравнение нескольких средних

Имеется k генеральных совокупностей: X_1, X_2, ..., X_k; из них взяты выборки объемом n_1, n_2, ..., n_k соответственно. Далее, как обычно, по (18.14) вычислены выборочные математические ожидания m_1^*, m_2^*, ..., m_k^*, а затем по (18.26) — выборочные дисперсии D_1^*, D_2^*, ..., D_k^*. Проверяется 0-гипотеза о равенстве всех генеральных математических ожиданий:

$$m_1 = m_2 = \ldots = m_k. \tag{22.28}$$

Мы ограничимся только одним случаем: когда все выборочные дисперсии D_j^* сравнимы, т. е. являются оценками одной и той же генеральной дисперсии. На практике чаще всего встречается именно такая ситуация. Например, на одном и том же экспериментальном стенде исследуются несколько партий изделий, и требуется подтвердить или опровергнуть вывод об идентичности партий. Если же дисперсии различаются значимо, то этот критерий можно рассматривать как приближенный.

Как и в предыдущем *разделе 22.3*, проще всего сравнить крайние (максимальное и минимальное) средние по t-критерию Стьюдента. Если они различаются незначимо, то все средние тем более различаются незначительно, и 0-гипотезу можно принять. Но если максимальное и минимальное математические ожидания различаются значимо, то вопрос остается открытым, и в этом случае нужно сравнивать все m_j^* одновременно. Значит, нам нужно построить критерий — меру различия средних, — по величине которого можно судить об истинности или ложности 0-гипотезы.

На рис. 22.5 слева показаны несколько выборок, для которых справедлива 0-гипотеза, а справа — выборки с большим разбросом средних. Следовательно, именно эта величина — разброс средних — может служить критерием проверки 0-гипотезы. Если разброс средних не очень большой по сравнению с разбросом каждой выборки вокруг ее среднего, то 0-гипотезу можно при-

нять. Так, на рис. 22.5 слева расстояние между пиками плотностей распределения (размах средних) примерно такое же, как размах каждой выборки. Поэтому мы и говорим, что здесь, скорее всего, 0-гипотеза имеет место. Справа же размах средних значительно превышает размах каждой выборки, и здесь, по-видимому, 0-гипотезу придется отвергнуть.

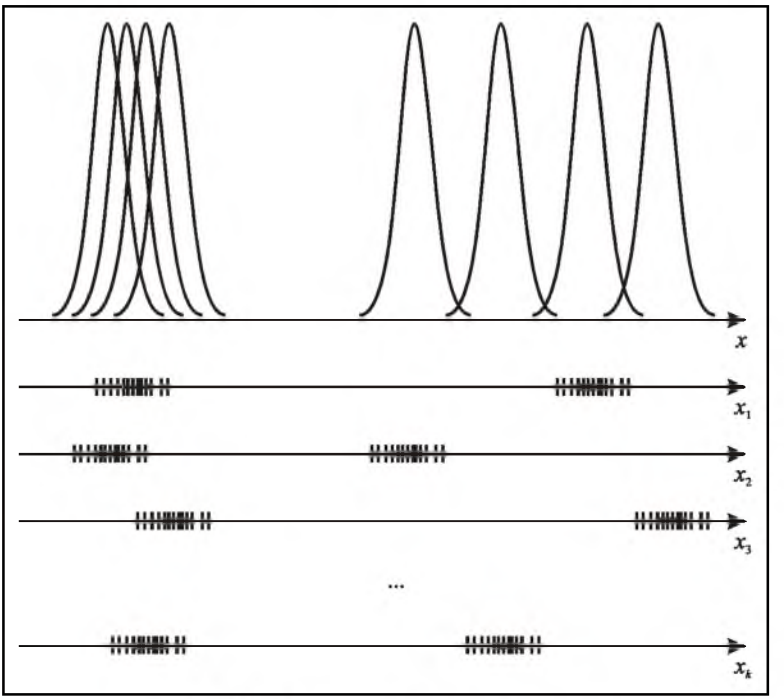


Рис. 22.5. Сравнение нескольких средних

Переведем эти слова в числа. У нас есть F-критерий сравнения двух дисперсий, поэтому разброс данных мы будем характеризовать не размахом, а дисперсией. Дисперсия каждой выборки D_j^* у нас есть. Поскольку D_j^* сравнимы, то более точной оценкой меры разброса каждой выборки вокруг ее среднего является средневзвешенная дисперсия:

$$D_x^* = \frac{1}{f}\sum_{j=1}^{k} f_j D_j^*, \tag{22.29}$$

где

$$f = \sum_{j=1}^{k} f_j \tag{22.30}$$

есть общее число степеней свободы всех выборок.

Повышение точности достигается здесь именно за счет увеличения числа степеней свободы (методика текущих измерений).

Полученная величина D_x^* характеризует разброс каждой выборки вокруг ее среднего. Мы должны сравнить ее с разбросом средних. Причем, т. к. мы будем сравнивать по F-критерию Фишера, разброс средних также нужно охарактеризовать их дисперсией. Поэтому вычислим среднее средних:

$$\tilde{m}_x = \frac{1}{k}\sum_{j=1}^{k} m_j^*, \tag{22.31}$$

а потом дисперсию средних:

$$D_m^* = \frac{1}{k-1}\sum_{j=1}^{k}\left(m_j^* - \tilde{m}_x\right)^2. \tag{22.32}$$

Величину D_m^* можно рассматривать как выборочную дисперсию с $k-1$ степенями свободы. Она как раз и характеризует разброс средних. Если эта величина не очень большая по сравнению с D_x^*, то можно принять 0-гипотезу. Сравнение двух дисперсий осуществляется по F-критерию Фишера, поэтому можно считать, что 0-гипотеза имеет место, если на уровне значимости q выполняется неравенство:

$$\frac{D_m^*}{D_x^*} \le F_{1-q}\left(k-1, f\right). \tag{22.33}$$

Нарушение (22.33) заставляет нас отвергнуть 0-гипотезу и принять альтернативную: m_i — разные. Мы применяем здесь 1-сторонний критерий (см. рис. 19.5), т. к. слишком малое значение D_m^* по сравнению с D_x^* свидетельствует о еще большей справедливости 0-гипотезы.

Применим этот критерий для проверки наших данных. Запрограммируем формулы (22.29—22.32). Проведем проверку по критерию (22.33) и сделаем соответствующий вывод. Нарисуем на одном графике гистограммы всех выборок. Они показаны на рис. 22.6 наложенными одна на другую. Полученная картинка напоминает рис. 22.5, на котором изображены теоретические плотности распределения. Ведь гистограмма — это и есть выборочная плотность распределения (с точностью до масштаба)! На другом графике (рис. 22.7) изобразим выборочные функции распределения.

```
ft=f*k % общее число степеней свободы
Dxt=mean(Dx) % средневзвешенная дисперсия
Dm=var(mx) % разброс средних
if Dm/Dxt<=finv(1-q,k-1,ft),
  disp('Принимаем 0-гипотезу.')
```

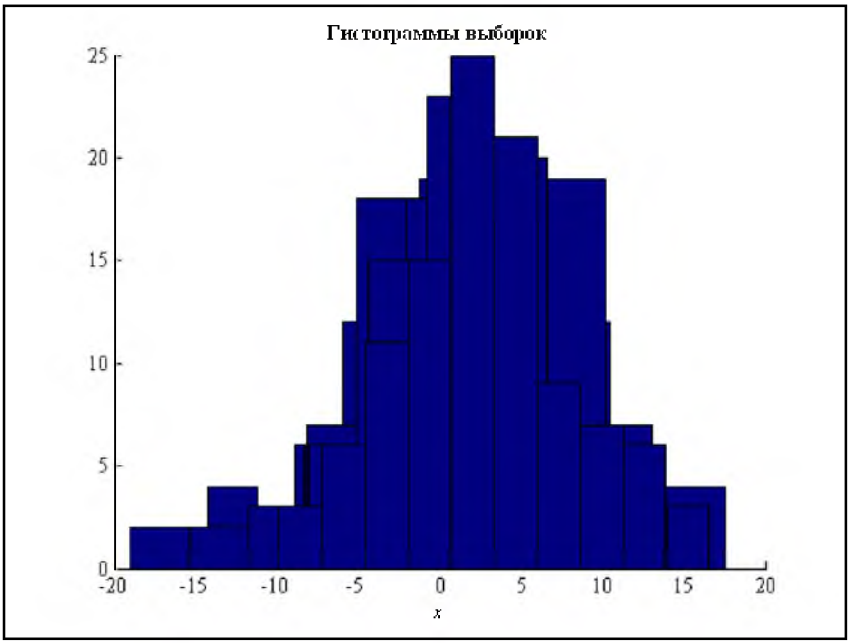


Рис. 22.6. Гистограммы нескольких выборок, выполненные с помощью MATLAB

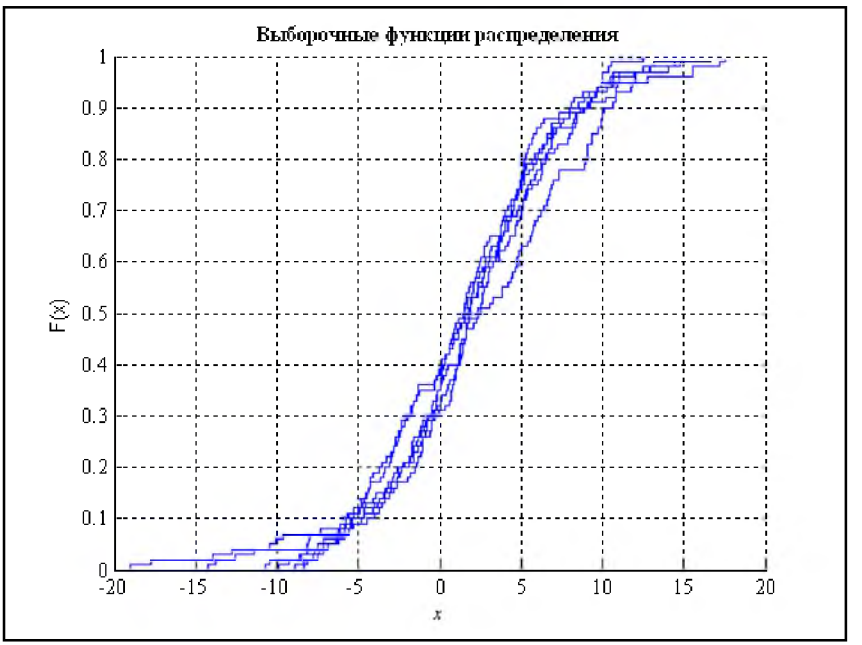


Рис. 22.7. Выборочные функции распределения, выполненные с помощью MATLAB

```
else
  disp('Отвергаем 0-гипотезу.')
end
figure % новая фигура
hold on % задержка для рисования на одной фигуре
for j=1:k,
  hist(x(:,j),round(n^0.5)); % гистограммы
end
hold off
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfГистограммы выборок')
xlabel('\itx') % метка оси OX
figure % новая фигура
hold on % задержка для рисования на одной фигуре
for j=1:k,
  cdfplot(x(:,j)); % функции распределения
end
hold off
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfВыборочные функции распределения')
xlabel('\itx') % метка оси OX
ft =
   594
Dxt =
  31.41713235381618
Dm =
   0.23547793276937
Принимаем 0-гипотезу.
```

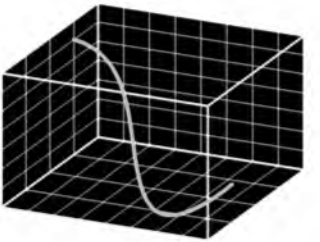
В нашем варианте D_m^* оказалась значительно меньше D_x^*, поэтому 0-гипотеза справедлива. Рисунки подтверждают этот вывод: гистограммы и функции распределения почти накладываются одна на другую.

22.5. Вопросы для самопроверки

1. Почему доверительный интервал (22.5) должен обязательно включать число 1?
2. Сформулируйте F-критерий Фишера для проверки 0-гипотезы $D_2 = 2D_1$.
3. Запишите выражение для доверительного интервала 0-гипотезы $D_1 = D_2$, если возможна только одна альтернативная гипотеза: $D_1 < D_2$, а $D_1 > D_2$ — невозможна.

4. Можно ли применять F-критерий Фишера, если генеральные совокупности не нормальные? Где в выводе F-распределения Фишера используется гипотеза о нормальном распределении генеральных совокупностей?
5. Выведите формулу (22.11), используя (20.1—20.5).
6. Какие альтернативные гипотезы справедливы для наборов данных, представленных на рис. 22.1 и 22.2 справа?
7. Найдите в литературе или Интернете доказательство 2-го варианта критерия проверки 0-гипотезы о равенстве 2-х генеральных средних.
8. Какой алгоритм реализован в функции `ttest2`: (22.15) или (22.16—22.18)?
9. Найдите в литературе или Интернете доказательство критерия Бартлета и вывод распределения Кохрана.
10. Перепишите с [51] функции, реализующие критерий Кохрана, и проверьте свой вариант данных. Совпадает ли результат с проверкой по критерию Бартлета?
11. Найдите в [51] готовые функции для сравнения нескольких средних.

ГЛАВА 23



Дисперсионный анализ

В предыдущих главах мы полагали, что все условия опытов остаются неизменными, а разброс экспериментальных данных вызывается лишь случайными причинами. Другая, не менее важная задача математической статистики, заключается в том, что во время эксперимента один или несколько подконтрольных нам факторов изменяются, и необходимо оценить влияние этих факторов. В более общей постановке этой задачи необходимо отделить изучаемые факторы от случайных, друг от друга и оценить, значимо ли влияет каждый из них на результат эксперимента по сравнению с неконтролируемыми (случайными) факторами.

В настоящей главе не решается вопрос о том, *как* зависит результат эксперимента от конкретного фактора. Мы будем выяснять, *зависит ли он вообще* от этого фактора, или изучаемый фактор лишь незначимо изменяет результат по сравнению со случайными факторами. Нахождение конкретного вида зависимости результата от факторов будет подробно рассмотрено в следующей *главе 24*. А в *главе 25* мы учтем случайность не только опытных данных, но и факторов, от которых они зависят.

Будем считать, что случайные ошибки наблюдений имеют нормальное распределение, а изучаемые факторы (назовем их A, B и т. д.):

- не зависят от случайных;
- не зависят друг от друга;
- влияют только на среднее результатов и не влияют на их дисперсию.

Последнее свойство показано на рис. 23.1: изменение некоторого фактора A может вызвать лишь смещение результатов, но не изменяет их разброса.

Чем же мы будем характеризовать меру влияния фактора A? Рассмотрим сначала случай абсолютно точных измерений, без погрешностей. Пусть при n

значениях фактора A (говорят: на n уровнях фактора A) получены истинные результаты x_1, x_2, ..., x_n. Если A не влияет на них, то все они одинаковые (нет случайностей). Но если A влияет, и нас интересует только степень этого влияния, то в качестве меры влияния удобно взять степень разброса результатов — выборочную дисперсию:

$$m_x^* = \frac{1}{n}\sum_{i=1}^{n} x_i; \tag{23.1}$$

$$D_A = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - m_x^*\right)^2. \tag{23.2}$$

Здесь мы обозначили выборочную дисперсию D_A, т. к. она характеризует влияние только этого фактора. Обратите внимание: все результаты x_i здесь истинные, никаких случайностей нет. Поэтому D_A — лишь аналог выборочной дисперсии. Его величина характеризует влияние не случайностей, а только лишь фактора A. Но эта оценка является очень удобной по следующим причинам:

- дисперсия является простейшей мерой рассеяния и легко вычисляется;
- показатель влияния фактора A определяется аналогично показателю влияния случайных факторов, что позволяет их сравнивать, как мы научились в *главе 22*.

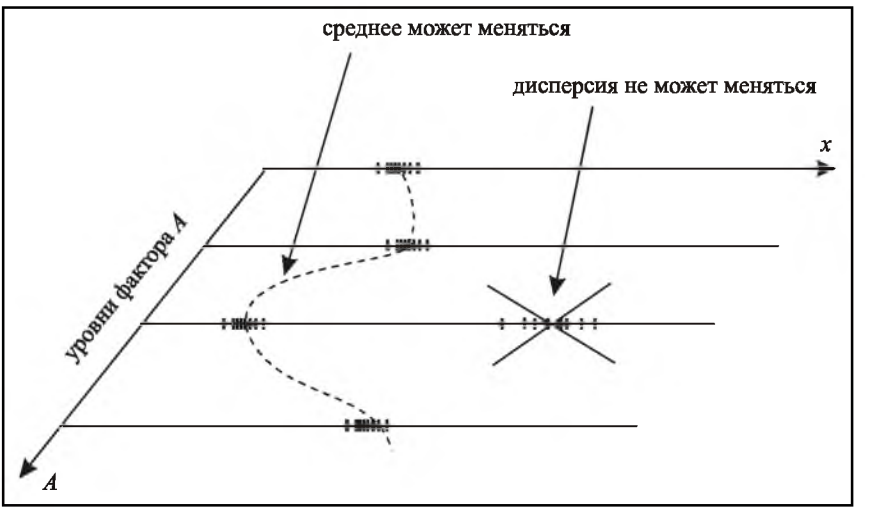


Рис. 23.1. Изменение фактора A может вызвать смещение результатов, но не изменяет разброса

Определение 23.1. Изучение переменных факторов по их дисперсиям называется *дисперсионным анализом*. □

При решении задач дисперсионного анализа мы полагаем действие факторов и случайностей независимыми случайными событиями. Совместное их действие приводит к тому, что экспериментальные результаты будут реализациями случайной величины, дисперсия которой равна сумме дисперсий факторов и случайностей. Например, если генеральная дисперсия X при неизменном A равна D_0, дисперсия фактора A при отсутствии случайностей равна D_A и других факторов нет, то генеральная дисперсия X, вычисленной при различных уровнях фактора A, будет равна:

$$D_x = D_A + D_0. \tag{23.3}$$

Такой подход позволяет сравнивать различные дисперсии и оценивать влияние тех или иных факторов. Рассмотрим простейший случай.

ПРИМЕР 23.1. Пусть D_0 известна, и исследуется влияние фактора A. На n уровнях фактора A получены экспериментальные значения $x_1, x_2, ..., x_n$. Спрашивается: является ли влияние фактора A значимым на уровне значимости q?

Найдем выборочное математическое ожидание по (23.1), а затем — дисперсию выборки:

$$D_x^* = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - m_x^*\right)^2. \tag{23.4}$$

Она является оценкой генеральной дисперсии D_x из формулы (23.3), т. к. в ней учитывается влияние и случайностей, и фактора A. Выдвинем 0-гипотезу: фактор A влияет незначимо. В этом случае D_x^* не должна быть слишком большой по сравнению с D_0. Две дисперсии сравниваются по F-критерию Фишера. 0-гипотезу можно принять на уровне значимости q, если выполняется неравенство:

$$\frac{D_x^*}{D_0} \le F_{1-q}\left(k-1,\infty\right), \tag{23.5}$$

где F_{1-q} — квантиль F-распределения Фишера, $k-1$ — число степеней свободы D_x^*, а у генеральной дисперсии D_0 бесконечное число степеней свободы. Если (23.5) нарушается, мы должны отвергнуть 0-гипотезу и принять альтернативную: фактор A влияет значимо. В этом случае можно вычислить его выборочную дисперсию:

$$D_A^* = D_x^* - D_0, \tag{23.6}$$

которая является оценкой генеральной дисперсии D_A из (23.3). □

23.1. 1-факторный дисперсионный анализ

В примере 23.1 генеральная дисперсия случайностей D_0 была известна. Поэтому там не нужно было проводить параллельных измерений на одном и том же уровне фактора A. Но на практике D_0 обычно неизвестна, и ее приходится оценивать по самой выборке. Поэтому теперь нам нужно на каждом уровне фактора A проводить несколько измерений. Эта задача похожа на задачу сравнения нескольких средних *(см. раздел 22.4)*, но ее решение рассматривается с несколько иных позиций.

Будем обозначать уровни фактора A через A_1, A_2, ..., A_j, ..., A_k (всего k уровней). Для простоты предположим, что на каждом уровне A_j проведено по одинаковому количеству опытов n. Это ограничение — не принципиальное, просто без него все формулы получатся более громоздкими. Обозначим измерения, проведенные на уровне A_j: x_{j1}, x_{j2}, ..., x_{ji}, ..., x_{jn}. Первый индекс здесь — номер уровня фактора, второй — номер опыта. Таким образом, получаем матрицу опыта размером $k \times n$:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{k1} & x_{k2} & \dots & x_{kn} \end{pmatrix}, \tag{23.7}$$

каждая строка которой соответствует одному уровню фактора A. Отсюда уже видно, как отсеять влияние A и вычислить дисперсию, которая учитывает влияние *только* случайностей. Для этого достаточно вычислить дисперсии всех строк, которые должны быть примерно одинаковые (фактор A не влияет на разброс результатов!):

$$m_j^* = \frac{1}{n}\sum_{i=1}^{n} x_{ji}; \tag{23.8}$$

$$D_j^* = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_{ji} - m_j^*\right)^2. \tag{23.9}$$

Пусть проверка по критериям Бартлета или Кохрана показала, что наши гипотезы справедливы и выборочные дисперсии D_j^* действительно сравнимы. В этом случае по методике текущих измерений мы можем вычислить средневзвешенную всех D_j^*, которую и можно считать оценкой дисперсии случайностей D_0:

$$D_0^* = \frac{1}{k}\sum_{j=1}^{k} D_j^*. \tag{23.10}$$

Число ее степеней свободы равно общему числу степеней свободы всех выборок:

$$f = k(n-1). \tag{23.11}$$

Итак, мы нашли знаменатель для формулы (23.5) — выборочную дисперсию, которая оценивает влияние только случайностей. Перейдем теперь к числителю — дисперсии, которая учитывает влияние и случайностей, и фактора A. Рассмотрим 2 способа решения этой задачи.

Способ 1. Проще всего собрать все nk опытов в одну выборку, найти ее среднее и дисперсию:

$$m_x^* = \frac{1}{kn}\sum_{j=1}^{k}\sum_{i=1}^{n} x_{ji}; \tag{23.12}$$

$$D_x^* = \frac{1}{kn-1}\sum_{j=1}^{k}\sum_{i=1}^{n}\left(x_{ji} - m_x^*\right)^2. \tag{23.13}$$

Полученная дисперсия имеет $kn-1$ степеней свободы. Она характеризует влияние и случайностей, и фактора A. Фактор A считаем незначимым на уровне значимости q, если выполняется неравенство:

$$\frac{D_x^*}{D_0^*} \le F_{1-q}\left(kn-1,\, k\left(n-1\right)\right). \tag{23.14}$$

При нарушении (23.14) нужно считать, что фактор A влияет значимо. В этом случае можно найти D_A^* по (23.6).

Способ 2. Можно оценивать разброс не всех данных x_{ji}, а только средних m_j^*. Этот способ точнее первого, т. к. дисперсия среднего в n раз меньше дисперсии каждого измерения (18.17). Поэтому, если мы вычислим дисперсию средних:

$$D_x^{**} = \frac{1}{k-1}\sum_{j=1}^{k}\left(m_j^* - m_x^*\right)^2, \tag{23.15}$$

то она будет включать в себя не D_0^*, а D_0^*/n. И вместо (23.3) в данном случае будем иметь:

$$D_x^{**} = D_A^* + \frac{D_0^*}{n}. \tag{23.16}$$

Умножив это выражение на n, видим, что теперь мы имеем в n раз больше шансов выловить влияние A, если оно вообще есть:

$$nD_x^{**} = nD_A^* + D_0^*. \tag{23.17}$$

Критерий проверки 0-гипотезы такой:

$$\frac{nD_x^{**}}{D_0^*} \le F_{1-q}\left(k-1, k\left(n-1\right)\right). \qquad (23.18)$$

Если это неравенство выполняется, то фактор A влияет незначимо, а если нет — то значимо.

В MATLAB задачу 1-факторного дисперсионного анализа решает функция `anova1`. Рассмотрим задание с теми же самыми k выборками, которое мы использовали в *главе 22* при сравнении нескольких выборок. Введем данные. Найдем объем каждой выборки, общее их количество, число степеней свободы.

```
clear all
sf='D:\Iglin\Matlab\ContData\compk.txt';
x=load(sf); % вводим ИД - k столбцов
[n,k]=size(x); % объем каждой выборки и их количество
f=n-1;
fprintf('Число выборок: k=%d;\nобъемы выборок: n=%d;\n',k,n);
fprintf('число степеней свободы каждой выборки f=%d.\n',f);
Число выборок: k=6;
объемы выборок: n=100;
число степеней свободы каждой выборки f=99.
```

Выберем уровень значимости и проведем 1-факторный дисперсионный анализ. Функция `anova1` возвращает критический уровень значимости. Если он выше заданного q, то 0-гипотезу можно принять, а если ниже — то ее нужно отвергнуть. По умолчанию функция `anova1` возвращает также таблицу статистических характеристик (рис. 23.2) и график данных. На графике (рис. 23.3) показаны границы данных по каждому столбцу, их средние, а также 25% и 75%-ные выборочные квантили.

```
q=0.1; % уровень значимости
fprintf('Выбран уровень значимости q=%4.2f\n',q);
p=anova1(x);
if p>=q,
  disp('Фактор A влияет незначимо.')
else
  disp('Фактор A влияет значимо.')
end
Выбран уровень значимости q=0.10
Фактор A влияет незначимо.
```

ANOVA Table

Source	SS	df	MS	F	Prob>F
Columns	117.7	5	23.5478	0.75	0.5866
Error	18661.8	594	31.4171		
Total	18779.5	599			

Рис. 23.2. Таблица данных, возвращаемая функцией `anoval`

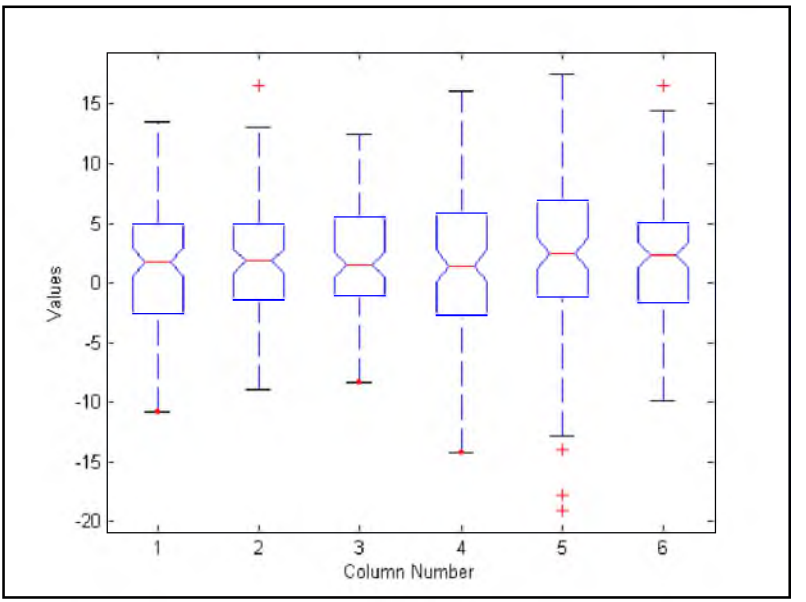


Рис. 23.3. Границы данных по столбцам

23.2. 2-факторный дисперсионный анализ

Дисперсионный анализ особенно удобен при изучении совместного действия нескольких факторов. Пусть изучаются 2 фактора: A и B. Уровни фактора A: A_1, A_2, ..., A_j, ..., A_k (всего k уровней). Уровни фактора B: B_1, B_2, ..., B_i, ..., B_m (всего m уровней). Пусть отсутствуют параллельные измерения, т. е. при каждом сочетании факторов A_jB_i проведено только 1 измерение x_{ji}. Матрица экспериментов имеет вид:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots \\ x_{k1} & x_{k2} & \dots & x_{km} \end{pmatrix}. \qquad (23.19)$$

Каждая j-я строка соответствует j-му уровню фактора A, каждый i-й столбец — i-му уровню B. Несмотря на то, что отсутствуют параллельные измерения, этих данных вполне достаточно для оценки и D_0^*, и D_A^*, и D_B^*. Нужно только правильно скомбинировать те два способа, которые мы применяли в 1-факторном дисперсионном анализе. По способу 2 найдем средние каждой строки:

$$m_j^{R*} = \frac{1}{m}\sum_{i=1}^{m} x_{ji}, \tag{23.20}$$

а затем их разброс вокруг общего среднего:

$$D_{A0}^* = \frac{1}{k-1}\sum_{j=1}^{k}\left(m_j^{R*} - m_x^*\right)^2, \tag{23.21}$$

где m_x^* — среднее всей матрицы (23.19), которое является одновременно и средним строк, и средним столбцов:

$$m_x^* = \frac{1}{km}\sum_{j=1}^{k}\sum_{i=1}^{m} x_{ji}. \tag{23.22}$$

В каждом из средних строки (23.20) произведено усреднение по фактору B, поэтому в их дисперсии (23.21) сказывается влияние только фактора A и случайностей. При этом дисперсия случайностей уменьшена в m раз, т. к. вместо конкретного измерения мы берем среднее из m измерений:

$$D_{A0}^* = D_A^* + \frac{D_0^*}{m}. \tag{23.23}$$

Теперь "транспонируем" выкладки (23.20—23.23). Найдем средние каждого столбца:

$$m_i^{C*} = \frac{1}{k}\sum_{j=1}^{k} x_{ji}, \tag{23.24}$$

а затем их разброс вокруг общего среднего (23.22):

$$D_{B0}^* = \frac{1}{m-1}\sum_{i=1}^{m}\left(m_i^{C*} - m_x^*\right)^2. \tag{23.25}$$

Эта дисперсия учитывает влияние фактора B и случайностей, причем дисперсия случайностей здесь в k раз меньше, чем в том случае, если бы мы вместо средних брали отдельные измерения:

$$D_{B0}^* = D_B^* + \frac{D_0^*}{k}. \tag{23.26}$$

Это мы воспользовались способом 2 из предыдущего раздела. Теперь проведем выкладки, аналогичные способу 1. Если собрать все km опытов в одну выборку, то ее дисперсия будет учитывать влияние и A, и B, и случайностей. Это будет 3-е уравнение, которое замкнет систему (23.23), (23.26) и позволит найти все 3 дисперсии. Но точность этого метода низка. Поэтому поступим так. На данном уровне A_j фактора A разброс элементов j-й строки матрицы (23.19) вокруг их среднего (23.20) характеризует влияние фактора B и случайностей, причем в одинаковой степени:

$$D_j^{R*} = \frac{1}{m-1}\sum_{i=1}^{m}\left(x_{ji} - m_j^{R*}\right)^2. \tag{23.27}$$

Различные уровни фактора A не влияют на дисперсию. Значит, все D_j^{R*} равнимы (примерно одинаковы), и более точной оценкой влияния фактора B и случайностей является средневзвешенное всех D_j^{R*}:

$$D_x^{R*} = \frac{1}{k}\sum_{j=1}^{k} D_j^{R*} = D_B^* + D_0^*. \tag{23.28}$$

Аналогично можно найти разброс элементов каждого столбца вокруг его среднего, который учитывает влияние только A и случайностей:

$$D_i^{C*} = \frac{1}{k-1}\sum_{j=1}^{k}\left(x_{ji} - m_i^{C*}\right)^2, \tag{23.29}$$

а затем взять средневзвешенное полученных величин, которое также будет учитывать влияние только фактора A и случайностей:

$$D_x^{C*} = \frac{1}{m}\sum_{i=1}^{m} D_i^{C*} = D_A^* + D_0^*. \tag{23.30}$$

Теперь из решения какой-либо пары уравнений: (23.23) + (23.30) или (23.26) + (23.28) можно найти D_0^*. Возьмем, например, пару (23.26) + (23.28):

$$\begin{cases} D_{B0}^* = D_B^* + \dfrac{D_0^*}{k}; \\ D_x^{R*} = D_B^* + D_0^*. \end{cases} \tag{23.31}$$

Вычтем из 2-го уравнения 1-е и подставим (23.27) и (23.25):

$$\frac{k-1}{k} D_0^* = \frac{1}{k(m-1)}\sum_{j=1}^{k}\sum_{i=1}^{m}\left(x_{ji} - m_j^{R*}\right)^2 - \frac{1}{m-1}\sum_{i=1}^{m}\left(m_i^{C*} - m_x^*\right)^2. \tag{23.32}$$

Отсюда выборочная дисперсия случайностей:

$$D_0^* = \frac{1}{(m-1)(k-1)}\left(\sum_{j=1}^{k}\sum_{i=1}^{m}\left(x_{ji} - m_j^{R*}\right)^2 - k\sum_{i=1}^{m}\left(m_i^{C*} - m_x^*\right)^2\right). \qquad (23.33)$$

Можно показать, что решение системы (23.23) + (23.30) дает такой же результат.

Как видим, дисперсия (23.33) имеет $(m-1)(k-1)$ степеней свободы. Ее можно применить для оценки влияния факторов A и B. При этом для повышения точности нужно брать для сравнения с D_0^* величины (23.23) и (23.26), т. к. в них влияние случайностей меньше. На уровне значимости q влиянием фактора A можно пренебречь, если

$$\frac{mD_{A0}^*}{D_0^*} \le F_{1-q}\left(k-1,\,(m-1)(k-1)\right). \qquad (23.34)$$

Нарушение этого неравенства свидетельствует о значимости фактора A. Аналогично записывается критерий проверки значимости фактора B.

Задачу 2-факторного дисперсионного анализа в MATLAB можно решить с помощью функции `anova2`. Входными данными являются матрица X вида (23.19) и различные параметры настройки. В результате получаем критические значения p, которые сравниваем с заданным уровнем значимости q. Превышение p над q свидетельствует о незначимом влиянии фактора, а выполнение условия $p < q$ — о справедливости альтернативной гипотезы.

Выполним ИДЗ по 2-факторному дисперсионному анализу. Введем матрицу исходных данных и определим ее размеры.

```
sf='D:\Iglin\Matlab\ContData\compkm.txt';
x=load(sf); % вводим ИД
[k,m]=size(x);
fprintf('Размеры матрицы: k=%d; m=%d.\n',k,m);
```

```
Размеры матрицы: k=14; m=13.
```

Зададим уровень значимости. Решим задачу 2-факторного дисперсионного анализа и проанализируем результат.

```
q=0.1; % уровень значимости
fprintf('Выбран уровень значимости q=%4.2f\n',q);
p=anova2(x,1,'off');
if p(1)>=q,
  disp('Столбцы различаются незначимо: фактор B не влияет.')
else
  disp('Столбцы различаются значимо: фактор B влияет.')
end
```

```
if p(2)>=q,
  disp('Строки различаются незначимо: фактор A не влияет.')
else
  disp('Строки различаются значимо: фактор A влияет.')
end
Выбран уровень значимости q=0.10
Столбцы различаются значимо: фактор B влияет.
Строки различаются незначимо: фактор A не влияет.
```

Проверим данный результат графически. Нарисуем на одной фигуре (рис. 23.4) 2 графика: вверху — средние столбцов (кружочками), внизу — средние строк (крестиками).

```
figure % новая фигура
subplot(2,1,1); % верхний рисунок
plot(mean(x),zeros(1,m),'ko'); % средние столбцов
title('\bfСредние столбцов')
subplot(2,1,2); % нижний рисунок
plot(mean(x'),zeros(1,k),'kx'); % средние строк
title('\bfСредние строк')
```

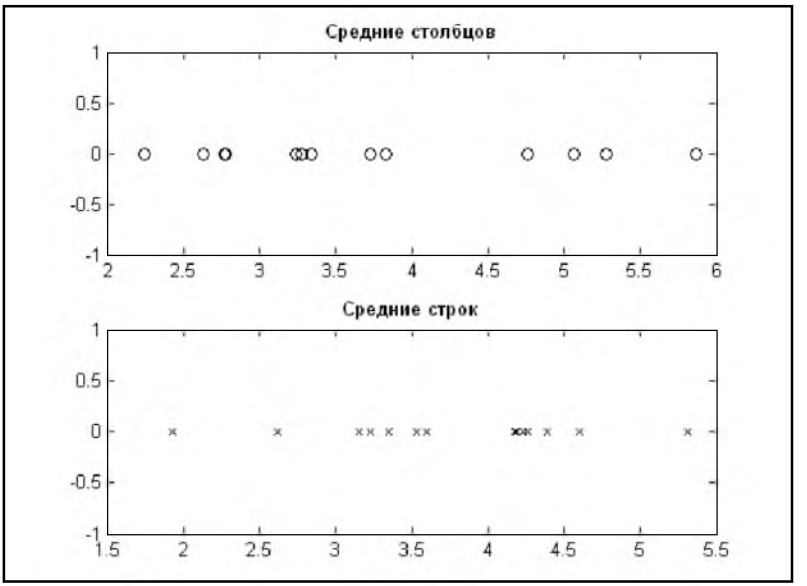


Рис. 23.4. Средние столбцов (вверху) и строк (внизу) в задаче 2-факторного дисперсионного анализа

Действительно, в этом варианте данных разброс средних столбцов незначительно выше разброса средних строк. Разброс средних строк не выделяется

на фоне случайностей, поэтому фактор A влияет незначимо. А разброс средних столбцов уже является значимым.

23.3. Многофакторный дисперсионный анализ

Мы рассмотрим лишь общую постановку задачи и пути ее решения. Пусть $m=k$, т. е. матрица опытов X — квадратная $k\times k$. Все опыты каждой строки проведены при одном и том же уровне фактора A, а все опыты каждого столбца — при одном и том же уровне фактора B. Из предыдущего раздела мы знаем, что этих k^2 опытов достаточно для оценки влияния двух факторов A и B. Но вот появился еще и 3-й фактор C (тоже на k уровнях). Можно ли его учесть, не увеличивая количества опытов? Оказывается, да! Разместим уровни фактора C таким образом, чтобы в каждой строке и каждом столбце матрицы (23.19) было ровно по одному уровню фактора C. Это можно сделать множеством способов. Вот один из них:

$$C=\begin{pmatrix} C_1 & C_2 & \dots & C_k \\ C_2 & C_3 & \dots & C_1 \\ \dots & \dots & \dots & \dots \\ C_k & C_1 & \dots & C_{k-1} \end{pmatrix}. \qquad (23.35)$$

Такое расположение уровней называется латинским квадратом. Каждый опыт x_{ji} будем проводить при A_j, B_i, а уровень C возьмем из (23.35). В выкладках предыдущего раздела усреднение по строкам соответствует одному и тому же уровню A, значит, учитываются B и C. Усреднение по столбцам соответствует учету A и C. А усреднение по элементам побочной диагонали и прямым, ей параллельным, устраняет влияние C. Теперь из этих данных можно найти D_0^*, а затем оценить влияние каждого фактора: A, B и C.

Добавим еще один, 4-й фактор D, тоже на k уровнях. Обойдемся ли мы по-прежнему k^2 опытами? Это можно будет сделать, если построить еще один латинский квадрат, в котором каждое сочетание факторов C (23.35) и D встречается ровно один раз. Такой квадрат называется ортогональным квадрату (23.35), а пара двух ортогональных латинских квадратов — латинским квадратом 2-го порядка. В нем, чтобы устранить влияние фактора D, нужно проводить усреднение по клеткам с одинаковым уровнем фактора D. При этом можно быть уверенным, что в этих k данных встретятся все уровни факторов A, B и C. Эти рассуждения можно продолжать и далее, но теория латинских квадратов выходит далеко за рамки этой книги. Отметим только, что в MATLAB план эксперимента на основе латинских квадратов строит функ-

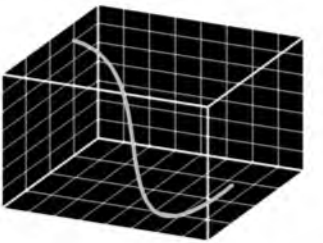
ция `lhsdesign`. Задачи многофакторного дисперсионного анализа можно решить с помощью функции `anovan`.

И еще одно замечание. Во всех наших выкладках мы использовали тот факт, что в опытах встречались все возможные уровни факторов. Такая схема проведения экспериментов называется полным факторным экспериментом (ПФЭ). Ее можно упростить, отбросив некоторые сочетания уровней факторов — получится дробный факторный эксперимент (ДФЭ). Принципы построения различных планов эксперимента изучает раздел математики, который называется "Планирование эксперимента". В [57] есть некоторые процедуры для построения различных планов эксперимента. Они перечислены в *главе 27*.

23.4. Вопросы для самопроверки

1. Почему в формулах (23.5), (23.14) и других применяется 1-сторонний критерий, а не 2-сторонний?
2. Какие данные представлены в таблице, возвращаемой функцией `anova1`?
3. Как подавить вывод таблицы и графика в функции `anova1`?
4. Покажите, что решение системы (23.23) + (23.30) дает результат (23.33).
5. Запишите критерий проверки значимости фактора B в 2-факторном дисперсионном анализе.
6. Выведите критерий проверки факторов A, B и C в 3-факторном дисперсионном анализе на основе сведений, изложенных в *разделе 23.3*.

ГЛАВА 24



Метод наименьших квадратов

Пусть при проведении опытов мы меняем некоторые подконтрольные факторы и хотим выяснить, как они влияют на результат эксперимента. В предыдущей *главе 23* мы исследовали, влияет ли конкретный фактор на результат вообще. Если влияет, то нужно решить следующую задачу: как влияет? В этой главе мы займемся построением теоретических зависимостей по опытным данным. Как правило, такие зависимости строятся с помощью метода наименьших квадратов (МНК), но в конце главы мы рассмотрим и другие подходы.

24.1. МНК и его связь с ПМП

Обозначим через x подконтрольный фактор, различные значения которого будем считать детерминированными. Буквой Y обозначим результат опыта — случайную величину.

ПРИМЕР 24.1. Через каждый час замеряется температура воздуха. Здесь x — время; x_i — моменты времени (детерминированные); Y — температура (случайная). □

Пусть из теоретических соображений известен вид зависимости Y от x. Поскольку величина x — детерминированная, а Y — случайная, то в функциональной зависимости Y от x обязательно должны быть какие-то числовые параметры, которые мы должны считать случайными. Обычно так и бывает: вид теоретической зависимости известен с точностью до числовых параметров, которые подлежат определению из опытных данных:

$$Y = y(x, B_1, B_2, ..., B_m). \qquad (24.1)$$

Здесь B_1, B_2, ..., B_m — параметры, которые являются случайными величинами. Если бы B_j были детерминированными, то для их нахождения достаточно было бы провести m опытов и найти B_j из решения системы уравнений, которые мы предполагаем независимыми:

$$\begin{cases} y(x_j, B_1, B_2, \ldots, B_m) = y_j; \\ j = \overline{1,m}. \end{cases} \tag{24.2}$$

Но из-за того, что B_j — случайные, приходится проводить гораздо больше опытов n: $n >> m$. Примерная картинка, которая при этом получается, показана на рис. 24.1.

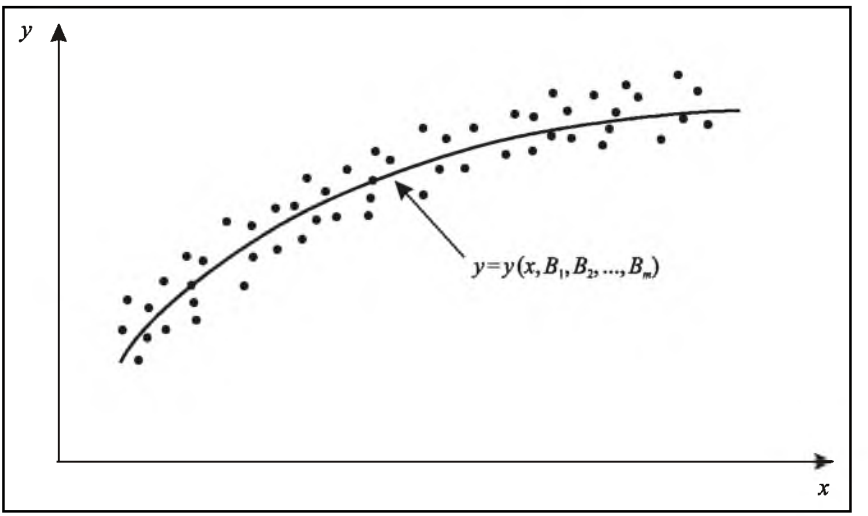


Рис. 24.1. Теоретическая кривая и экспериментальные точки

Понятно, что, т. к. число опытов n больше, чем количество параметров m, мы в общем случае не сможем провести теоретическую кривую через *все* точки (x_i, y_i). Поэтому нужно попытаться провести "наилучшую" теоретическую кривую. Какую же кривую считать наилучшей? Ответ на этот вопрос дает ПМП. Мы должны проводить теоретическую кривую так, чтобы вероятность появления тех экспериментальных данных, которые проявились на практике, была максимальной.

Преобразуем (24.1). Заменим случайные величины B_j их генеральными математическими ожиданиями β_j (это неизвестные детерминированные величины), а все случайности выделим в отдельное слагаемое:

$$Y = y(x, \beta_1, \beta_2, \ldots, \beta_m) + Z. \tag{24.3}$$

Примем следующие гипотезы, которые обычно выполняются на практике.

1. Все опыты независимые.
2. При измерениях нет систематических ошибок и промахов.
3. Ошибка измерений распределена по нормальному закону.
4. Все опыты равноточные.

Поскольку первое слагаемое в (24.3) — детерминированное, то случайная величина Z имеет нормальное распределение и нулевое математическое ожидание. Обозначим неизвестную дисперсию Z через D_z, а среднеквадратичное отклонение — через σ_z. В каждом i-м опыте величина Y принимает конкретное значение y_i за счет того, что Z принимает значение z_i, а B_j во всех опытах принимают значения b_j:

$$z_i = y_i - y(x_i, b_1, b_2, ..., b_m). \tag{24.4}$$

В соответствии с общей схемой выборочного метода считаем каждую z_i единственной реализацией случайной величины Z_i, которая имеет такое же распределение, что и Z; и все Z_i независимы. В нашем случае все Z_i имеют нормальное распределение с нулевым математическим ожиданием и одинаковыми дисперсиями D_z. Плотность распределения каждой из Z_i:

$$f(z_i) = \frac{1}{\sigma_z\sqrt{2\pi}} e^{-\frac{z_i^2}{2\sigma_z^2}}, \tag{24.5}$$

а совместная плотность распределения независимых Z_i:

$$f(z_1, z_2, \ldots, z_n) = \frac{1}{\sigma_z^n (2\pi)^{\frac{n}{2}}} \prod_{i=1}^{n} e^{-\frac{z_i^2}{2\sigma_z^2}} = \frac{1}{\sigma_z^n (2\pi)^{\frac{n}{2}}} e^{-\frac{\sum_{i=1}^{n} z_i^2}{2\sigma_z^2}}. \tag{24.6}$$

На практике проявились значения z_i из (24.4). Согласно ПМП это значит, что совместная плотность распределения (24.6) должна быть максимально возможной. Максимизируем (24.6) за счет подбора b_j, входящих в z_i по (24.4). Первый множитель от b_j не зависит. Экспонента — монотонно возрастающая функция, поэтому нужно максимизировать ее показатель. Знаменатель в показателе также от b_j не зависит, а из-за минуса перед дробью получаем, что максимальная совместная плотность распределения достигается при выполнении условия:

$$L(b_1, b_2, \ldots, b_m) = \sum_{i=1}^{n} z_i^2 = \sum_{i=1}^{n} \left(y_i - y(x_i, b_1, b_2, \ldots, b_m)\right)^2 \to \min. \tag{24.7}$$

Таким образом, ПМП привел нас к результату: параметры теоретической кривой нужно подбирать так, чтобы минимизировать сумму квадратов отклонений экспериментальных ординат y_i от теоретических $y(x_i, b_1, b_2, ..., b_m)$. Это и есть МНК. Функция L с точностью до постоянных слагаемых представляет взятый со знаком минус натуральный логарифм функции правдоподобия.

Если измерения неравноточные, то у каждой Z_i — своя дисперсия D_i и среднеквадратичное отклонение σ_i. В этом случае вместо (24.6) имеем:

$$f(z_1, z_2, \ldots, z_n) = \frac{1}{(2\pi)^{\frac{n}{2}} \prod_{i=1}^{n} \sigma_i} \prod_{i=1}^{n} e^{-\frac{z_i^2}{2\sigma_i^2}} = \frac{1}{(2\pi)^{\frac{n}{2}} \prod_{i=1}^{n} \sigma_i} e^{-\frac{1}{2}\sum_{i=1}^{n} \frac{z_i^2}{\sigma_i^2}}, \qquad (24.8)$$

и вместо обычного МНК получаем взвешенный МНК: нужно минимизировать

$$L(b_1, b_2, \ldots, b_m) = \sum_{i=1}^{n} \frac{z_i^2}{\sigma_i^2} = \sum_{i=1}^{n} \frac{1}{\sigma_i^2} \left(y_i - y(x_i, b_1, b_2, \ldots, b_m)\right)^2 \to \min. \qquad (24.9)$$

При каждом i-м измерении появляется множитель (весовой коэффициент), обратный дисперсии этого измерения. Чем точнее данное измерение (меньше его дисперсия), тем больше весовой коэффициент. Это значит, что теоретическая кривая будет сильнее "притягиваться" к этой точке.

Представление случайной величины Y в виде (24.3) означает, что математическое ожидание каждого измерения Y_i равно:

$$M(Y_i) = y(x_i, \beta_1, \beta_2, ..., \beta_m). \qquad (24.10)$$

Определение 24.1. Уравнение (24.10), определяющее "наилучшую" (в смысле математического ожидания) зависимость y от x, *называется регрессией y на x*. □

Поэтому часто раздел статистики, который занимается построением теоретических зависимостей по экспериментальным данным, называют *регрессионным анализом*.

На практике после минимизации (24.7) или (24.9) мы получаем выборочные b_j, которые являются оценками генеральных β_j. Дальше в этой главе мы научимся строить доверительные интервалы для β_j.

24.2. Система нормальных уравнений Гаусса

Рассмотрим задачу минимизации функции (24.7). По общим правилам исследования на экстремум мы должны найти частные производные, приравнять их нулю и решить полученную систему уравнений:

$$\begin{cases} \dfrac{\partial L}{\partial b_j} = 0; \\ j = \overline{1,m}. \end{cases} \tag{24.11}$$

Затем проводится исследование на выполнение достаточных условий минимума. Это — общая схема. Далее почти вся глава будет посвящена различным частным случаям. Рассмотрим такой вид зависимости от параметров b_j:

$$y = \sum_{j=1}^{m} b_j \psi_j(x), \tag{24.12}$$

где $\psi_j(x)$ — известные (заданные заранее) функции.

Определение 24.2. Функции $\psi_j(x)$ в (24.12) называются *базисными*. □

Представление (24.12) напоминает разложение вектора по ортам координатных осей или вообще любого элемента линейного пространства по его базису. Отсюда и название функций $\psi_j(x)$. Мы будем пользоваться тем, что (24.12) фактически есть разложение n-мерного вектора $y = \{y_1, y_2, ..., y_n\}$ по базису в n-мерном линейном пространстве. Правда, базис этот неполный: вместо n базисных функций есть только m. Но из линейной алгебры мы точно знаем, какими должны быть $\psi_j(x)$: они должны быть линейно-независимыми на множестве точек x_i. Это значит, что система уравнений относительно неизвестных C_j:

$$\begin{cases} \sum_{j=1}^{m} C_j \psi_j(x_i) = 0; \\ i = \overline{1,n}; \end{cases} \tag{24.13}$$

должна иметь только тривиальное решение. Матрица коэффициентов при неизвестных C_j, которая вытянута в высоту ($n > m$), при этом будет иметь полный ранг m. Из теории линейных пространств также известно, что таких функций может быть не больше, чем n. Если базисных функций ровно n, то мы будем иметь полный базис, и кривая пройдет через все точки.

На самом деле вид (24.12) является не таким уж и частным. Вот примеры.

ПРИМЕР 24.2. Степенная зависимость:

$$y = b_0 + b_1 x + b_2 x^2 + ... + b_m x^m. \tag{24.14}$$

Здесь базисные функции: $1, x, x^2, ..., x^m$, и всего их $m+1$. □

ПРИМЕР 24.3. Тригонометрический многочлен (отрезок ряда Фурье):

$$y = \frac{a_0}{2} + \sum_{j=1}^{m}\left(a_j \cos\frac{j\pi x}{l} + b_j \sin\frac{j\pi x}{l} \right). \qquad (24.15)$$

В этом разложении неизвестные коэффициенты обозначены a_0, a_1, b_1, a_2, b_2, ..., a_m, b_m, т. е. всего $2m+1$ параметров и столько же базисных функций. □

ПРИМЕР 24.4. Линейная функция двух переменных:

$$z = b_1 + b_2 x + b_3 y. \qquad (24.16)$$

Имеет 3 базисные функции: 1, x и y. □

Итак, пусть теоретическая кривая имеет вид (24.12). Подставим его в (12.7):

$$L = \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{m} b_j \psi_j(x_i) \right)^2 \to \min. \qquad (24.17)$$

Находим частную производную по конкретному b_k и приравниваем ее нулю:

$$\frac{\partial L}{\partial b_k} = -2\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{m} b_j \psi_j(x_i) \right)\psi_k(x_i) = 0. \qquad (24.18)$$

Сокращаем на 2, раскрываем скобки и перегруппировываем слагаемые:

$$-\sum_{i=1}^{n} y_i \psi_k(x_i) + \sum_{j=1}^{m} b_j \sum_{i=1}^{n} \psi_j(x_i)\psi_k(x_i) = 0. \qquad (24.19)$$

Продолжим нашу аналогию с линейными пространствами, на этот раз уже с эвклидовыми. Внутренняя сумма во втором слагаемом — это скалярное произведение базисных функций $\psi_j(x)$ и $\psi_k(x)$: сумма произведений одноименных координат. Обозначим ее так:

$$(\psi_j, \psi_k) = \sum_{i=1}^{n} \psi_j(x_i)\psi_k(x_i). \qquad (24.20)$$

Первое слагаемое в (24.19) тогда — скалярное произведение вектора $\mathbf{y} = \{y_1, y_2, ..., y_n\}$ на базисную функцию $\psi_k(x)$:

$$(\mathbf{y}, \psi_k) = \sum_{i=1}^{n} y_i \psi_k(x_i). \qquad (24.21)$$

Применяя эти обозначения, из (24.19) имеем систему линейных алгебраических уравнений (СЛАУ) для нахождения неизвестных параметров b_j:

$$\begin{cases} \sum_{j=1}^{m} b_j \left(\psi_j, \psi_k\right) = \left(\mathbf{y}, \psi_k\right); \\ k = \overline{1,m}. \end{cases} \tag{24.22}$$

Определение 24.3. СЛАУ (24.22) называются *системой нормальных уравнений Гаусса* (Карл Фридрих Гаусс (Carl Friedrich Gauss, 1777—1855, рис. 24.2)). □

Рис. 24.2. К. Ф. Гаусс

Из линейной алгебры известно, что система (24.22) имеет единственное решение, если базисные функции линейно-независимые (24.13). Легко также доказать, что полученное решение доставляет минимум функции (24.7). Действительно, при неограниченном увеличении или уменьшении любого b_l величина L стремится к бесконечности. А поскольку полученная стационарная точка единственная, то она доставляет минимум функции L.

Свойство линейной независимости базисных функций является обязательным. Без него набор функций $\psi_1(x)$, $\psi_2(x)$, ..., $\psi_m(x)$ вообще не является базисом. В частности, мы не можем взять в качестве базисной функции нулевую, т. е. такую, которая во всех точках x_i принимает значение 0. Такая функция будет линейно зависима с любой другой. Действительно, можно, например, взять коэффициент при ней $C=1$, а при остальных функциях $C_j=0$, и (24.13) будет удовлетворяться.

Но есть еще одно свойство, которое, не являясь обязательным, значительно упрощает дальнейшие выкладки. Это ортогональность, или его частный случай — ортонормированность.

Определение 24.4. Две различные базисные функции $\psi_j(x)$ и $\psi_k(x)$ называются *ортогональными*, если их скалярное произведение равно нулю:

$$(\psi_j, \psi_k) = 0. \square \tag{24.23}$$

Определение 24.5. *Нормой* базисной функции $\psi_j(x)$ называется квадратный корень из скалярного произведения $\psi_j(x)$ на саму себя:

$$\|\psi_j\| = \sqrt{(\psi_j, \psi_j)} = \sqrt{\sum_{i=1}^{n} \psi_j^2(x_i)}. \square \tag{24.24}$$

Определение 24.6. Система базисных функций $\psi_1(x)$, $\psi_2(x)$, ..., $\psi_m(x)$ называется *ортонормированной*, если они взаимно ортогональны и их нормы равны единице:

$$(\psi_j, \psi_k) = \delta_{jk}. \square \tag{24.25}$$

Здесь δ_{jk} — символ Кронекера, равный 0 при $j \neq k$ и 1 при $j = k$. Если базисные функции ортонормированные (говорят: образуют ортонормированный базис — ОНБ), то матрица коэффициентов системы (24.22) становится единичной, и ее решение находятся сразу:

$$\begin{cases} b_k = (\mathbf{y}, \psi_k); \\ k = \overline{1, m}. \end{cases} \tag{24.26}$$

Но удобство применения ОНБ не исчерпывается только этим фактом. При применении ОНБ легко вычисляется минимальное значение функции L (24.17), которое нам понадобится дальше для нахождения доверительных интервалов параметров аппроксимации. Вычислим $L_{\min}$:

$$\begin{aligned} L_{\min} &= \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{m} b_j \psi_j(x_i) \right)^2 = \\ &= \sum_{i=1}^{n} y_i^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{m} b_j y_i \psi_j(x_i) + \sum_{i=1}^{n} \left(\sum_{j=1}^{m} b_j \psi_j(x_i) \right)^2. \end{aligned} \tag{24.27}$$

Первое слагаемое — это квадрат нормы вектора y. Во втором поменяем порядок суммирования. Квадрат суммы в третьем слагаемом — это двойная сумма с разными индексами. Имеем:

$$L_{\min} = \|\mathbf{y}\|^2 - 2\sum_{j=1}^{m} b_j \sum_{i=1}^{n} y_i \psi_j(x_i) + \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{m} b_j b_k \psi_j(x_i) \psi_k(x_i). \tag{24.28}$$

Внутренняя сумма во 2-м слагаемом — b_j. В 3-м слагаемом меняем порядок суммирования:

$$L_{\min} = \|\mathbf{y}\|^2 - 2\sum_{j=1}^{m} b_j^2 + \sum_{j=1}^{m} b_j \sum_{k=1}^{m} b_k \left(\psi_j, \psi_k\right). \tag{24.29}$$

Из внутренней суммы 3-го слагаемого в силу (24.25) остается только одно j-е слагаемое, равное 1. Поэтому окончательно имеем:

$$L_{\min} = \|y\|^2 - \sum_{j=1}^{m} b_j^2 . \tag{24.30}$$

Отсюда видим еще одно преимущество применения ОНБ: $L_{\min}$ вычисляется по (24.30) легко, а добавление новой базисной функции $\psi_{m+1}(x)$, ортонормированной по отношению ко всем остальным, не требует пересчета предыдущих коэффициентов b_j. Нужно только вычислить новый коэффициент b_{m+1} по формуле (24.26), а из выражения (24.30) вычесть дополнительно квадрат этого коэффициента.

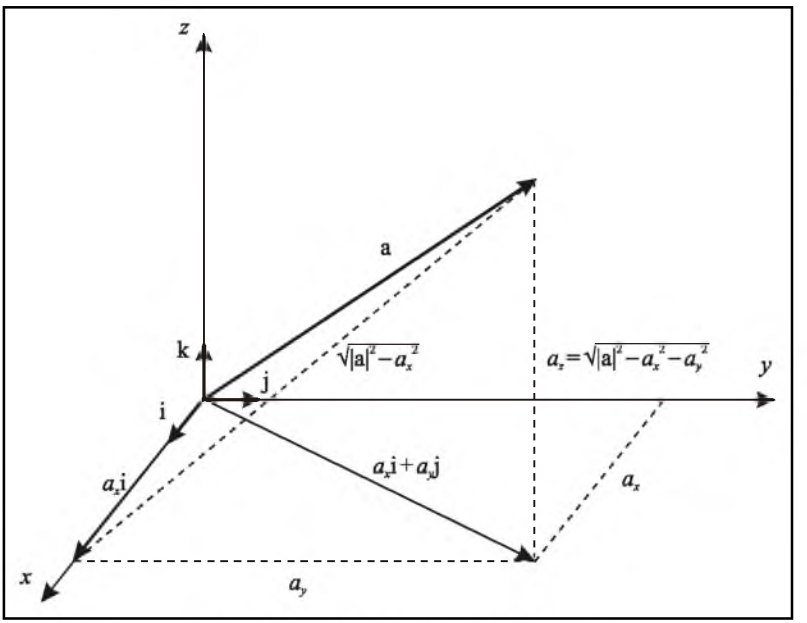


Рис. 24.3. Разложение вектора по ОНБ

Геометрическая иллюстрация этих выкладок на примере разложения вектора a по ОНБ (i, j, k) показана на рис. 24.3. Вектор a здесь — аналог n-мерного

набора экспериментальных данных y. Если мы раскладываем a по ОНБ (i, j, k), то координаты этого разложения находятся так:

$$\begin{cases} a_x = (\mathbf{a}, \mathbf{i}); \\ a_y = (\mathbf{a}, \mathbf{j}); \\ a_z = (\mathbf{a}, \mathbf{k}); \end{cases} \tag{24.31}$$

что является аналогом формулы (24.26). А теперь представьте, что мы хотим "наилучшим образом" приблизить вектор a каким-либо вектором, направленным вдоль оси *Ox*. Это соответствует разложению (24.12) с одной базисной функцией i. Если под наилучшим понимать такое приближение, при котором квадрат модуля разности принимает минимальное значение (МНК!), то нужно ортогонально спроектировать конец вектора a на ось *Ox*. При этом наилучшим окажется вектор a_xi, где a_x находится из (24.31). А погрешность приближения (аналог $L_{\min}$) — это квадрат расстояния между концами векторов, т. е. $|a|^2 - a_x^2$, что соответствует (24.30).

Далее мы хотим повысить точность приближения вектора a и добавляем еще один базисный вектор j, ортонормированный с i. "Наилучшее" приближение (с точки зрения МНК) — это ортогональное проектирование на плоскость *xOy*. Старый коэффициент a_x пересчитывать не нужно, а новый a_y находится также из (24.31). Погрешность приближения теперь равна $|a|^2 - a_x^2 - a_y^2$. В 3-мерном пространстве мы можем добавить еще только одну базисную функцию k, а в *n*-мерном максимальное количество базисных функций равно *n*, при этом $L_{\min}$ уменьшится до нуля, что соответствует прохождению теоретической кривой через все экспериментальные точки.

Так МНК связан с теорией эвклидовых пространств и ортогональным проектированием. Везде, где это возможно и целесообразно, мы будем строить ОНБ, применяя, например, стандартную процедуру ортонормирования по методу Сонина — Шмидта.

24.3. Доверительные интервалы для генеральных параметров аппроксимации

Найденные из решения системы (24.22) или, в случае ОНБ, полученные по (24.26) величины b_k — это оценки соответствующих генеральных параметров β_k. Найдем доверительные интервалы для β_k. Мы ограничимся только случаем ОНБ, т. к. выкладки при этом будут значительно проще. В соответствии с (24.26) каждая случайная величина B_k, реализацией которой является коэффициент b_k, есть линейная комбинация независимых случайных величин Y_i, имеющих нормальное распределение:

$$\begin{cases} B_k = (Y, \psi_k) = \sum_{i=1}^{n} Y_i \psi_k(x_i); \\ k = \overline{1,m}. \end{cases} \tag{24.32}$$

Поэтому распределение всех B_k — нормальное. Математические ожидания B_k:

$$M(B_k) = \sum_{i=1}^{n} M(Y_i)\psi_k(x_i). \tag{24.33}$$

В соответствии с (24.10) и нашим разложением (24.12) математические ожидания величин Y_i, реализациями которых являются измерения y_i, равны:

$$M(Y_i) = \sum_{j=1}^{m} \beta_j \psi_j(x_i). \tag{24.34}$$

Подставим (24.34) в (24.33) и поменяем порядок суммирования:

$$M(B_k) = \sum_{i=1}^{n}\sum_{j=1}^{m} \beta_j \psi_j(x_i)\psi_k(x_i) = \sum_{j=1}^{m} \beta_j (\psi_j, \psi_k). \tag{24.35}$$

В силу ортонормированности базиса (24.25) получаем:

$$M(B_k) = \beta_k. \tag{24.36}$$

Найдем теперь дисперсию B_k. Опять исходим из (24.32). Измерения Y_i — независимые, поэтому $D(B_k)$ находится так:

$$D(B_k) = \sum_{i=1}^{n} D(Y_i)\psi_k^2(x_i). \tag{24.37}$$

В случае равноточных измерений дисперсии всех Y_i одинаковые и равны D_z, значит

$$D(B_k) = D_z \|\psi_k\| = D_z. \tag{24.38}$$

Мы выяснили, что в случае применения ОНБ оценки B_k — нормальные с математическими ожиданиями (24.36) и дисперсиями (24.38). Применяя выкладки (21.1—21.5), получаем доверительный интервал для любого генерального параметра β_k:

$$b_k - \sigma_z u_{1-\frac{q}{2}} \le \beta_k \le b_k + \sigma_z u_{1-\frac{q}{2}}. \tag{24.39}$$

Эту оценку можно применять, если известна генеральная дисперсия D_z. Если же D_z неизвестна (а так чаще всего и бывает), то вместо нее можно взять вы-

борочную дисперсию D_z^*, но тогда вместо квантилей стандартного нормального распределения в (24.39) нужно подставить квантили t-распределения Стьюдента, как мы это делали в (20.8):

$$b_k - \sigma_z^* t_{1-\frac{q}{2}}(f) \le \beta_k \le b_k + \sigma_z^* t_{1-\frac{q}{2}}(f). \qquad (24.40)$$

Осталось только найти выборочную дисперсию и число ее степеней свободы f. По общему правилу число степеней свободы равно числу экспериментальных данных n минус число ограничений. В нашем случае число ограничений равно числу базисных функций m, поэтому

$$f = n - m. \qquad (24.41)$$

А выборочная дисперсия равна деленной на f сумме квадратов отклонений всех экспериментальных данных y_i от их выборочных средних, т. е. теоретических значений:

$$D_z^* = \frac{I_{\min}}{n-m} = \frac{1}{n-m}\left(\sum_{i=1}^{n} y_i^2 - \sum_{j=1}^{m} b_j^2\right). \qquad (24.42)$$

Теперь границы доверительных интервалов (24.40) для каждого коэффициента полностью определены. В случае ОНБ ширина всех интервалов одинаковая. По ним мы можем судить, значимо ли влияет какое-либо слагаемое разложения (24.12). Если доверительный интервал для соответствующего коэффициента включает (охватывает) 0, то на заданном уровне значимости этим слагаемым можно пренебречь. Такой подход позволяет отсеять незначимые факторы и тем самым упростить математическую модель.

Для упрощения модели можно использовать и такой подход. Обычно базисные функции имеют однотипную структуру, а от номера j зависят, как от параметра. Сколько функций нужно взять и где остановиться? Можно поступить так: последовательно увеличивать число функций m и сравнивать две выборочные дисперсии. Если "новая" выборочная дисперсия значительно меньше "старой", то добавленную базисную функцию обязательно нужно учесть. Если же уменьшение дисперсии незначительное или его вообще нет, то добавление новой функции ничего не дает и ее можно отбросить. Вот как выглядит критерий проверки 0-гипотезы о том, что $(m+1)$-я базисная функция влияет незначительно:

$$F_q(n-m, n-m-1) \le \frac{D_{m+1}^*}{D_m^*}. \qquad (24.43)$$

Если (24.43) нарушается, то нужно учитывать слагаемое с ψ_{m+1}. Это правило нужно применять с осторожностью. Может, например, оказаться, что слагае-

мое с ψ_{m+1} влияет незначимо, а следующее с функцией ψ_{m+2} уже значимо. Но обычно в нашем распоряжении есть какие-либо теоретические соображения, позволяющие судить, все ли базисные функции учтены.

24.4. Аппроксимация степенными полиномами

Мы рассмотрели общую теорию МНК и теперь переходим к конкретным задачам. Обычно, если мы ничего не можем сказать о характере поведения теоретической кривой, то первое, что приходит на ум — это попытаться построить ее в виде степенной функции *(см. пример 24.1)*. Для решения этой задачи в MATLAB есть функции `polyfit` и `polyval`. Первая из них находит коэффициенты аппроксимирующего полинома заданной степени, а вторая — вычисляет значения этого полинома в заданных точках. Кроме того, с помощью функции `polyconf` можно вычислить доверительные интервалы, но не для b_j, а для y_i. Если же мы хотим найти оптимальную (минимально возможную) степень аппроксимирующего полинома, то нужно воспользоваться оценками (24.43). Но базисные функции $1, x, x^2, ..., x^m$ в общем случае не образуют ОНБ. Поэтому для применения оценки (24.43) нужно из функций $1, x, x^2, ..., x^m$ построить ОНБ, применяя метод Сонина — Шмидта. Напомним его основные формулы. Пусть имеется некоторый базис $(g_1, g_2, ..., g_m)$. Первый вектор (или, как в нашем случае, функцию) мы просто нормируем:

$$h_1 = \frac{g_1}{\|g_1\|}. \tag{24.44}$$

Из каждого следующего вектора вычитаем составляющие, коллинеарные предыдущим — остается только нормальная составляющая:

$$g_k^* = g_k - (g_k, h_1)h_1 - (g_k, h_2)h_2 - ... - (g_k, h_{k-1})h_{k-1}. \tag{24.45}$$

Полученный вектор g_k^* ортогонален всем $h_1, h_2, ..., h_{k-1}$. Осталось его нормировать:

$$h_k = \frac{g_k^*}{\|g_k^*\|}. \tag{24.46}$$

Полученные векторы $(h_1, h_2, ..., h_m)$ образуют ОНБ.

Выполним ИДЗ по аппроксимации экспериментальных данных степенными полиномами. Будем считать, что исходные данные находятся в текстовом файле poldata.txt в виде двух столбцов: 1-й — аргументы x_i, 2-й — экспериментальные значения y_i. Введем их в программу. Найдем объем выборки n.

```
clear all
sf='D:\Iglin\Matlab\ContData\poldata.txt';
xy=load(sf); % вводим ИД - 2 столбца
x=xy(:,1); % аргументы
y=xy(:,2); % функции
n=length(x) % количество точек
n =
   101
```

Зададим уровень значимости. Выберем оптимальную степень аппроксимирующего полинома. Для этого составим цикл. На каждом его шаге вначале задаем очередную базисную функцию. Затем ортонормируем ее по отношению к предыдущим по формулам (24.44—24.46). Далее вычисляем b_k по (24.26), $L_{\min}$ по (24.30) и выборочные дисперсии по (24.42). Сравниваем выборочные дисперсии текущего и предыдущего приближений по F-критерию Фишера (24.43). Если текущую степень учитывать не нужно, выходим из цикла. То есть мы здесь предполагаем, что, если x^m учитывать не нужно, то и более высокие степени также учитывать не надо. Как мы уже отмечали ранее, на практике это не всегда так. Может, например, случиться, что x^3 не влияет (коэффициент при этой функции малый), а x^4 учитывать уже надо. Поэтому проверку по F-критерию Фишера начнем только со 2-й степени. Во всяком случае, дальше мы построим графики и посмотрим, что получилось.

```
q=0.3 % уровень значимости
Lmin=y'*y; % сумма квадратов y(i)
for m=1:n, % подбираем степень полинома
  psi(:,m)=x.^(m-1); % очередная базисная функция
  if m>1, % проводим ортогонализацию базиса
    for k=1:m-1, % вычитаем || составляющие
      psi(:,m)=psi(:,m)-((psi(:,m))'*psi(:,k))*psi(:,k);
    end
  end
  psi(:,m)=psi(:,m)/norm(psi(:,m)); % нормируем
  bk=y'*psi(:,m); % очередной коэффициент
  Dold=Lmin/(n-m); % старая дисперсия
  Lmin=Lmin-bk^2; % новое значение Lmin
  Dnew=Lmin/(n-m-1); % новая дисперсия
  if m>1, % проверку проводим начиная со 2-й степени
    if Dnew/Dold<finv(q,n-m,n-m-1),
      fprintf('Степень %d нужно учитывать.\n',m-1);
    else
      fprintf('Степень %d не нужно учитывать, выходим.\n',m-1);
      mmax=m-2; % степень аппроксимирующего полинома
```

```
      break; % выход из цикла
    end
  end
end
q =
   0.30000000000000
Степень 1 нужно учитывать.
Степень 2 нужно учитывать.
Степень 3 не нужно учитывать, выходим.
```

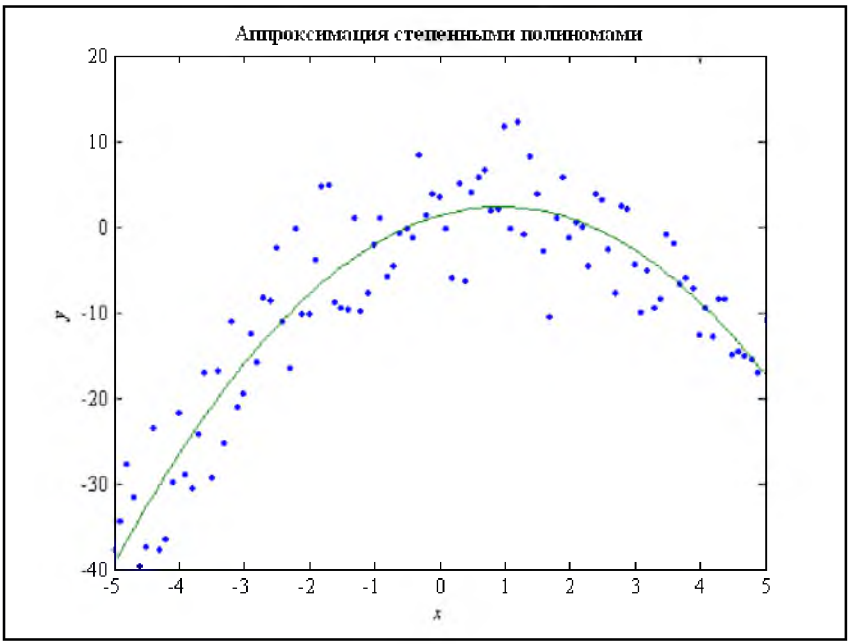


Рис. 24.4. Аппроксимация степенными полиномами

Можно было бы сохранить коэффициенты bk, построить аналитические выражения для ортонормированных базисных функций и записать выражение для аппроксимирующего полинома. Но проще сделать это с помощью функций MATLAB. Поэтому теперь, когда степень полинома определена, вычисляем коэффициенты полинома найденной степени и строим график, похожий на рис. 24.1: теоретическую кривую и экспериментальные точки. Это график показан на рис. 24.4.

```
p=polyfit(x,y,mmax);
fprintf('Аппроксимирующий полином %d-й степени:\ny(x)=',mmax)
fprintf('%+f12*x^%d',[p;[mmax:-1:0]])
```

```
disp(' ');
yt=polyval(p,x); % теоретические значения
figure;
plot(x,y,'.',x,yt,'-');
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfАппроксимация степенными полиномами')
xlabel('\itx') % метка оси OX
ylabel('\ity') % метка оси OY
Аппроксимирующий полином 2-й степени:
y(x)=-1.18454112*x^2+2.20266512*x^1+1.33030512*x^0
```

24.5. Тригонометрическая аппроксимация

Иногда из теории известно, что теоретическая функция $y(x)$ — периодическая с заданным периодом T. Например, температура или деформация на ободе диска ($T=2\pi$) или атмосферное давление в течение суток ($T=1$ сутки). В этом случае теоретическую кривую также удобно искать в виде периодической функции с периодом T. Такими функциями являются константа, синусы и косинусы, поэтому уравнение теоретической кривой ищем в виде отрезка ряда Фурье, как в *примере 24.3*:

$$y=\frac{a_0}{2}+\sum_{j=1}^{m}\left(a_j\cos\frac{2\pi jx}{T}+b_j\sin\frac{2\pi jx}{T}\right). \qquad (24.47)$$

Здесь базисные функции: $\frac{1}{2}$; $\cos\frac{2\pi jx}{T}$; $\sin\frac{2\pi jx}{T}$ — всего $2m+1$ функций. Хорошо было бы на их базе построить ОНБ. Оказывается, это очень просто сделать. Если удачно выбрать точки x_i, то эти функции *уже сами по себе* окажутся ортогональными, и их останется только пронормировать. А удачно выбрать точки x_i тоже очень просто: нужно разбить отрезок $[0; T]$ на n равных интервалов длиной T/n, и взять в качестве x_i концы этих интервалов (или начала — это все равно, т. к. замена последней точки на первую ничего не меняет):

$$x_i=\frac{iT}{n}. \qquad (24.48)$$

Эти точки показаны на рис. 24.5.

■ **ТЕОРЕМА 24.1.** Система базисных функций $\frac{1}{2}$; $\cos\frac{2\pi kx}{T}$; $\sin\frac{2\pi kx}{T}$; где $k=\overline{1,m}$; является ортогональной на множестве точек (24.48).

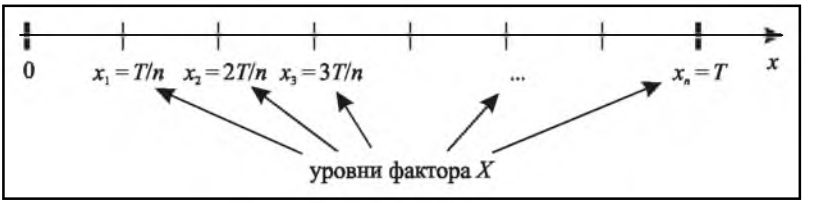


Рис. 24.5. Точки x_i, обеспечивающие ортогональность базисных функций из (24.47)

Доказательство. Проверяем ортогональность 1/2 и любого косинуса:

$$\left(\frac{1}{2}, \cos\frac{2\pi kx}{T}\right) = \frac{1}{2}\sum_{i=1}^{n}\cos\frac{2\pi ki}{n} = \frac{1}{2}\operatorname{Re}\sum_{i=1}^{n} e^{j\frac{2\pi ki}{n}}, \tag{24.49}$$

где j — мнимая единица. Имеем сумму геометрической прогрессии с первым членом $e^{j\frac{2\pi k}{n}}$ и таким же знаменателем. Вычисляем ее:

$$\left(\frac{1}{2}, \cos\frac{2\pi kx}{T}\right) = \frac{1}{2}\operatorname{Re}\frac{e^{j\frac{2\pi k}{n}}\left(1 - e^{j2\pi k}\right)}{1 - e^{j\frac{2\pi k}{n}}} = 0, \tag{24.50}$$

т. к. при любом целом k: $e^{j2\pi k}=1$. В этой формуле при k, кратном n, знаменатель обращается в нуль, но в этом случае можно раскрыть неопределенность и показать, что результат будет такой же. Аналогично 1/2 ортогональна любому синусу: вместо функции Re в формулы (24.49—24.50) нужно поставить значок Im.

Проверяем дальше. Находим скалярное произведение двух различных косинусов (здесь и далее j — уже не мнимая единица, а номер гармоники):

$$\begin{aligned}\left(\cos\frac{2\pi jx}{T}, \cos\frac{2\pi kx}{T}\right) &= \sum_{i=1}^{n}\cos\frac{2\pi ji}{n}\cos\frac{2\pi ki}{n} = \frac{1}{2}\sum_{i=1}^{n}\cos\frac{2\pi(j+k)i}{n} + \\ &+ \frac{1}{2}\sum_{i=1}^{n}\cos\frac{2\pi(j-k)i}{n} = \left(\frac{1}{2}, \cos\frac{2\pi(j+k)x}{T}\right) + \left(\frac{1}{2}, \cos\frac{2\pi(j-k)x}{T}\right).\end{aligned} \tag{24.51}$$

Каждое из слагаемых по (24.50) обращается в нуль, причем второе скалярное произведение мы рассматриваем только при $j \neq k$, поэтому оно тоже равно нулю. Точно так же доказывается ортогональность двух различных синусов. Результат будет отличаться от (24.51) лишь минусом между скалярными произведениями во второй строке. И наконец, докажем ортогональность любых синуса и косинуса (и разных, и одинаковых):

$$\left(\sin\frac{2\pi jx}{T},\cos\frac{2\pi kx}{T}\right)=\sum_{i=1}^{n}\sin\frac{2\pi ji}{n}\cos\frac{2\pi ki}{n}=\frac{1}{2}\sum_{i=1}^{n}\sin\frac{2\pi(j+k)i}{n}+$$
$$+\frac{1}{2}\sum_{i=1}^{n}\sin\frac{2\pi(j-k)i}{n}=\left(\frac{1}{2},\sin\frac{2\pi(j+k)x}{T}\right)+\left(\frac{1}{2},\sin\frac{2\pi(j-k)x}{T}\right). \quad (24.52)$$

Даже при $j=k$ каждое из скалярных произведений равно нулю. □

Эта теорема — дискретный аналог соответствующего свойства ряда Фурье. Только там под скалярным произведением понимался интеграл на отрезке [0; T], а здесь — сумма на равноотстоящих точках.

Применим для тригонометрической аппроксимации теорию ортонормированных базисных функций. По доказанной выше теореме ортогонализация здесь не нужна, наши функции уже ортогональные. Найдем квадраты их норм:

$$\left\|\frac{1}{2}\right\|^2=\left(\frac{1}{2},\frac{1}{2}\right)=\sum_{i=1}^{n}\frac{1}{4}=\frac{n}{4}; \quad (24.53)$$

$$\left\|\cos\frac{2\pi kx}{T}\right\|^2=\sum_{i=1}^{n}\cos^2\frac{2\pi ki}{n}=\frac{1}{2}\sum_{i=1}^{n}\left(1+\cos\frac{4\pi ki}{n}\right)=\frac{n}{2}+\left(\frac{1}{2},\cos\frac{4\pi kx}{T}\right)=\frac{n}{2}; \quad (24.54)$$

$$\left\|\sin\frac{2\pi kx}{T}\right\|^2=\sum_{i=1}^{n}\sin^2\frac{2\pi ki}{n}=\frac{1}{2}\sum_{i=1}^{n}\left(1-\cos\frac{4\pi ki}{n}\right)=\frac{n}{2}-\left(\frac{1}{2},\cos\frac{4\pi kx}{T}\right)=\frac{n}{2}. \quad (24.55)$$

Теперь у нас есть все данные для выполнения ИДЗ по тригонометрической аппроксимации. В исходных данных для этого ИДЗ (файл trigdata.txt) заданы 2 столбца: x_i и y_i, и предполагается, что для x_i выполняется (24.48). Вводим исходные данные, находим объем выборки.

```
clear all
sf='D:\Iglin\Matlab\ContData\trigdata.txt';
xy=load(sf); % вводим ИД - 2 столбца
x=xy(:,1); % аргументы
y=xy(:,2); % функции
n=length(x) % количество точек
n =
   100
```

Задаем уровень значимости. Вначале вычисляем коэффициент a_0 в разложении (24.47). Для этого используем формулу (24.26), для которой нормируем базисную функцию 1/2. Далее в цикле добавляем очередную гармонику (пару слагаемых (24.47)) и проверяем, нужно ли ее учитывать. Для этого находим коэффициенты a_k и b_k, пересчитываем $L_{\min}$, находим выборочные дисперсии

с учетом и без учета этой пары слагаемых, и сравниваем их по F-критерию Фишера. Если вновь добавляемая гармоника влияет незначимо, мы считаем, что и следующие гармоники также будут влиять незначимо, и выходим из цикла.

```
q=0.3 % уровень значимости
a0=y'*ones(size(x))/2/(n/4)^0.5; % коэффициент a0
Lmin=y'*y-a0^2; % начинаем считать Lmin
for m=1:fix(n/2), % подбираем номер гармоники
  a(m)=y'*cos(2*pi*x*m/x(end))/(n/2)^0.5; % коэффициент при cos
  b(m)=y'*sin(2*pi*x*m/x(end))/(n/2)^0.5; % коэффициент при sin
  Dold=Lmin/(n-(2*m-1)); % старая дисперсия
  Lmin=Lmin-a(m)^2-b(m)^2; % новое значение Lmin
  Dnew=Lmin/(n-(2*m+1)); % новая дисперсия
  if Dnew/Dold<finv(q,n-(2*m+1),n-(2*m-1)),
    fprintf('Гармонику %d нужно учитывать.\n',m);
  else
    fprintf('Гармонику %d не нужно учитывать, выходим.\n',m);
    mmax=m-1; % максимальный номер гармоники
    break; % выход из цикла
  end
end
q =
   0.30000000000000
Гармонику 1 нужно учитывать.
Гармонику 2 нужно учитывать.
Гармонику 3 не нужно учитывать, выходим.
```

Печатаем найденный тригонометрический полином. Нам удобнее выразить его не через нормированные тригонометрические функции, а через исходные, как в формуле (24.47), что мы и делаем.

```
fprintf('Тригонометрический полином %d-й степени:\n',mmax)
fprintf('y(x)=%+f12',a0/2/(n/4)^0.5); % средний уровень
w=[1:mmax;ones(1,mmax)*x(end)]; % вспомогательный массив
fprintf('%+f12*cos(2*pi*%d*x/%d)%+f12*sin(2*pi*%d*x/%d)',...
  [a(1:mmax)/(n/2)^0.5;w;b(1:mmax)/(n/2)^0.5;w]);
disp('');
yt=a0*ones(size(x))/2/(n/4)^0.5; % теоретические значения
for k=1:mmax,
  yt=yt+a(k)/(n/2)^0.5*cos(2*pi*x*k/x(end))+...
    b(k)/(n/2)^0.5*sin(2*pi*x*k/x(end));
end
Тригонометрический полином 2-й степени:
y(x)=+1.30468512+1.05934412*cos(2*pi*1*x/10)-2.84942512*sin(2*pi*1*x/10)+
2.38673712*cos(2*pi*2*x/10)-1.09436112*sin(2*pi*2*x/10)
```

И наконец, рисуем картинку (рис. 24.6): теоретическую кривую и экспериментальные точки.

```
figure
plot(x,y,'.',x,yt,'-')
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfТригонометрическая аппроксимация')
xlabel('\itx') % метка оси OX
ylabel('\ity') % метка оси OY
```

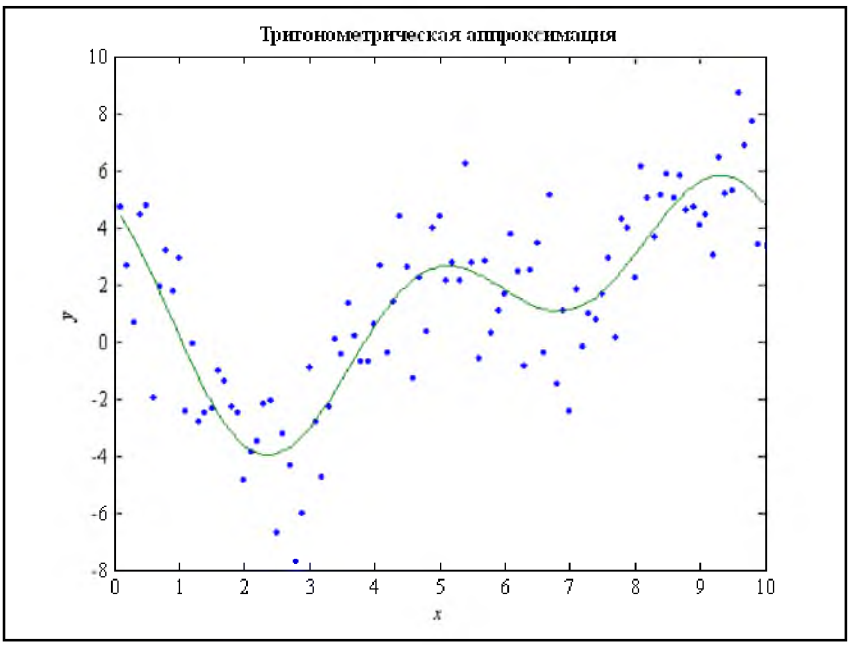


Рис. 24.6. Аппроксимация тригонометрическими полиномами, выполненная с помощью MATLAB

24.6. Аппроксимация функции нескольких переменных

МНК можно применять для аппроксимации функции не только одной, но и нескольких переменных. Если мы используем разложение вида (24.12), то все базисные функции должны зависеть в общем случае от всех переменных:

$$\psi_i(\mathrm{x}) = \psi_i(x_1, x_2, ..., x_N), \tag{24.56}$$

где нижний индекс — это номер переменной, а не точки; N — число переменных (координат вектора x). Особенностью аппроксимации функции нескольких переменных является то, что здесь ортогональность базисных функций достигается не только за счет выбора самих функций, но и за счет выбора точек для эксперимента (уровней факторов). Иными словами, для многофакторного эксперимента необходимо правильное планирование. Рассмотрим некоторые примеры.

24.6.1. Линейная модель для 2-факторного эксперимента

Обозначим независимые переменные через x и y, а результат эксперимента — через z. Пусть по каким-либо теоретическим соображениям z должна быть линейной функцией:

$$z = b_0 + b_x x + b_y y. \tag{24.57}$$

На основании экспериментальных данных (x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n) требуется по МНК найти параметры линейной модели b_0, b_x и b_y. Базисные функции здесь 1, x, y, и в общем случае они не ортогональны. При решении задачи в MATLAB можно воспользоваться интерфейсом `rstool` или просто решить систему нормальных уравнений Гаусса (24.22). Но если мы в дальнейшем захотим наращивать степень аппроксимирующего полинома по одной или обеим переменным, то удобнее это делать в ОНБ. Поэтому рассмотрим построение ОНБ из функций 1, x, y, хотя ИДЗ будем выполнять без него. Условия ортогональности для наших базисных функций имеют вид:

$$(1, x) = \sum_{i=1}^{n} x_i = 0; \quad (1, y) = \sum_{i=1}^{n} y_i = 0; \quad (x, y) = \sum_{i=1}^{n} x_i y_i = 0. \tag{24.58}$$

Первые 2 условия могут быть выполнены, если вести отсчет не от 0, а от средних точек m_x^* и m_y^*:

$$m_x^* = \frac{1}{n}\sum_{i=1}^{n} x_i; \quad m_y^* = \frac{1}{n}\sum_{i=1}^{n} y_i. \tag{24.59}$$

Теперь в линейной модели:

$$z = b_0 + b_x(x - m_x^*) + b_y(y - m_y^*) \tag{24.60}$$

первые 2 условия ортогональности из (24.58) выполняются:

$$\left(1, x - m_x^*\right) = \sum_{i=1}^{n}\left(x_i - m_x^*\right) = 0; \quad \left(1, y - m_y^*\right) = \sum_{i=1}^{n}\left(y_i - m_y^*\right) = 0. \tag{24.61}$$

Третьему условию:

$$\left(x - m_x^*,\, y - m_y^*\right) = \sum_{i=1}^{n}\left(x_i - m_x^*\right)\left(y_i - m_y^*\right) = 0 \tag{24.62}$$

можно удовлетворить, если спланировать эксперимент таким образом, чтобы точки (x_i, y_i) занимали все ячейки прямоугольной таблицы $l \times m$. Такая схема называется *полный факторный эксперимент* (ПФЭ). В данном случае имеем ПФЭ $l \times m$. Будем обозначать уровни фактора x через x_i, $i=1, 2, ..., l$ (строки матрицы эксперимента), а уровни фактора y — через y_j, $j=1, 2, ..., m$ (столбцы матрицы эксперимента). Экспериментальные значения пометим двумя индексами: z_{ij}, они образуют матрицу:

$$\mathbf{Z} = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & z_{22} & \cdots & z_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ z_{l1} & z_{l2} & \cdots & z_{lm} \end{pmatrix}. \tag{24.63}$$

При такой схеме проведения опытов 3-е условие ортогональности (24.62) также выполняется:

$$\left(x - m_x^*,\, y - m_y^*\right) = \sum_{i=1}^{l}\left(x_i - m_x^*\right)\sum_{j=1}^{m}\left(y_j - m_y^*\right) = 0, \tag{24.64}$$

т. к. двойная сумма распадается на произведение двух сумм, каждая из которых равна нулю.

Нормируем наши базисные функции. Квадраты норм:

$$\|1\|^2 = \sum_{i=1}^{l}\sum_{j=1}^{m} 1^2 = lm; \tag{24.65}$$

$$\left\|x - m_x^*\right\|^2 = \sum_{i=1}^{l}\sum_{j=1}^{m}\left(x_i - m_x^*\right)^2 = m\sum_{i=1}^{l}\left(x_i - m_x^*\right)^2; \tag{24.66}$$

$$\left\|y - m_y^*\right\|^2 = \sum_{i=1}^{l}\sum_{j=1}^{m}\left(y_j - m_y^*\right)^2 = l\sum_{i=1}^{l}\left(y_j - m_y^*\right)^2. \tag{24.67}$$

Теперь базисные функции:

$$\psi_0 = \frac{1}{\sqrt{lm}}; \quad \psi_x = \frac{x - m_x^*}{\left\|x - m_x^*\right\|}; \quad \psi_y = \frac{y - m_y^*}{\left\|y - m_y^*\right\|} \tag{24.68}$$

представляют собой ОНБ. По (24.26) находим коэффициенты аппроксимации:

$$b_0 = (\mathbf{z}, \psi_0) = \frac{1}{\sqrt{lm}} \sum_{i=1}^{l} \sum_{j=1}^{m} z_{ij}; \tag{24.69}$$

$$b_x = (\mathbf{z}, \psi_x) = \frac{\sum_{i=1}^{l} \sum_{j=1}^{m} z_{ij} \left(x_i - m_x^*\right)}{\sqrt{m \sum_{i=1}^{l} \left(x_i - m_x^*\right)^2}}; \tag{24.70}$$

$$b_y = (\mathbf{z}, \psi_y) = \frac{\sum_{i=1}^{l} \sum_{j=1}^{m} z_{ij} \left(y_j - m_y^*\right)}{\sqrt{l \sum_{j=1}^{m} \left(y_j - m_y^*\right)^2}}; \tag{24.71}$$

и строим линейную модель, которая вместо (24.60) теперь имеет вид:

$$z = b_0 \psi_0 + b_x \psi_x + b_y \psi_y. \tag{24.72}$$

Проверку адекватности линейной модели можно провести, сравнивая генеральную дисперсию опытов D_0 (если она известна) с выборочной дисперсией линейного приближения D_l, которая вычисляется по формуле (24.42). В нашем случае:

$$D_l = \frac{1}{lm-3} \left(\sum_{i=1}^{l} \sum_{j=1}^{m} z_{ij}^2 - b_0^2 - b_1^2 - b_2^2 \right), \tag{24.73}$$

а число степеней свободы $f = lm - 3$, т. к. на lm опытов накладывается 3 ограничения (выбраны 3 базисные функции). Линейную модель можно считать подходящей на уровне значимости q, если D_l не слишком велика по сравнению с D_0:

$$\frac{D_l}{D_0} \le F_{1-q}(lm-3, \infty). \tag{24.74}$$

Если же (24.74) нарушается, то линейная модель не годится. В этом случае нужно подбирать другие теоретические зависимости.

Если генеральная дисперсия D_0 неизвестна, то пока что линейную модель нам не с чем сравнить. В этом случае нужно увеличивать степени полиномов по x и (или) по y и сравнивать выборочные дисперсии различных приближений.

Выполним ИДЗ по линейной аппроксимации функции двух переменных. Исходные данные для этого задания находятся в текстовом файле twolin.txt в виде матрицы размером $(l+1)\times(m+1)$. Первый столбец (начиная со 2-го элемента) — уровни фактора x. Первая строка (начиная со 2-го элемента) — уровни фактора y. Остальная часть матрицы — экспериментальные данные z_{ij}. Введем исходные данные. Определим размерности задачи.

```
clear all
sf='D:\Iglin\Matlab\ContData\twolin.txt';
od=load(sf); % вводим ИД
x=od(2:end,1); % уровни фактора x
y=od(1,2:end); % уровни фактора y
z=od(2:end,2:end); % результаты эксперимента
l=length(x)
m=length(y)
l =
    18
m =
    14
```

Построим сетку ПФЭ. Вычислим коэффициенты при неизвестных и правые части системы нормальных уравнений Гаусса (24.22). Коэффициенты — это всевозможные скалярные произведения базисных функций: 1, x, y. Правые части — скалярные произведения экспериментальных данных z_{ij} на базисные функции. Решим систему (24.22). Напечатаем решение.

```
[X,Y]=meshgrid(y,x); % сетка узловых точек
A(1,1)=l*m; % начинаем заполнять матрицу
A(1,2)=sum(sum(X)); % коэффициентов системы
A(1,3)=sum(sum(Y)); % уравнений Гаусса
A(2,2)=sum(sum(X.^2));
A(2,3)=sum(sum(X.*Y));
A(3,3)=sum(sum(Y.^2));
A=A+(triu(A,1))'; % дополнили симметрично
c(1)=sum(sum(z)); % правые части
c(2)=sum(sum(z.*X));
c(3)=sum(sum(z.*Y));
c=c(:); % сделали столбец
b=A\c; % решаем систему уравнений Гаусса
disp('Линейная модель:')
fprintf('z(x,y)=%f12%+f12*x%+f12*y.',b);
Линейная модель:
z(x,y)=2.55280112-2.31240312*x-2.31925912*y.
```

Вычислим по этой формуле теоретические значения z. Построим на одном графике (рис. 24.7) теоретическую плоскость и экспериментальные точки. Выберем точку просмотра.

```
zt=b(1)+b(2)*X+b(3)*Y; % теоретические аппликаты
figure;
surf(X,Y,zt); % теоретическая поверхность
hold on
plot3(X,Y,z,'b.'); % экспериментальные точки
hold off
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfЛинейная модель для 2-факторного эксперимента')
xlabel('\itx') % метка оси OX
ylabel('\ity') % метка оси OY
zlabel('\itz') % метка оси OZ
box on % ограничивающий прямоугольник
grid on % сетка
view(70,40) % выбрали точку просмотра
```

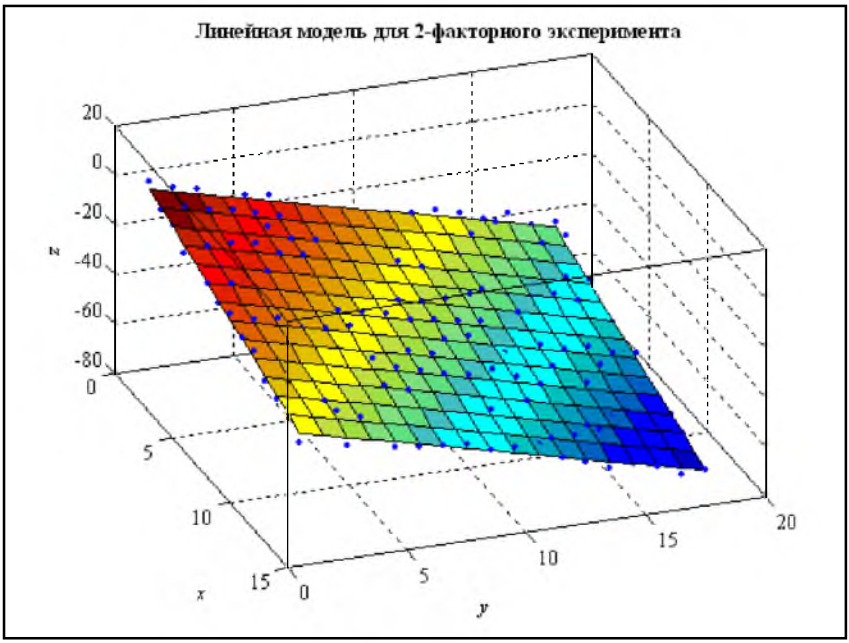


Рис. 24.7. Линейная аппроксимация функции двух переменных, выполненная с помощью MATLAB

24.6.2. Полином для 2-факторного эксперимента

Пусть после проверки (24.74) выяснилось, что линейная модель непригодна для аппроксимации имеющихся экспериментальных данных. В этом случае нужно повысить степень аппроксимирующего полинома по одной или обеим переменным. Если мы заранее определились со степенями по x и y, можно просто добавлять новые базисные функции в разложение (24.57). В этом случае построение системы уравнений Гаусса (24.22) и ее решение средствами MATLAB не вызывает затруднений. Гораздо интереснее решить задачу подбора оптимальной степени полинома. До каких пор наращивать степень по той или иной переменной? Включать ли слагаемые с произведением xy, т. е. учитывать ли взаимовлияние факторов? Для ответа на эти вопросы нужно построить ОНБ. Тогда мы сможем, добавляя те или иные функции в набор (24.12), проверять значимость этих добавок по (24.43). А построить ОНБ в случае ПФЭ очень просто.

■ **ТЕОРЕМА 24.2.** Пусть функции $p_1(x)$, $p_2(x)$, ..., $p_l(x)$ образуют ОНБ на множестве точек x_1, x_2, ..., x_l (очевидно, $L \le l$). Пусть другая система функций $q_1(y)$, $q_2(y)$, ..., $q_M(y)$ также образует ОНБ на множестве точек y_1, y_2, ..., y_m, где $M \le m$. Тогда их всевозможные произведения $T_{\alpha\beta}(x, y) = p_\alpha(x)q_\beta(y)$ образуют ОНБ на сетке ПФЭ $l \times m$.

Доказательство. Вычислим скалярное произведение двух различных функций T, т. е. функций, у которых отличается хотя бы один нижний индекс:

$$\left(T_{\alpha\beta}, T_{\gamma\delta}\right) = \sum_{i=1}^{l}\sum_{j=1}^{m} p_\alpha\left(x_i\right) q_\beta\left(y_j\right) p_\gamma\left(x_i\right) q_\delta\left(y_j\right) = \left(p_\alpha, p_\gamma\right)\left(q_\beta, q_\delta\right) = 0, \quad (24.75)$$

т. к. хотя бы одно из скалярных произведений равно нулю. Из этой же формулы видно, что норма каждой функции T равна 1. □

Эта теорема позволяет легко строить ОНБ для функции двух, трех и большего числа переменных. Заметим, что иногда удается построить и ДФЭ, для которого условие (24.75) выполняется, но это гораздо более сложная задача.

Далее задача решается стандартными методами, которые мы рассмотрели выше. Коэффициенты аппроксимации находим по (24.26). Выборочные дисперсии вычисляем по (24.42), а сравниваем по (24.43). По результатам сравнения делаем вывод: включать или нет проверяемое слагаемое в (24.12).

24.7. Нелинейная зависимость от параметров

До сих пор во всех задачах мы использовали теоретическую зависимость вида (24.12). Теоретическая кривая (или поверхность) могла нелинейно зависеть

от аргументов, но всегда линейно зависела от параметров аппроксимации b_j. Иногда теоретические соображения подсказывают другое.

ПРИМЕР 24.5. Радиоактивный распад. Скорость распада вещества в данный момент времени прямо пропорциональна его количеству в данный момент. Теоретическая зависимость массы y от времени x:

$$y = b_1 e^{-b_2 x}, \tag{24.76}$$

где $b_1 > 0$ — начальная масса, $b_2 > 0$ — параметр скорости распада. Если заданы экспериментальные точки, то уравнение теоретической кривой нужно искать в виде (24.76). □

ПРИМЕР 24.6. Насыщение. По этому закону изменяется, например, ток во время переходного процесса при включении напряжения и большом сопротивлении в цепи. Уравнение кривой насыщения:

$$y = b_1 \left(1 - e^{-b_2 x}\right). \tag{24.77}$$

Здесь $b_1 > 0$ — предельное значение, к которому стремится функция, $b_2 > 0$ — параметр скорости достижения предельного значения. □

Разумеется, могут быть и другие виды теоретических зависимостей, нелинейных относительно параметров b_j. Для решения этой задачи в MATLAB имеется функция `nlinfit`. В качестве входных параметров ей нужно передать экспериментальные точки (x_i, y_i), вид теоретической функции и начальные приближения для параметров b_j. Процедура для вычисления теоретической функции должна быть составлена по определенным правилам. Ее заголовок должен иметь вид:

```
function y=MyFunc(beta,x)
```

Здесь первый аргумент — параметры аппроксимации, а второй — аргументы x_i. Возвращаться должны теоретические ординаты y_i. Функцию нужно записать в какой-либо каталог, доступный системе MATLAB, как обычную m-функцию. В данном случае нужно записать ее в файл с именем MyFunc.m. Возможны также и другие варианты задания функции, которые можно посмотреть в справочной службе.

Выполним ИДЗ по построению кривой насыщения. Пусть исходные данные для него хранятся в текстовом файле с именем nlin.txt в виде двух столбцов. Первый столбец — это x_i, а второй — y_i. Введем данные в программу, определим объем выборки.

```
clear all
sf='D:\Iglin\Matlab\ContData\nlin.txt';
```

```
xy=load(sf); % вводим ИД - 2 столбца
x=xy(:,1); % аргументы
y=xy(:,2); % функции
n=length(x) % количество точек
n =
   101
```

Составим функцию, реализующую вычисление ординат кривой насыщения (24.77). Запишем ее в текущий каталог и выведем на экран.

```
s{1}='function y=MyFunc(beta,x)'; % заголовок
s{2}='y=beta(1)*(1-exp(-beta(2)*x));';
filename=fullfile(pwd,'MyFunc.m'); % файл
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:});
fid=fopen(filename,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записали файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\ContData\MyFunc.m:
function y=MyFunc(beta,x)
y=beta(1)*(1-exp(-beta(2)*x));
```

Решим задачу нелинейной аппроксимации с помощью функции `nlinfit`. Напечатаем уравнение найденной кривой насыщения.

```
b=nlinfit(x,y,'MyFunc',[1 1]);
disp('Уравнение кривой насыщения:')
fprintf('y=%f12*(1-exp(%+f12*x)).\n',b(1),-b(2))
Уравнение кривой насыщения:
y=1.46252112*(1-exp(-0.83460912*x)).
```

Вычислим теоретические значения в наших точках x_i. Нарисуем на одном графике (рис. 24.8) экспериментальные точки и теоретическую кривую насыщения. Сотрем ненужный теперь файл MyFunc.m.

```
yt=MyFunc(b,x); % вычислили теоретические y
figure;
plot(x,y,'.',x,yt,'-');
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfКривая насыщения')
xlabel('\itx') % метка оси OX
ylabel('\ity') % метка оси OY
delete(filename) % удалили файл
```

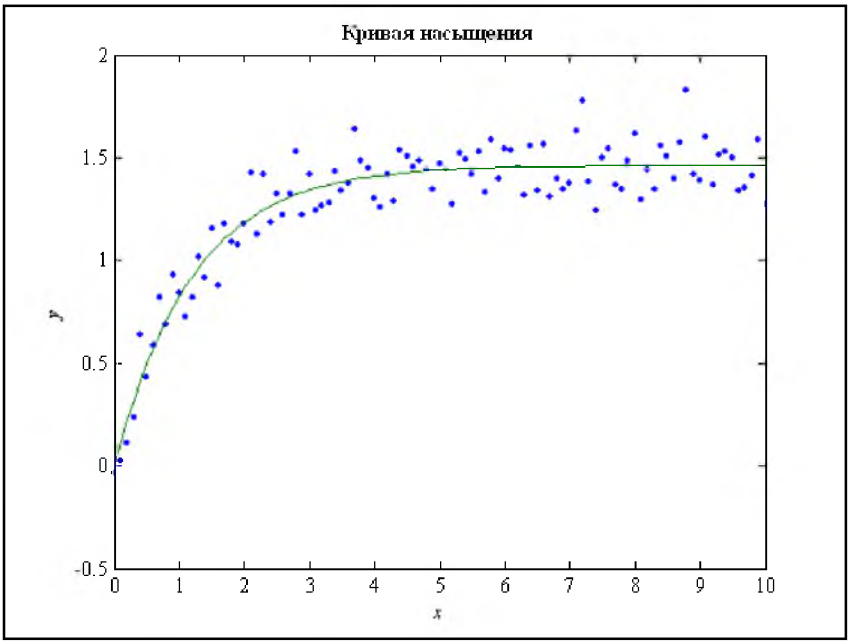


Рис. 24.8. Нелинейная зависимость от параметров, выполненная с помощью MATLAB

24.8. Метод наименьших, но не квадратов

Если распределение случайных ошибок нормальное, а измерения независимые, то из ПМП следует МНК. Но не всегда гипотезы из *раздела 24.1* выполняются. Поэтому иногда вместо минимизации суммы квадратов отклонений экспериментальных точек от теоретических значений приходится минимизировать совсем другую величину:

- сумму модулей отклонений;
- максимальное по модулю отклонение;
- сумму модулей кубов или более высоких степеней отклонений;
- другую величину, характеризующую отклонение экспериментальных точек от теоретической кривой.

Любую из этих задач легко можно решить с помощью MATLAB. Рассмотрим, например, первые две постановки: минимум суммы модулей отклонений и минимум максимального по модулю отклонения. Решим эти задачи для того же самого варианта данных, что и в *разделе 24.4*. При этом ограничимся для всех случаев полиномом 2-й степени. Вначале введем исходные данные и найдем параметры квадратичной модели по МНК.

```
clear all
sf='D:\Iglin\Matlab\ContData\poldata.txt';
xy=load(sf); % вводим ИД - 2 столбца
x=xy(:,1); % аргументы
y=xy(:,2); % функции
n=length(x) % количество точек
p=polyfit(x,y,2);
fprintf('Полином 2-й степени по МНК:\ny(x)=')
fprintf('%+f12*x^%d',[p;[2:-1:0]]);
fprintf('\n');
n =
   101
Полином 2-й степени по МНК:
y(x)=-1.18454112*x^2+2.20266512*x^1+1.33030512*x^0
```

В инструментарии MATLAB для решения задач оптимизации [53] есть функция `fminsearch`, с помощью которой решается задача минимизации функции нескольких переменных. Ей нужно передать в качестве параметров имя минимизируемой функции (m-файла), начальное приближение, параметры настройки, а также параметры, которые нужно передать минимизируемой функции. Составим такую функцию и запишем ее в текущий каталог.

```
s{1}='function L=MyFuncSmod(b,x,y)'; % заголовок
s{2}='L=sum(abs(y-b(1)*x.^2-b(2)*x-b(3)));';
fn1=fullfile(pwd,'MyFuncSmod.m'); % файл
disp(['Текст файла ' fn1 ':'])
fprintf('%s\n',s{:});
fid=fopen(fn1,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записали файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\ContData\MyFuncSmod.m:
function L=MyFuncSmod(b,x,y)
L=sum(abs(y-b(1)*x.^2-b(2)*x-b(3)));
```

Эта функция будет минимизироваться по аргументам `b`, а переменные `x` и `y` — параметры. В качестве начальных приближений коэффициентов b_j берем значения, полученные по МНК. Решаем задачу и печатаем найденный аппроксимирующий полином.

```
p1=fminsearch('MyFuncSmod',p,[],x,y);
fprintf('Полином 2-й степени по минимуму суммы модулей:\ny(x)=')
fprintf('%f12*x^%d',[p1;[2:-1:0]]);
fprintf('\n');
Полином 2-й степени по минимуму суммы модулей:
y(x)=-1.17954412*x^2+2.20674612*x^1+0.99880712*x^0
```

Точно так же поступаем и во втором случае. Составляем функцию, которая будет минимизироваться, и записываем ее в текущий каталог.

```
s{1}='function L=MyFuncMmod(b,x,y)'; % заголовок
s{2}='L=max(abs(y-b(1)*x.^2-b(2)*x-b(3)));';
fn2=fullfile(pwd,'MyFuncMmod.m'); % файл
disp(['Текст файла ' fn2 ':'])
fprintf('%s\n',s{:});
fid=fopen(fn2,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записали файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\ContData\MyFuncMmod.m:
function L=MyFuncMmod(b,x,y)
L=max(abs(y-b(1)*x.^2-b(2)*x-b(3)));
```

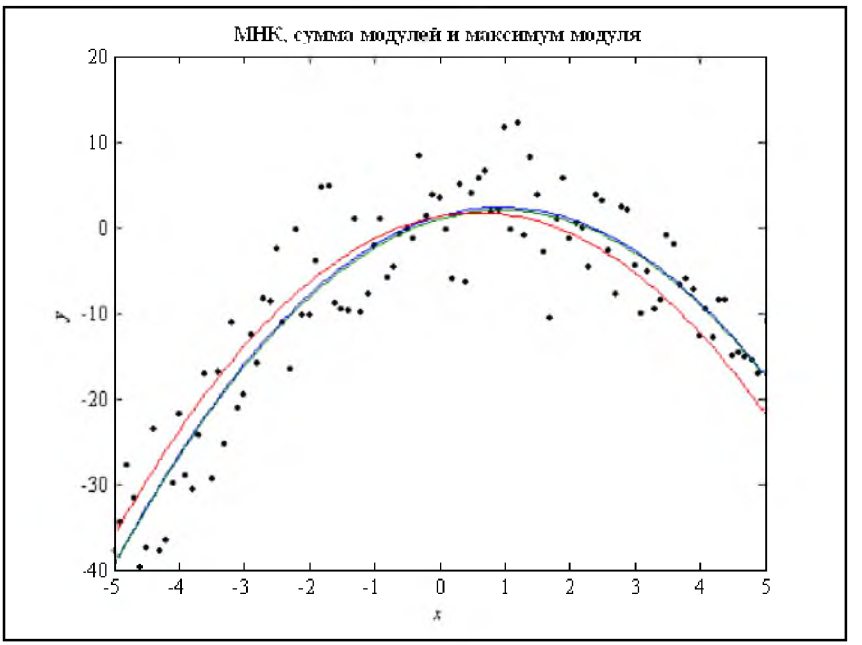


Рис. 24.9. Метод наименьшей суммы модулей и наименьшего максимума модуля отклонений

Минимизируем эту функцию — находим коэффициенты полинома 2-й степени по методу наименьшего максимального отклонения.

```
p2=fminsearch('MyFuncMmod',p,[],x,y);
fprintf(['Полином 2-й степени по '...
  'минимуму максимума модуля:\ny(x)='])
```

```
fprintf('%f12*x^%d',[p2;[2:-1:0]]);
fprintf('\n');
Полином 2-й степени по минимуму максимума модуля:
y(x)=-1.20620612*x^2+1.42225112*x^1+1.28805612*x^0
```

Вычисляем теоретические значения и строим все 3 графика на одном рисунке (рис. 24.9). Удаляем ненужные теперь файлы.

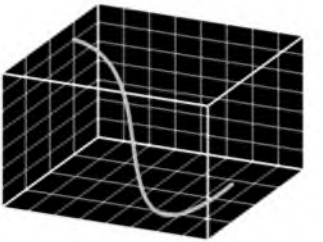
```
yt=polyval(p,x); % теоретические значения по МНК
yt1=polyval(p1,x); % теоретические значения по сумме модулей
yt2=polyval(p2,x); % теоретические значения по max модуля
figure;
plot(x,y,'.k',x,yt,'-',x,yt1,'-',x,yt2,'-');
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfМНК, сумма модулей и максимум модуля')
xlabel('\itx') % метка оси OX
ylabel('\ity') % метка оси OY
delete(fn1) % удалили файлы
delete(fn2)
```

24.9. Вопросы для самопроверки

1. Почему в МНК предполагается, что все случайные измерения распределены по нормальному закону? Как изменится метод, если все ошибки будут иметь равномерное распределение с нулевыми математическими ожиданиями и одинаковыми дисперсиями?
2. Можно ли построить больше, чем n базисных функций? Почему (или как)?
3. Выведите доверительные интервалы для генеральных параметров аппроксимации β_j, если базисные функции не ортогональны.
4. Найдите в литературе или Интернете доказательство формул (24.41—24.42).
5. Какое максимальное значение m возможно в формуле (24.47)? Будут ли выполняться условия ортогональности для более высоких гармоник?
6. Будут ли ортогональными базисные функции (24.47), если точки x_i выбирать не в конце каждого участка, как на рис. 24.3, а, например, в середине? Или на любом, но одинаковом расстоянии от начала?
7. Решите с помощью MATLAB задачу построения линейной модели для 2-факторного эксперимента, используя ОНБ. Будет ли отличаться окончательный результат?

8. Для исходных данных из *раздела 24.6.1* постройте полную квадратичную модель, т. е. учтите дополнительно слагаемые с базисными функциями x^2, xy и y^2. Какими получились коэффициенты аппроксимации для них?
9. Найдите доверительные интервалы для параметров кривой насыщения с помощью функции `nlparci`.
10. Сильно ли отличаются кривые МНК, метода наименьшей суммы модулей отклонений и метода наименьшего максимального по модулю отклонения?

ГЛАВА 25



Корреляционный анализ

В курсе теории вероятностей обычно изучается тема "функции случайных величин". Там рассматриваются *детерминированные* функции *случайных* величин. Например:

$$Y = \varphi(X). \tag{25.1}$$

Здесь X — случайная величина, имеющая некоторый закон распределения, а φ — детерминированная функция. В результате применения функции φ к величине X получается другая случайная величина Y, которая имеет уже свой закон распределения, в общем случае отличный от закона распределения X. Если провести статистические испытания (взять выборку x_i), то соответствующие значения y_i будут равны $\varphi(x_i)$. Экспериментальные точки (x_i, y_i) в точности ложатся на график функции (25.1), как показано на рис. 25.1, *а*.

Другая, в некотором смысле прямо противоположная задача была нами рассмотрена в предыдущей *главе 24*. Там аргументы x_i были детерминированными, а y_i — случайными. Эта ситуация изображена на рис. 25.1, *в*.

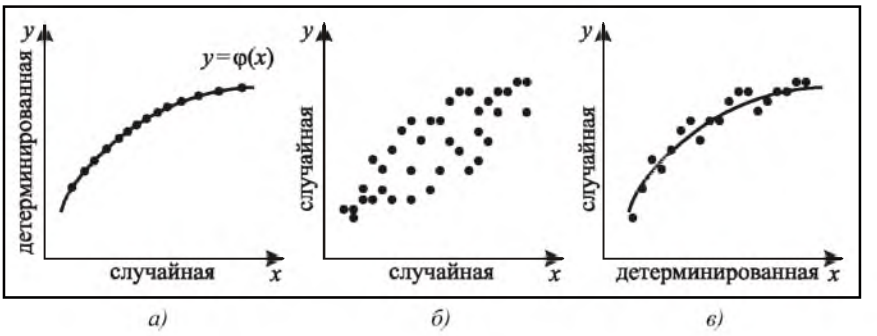


Рис. 25.1. Связь корреляционного анализа с другими задачами математической статистики: *а* – функция случайной величины; *б* – корреляционный анализ; *в* – регрессионный анализ

В этой главе мы рассмотрим промежуточную (или смешанную) задачу, иллюстрация к которой — на рис. 25.1, *б*. Здесь реализации и X, и Y — случайные, и мы будем оценивать, взаимосвязаны ли между собой эти величины, и если да, то насколько сильно. В более общем случае задан многомерный случайный вектор X, и нужно по заданной выборке оценить силу взаимного влияния его координат. Но этот случай сводится к предыдущему, т. к. координаты вектора X можно рассматривать попарно.

25.1. Понятие о корреляции

Связь между двумя случайными величинами X и Y является связью особого рода: когда при изменении X меняется Y, то нельзя заранее сказать, является ли это следствием зависимости Y от X или здесь сказывается влияние случайных факторов в самих X и Y. Связь такого рода называется стохастической.

Определение 25.1. Связь между координатами случайного вектора называется *стохастической.* □

ПРИМЕР 25.1. Температура воздуха и атмосферное давление в данный момент времени — две случайные величины. Связь между ними — стохастическая. □

В стохастической связи есть две составляющие. Одна из них зависит от взаимного влияния X и Y, а другая — от случайностей в самих X и Y.

Определение 25.2. Та часть стохастической связи, которая определяется взаимовлиянием X и Y, называется *стохастической составляющей* стохастической связи. □

Определение 25.3. Та часть стохастической связи, которая определяется случайностями самих X и Y, называется *случайной составляющей* стохастической связи. □

Структура стохастической связи показана на рис. 25.2. Из определений 25.2 и 25.3 следует, что чем больше доля стохастической составляющей в стохастической связи, тем сильнее связаны между собой X и Y. И наоборот, чем выше доля случайной составляющей, тем меньше они связаны. Поэтому, если мы хотим численно оценить силу взаимосвязи X и Y, нужно ввести в рассмотрение такую числовую характристику, которая могла бы характеризовать долю стохастической составляющей в стохастической связи. Давайте посмотрим, как это можно сделать.

Начнем с независимых величин. Мы знаем, что для независимых X и Y дисперсия их суммы равна сумме дисперсий:

$$D(X+Y)=D_x+D_y. \qquad (25.2)$$

Это условие могло бы служить критерием разделения величин на зависимые и независимые, если бы было справедливо и обратное утверждение. Но это, к сожалению, не так: если X и Y зависимые, то это не значит, что (25.2) нарушается. Бывают такие зависимые величины, для которых (25.2) выполняется. Значит, мера нарушения (25.2) может служить характеристикой не всей стохастической составляющей, а только некоторой ее части, которая называется корреляцией.

Рис. 25.2. Структура стохастической связи

Определение 25.4. *Корреляцией* называется та часть стохастической составляющей стохастической связи, которая влияет на нарушение равенства (25.2). □

Это определение также проиллюстрировано на рис. 25.2. А на рис. 25.3 показано разделение случайных величин на зависимые и независимые, коррелированные и некоррелированные.

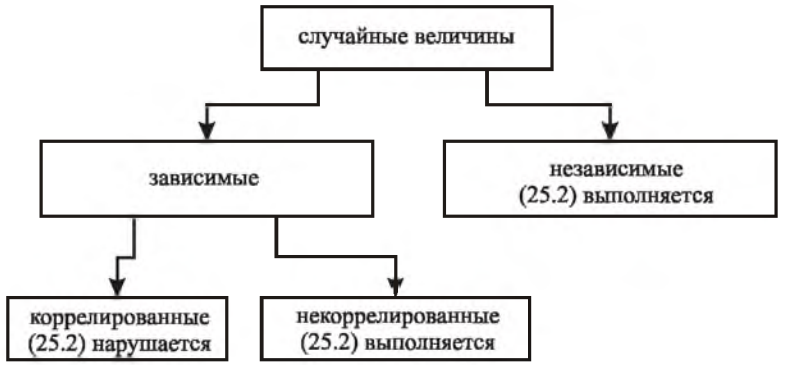


Рис. 25.3. Зависимые и независимые, коррелированные и некоррелированные случайные величины

Определение 25.5. Случайные величины X и Y называются *коррелированными*, если для них (25.2) нарушается, и *некоррелированными*, если (25.2) имеет место. □

Очевидно, если X и Y коррелированные, то они и зависимые, но не наоборот. Выполнение или невыполнение условия (25.2) служит критерием разделения величин не на зависимые и независимые, а на коррелированные и некоррелированные. На первый взгляд кажется, что это не совсем то, что нам нужно, но на самом деле это не так. Существует очень широкий класс случайных величин, для которых *любая* зависимость означает коррелированность. В частности, такими являются все нормально распределенные величины (дальше мы это докажем). А поскольку в математической статистике мы часто имеем дело именно с нормальными величинами, то будем оценивать силу взаимосвязи между случайными величинами X и Y по корреляции между ними. Оценим ее численно. В общем случае, если X и Y зависимые, дисперсия их суммы равна:

$$D(X+Y)=M((X-m_x)+(Y-m_y))^2=D_x+D_y+2M((X-m_x)(Y-m_y)). \quad (25.3)$$

Появление второго смешанного центрального момента свидетельствует о зависимости X и Y. Если он отличен от нуля, то наши X и Y коррелированные. Поэтому по величине $M((X-m_x)(Y-m_y))$ можно судить об абсолютной величине корреляции. Но нам было бы удобнее иметь относительную величину, которая характеризовала бы *долю* корреляции в стохастической связи. Введем такую величину.

Определение 25.6. *Коэффициентом корреляции* ρ называется отношение второго смешанного центрального момента величин X и Y к произведению их среднеквадратичных отклонений:

$$\rho=\frac{M\left((X-m_x)(Y-m_y)\right)}{\sqrt{D_xD_y}}=\frac{M\left((X-m_x)(Y-m_y)\right)}{\sigma_x\sigma_y}. \quad \square \quad (25.4)$$

Если $\rho=0$, то X и Y некоррелированные, т. к. в этом случае выполняется (25.2). И наоборот, если $\rho\neq 0$, то они коррелированные: (25.2) нарушается. Сформулируем и докажем еще некоторые свойства коэффициента корреляции.

Свойство 25.1. Величина ρ не меняется, если к X и (или) к Y прибавить произвольное детерминированное слагаемое.

Доказательство. Добавление детерминированной константы C добавляет к математическому ожиданию такую же константу и не влияет на дисперсию. При вычислении 2-го смешанного центрального момента в числителе формулы (25.4) C взаимно уничтожается, поэтому величина ρ не меняется. □

Свойство 25.2. Величина ρ не меняется, если X и (или) Y умножить на произвольный детерминированный положительный множитель C.

Доказательство. Умножение случайной величины на константу умножает ее математическое ожидание на эту же константу, а дисперсию — на квадрат этой константы. При вычислении ρ по формуле (25.4) в числителе выходит за скобки множитель C, а в знаменателе $|C|$. Так как $C>0$, то они сокращаются, и ρ не меняется. □

Свойство 25.3. Величина ρ изменит только знак, если одну из величин X или Y умножить на –1.

Доказательство. По предыдущему свойству при вычислении ρ по формуле (25.4) в числителе выходит за скобки множитель –1, а в знаменателе 1. Таким образом, ρ меняет только знак. □

Свойство 25.4. Величина ρ не изменится, если перейти от X и Y к центрированным и нормированным величинам по формулам:

$$X_0 = \frac{X - m_x}{\sigma_x}; \quad Y_0 = \frac{Y - m_y}{\sigma_y}. \tag{25.5}$$

Доказательство следует из свойств 25.1 и 25.2. Сначала мы вычитаем из случайной величины детерминированную константу, а затем делим на положительное число. Каждое из этих действий не меняет ρ. □

Свойство 25.5. Величина ρ лежит в пределах:

$$-1 \le \rho \le 1. \tag{25.6}$$

Доказательство. Если в (25.3) подставить (25.4), то получим:

$$D(X+Y) = D_x + D_y + 2\rho\sqrt{D_x D_y}. \tag{25.7}$$

Умножим теперь величину Y на –1. По свойству 25.3 ρ изменит только знак, а дисперсии не изменятся:

$$D(X-Y) = D_x + D_y - 2\rho\sqrt{D_x D_y}. \tag{25.8}$$

Применим формулы (25.7—25.8) к центрированным и нормированным величинам X_0 и Y_0. Для них дисперсии равны 1, поэтому имеем:

$$\begin{cases} D(X_0 + Y_0) = 2 + 2\rho; \\ D(X_0 - Y_0) = 2 - 2\rho. \end{cases} \tag{25.9}$$

Дисперсия любой случайной величины неотрицательна. Решаем неравенства:

$$\begin{cases} 2+2\rho \ge 0; \\ 2-2\rho \ge 0; \end{cases} \quad \begin{cases} \rho \ge -1; \\ \rho \le 1; \end{cases} \tag{25.10}$$

откуда и следует (25.6). □

Свойство 25.6. Если $\rho=1$, то X и Y связаны детерминированной линейной зависимостью $Y=kX+b$ с $k>0$. При $\rho=-1$ они связаны детерминированной линейной зависимостью $Y=kX+b$ с $k<0$.

Доказательство. Из 2-го уравнения системы (25.9) при $\rho=1$ получаем, что $D(X_0-Y_0)=0$, а это свидетельствует о том, что $X_0-Y_0=C$ — детерминированная величина. Подставляя сюда (25.5), получим, что связь между Y и X линейная с положительным коэффициентом. Доказательство второго утверждения следует из такого же анализа первого уравнения системы (25.9). □

Свойство 25.7 (обратное). Если $Y=kX+b$ с $k>0$, то $\rho=1$. Если $Y=kX+b$ с $k<0$, то $\rho=-1$ (здесь k и b — детерминированные константы).

Доказательство. Вычислим математические ожидания, дисперсии и 2-й смешанный центральный момент: $m_y=km_x+b$; $D_y=k^2D_x$; $M((X-m_x)(Y-m_y))=$ $=kD_x$. Подставим в (25.4):

$$\rho=\frac{kD_x}{\sqrt{k^2D_x^2}}=\frac{k}{|k|}=\operatorname{sign} k. \tag{25.11}$$

Имеем: при $k>0$: $\rho=1$, а при $k<0$: $\rho=-1$. □

Свойство 25.8. Если X и Y нормальные и некоррелированные ($\rho=0$), то они независимые.

Доказательство. Двумерная плотность нормального распределения имеет вид:

$$f(x,y)=\frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}}e^{-\frac{1}{2(1-\rho^2)}\left(\frac{(x-m_x)^2}{\sigma_x^2}+\frac{2\rho(x-m_x)(y-m_y)}{\sigma_x\sigma_y}+\frac{(y-m_y)^2}{\sigma_y^2}\right)}. \tag{25.12}$$

Если в этом выражении положить $\rho=0$, то получим:

$$f(x,y)=\frac{1}{2\pi\sigma_x\sigma_y}e^{-\frac{(x-m_x)^2}{2\sigma_x^2}-\frac{(y-m_y)^2}{2\sigma_y^2}}=\frac{1}{\sigma_x\sqrt{2\pi}}e^{-\frac{(x-m_x)^2}{2\sigma_x^2}}\frac{1}{\sigma_y\sqrt{2\pi}}e^{-\frac{(y-m_y)^2}{2\sigma_y^2}}, \tag{25.13}$$

т. е. 2-мерная плотность распределения представляется в виде произведения 1-мерных плотностей. А это возможно только при независимых X и Y. □

Из этих свойств следует, что коэффициент корреляции действительно является относительной характеристикой. Его значение лежит в пределах (25.6). Крайним значениям $\rho = \pm 1$ соответствует детерминированная — причем, линейная — зависимость между X и Y. Значению $\rho = 0$ соответствует некоррелированность, а в случае нормальных величин и независимость X и Y. На рис. 25.4 показаны примерные картинки разброса экспериментальных точек (x_i, y_i) при различных значениях ρ.

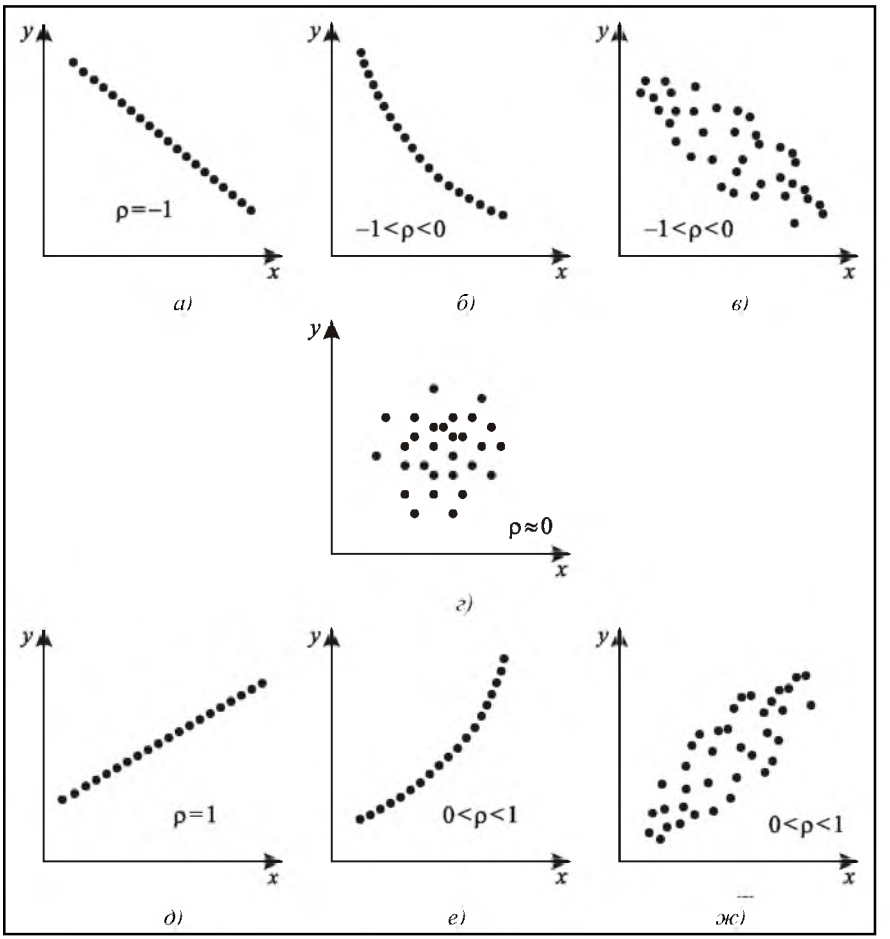


Рис. 25.4. Разброс экспериментальных точек при различных значениях ρ

График *а* — детерминированная линейная зависимость с отрицательным угловым коэффициентом. Для этого случая $\rho = -1$. На графике *б* — тоже детерминированная и убывающая зависимость, но нелинейная. Поэтому здесь ρ

отрицательный, но до –1 не доходит. Такое же значение имеет ρ и на графике *в*, но по иной причине: здесь взаимосвязь между X и Y имеет, кроме стохастической, еще и случайную составляющую. А отрицательное значение ρ связано с общим убыванием значений Y при возрастании X. Величины, изображенные на графике *г*, практически некоррелированные. Здесь ρ близко к нулю. Графики *д*, *е* и *ж* повторяют соответственно *а*, *б* и *в* с точностью до знака ρ.

25.2. Оценка коэффициента корреляции по данным наблюдений

Пусть проведено n испытаний, и получена выборка (x_1, y_1), (x_2, y_2), ..., (x_n, y_n). Требуется по этим данным оценить коэффициент корреляции генеральной совокупности.

Определение 25.7. Оценка коэффициента корреляции по данным наблюдений называется *корреляционным анализом*. □

Эту задачу можно решить по общей схеме выборочного метода. Предположим, что генеральные совокупности X и Y имеют нормальное распределение с математическими ожиданиями m_x, m_y, дисперсиями D_x, D_y и коэффициентом корреляции ρ. Плотность такого распределения — это (25.12). Тогда каждое x_i можно считать единственной реализацией случайной величины X_i, а y_i — реализацией Y_i, и эти величины имеют такое же распределение, что и исходные генеральные совокупности. Величины с разными индексами независимы (опыты мы считаем независимыми), но в каждой паре (X_i, Y_j) есть корреляция с коэффициентом ρ. По этим величинам находим выборочные математические ожидания M_x^* и M_y^* (18.15), выборочные дисперсии D_x^* и D_y^* (18.26), выборочный 2-й смешанный центральный момент:

$$M_{xy}^* = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - M_x^*\right)\left(Y_i - M_y^*\right), \tag{25.14}$$

а затем функцию, реализацией которой является выборочный коэффициент корреляции ρ^*:

$$R = \frac{M_{xy}^*}{\sqrt{D_x^* D_y^*}}. \tag{25.15}$$

Распределение этой случайной величины зависит от объема выборки n и от генерального коэффициента корреляции ρ (можно показать, что ρ является математическим ожиданием R, т. е. доказать несмещенность оценки). От m_x,

m_y, D_x и D_y распределение величины R не зависит: математические ожидания взаимно уничтожаются в (25.14), а дисперсии сокращаются в (25.15).

Определение 25.8. Распределение случайной величины R (25.15) называется *r-распределением выборочного коэффициента корреляции*, соответствующего заданному генеральному ρ. □

Обычно в первую очередь интересуются принципиальным вопросом: коррелированные X и Y вообще или нет? Для решения этого вопроса достаточно иметь таблицы (или формулы для вычисления) квантилей r-распределения лишь для одного значения $\rho = 0$. Тогда 0-гипотезу о некоррелированности величин X и Y можно принять на уровне значимости q, если выборочный коэффициент корреляции ρ^* не слишком большой по модулю: его значение должно попадать в квантильные границы:

$$-r_{1-\frac{q}{2}} \le \rho^* \le r_{1-\frac{q}{2}}. \qquad (25.16)$$

Если (25.16) нарушается, нужно принять одну из двух альтернативных гипотез: $\rho > 0$ или $\rho < 0$.

Выполним ИДЗ по корреляционному анализу — проведем проверку коррелированности двух столбцов данных. Данные находятся в текстовом файле с именем corrdata.txt. Их формат — 2 столбца: соответственно x_i и y_i. Введем их в программу, найдем объем выборки.

```
clear all
sf='D:\Iglin\Matlab\ContData\corrdata.txt';
xy=load(sf); % вводим ИД - 2 столбца
n=size(xy,1) % количество точек
n =
   500
```

Имеющаяся в MATLAB функция `corrcoef` вычисляет корреляционную матрицу для массива, состоящего из нескольких столбцов данных. На диагонали возвращаются единицы, а внедиагональные элементы представляют собой коэффициенты корреляции между соответствующими столбцами. Если затребовать второй выходной параметр, то в нем возвращаются (на внедиагональных местах) критические числа для проверки 0-гипотезы. Маленьким числам (меньше заданного уровня значимости) соответствует альтернативная гипотеза, а большим — 0-гипотеза.

Зададим уровень значимости. Вычислим коэффициент корреляции между столбцами и критическое значение для проверки 0-гипотезы. Сравним критическое значение с заданным уровнем значимости и сделаем вывод.

```
q=0.1; % уровень значимости
fprintf('Выбран уровень значимости q=%4.2f\n',q);
[r,p]=corrcoef(xy);
fprintf('Выборочный коэффициент корреляции r=%14.8f\n',r(1,2));
fprintf('Статистика =%14.8f, и она ',p(1,2));
if p(1,2)<q,
  disp('меньше q => корреляция значима.')
else
  disp('больше q => корреляция незначима.')
end
```

```
Выбран уровень значимости q=0.10
Выборочный коэффициент корреляции r=   -0.10103463
Статистика =    0.02386270, и она меньше q => корреляция значима.
```

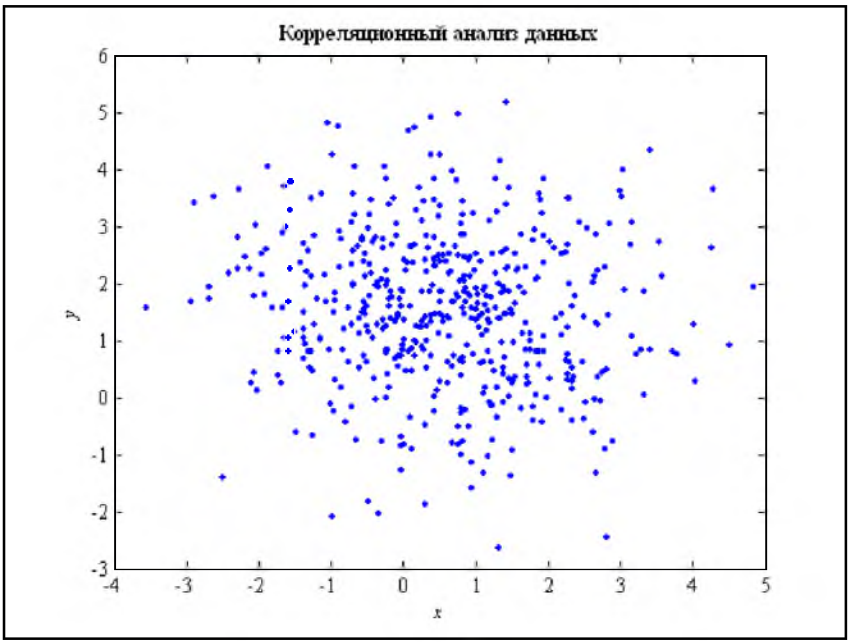


Рис. 25.5. Разброс экспериментальных точек в ИДЗ по корреляционному анализу данных

Нарисуем картинку наподобие той, что изображена на одном из графиков рис. 25.4. Она показана на рис. 25.5.

```
figure % новая фигура
plot(xy(:,1),xy(:,2),'b.')
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
```

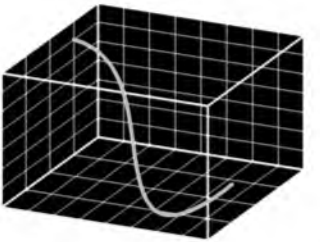
```
title('\bfКорреляционный анализ данных')
xlabel('\itx')
ylabel('\ity')
```

На первый взгляд кажется, что картинка похожа на рис. 25.4, *г*, и величины некоррелированные. Но из-за достаточно большого объема выборки тест (25.16) смог выявить корреляцию.

25.3. Вопросы для самопроверки

1. Случайная величина X — нормальная, а $Y=X^3$. Чему равен коэффициент корреляции между ними?
2. Как с помощью функции `corrcoef` найти доверительные интервалы для коэффициента корреляции при заданном уровне значимости? Допишите соответствующий фрагмент кода.

ГЛАВА 26



Генерация вариантов заданий

Эта глава предназначена для преподавателей, ведущих занятия по математической статистике. Но студентам тоже интересно будет посмотреть, как готовятся для них варианты индивидуальных домашних заданий (ИДЗ). Заголовки разделов этой главы соответствуют названиям заданий. Для подготовки каждого из них мы будем применять средства MATLAB, в частности, датчики случайных чисел. Используя программы этой главы, преподаватели в зависимости от объема и содержания изучаемого курса могут выбрать те или иные ИДЗ для своих студентов. В первом разделе главы подробно разобран механизм работы с файлами и папками. В следующих разделах этой главы он используется при генерации других ИДЗ, поэтому подробно не описывается.

26.1. Обработка массива данных

Студенту предлагается n измерений одной и той же величины, имеющей некоторый неизвестный ему заранее закон распределения. Порядок выполнения этого ИДЗ рассмотрен в *главах 18, 20 и 21*. При "ручном" счете обычно ограничиваются небольшим объемом выборки $n \approx 50$, и задание выдается студентам в напечатанном виде. Но при использовании различных программных пакетов, в частности MATLAB, можно задавать достаточно большой объем выборки: 200, 500, 1000 и больше чисел. А само задание можно выдать студентам в виде дискеты с текстовыми файлами или разместить на сайте преподавателя в Интернете. Для удобства проверки нужно сгенерировать также файл с ответами.

Пусть текстовый файл со списком студентов имеет имя, совпадающее с названием группы, расширение txt и хранится в некоторой папке. Для примера рассмотрим условную студенческую группу факультета прикладной биологии, специализация номер 8, 2002 год приема. По существующей в нашем

университете системе индексации такая группа называлась бы ПБ-82. Файл со списком студентов этой группы будет иметь имя ПБ-82.txt. Он должен содержать только текстовые строки. Каждая строка файла — это ФИО одного студента. Пусть в этой группе учатся такие студенты (все ФИО вымышленные, любые совпадения — случайные):

Аланьев Георгий Алексеевич
Антонова Юлия Владиславовна
Бережков Лев Иванович
...
Шерванский Эраст Арнольдович
Щербатько Василиса Владимировна

Здесь приведены только начало и конец списка. Для сокращения листингов мы оставим в файле ПБ-82.txt только 5 строк с перечисленными ФИО.

Содержимое файла ПБ-82.txt

```
Аланьев Георгий Алексеевич
Антонова Юлия Владиславовна
Бережков Лев Иванович
Шерванский Эраст Арнольдович
Щербатько Василиса Владимировна
```

Данные с вариантами ИДЗ и ответами удобно записать в одну новую папку. Для этого создадим в папке, где содержится файл ПБ-82.txt, дочернюю папку с именем, которое состоит из названия группы и названия ИДЗ. В данном случае это будет папка \ПБ-82-Обработка массива данных. В нее мы поместим файл для каждого студента и файл для преподавателя с ответами. Имя файла для студента — это его ФИО (расширение txt), а файл ответов так и будет называться: Ответы.txt.

Будем руководствоваться этими правилами и при генерации других заданий в следующих разделах. Тогда в папке, где хранится файл со списком студентов, будут собраны и все папки с вариантами ИДЗ для них.

Реализуем этот план. Зададим полный путь доступа к файлу со списком студентов (идентификатор `filegr`) и объем выборки. Введем и напечатаем список группы. Выделим в полном имени название папки, имя файла (название группы) и расширение.

```
clear all % очислили рабочую область
filegr='D:\Iglin\Matlab\ContData\ПБ-82.txt'; % файл группы
n=500; % задаем объем выборки
```

```
nametask='Обработка массива данных'; % название ИДЗ
gr=textread(filegr,'%s','delimiter','\n'); % ввели данные
ngr=length(gr); % число студентов группы
bsl=findstr(filegr,'\'); % ищем вхождения символа \
dirgr=filegr(1:bsl(end)); % имя папки с файлом группы
nameextgr=filegr(bsl(end)+1:end); % имя файла группы
p=findstr(nameextgr,'.'); % ищем расширение
namegr=nameextgr(1:p(1)-1); % название группы
fprintf('Список группы %s\n',namegr);
for k=1:ngr,
  fprintf('%d. %s\n',k,gr{k});
end
```

```
Список группы ПБ-82
1. Аланьев Георгий Алексеевич
2. Антонова Юлия Владиславовна
3. Бережков Лев Иванович
4. Шерванский Эраст Арнольдович
5. Щербатько Василиса Владимировна
```

Сформируем из названия группы имя папки и создадим ее. Если она уже существовала, очистим ее. Приготовим и печатаем названия распределений.

```
[succ,mess,messid]=mkdir(dirgr,...
  [namegr '-' nametask]); % создаем папку
if ~succ, % выход при ошибке
  error(['Не удалось создать папку ' dirstr nsmegr]);
end
newdirgr=[dirgr namegr '-' nametask '\'];
delete([newdirgr '*.*']); % удалили все из папки
tdistr={'нормальное','экспоненциальное',...
  'равномерное','рэлеевское'}; % типы распределений
disp('Возможные типы распределений:');
fprintf('%s\n',tdistr{:});
```

```
Возможные типы распределений:
нормальное
экспоненциальное
равномерное
рэлеевское
```

Проводим основной цикл. Для каждого студента группы с помощью датчика случайных чисел выбирается сначала тип распределения, потом его параметры и, наконец, генерируется сама выборка. Выборка записывается в файл для студента, а тип распределения и его параметры — в файл ответов. Для контроля тип распределения и его параметры выводятся также на экран.

```
fileans=[newdirgr 'Ответы.txt']; % имя файла ответов
fidans=fopen(fileans,'w'); % открыли файл с ответами
for k=1:ngr, % генерируем ИДЗ
  ndistr=unidrnd(4); % случайный тип распределения
  fprintf(fidans,'%d. %s\nТип распределения: %s\n',...
    k,gr{k},tdistr{ndistr}); % имя студента, тип распределения
  fprintf('%d. %s\nТип распределения: %s\n',...
    k,gr{k},tdistr{ndistr}); % в зону вывода для контроля
  switch ndistr, % параметры распределения
    case 1, % нормальное распределение
      mu=unifrnd(-2,2); % случайное среднее
      sigma=unifrnd(0.5,3); % случайное СКО
      fprintf('mu=%14.8f;  sigma=%14.8f.\n',mu,sigma); % на экран
      fprintf(fidans,'mu=%14.8f;  sigma=%14.8f.\n\n',mu,sigma);
      x=normrnd(mu,sigma,n,1); % генерация данных
    case 2, % экспоненциальное распределение
      lambda=unifrnd(0.5,2.5);
      fprintf('lambda=%14.8f.\n',lambda); % на экран
      fprintf(fidans,'lambda=%14.8f.\n\n',lambda);
      x=exprnd(1/lambda,n,1); % генерация данных
    case 3, % равномерное распределение
      a=unifrnd(-2,0.1);
      b=unifrnd(0.5,4);
      fprintf('a=%14.8f;  b=%14.8f.\n',a,b); % на экран
      fprintf(fidans,'a=%14.8f;  b=%14.8f.\n\n',a,b);
      x=unifrnd(a,b,n,1); % генерация данных
    case 4, % рэлеевское распределение
      sigma=unifrnd(0.5,2.5);
      fprintf('sigma=%14.8f.\n',sigma); % на экран
      fprintf(fidans,'sigma=%14.8f.\n\n',sigma);
      x=raylrnd(sigma,n,1); % генерация данных
  end
  filestud=[newdirgr gr{k} '.txt']; % имя файла студента
  fid=fopen(filestud,'w'); % открыли файл студента
  fprintf(fid,'%14.8f  %14.8f  %14.8f  %14.8f  %14.8f\n',x);
  fclose(fid); % закрыли файл студента
end
fclose(fidans); % закрыли файл с ответами
1. Аланьев Георгий Алексеевич
Тип распределения: экспоненциальное
lambda=    1.33977679.
2. Антонова Юлия Владиславовна
```

```
Тип распределения: нормальное
mu=   -1.38724815;  sigma=    0.96149499.
3. Бережков Лев Иванович
Тип распределения: нормальное
mu=    0.36393252;  sigma=    1.70951715.
4. Шерванский Эраст Арнольдович
Тип распределения: равномерное
a=   -1.91859077;  b=    1.75877658.
5. Щербатько Василиса Владимировна
Тип распределения: рэлеевское
sigma=    0.77438936.
```

Фактически в зоне вывода (на экране) повторяется содержимое файла Ответы.txt. Этот файл преподаватель оставляет себе, а все остальное содержимое папки \ПБ-82-Обработка массива данных отдает студентам. Каждый из них находит там свой файл и выполняет ИДЗ.

26.2. Сравнение двух выборок

В этом ИДЗ студенты должны сравнить математические ожидания и дисперсии двух выборок. Выполнение задания описано в *разделах 22.1 и 22.2*. Для каждого студента нужно сгенерировать по 2 нормально распределенные выборки.

Будем использовать те же принципы работы с папками и файлами, что и в *разделе 26.1*. По заданному имени группы создаем новую папку с названием задания, куда помещаем файлы для студентов и файл с ответами. Для каждого студента генерируем две нормально распределенные выборки со случайными математическими ожиданиями и среднеквадратичными отклонениями. Выборки записываем в файл для студента (в виде двух столбцов), а их параметры — в файл ответов и в область вывода. Вот как выглядит текст программы для генерации этого ИДЗ.

```
clear all % очистили рабочую область
filegr='D:\Iglin\Matlab\ContData\ПБ-82.txt'; % файл группы
n=100; % задаем объем выборки
nametask='Сравнение двух выборок'; % название ИДЗ
gr=textread(filegr,'%s','delimiter','\n'); % ввели данные
ngr=length(gr); % число студентов группы
bsl=findstr(filegr,'\'); % ищем вхождения символа \
dirgr=filegr(1:bsl(end)); % папка с файлом группы
nameextgr=filegr(bsl(end)+1:end); % имя файла группы
p=findstr(nameextgr,'.'); % ищем расширение
namegr=nameextgr(1:p(1)-1); % название группы
```

```
[succ,mess,messid]=mkdir(dirgr,...
  [namegr '-' nametask]); % создаем папку
if ~succ, % выход при ошибке
  error(['Не удалось создать папку ' dirstr nsmegr]);
end
newdirgr=[dirgr namegr '-' nametask '\'];
delete([newdirgr '*.*']); % удалили все из папки
fileans=[newdirgr 'Ответы.txt']; % имя файла ответов
fidans=fopen(fileans,'w'); % открыли файл с ответами
for k=1:ngr, % генерируем ИДЗ
  mu=unifrnd(1.2,2.2,1,2); % средние
  sigma=unifrnd(4,8,1,2); % СКО
  fprintf(fidans,['%d. %s\nmu1=%14.8f;  mu2=%14.8f;  '...
    's1=%14.8f;  s2=%14.8f.\n'],k,gr{k},mu,sigma);
  fprintf(['%d. %s\nmu1=%14.8f;  mu2=%14.8f;  '...
    's1=%14.8f;  s2=%14.8f.\n'],k,gr{k},mu,sigma);
  x=normrnd(mu(1),sigma(1),n,1); % 1-я выборка
  x=[x,normrnd(mu(2),sigma(2),n,1)]; % 2-я выборка
  filestud=[newdirgr gr{k} '.txt'];  % имя файла студента
  fid=fopen(filestud,'w'); % открыли файл студента
  fprintf(fid,'%14.8f  %14.8f\n',x');
  fclose(fid); % закрыли файл студента
end
fclose(fidans); % закрыли файл с ответами
```

```
1. Аланьев Георгий Алексеевич
mu1=  2.02298596;  mu2=  2.14967609;  s1=  6.26648350;  s2=  4.23785176.
2. Антонова Юлия Владиславовна
mu1=  1.39077935;  mu2=  1.71716460;  s1=  4.13877042;  s2=  4.82336794.
3. Бережков Лев Иванович
mu1=  2.18428329;  mu2=  2.09530155;  s1=  6.78580244;  s2=  6.25134635.
4. Шерванский Эраст Арнольдович
mu1=  1.31624057;  mu2=  1.79414072;  s1=  4.49728947;  s2=  4.28294821.
5. Щербатько Василиса Владимировна
mu1=  2.07725455;  mu2=  2.19678810;  s1=  6.62553560;  s2=  6.65336497.
```

26.3. Сравнение нескольких выборок

Здесь ставится задача сравнить математические ожидания и дисперсии нескольких выборок. Это задание выполняется в *разделах 22.3 и 22.4*. Исходные данные для каждого студента здесь — несколько нормально распределенных выборок.

Программа для генерации этого ИДЗ отличается от программы предыдущего раздела только тем, что количество выборок может случайным образом ме-

няться от 4 до 8. Каждая выборка помещается в файл студента в отдельный столбец.

```
clear all % очислили рабочую область
filegr='D:\Iglin\Matlab\ContData\ПБ-82.txt'; % файл группы
n=100; % задаем объем выборки
nametask='Сравнение нескольких выборок';
gr=textread(filegr,'%s','delimiter','\n'); % ввели данные
ngr=length(gr); % число студентов группы
bsl=findstr(filegr,'\'); % ищем вхождения символа \
dirgr=filegr(1:bsl(end)); % папка с файлом группы
nameextgr=filegr(bsl(end)+1:end); % имя файла группы
p=findstr(nameextgr,'.'); % ищем расширение
namegr=nameextgr(1:p(1)-1); % название группы
[succ,mess,messid]=mkdir(dirgr,...
  [namegr '-' nametask]); % создаем папку
if ~succ, % выход при ошибке
  error(['Не удалось создать папку ' dirstr nsmegr]);
end
newdirgr=[dirgr namegr '-' nametask '\'];
delete([newdirgr '*.*']); % удалили все из папки
fileans=[newdirgr 'Ответы.txt']; % имя файла ответов
fidans=fopen(fileans,'w'); % открыли файл с ответами
for k=1:ngr, % генерируем ИДЗ
  nd=unidrnd(5)+3; % случайное количество выборок от 4 до 8
  mu=unifrnd(1.2,2.2,1,nd); % средние
  sigma=unifrnd(4,8,1,nd); % СКО
  fprintf(fidans,'%d. %s\nКоличество выборок %d\n',k,gr{k},nd);
  fprintf(fidans,'mu%d=%14.8f;  sigma%d=%14.8f;\n',...
    [1:nd;mu;1:nd;sigma]);
  fprintf('%d. %s\nКоличество выборок %d\n',k,gr{k},nd);
  fprintf('mu%d=%14.8f;  sigma%d=%14.8f;\n',...
    [1:nd;mu;1:nd;sigma]);
  x=[]; % генерируем выборки
  stre='fprintf(fid,''''; % для печати
  for kd=1:nd,
    x=[x,normrnd(mu(kd),sigma(kd),n,1)];
    stre=[stre '%14.8f  '];
  end
  stre=[stre '\n'',x'');'];
  filestud=[newdirgr gr{k} '.txt'];  % имя файла студента
  fid=fopen(filestud,'w'); % открыли файл студента
  eval(stre); % выполнили команду печати
```

```
  fclose(fid); % закрыли файл студента
end
fclose(fidans); % закрыли файл с ответами
1. Аланьев Георгий Алексеевич
Количество выборок 8
mu1=    1.27782984;  sigma1=    6.13629403;
mu2=    1.53735967;  sigma2=    7.96492203;
mu3=    1.60850648;  sigma3=    5.26093767;
mu4=    1.62834036;  sigma4=    4.02969771;
mu5=    2.17169041;  sigma5=    4.11292217;
mu6=    1.99305860;  sigma6=    4.53305789;
mu7=    1.61304248;  sigma7=    6.37556722;
mu8=    2.19643493;  sigma8=    6.13334656;
2. Антонова Юлия Владиславовна
Количество выборок 4
mu1=    1.70886080;  sigma1=    7.78724146;
mu2=    1.37437779;  sigma2=    7.97953592;
mu3=    1.69733066;  sigma3=    5.21175616;
mu4=    1.83185112;  sigma4=    5.77151771;
3. Бережков Лев Иванович
Количество выборок 8
mu1=    1.77622599;  sigma1=    7.29623983;
mu2=    1.81866071;  sigma2=    7.58035121;
mu3=    1.99893532;  sigma3=    6.65378685;
mu4=    1.75205761;  sigma4=    5.05982062;
mu5=    1.81393058;  sigma5=    7.75325782;
mu6=    1.62198178;  sigma6=    7.81598229;
mu7=    2.11672707;  sigma7=    7.18615699;
mu8=    1.51721125;  sigma8=    6.36596790;
4. Шерванский Эраст Арнольдович
Количество выборок 7
mu1=    1.66945396;  sigma1=    5.00242936;
mu2=    1.88242546;  sigma2=    5.08533417;
mu3=    1.36776404;  sigma3=    4.11083873;
mu4=    1.61892133;  sigma4=    4.59881454;
mu5=    1.98185765;  sigma5=    4.84525025;
mu6=    1.30881356;  sigma6=    5.74918956;
mu7=    1.82259594;  sigma7=    5.43055824;
5. Щербатько Василиса Владимировна
Количество выборок 5
mu1=    2.15059479;  sigma1=    6.72982154;
mu2=    1.33573625;  sigma2=    7.82426083;
```

```
mu3=    1.74308905;  sigma3=    5.83479440;
mu4=    1.75993147;  sigma4=    6.71376726;
mu5=    1.96094530;  sigma5=    6.79389297;
```

Эти же исходные данные используются для ИДЗ по 1-факторному дисперсионному анализу *(см. раздел 23.1)*.

26.4. Двухфакторный дисперсионный анализ

В этом задании исходные данные — матрица экспериментов размером $k \times m$. Для каждого студента зададим случайные k, m, математические ожидания и дисперсии столбцов, выборки (сами столбцы), а затем еще транспонируем эту матрицу или не транспонируем в зависимости от случайного числа. Ответами здесь будут решения задачи 2-факторного дисперсионного анализа на уровне значимости $q = 0{,}1$.

```
clear all % очислили рабочую область
filegr='D:\Iglin\Matlab\ContData\ПБ-82.txt'; % файл группы
nametask='Двухфакторный дисперсионный анализ';
gr=textread(filegr,'%s','delimiter','\n'); % ввели данные
ngr=length(gr); % число студентов группы
bsl=findstr(filegr,'\'); % ищем вхождения символа \
dirgr=filegr(1:bsl(end)); % папка с файлом группы
nameextgr=filegr(bsl(end)+1:end); % имя файла группы
p=findstr(nameextgr,'.'); % ищем расширение
namegr=nameextgr(1:p(1)-1); % название группы
[succ,mess,messid]=mkdir(dirgr,...
  [namegr '-' nametask]); % создаем папку
if ~succ, % выход при ошибке
  error(['Не удалось создать папку ' dirstr nsmegr]);
end
newdirgr=[dirgr namegr '-' nametask '\'];
delete([newdirgr '*.*']); % удалили все из папки
fileans=[newdirgr 'Ответы.txt']; % имя файла ответов
fidans=fopen(fileans,'w'); % открыли файл с ответами
for k=1:ngr, % генерируем ИДЗ
  km=unidrnd(6,1,2)+8; % размеры матрицы
  mu=unifrnd(2,5,1,km(2)); % случайные средние столбцов
  sigma=unifrnd(2,4,1,km(2)); % случайные СКО
  x=[];
  for kd=1:km(2), % набираем статистику
    x=[x,normrnd(mu(kd),sigma(kd),km(1),1)];
  end
```

```
  if unidrnd(2)-1, % случайным образом транспонируем
    x=x';
  end
  q=0.1;
  p=anova2(x,1,'off'); % проверяем для ответа
  fprintf('%d. %s\n',k,gr{k});
  fprintf(fidans,'%d. %s\n',k,gr{k});
  if p(1)>=q,
    fprintf('Столбцы различаются незначимо, ');
    fprintf(fidans,'Столбцы различаются незначимо, ');
  else
    fprintf('Столбцы различаются значимо, ');
    fprintf(fidans,'Столбцы различаются значимо, ');
  end
  if p(2)>=q,
    fprintf('строки различаются незначимо.\n');
    fprintf(fidans,'строки различаются незначимо.\n');
  else
    fprintf('строки различаются значимо.\n');
    fprintf(fidans,'строки различаются значимо.\n');
  end
  m=size(x,2); % длина строки
  stre='fprintf(fid,''%14.8f  '; % для печати
  for k1=1:m,
    stre=[stre '%14.8f  '];
  end
  stre=[stre '\n'',x'');'];
  filestud=[newdirgr gr{k} '.txt'];  % имя файла студента
  fid=fopen(filestud,'w'); % открыли файл студента
  eval(stre); % выполнили команду печати
  fclose(fid); % закрыли файл студента
end
fclose(fidans); % закрыли файл с ответами
1. Аланьев Георгий Алексеевич
Столбцы различаются незначимо, строки различаются значимо.
2. Антонова Юлия Владиславовна
Столбцы различаются незначимо, строки различаются незначимо.
3. Бережков Лев Иванович
Столбцы различаются значимо, строки различаются значимо.
4. Шерванский Эраст Арнольдович
Столбцы различаются незначимо, строки различаются значимо.
5. Щербатько Василиса Владимировна
Столбцы различаются незначимо, строки различаются значимо.
```

26.5. Аппроксимация степенными полиномами

Это задание выполняется в *разделе 24.4*. Для него нужно задать аргументы и значения функции для построения по МНК теоретической кривой в виде степенного многочлена. Зададим одинаковые аргументы для всех студентов. Случайным образом сгенерируем степень и коэффициенты многочлена. Это будут ответы. А экспериментальные значения y_i получим, прибавляя к теоретическим ординатам случайное число с нормальным распределением. Дисперсию этого распределения тоже возьмем случайной, но одинаковой для всех опытов выборки.

```
clear all % очислили рабочую область
filegr='D:\Iglin\Matlab\ContData\ПБ-82.txt'; % файл группы
x=[-5:0.1:5]; % аргументы - одинаковые для всех
nametask='Аппроксимация степенными многочленами';
gr=textread(filegr,'%s','delimiter','\n'); % ввели данные
ngr=length(gr); % число студентов группы
bsl=findstr(filegr,'\'); % ищем вхождения символа \
dirgr=filegr(1:bsl(end)); % папка с файлом группы
nameextgr=filegr(bsl(end)+1:end); % имя файла группы
p=findstr(nameextgr,'.'); % ищем расширение
namegr=nameextgr(1:p(1)-1); % название группы
[succ,mess,messid]=mkdir(dirgr,...
  [namegr '-' nametask]); % создаем папку
if ~succ, % выход при ошибке
  error(['Не удалось создать папку ' dirstr nsmegr]);
end
newdirgr=[dirgr namegr '-' nametask '\'];
delete([newdirgr '*.*']); % удалили все из папки
fileans=[newdirgr 'Ответы.txt']; % имя файла ответов
fidans=fopen(fileans,'w'); % открыли файл с ответами
for k=1:ngr, % генерируем ИДЗ
  n=unidrnd(3); % степень аппроксимирующего полинома
  a=unifrnd(-3,3,1,n+1); % коэффициенты полинома
  fprintf('%d. %s\ny=',k,gr{k});
  fprintf('%+f14*x^%d',[a;n:-1:0]);
  fprintf('\n');
  fprintf(fidans,'%d. %s\ny=',k,gr{k});
  fprintf(fidans,'%+f14*x^%d',[a;n:-1:0]);
  fprintf(fidans,'\n');
  sigma=unifrnd(3,5); % случайная дисперсия
  y=polyval(a,x)+normrnd(0,sigma,size(x)); % ординаты
```

```
  filestud=[newdirgr gr{k} '.txt'];  % имя файла ответов
  fid=fopen(filestud,'w'); % открыли файл студента
  fprintf(fid,'%14.8f  %14.8f\n',[x;y]);
  fclose(fid); % закрыли файл студента
end
fclose(fidans); % закрыли файл с ответами
1. Аланьев Георгий Алексеевич
y=+2.78753414*x^3-0.37638914*x^2+1.28394214*x^1+1.59545014*x^0
2. Антонова Юлия Владиславовна
y=+2.32278814*x^1+1.44869514*x^0
3. Бережков Лев Иванович
y=+2.22077414*x^2-2.45450714*x^1+1.00527514*x^0
4. Шерванский Эраст Арнольдович
y=+2.70125014*x^1+2.38202314*x^0
5. Щербатько Василиса Владимировна
y=-0.86373614*x^2-2.29813614*x^1+2.03142714*x^0
```

26.6. Тригонометрическая аппроксимация

Задание по этой теме описано в *разделе 24.5*. Аргументы для него задаем в соответствии с (24.48). Случайным образом генерируем степень тригонометрического многочлена и его коэффициенты. Эти данные будут ответами задания. Экспериментальные значения y_i получим так же, как и в предыдущем разделе: прибавляя к теоретическим ординатам случайное число с нормальным распределением.

```
clear all % очислили рабочую область
filegr='D:\Iglin\Matlab\ContData\ПБ-82.txt'; % файл группы
x=[0.1:0.1:10]; % аргументы - одинаковые для всех
nametask='Тригонометрическая аппроксимация';
gr=textread(filegr,'%s','delimiter','\n'); % ввели данные
ngr=length(gr); % число студентов группы
bsl=findstr(filegr,'\'); % ищем вхождения символа \
dirgr=filegr(1:bsl(end)); % папка с файлом группы
nameextgr=filegr(bsl(end)+1:end); % имя файла группы
p=findstr(nameextgr,'.'); % ищем расширение
namegr=nameextgr(1:p(1)-1); % название группы
[succ,mess,messid]=mkdir(dirgr,...
  [namegr '-' nametask]); % создаем папку
if ~succ, % выход при ошибке
  error(['Не удалось создать папку ' dirstr nsmegr]);
end
```

```
newdirgr=[dirgr namegr '-' nametask '\'];
delete([newdirgr '*.*']); % удалили все из папки
T=x(end); % период
fileans=[newdirgr 'Ответы.txt']; % имя файла ответов
fidans=fopen(fileans,'w'); % открыли файл с ответами
for k=1:ngr, % генерируем ИДЗ
  n=unidrnd(3); % степень тригонометрического многочлена
  a0=unifrnd(-3,3,1,1); % средний уровень
  a=unifrnd(-3,3,1,n); % коэффициенты при cos
  b=unifrnd(-3,3,1,n); % коэффициенты при sin
  fprintf('%d. %s\ny=%f14',k,gr{k},a0);
  fprintf('%+f14*cos(2*pi*x*%d/%d)%+f14*sin(2*pi*x*%d/%d)',...
    [a;1:n;ones(1,n)*T;b;1:n;ones(1,n)*T]);
  fprintf('\n');
  fprintf(fidans,'%d. %s\ny=%f14',k,gr{k},a0);
  fprintf(fidans,...
    '%+f14*cos(2*pi*x*%d/%d)%+f14*sin(2*pi*x*%d/%d)',...
    [a;1:n;ones(1,n)*T;b;1:n;ones(1,n)*T]);
  fprintf(fidans,'\n');
  sigma=unifrnd(1,2.5); % случайная дисперсия
  y=ones(size(x))*a0; % средний уровень
  for kg=1:n, % добавляем гармоники
    y=y+a(kg)*cos(2*pi*x*kg/T)+b(kg)*sin(2*pi*x*kg/T);
  end
  y=y+normrnd(0,sigma,size(x)); % ординаты
  filestud=[newdirgr gr{k} '.txt'];  % имя файла ответов
  fid=fopen(filestud,'w'); % открыли файл студента
  fprintf(fid,'%14.8f  %14.8f\n',[x;y]);
  fclose(fid); % закрыли файл студента
end
fclose(fidans); % закрыли файл с ответами
```

```
1. Аланьев Георгий Алексеевич
y=0.67321214+1.67115114*cos(2*pi*x*1/10)-0.25125714*sin(2*pi*x*1/10)+
0.74708614*cos(2*pi*x*2/10)+2.30111514*sin(2*pi*x*2/10)-
1.97371814*cos(2*pi*x*3/10)-1.24187314*sin(2*pi*x*3/10)
2. Антонова Юлия Владиславовна
y=-0.15124514+2.55006614*cos(2*pi*x*1/10)-1.69440614*sin(2*pi*x*1/10)-
0.42559314*cos(2*pi*x*2/10)-1.41091314*sin(2*pi*x*2/10)+
0.31950214*cos(2*pi*x*3/10)-1.21734914*sin(2*pi*x*3/10)
3. Бережков Лев Иванович
y=1.57053014-0.33977314*cos(2*pi*x*1/10)+2.65664114*sin(2*pi*x*1/10)-
1.41876114*cos(2*pi*x*2/10)+1.38578314*sin(2*pi*x*2/10)
4. Шерванский Эраст Арнольдович
y=0.75154614+1.87081914*cos(2*pi*x*1/10)-1.09510214*sin(2*pi*x*1/10)+
2.05781714*cos(2*pi*x*2/10)-0.48206614*sin(2*pi*x*2/10)
```

```
5. Щербатько Василиса Владимировна
y=-1.06864914-0.84722614*cos(2*pi*x*1/10)-1.20981914*sin(2*pi*x*1/10)-
0.05331214*cos(2*pi*x*2/10)+0.72651414*sin(2*pi*x*2/10)+
2.50423514*cos(2*pi*x*3/10)-0.52513314*sin(2*pi*x*3/10)
```

26.7. Линейная функция двух переменных

Целью этого ИДЗ является построение линейной модели для 2-факторного эксперимента. Выполнение этого задания — в *разделе 24.6.1*. Экспериментальные данные z_{ij} задаются на прямоугольной сетке $l \times m$. Принцип работы этого генератора заданий такой же, как и в предыдущих разделах. Задаем случайные размеры l и m. Значения аргументов берем с одинаковым шагом 1 по каждой переменной. Генерируем случайные параметры линейной модели (это будет ответ). На сетке $l \times m$ вычисляем отклик линейной модели z_{ij}, который зашумляем случайным нормальным сигналом. Эти данные печатаем в файл студента.

```
clear all % очислили рабочую область
filegr='D:\Iglin\Matlab\ContData\ПБ-82.txt'; % файл группы
nametask='Линейная функция двух переменных';
gr=textread(filegr,'%s','delimiter','\n'); % ввели данные
ngr=length(gr); % число студентов группы
bsl=findstr(filegr,'\'); % ищем вхождения символа \
dirgr=filegr(1:bsl(end)); % папка с файлом группы
nameextgr=filegr(bsl(end)+1:end); % имя файла группы
p=findstr(nameextgr,'.'); % ищем расширение
namegr=nameextgr(1:p(1)-1); % название группы
[succ,mess,messid]=mkdir(dirgr,...
  [namegr '-' nametask]); % создаем папку
if ~succ, % выход при ошибке
  error(['Не удалось создать папку ' dirstr nsmegr]);
end
newdirgr=[dirgr namegr '-' nametask '\'];
delete([newdirgr '*.*']); % удалили все из папки
fileans=[newdirgr 'Ответы.txt']; % имя файла ответов
fidans=fopen(fileans,'w'); % открыли файл с ответами
for k=1:ngr, % генерируем ИДЗ
  a=unifrnd(-3,3,1,3); % коэффициенты линейной модели
  fprintf('%d. %s\ny=',k,gr{k});
  fprintf('%f14%+f14*x%+f14*y\n',a);
  fprintf(fidans,'%d. %s\ny=',k,gr{k});
  fprintf(fidans,'%f14%+f14*x%+f14*y\n',a);
  km=unidrnd(5,1,2)+13; % размеры матрицы
  x=[1:1:km(1)]; % аргументы x
```

```
  y=[1:1:km(2)]; % аргументы y
  [X,Y]=meshgrid(y,x); % сетка
  sigma=unifrnd(2,4); % случайная дисперсия
  z=a(1)+a(2)*X+a(2)*Y+normrnd(0,sigma,size(X)); % точки zij
  stre='fprintf(fid,''''; % для печати
  for k1=1:km(2)+1,
    stre=[stre '%14.8f  '];
  end
  stre=[stre '\n'',xyz'');'];
  xyz=[[0 y];[x' z]]; % массив для печати
  filestud=[newdirgr gr{k} '.txt'];  % имя файла студента
  fid=fopen(filestud,'w'); % открыли файл студента
  eval(stre); % выполнили команду печати
  fclose(fid); % закрыли файл студента
end
fclose(fidans); % закрыли файл с ответами
1. Аланьев Георгий Алексеевич
y=-2.77587014+2.61683014*x-2.18650314*y
2. Антонова Юлия Владиславовна
y=1.38989514-2.20145214*x+1.91724114*y
3. Бережков Лев Иванович
y=-1.40669114+0.78088314*x+0.82689414*y
4. Шерванский Эраст Арнольдович
y=1.14893614+0.83954214*x-0.13399814*y
5. Щербатько Василиса Владимировна
y=2.28697914+0.84285314*x+2.72443514*y
```

26.8. Кривая насыщения

Нелинейная зависимость от параметров аппроксимации была рассмотрена в *разделе 24.7*. Здесь генерируются данные для построения кривой насыщения (24.77). Аргументы задаются одинаковые во всех вариантах. Параметры кривой генерируются случайными, это будут ответы. Далее на кривую насыщения с этими параметрами накладывается нормальный шум. В файле студента — экспериментальные точки.

```
clear all % очислили рабочую область
filegr='D:\Iglin\Matlab\ContData\ПБ-82.txt'; % файл группы
nametask='Кривая насыщения';
x=[0:0.1:10]; % одинаковые аргументы
gr=textread(filegr,'%s','delimiter','\n'); % ввели данные
ngr=length(gr); % число студентов группы
bsl=findstr(filegr,'\'); % ищем вхождения символа \
```

```
dirgr=filegr(1:bsl(end)); % папка с файлом группы
nameextgr=filegr(bsl(end)+1:end); % имя файла группы
p=findstr(nameextgr,'.'); % ищем расширение
namegr=nameextgr(1:p(1)-1); % название группы
[succ,mess,messid]=mkdir(dirgr,...
  [namegr '-' nametask]); % создаем папку
if ~succ, % выход при ошибке
  error(['Не удалось создать папку ' dirstr nsmegr]);
end
newdirgr=[dirgr namegr '-' nametask '\'];
delete([newdirgr '*.*']); % удалили все из папки
fileans=[newdirgr 'Ответы.txt']; % имя файла ответов
fidans=fopen(fileans,'w'); % открыли файл с ответами
for k=1:ngr, % генерируем ИДЗ
  a=unifrnd([1 0.2],[3 1]); % параметры модели
  fprintf('%d. %s\ny=',k,gr{k});
  fprintf('%f14*(1-exp(%+f14*x))\n',a(1),-a(2));
  fprintf(fidans,'%d. %s\ny=',k,gr{k});
  fprintf(fidans,'%f14*(1-exp(%+f14*x))\n',a(1),-a(2));
  sigma=unifrnd(0.1,0.15); % случайная дисперсия
  y=a(1)*(1-exp(-a(2)*x))+normrnd(0,sigma,size(x)); % ординаты
  filestud=[newdirgr gr{k} '.txt'];  % имя файла ответов
  fid=fopen(filestud,'w'); % открыли файл студента
  fprintf(fid,'%14.8f  %14.8f\n',[x;y]);
  fclose(fid); % закрыли файл студента
end
fclose(fidans); % закрыли файл с ответами
1. Аланьев Георгий Алексеевич
y=1.13780214*(1-exp(-0.39015014*x))
2. Антонова Юлия Владиславовна
y=2.49025314*(1-exp(-0.26978514*x))
3. Бережков Лев Иванович
y=1.71927614*(1-exp(-0.61090714*x))
4. Шерванский Эраст Арнольдович
y=1.45824414*(1-exp(-0.82538814*x))
5. Щербатько Василиса Владимировна
y=1.98322014*(1-exp(-0.85222014*x))
```

26.9. Корреляционный анализ

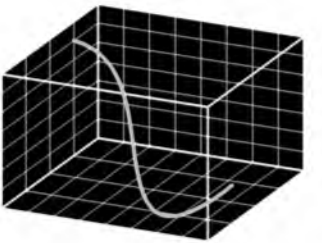
Это ИДЗ выполняется в *главе 25*. Исходными данными для него являются 2 столбца, и требуется проверить значимость коэффициента корреляции. Сгенерируем данные в виде 2-мерного нормального распределения со слу-

чайными параметрами, причем коэффициент корреляции возьмем малым, чтобы величины могли получиться и коррелированными, и некоррелированными. В качестве ответа здесь удобнее взять результат выполнения ИДЗ.

```
clear all % очислили рабочую область
filegr='D:\Iglin\Matlab\ContData\ПБ-82.txt'; % файл группы
nametask='Корреляционный анализ';
n=500; % число точек
gr=textread(filegr,'%s','delimiter','\n'); % ввели данные
ngr=length(gr); % число студентов группы
bsl=findstr(filegr,'\'); % ищем вхождения символа \
dirgr=filegr(1:bsl(end)); % папка с файлом группы
nameextgr=filegr(bsl(end)+1:end); % имя файла группы
p=findstr(nameextgr,'.'); % ищем расширение
namegr=nameextgr(1:p(1)-1); % название группы
[succ,mess,messid]=mkdir(dirgr,...
  [namegr '-' nametask]); % создаем папку
if ~succ, % выход при ошибке
  error(['Не удалось создать папку ' dirstr nsmegr]);
end
newdirgr=[dirgr namegr '-' nametask '\'];
delete([newdirgr '*.*']); % удалили все из папки
q=0.1; % уровень значимости
fileans=[newdirgr 'Ответы.txt']; % имя файла ответов
fidans=fopen(fileans,'w'); % открыли файл с ответами
for k=1:ngr, % генерируем ИДЗ
  mu=unifrnd([-1;1],[1;2]); % математические ожидания
  rho=unifrnd(-0.2,0.2); % коэффициент корреляции
  sigma=[unifrnd(1,2) rho; rho unifrnd(1.5,2)]; % дисперсии
  xy=mvnrnd(mu,sigma,n); % набираем статистику
  [r,p]=corrcoef(xy); % решаем задачу
  fprintf('%d. %s\n',k,gr{k});
  fprintf(fidans,'%d. %s\n',k,gr{k});
  if p(1,2)<q,
    disp('Корреляция значима.')
    fprintf(fidans,'Корреляция значима.');
  else
    disp('Корреляция незначима.')
    fprintf(fidans,'Корреляция незначима.');
  end
  filestud=[newdirgr gr{k} '.txt'];  % имя файла ответов
  fid=fopen(filestud,'w'); % открыли файл студента
  fprintf(fid,'%14.8f  %14.8f\n',xy');
```

```
  fclose(fid); % закрыли файл студента
end
fclose(fidans); % закрыли файл с ответами
1. Аланьев Георгий Алексеевич
Корреляция значима.
2. Антонова Юлия Владиславовна
Корреляция незначима.
3. Бережков Лев Иванович
Корреляция значима.
4. Шерванский Эраст Арнольдович
Корреляция незначима.
5. Щербатько Василиса Владимировна
Корреляция значима.
```

ГЛАВА 27



Функции пакета Statistics Toolbox

Для удобства читателей приведем список и краткое описание функций пакета Statistics Toolbox. Классификация функций пакета (заголовки разделов данной главы) дается в соответствии со справочной информацией, которую можно получить, набрав в командном окне MATLAB команду `help stats`. Полное описание этого инструментария есть в официальной справочной документации на английском языке [57]. На русском языке некоторые функции пакета описаны в [16, 36, 42].

27.1. Функции распределения

Они вычисляют функции распределения $F(x)$ для различных законов вероятностей. Все имена функций имеют суффикс *cdf (сокращение от cumulative distribution function — функция распределения).

- **betacdf** — β-распределение;
- **binocdf** — биномиальное распределение;
- **chi2cdf** — χ^2-распределение Пирсона;
- **expcdf** — экспоненциальное (показательное) распределение;
- **fcdf** — F-распределение Фишера;
- **gamcdf** — γ-распределение;
- **geocdf** — геометрическое распределение;
- **hygecdf** — гипергеометрическое распределение;
- **logncdf** — логнормальное распределение;
- **nbincdf** — отрицательное биномиальное распределение;
- **ncfcdf** — смещенное F-распределение Фишера;

- **nctcdf** — смещенное *t*-распределение Стьюдента;
- **ncx2cdf** — смещенное χ^2-распределение Пирсона;
- **normcdf** — нормальное распределение;
- **poisscdf** — пуассоновское распределение;
- **raylcdf** — рэлеевское распределение;
- **tcdf** — *t*-распределение Стьюдента;
- **unidcdf** — дискретное равномерное распределение;
- **unifcdf** — непрерывное равномерное распределение;
- **weibcdf** — распределение Вейбулла;
- **cdf** — выбранное из списка распределение;
- **ecdf** — эмпирическая функция распределения по Каплану — Мейеру.

27.2. Плотности распределения

В этих функциях вычисляются плотности распределения $f(x)$ для различных типов распределений. Все имена функций имеют суффикс *pdf (сокращение от probability density function — плотность распределения).

- **betapdf** — β-распределение;
- **binopdf** — биномиальное распределение;
- **chi2pdf** — χ^2-распределение Пирсона;
- **exppdf** — экспоненциальное распределение;
- **fpdf** — *F*-распределение Фишера;
- **gampdf** — γ-распределение;
- **geopdf** — геометрическое распределение;
- **hygepdf** — гипергеометрическое распределение;
- **lognpdf** — логнормальное распределение;
- **mvnpdf** — многомерное нормальное распределение;
- **nbinpdf** — отрицательное биномиальное распределение;
- **ncfpdf** — смещенное *F*-распределение Фишера;
- **nctpdf** — смещенное *t*-распределение Стьюдента;
- **ncx2pdf** — смещенное χ^2-распределение Пирсона;
- **normpdf** — нормальное распределение;
- **poiss3df** — пуассоновское распределение;

- **raylpdf** — рэлеевское распределение;
- **tpdf** — *t*-распределение Стьюдента;
- **unidpdf** — дискретное равномерное распределение;
- **unifpdf** — непрерывное равномерное распределение;
- **weibpdf** — распределение Вейбулла;
- **pdf** — выбранное из списка распределение.

27.3. Квантили распределения

В этих функциях вычисляются критические значения, или квантили, или обратная функция распределения для различных законов. Все имена функций имеют суффикс *inv (сокращение от inverse cumulative distribution function — обратная функция распределения).

- **betainv** — β-распределение;
- **binoinv** — биномиальное распределение;
- **chi2inv** — χ^2-распределение Пирсона;
- **expinv** — экспоненциальное распределение;
- **finv** — *F*-распределение Фишера;
- **gaminv** — γ-распределение;
- **geoinv** — геометрическое распределение;
- **hygeinv** — гипергеометрическое распределение;
- **logninv** — логнормальное распределение;
- **nbininv** — отрицательное биномиальное распределение;
- **ncfinv** — смещенное *F*-распределение Фишера;
- **nctinv** — смещенное *t*-распределение Стьюдента;
- **ncx2inv** — смещенное χ^2-распределение Пирсона;
- **norminv** — нормальное распределение;
- **poissinv** — пуассоновское распределение;
- **raylinv** — рэлеевское распределение;
- **tinv** — *t*-распределение Стьюдента;
- **unidinv** — дискретное равномерное распределение;
- **unifinv** — непрерывное равномерное распределение;
- **weibinv** — распределение Вейбулла;
- **icdf** — выбранное из списка распределение.

27.4. Генераторы случайных чисел

Имена этих функций имеют суффикс *rnd. Они генерируют случайные числа, имеющие соответствующий закон распределения.

- **betarnd** — β-распределение;
- **binornd** — биномиальное распределение;
- **chi2rnd** — χ^2-распределение Пирсона;
- **exprnd** — экспоненциальное распределение;
- **frnd** — *F*-распределение Фишера;
- **gamrnd** — γ-распределение;
- **geornd** — геометрическое распределение;
- **hygernd** — гипергеометрическое распределение;
- **iwishrnd** — обратное распределение Уишарта;
- **lognrnd** — логнормальное распределение;
- **mvnrnd** — многомерное нормальное распределение;
- **mvtrnd** — многомерное *t*-распределение Стьюдента;
- **nbinrnd** — отрицательное биномиальное распределение;
- **ncfrnd** — смещенное *F*-распределение Фишера;
- **nctrnd** — смещенное *t*-распределение Стьюдента;
- **ncx2rnd** — смещенное χ^2-распределение Пирсона;
- **normrnd** — нормальное распределение;
- **poissrnd** — пуассоновское распределение;
- **raylrnd** — рэлеевское распределение;
- **trnd** — *t*-распределение Стьюдента;
- **unidrnd** — дискретное равномерное распределение;
- **unifrnd** — непрерывное равномерное распределение;
- **weibrnd** — распределение Вейбулла;
- **wishrnd** — распределение Уишарта;
- **random** — выбранный из списка закон распределения.

27.5. Статистические характеристики

Функции этой группы вычисляют математические ожидания и дисперсии различных распределений. Имена этих функций имеют суффикс *stat.

- **betastat** — β-распределение;
- **binostat** — биномиальное распределение;
- **chi2stat** — χ^2-распределение Пирсона;
- **expstat** — экспоненциальное распределение;
- **fstat** — F-распределение Фишера;
- **gamstat** — γ-распределение;
- **geostat** — геометрическое распределение;
- **hygestat** — гипергеометрическое распределение;
- **lognstat** — логнормальное распределение;
- **nbinstat** — отрицательное биномиальное распределение;
- **ncfstat** — смещенное F-распределение Фишера;
- **nctstat** — смещенное t-распределение Стьюдента;
- **ncx2stat** — смещенное χ^2-распределение Пирсона;
- **normstat** — нормальное распределение;
- **poisstat** — пуассоновское распределение;
- **raylstat** — рэлеевское распределение;
- **tstat** — t-распределение Стьюдента;
- **unidstat** — дискретное равномерное распределение;
- **unifstat** — непрерывное равномерное распределение;
- **weibstat** — распределение Вейбулла.

27.6. Подбор параметров

Функции с суффиксом *fit по заданной выборке и заданному виду теоретического распределения подбирают максимально правдоподобные оценки параметров теоретического распределения и находят доверительные интервалы для них.

- **betafit** — β-распределение;
- **binofit** — биномиальное распределение;
- **expfit** — экспоненциальное распределение;
- **gamfit** — γ-распределение;
- **nbinfit** — отрицательное биномиальное распределение;
- **normfit** — нормальное распределение;

- **poissfit** — пуассоновское распределение;
- **raylfit** — рэлеевское распределение;
- **unifit** — непрерывное равномерное распределение;
- **weibfit** — распределение Вейбулла;
- **mle** — выбранный из списка закон распределения.

27.7. Функции правдоподобия

В функциях с суффиксом *like по заданной выборке, заданному виду теоретического распределения и заданным значениям параметров вычисляется взятый со знаком минус логарифм функции правдоподобия.

- **betalike** — β-распределение;
- **gamlike** — γ-распределение;
- **nbinlike** — отрицательное биномиальное распределение;
- **normlike** — нормальное распределение;
- **weiblike** — распределение Вейбулла.

27.8. Функции описательной статистики

- **bootstrp** — оценка статистик для данных с дополненным объемом выборки посредством математического моделирования;
- **corrcoef** — коэффициент корреляции (функция MATLAB);
- **cov** — ковариационная матрица (функция MATLAB);
- **crosstab** — перекрестная табуляция вектор-столбцов;
- **geomean** — среднее геометрическое;
- **grpstats** — сводные статистики по группам;
- **harmmean** — среднее гармоническое;
- **iqr** — разность между 75 и 25% квантилями;
- **kurtosis** — эксцесс без вычитания числа 3;
- **mad** — среднее абсолютное отклонение от среднего значения;
- **mean** — среднее арифметическое (функция MATLAB);
- **median** — медиана (функция MATLAB);
- **moment** — центральный момент заданного порядка;
- **nanmax** — максимальное значение без учета нечисловых значений;

- **nanmean** — среднее арифметическое без учета нечисловых значений;
- **nanmedian** — медиана без учета нечисловых значений;
- **nanmin** — минимальное значение без учета нечисловых значений;
- **nanstd** — среднеквадратичное отклонение без учета нечисловых значений;
- **nansum** — сумма элементов без учета нечисловых значений;
- **pretile** — выборочный процентиль (квантиль в %);
- **range** — размах выборки;
- **skewness** — асимметрия;
- **std** — среднеквадратичное отклонение (функция MATLAB);
- **tabulate** — определение частот целых положительных элементов вектора;
- **trimmean** — среднее арифметическое с игнорированием заданного процента минимальных и максимальных элементов;
- **var** — дисперсия (функция MATLAB).

27.9. Статистическая графика

- **boxplot** — график распределения данных по 0, 25, 50, 75 и 100% выборочным квантилям;
- **cdfplot** — график эмпирической функции распределения;
- **fsurfht** — интерактивное построение контурного графика заданной функции;
- **gline** — интерактивное рисование прямой линии на графике;
- **gname** — интерактивное рисование меток на графике;
- **gplotmatrix** — график матрицы рассеяния с группировкой переменной;
- **gseatter** — график рассеяния с группировкой переменной;
- **lsline** — добавляет на график рассеяния линию регрессии, построенную по методу наименьших квадратов;
- **normplot** — график нормальной функции распределения;
- **qqplot** — график "квантиль-квантиль" для двух выборок;
- **refcurve** — добавление полиномиальной кривой на текущий график;
- **refline** — добавление прямой на текущий график;
- **surfht** — интерактивное построение контура по матрице данных;
- **weibplot** — график функции распределения Вейбулла.

27.10. Статистическое управление процессами

- **capable** — расчет вероятности выхода процесса за заданный уровень;
- **capaplot** — рисует соответствующий график;
- **ewmaplot** — карта скользящего среднего для взвешенного экспоненциального распределения;
- **histfit** — гистограмма с подобранной плотностью нормального распределения;
- **normspec** — график плотности нормального распределения в заданных границах;
- **schart** — карта для контроля среднеквадратичного отклонения;
- **xbarplot** — карта для контроля среднего арифметического.

27.11. Линейные модели

- **anova1** — 1-факторный дисперсионный анализ;
- **anova2** — 2-факторный дисперсионный анализ;
- **anovan** — многофакторный дисперсионный анализ;
- **aoctool** — 1-факторный ковариационный анализ;
- **dummyvar** — условное кодирование переменных;
- **friedman** — тест Фридмана (непараметрический 2-факторный дисперсионный анализ);
- **glmfit** — определение параметров обобщенной линейной модели;
- **glmval** — вычисление значений с использованием обобщенной линейной модели;
- **kruskalwallis** — тест Краскала — Уоллиса (непараметрический 1-факторный дисперсионный анализ);
- **leverage** — регрессионная диагностика: оценка влияния каждого наблюдения на значения параметров регрессии;
- **lscov** — метод наименьших квадратов при заданной ковариационной матрице (функция MATLAB);
- **manova1** — однофакторный многомерный дисперсионный анализ;
- **manovacluster** — кластерный анализ (группировка данных) по результатам однофакторного многомерного дисперсионного анализа (manova1);

- **multcompare** — множественное сравнение средних и других оценок;
- **polyconf** — определение доверительных интервалов для ординат линии регрессии;
- **polyfit** — полиномиальная регрессия (функция MATLAB);
- **polyval** — вычисление значений с использованием полиномиальной регрессии (функция MATLAB);
- **rcoplot** — график остатков;
- **regress** — множественная линейная регрессия с использованием метода наименьших квадратов;
- **regstats** — регрессионная диагностика для линейной модели;
- **ridge** — линейная регрессия с применением гребневых оценок;
- **robustfit** — робастная линейная регрессия;
- **rstool** — интерактивный подбор и визуализация многомерной поверхности отклика;
- **stepwise** — интерактивная пошаговая регрессия;
- **x2fx** — преобразование матрицы входов системы в проектной матрице для регрессионного анализа.

27.12. Нелинейные модели

- **lsqnonneg** — метод наименьших квадратов для неотрицательных значений параметров модели (функция MATLAB);
- **nlinfit** — нелинейный метод наименьших квадратов по методу Гаусса — Ньютона;
- **nlintool** — интерактивное построение нелинейной модели;
- **nlparci** — доверительные интервалы для параметров нелинейной модели;
- **nlpredci** — прогнозируемые значения и их доверительные интервалы.

27.13. Планирование эксперимента

- **bbdesign** — планы Бокса — Бенкена;
- **candexch** — *D*-оптимальный план на основе перестановки строк;
- **candgen** — генерирует множество возможных сочетаний факторов соответствующих *D*-оптимальному плану;
- **ccdesign** — центральный композиционный план;

- **cordexch** — точный *D*-оптимальный план на основе алгоритма обмена координатами;
- **daugment** — дополнение матрицы заданного плана до *D*-оптимального;
- **dcovary** — *D*-оптимальный план с некоторыми фиксированными данными;
- **ff2n** — полный факторный эксперимент для факторов с 2 уровнями;
- **fracfact** — дробный факторный эксперимент для факторов с 2 уровнями;
- **fullfact** — многоуровневый полный факторный эксперимент;
- **hadamard** — матрица Адамара (функция MATLAB);
- **lhsdesign** — план на основе латинских квадратов;
- **lhsnorm** — латинские квадраты для нормального распределения;
- **rowexch** — *D*-оптимальный план на основе алгоритма обмена строк.

27.14. Кластерный анализ

- **cluster** — строит кластеры из выходных данных функции linkage;
- **clusterdata** — разбивает данные на кластеры;
- **cophenet** — коэффициент качества разбиения исходных данных на кластеры;
- **dendrogram** — строит график дендрограммы кластеров;
- **inconsistent** — вычисляет несовместимые значения в дереве кластеров;
- **kmeans** — кластеризация на основе внутригрупповых средних;
- **linkage** — строит иерархическое дерево кластеров;
- **pdist** — находит расстояния между всеми парами точек;
- **silhouette** — строит график силуэта кластеров;
- **squareform** — построение квадратной матрицы расстояний.

27.15. Понижение размерности задач

- **factoran** — общий факторный анализ на основе принципа максимума правдоподобия;
- **pcacov** — анализ главных компонент по ковариационной матрице;
- **pcares** — остаток после удаления главных компонент;
- **princomp** — анализ главных компонент с центрированием и масштабированием данных.

27.16. Многомерный анализ данных

- **barttest** — тест Бартлета;
- **canoncorr** — канонический корреляционный анализ;
- **cmdscale** — классическое многомерное масштабирование;
- **classify** — дискриминантный анализ;
- **mahal** — расстояния Махаланобиса;
- **manova1** — однофакторный многомерный дисперсионный анализ;
- **procrustes** — прокрустов анализ (перенос, отражение, вращение, масштабирование).

27.17. Анализ на основе дерева возможных решений

- **treedisp** — отображает классификацию или дерево возможных решений;
- **treefit** — строит дерево возможных решений;
- **treeprune** — строит последовательность поддеревьев путем подрезки;
- **treetest** — оценка погрешности в дереве возможных решений;
- **treeval** — вычисляет значения данных на дереве решений.

27.18. Проверка статистических гипотез

- **ranksum** — ранговый тест Вилкоксона для проверки однородности двух генеральных совокупностей;
- **signrank** — знаковый тест Вилкоксона для проверки гипотезы о равенстве медиан двух выборок;
- **signtest** — знаковый тест для проверки гипотезы о равенстве медиан двух выборок;
- **ttest** — проверка гипотезы о равенстве математического ожидания выборки заданному значению при неизвестной дисперсии;
- **ttest2** — проверка гипотезы о равенстве математических ожиданий двух выборок при неизвестных дисперсиях;
- **ztest** — *Z*-тест: проверка гипотезы о равенстве (или неравенстве) математического ожидания выборки заданному значению при известной дисперсии.

27.19. Проверка теоретического распределения

- **jbtest** — тест Бера — Жарка на соответствие выборки нормальному распределению с неопределенными параметрами;
- **kstest** — тест Колмогорова — Смирнова на соответствие одной выборки заданному распределению;
- **kstest2** — тест Колмогорова — Смирнова на соответствие распределений двух выборок;
- **lllietest** — тест на соответствие выборки нормальному распределению, параметры которого находятся по выборке.

27.20. Функции непараметрической статистики

- **friedman** — тест Фридмана (непараметрический 2-факторный дисперсионный анализ);
- **kruskalwallis** — тест Краскала — Уоллиса (непараметрический 1-факторный дисперсионный анализ);
- **ksdensity** — вычисляет оценку плотности распределения по данным;
- **ranksum** — ранговый тест Вилкоксона для проверки однородности двух генеральных совокупностей;
- **signrank** — знаковый тест Вилкоксона для проверки гипотезы о равенстве медиан двух выборок;
- **signtest** — знаковый тест для проверки гипотезы о равенстве медиан двух выборок.

27.21. Демонстрационные примеры

- **aoctool** — 1-факторный ковариационный анализ;
- **disttool** — демонстрация различных функций и плотностей распределения;
- **glmdemo** — демонстрация построения линейной модели;
- **randtool** — демонстрация различных генераторов случайных чисел;
- **polytool** — интерактивное определение параметров полиномиальной модели;

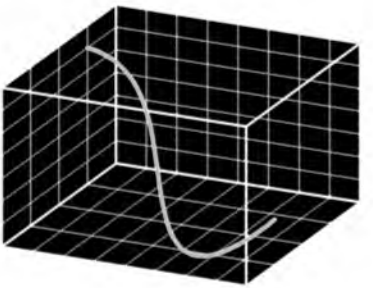
- **rsmdemo** — интерактивное моделирование нелинейной регрессии на примере исследования химической реакции;
- **robustdemo** — демонстрирует разницу между обычным регрессионным анализом на основе метода наименьших квадратов и робастным анализом.

27.22. Функции ввода-вывода

- **caseread** — читает текстовые имена полей данных из текстового файла;
- **casewrite** — записывает текстовые имена полей данных в текстовый файл;
- **tblread** — ищет табличные данные в файловой системе;
- **tblwrite** — записывает табличные данные в файловую систему;
- **tdfread** — ищет табличные данные в файловой системе.

27.23. Вспомогательные функции

- **combnk** — вычисляет все возможные выборки заданного объема из заданного вектора;
- **grp2idx** — создает индексный вектор по сгруппированным данным;
- **hougen** — модель Хогена — Ватсона для кинетики реакций;
- **tiedrank** — расчет ранга выборки с учетом ее объема;
- **zscore** — нормализация матрицы данных по столбцам.



ЧАСТЬ III

Теория графов

Глава 28. Графы и орграфы

Глава 29. Больше ребер, меньше вершин

Глава 30. Жадные алгоритмы и минимальные остовные деревья

Глава 31. Базис в пространстве циклов

Глава 32. Правильная раскраска вершин

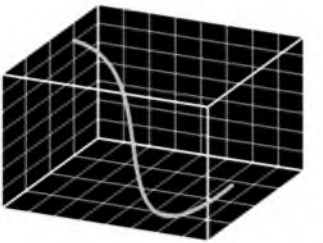
Глава 33. Кратчайший путь

Глава 34. Разбиваем и упорядочиваем

Глава 35. Максимальный поток в сети

Глава 36. Задача коммивояжера

ГЛАВА 28



Графы и орграфы

Переходим к третьей, заключительной части нашей книги. Она посвящена теории графов. На уровне наших обыденных представлений граф — это любые объекты (обычно их рисуют в виде точек или кружков), соединенные линиями или стрелками (рис. 28.1). В первом случае граф называется неориентированным (просто граф), а во втором — ориентированным (орграф).

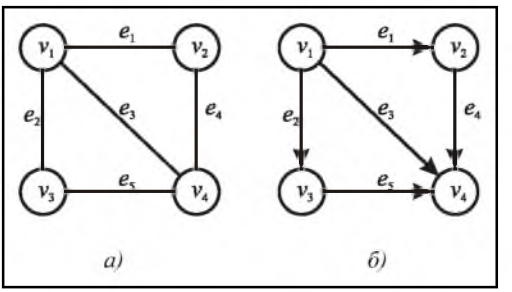


Рис. 28.1. Графы (граф и орграф): *а* — неориентированный; *б* — ориентированный

В частности, в виде графов можно представить электрические схемы, маршруты перевозок, схемы взаимосвязи подразделений предприятия, денежные и ресурсные потоки, системы управления различными объектами и т. д. Издавна люди заметили, что графы обладают общими свойствами независимо от того, какой реальный объект они представляют. На основе изучения этих свойств и возникла наука под названием "Теория графов". В процессе ее развития выяснилось, что она тесно связана с другими разделами математики: теорией множеств и комбинаторным анализом. Поэтому в технических вузах обычно теорию графов наряду с теорией множеств, комбинаторикой, топологией и некоторыми другими разделами математики изучают в курсе под названием "Дискретная математика".

Для решения задач теории графов создано множество алгоритмов и программ. Некоторые из них описаны в [2, 33] и других книгах. Однако создатели MATLAB почему-то обошли стороной этот вопрос: в нем нет процедур для решения задач на графах. На сайте компании [51] в коллекции файлов для обмена отсутствует даже такая категория. Пользователи MATLAB пытаются устранить этот недостаток. Так, в [52, 56] приведены решения некоторых задач. В этой части книги мы рассмотрим некоторые задачи на графах, в том числе и те, которых нет в [52, 56]. Часть из них формулируется как задачи целочисленного линейного программирования (ЦЛП), и для их решения применяются готовые процедуры, которые можно свободно переписать с сайтов [52, 56, 60]. Для решения других задач используются алгоритмы, описанные в [33]. И наконец, отдельные задачи решается на основе оригинальных алгоритмов, разработанных автором. Возможно, эти алгоритмы не являются оптимальными по быстродействию, но они очень просты в реализации и требуют написания минимального количества программного кода.

Все описанные в этой части книги процедуры объединены в инструментарий GrTheory Toolbox, который доступен для свободного копирования по адресу [51], страница "Математика — общие вопросы".

28.1. Основные определения теории графов

Дадим некоторые определения теории графов с точки зрения теории множеств. Это — наиболее общий подход к изучению графов и их свойств.

Определение 28.1. *Граф* G — это совокупность двух множеств V и E:

$$G=(V,E), \tag{28.1}$$

где V — множество (оно называется основным), а E — множество двухэлементных подмножеств множества V. □

Определение 28.2. Элементы основного множества V называются *вершинами*. □

Будем обозначать вершины графа $v_1, v_2, ..., v_n$ (от англ. vertex — вершина).

Определение 28.3. Количество вершин графа G (мощность множества V) называется *размером* графа. □

Размер графа обычно обозначается буквой n: $|V|=n$. Здесь функция $|...|$ — количество элементов множества.

Определение 28.4. Элементы множества E называются *ребрами*. □

Как правило, ребра обозначают буквами $e_1, e_2, ..., e_m$ (от англ. edge — ребро).

Определение 28.5. Количество ребер графа G (мощность множества E) называется *мощностью* графа. □

Мощность графа обычно обозначается m: $|E| = m$.

По основным аксиомам теории множеств одинаковые элементы множества E считаются одним элементом, поэтому в соответствии с определением 28.1 кратных ребер в графе не может быть. По этим же аксиомам в каждом ребре должны быть две различные вершины, т. к. две одинаковые вершины — это все равно, что одна вершина, а элементы множества E обязательно должны быть двухэлементными подмножествами множества V. Поэтому петель в графе по определению 28.1 тоже нет.

ПРИМЕР 28.1. С точки зрения теории множеств граф, изображенный на рис. 28.1, *а* — это совокупность множеств $V = \{v_1, v_2, v_3, v_4\}$ и $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_4\}, \{v_3, v_4\}\}$. Здесь $n = 4$, $m = 5$. Порядок нумерации вершин (элементов множества V) не имеет значения, т. к. элементы множества считаются неупорядоченными. Точно так же не имеет значения порядок нумерации ребер (элементов множества E) и вершин в ребре (элементов множеств e_1, e_2, e_3, e_4, e_5). Поэтому любой граф по определению 28.1 — неориентированный. Для графа с рис. 28.1, *а* множества V, E и их элементы показаны на рис. 28.2. □

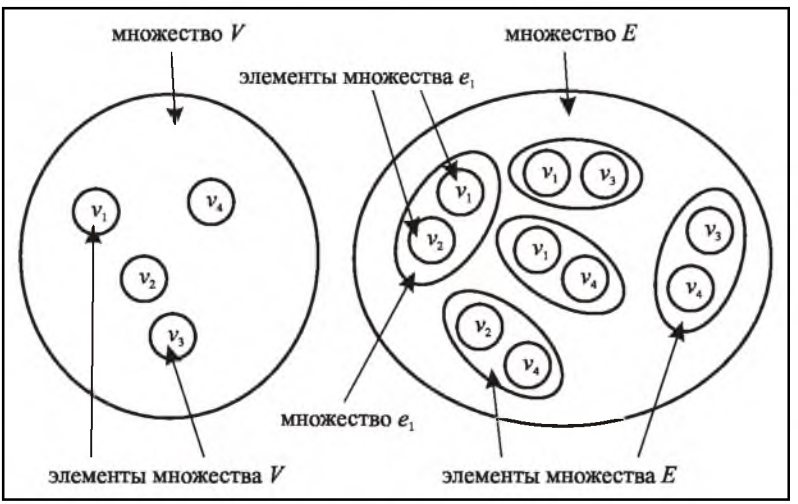


Рис. 28.2. Граф $G = (V, E)$ и его множества V и E

Но реальная жизнь шире определения 28.1. Иногда приходится рассматривать графы с кратными ребрами и петлями.

Определение 28.6. *Мультиграфом* G называется совокупность (28.1) множества V (основное множество) и мультимножества E двухэлементных подмножеств множества V. □

В этом определении E является уже мультимножеством, т. е. в нем могут быть повторяющиеся элементы, которые считаются разными. Это соответствует нескольким ребрам, соединяющим одну и ту же пару вершин (кратным ребрам). Но элементы E по-прежнему остаются двухэлементными множествами, поэтому каждое ребро должно соединять две разные вершины — петель нет. На рис. 28.3 слева показан мультиграф, а справа — его множество V и мультимножество E.

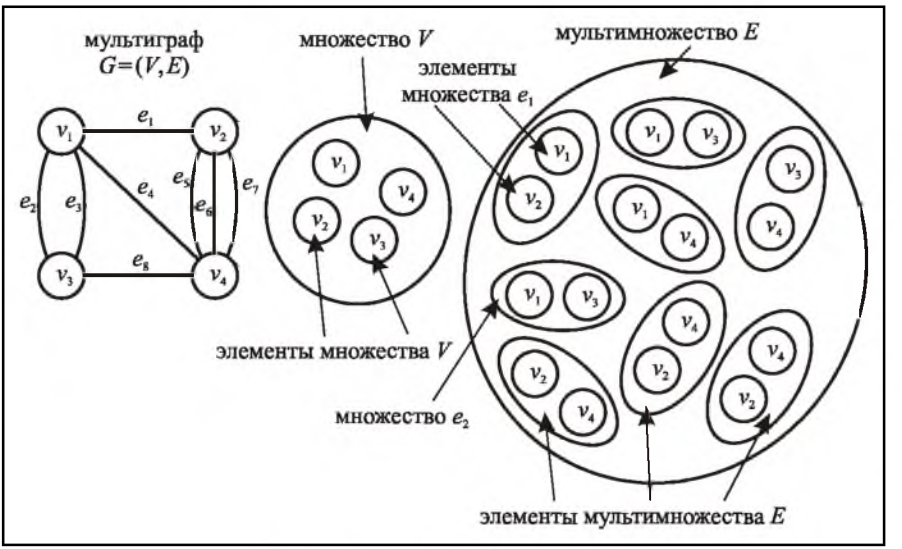


Рис. 28.3. Мультиграф $G=(V,E)$

Определение 28.7. *Псевдографом* G называется совокупность (28.1) множества V (основное множество) и мультимножества E двухэлементных мультиподмножеств множества V. □

В соответствии с этим определением допускаются не только одинаковые элементы множества E (кратные ребра), но и одинаковые элементы в каждом подмножестве e_i, т. е. ребро может соединять вершину саму с собой. Такие ребра называются *петлями*. Разумеется, петли в псевдографе также могут быть кратными. На рис. 28.4 показан псевдограф, его множество V и мультимножество E.

Определение 28.8. *Гиперграфом* G называется совокупность (28.1) множества V (основное множество) и мультимножества E непустых подмножеств множества V (не обязательно двухэлементных). □

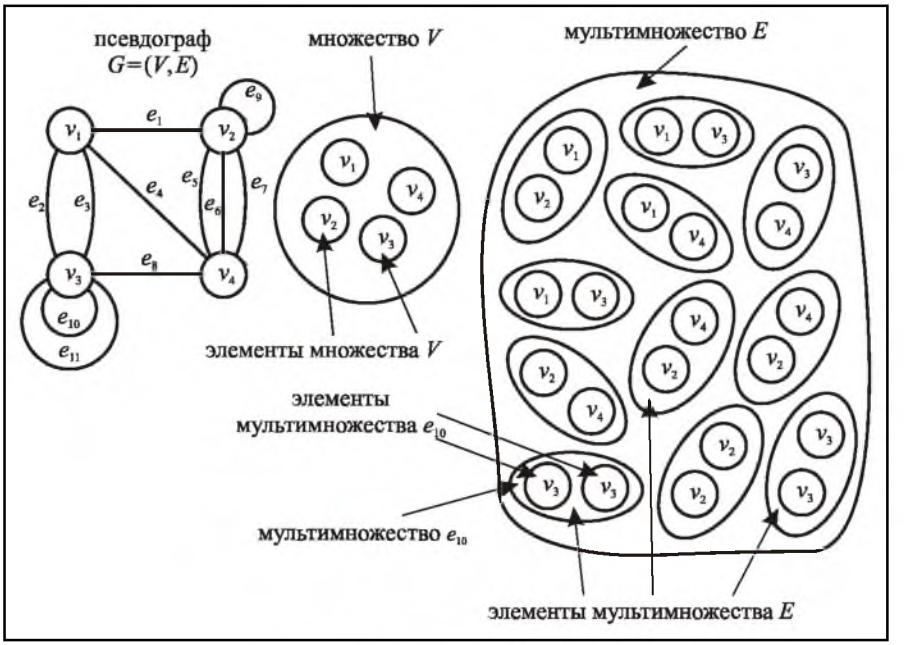


Рис. 28.4. Псевдограф $G=(V, E)$

Согласно этому определению ребра гиперграфа (они так и называются — гиперребра) могут соединять не только 1 или 2, но и любое количество вершин. Определение 28.8 допускает кратные гиперребра, в том числе и петли. На рис. 28.5 приведен пример гиперграфа. Здесь для описания петель достаточно 1-элементных подмножеств множества V.

У гиперграфа на этом рисунке есть гиперребра, соединяющие 3 вершины. Они обозначены линиями, соединенными точками. Но могут быть и гиперребра, соединяющие 4, 5 и вообще любое количество вершин из имеющихся в V.

Определение 28.9. Ребро (гиперребро) e_j называется *инцидентным* вершине v_i, если v_i является одним из концов e_j. □

Определение 28.10. Ребро (гиперребро) e_j называется *инцидентным* гиперребру e_k, если существует вершина v_i, инцидентная им обоим. □

Например, на рис. 28.5 ребро e_{12} инцидентно всем остальным ребрам.

Определение 28.11. Две вершины v_i и v_j называются *смежными*, если существует ребро, инцидентное им обеим. □

Определение 28.12. Граф (и все его обобщения) G называется *графом (соответственно мультиграфом и т. д.) со взвешенными вершинами*, если задано отображение V на множество действительных чисел:

$$\varphi: V \to R. \qquad (28.2)$$

Действительные числа b_i, характеризующие каждую вершину, называются в этом случае *весами вершин*. □

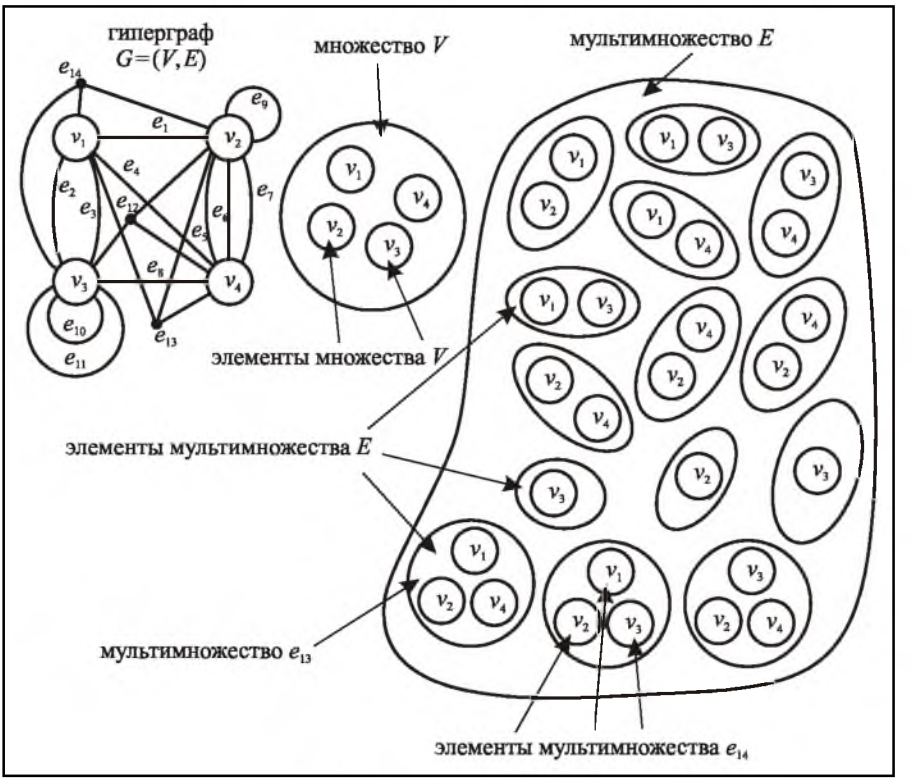


Рис. 28.5. Гиперграф $G=(V,E)$

Определение 28.13. Граф (и все его обобщения) G называется *графом со взвешенными ребрами*, если задано отображение E на множество действительных чисел:

$$\psi: E \to R. \tag{28.3}$$

Действительные числа c_k, характеризующие каждое ребро, называются в этом случае *весами ребер*. □

В частном случае, если веса вершин или ребер — целые числа (или числа любого счетного множества), мы можем взять набор красок, перенумеровать их в соответствии с этими числами, и сказать, что наш граф имеет *раскрашенные* вершины или ребра.

Рассмотрим некоторые часто встречающиеся в приложениях виды графов.

Определение 28.14. Граф (в смысле определения 28.1) K_n с n вершинами называется *полным графом* или *кликой*, если каждую пару его вершин соединяет ребро. □

Легко показать, что мощность клики K_n:

$$m = \frac{n(n-1)}{2}. \tag{28.4}$$

Действительно, каждая из n вершин соединяется с любой из оставшихся $(n-1)$, поэтому общее число концов ребер равно $n(n-1)$. Но у каждого ребра 2 конца, откуда и получаем формулу (28.4). На рис. 28.6 показана клика K_5.

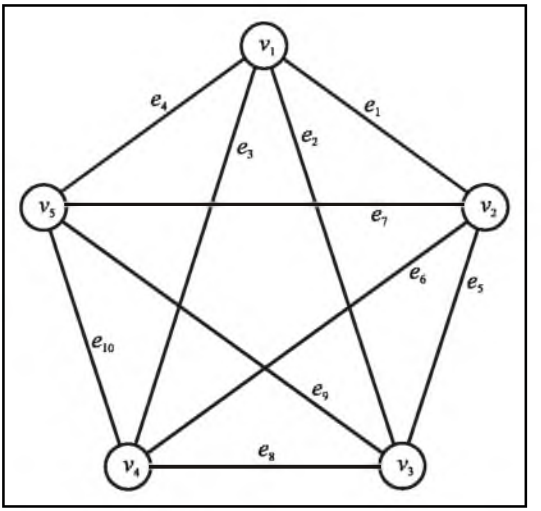


Рис. 28.6. Полный граф (клика) K_5

Определение 28.15. Граф G называется *двудольным*, если множество его вершин разбивается на 2 непересекающихся подмножества, которые обозначаются V и W, таких, что любое ребро e_i соединяет вершину из V с вершиной из W. □

Двудольные графы обычно обозначают как совокупность трех множеств:

$$D = (V, W, E). \tag{28.5}$$

Примерами двудольных графов могут служить звезда с любым числом лучей и многоугольник с четным числом вершин (рис. 28.7). Разделение множества вершин на доли показано здесь тонкой штриховой линией. Но обычно вершины двудольных графов изображают на двух вертикалях или горизонталях, как на рис. 28.8, *б*.

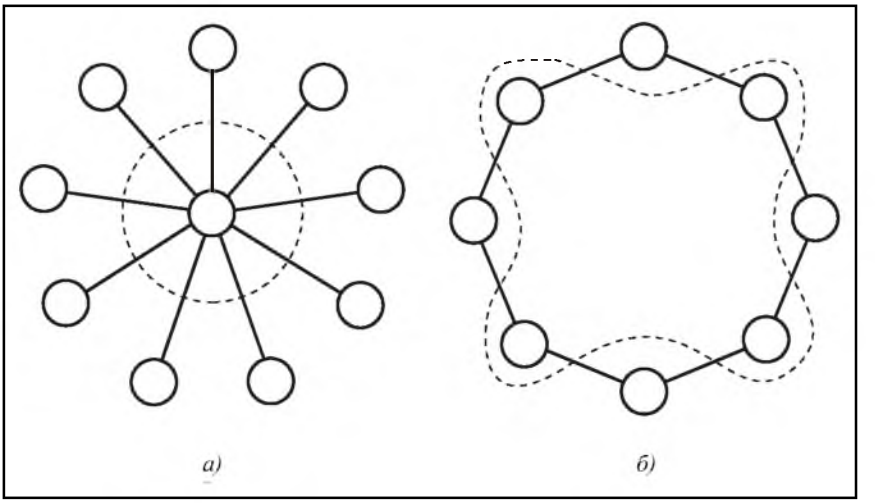


Рис. 28.7. Примеры двудольных графов: *а* — звезда; *б* — многоугольник с четным числом вершин

Двудольные графы играют очень большую роль в приложениях. Например, задачи назначения, распределения, планы перевозок часто описываются двудольными графами (работники и работы, источники ресурсов и потребители, склады и магазины). Но и в теоретических исследованиях нам приходится сталкиваться с ними. В частности, любому гиперграфу можно поставить во взаимно однозначное соответствие двудольный граф. Для этого достаточно вершины исходного гиперграфа отнести к первой доле V, а каждому *гиперребру* поставить в соответствие *вершину* из второй доли W. Теперь соединяем каждую вершину v_i из V с теми вершинами из W, которые соответствуют гиперребрам исходного гиперграфа, инцидентным с v_i.

Определение 28.16. Двудольный граф $D=(V_D, W_D, E_D)$ называется *соответствующим* гиперграфу $G=(V, E)$, если $|V_D|=|V|$, $|W_D|=|E|$, а ребро $(e_D)_k=\{(v_D)_i, (w_D)_j\}\in E_D$ тогда и только тогда, когда вершина v_i инцидентна ребру e_j. □

ПРИМЕР 28.2. Построим двудольный граф, соответствующий гиперграфу с рис. 28.5. На рис. 28.8, *а* показан исходный гиперграф, а на 28.8, *б* — соответствующий ему двудольный граф. У исходного гиперграфа $n=4$, $m=14$, поэтому в первой доле двудольного графа будет 4 вершины, а во второй — 14. В исходном гиперграфе v_1 инцидентна к e_1, e_2, e_3, e_4, e_{13} и e_{14}, поэтому в двудольном графе соединяем v_1 с w_1, w_2, w_3, w_4, w_{13} и w_{14}. Точно так же поступаем с остальными вершинами. □

Мы видим, что по гиперграфу G соответствующий ему двудольный граф строится однозначно. И наоборот, по двудольному графу однозначно восста-

навливается исходный гиперграф. Восстановление можно провести, например, так. В соответствующем двудольном графе $|V|=4$, поэтому рисуем 4 вершины исходного гиперграфа. Просматриваем каждую вершину второй доли W: это будет ребро восстанавливаемого гиперграфа. Вершина w_1 инцидентна v_1 и v_2, значит, ребро e_1 гиперграфа будет соединять v_1 и v_2, и т. д. Если w_9 инцидентна только одной вершине v_2, то в исходном гиперграфе будет петля e_9 при v_2. Вершина w_{12} инцидентна сразу трем вершинам доли V: v_2, v_3 и v_4 — значит, в исходном гиперграфе появится гиперребро $e_{12}=\{v_2, v_3, v_4\}$.

Что будет, если мы поменяем местами доли V и W в двудольном графе, а потом попытаемся восстановить гиперграф? Мы получим гиперграф, двойственный исходному.

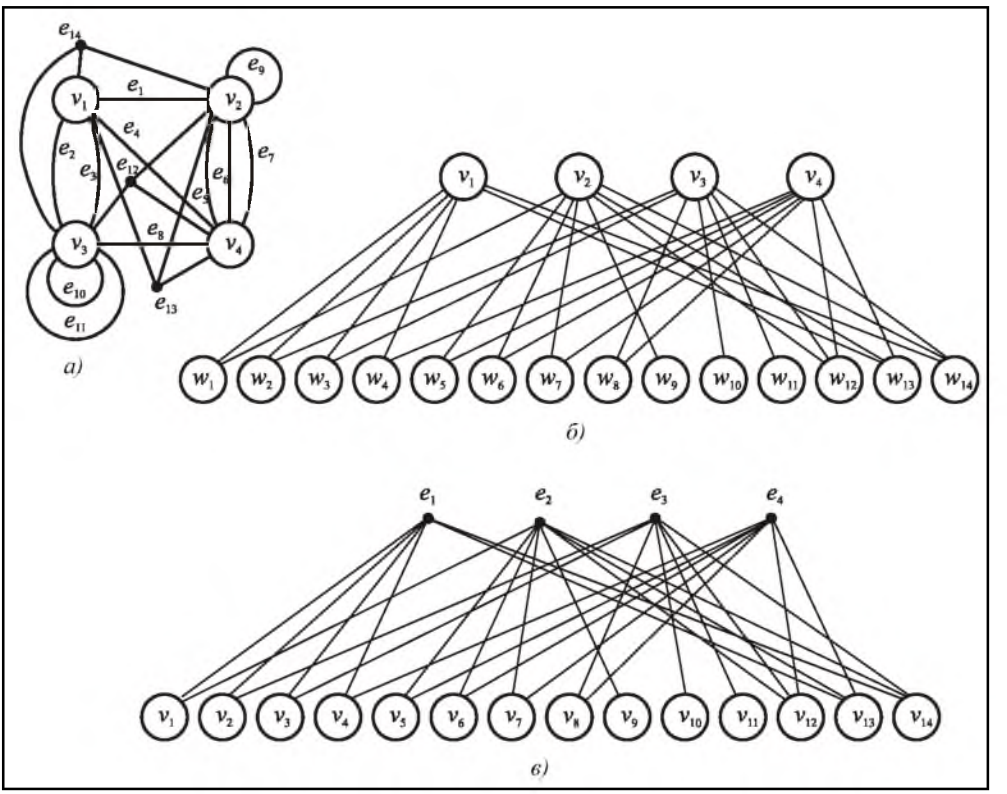


Рис. 28.8. Гиперграф: *а* — исходный; *б* — соответствующий ему двудольный граф; *в* — двойственный

Определение 28.17. Гиперграф $G'=\{V', E'\}$ называется *двойственным* по отношению к гиперграфу $G=\{V, E\}$, если $|V'|=|E|$, $|E'|=|V|$, и гиперребро e_j'

инцидентно вершине v_i' тогда и только тогда, когда гиперребро e_i инцидентно вершине v_j. □

ПРИМЕР 28.3. Построим гиперграф, двойственный гиперграфу с рис. 28.5. Мы уже построили двудольный граф, соответствующий исходному гиперграфу (рис. 28.8, *б*). Поменяем местами доли V и W (это можно сделать в уме). Восстановим теперь гиперграф с 14 вершинами и 4 гиперребрами. Гиперребро e_1 будет соединять вершины v_1, v_2, v_3, v_4, v_{13} и v_{14}. Аналогично строим другие гиперребра. Результат показан на рис. 28.8, *в*. □

Определение 28.18. *Матрицей инцидентности* гиперграфа G называется булева матрица A размером $n \times m$, каждый элемент которой a_{ij} = true тогда и только тогда, когда v_i инцидентна e_j. □

При работе в MATLAB нам удобнее будет записывать A в виде числовой матрицы, состоящей из нулей и единиц. Если такую матрицу преобразовать в булеву средствами MATLAB, то получится матрица в смысле определения 28.17. И наоборот, преобразование матрицы в смысле определения 28.17 в числовую дает матрицу, состоящую из нулей и единиц.

ПРИМЕР 28.4. Построим матрицу инцидентности для гиперграфа с рис. 28.5. Ее размеры 4×14. Поскольку v_1 инцидентна к e_1, e_2, e_3, e_4, e_{13} и e_{14}, то в 1-й строке равны единице элементы 1, 2, 3, 4, 13 и 14. Аналогично заполняем другие строки. Получаем:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}. \quad \square \qquad (28.6)$$

Если построить матрицу инцидентности для двойственного гиперграфа (рис. 28.8, *в*), то нетрудно заметить, что будет транспонированной по отношению к матрице исходного гиперграфа. Можно доказать соответствующую теорему — она почти очевидна из определения 28.16.

Теперь перейдем к ориентированным графам (орграфам). Здесь каждому ребру приписывается направление, и они называются *дугами* или *стрелками*. С точки зрения теории множеств каждая дуга — это не просто 2-элементное подмножество множества V, а упорядоченное 2-элементное подмножество.

Определение 28.19. *Орграф* G — это совокупность двух множеств V и E:

$$G = (V, E), \qquad (28.7)$$

где V — основное множество (вершины), а E — множество упорядоченных двухэлементных подмножеств множества V (*дуги* или *стрелки*). □

Пример орграфа — на рис. 28.1, *б*. Как и для графов, для орграфов можно ввести в рассмотрение кратные дуги и петли (возможно, тоже кратные). Орграф и его обобщения могут также иметь взвешенные или раскрашенные вершины и (или) дуги. А вот ориентированные гиперграфы мы вообще рассматривать не будем.

28.2. Как задать граф

Рассмотрим различные варианты задания исходных данных для графов (мульти-, псевдо- и т. д.). Что необходимо знать о графе для решения различных задач и рисования? Как правило, для решения задач важна структура связи вершин и ребер (дуг), а также веса, если они есть, а конкретное расположение вершин не имеет значения. При рисовании нужно знать и координаты вершин. С координатами можно поступить так: задать массив действительных чисел размером $n \times 2$, в 1-м столбце которого будут абсциссы вершин, а во 2-м — ординаты. Если координаты не заданы, можно попытаться нарисовать граф примерно в таком виде, как на рис. 28.8, *в*.

Информацию о структуре графа можно вводить (и хранить в памяти) поразному. Один из вариантов — задать матрицу инцидентности размером $n \times m$. Это булева матрица, и ее можно хранить очень экономно (например, побитно). Но в MATLAB логические переменные занимают 1 байт. Поэтому хотя экономия и есть, она не такая существенная, как при побитовом хранении. А вводить исходные данные из текстового файла все равно приходится как числа. Поэтому удобнее вводить информацию о структуре графа (орграфа) в виде списка его ребер (дуг), а матрицу инцидентности при необходимости формировать в процессе решения задачи.

Список ребер — это матрица размером $m \times 2$, в каждой строке которой — 2 целых числа: номера соединяемых вершин. Например, для графа с рис. 28.1, *а* и орграфа с рис. 28.1, *б* список ребер (дуг) имеет вид:

```
1  2
1  3
1  4
2  4
3  4
```

Для графа столбцы этой матрицы можно переставлять, а для орграфа — нет: условимся считать, что каждая стрелка направлена от вершины с номером, стоящим в 1-м столбце, к вершине, номер которой указан во 2-м столбце. При таком задании графов нет проблем с кратными ребрами (дугами) и петлями. Но гиперграфы задавать таким образом нельзя. Для них нужно использовать массив размером не $m \times 2$, а $m \times e_{max}$, где e_{max} — максимальное количество вер-

шин, которое может соединять гиперребро. Тогда более короткие строки дополняются нулями. Например, исходная информация о гиперграфе с рис. 28.8, *в* будет такой:

```
1   2   3   4  13  14   0   0
1   5   6   7   9  12  13  14
2   3   8  10  11  12  14   0
4   5   6   7   8  12  13   0
```

Информация в таком виде обрабатывается просто, т. к. мы нумеруем вершины числами от 1 до n, а вершины с номером 0 нет. По этим данным легко находятся m и n. Мощность графа m — это число строк матрицы, а размер n — максимальное число в ней. При необходимости также задаются веса вершин и ребер (дуг). Это одномерные массивы длиной n и m соответственно. Заметим, что приведенный выше набор данных однозначно характеризует также и гиперграф с рис. 28.8, *а*. Для каждой вершины (строки) здесь указаны номера инцидентных ей ребер. Поэтому информацию о графе (и всех его обобщениях) можно задавать также в виде массива из n строк, в каждой из которых перечислены номера ребер, инцидентных соответствующей вершине, и более короткие строки дополнены нулями.

Но орграф задавать в таком виде неудобно. Поэтому мы будем задавать граф (мультиграф, псевдограф, орграф) в виде списка его ребер размером $m \times 2$ и дополнять при необходимости одномерными векторами весов.

28.3. Описание процедуры *PlotGraph*

Мы ограничимся рисованием только графов (в т. ч. мульти- и псевдо-) и орграфов, но не гиперграфов. При этом будем сами задавать координаты вершин. Назовем нашу функцию `PlotGraph`. По правилам MATLAB это значит, что она должна быть размещена в файле с именем PlotGraph.m. Определимся с входными и выходными параметрами. Во-первых, мы должны задать координаты вершин — это 2-мерный массив размером $n \times 2$, который мы обозначим идентификатором `V`. Граф (орграф) может быть со взвешенными вершинами, значит, нужно дать пользователю выбор: если он задаст `V` размером не $n \times 2$, а $n \times 3$, то в 3-м столбце должны задаваться веса вершин. При рисовании веса задают, чтобы изобразить их. Поэтому, если пользователь не задает веса, то в каждом кружке, изображающем вершину, мы напишем ее номер, а если задает — то вес.

Далее мы должны задать структуру графа (орграфа) — список его ребер (дуг) в виде массива размером $m \times 2$. Обозначим его идентификатором `E`. Если ребра (дуги) взвешены, нужно также задать и их веса. Это удобно сделать в 3-м

столбце массива E. Как и для вершин, предоставим пользователю выбор: если он задает массив E размером $m \times 2$, то возле каждого ребра (дуги) мы будем писать его номер, а если $m \times 3$, то вес. Если в графе вообще нет ребер (одни вершины), то условимся, что можно или вообще не задавать второй аргумент, или задать его в виде пустого массива.

И наконец, пользователь должен указать, что же мы будем все-таки рисовать: граф или орграф. Здесь достаточно одного символа (идентификатор p). Если это символ 'o', то будем рисовать орграф, во всех других случаях (т. е. по умолчанию) рисуем граф.

Во всех процедурах рисования MATLAB возвращает дескриптор созданной фигуры. Не будем отступать от этого правила и в качестве выходного параметра (идентификатор h) возвратим дескриптор фигуры с нарисованным на ней графом.

Итак, с заголовком функции мы определились. После заголовка обычно размещается справочная информация, которая появляется в командном окне MATLAB при вызове справки по данной функции. Мы опишем в ней порядок вызова функции, входные и выходные аргументы, сведения об авторе и список других функций, которые использует данная функция и которых нет в стандартном наборе функций MATLAB. В частности, нам будет нужна функция arrow, которая рисует стрелки. Мы должны проинформировать об этом пользователя и указать, где ее можно взять. Вот как выглядит заголовок функции и справочная информация. По сравнению с версией, размещенной на сайте [51], здесь все комментарии переведены на русский язык.

```
function h=PlotGraph(V,E,p)
% Функция h=PlotGraph(V,E,p) рисует граф (орграф).
% Входные параметры:
%   V(n,2) или (n,3) - координаты вершин
%      (1-й столбец - x, 2-й - y) и, может быть, 3-й - веса;
%      n - количество вершин;
%      если V(n,2), рисуем номера вершин,
%      если V(n,3), рисуем веса вершин.
%   E(m,2) или (m,3) - ребра графа (дуги орграфа) и их веса;
%      1-й и 2-й элементы каждой строки - это номера вершин;
%      3-й элемент каждой строки это вес дуги;
%      m - количество дуг.
%      если E(m,2), рисуем номера ребер (дуг);
%      если E(m,3), рисуем веса ребер (дуг);
%      для графа без ребер задаем E=[];
%   p = 'g' (рисовать граф) или 'o' (рисовать орграф);
%    (необязательный параметр, по умолчанию 'g').
```

```
% Выходной параметр:
%   h - дескриптор фигуры.
% Автор: Сергей Иглин
% Электронная почта: iglin@kpi.kharkov.ua
% или: siglin@yandex.ru
% персональная страница: http://iglin.exponenta.ru
% Необходимые другие функции: ARROW.M.
% Эта функция может быть свободно скопирована с сайта Mathworks:
% http://www.mathworks.com.
% Выражаю признательность г-ну Ховарду (howardz@cc.gatech.edu)
% за тестирование этого алгоритма.
```

После этого нужно пропустить одну строку, которая рассматривается системой MATLAB как конец справочной информации. Далее начинаем проверку исходных данных. Начнем с того, что должно быть не менее одного входного параметра:

```
if nargin<1,
  error('Нет исходных данных!')
end
```

Находим размеры 1-го входного параметра V. Если это не 2-мерный массив — ошибка.

```
sv=size(V); % размер массива V
if length(sv)~=2,
  error('Массив V должен быть 2-мерным!')
end
```

Проверяем количество столбцов V.

```
if (sv(2)<2),
  error('Массив V должен иметь 2 или 3 столбца!'),
end
```

Если пользователь не задал второй аргумент (список ребер), задаем его пустым.

```
if nargin==1, % граф без ребер
  E=[];
end
```

Переходим к проверке второго входного параметра — массива E, если он не пустой.

```
if ~isempty(E), % граф с ребрами
  se=size(E); % размер массива E
```

```
  if length(se)~=2,
    error(' Массив E должен быть 2-мерным!')
  end
  if (se(2)<2),
    error(' Массив E должен иметь 2 или 3 столбца!'),
  end
```

Следующая проверка: номера всех вершин в массиве E должны быть целыми положительными числами:

```
  if ~all(all(E(:,1:2)>0)),
    error('1-й и 2-й столбцы массива E должны быть положительными!')
  end
  if ~all(all((E(:,1:2)==round(E(:,1:2))))),
    error('1-й и 2-й столбцы массива E должны быть целыми!')
  end
```

И наконец, еще одна проверка — на совместимость размера V и имеющихся номеров вершин в E:

```
    if max(max(E(:,1:2)))>sv(1),
      error('Координаты некоторых вершин не определены!');
    end
  end
```

На этом проверка исходных данных заканчивается, и мы приступаем к их анализу. Находим размер и мощность графа. Проверяем, что надо рисовать: граф или орграф.

```
if isempty(E), % нет ребер
  m=0;
else
  m=size(E,1); % количество дуг
end
n=sv(1); % количество вершин
if nargin<3, % только 2 входных параметра
  p1='g';
else
  if ~ischar(p),
    error('3-й аргумент p должен быть строкой!')
  else
    p1=p(1);
  end
end
```

Вершины мы будем рисовать кружочками. Чтобы не затенять чертеж, условимся, что радиус кружков должен быть равен 0,1 от минимального расстоя-

ния между вершинами. Найдем эту величину и вычислим координаты точек, по которым будем строить кружки-вершины.

```
rc=0.1; % коэффициент в радиусе
t=linspace(-pi/2,3*pi/2); % массив параметров
r=rc*min(pdist(V(:,1:2))); % 0.1 от минимального расстояния
xc=r*cos(t);
yc=r*sin(t);
```

В нашем графе могут встретиться кратные ребра (дуги). Если ребро не кратное, его будем рисовать в виде прямой. Но кратные ребра удобнее рисовать изогнутыми, причем с разными степенями изогнутости, чтобы они не накладывались друг на друга. Проще всего заранее подготовить полуэллипс, а затем по мере необходимости деформировать его так, как нужно: растягивать, сжимать, поворачивать. Подготовим такой полуэллипс (т. е. координаты его точек):

```
tr=linspace(0,pi); % для рисования кратных ребер
xr=0.5-cos(tr)/2;
yr=sin(tr);
```

Следующий этап — сортировка ребер. Она нужна для того, чтобы выявить кратные ребра и петли, которые рисуются по-разному. Простые ребра — это прямые, кратные — полуэллипсы, а петли — дуги окружностей. Если еще и петли кратные, нужно будет рисовать несколько дуг окружностей разных радиусов. Первый этап сортировки — сделать так, чтобы первая вершина была меньше второй. Для этого нужно переставить местами первые и вторые элементы некоторых строк. И надо запомнить, в каких строках мы провели перестановку, чтобы правильно нарисовать потом стрелки. Поэтому мы дополним список ребер (дуг) слева двумя столбцами: в первом будем запоминать сделанную перестановку вершин, а во втором сохраним номера ребер.

```
if ~isempty(E), % есть ребра
  E=[zeros(m,1),[1:m]',E]; % добавили столбец 0 и номера ребер
  need2=find(E(:,4)<E(:,3)); % необходима замена v1<->v2
  tmp=E(need2,3);
  E(need2,3)=E(need2,4);
  E(need2,4)=tmp; % заменили 3-й и 4-й столбцы при необходимости
  E(need2,1)=1; % равно 1, если заменили v1 и v2
```

Теперь у нас номер первой вершины всегда меньше номера второй. Нам все равно, в каком порядке рисовать ребра, поэтому рассортируем их в порядке возрастания 1-х вершин. Это — второй этап сортировки:

```
  [e1,ie1]=sort(E(:,3)); % сортируем по 1-й вершине
  E1=E(ie1,:);
```

Продолжаем сортировку ребер, теперь уже по номеру 2-й вершины. Это третий этап:

```
  for k2=E1(1,3):E1(end,3), % просматриваем все номера 1-х вершин
    num2=find(E1(:,3)==k2); % одинаковая 1-я вершина
    if ~isempty(num2), % сортируем по 2-й вершине
      E3=E1(num2,:);
      [e3,ie3]=sort(E3(:,4));
      E4=E3(ie3,:);
      E1(num2,:)=E4;
    end
  end
```

Теперь лексикографическое упорядочение ребер закончено. Если ребра есть, то они расположены в порядке возрастания 1-х вершин, а при одинаковых номерах 1-х вершин — в порядке возрастания номеров 2-х вершин. Такое расположение позволяет легко выявить кратные ребра: мы просто будем проверять для каждого ребра следующие за ним, пока не обнаружим другое ребро. Последний этап сортировки — это удаление петель. Они будут рисоваться по другому правилу, поэтому удалим их в отдельный массив. Заметим, что удаляемые петли также лексикографически упорядочены, поэтому легко будет выявить кратные петли. Удаляем петли:

```
  ip=find(E1(:,3)==E1(:,4)); % ищем петли
  Ep=E1(ip,:); % петли
  E2=E1(setdiff([1:m],ip),:); % ребра без петель
end % есть ребра
```

Подготовительная работа закончена. Переходим непосредственно к рисованию. Создаем новую фигуру и присваиваем ее дескриптор выходному параметру. Включаем задержку, т. к. рисовать нам придется много.

```
h=figure;
hold on
```

Вначале рисуем вершины. У нас есть координаты каждой вершины — это будет центр кружка. Точки для рисования кружка у нас также готовы. Поэтому рисуем каждую вершину в нужном месте сплошной синей линией. Выясняем, как ее подписывать. Если пользователь задал веса вершин, пишем внутри кружка вес, а если нет — номер.

```
for k=1:sv(1), % для каждой вершины
  plot(V(k,1)+xc,V(k,2)+yc,'b-'); % рисуем кружок
  if sv(2)==3, % веса вершин заданы
    s=num2str(V(k,3)); % вес
```

```
  else
    s=num2str(k); % номер вершины
  end
  text(V(k,1)-0.028*length(s),V(k,2),s); % добавляем текст внутри кружка
end
```

Переходим к рисованию ребер (дуг), если они есть. Его удобнее организовать в виде цикла `while`, т. к. одиночные и кратные ребра рисуются по-разному. В начале цикла находим координаты вершин текущего ребра.

```
if ~isempty(E), % есть ребра
  k=0;
  m2=size(E2,1); % количество ребер без петель
  while k<m2, % пока не исчерпаны все ребра
    k=k+1; % текущее ребро
    MyE=V(E2(k,3:4),1:2); % координаты вершин 1, 2
```

Проверяем, сколько раз встречается данное ребро в списке. Благодаря тому, что ребра рассортированы, мы просто проверяем ребра, следующие за данным, пока не обнаружим ребро с другими номерами вершин или конец списка ребер.

```
    k1=1; % ищем кратные ребра
    if k<m2, % не последнее ребро
      while all(E2(k,3:4)==E2(k+k1,3:4)), % обе вершины совпадают
        k1=k1+1; % увеличили кратность
        if k+k1>m2, % последнее ребро
          break; % выходим из цикла while
        end
      end
    end % в переменной k1 - кратность текущего ребра
```

Рисовать кратные ребра будем дужками разных радиусов симметрично относительно прямой, соединяющей вершины. Находим радиусы дужек, расстояние между вершинами, угол поворота дужки. Растягиваем дужку до нужной длины.

```
    ry=r*[1:k1]; % радиусы дужек
    ry=ry-mean(ry); % симметрично относительно прямой
    l=norm(MyE(1,:)-MyE(2,:)); % длина линии
    dx=MyE(2,1)-MyE(1,1);
    dy=MyE(2,2)-MyE(1,2);
    alpha=atan2(dy,dx); % угол поворота
    cosa=cos(alpha);
    sina=sin(alpha);
    MyX=xr*l; % растянули дужку
```

Начинаем рисовать ребро (или кратные ребра). Растягиваем дужку в поперечном направлении до нужного радиуса, поворачиваем, обрезаем с концов, чтобы она не накладывалась на вершину-кружок, и рисуем черной сплошной линией.

```
    for k2=1:k1, % рисуем ребра (дуги)
      MyY=yr*ry(k2); % радиус дужки
      MyXg=MyX*cosa-MyY*sina; % поворачиваем
      MyYg=MyX*sina+MyY*cosa;
      MyL=length(find(MyXg.^2+MyYg.^2<r^2)); % попадают в вершину
      MyXpl=MyXg(MyL+1:end-MyL)+MyE(1,1); % обрезаем с одного конца
      MyYpl=MyYg(MyL+1:end-MyL)+MyE(1,2); % обрезаем с другого конца
      plot(MyXpl,MyYpl,'k-'); % рисуем
```

Проверяем, что нам нужно рисовать: граф или орграф. Если орграф, то на нужном конце рисуем стрелку.

```
      if pl=='o', % задано рисовать орграф
        if E2(k+k2-1,1)==1, % нужно рисовать стрелку в начале
          arrow([MyXpl(2);MyYpl(2)],[MyXpl(1);MyYpl(1)]);
        else % нужно рисовать стрелку в конце
          arrow([MyXpl(end-1);MyYpl(end-1)],[MyXpl(end);MyYpl(end)]);
        end
      end
```

Теперь делаем нужные надписи над ребрами (дугами). Если ребра взвешены, пишем вес, а если нет — номер ребра.

```
    if se(2)==3, % ребра взвешены
      s=num2str(E2(k+k2-1,5));
    else % ребра не взвешены
      s=num2str(E2(k+k2-1,2));
    end
    text(MyXpl(length(MyXpl)/2),MyYpl(length(MyYpl)/2),s);
  end
  k=k+k1-1; % номер последнего нарисованного ребра
end
```

Осталось нарисовать петли, если они есть. Их рисование организовано в таком же цикле `while`, как и для ребер. Для очередной петли проверяем, какова ее кратность.

```
k=0;
ml=size(Ep,1); % количество петель
while k<ml,
  k=k+1; % текущая петля
```

```
MyV=V(Ep(k,3),1:2); % координаты вершины
k1=1; % ищем кратность
if k<ml,
  while all(Ep(k,3:4)==Ep(k+k1,3:4)),
    k1=k1+1;
    if k+k1>ml,
      break;
    end
  end
end % в k1 - кратность текущей петли
```

Задаем радиусы окружностей, которыми будут рисоваться петли. Возьмем их больше, чем радиус кружков-вершин. Для каждой петли найдем центр окружности, отрежем кусочки, попадающие внутрь кружка-вершины, а остальное нарисуем.

```
  ry=[1:k1]+1; % радиус
  for k2=1:k1, % рисуем петли
    MyX=xc*ry(k2); % центр
    MyY=(yc+r)*ry(k2); % центр
    MyL=length(find(MyX.^2+MyY.^2<r^2))/2; % попадает внутрь вершины
    MyXpl=MyX(MyL+1:end-MyL)+MyV(1); % отрезаем лишнее
    MyYpl=MyY(MyL+1:end-MyL)+MyV(2);
    plot(MyXpl,MyYpl,'k-'); % рисуем сплошной черной линией
```

При необходимости рисуем стрелку:

```
    if pl=='o', % нужно рисовать орграф
      arrow([MyXpl(10);MyYpl(10)],[MyXpl(1);MyYpl(1)]);
    end
```

Делаем надпись: вес или номер ребра-петли:

```
    if se(2)==3, % ребра взвешены
      s=num2str(Ep(k+k2-1,5)); % надписываем вес
    else % ребра не взвешены
      s=num2str(Ep(k+k2-1,2)); % надписываем номер
    end
    text(MyXpl(length(MyXpl)/2),MyYpl(length(MyYpl)/2),s);
  end
  k=k+k1-1; % номер последней нарисованной петли
 end
end % если есть ребра
```

Заканчиваем оформление рисунка. Выключаем задержку, т. к. уже все нарисовали. Убираем с рисунка оси. Выравниваем масштаб, чтобы кружочки-вершины действительно были кружками, а не эллипсами.

```
hold off
axis off
da=daspect;
da(1:2)=min(da(1:2));
daspect(da); % одинаковый масштаб
return
```

Так выглядит простейшая процедура рисования графов. Конечно, совершенству нет предела, и ее без конца можно улучшать. Можно, например, учесть, что надписи не должны выходить за пределы вершин и автоматически выбирать для этого нужный размер шрифта. Можно научить программу рисовать ребра так, чтобы они не залезали на вершины, а обходили их. Можно (продолжайте сами)... В общем, для хорошего программиста-дизайнера эта процедура — лишь заготовка для создания своего детища.

28.4. Пример обращения

Проиллюстрируем работу этой функции на примере рисования орграфа с 25 вершинами и 46 дугами. Этот пример взят из [22]. Но нарисуем орграф

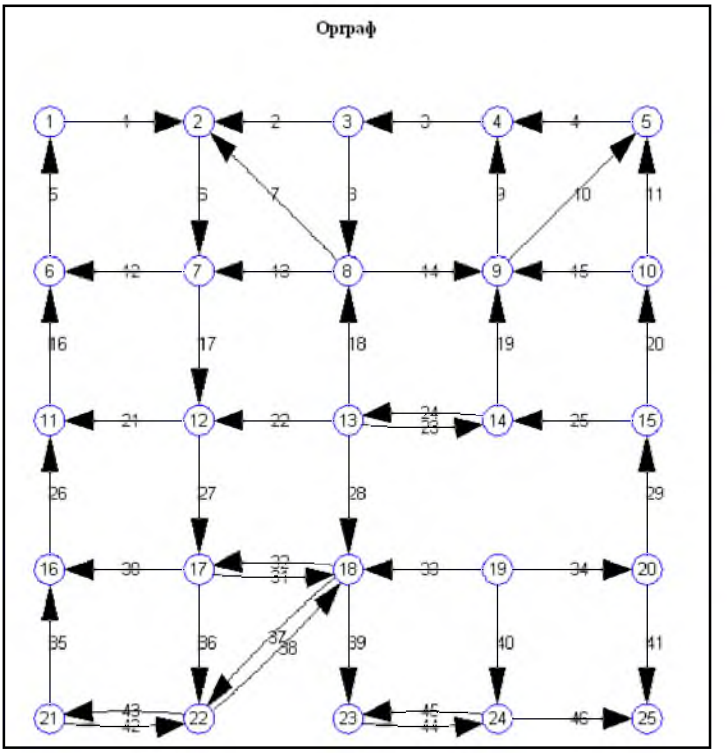


Рис. 28.9. Пример рисования орграфа

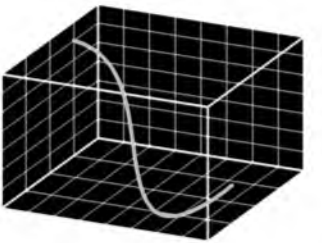
с невзвешенными вершинами и невзвешенными дугами. Установим шрифт и подпишем заголовок. Результат показан на рис. 28.9.

```
clear all
V=[[0 4];[1 4];[2 4];[3 4];[4 4];[0 3];[1 3];[2 3];[3 3];[4 3];...
   [0 2];[1 2];[2 2];[3 2];[4 2];[0 1];[1 1];[2 1];[3 1];[4 1];...
   [0 0];[1 0];[2 0];[3 0];[4 0]];
E=[[ 1  2];[ 3  2];[ 4  3];[ 5  4];[ 6  1];...
   [ 2  7];[ 8  2];[ 3  8];[ 9  4];[ 9  5];...
   [10  5];[ 7  6];[ 8  7];[ 8  9];[10  9];...
   [11  6];[ 7 12];[13  8];[14  9];[15 10];...
   [12 11];[13 12];[13 14];[14 13];[15 14];...
   [16 11];[12 17];[13 18];[20 15];[17 16];...
   [17 18];[18 17];[19 18];[19 20];[21 16];...
   [17 22];[18 22];[22 18];[18 23];[19 24];...
   [20 25];[21 22];[22 21];[23 24];[24 23];[24 25]];
PlotGraph(V,E,'o'); % рисуем орграф
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfОрграф')
```

28.5. Вопросы для самопроверки

1. Докажите, что для взаимно двойственных гиперграфов соответствующие им двудольные графы отличаются только порядком долей вершин.
2. Нарисуйте гиперграф с рис. 28.8, *а* в таком же виде, как и двойственный ему на рис. 28.8, *в*. Будет ли этот гиперграф тем же, что и исходный?
3. Докажите, что гиперграф, двойственный двойственному, совпадает с исходным.
4. Докажите теорему, что матрицы инцидентности взаимно двойственных гиперграфов взаимно транспонированы.
5. Какую структуру будет иметь матрица инцидентности двудольного графа, если первые $|V|$ ее строк соответствуют вершинам первой доли, а последние $|W|$ — второй?
6. Разработайте процедуру рисования гиперграфов. Для начала попробуйте нарисовать средствами MATLAB гиперграф в таком виде, как на рис. 28.8, *в*. Нужны ли здесь координаты вершин?
7. Все ли проверки на правильность исходных данных выполнены в функции `PlotGraph`? Что будет, если задать строки, ячейки, нечисловые значения?
8. Запишите определение смешанного графа, у которого есть и ребра, и дуги. Как свести такой граф к орграфу?

ГЛАВА 29



Больше ребер, меньше вершин

В этой главе мы будем рассматривать только графы в смысле *определения 28.1*, т. е. без петель, кратных ребер и дуг.

29.1. Максимальное паросочетание

Рассмотрим решение задачи о максимальном паросочетании, в том числе взвешенном, средствами MATLAB. При этом будем стремиться как можно больше использовать готовые процедуры, созданные ранее пользователями MATLAB.

29.1.1. Основные определения

Определение 29.1. Подмножество ребер $E_1 \subseteq E$ графа $G=(V, E)$ называется *паросочетанием*, если среди них нет инцидентных. Паросочетание называется *максимальным по включению*, если оно не является подмножеством паросочетания с большим числом ребер. Паросочетание называется *максимальным*, если оно состоит из максимально возможного количества ребер. □

ПРИМЕР 29.1. На рис. 29.1, *а* показан граф с $n=8$ и $m=13$. Подмножество ребер $\{e_1, e_{11}\}$ образует паросочетание, т. к. эти ребра не инцидентные. Но это паросочетание не является максимальным по включению: его можно дополнить, например, ребром e_9. Полученное подмножество $\{e_1, e_{11}, e_9\}$ также является паросочетанием, причем максимальным по включению, т. к. любое другое ребро инцидентно хотя бы одному из ребер e_1, e_9 или e_{11}. Оно показано на рис. 29.1, *б*. Но, хотя $\{e_1, e_{11}, e_9\}$ максимально по включению, оно не является максимальным. Если мы дополним $\{e_1, e_{11}\}$ не e_9, а, например, e_8 и e_{10}, то получим подмножество уже не трех, а четырех ребер $\{e_1, e_{11}, e_8, e_{10}\}$. Это — одно из возможных максимальных паросочетаний (рис. 29.1, *в*). Здесь размер и

мощность графа небольшие, и можно просто перебрать все варианты, чтобы убедиться: среди любых пяти ребер обязательно будут инцидентные. □

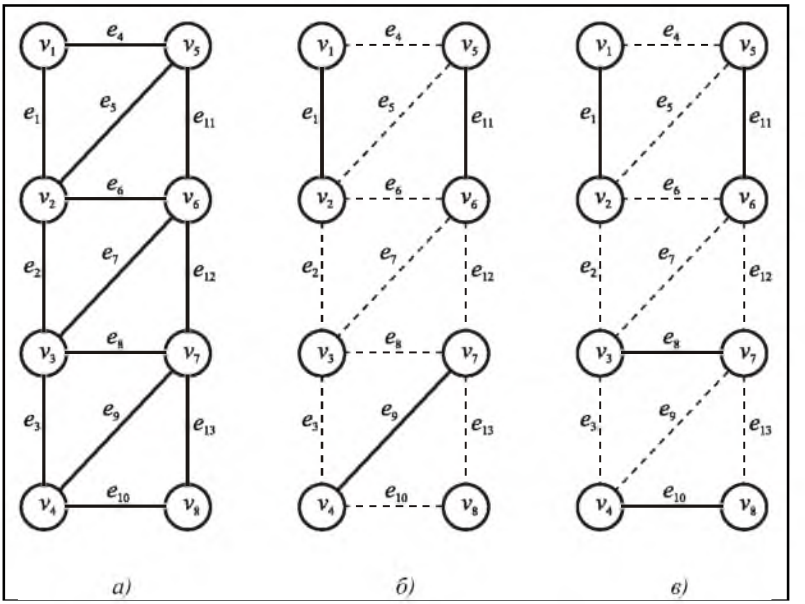


Рис. 29.1. Паросочетание: *а* — исходный граф;
б — максимальное по включению паросочетание; *в* — максимальное паросочетание

29.1.2. Сведение к задаче целочисленного линейного программирования

В этом разделе мы рассмотрим задачу о нахождении максимального паросочетания в графе. Если ребра графа взвешены, то обобщением здесь является задача о максимальном взвешенном паросочетании. В ней требуется найти паросочетание максимально возможного общего веса. Примером такой задачи является разбиение коллектива людей на пары: экипажи, бригады и т. п. Если каждое ребро означает возможность совместной работы, то получаем невзвешенную задачу: создать максимально возможное количество работоспособных бригад. Если же ребра взвешенные, то обычно вес ребра означает производительность данной бригады. В этом случае формирование коллектива с максимальной общей производительностью является задачей о максимальном взвешенном паросочетании.

Задача о максимальном (взвешенном) паросочетании — одна из классических задач теории графов. Для ее решения создано множество алгоритмов, обзор которых есть в [2, 33]. Наша цель — создать наиболее простой алгоритм с минимальным количеством кода. Поэтому воспользуемся тем, что эта задача

может быть записана в терминах целочисленного линейного программирования (ЦЛП), а для решения задачи ЦЛП пользователями MATLAB созданы эффективные процедуры [52, 56, 60].

Введем в рассмотрение вектор-столбец e длиной m. Назовем этот вектор *ассоциированным* с ребрами графа E, т. к. каждая координата e_k вектора e будет характеризовать соответствующее ребро. Если ребро e_k входит в паросочетание, то ассоциированная с ней переменная e_k будет принимать значение 1, а если нет — то 0. Тогда общее количество ребер, входящих в паросочетание, можно записать в виде:

$$z = \sum_{k=1}^{m} e_k = (\mathbf{1}, \mathbf{e}), \tag{29.1}$$

где 1 — вектор-столбец из единиц нужной размерности. В задаче о максимальном паросочетании эту величину нужно максимизировать при условии, что все переменные e_k могут принимать значения только 0 или 1:

$$\begin{cases} e_k = 0 \vee 1; \\ k = \overline{1, m}, \end{cases} \tag{29.2}$$

и среди ребер нет инцидентных. Последнее требование означает, что для каждой вершины существует не более одного ребра, инцидентного ей. Перейдем от ребер e_k к ассоциированным с ними переменным e_k. Тогда для каждой вершины v_i сумма переменных e_k, ассоциированных с ребрами, инцидентными с v_i, не превышает единицы. Так, для графа с рис. 29.1, *а* система ограничений-неравенств имеет вид:

$$\begin{cases} e_1 + e_4 \le 1; & e_4 + e_5 + e_{11} \le 1; \\ e_1 + e_2 + e_5 + e_6 \le 1; & e_6 + e_7 + e_{11} + e_{12} \le 1; \\ e_2 + e_3 + e_7 + e_8 \le 1; & e_8 + e_9 + e_{12} + e_{13} \le 1; \\ e_3 + e_9 + e_{10} \le 1; & e_{10} + e_{13} \le 1. \end{cases} \tag{29.3}$$

Здесь для экономии места уравнения системы выписаны не в один, а в два столбика. Если воспользоваться матрицей инцидентности (28.6), то условие отсутствия инцидентных ребер записывается так:

$$\mathrm{Ae} \le 1. \tag{29.4}$$

Таким образом, имеем задачу ЦЛП:

$$\begin{cases} z = (\mathbf{1}, \mathbf{e}) \to \max; \\ \mathbf{Ae} \le \mathbf{1}; \\ e_k = 0 \vee 1;\ k = \overline{1, m}. \end{cases} \tag{29.5}$$

В задаче о максимальном взвешенном паросочетании нужно максимизировать не общее количество ребер, а их суммарный вес. Обозначим вектор-столбец весов ребер c. Тогда вместо (29.5) будем иметь задачу, которая отличается только целевой функцией:

$$\begin{cases} z = (\mathbf{c},\mathbf{e}) \to \max; \\ \mathbf{Ae} \le \mathbf{1}; \\ e_k = 0 \vee 1;\ k = \overline{1,m}. \end{cases} \tag{29.6}$$

В случае двудольного графа задача о максимальном взвешенном паросочетании — это классическая задача о назначениях. Здесь первая доля вершин V — это работники; вторая W — работы; ребра — возможность того или иного работника выполнять данную работу; вес ребра — производительность. Требуется расставить работников на работы так, чтобы каждую работу выполнял максимум один работник, каждый работник выполнял максимум одну работу, а общая производительность была максимальной. Если вы изучали теорию линейного программирования, то знаете, что задача о назначениях — это частный случай транспортной задачи, для которой при целочисленных исходных данных решение также получается целочисленным. Поэтому для двудольных графов условие целочисленности (29.2) можно заменить более слабым условием ограниченности:

$$\begin{cases} 0 \le e_k \le 1; \\ k = \overline{1,m}. \end{cases} \tag{29.7}$$

Все равно решение полученной задачи линейного программирования (уже не целочисленного!) автоматически получится целым: все e_k будут равны 0 или 1. Но для произвольного графа это не так: для него условия (29.2) нужно задавать принудительно.

29.1.3. Описание процедуры *MaxMatch*

Рассмотрим, как реализовать в MATLAB решение задачи о максимальном (взвешенном) паросочетании. Очевидно, для решения этой задачи нам не нужны координаты вершин. Достаточно списка ребер — массива размером $m \times 2$ или $m \times 3$. Обозначим его идентификатором E. Будем предполагать, что, если пользователь задал 2 столбца, то нужно решать невзвешенную задачу, а если 3 — то взвешенную. Возвращать будем список номеров ребер, входящих в максимальное (взвешенное) паросочетание. Вот как выглядит заголовок функции и справочная информация к ней:

```
function nMM=MaxMatch(E)
% Функция nMM=MaxMath(E) решает задачу о максимальном паросочетании.
```

```
% Входной параметр:
%   E(m,2) или (m,3) - ребра графа и их веса;
%      1-й и 2-й элементы каждой строки - это номера вершин;
%      3-й элемент каждой строки - это вес ребра;
%      m - количество ребер.
%      Если задан массив E(m,2), то все веса равны 1.
% Выходной параметр:
%   nMM - список номеров ребер, включенных в максимальное
%      (взвешенное) паросочетание.
% Используется приведение к задаче ЦЛП.
% Автор: Сергей Иглин
% Электронная почта: iglin@kpi.kharkov.ua
% или: siglin@yandex.ru
% персональная страница: http://iglin.exponenta.ru
% Необходимые другие функции: MILP.M.
% Эта функция может быть свободно скопирована с сайта:
% Matlog: Logistics Engineering Matlab Toolbox,
% http://www.ie.ncsu.edu/kay/matlog/.
```

В справочной информации, которая появляется в командном окне MATLAB после ввода команды:

```
help maxmatch
```

помимо другой информации, указывается также, что для решения задачи ЦЛП будет использована функция `milp`, которой нет в MATLAB, и дается ссылка на ресурс Интернета, где ее можно найти.

После пропуска строки начинаем проверку исходных данных. Проверяем, есть ли они вообще. Если нет — выходим с сообщением об ошибке.

```
if nargin<1,
  error('Нет исходных данных!')
end
```

Находим размеры массива входных данных. Проверяем, чтобы массив был двумерным.

```
sz=size(E); % размеры массива E
if length(sz)~=2,
  error('Массив E должен быть двумерным!')
end
```

Для работы процедуры нужно, чтобы входной массив имел не менее двух столбцов.

```
if (sz(2)<2),
  error('В массиве E должно быть 2 или 3 столбца!'),
end
```

В 1-м и 2-м столбцах входного массива должны быть целые положительные числа: номера вершин.

```
if ~all(all(E(:,1:2)>0)),
  error('1-й и 2-й столбцы массива E должны быть положительными!')
end
if ~all(all((E(:,1:2)==round(E(:,1:2))))),
  error('1-й и 2-й столбцы массива E должны быть целыми!')
end
```

На этом проверка исходных данных закончена. Переходим к решению задачи. Найдем размер и мощность графа. Если пользователь задал невзвешенный граф, зададим единичные веса ребер.

```
m=size(E,1); % количество ребер
n=max(max(E(:,1:2))); % количество вершин
if sz(2)==2, % веса не заданы
  E(:,3)=1; % задаем все веса =1
end
```

Формируем задачу ЦЛП в виде (29.6). Строим матрицу инцидентности A, вектор l в правой части (29.4), вектор коэффициентов при целевой функции c, нижние и верхние границы по каждой переменной (29.2).

```
A=zeros(n,m); % для матрицы инцидентности
for k=1:m, % просматриваем все ребра
  A(E(k,1:2),k)=1; % заполняем матрицу инцидентности
end
b=ones(n,1); % правые части ограничений-неравенств
c=-E(:,3); % коэффициенты при целевой функции
vlb=zeros(1,m); % нижние границы
vub=ones(1,m); % верхние границы
```

И наконец, решаем задачу ЦЛП и возвращаем результаты в вызывающую программу. Функция `milp` может возвращать не совсем целые результаты, поэтому округляем ее решения.

```
xmin=milp(c,A,b,vlb,vub); % решаем задачу ЦЛП
nMM=find(round(xmin)); % ответ - номера ребер
return
```

Как видим, процедура почти полностью состоит из проверки исходных данных и подготовки решения. Само решение сводится к вызову готовой процедуры `milp`. Это — один из приемов эффективного программирования в MATLAB. Библиотека уже созданного настолько обширна, что почти всегда можно найти одну или несколько готовых функций, из которых, как из кубиков, сложить свою программу.

29.1.4. Пример обращения к процедуре *MaxMatch*

Рассмотрим пример использования функции `MaxMatch`. Введем такой же граф, как и в *главе 28*, но со взвешенными ребрами. Нарисуем его в виде неориентированного графа с невзвешенными вершинами и взвешенными ребрами (рис. 29.2).

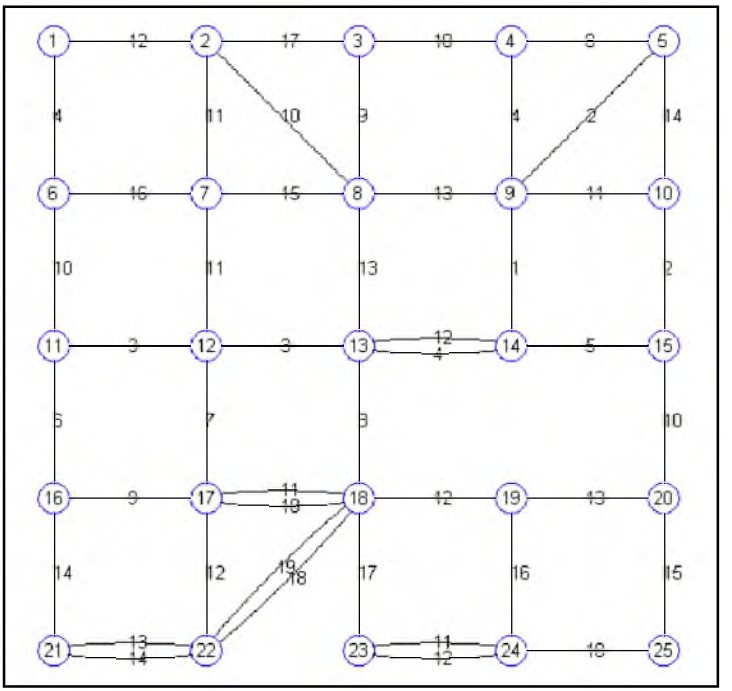


Рис. 29.2. Исходный граф со взвешенными ребрами

```
clear all
V=[[0 4];[1 4];[2 4];[3 4];[4 4];...
    [0 3];[1 3];[2 3];[3 3];[4 3];...
    [0 2];[1 2];[2 2];[3 2];[4 2];...
    [0 1];[1 1];[2 1];[3 1];[4 1];...
    [0 0];[1 0];[2 0];[3 0];[4 0]]; % координаты вершин
E=[[ 1  2 12];[ 3  2 17];[ 4  3 10];[ 5  4  8];[ 6  1  4];...
   [ 2  7 11];[ 8  2 10];[ 3  8  9];[ 9  4  4];[ 9  5  2];...
   [10  5 14];[ 7  6 16];[ 8  7 15];[ 8  9 13];[10  9 11];...
   [11  6 10];[ 7 12 11];[13  8 13];[14  9  1];[15 10  2];...
   [12 11  3];[13 12  3];[13 14  4];[14 13 12];[15 14  5];...
   [16 11  6];[12 17  7];[13 18  8];[20 15 10];[17 16  9];...
```

```
   [17 18 10];[18 17 11];[19 18 12];[19 20 13];[21 16 14];...
   [17 22 12];[18 22 19];[22 18 18];[18 23 17];[19 24 16];...
   [20 25 15];[21 22 14];[22 21 13];[23 24 12];[24 23 11];[24 25 10]];
PlotGraph(V,E); % рисуем граф
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfИсходный граф со взвешенными ребрами ')
```

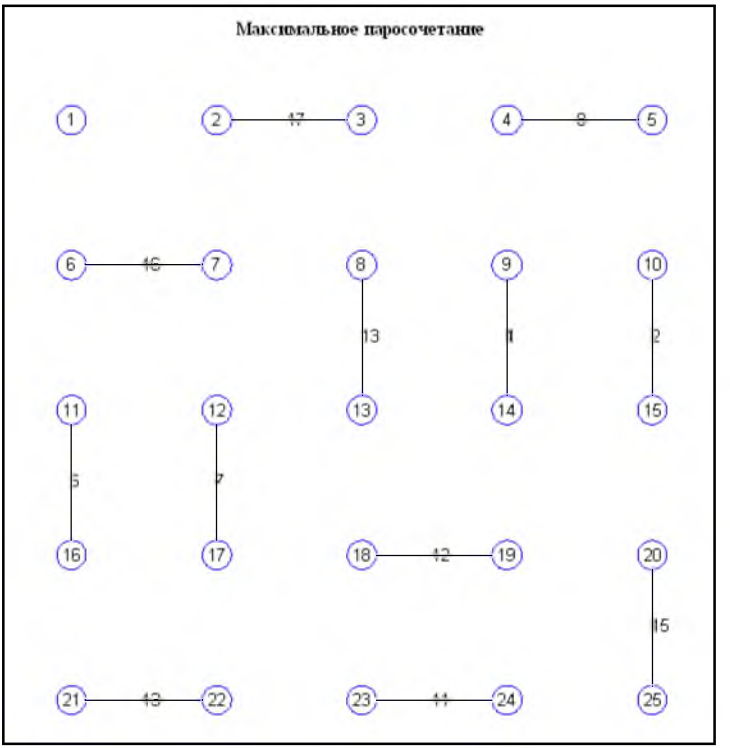


Рис. 29.3. Максимальное паросочетание, выполненное с помощью MATLAB

Вначале решим обычную (невзвешенную) задачу о максимальном паросочетании. Напечатаем количество ребер, их номера и общий вес. Нарисуем результат (рис. 29.3): граф со всеми вершинами, но только с теми ребрами, которые входят в найденное максимальное паросочетание.

```
nMM=MaxMatch(E(:,1:2)); % решаем задачу
fprintf('Количество ребер в паросочетании = %d\n',length(nMM));
disp('В паросочетание вошли ребра с номерами:');
fprintf('%d  ',nMM);
fprintf('\nОбщий вес = %d\n',sum(E(nMM,3)));
```

```
PlotGraph(V(:,1:2),E(nMM,:),'g'); % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfМаксимальное паросочетание')
Количество ребер в паросочетании = 12
В паросочетание вошли ребра с номерами:
2   4   12   18   19   20   26   27   33   41   43   45
Общий вес = 121
```

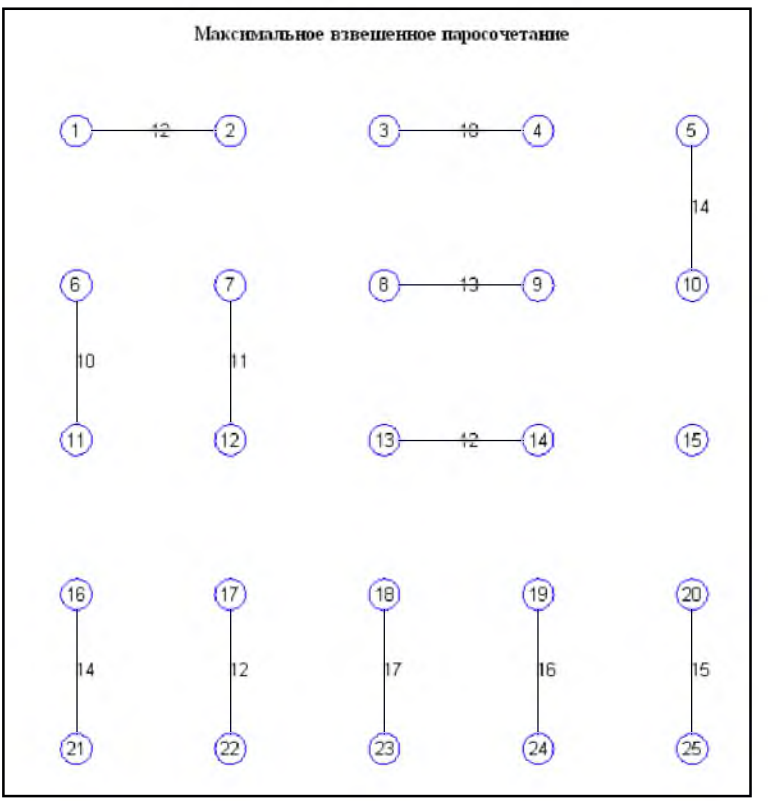


Рис. 29.4. Максимальное взвешенное паросочетание, выполненное с помощью MATLAB

Теперь решим задачу о максимальном взвешенном паросочетании. Напечатаем те же результаты. Нарисуем граф с ребрами, вошедшими в найденное паросочетание (рис. 29.4).

```
nMM=MaxMatch(E); % решаем задачу
fprintf('Количество ребер в паросочетании = %d\n',length(nMM));
disp('В паросочетание вошли ребра с номерами:');
fprintf('%d  ',nMM);
```

```
fprintf('\nОбщий вес = %d\n',sum(E(nMM,3)));
PlotGraph(V(:,1:2),E(nMM,:)); % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfМаксимальное взвешенное паросочетание')
Количество ребер в паросочетании = 12
В паросочетание вошли ребра с номерами:
1  3  11  14  16  17  24  35  36  39  40  41
Общий вес = 156
```

В обоих вариантах максимальное паросочетание состоит из 12 ребер, но во втором случае общий вес ребер выше.

29.2. Минимальное вершинное покрытие

При решении этой задачи будем руководствоваться теми же соображениями, что и в предыдущем разделе: меньше своего кода, больше готового.

29.2.1. Основные определения и постановка задачи

Определение 29.2. Подмножество вершин $V_1 \subseteq V$ графа $G=(V, E)$ называется *вершинным покрытием*, если они инцидентны всем ребрам. Вершинное покрытие называется *минимальным по включению*, если любое его подмножество с меньшим числом вершин не является вершинным покрытием. Вершинное покрытие называется *минимальным*, если оно состоит из минимально возможного количества вершин. □

Одной из классических задач теории графов является нахождение минимального вершинного покрытия. Если вершины взвешены, то можно говорить о минимальном взвешенном вершинном покрытии, когда требуется минимизировать общий вес вершин при условии, что они должны быть инцидентны всем ребрам.

ПРИМЕР 29.2. На рис. 29.5, *а* показан тот же самый граф с $n=8$ и $m=13$, что и на рис. 29.1, *а*. Множество всех вершин V, очевидно, образует вершинное покрытие, т. к. они инцидентны всем ребрам. Если мы отбросим вершину v_4, то оставшиеся вершины также покрывают все ребра. Но это покрытие не является минимальным по включению: мы можем выбросить из него еще и вершину v_5. Оставшееся подмножество $\{v_1, v_2, v_3, v_6, v_7, v_8\}$ является минимальным по включению вершинным покрытием: из него уже нельзя выбросить ни одной вершины, чтобы не оголить какое-нибудь ребро. Оно показано на рис. 29.5, *б*. Но если удалять из V вершины по-другому, то можно добиться, что не 6, а только 5 вершин будут покрывать все ребра. Одно из возмож-

ных минимальных вершинных покрытий показано на рис. 29.5, *в*. Как мы ни будем стараться, меньшим количеством вершин обойтись не удастся. □

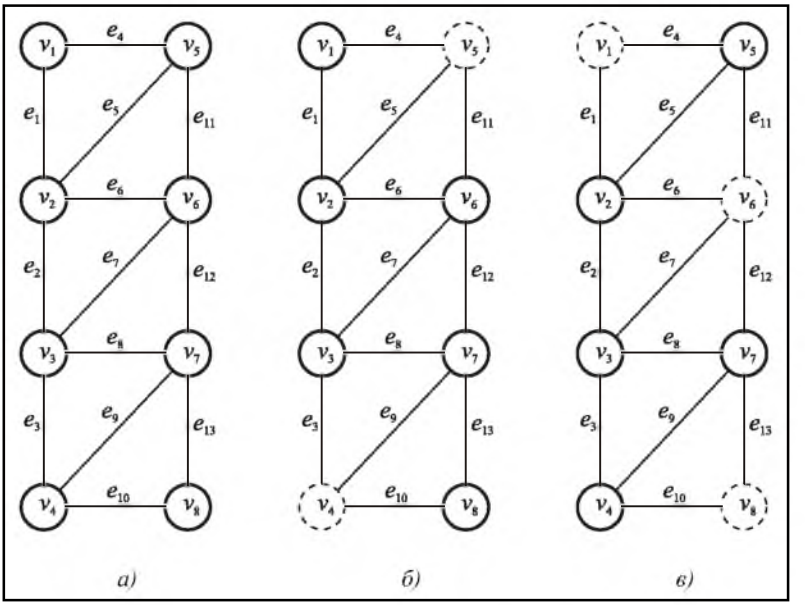


Рис. 29.5. Нахождение минимального вершинного покрытия: *а* — граф; *б* — минимальное по включению вершинное покрытие; *в* — минимальное вершинное покрытие

К нахождению минимального вершинного покрытия сводится, например, военная задача о разрушении всех коммуникаций противника путем нанесения минимального количества ударов по его опорным пунктам.

29.2.2. Сведение к задаче ЦЛП

В отличие от задачи о максимальном паросочетании, для которой существуют полиномиальные алгоритмы решения, задача о минимальном вершинном покрытии является NP-полной [33]. Тем не менее, как показывает опыт, с помощью процедуры `milp` она решается достаточно быстро. Сформулируем ее в терминах ЦЛП. Введем в рассмотрение вектор-столбец v длиной n. Этот вектор будет *ассоциированным* с вершинами: если какая-либо вершина v_i войдет в вершинное покрытие, то соответствующая переменная v_i примет значение 1, а если нет — то 0. То есть координаты вектора v — целочисленные и могут принимать только одно из двух возможных значений — 0 или 1:

$$\begin{cases} v_i = 0 \vee 1; \\ i = \overline{1,n}. \end{cases} \tag{29.8}$$

Требуется минимизировать общее количество вершин:

$$t = \sum_{i=1}^{n} v_i = (\mathbf{1}, \mathbf{v}) \tag{29.9}$$

при условии, что эти вершины инцидентны всем ребрам. Мы должны для каждого ребра найти номера инцидентных ему вершин и убедиться, что хотя бы одна из них (или обе) входит в искомое минимальное вершинное покрытие. Переходя от вершин к ассоциированным переменным, получаем: сумма переменных v_i, ассоциированных с вершинами v_i, инцидентными каждому ребру, должна быть не меньше 1. Например, для графа, показанного на рис. 29.5, *а*, эта система неравенств имеет вид:

$$\begin{cases} v_1 + v_2 \ge 1; & v_1 + v_5 \ge 1; & v_3 + v_6 \ge 1; & v_4 + v_8 \ge 1; \\ v_2 + v_3 \ge 1; & v_2 + v_5 \ge 1; & v_3 + v_7 \ge 1; & v_5 + v_6 \ge 1; & v_7 + v_8 \ge 1. \\ v_3 + v_4 \ge 1; & v_2 + v_6 \ge 1; & v_4 + v_7 \ge 1; & v_6 + v_7 \ge 1; \end{cases} \tag{29.10}$$

Для экономии места здесь все 13 уравнений выписаны не в один, а в несколько столбиков. С помощью матрицы инцидентности A эта система записывается так:

$$\mathrm{A}^{\mathrm{T}}\mathrm{v} \ge 1. \tag{29.11}$$

Имеем задачу ЦЛП: минимизировать целевую функцию (29.9) при ограничениях-неравенствах (29.11) и целочисленных переменных (29.8):

$$\begin{cases} t = (\mathbf{1}, \mathbf{v}) \to \min; \\ \mathbf{A}^{\mathrm{T}}\mathbf{e} \ge \mathbf{1}; \\ v_i = 0 \vee 1;\ i = \overline{1,n}. \end{cases} \tag{29.12}$$

Задача о минимальном взвешенном вершинном покрытии отличается от (29.12) только коэффициентами при целевой функции: вместо единиц будут веса вершин. Если вектор-столбец весов вершин обозначить d, то задача о минимальном взвешенном вершинном покрытии формулируется так:

$$\begin{cases} t = (\mathbf{d}, \mathbf{v}) \to \min; \\ \mathbf{A}^{\mathrm{T}}\mathbf{e} \ge \mathbf{1}; \\ v_i = 0 \vee 1;\ i = \overline{1,n}. \end{cases} \tag{29.13}$$

29.2.3. Описание процедуры *MinVerCover*

При решении задачи в MATLAB нужно задать список ребер графа и, возможно, веса вершин. Будем предполагать, что, если пользователь задал толь-

ко один входной параметр — список вершин, то нужно решать невзвешенную задачу (29.12), а если задал дополнительно веса вершин — то взвешенную (29.13). Выходным параметром будет список вершин, входящих в минимальное (взвешенное) вершинное покрытие. Вот как выглядит заголовок и справочная информация к этой процедуре (здесь и во всех следующих главах для экономии места сведения об авторе опущены):

```
function nMC=MinVerCover(E,d)
% Функция nMC=MinVerCover(E,d) решает задачу
% о минимальном вершинном покрытии.
% Входные параметры:
%   E(m,2) - список ребер графа;
%     1-й и 2-й элементы каждой строки - это номера вершин;
%     m - количество ребер.
%   d(n) (необязательный параметр) - веса вершин,
%     n - количество вершин.
%     Если задан только 1-й параметр E, то все d=1.
% Выходной параметр:
%   nMC - список номеров вершин, включенных
%     в минимальное (взвешенное) вершинное покрытие.
% Используется сведение к задаче ЦЛП.
% Необходимые другие функции: MILP.M.
% Эта функция может быть свободно скопирована с сайта:
% Matlog: Logistics Engineering Matlab Toolbox,
% http://www.ie.ncsu.edu/kay/matlog/.
```

Далее, как обычно, следует проверка исходных данных. Вначале проверяем вообще их наличие:

```
if nargin<1,
  error('Нет исходных данных!')
end
```

Находим размеры первого входного параметра. Проверяем, чтобы этот массив был двумерным и содержал не менее двух столбцов:

```
sz=size(E); % размер массива E
if length(sz)~=2,
  error('Массив E должен быть 2-мерным!')
end
if (sz(2)<2),
  error(' Массив E должен иметь 2 столбца!'),
end
```

Если хотя бы 2 столбца есть, проверяем их на положительность и целочисленность:

```
if ~all(all(E(:,1:2)>0)),
  error('1-й и 2-й столбцы массива E должны быть положительными!')
end
if ~all(all((E(:,1:2)==round(E(:,1:2))))),
  error('1-й и 2-й столбцы массива E должны быть целыми!')
end
```

Проверка первого входного параметра закончена. Находим из него размер и мощность графа.

```
m=size(E,1); % количество ребер
n=max(max(E(:,1:2))); % количество вершин
```

Исследуем теперь второй входной параметр. Если его нет, задаем все веса вершин единичными. Если же пользователь задал веса, проверяем, достаточна ли длина заданного вектора. Если нет, выдаем сообщение об ошибке, а если достаточна, формируем вектор-столбец весов вершин.

```
if nargin<2, % задан только 1 входной параметр
  d=ones(n,1); % все веса =1
else
  d=d(:); % преобразуем в вектор-столбец
  if length(d)<n, % недостаточная длина
    error('Длина вектора d недостаточна!')
  else
    d=d(1:n); % первые n чисел
  end
end
```

Теперь у нас сформирован вектор коэффициентов при неизвестных в целевой функции t. Строим матрицу инцидентности A, вектор правых частей системы неравенств (29.11), нижние и верхние границы по переменным (29.8):

```
A=zeros(n,m); % для матрицы инцидентности
for k=1:m, % просматриваем все ребра
  A(E(k,1:2),k)=1; % заполняем матрицу инцидентности
end
b=ones(m,1); % правые части системы ограничений-неравенств
vlb=zeros(1,n); % нижние границы
vub=ones(1,n); % верхние границы
```

Заключительный этап — решение задачи ЦЛП и возврат результатов.

```
xmin=milp(d,-A',-b,vlb,vub); % решаем задачу ЦЛП
nMC=find(round(xmin)); % ответ - номера вершин
return
```

Как и в предыдущей процедуре, нужно округлить решение, полученное из функции `milp`.

29.2.4. Пример обращения к процедуре *MinVerCover*

Проиллюстрируем применение этой функции на графе с 11 вершинами и 22 ребрами. Этот пример взят из [33]. Введем исходные данные. Нарисуем исходный граф со взвешенными вершинами (рис. 29.6).

```
clear all
V=[[0 0 2];[1 1 3];[1 0 3];[1 -1 4];...
   [2 1 1];[2 0 2];[2 -1 3];[3 1 4];...
   [3 0 5];[3 -1 1];[4 0 5]]; % координаты вершин и их веса
E=[[1 2];[1 3];[1 4];[2 3];[3 4];[2 5];...
   [2 6];[3 6];[3 7];[4 7];[6 5];[6 7];...
   [5 8];[6 8];[6 9];[7 9];[7 10];[8 9];...
   [9 10];[8 11];[9 11];[10 11]]; % ребра
PlotGraph(V,E); % plot the digraph
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfИсходный граф со взвешенными вершинами')
```

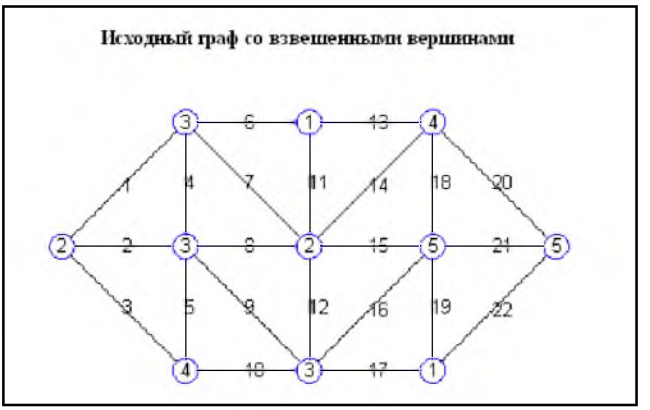


Рис. 29.6. Исходный граф со взвешенными вершинами, выполненный с помощью MATLAB

Решаем невзвешенную задачу. Печатаем количество вершин, их номера и общий вес. Рисуем вершины, вошедшие в найденное минимальное вершинное покрытие (рис. 29.7).

```
nMC=MinVerCover(E);
fprintf('Количество вершин в минимальном вершинном покрытии = %d\n',...
  length(nMC));
disp('В минимальное вершинное покрытие вошли вершины с номерами:');
fprintf('%d  ',nMC);
fprintf('\nОбщий вес = %d\n',sum(V(nMC,3)));
```

```
PlotGraph(V(nMC,:)); % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfМинимальное вершинное покрытие')
Количество вершин в минимальном вершинном покрытии = 7
В минимальное вершинное покрытие вошли вершины с номерами:
2 3 4 6 8 9 10
Общий вес = 22
```

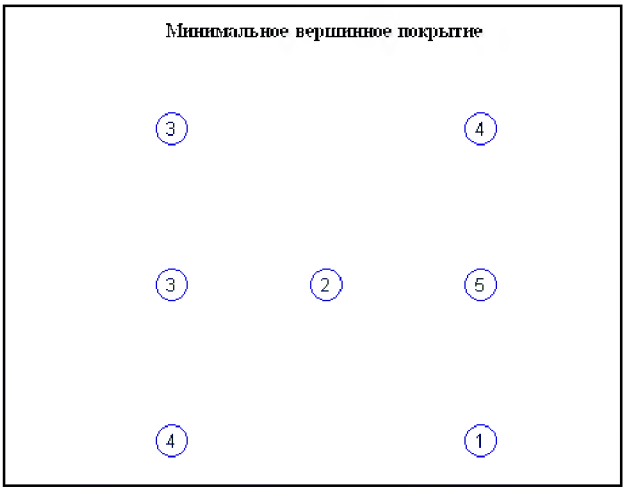


Рис. 29.7. Минимальное вершинное покрытие, найденное с помощью MATLAB

Теперь решаем взвешенную задачу. Выводим те же результаты. Вершины показаны на рис. 29.8.

```
nMC=MinVerCover(E,V(:,3));
fprintf(['Количество вершин в минимальном взвешенном '...
  'вершинном покрытии = %d\n'],length(nMC));
disp(['В минимальное взвешенное вершинное покрытие '...
  'вошли вершины с номерами:']);
fprintf('%d  ',nMC);
fprintf('\nОбщий вес = %d\n',sum(V(nMC,3)));
PlotGraph(V(nMC,:)); % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfМинимальное взвешенное вершинное покрытие')
Количество вершин в минимальном взвешенном вершинном покрытии = 8
В минимальное взвешенное вершинное покрытие вошли вершины с номерами:
1  3  5  6  7  8  10  11
Общий вес = 21
```

Рис. 29.8. Минимальное взвешенное вершинное покрытие, найденное с помощью MATLAB

Посмотрите на разницу в решениях. В первом случае (невзвешенная задача) мы стремимся уменьшить общее количество вершин. В результате оказалось достаточно 7 вершин из 11, чтобы покрыть все ребра. Во втором случае минимизировался суммарный вес вершин, а не их число. В итоге получили 8 вершин, суммарный вес которых меньше, чем у 7 вершин в первом случае.

29.3. Немного о двойственности

Сравним две невзвешенные задачи: о максимальном паросочетании (29.5) и о минимальном вершинном покрытии (29.12):

$$\begin{cases} z = (\mathbf{1}, \mathbf{e}) \to \max; \\ \mathbf{Ae} \le \mathbf{1}; \\ e_k = 0 \vee 1;\ k = \overline{1,m}; \end{cases} \quad \begin{cases} t = (\mathbf{1}, \mathbf{v}) \to \min; \\ \mathbf{A}^{\mathrm{T}}\mathbf{e} \ge \mathbf{1}; \\ v_i = 0 \vee 1;\ i = \overline{1,n}. \end{cases} \tag{29.14}$$

Если отказаться от требования целочисленности (третья строка в каждой системе) и заменить его требованием неотрицательности: $\forall e_k \ge 0;\ \forall v_i \ge 0$; то получим пару взаимно двойственных задач ЛП [33]:

$$\begin{cases} z = (\mathbf{1}, \mathbf{e}) \to \max; \\ \mathbf{Ae} \le \mathbf{1}; \\ e_k \ge 0;\ k = \overline{1,m}; \end{cases} \quad \begin{cases} t = (\mathbf{1}, \mathbf{v}) \to \min; \\ \mathbf{A}^{\mathrm{T}}\mathbf{e} \ge \mathbf{1}; \\ v_i \ge 0;\ i = \overline{1,n}. \end{cases} \tag{29.15}$$

Для таких пар имеют место ряд теорем, которые обычно изучаются студентами в курсе математического программирования. В частности, доказывается,

что если существует решение одной из задач, то существует решение и другой, и в этом случае $z_{max} = t_{min}$. Исходя из структуры матрицы инцидентности A (она состоит из нулей и единиц), мы можем быть уверены, что решения обеих задач (29.15) наверняка существуют. Значит, условие $z_{max} = t_{min}$ для (29.15) выполняется. Обозначим это число:

$$z_{max} = t_{min} = \omega_{opt}. \tag{29.16}$$

В наших задачах (29.14) по сравнению с (29.15) области допустимых решений сужены. Поэтому z_{max}, возможно, не достигнет ω_{opt}, а будет меньше его, а t_{min}, возможно, будет больше ω_{opt}. Поэтому для пары задач (29.14) имеет место неравенство:

$$z_{max} \le t_{min}. \tag{29.17}$$

✔ Количество ребер в максимальном паросочетании никогда не может превысить количества вершин в минимальном вершинном покрытии.

ПРИМЕР 29.3. Для графа, изображенного на рис. 29.1, *а* и 29.5, *а*, $z_{max} = 4$, а $t_{min} = 5$. □

Рассмотрим теперь взвешенные задачи (29.6) и (29.13). Как бы мы ни взвешивали ребра, при построении паросочетания требование неинцидентности ребер остается в силе. Поэтому число ребер во взвешенном максимальном паросочетании никогда не будет больше, чем в невзвешенном. Аналогично, при построении вершинного покрытия, какими бы ни были веса вершин, они все равно должны покрывать все ребра. Поэтому количество вершин во взвешенном минимальном вершинном покрытии никогда не будет меньше, чем в невзвешенном. Значит, и во взвешенных задачах мы можем сделать тот же вывод:

✔ Количество ребер в максимальном взвешенном паросочетании никогда не может превысить количества вершин в минимальном взвешенном вершинном покрытии.

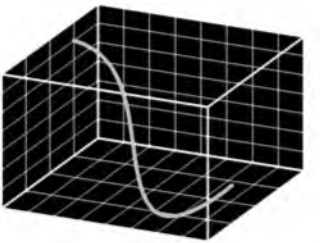
Но неравенство (29.17) во взвешенных задачах в общем случае уже не имеет места, т. к. теперь в вычислении z_{max} и t_{min} участвуют веса ребер и вершин.

29.4. Вопросы для самопроверки

1. Проверьте функцию `MaxMatch` на графе с рис. 29.1, *а*. Какой результат получился? Является ли это решение единственным?
2. Является ли используемый нами алгоритм нахождения максимального паросочетания полиномиальным?

3. Существуют ли вообще полиномиальные алгоритмы решения задачи о максимальном паросочетании? Какова их полиномиальная сложность?
4. Существуют ли полиномиальные алгоритмы нахождения *всех* максимальных паросочетаний? Сколько существует максимальных паросочетаний для полного двудольного графа $G=(V,W,E)$, у которого $|V|=|W|=n$; $|E|=m=n^2$?
5. Как выяснить, входит ли данное ребро в *какое-либо* максимальное паросочетание? Является ли алгоритм решения этой задачи полиномиальным?
6. Переделайте функцию `MaxMatch` так, чтобы она работала для гиперграфов.
7. Какую еще проверку исходных данных можно организовать в процедурах `MaxMatch` и `MinVerCover`?
8. Сформулируйте в терминах ЦЛП задачу: найти подмножество вершин максимальной мощности (максимального веса), среди которых нет смежных. Составьте процедуру ее решения.
9. Проверьте выполнение условия (29.17) для своего варианта ИДЗ.

ГЛАВА 30



Жадные алгоритмы и минимальные остовные деревья

В этой главе мы построим минимальное остовное дерево (МОД) связного графа. Будет доказано, что для решения этой задачи можно применить очень простой алгоритм, который называется жадным. Поэтому вначале выясним, что он из себя представляет.

30.1. Жадность помогает и губит

Рассмотрим 2 примера.

ПРИМЕР 30.1. Требуется найти максимум линейной функции $z=(c, x)\to\max$ на перестановке P_n значений ее аргументов x. Вот численный пример:

$$z=4x_1-3x_2-6x_3+2x_4\to\max, \qquad (30.1)$$

где вектор аргументов (x_1, x_2, x_3, x_4) — одна из перестановок чисел (2, 3, 4, 5). На какие места в векторе x нужно поставить числа 2, 3, 4 и 5, чтобы максимизировать функцию (30.1)? Попробуем применить для решения задачи такой простой алгоритм. Мы видим, что коэффициент при x_1 максимальный: он равен 4. Поэтому присвоим переменной x_1 максимальное из имеющихся значений: 5. Следующий по величине коэффициент — это 2 при x_4, поэтому переменной x_4 присваиваем максимальное из оставшихся значений 4. Далее переменной x_2 присваиваем значение 3, а x_3 – 2. Этот процесс показан на рис. 30.1. Получаем $z=7$. Можно перебрать все 24 перестановки и убедиться, что действительно найдено максимальное значение. Доказательство этой теоремы для перестановок и размещений в общем виде есть в [34]. □

Алгоритм, который мы применили для решения этого примера, называется *жадным*. Его общий принцип так же прост, как человеческая жадность: бери

как можно больше из того, что осталось. Попытаемся применить этот алгоритм в следующем примере.

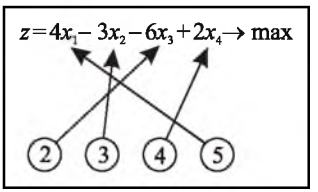


Рис. 30.1. Максимизация линейной функции на множестве перестановок значений ее аргументов

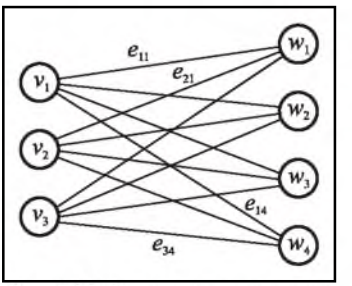


Рис. 30.2. Задача о назначениях

ПРИМЕР 30.2. Задача о максимальном взвешенном паросочетании на полном двудольном графе $G=(V,W,E)$ с $|V|=3$; $|W|=4$; $|E|=12$. Этот граф показан на рис. 30.2. Будем нумеровать ребра двойными индексами: первая цифра — номер вершины из V, а вторая — из W. Чтобы не затенять чертеж, на рис. 30.2 показаны номера только некоторых ребер. Матрица весов ребер:

$$\mathbf{C}=\begin{pmatrix}8 & 7 & 7 & 6\\ 7 & 7 & 5 & 8\\ 8 & 6 & 6 & 7\end{pmatrix}. \qquad (30.2)$$

Воспользуемся жадным алгоритмом. Мы видим, что 1-й работник лучше всего справится с 1-й работой, поэтому включим в паросочетание ребро e_{11}. Второго работника посылаем на 4-ю работу, т. к. именно на ней его производительность максимальна: добавляем ребро e_{24}. Теперь у 3-го работника остался небольшой выбор: или 2-я, или 3-я работы с одинаковой производительностью 6. Выберем, например, e_{32}. Таким образом, построено максимальное по включению взвешенное паросочетание $\{e_{11}, e_{24}, e_{32}\}$ с общим весом 22. Но это решение — не наилучшее. Можно взять $\{e_{12}, e_{24}, e_{31}\}$ с общим весом 23 и показать, что это действительно будет максимальное взвешенное паросочетание. Здесь жадный алгоритм дал осечку: после построения начального плана методом максимальной стоимости понадобилась еще и переброска работников с работы на работу (в теории транспортных задач эта операция называется переброской груза по циклу). □

Почему же в первом примере жадный алгоритм привел к успеху, а во втором нет? Ответ на этот вопрос дает структура области допустимых значений. В первом примере, когда на очередном этапе решения задачи мы забирали одно значение из множества оставшихся, мы забирали только его, а остальные

ные оставались в неприкосновенности. Во втором же примере, когда на 1-м этапе мы забрали ребро e_{11}, то мы забрали на самом деле не только его. Мы лишили себя возможности на следующих этапах забирать все ребра, инцидентные v_1 и w_1. А именно e_{12} и e_{31}, как выясняется, входят в одно из максимальных взвешенных паросочетаний.

Структуры, для которых работает жадный алгоритм, называются в математике матроидами. Существует около полусотни различных определений матроидов. Одно из них так и формулируется: матроиды — это структуры, на которых работает жадный алгоритм. В нашу задачу не входит изучение теории матроидов. Если вы заинтересуетесь этой проблемой, то на поисковых серверах в Интернете можно найти обширную библиографию по данной тематике. Мы только рассмотрим одну важную задачу теории графов, которая является задачей на матроидах, т. к. для ее решения можно будет применить жадный алгоритм.

30.2. Остовное дерево минимального веса

Здесь мы будем рассматривать графы и мультиграфы, но не псевдографы.

Определение 30.1. *Путь (маршрут)* в графе $G=(V,E)$ — это последовательность вершин и ребер вида $v_1e_1v_2e_2...v_k$, в которой соседние элементы инцидентны. □

Определение 30.2. Маршрут называется *простым*, если каждая вершина встречается в нем только один раз. □

В простом маршруте нет пересечений (повторяющихся вершин) и, следовательно, не может быть повторяющихся ребер.

Определение 30.3. Граф $G=(V,E)$ называется *связным*, если существует путь из любой его вершины в любую другую. □

До сих пор все примеры, которые мы рассматривали, — это были связные графы. В несвязных графах можно выделить отдельные связные компоненты. В самом крайнем случае, когда у графа вообще нет ребер: $E=\varnothing$, у него будет n компонент — по одной вершине в каждой. Здесь мы будем рассматривать только связные графы.

Рассмотрим связный граф (или мультиграф) G. Если из него удалить некоторые ребра, он может остаться связным, а может и распасться на отдельные компоненты. Сколько (по максимуму) ребер и какие можно удалить, чтобы граф остался связным? Если задан граф со взвешенными ребрами, то можно поставить задачу так: как по максимуму удалить ребра максимального веса, чтобы остался связный граф с общим минимальным весом ребер?

Чтобы подойти к решению этой задачи, сформулируем вопрос по-другому: какое минимальное количество ребер необходимо для связности графа с n вершинами?

■ **ТЕОРЕМА 30.1.** В связном графе $G=(V,E)$ выполняется: $m \geq n-1$, т. е. минимально возможное число ребер связного графа равно $n-1$.

Доказательство. При $n=1$ говорить о связности вообще нет смысла: одну вершину не с чем связывать. Можно сказать, что одна вершина связана с собой нулевым количеством ребер. Две вершины можно соединить одним ребром — и граф станет связным. Третью вершину можно подсоединить с помощью еще одного, уже второго ребра. По индукции: пусть теорема верна для $n=k$, т. е. граф с k вершинами и $k-1$ ребрами связный. Очередную, $(k+1)$-ю вершину можно подсоединить к нему с помощью k-го ребра. Полученный граф с $k+1$ вершинами и k ребрами тоже будет связным. По индукции теорема доказана. □

Эта теорема показывает, сколько (минимум) ребер нужно оставить в графе, чтобы он оставался связным.

Определение 30.4. Связный граф $G=(V,E)$ с n вершинами и $n-1$ ребрами называется *остовным деревом*. □

На рис. 30.3 показан граф (*а*) и некоторые из его остовных деревьев (*б*, *в*).

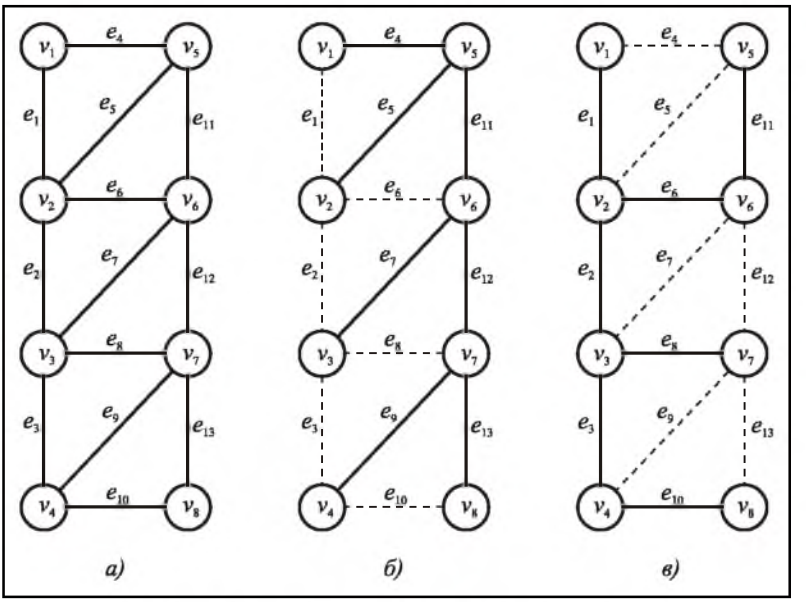


Рис. 30.3. *а* — граф; *б*, *в* — его остовные деревья

Если связный граф $G=(V,E)$ не является остовным деревом, то, очевидно, из него можно удалить некоторые ребра так, что оставшиеся ребра $E_1 \subseteq E$ вместе с множеством вершин V образуют остовное дерево $G_1=(V,E_1)$. Остовное дерево обладает интересными свойствами, некоторые из которых мы докажем.

Определение 30.5. Путь $v_1e_1v_2e_2...v_k$ называется *циклом*, если в нем начальная и конечная вершины совпадают. □

Определение 30.6. Цикл называется *простым*, если в определяющем его пути каждая промежуточная вершина встречается только один раз. □

Простой цикл — это цикл без самопересечений. Если первую и последнюю вершины считать за одну, то в нем нет повторяющихся вершин, а значит, нет и повторяющихся ребер. Очевидно, из любого непростого цикла можно выделить, по крайней мере, 2 разных простых цикла, а из непростого маршрута — по крайней мере 1 простой цикл.

■ **ТЕОРЕМА 30.2.** В остовном дереве $G_1=(V,E_1)$ связного графа $G=(V,E)$ нет циклов.

Доказательство. От противного: если в G_1 есть хотя бы один цикл, то можно без ущерба для связности удалить одно из ребер этого цикла: путь из любой вершины в любую можно при необходимости организовать по оставшейся части цикла. Значит, в G_1 больше, чем $n-1$ ребер: удаление из него одного ребра не нарушает связности. Следовательно, G_1 не является остовным деревом. □

■ **ТЕОРЕМА 30.3.** В остовном дереве G_1 связного графа существует единственный путь из каждой вершины в каждую другую.

Доказательство. От противного: если бы из некоторой вершины v_i существовало два различных пути в v_j, то существовал бы цикл $v_i...v_j...v_i$, что противоречит теореме 30.2. □

Можно доказать и обратные теоремы и установить тем самым эквивалентность следующих утверждений:

- остовное дерево — это связный граф с $|E|=|V|-1$;
- остовное дерево — это связный граф без циклов;
- остовное дерево — это связный граф, в котором существует единственный путь из любой вершины в любую другую.

Если посмотреть на рис. 30.3, можно увидеть, что в остовных деревьях действительно нет циклов, и из любой вершины можно перейти в любую другую по единственному пути.

Обычно в приложениях ставится задача нахождения остовного дерева минимального веса (в дальнейшем — МОД, минимальное остовное дерево). Такая

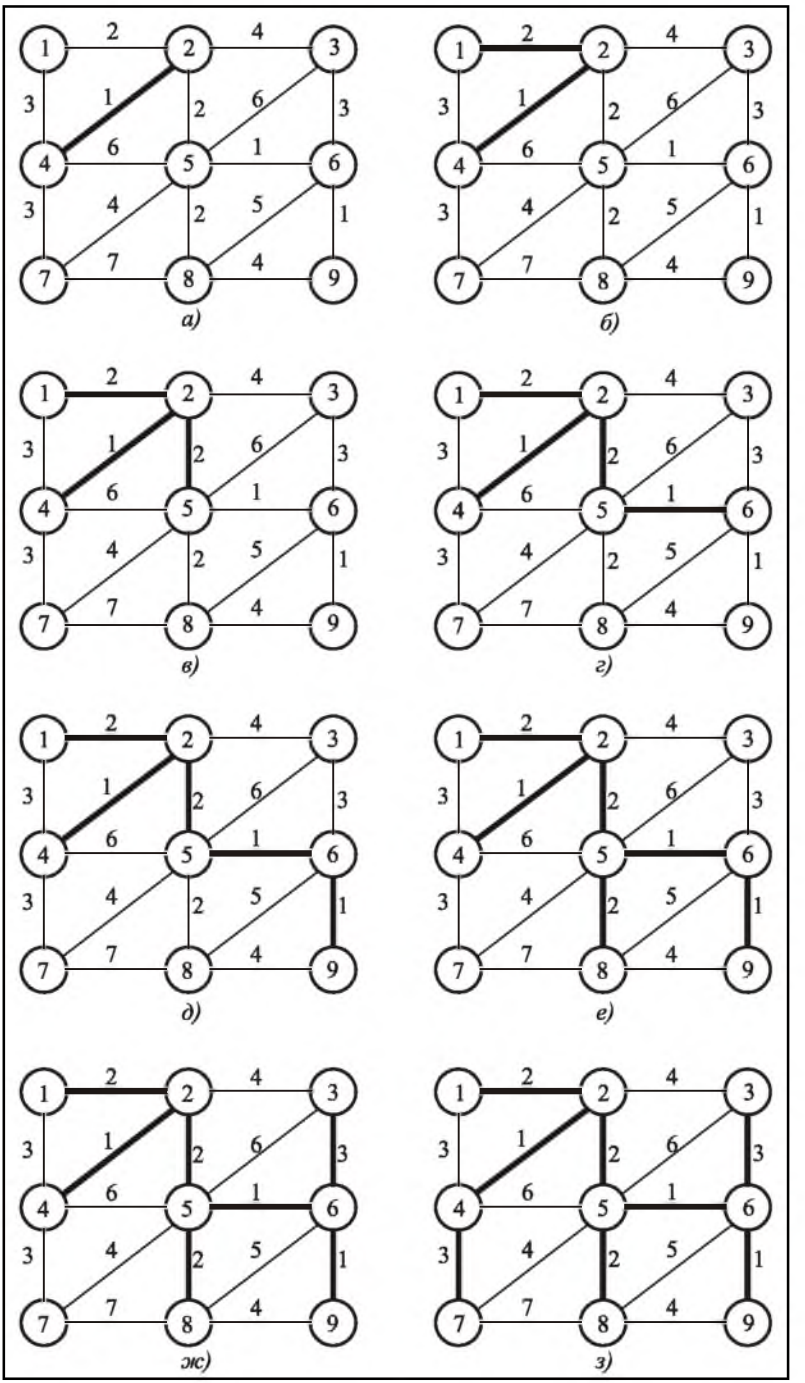


Рис. 30.4. Схема 1 построения МОД при помощи жадного алгоритма

задача возникает, например, при проектировании линий электропередач: для прокладки линий выбирают участки с минимальной стоимостью работ, обеспечивающие связность сети.

Исследуем возможности применения жадного алгоритма для нахождения МОД. Вначале мы рассмотрим две схемы алгоритма, а затем докажем, что обе они приводят к одному и тому же правильному результату. Основной принцип построения МОД: мы будем не *удалять* ребра из исходного графа, а *добавлять* их в строящееся МОД.

Первая схема. Она показана на рис. 30.4. Здесь в вершинах проставлены их номера, а возле ребер — веса. Будем обозначать каждое ребро буквой e с двумя индексами — номерами соединяемых вершин. В нашем графе $n=9$, $m=16$, поэтому в построенном МОД должно остаться только 8 ребер из 16. Выберем первое попавшееся ребро минимального веса. У нас минимальный вес ребер 1, и первым в списке оказалось e_{24}. Поэтому начинаем построение МОД с него (*а*). Далее просматриваем все ребра, инцидентные уже построенному фрагменту МОД, т. е. к e_{24}. Всего таких ребер 6 (по 3 в вершинах v_2 и v_4). Выбираем из них ребро минимального веса. Минимальный вес среди этих 6 ребер — 2, он есть у e_{12} и e_{25}. Первым в списке идет e_{12}, его и подсоединяем к строящемуся МОД (*б*). Продолжаем просматривать ребра, инцидентные к уже построенному фрагменту МОД и не образующие в нем циклов. Берем все ребра, инцидентные к v_1, v_2 и v_4 (кроме v_{14} — его нельзя брать, т. к. образуется цикл), и выбираем из них ребро минимального веса. Минимальный вес 2 — у ребра e_{25}, его и присоединяем (*в*). Продолжаем процесс. Теперь у нас появилась возможность присоединить к МОД ребро e_{56} веса 1 (*г*), а затем — e_{69} (*д*). Среди оставшихся ребер минимальный вес 2 — у ребра e_{58}, его также можно присоединить к МОД (*е*). И, наконец, присоединяем ребра e_{36} и e_{47} веса 3 (*ж*, *з*). Получили 8 ребер — МОД построено. Его вес 15. □

Вторая схема. Она показана на рис. 30.5. Основное ее отличие от первой схемы — мы будем добавлять в строящееся МОД не обязательно инцидентные ребра. Главное, чтобы не образовывалось циклов. Просматриваем все ребра минимального веса 1. Начинаем с e_{24} — оно встретилось первым (*а*). Следующее ребро веса 1 — это e_{56}, и оно не образует циклов — добавляем его (*б*). Затем проверяем e_{69}: циклов нет, присоединяем (*в*). Все ребра веса 1 исчерпаны. Переходим к оставшимся ребрам минимального веса 2. Ребро e_{12} можно присоединить (*г*), e_{25} — тоже (*д*), e_{58} тоже подходит (*е*). Далее — ребра веса 3. Ребро e_{14} не годится (образуется цикл), а вот e_{36} подходит (*ж*), и e_{47} — тоже (*з*). В результате получили то же самое МОД веса 15, что и по схеме 1. □

Как видим, эти схемы отличаются порядком присоединения ребер. В 1-й схеме мы все время присоединяем инцидентные ребра, т. е. наращиваем МОД,

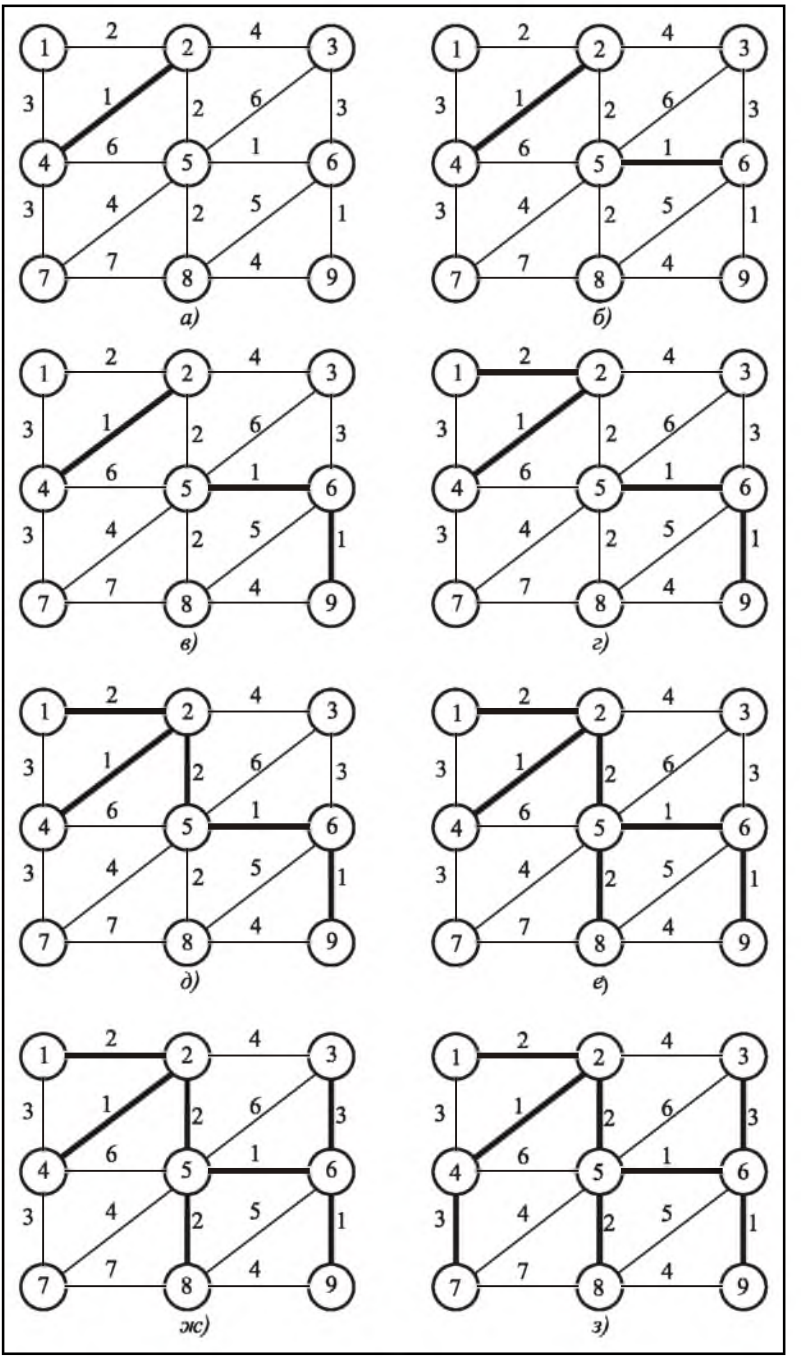


Рис. 30.5. Схема 2 построения МОД при помощи жадного алгоритма

оставляя его связным. Во 2-й схеме строятся отдельные куски МОД, которые затем соединяются. Каждый из этих кусков называется *деревом*, а их совокупность — *лесом*.

Определение 30.7. *Деревом* называется остовное дерево, построенное на некотором подмножестве вершин $V_1 \subseteq V$ и инцидентным ко всем им ребрам. □

Определение 30.8. *Лесом* называется множество непересекающихся деревьев. □

Например, лес на рис. 30.5, *г* состоит из деревьев, построенных на подмножествах вершин $\{v_1, v_2, v_4\}$, $\{v_3\}$, $\{v_5, v_6, v_9\}$, $\{v_7\}$ и $\{v_8\}$. В подмножествах из одной вершины никаких ребер нет, и соответствующее дерево состоит только из вершины.

Сейчас мы сформулируем и докажем теорему, которая устанавливает применимость жадного алгоритма к построению МОД. Заметим вначале, что МОД может быть не единственным. Если в графе много ребер одинакового малого веса, то, вполне возможно, существует несколько МОД. В частности, если все ребра имеют одинаковый вес, то любое остовное дерево будет минимальным. Поэтому мы будем говорить не о МОД, а об одном из МОД или любом МОД.

■ **ТЕОРЕМА 30.4.** Пусть на непересекающихся подмножествах вершин V_1, V_2, ..., V_k построены деревья минимального веса $G_1=(V_1, E_1)$, $G_2=(V_2, E_2)$, ..., $G_k=(V_k, E_k)$. Пусть также e_j — ребро минимального веса, один конец которого входит в G_1 (т. е. инцидентен какой-либо вершине из V_1), а другой — в какое-либо другое дерево: G_2, G_3, ..., G_k. Тогда среди всех деревьев минимального веса, построенных на объединении ребер $\bigcup E_i$, существует дерево минимального веса, содержащее ребро e_j.

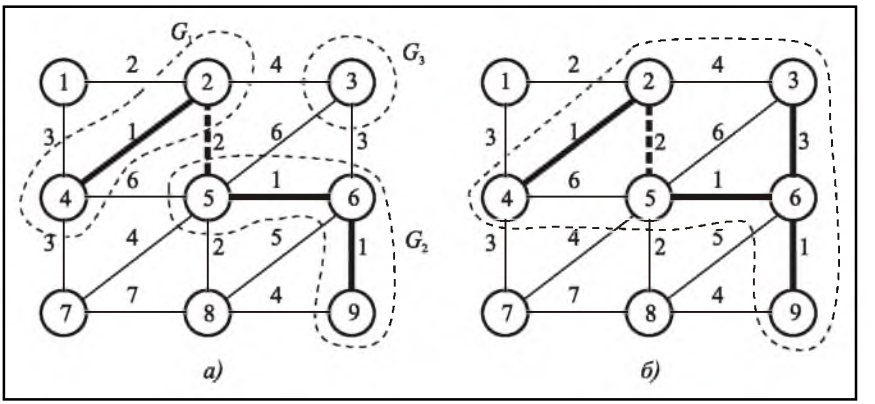


Рис. 30.6. Пояснение к теореме 30.4

Поясним формулировку теоремы на примере. Возьмем один из промежуточных этапов построения МОД, показанный на рис. 30.5, *в*. Изобразим его отдельно (рис. 30.6, *а*). Здесь у нас построено несколько минимальных деревьев. Рассмотрим случай $k=3$. Пусть, например, $V_1=\{v_2, v_4\}$, $V_2=\{v_5, v_6, v_9\}$, $V_3=\{v_3\}$, и на каждом подмножестве вершин построено минимальное дерево. Они обведены штриховыми контурами. У нас $E_1=\{e_{24}\}$, $E_2=\{e_{56}, e_{69}\}$, $E_3=\varnothing$. Найдем ребро минимального веса, выходящее из G_1 в сторону G_2 или G_3. В сторону G_2 из G_1 выходят ребра e_{25} и e_{45}, а в сторону G_3 — e_{23}. Из этих трех ребер минимальный вес 2 — у e_{25}. В формулировке теоремы — это e_j, оно обозначено на рисунке жирной штриховой линией. Теорема утверждает следующее. Если мы хотим построить минимальное дерево на множестве вершин $\{v_2, v_4, v_5, v_6, v_9, v_3\}$, то среди всех минимальных деревьев, в которых есть ребра e_{24}, e_{56}, e_{69}, наверняка найдется хотя бы одно минимальное дерево, содержащее e_{23} (рис. 30.6, *б*).

Доказательство. От противного: предположим, что существует дерево $G_0=(V_0, E_0)$, построенное на объединении вершин: $V_0=\bigcup V_i$, включающее в себя объединение ребер: $E_0\supseteq\bigcup E_i$, не включающее в себя ребро e_j и имеющее вес меньше, чем вес любого минимального дерева, включающего e_j. В иллюстрации на рис. 30.6 такое случилось бы, если бы вместо e_{25} мы попытались бы соединить G_1 с G_2 ребром e_{45}. Докажем, что такое невозможно. Для этого добавим в G_0 ребро e_j. После этого G_0 перестанет быть деревом: в нем образуется один цикл. Этот цикл будет заходить в G_1, но не будет лежать в нем полностью: он будет заходить и в какое-либо другое множество вершин из V_2, V_3, ..., V_k. Действительно: один из концов e_j инцидентен вершине из G_1, а другой — какой-либо другой вершине из G_2, G_3, ..., G_k. Так, в графе на рис. 30.6 образовался бы цикл $v_2e_{25}v_5e_{45}v_4e_{24}v_2$. Этот цикл заходит в G_1, но захватывает еще и G_2. Удалим из этого цикла какое-либо другое ребро, одним концом входящее в G_1, а другим — в другое G_i. Такое ребро наверняка существует, иначе не было бы цикла, охватывающего G_1 и какое-либо другое G_i. И вес удаляемого ребра не меньше (а, может быть, и больше), чем присоединенного e_j. В примере с графом на рис. 30.6 мы удаляем e_{45}. Связность всех вершин в V_0 при этом сохранится: мы можем перейти от G_1 к другому G_i по e_j вместо удаленного ребра. Значит, после такой замены ребер у нас останется дерево, и вес его — не меньше, чем у исходного дерева. А это противоречит предположению о том, что G_0 имеет вес меньше, чем у дерева с e_j. □

На основании этой теоремы мы можем утверждать, что обе рассмотренные выше схемы дают МОД (хотя, может быть, и разные). В схеме 1 вначале каждое G_i состоит из одной вершины без ребер. На первом этапе мы выбираем ребро минимального веса. Оно соединяет два G_i, одно из которых можно обозначить G_1. Значит, оно наверняка войдет в одно из МОД. Далее обозначаем через G_1 минимальное дерево, образованное двумя вершинами и соединяю-

щим их ребром, и находим ребро минимального веса, инцидентное G_1 одним своим концом. В теореме 30.4 это e_j. Согласно этой теореме мы можем присоединить его к строящемуся дереву, при этом не теряется возможность построить одно из МОД. Ну а схема 2 — это просто иллюстрация к доказательству теоремы!

30.3. Описание процедуры *MinSpanTree*

При реализации на MATLAB схема 1 программируется проще, чем схема 2, т. к. проверять каждое ребро на инцидентность и отсутствие циклов легче с одним строящимся деревом, а не с несколькими. Входным параметром в процедуре поиска МОД является список ребер размером $m \times 2$ или $m \times 3$, который обозначен идентификатором E. Как и в задаче о максимальном паросочетании *(см. раздел 29.1)*, будем предполагать, что если пользователь задал 2 столбца, то нужно решать невзвешенную задачу, а если 3 — то взвешенную. Выходным параметром будет список номеров ребер, входящих в МОД. Заголовок функции и справочная информация выглядят так:

```
function nMST=MinSpanTree(E)
% Функция nMST=MinSpanTree(E) решает задачу о минимальном
% остовном дереве для связного графа.
% Входной параметр:
%   E(m,2) или (m,3) - ребра графа и их веса;
%      1-й и 2-й элементы каждой строки - это номера вершин;
%      3-й элемент каждой строки - это вес ребра;
%      m - количество ребер.
%      Если задан массив E(m,2), то все веса равны 1.
% Выходной параметр:
%   nMST - список номеров ребер, включенных в минимальное
%      (взвешенное) остовное дерево в порядке включения.
% Используется жадный алгоритм.
```

Проверка исходных данных организована так же, как в *разделе 29.1*. Мы проверяем наличие данных, размерность и количество столбцов входного массива, положительность и целочисленность первых двух столбцов.

```
if nargin<1,
  error('Нет исходных данных!')
end
sz=size(E); % размеры массива E
if length(sz)~=2,
  error('Массив E должен быть двумерным!')
end
```

```
if (sz(2)<2),
  error('В массиве E должно быть 2 или 3 столбца!'),
end
if ~all(all(E(:,1:2)>0)),
  error('1-й и 2-й столбцы массива E должны быть положительными!')
end
if ~all(all((E(:,1:2)==round(E(:,1:2))))),
  error('1-й и 2-й столбцы массива E должны быть целыми!')
end
```

Находим размер и мощность графа. Если пользователь не задал веса ребер, полагаем все их равными 1.

```
m=size(E,1); % количество ребер
n=max(max(E(:,1:2))); % количество вершин
if sz(2)==2, % веса не заданы
  E(:,3)=1; % задаем все веса =1
end
```

Нам нужно будет возвращать список номеров ребер, входящих в МОД. Поэтому добавим к списку ребер `E` их номера.

```
En=[(1:m)',E]; % добавляем номера ребер
```

Начинаем формировать МОД по схеме 1. Ищем ребро минимального веса и запоминаем его номер в выходном параметре `nMST`. В массиве `nVer` сохраняем номера вершин, которые входят в уже построенный фрагмент МОД. В теореме 30.4 это V_1. А само ребро минимального веса исключаем из списка `En`.

```
[Emin,nMST]=min(En(:,4)); % ребро с минимальным весом и его номер
nVer=[En(nMST,2:3)]; % начинаем формировать список вершин
En=En(setdiff([1:m],nMST),:); % удалили ребро из списка ребер
```

Сейчас у нас в списке номеров `nMST` одно ребро минимального веса, а в списке вершин `nVer` — две инцидентные ему вершины. Наращиваем дерево с помощью цикла `while`. Вначале находим все ребра, один конец которых инцидентен уже построенному дереву, а другой — нет. Сохраняем номера этих ребер в массиве `Encurr`.

```
while length(nVer)<n, % пока не все вершины включены в дерево
  Encurr=[]; % пустой массив для списка допустимых ребер
  for k=1:size(En,1), % просматриваем каждое оставшееся ребро
    if (ismember(En(k,2),nVer) & ~ismember(En(k,3),nVer)) | ...
       (ismember(En(k,3),nVer) & ~ismember(En(k,2),nVer)),
      Encurr=[Encurr;En(k,:)]; % добавляем допустимое ребро
    end
  end
```

Проверяем: если допустимых ребер нет, то наш граф — несвязный. В этом случае выдаем сообщение об ошибке и заканчиваем работу процедуры.

```
if isempty(Encurr), % нет допустимых ребер
  error('Граф несвязный!')
end
```

Если же граф связный (допустимые ребра есть), мы из всех допустимых ребер выбираем ребро минимального веса (в теореме 30.4 это e_j), номер его присоединяем в список `nMST`, вершины добавляем в список `nVer`, а само ребро удаляем из списка `En`.

```
  [Emin,imin]=min(Encurr(:,4)); % ищем минимальное ребро из допустимых
  nEdge=Encurr(imin,1); % номер этого ребра
  nMST=[nMST,nEdge]; % добавили номер ребра в список ребер
  nVer=unique([nVer,Encurr(imin,2:3)]); % добавили вершину в список
  En=En(setdiff([1:size(En,1)],find(En(:,1)==nEdge)),:); % удалили ребро
end
return
```

По окончании цикла `while` в массиве `nMST` будут номера ребер, включенных в МОД, в порядке включения. Заканчиваем работу процедуры.

30.4. Пример обращения к процедуре *MinSpanTree*

Проиллюстрируем работу функции `MinSpanTree` на том же графе, что и в *разделе 29.1.4*. Введем исходные данные: координаты вершин, список ребер и их веса. Нарисуем исходный граф со взвешенными ребрами (рис. 30.7).

```
clear all
V=[[0 4];[1 4];[2 4];[3 4];[4 4];...
    [0 3];[1 3];[2 3];[3 3];[4 3];...
    [0 2];[1 2];[2 2];[3 2];[4 2];...
    [0 1];[1 1];[2 1];[3 1];[4 1];...
    [0 0];[1 0];[2 0];[3 0];[4 0]]; % координаты вершин
E=[[ 1  2 12];[ 3  2 17];[ 4  3 10];[ 5  4  8];[ 6  1  4];...
   [ 2  7 11];[ 8  2 10];[ 3  8  9];[ 9  4  4];[ 9  5  2];...
   [10  5 14];[ 7  6 16];[ 8  7 15];[ 8  9 13];[10  9 11];...
   [11  6 10];[ 7 12 11];[13  8 13];[14  9  1];[15 10  2];...
   [12 11  3];[13 12  3];[13 14  4];[14 13 12];[15 14  5];...
   [16 11  6];[12 17  7];[13 18  8];[20 15 10];[17 16  9];...
   [17 18 10];[18 17 11];[19 18 12];[19 20 13];[21 16 14];...
   [17 22 12];[18 22 19];[22 18 18];[18 23 17];[19 24 16];...
   [20 25 15];[21 22 14];[22 21 13];[23 24 12];[24 23 11];[24 25 10]];
```

```
PlotGraph(V(:,1:2),E); % рисуем граф
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfИсходный граф со взвешенными ребрами ')
```

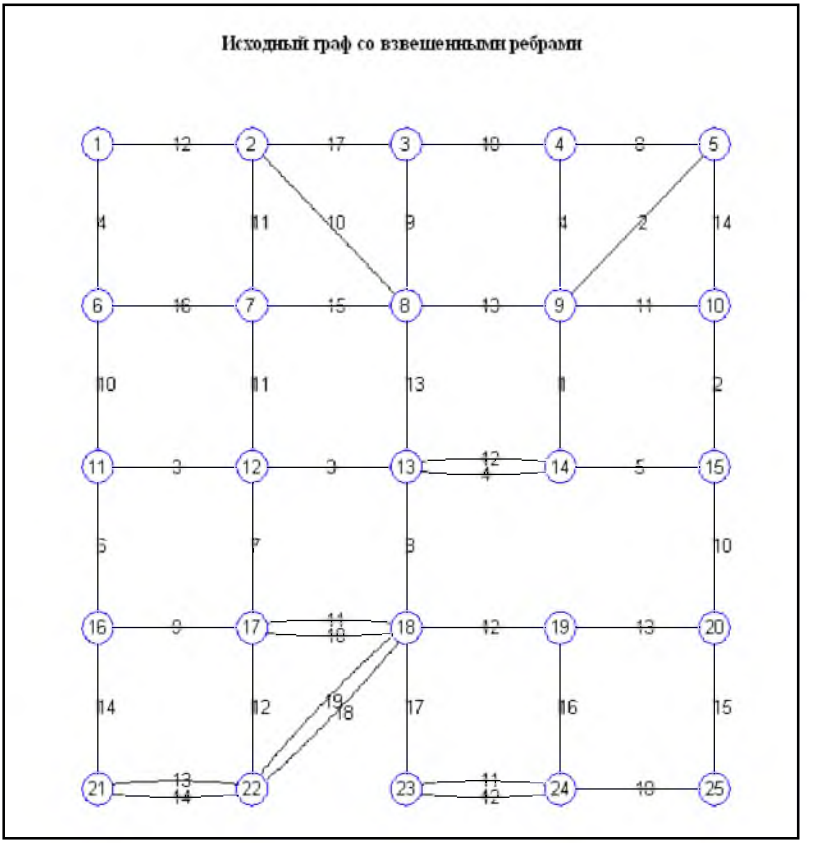


Рис. 30.7. Исходный граф со взвешенными ребрами, выполненный с помощью MATLAB

Решаем две задачи: невзвешенную и взвешенную. В первом случае функция `MinSpanTree` строит первое встретившееся остовное дерево, а во втором — МОД. Печатаем количество ребер и их суммарный вес в обоих вариантах. Рисуем оба решения (рис. 30.8-9): показываем все вершины, но только те ребра, которые входят в найденное остовное дерево.

```
nMST=MinSpanTree(E(:,1:2)); % невзвешенная задача
fprintf('Количество ребер в остовном дереве = %d\n',length(nMST));
fprintf('Общий вес = %d\n',sum(E(nMST,3)));
PlotGraph(V(:,1:2),E(nMST,:),'g');
```

```
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfОстовное дерево')
nMST=MinSpanTree(E); % взвешенная задача
fprintf('Количество ребер в МОД = %d\n',length(nMST));
fprintf('Общий вес = %d\n',sum(E(nMST,3)));
PlotGraph(V(:,1:2),E(nMST,:),'g');
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfМинимальное остовное дерево')
Количество ребер в остовном дереве = 24
Общий вес = 244
Количество ребер в МОД = 24
Общий вес = 182
```

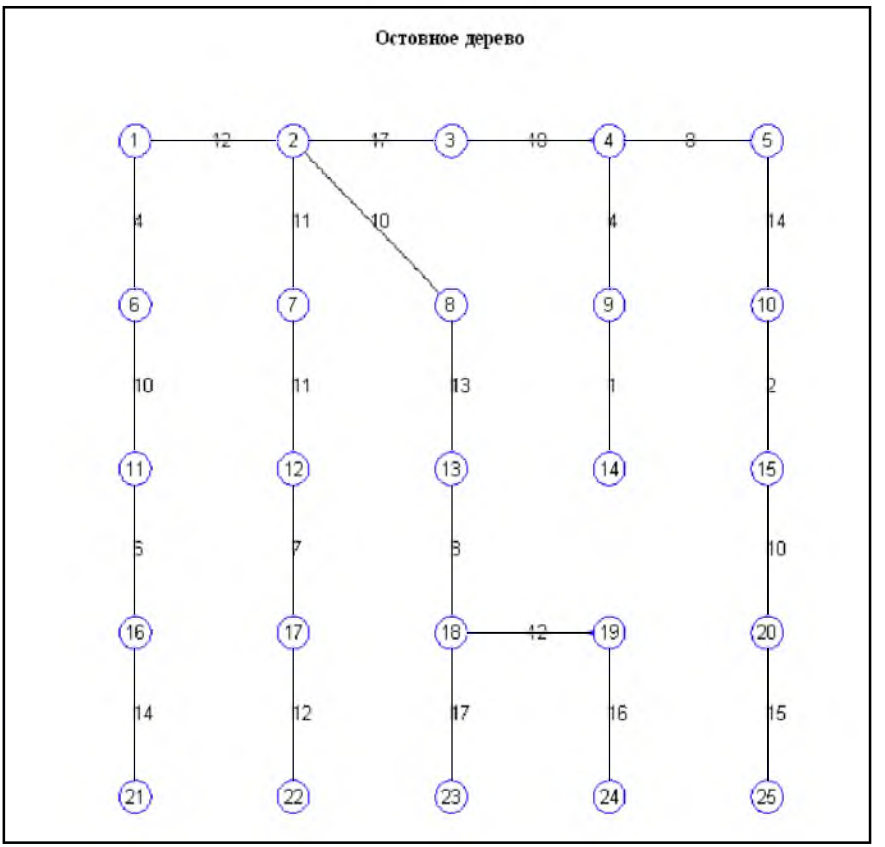


Рис. 30.8. Остовное дерево, построенное с помощью MATLAB

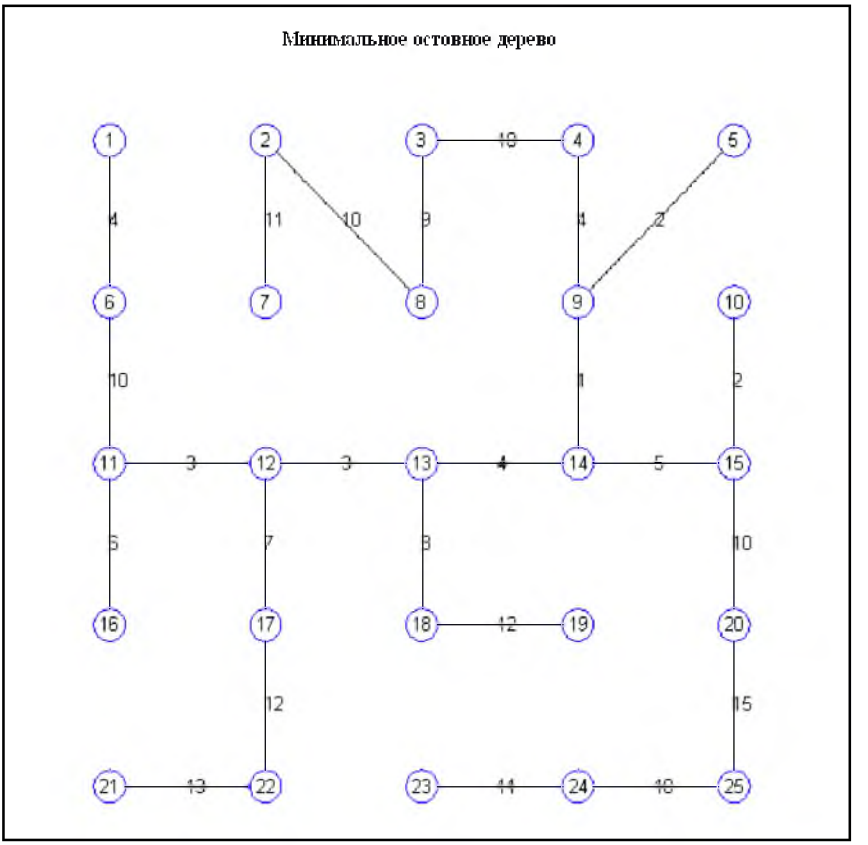
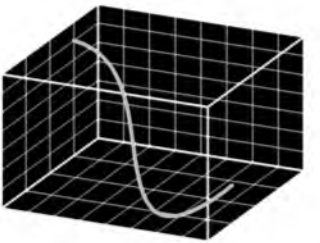


Рис. 30.9. Остовное дерево минимального веса, построенное с помощью MATLAB

30.5. Вопросы для самопроверки

1. Переберите все 4! перестановки в примере 30.1 и убедитесь, что действительно $z_{max} = 7$ достигается только на векторе x = {5; 3; 2; 4}.
2. Найдите в литературе или Интернете хотя бы 10 определений матроидов.
3. Сформулируйте и докажите теоремы, обратные к теоремам 30.2 и 30.3.
4. Докажите теорему: если к остовному дереву добавить одно ребро, то образуется ровно один простой цикл.
5. Сколько всего существует остовных деревьев у графа $G = (V, E)$? Является ли полиномиальным алгоритм построения всех остовных деревьев?
6. Переделайте функцию `MinSpanTree` так, чтобы она работала и с несвязными графами: строила для них МОД каждой связной компоненты.

ГЛАВА 31



Базис в пространстве циклов

Вспомните некоторые определения теории линейных пространств. В частности, понятия линейно независимых элементов и базиса. В этой главе мы будем строить базис множества из конечного числа элементов: циклов графа.

31.1. Сколько нужно циклов?

Возможно, вы изучали электротехнику, и вам приходилось рассчитывать цепи по законам Кирхгофа. Электрическая цепь представляется в виде графа (или мультиграфа) с n вершинами и m ребрами. Уравнения 1-го закона Кирхгофа — это баланс токов в узлах. Суммарный баланс токов во всех узлах равен нулю, поэтому из n уравнений 1-го закона независимыми будут только $n-1$. Чтобы найти токи во всех m ветвях цепи (ребрах графа), нужны еще $m-n+1$ независимых уравнений. Их дает 2-й закон Кирхгофа: суммарное падение напряжений в каждом замкнутом контуре равно сумме ЭДС в этом контуре. Поэтому нужно найти такие $m-n+1$ контуров, чтобы уравнения для них были независимыми. Анализируя структуру уравнений 2-го закона Кирхгофа, видим, что это выполняется, если каждый из контуров будет отличаться от любого другого хотя бы одной ветвью. Поэтому, решая соответствующую задачу на графах, мы должны построить $m-n+1$ простых циклов *(см. определения 30.5—30.6)*, каждый из которых отличается от любого другого хотя бы одним ребром.

Ранее в *определениях 30.1 и 30.5* мы задавали маршрут и его частный случай — цикл — как последовательность чередующихся вершин и ребер. Но эта информация избыточна: чтобы однозначно определить любой маршрут (в том числе и цикл), достаточно задать только последовательность ребер. Более того, если нас интересуют простые маршруты и циклы, то не важен

даже порядок ребер. Поэтому будем задавать простой цикл как некоторое подмножество множества ребер.

ПРИМЕР 31.1. Простые циклы графа на рис. 28.1, *а* — это подмножества множества ребер: $\{e_1, e_3, e_4\}$, $\{e_2, e_3, e_5\}$, $\{e_1, e_2, e_4, e_5\}$. □

Но не любое подмножество множества E является простым циклом. Так, в примере 31.1 $\{e_1, e_2\}$ или $\{e_1, e_2, e_4\}$ с того же рисунка циклами не являются. Поэтому мы будем рассматривать только такие подмножества множества ребер E, которые действительно являются простыми циклами. Будем обозначать их C_1, C_2, ... В нашем примере у графа на рис. 28.1, *а* всего 3 простых цикла: $C_1 = \{e_1, e_3, e_4\}$; $C_2 = \{e_2, e_3, e_5\}$; $C_3 = \{e_1, e_2, e_4, e_5\}$.

Если мы удалим из простого цикла хотя бы одно ребро, то он разомкнется и перестанет быть циклом. Это утверждение настолько очевидно, что обычно оно формулируется в виде аксиомы.

■ **АКСИОМА 31.1.** Если C_i является простым циклом, то любое его собственное подмножество циклом не является. □

Введем теперь над простыми циклами операцию сложения по логике XOR (исключающего или): из двух подмножеств ребер C_i и C_j формируем новое подмножество, включая в него только неповторяющиеся элементы из C_i и C_j.

Определение 31.1. *Суммой* двух простых циклов C_i и C_j называется подмножество ребер, которые входят или только в C_i, или только в C_j. □

Будет ли сумма простых циклов простым циклом? Иногда да, а иногда и нет.

ПРИМЕР 31.2. Проверим, что получится при сложении простых циклов графа с рис. 28.1, *а*. У нас есть $C_1 = \{e_1, e_3, e_4\}$; $C_2 = \{e_2, e_3, e_5\}$; $C_3 = \{e_1, e_2, e_4, e_5\}$.

$$C_1 + C_2 = \{e_1, e_2, e_4, e_5\} = C_3;$$
$$C_1 + C_3 = \{e_2, e_3, e_5\} = C_2;$$
$$C_2 + C_3 = \{e_1, e_3, e_4\} = C_1.$$

Здесь сумма двух любых простых циклов дает третий, также простой цикл. □

ПРИМЕР 31.3. На рис. 31.1, *а* показаны два простых цикла: $C_1 = \{e_{23}, e_{25}, e_{36}, e_{58}, e_{69}, e_{89}\}$ и $C_2 = \{e_{23}, e_{24}, e_{36}, e_{45}, e_{56}\}$. В результате их сложения получаем подмножество ребер $\{e_{24}, e_{25}, e_{45}, e_{56}, e_{58}, e_{69}, e_{89}\}$, которое не является простым циклом. Это видно и на рис. 31.1, *б*, и из проверки аксиомы 31.1. У этого множества есть собственные подмножества $\{e_{24}, e_{25}, e_{45}\}$ и $\{e_{56}, e_{58}, e_{69}, e_{89}\}$, которые являются простыми циклами. Значит, аксиома 31.1 нарушается, и поэтому полученное множество $C_1 + C_2$ не является простым циклом. □

Но можно утверждать, что $C_1 + C_2$ содержит в себе в качестве подмножества какой-либо простой цикл. Это — вторая аксиома.

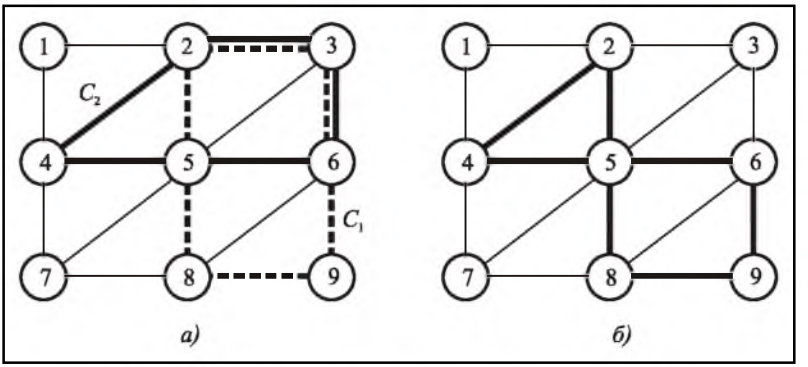


Рис. 31.1. Сложение простых циклов по логике XOR

■ **АКСИОМА 31.2.** Если C_i и C_j — простые циклы, то существует простой цикл $C_k \subseteq C_i + C_j$. □

Если вы отвечали на вопросы *главы 30*, то можете заметить, что множество всех простых циклов графа образует матроид на множестве его ребер согласно одному из определений матроидов (через циклы). Вновь повторим, что мы не будем отвлекаться на изучение этой интереснейшей области комбинаторного анализа — теории матроидов. Наша задача проще: нам нужно найти такое подмножество простых циклов, из которых можно было бы получить любой другой простой цикл согласно аксиоме 31.2. Желательно также, чтобы этих циклов (из которых мы будем строить все другие) было как можно меньше. Множество всех простых циклов графа можно рассматривать как некоторое комбинаторное пространство. Тогда тот минимальный набор циклов, из которых можно построить все остальные, естественно назвать базисом этого пространства.

Определение 31.2. *Базисом в пространстве циклов (цикловым базисом)* называется минимальное по количеству подмножество простых циклов, из которых можно построить любой другой простой цикл по аксиоме 31.2. □

В обычных линейных пространствах базисные элементы являются линейно независимыми. Для проверки этого факта нужно построить их линейную комбинацию и попытаться найти ненулевые коэффициенты. В комбинаторных пространствах ситуация другая: мы не вводили операцию умножения на скаляр. У нас есть только операция сложения с последующим выделением подмножества. Поэтому и определение линейной независимости здесь будет несколько иным.

Определение 31.3. Простые циклы называются *независимыми*, если они отличаются друг от друга хотя бы одним ребром. □

Далее мы сформулируем несколько теорем, но докажем только часть из них, включая основную. Остальные теоремы легко проверяются на конкретных случаях, а их доказательство мы оставляем читателю для самостоятельной работы.

■ **ТЕОРЕМА 31.1.** В системе независимых простых циклов ни один из них нельзя построить из других путем сложения и выборки подмножества (т. е. по аксиоме 31.2).

Доказательство. Возьмем любой цикл C_0 из системы независимых простых циклов. В нем обязательно есть ребро, которого нет ни в одном другом цикле системы. Поэтому, как бы мы ни объединяли множества ребер из остальных циклов и что бы мы ни выбирали из этих объединений, все равно недостающее ребро никогда не появится. Поэтому построить C_0 никогда не удастся. □

■ **ТЕОРЕМА 31.2 (обратная).** Если некоторая система простых циклов не является независимой, то, по крайней мере, один из них можно построить из остальных по аксиоме 31.2. □

Это означает следующее. Если у нас есть какой-либо простой цикл C_0, все ребра которого входят в другие простые циклы, то C_0 всегда можно построить из этих других по аксиоме 31.2.

ПРИМЕР 31.4. Вернемся к рис. 31.1, *а*. Здесь показаны 2 простых цикла: $C_1 = \{e_{23}, e_{25}, e_{36}, e_{58}, e_{69}, e_{89}\}$ и $C_2 = \{e_{23}, e_{24}, e_{36}, e_{45}, e_{56}\}$. Рассмотрим простой цикл $C_0 = \{e_{23}, e_{25}, e_{36}, e_{56}\}$. Он зависим с C_1 и C_2, т. к. все его ребра есть или в C_1, или в C_2. Можно ли построить его из C_1 и C_2 по аксиоме 31.2? Теорема 31.2 утверждает, что да. Это можно сделать, например, так. Вначале строим $C_3 = \{e_{24}, e_{25}, e_{45}\} \subseteq C_1 + C_2$, а затем находим $C_0 = \{e_{23}, e_{25}, e_{36}, e_{56}\} = C_2 + C_3$. На втором шаге нам даже не понадобилось выбирать подмножество: сумма C_2 и C_3 сразу дает нужный нам простой цикл C_0. □

■ **ТЕОРЕМА 31.3 (основная).** В связном графе $G = (V, E)$ с $|V| = n$, $|E| = m$ существует не менее чем $m - n + 1$ независимых простых циклов.

Доказательство. Построим любое остовное дерево графа G. В него войдут $n-1$ ребер, и не войдут остальные $m - n + 1$ ребер. При добавлении каждого из них к остовному дереву образуется один простой цикл — всего имеем $m - n + 1$ простых циклов. В каждом из них есть, по крайней мере, одно ребро, которого нет в остальных: это то ребро, которое мы добавили к остовному дереву для образования простого цикла. Поэтому полученные $m - n + 1$ простых циклов являются независимыми. □

Эта теорема дает нам очень удобный алгоритм построения $m - n + 1$ независимых простых циклов: нужно взять остовное дерево и добавить к нему одно ребро. При этом образуется один простой цикл с "хвостами" (как голова Ме-

дузы Горгоны с торчащими из нее во все стороны змеями-щупальцами). Отрубив голову (отрезав хвосты-щупальцы), получим простой цикл. Затем повторяем эту процедуру с другим ребром — получаем другой простой цикл, и т. д. — всего будем иметь $m-n+1$ независимых простых циклов. Но достаточно ли их? Можно ли сказать, что любой другой простой цикл в графе можно будет построить из них по аксиоме 31.2? Ответ на этот вопрос дает следующая теорема.

■ **ТЕОРЕМА 31.4.** Любой простой цикл связного графа $G=(V, E)$ с $|V|=n$, $|E|=m$ является зависимым с теми $m-n+1$ независимыми простыми циклами, которые построены в теореме 31.3. □

В соответствии с этой теоремой мы можем утверждать, что $m-n+1$ независимых простых циклов действительно являются базисом в пространстве циклов: любой другой простой цикл строится из них путем сложения с последующей выборкой в соответствии с аксиомой 31.2.

31.2. Описание процедуры *CicleBasis*

Рассмотрим реализацию на MATLAB алгоритма построения циклового базиса. Входным параметром должен быть список ребер графа (веса здесь не нужны). Этой информации достаточно для работы алгоритма. На выходе мы будем получать список всех $m-n+1$ независимых простых циклов. Каждый цикл — это список входящих в него ребер, причем порядок их нумерации не важен. Циклы могут быть разной длины. Поэтому возможны такие варианты выходного параметра.

1. Массив из $m-n+1$ ячеек. В каждой ячейке — одномерный массив номеров ребер, входящих в данный цикл.
2. Двумерный массив из $m-n+1$ строк. В каждой строке — номера ребер, входящих в данный цикл. Более короткие строки дополняются нулями.
3. Булевский массив размером $(m-n+1)\times m$ или $m\times(m-n+1)$, в каждой строке (столбце) которого элемент равен true, если ребро с этим номером входит в цикл.

Ячейки в MATLAB обрабатываются медленнее действительных чисел, поэтому 1-й вариант годится только для графов небольших размеров. Во 2-м варианте максимальная длина цикла заранее неизвестна, а перестройка массива на другой (больший) размер занимает время, сравнимое со временем работы алгоритма. Поэтому в описываемой ниже процедуре результаты возвращаются в булевском массиве размером $m\times(m-n+1)$. Вначале, как всегда, записываем заголовок процедуры и приводим справочную информацию.

```
function Cycles=CycleBasis(E)
% Функция Cycles=CycleBasis(E) ищет все независимые циклы связного графа.
% Входной параметр:
%   E(m,2) - список ребер графа;
%     1-й и 2-й элементы каждой строки - это номера вершин;
%     m - количество ребер.
% Выходной параметр:
%   Cycles(m,m-n+1) - булевский массив номеров ребер.
%     n - количество вершин;
%     m-n+1 - количество независимых циклов.
%     В каждом столбце массива Cycles значение True имеют элементы
%     с номерами ребер этого цикла.
```

При построении циклового базиса нам нужно будет найти остовное дерево. Воспользуемся готовой процедурой `MinSpanTree`. Заодно в этой процедуре мы проверим исходные данные. Если в них обнаружится ошибка, сообщение об этом будет выдано в командное окно MATLAB, а процедура прекратит свою работу.

```
nMST=MinSpanTree(E); % проверка данных и построение МОД
```

Оставляем в списке ребер только 2 первых столбца (ведь пользователь мог задать больше). Находим размер и мощность графа. Добавляем в список ребер их номера.

```
E=E(:,1:2); % оставили первые два столбца
m=size(E,1); % количество ребер
n=max(max(E)); % количество вершин
En=[[1:m]',E]; % добавили номера ребер
```

Формируем список оставшихся (не вошедших в МОД) ребер. Задаем массив нужных размеров для циклов.

```
Erest=En(setdiff([1:m],nMST),:); % оставшиеся ребра
nr=m-n+1; % количество оставшихся ребер
Cycles=zeros(m,nr); % массив для циклов
```

Вся подготовительная работа проделана. Начинаем основной цикл. Добавляем одно очередное ребро к МОД.

```
for k1=1:nr, % строим независимые циклы
  Ecurr=[En(nMST,:);Erest(k1,:)]; % МОД + еще одно ребро
```

Теперь нужно удалить хвосты в полученном множестве ребер и оставить только цикл. Простейший алгоритм очистки цикла такой. Просматриваем все вершины и находим, какая из них встречается только один раз. Это — конец хвоста. Если такая вершина есть, то удаляем ребро с этой вершиной. Далее

процесс повторяется. Как только выясним, что одиноких вершин нет, заканчиваем: хвостов тоже больше нет. Всего в графе n вершин, поэтому процесс просмотра и удаления ребер с одинокими вершинами будет повторяться не более n раз. Удобнее всего записать цикл `for`, а досрочный выход при необходимости организовать по команде `break`. В начале цикла мы формируем 2 списка: всех вершин с учетом их кратности `fa` и такого же списка, но без учета кратности `ufa`.

```
for k2=1:n, % начало цикла по удалению хвостов
  fa=sort(reshape(Ecurr(:,2:3),1,2*size(Ecurr,1))); % все вершины
  ufa=unique(fa); % все вершины без учета кратности
```

Ищем, сколько раз встречается каждая вершина в списке. Если очередная вершина встречается только один раз, поиск прекращаем: найден конец хвоста.

```
  lv=[]; % количество повторений
  for k3=1:length(ufa), % ищем неповторяющуюся вершину
    lv(k3)=length(find(fa==ufa(k3))); % количество повторений
    if lv(k3)==1, % нашли вершину без повторений
      break; % дальше искать не нужно - найден конец хвоста
    end
  end
```

Проверяем, как мы закончили цикл по `k3`: нашли конец хвоста или нет. Если в массиве `iv` нет единиц, то это значит, что среди вершин нет таких, которые встречаются только один раз. Все хвосты удалены — выходим из цикла по `k2`.

```
  llv1=find(lv==1); % ищем, есть ли одинокие вершины
  if isempty(llv1), % нет одиноких вершин - все хвосты удалены
    break; % выходим из цикла по k2
  end
```

Если же одинокая вершина обнаружена, мы находим ребро, которому она принадлежит, и удаляем его.

```
  sv=ufa(llv1(1)); % номер одинокой вершины
  nume=find(sum(Ecurr(:,2:3)==sv,2)); % номер ребра с одинокой вершиной
  Ecurr=Ecurr(setdiff([1:size(Ecurr,1)],nume),:); % удаляем ребро
end % конец цикла по k2
```

Теперь все хвосты удалены — остался только цикл. Записываем номера его ребер в нужный столбец массива `Cycles`.

```
  Cycles(Ecurr(:,1),k1)=logical(1); % записываем номера ребер
end % конец цикла по k1
return
```

31.3. Пример обращения к процедуре *CicleBasis*

Рассмотрим пример использования этой функции. Чтобы не рисовать слишком много, возьмем небольшой граф с 11 вершинами и 22 ребрами, использованный нами ранее в *разделе 29.2*. Введем его без весов и нарисуем (рис. 31.2).

```
clear all
V=[[0 0];[1 1];[1 0];[1 -1];...
   [2 1];[2 0];[2 -1];[3 1];...
   [3 0];[3 -1];[4 0]]; % координаты вершин
E=[[1 2];[1 3];[1 4];[2 3];[3 4];[2 5];...
   [2 6];[3 6];[3 7];[4 7];[6 5];[6 7];...
   [5 8];[6 8];[6 9];[7 9];[7 10];[8 9];...
   [9 10];[8 11];[9 11];[10 11]]; % ребра
PlotGraph(V,E); % рисуем граф
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfИсходный граф')
```

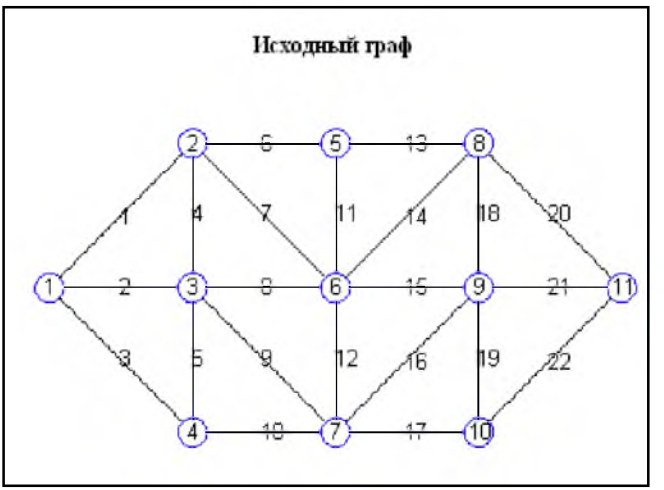


Рис. 31.2. Исходный граф, построенный с помощью MATLAB

У нашего графа $n=11$ вершин и $m=22$ ребер. Процедура `CycleBasis` найдет $m-n+1=12$ независимых циклов. Найдем и нарисуем их все (рис. 31.3—31.14). На каждом рисунке покажем все вершины и те ребра, которые входят в данный цикл. В каждом рисунке надпишем заголовок — номер цикла.

```
Cycles=CycleBasis(E); % ищем все независимые циклы
for k1=1:size(Cycles,2),
  PlotGraph(V(:,1:2),E(find(Cycles(:,k1)),1:2)); % рисуем цикл
  set(get(gcf,'CurrentAxes'),...
    'FontName','Times New Roman Cyr','FontSize',10)
  title(['\bfЦикл N' num2str(k1)]) % заголовок рисунка
end
```

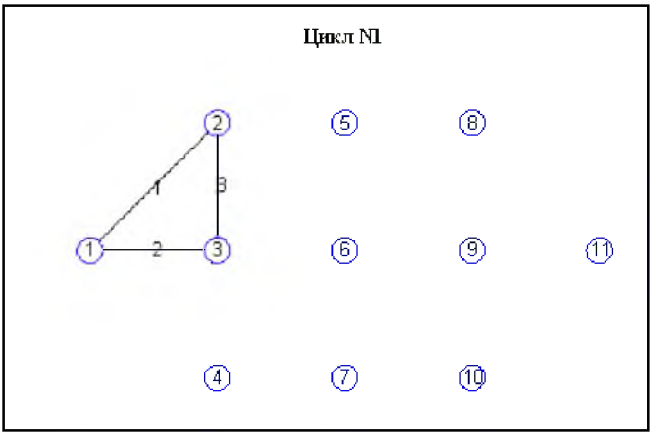


Рис. 31.3. Цикл № 1, найденный с помощью MATLAB

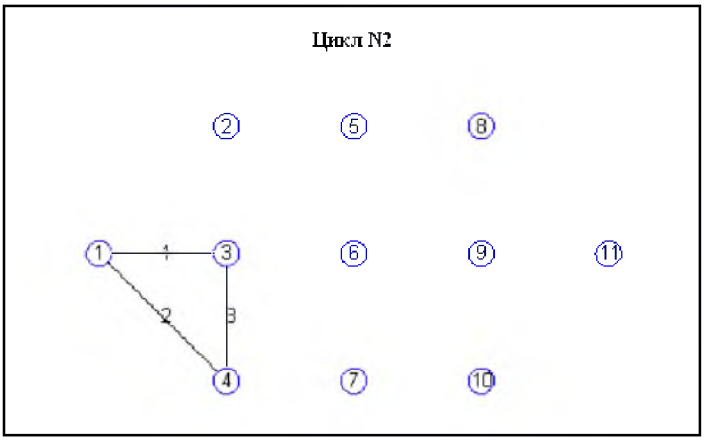


Рис. 31.4. Цикл № 2, найденный с помощью MATLAB

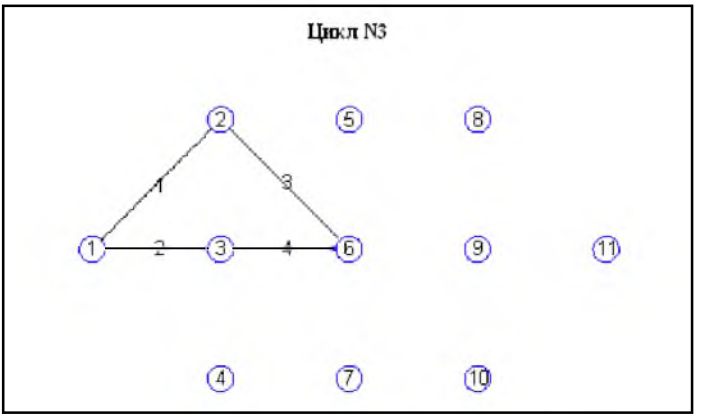


Рис. 31.5. Цикл № 3, найденный с помощью MATLAB

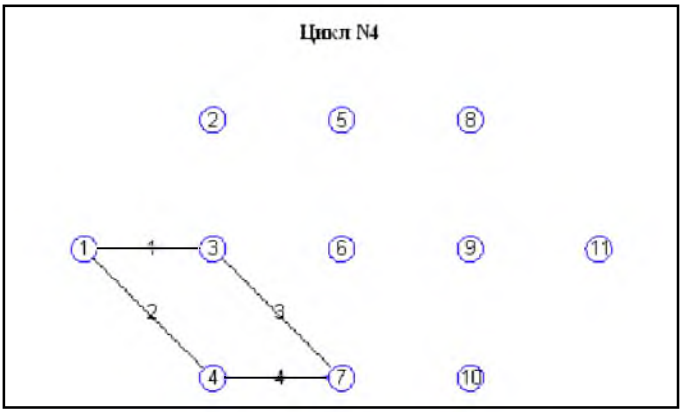


Рис. 31.6. Цикл № 4, найденный с помощью MATLAB

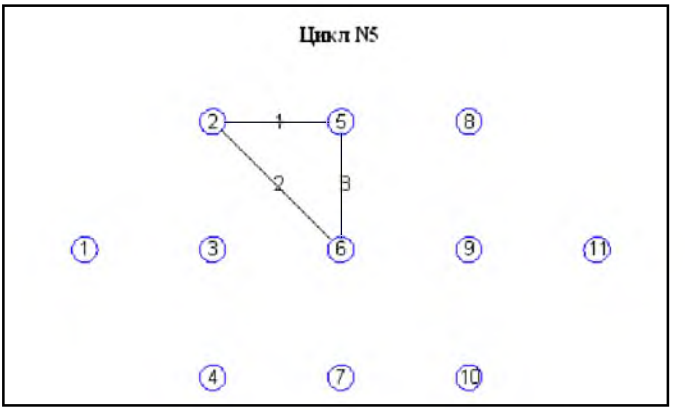


Рис. 31.7. Цикл № 5, найденный с помощью MATLAB

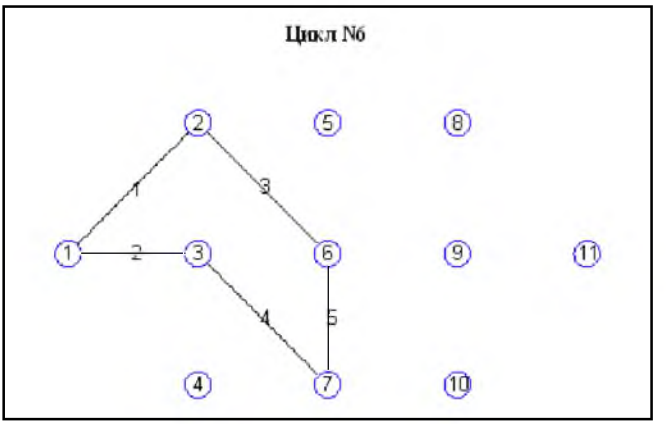


Рис. 31.8. Цикл № 6, найденный с помощью MATLAB

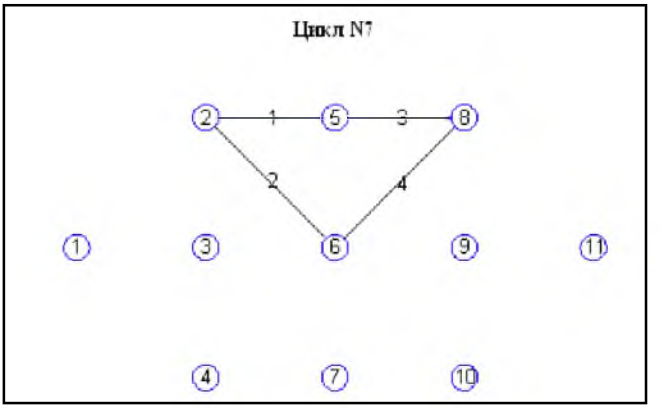


Рис. 31.9. Цикл № 7, найденный с помощью MATLAB

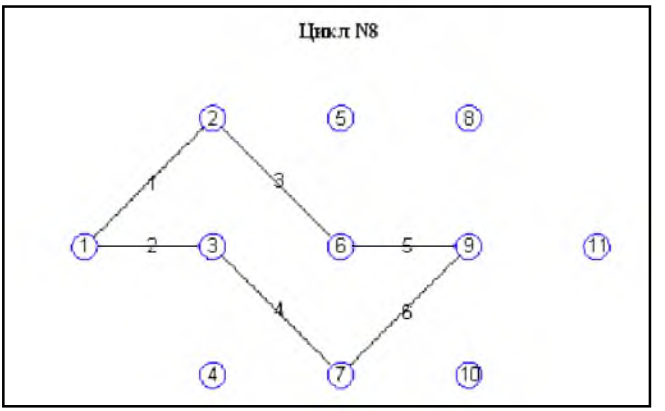


Рис. 31.10. Цикл № 8, найденный с помощью MATLAB

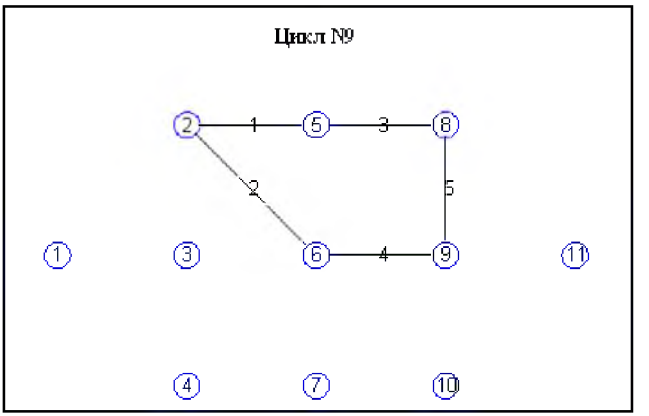


Рис. 31.11. Цикл № 9, найденный с помощью MATLAB

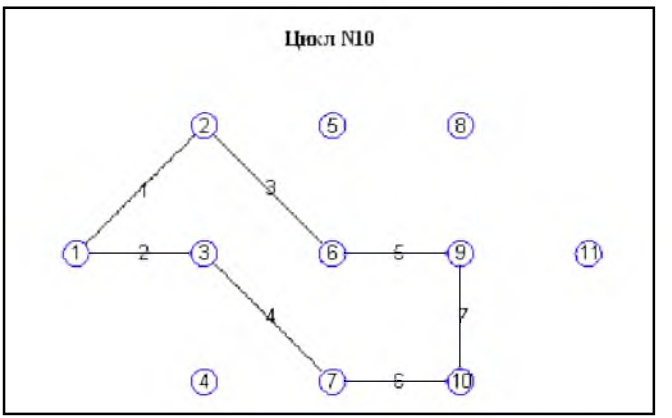


Рис. 31.12. Цикл № 10, найденный с помощью MATLAB

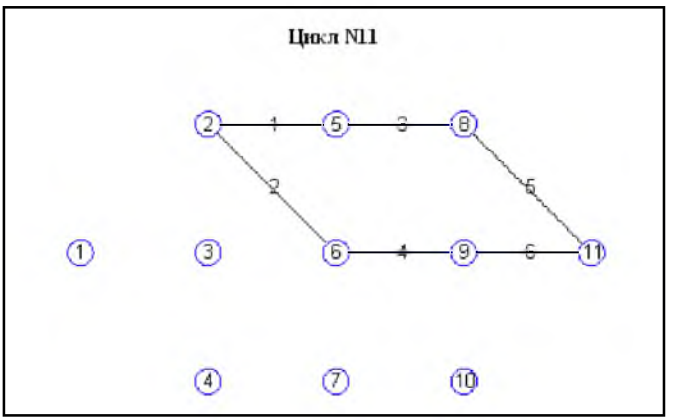


Рис. 31.13. Цикл № 11, найденный с помощью MATLAB

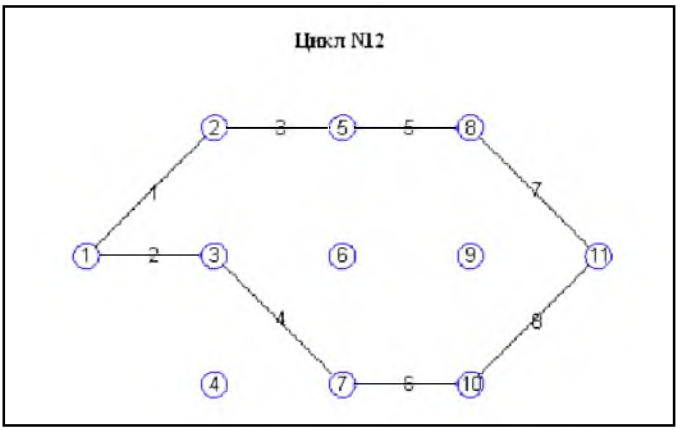
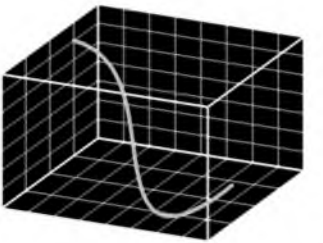


Рис. 31.14. Цикл № 12, найденный с помощью MATLAB

31.4. Вопросы для самопроверки

1. Сколько циклов существует в K_n? Сколько из них независимых?
2. Составьте процедуру сложения простых циклов по логике XOR и выделения из суммы всех входящих в нее простых циклов.
3. Докажите теоремы 31.2 и 31.4.
4. Переделайте процедуру `CycleBasis` так, чтобы она возвращала двумерный массив из $m-n+1$ строк, в каждой строке которого — номера ребер, входящих в данный цикл, а более короткие строки дополнены нулями. Сравните при помощи профилировщика время выполнения процедуры в одном и другом случаях. Какой вариант работает быстрее?
5. Проделайте то же самое для массива из $m-n+1$ ячеек, в каждой из которых — одномерный массив номеров ребер, входящих в данный цикл.

ГЛАВА 32



Правильная раскраска вершин

Задача о правильной раскраске вершин графа является одной из классических задач теории графов. Рассмотрим ее решение средствами MATLAB.

32.1. Сколько нужно красок?

Представьте себе, что вы занимаетесь полиграфией и издаете географические карты. Вам нужно напечатать красивую политическую карту, но для экономии средств хочется обойтись минимальным количеством красок. При этом должно быть выполнено основное требование к политическим картам: никакие две соседние страны (провинции, области) не должны быть окрашены в одинаковые цвета.

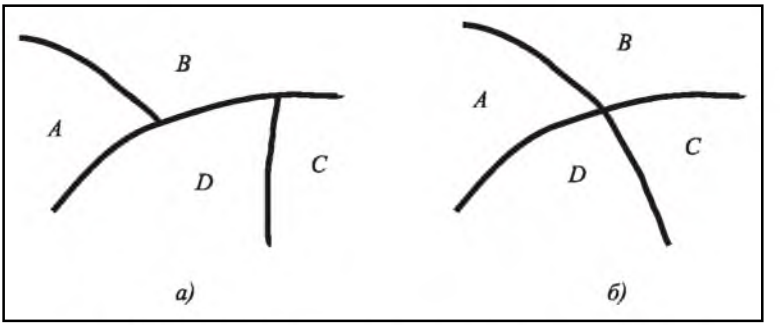


Рис. 32.1. Границы: *а* — невырожденные; *б* — вырожденные

Как правило, границы между соседними странами невырожденные: в любой точке границы соседствуют не более трех стран (рис. 32.1, *а*). Если в какой-либо точке граничат 4 или более страны (на рис. 32.1, *б*), то будем полагать,

что можно закрашивать одним цветом страны, которые граничат только в одной точке и не имеют общих линейных границ. Например, на рис. 32.1, *б* страны *A* и *C* можно закрасить одним цветом, если, конечно, они не соседствуют еще где-то за страной *D* или *B* вдоль линии. Будем также полагать, что каждая страна образует на карте связную область (без анклавов). Какое минимальное количество цветов потребуется для раскраски?

Эта задача может быть сформулирована как задача на графе. Для этого поместим в каждой стране (например, в ее столице) вершину графа, а столицы соседних стран соединим ребром, которое пересекает их общую линейную границу и не заходит в третьи страны. Через вырожденные границы-точки ребер проводить не будем. После такой операции получим граф, причем планарный.

Определение 32.1. Граф называется *планарным*, если путем перемещения вершин и (или) ребер его можно разместить на плоскости без пересечения ребер. □

ПРИМЕР 32.1. На рис. 32.2, *а* показан планарный граф K_4 (ребро e_{14} можно переместить в другое место), а на рис. 32.2, *б* — непланарный граф: полный двудольный граф с $|V| = |W| = 3$. □

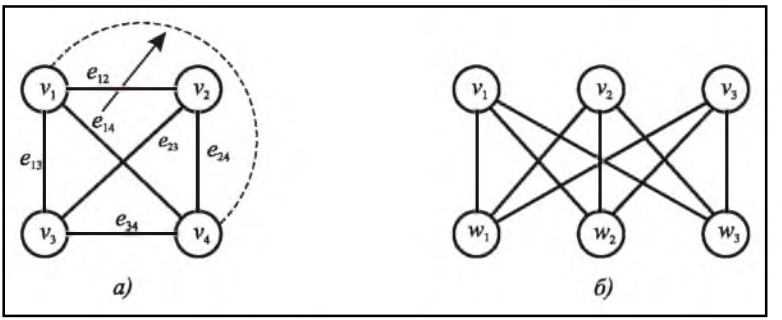


Рис. 32.2. Графы: *а* — планарный; *б* — непланарный

После такой замены получаем задачу о правильной раскраске планарного графа. Если рассматривать ее для произвольных графов (не обязательно планарных), то она формулируется так: нужно раскрасить все вершины графа минимально возможным количеством красок так, чтобы никакие смежные вершины не были окрашены в одинаковый цвет. Известно, что для планарных графов достаточно 4-х красок. Но это только количественная оценка числа красок, причем лишь для одного частного случая планарных графов, а сами краски (их номера) нужно еще приписать вершинам.

32.2. Правильная раскраска графа — задача ЦЛП

Рассмотрим один из возможных подходов к построению правильной раскраски вершин графа — сведение к задаче ЦЛП. Номер краски — это целое число, поэтому введем в рассмотрение целочисленные переменные v_i, ассоциированные с вершинами. Всего имеем n таких переменных, и каждая из них может принимать значения от 1 до n — это номер краски вершины v_i. В наихудшем случае каждая вершина будет выкрашена в свой цвет, поэтому максимальное значение каждой переменной v_i — это n:

$$\begin{cases} v_i \in [1, n]; \\ i \in [1, n]. \end{cases} \tag{32.1}$$

Нам нужно минимизировать максимальное v_i:

$$z = \max_{i \in [1, n]} v_i \to \min. \tag{32.2}$$

Удобнее всего ввести для этого в рассмотрение еще одну дополнительную переменную v_0 (тоже целочисленную) и связать ее с остальными v_i системой неравенств:

$$\begin{cases} v_i \le v_0; \\ i \in [1, n]. \end{cases} \tag{32.3}$$

Тогда целевая функция — это:

$$z = v_0 \to \min. \tag{32.4}$$

В задаче о правильной раскраске есть ограничение: соседние вершины должны иметь разные цвета (номера цветов). Это значит: для каждого ребра $e_{ik} \in E$ переменные v_i и v_k должны отличаться хотя бы на единицу:

$$\begin{cases} |v_i - v_k| \ge 1; \\ \forall e_{ik} \in E. \end{cases} \tag{32.5}$$

К сожалению, ограничения (32.5) — нелинейные: в них присутствует модуль. Переход от каждого неравенства (32.5) к двум неравенствам:

$$\begin{bmatrix} v_i - v_k \ge 1; \\ v_k - v_i \ge 1; \end{bmatrix} \tag{32.6}$$

также ничего не дает, т. к. получается не система, а объединение неравенств (должно выполняться или одно, или другое). Тем не менее сформулировать

(32.5) как систему линейных неравенств можно. Чтобы это сделать, заметим вначале, что переменные v_i отличаются друг от друга не более чем на $n-1$, т. к. максимальный номер краски — это n, а минимальный — 1. Поэтому на самом деле вместо (32.6) мы имеем:

$$\begin{cases} 1 \le v_i - v_k \le n-1; \\ 1 \le v_k - v_i \le n-1; \end{cases} \tag{32.7}$$

причем правые неравенства автоматически выполняются в силу (32.1). Теперь введем в рассмотрение целочисленные переменные e_{ik}, ассоциированные с ребрами, и разрешим им принимать только одно из двух возможных значений — 0 или 1:

$$\begin{cases} e_{ik} = 0 \vee 1; \\ \forall e_{ik} \in E. \end{cases} \tag{32.8}$$

Рассмотрим систему неравенств (уже не объединение, а именно систему):

$$\begin{cases} v_i - v_k - ne_{ik} \le -1; \\ v_k - v_i + ne_{ik} \le n-1. \end{cases} \tag{32.9}$$

Если переменная e_{ik} принимает значение 0, то система (32.9) дает вторую пару неравенств из (32.7), а если 1 — то первую.

Таким образом, имеем следующую задачу ЦЛП. Необходимо минимизировать функцию z (32.4), которая формально зависит от $n+m+1$ переменных v_0, v_1, ..., v_n, $\forall e_{ik}$, но фактически в нее входит только v_0. На каждую v_i наложено ограничение (32.3) — всего n ограничений. Для каждой переменной e_{ik} и соответствующих ей v_i и v_k должны выполняться по 2 ограничения (32.9) — всего $2m$ ограничений. Все переменные v_i — целочисленные и могут принимать значения от 1 до n (32.1). Все переменные e_{ik} — целочисленные и могут принимать значения 0 или 1 (32.8).

Как показали численные исследования, ограничения вида (32.9) сильно усложняют процесс решения задачи ЦЛП. Процедура `milp` из [52] уже не справляется с этой задачей. Поэтому пришлось воспользоваться более мощной процедурой `miqp` из [59]. Она работает медленнее `milp`, но успешно раскрашивает небольшие графы.

32.3. Описание процедуры *ColorGraph*

Опишем процедуру `ColorGraph`, которая решает задачу правильной раскраски графа. Входным параметром будет список ребер размером $m \times 2$ (идентификатор `E`). Веса ребер здесь нас не интересуют, поэтому наличие или отсутствие

3-го столбца в массиве E не имеет значения. На выходе мы получим целочисленный вектор длины n — номера красок вершин. Пишем заголовок функции и справочную информацию:

```
function nCol=ColorGraph(E)
% Функция nCol=ColorGraph(E) решает задачу о правильной раскраске графа.
% Входной параметр:
%   E(m,2) - список ребер графа;
%      1-й и 2-й элементы каждой строки - это номера вершин;
%      m - количество ребер.
% Выходной параметр:
%   nCol(n,1) - список цветов вершин.
% Используется приведение к задаче ЦЛП.
% Необходимые другие функции: MIQP.M.
% Эта функция может быть свободно скопирована с сайта:
% http://control.ee.ethz.ch/~hybrid/miqp/
```

Проверка исходных данных — как и в других функциях пакета. Проверяем наличие данных, размерность и количество столбцов входного массива, положительность и целочисленность первых двух столбцов.

```
if nargin<1,
  error('Нет исходных данных!')
end
sz=size(E); % размеры массива E
if length(sz)~=2,
  error('Массив E должен быть двумерным!')
end
if (sz(2)<2),
  error('В массиве E должно быть 2 столбца!'),
end
if ~all(all(E(:,1:2)>0)),
  error('1-й и 2-й столбцы массива E должны быть положительными!')
end
if ~all(all((E(:,1:2)==round(E(:,1:2))))),
  error('1-й и 2-й столбцы массива E должны быть целыми!')
end
```

Находим размер и мощность графа.

```
m=size(E,1); % количество ребер
n=max(max(E(:,1:2))); % количество вершин
```

Начинаем формировать данные для решения задачи ЦЛП. Нам нужно построить матрицу коэффициентов при неизвестных и вектор правых частей

ограничений-неравенств (32.9), (32.3). Неизвестные (столбцы матрицы ограничений) будем нумеровать в порядке: v_0, v_1, ..., v_n, далее $\forall e_{ik}$ в порядке их нахождения в массиве E. Неравенства (строки матрицы) нумеруем в таком порядке: вначале $2m$ ограничений (32.9), а затем n ограничений (32.3). В соответствии с этой нумерацией заполняем матрицу коэффициентов при неизвестных в системе, определяемой неравенствами (32.9), (32.3). Вот как выглядят операторы заполнения этой матрицы.

```
A1=zeros(2*m,n); % блок для vi в (32.9)
for k=1:m, % заполняем коэффициенты для vi: 1 и -1 в (32.9)
  A1(2*k-1,E(k,1:2))=[1 -1];
  A1(2*k,  E(k,1:2))=[-1 1];
end
A=[zeros(2*m,1),A1,...
  reshape([-n*reshape(eye(m),1,m*m);n*reshape(eye(m),1,m*m)],2*m,m);...
  -ones(n,1),eye(n),zeros(n,m)]; % заполнили матрицу коэффициентов
```

Получили матрицу размером $(2m+n)\times(1+n+m)$. Первые $2m$ строк соответствуют неравенствам (32.9), а последние n — (32.3). Теперь — правые части для этой системы:

```
b=[reshape([-ones(1,m);(n-1)*ones(1,m)],2*m,1);zeros(n,1)];
```

Вектор коэффициентов при неизвестных в целевой функции — очень простой: единица при v_0, нули при остальных переменных:

```
c=[1;zeros(n+m,1)]; % коэффициенты в целевой функции
```

Задаем нижние и верхние границы по переменным v_i и e_{ik}:

```
vlb=[ones(n+1,1);zeros(m,1)]; % нижние границы
vub=[n*ones(n+1,1);ones(m,1)]; % верхние границы
```

Для вызова процедуры miqp нужно также задать гессиан (задаем его нулевым), список бинарных переменных (у нас это e_{ik} — только они принимают значения 0 или 1) и начальное приближение.

```
H=zeros(n+m+1); % гессиан
vartype=[n+2:n+m+1]; % список бинарных переменных
x0=[n;[1:n]';zeros(m,1)]; % начальное приближение
```

А теперь — само решение. Вызываем процедуру miqp и возвращаем ответ — список номеров красок вершин.

```
[xmin,fmin]=miqp(H,c,A,b,[],[],vartype,vlb,vub,x0); % вызов miqp
nCol=round(xmin(2:n+1)); % округляем ответ
return
```

32.4. Пример обращения к процедуре *ColorGraph*

Продемонстрируем обращение к процедуре. Для примера возьмем тот же граф, что и в предыдущей главе. Введем исходные данные — координаты вершин и список ребер. Нарисуем граф (рис. 32.3).

```
clear all
V=[[0 0];[1 1];[1 0];[1 -1];...
   [2 1];[2 0];[2 -1];[3 1];...
   [3 0];[3 -1];[4 0]]; % координаты вершин
E=[[1 2];[1 3];[1 4];[2 3];[3 4];[2 5];...
   [2 6];[3 6];[3 7];[4 7];[6 5];[6 7];...
   [5 8];[6 8];[6 9];[7 9];[7 10];[8 9];...
   [9 10];[8 11];[9 11];[10 11]]; % ребра
PlotGraph(V,E); % рисуем граф
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfИсходный граф')
```

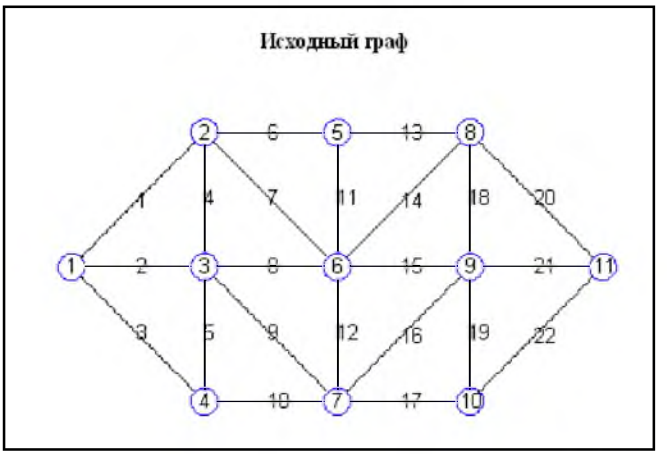


Рис. 32.3. Исходный граф, нарисованный с помощью MATLAB

Этот граф — планарный, поэтому для его раскраски должно хватить 4-х красок. Проверим это. Решим задачу о раскраске и нарисуем результат (рис. 32.4): этот же граф, у которого веса вершин — это найденные номера красок.

```
nCol=ColorGraph(E); % решение задачи о раскраске
PlotGraph([V,nCol],E); % рисуем
```

```
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfПравильная раскраска вершин графа')
```

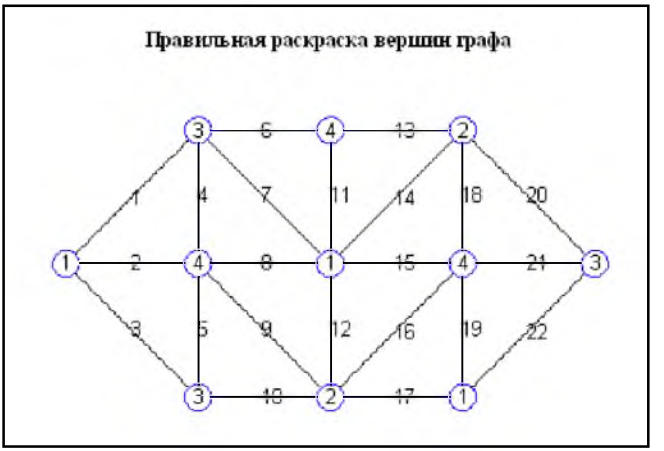
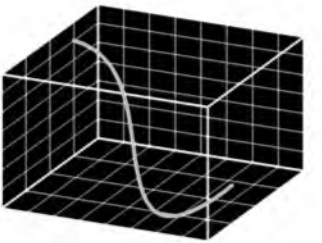


Рис. 32.4. Правильная раскраска вершин графа, выполненная с помощью MATLAB

32.5. Вопросы для самопроверки

1. Найдите в литературе доказательство проблемы 4-х красок для планарных графов. Начните с Трудов американского математического общества за 1976 год.
2. Переделайте процедуру `ColorGraph` под использование функции `milp` вместо `miqp`. Какие задачи она теперь может решать, а какие нет?
3. Попробуйте применить жадный алгоритм к задаче о раскраске. Будет ли он работать?
4. Какие вы еще знаете алгоритмы решения задачи о правильной раскраске графа?

ГЛАВА 33



Кратчайший путь

До сих пор мы рассматривали неориентированные графы (в том числе мультиграфы, псевдографы и гиперграфы). Начиная с этой главы, мы переходим к рассмотрению задач на орграфах.

33.1. Постановка задачи о кратчайшем пути

Пусть задан орграф $G=(V, E)$. Можно ли из вершины v_i попасть в v_j, двигаясь только по стрелкам? Иными словами: существует ли путь из v_i в v_j? Если таких путей несколько, то какой из них кратчайший (имеет минимальное количество дуг)? Для орграфа со взвешенными дугами можно сформулировать задачу так: если существует несколько путей из v_i в v_j, то какой из них имеет минимальный общий вес?

Эти вопросы — различные постановки *задачи о кратчайшем пути*. Как и в других задачах, которые мы решали раньше, невзвешенный орграф можно рассматривать как частный случай взвешенного, если всем дугам приписать веса, равные 1. С другой стороны, можно обобщить и взвешенную задачу: дополнить орграф до взвешенной клики недостающими дугами, приписав им бесконечные веса. Тогда формально можно определять кратчайший путь от любой вершины до любой другой. Если результатом будет бесконечность, то это значит, что реальный путь отсутствует.

Будем предполагать, что все веса дуг в этой задаче неотрицательные. Это существенное ограничение: если вдруг в орграфе обнаружится цикл с общим отрицательным весом, то задача минимизации потеряет смысл: можно будет все время кружить по этому циклу, уменьшая общий вес до минус бесконечности.

Мы рассмотрим два наиболее часто используемых алгоритма решения задачи о кратчайшем пути. В одном из них (алгоритм Дейкстры, E. W. Dijkstra) находятся кратчайшие пути из заданной вершины во все остальные. Другой алгоритм (Флойда — Уоршелла, R. W. Floyd, S. Warshall) решает более общую задачу: строит матрицу кратчайших путей из любой вершины в любую другую. Этот алгоритм мы реализуем на MATLAB.

33.2. Алгоритм Дейкстры

Задана конкретная вершина v_s, и находятся кратчайшие пути из нее во все остальные вершины. Алгоритм основан на постепенном разрастании путей от v_s до всех других вершин по дугам минимального веса. Строящийся на каждом шаге орграф при этом становится похожим на растущий кристалл или трещину. Опишем шаги этого алгоритма. Входным параметром является матрица весов дуг между любой парой вершин. Если какая-либо дуга отсутствует, полагаем вес бесконечным. На выходе получаем вектор M длины n, в координатах которого — кратчайшие пути из заданной вершины v_s до всех остальных.

Шаг 1. Вначале s-й элемент вектора кратчайших путей M полагаем равным 0, а остальные элементы — это веса дуг из v_s во все вершины. Тем самым мы определяем минимальную длину пути за 1 переход по дуге. Если для какой-то вершины v_i соответствующий элемент вектора M равен бесконечности, то это значит, что мы не можем перейти из v_s в v_i за один переход по дуге, т. к. соответствующей дуги нет. Обозначим через W подмножество вершин, в которое включим пока что только одну исходную вершину v_s. На следующих шагах мы будем пополнять W другими вершинами и искать кратчайший путь из v_s в другие вершины за 2 перехода по дугам, затем за 3 и т. д.

Шаг 2. Из всех вершин, не входящих в подмножество W, выберем вершину с кратчайшим путем (элементом вектора M). Если таких вершин несколько, берем первую попавшуюся. Полученную вершину (обозначим ее v_i) добавляем в подмножество W, и i-й элемент вектора M оставляем без изменения. Для остальных вершин, не входящих в W, пересчитываем длину кратчайшего пути (координату вектора M). Делаем это так: для каждой j-й вершины находим сумму i-го элемента вектора M и веса дуги e_{ij}. Если эта сумма меньше j-го элемента вектора M, ставим эту сумму на его место.

Шаг 3. Если в W осталось включить только последнюю вершину, то выходим, а если больше одной — идем на шаг 2. □

Продемонстрируем работу алгоритма на примере из книги [33] (рис. 33.1).

На рис. 33.1, *а* показан исходный орграф, возле дуг обозначены их веса. Требуется найти кратчайшие пути от v_1 до остальных вершин. Проводим 1-й шаг.

Расстояния от v_1 до остальных вершин заполняем в соответствии с весами дуг. Если дуги нет, пишем бесконечность. В подмножество W заносим одну вершину v_1. Этот этап показан на рис. 33.1, *б*. Здесь вместо номеров вершин написаны расстояния от v_1 до них на 1-м шаге, т. е. текущее состояние вектора M. Подмножество W выделено штриховой линией.

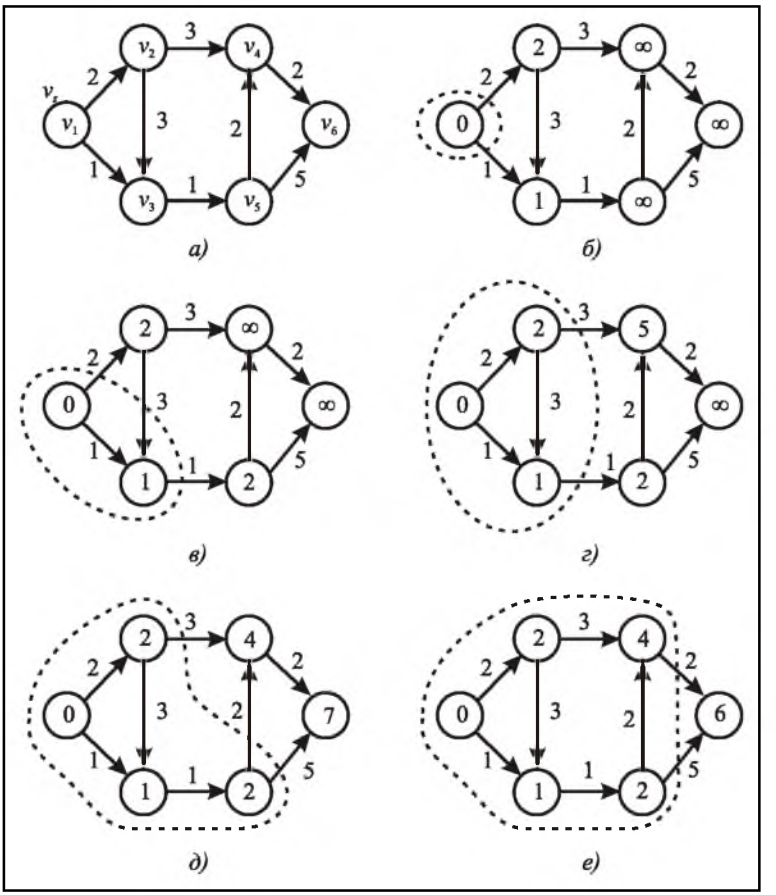


Рис. 33.1. Алгоритм Дейкстры решения задачи о кратчайшем пути

Выполняем шаг 2. На рис. 33.1, *б* среди всех вершин, не входящих в W, минимальное значение пути, равное 1, — у v_3, поэтому присоединяем ее к W (рис. 33.1, *в*). Пересчитываем пути для вершин, не входящих в W. Здесь далее через |...| обозначены пути до вершин и веса ребер. Проверяем вершину v_2: $|v_2| = 2$; $|v_1| + |e_{12}| = \infty$. Уменьшения нет — оставляем $|v_2| = 2$ без изменения. Следующая в списке v_4: $|v_4| = \infty$; $|v_1| + |e_{14}| = \infty$. Также нет уменьшения. Далее

проверяем v_5: $|v_5|=\infty$; $|v_1|+|e_{15}|=2$. Есть уменьшение, поэтому теперь будет $|v_5|=2$. И, наконец, v_6: $|v_6|=\infty$; $|v_1|+|e_{16}|=\infty$. Оставляем без изменений.

Шаг 3: проверяем, сколько вершин не попало в W. Осталось более одной вершины — снова идем на шаг 2. Теперь минимальный путь у вершин v_2 и v_4: по 2. Первой в списке идет v_2 — ее и добавляем в W. Пересчитываем пути до остальных вершин (рис. 33.1, *г*). Для v_4: $|v_4|=\infty$; $|v_2|+|e_{24}|=5$. Есть уменьшение, поэтому полагаем $|v_4|=5$. Для v_5: $|v_5|=2$; $|v_2|+|e_{25}|=\infty$ — оставляем без изменения. И для v_6: $|v_6|=\infty$; $|v_2|+|e_{26}|=\infty$ — также оставляем без изменения. Проверка на шаге 3 показывает, что осталось более одной вершины, не включенной в W, поэтому продолжаем процесс.

На этом этапе (это снова шаг 2) минимальный путь у v_5: $|v_5|=2$. Добавляем эту вершину в W. Остались 2 вершины, не включенные в W: v_4 и v_6 (рис. 33.1, *д*). Пересчитываем их пути: $|v_4|=5$; $|v_5|+|e_{54}|=4$ — есть уменьшение, полагаем $|v_4|=4$. Далее: $|v_6|=\infty$; $|v_5|+|e_{56}|=7$ — тоже есть уменьшение, поэтому теперь $|v_6|=7$. Шаг 3: пока что осталось 2 вершины, поэтому снова идем на шаг 2.

Из двух оставшихся вершин минимальный путь у v_4: $|v_4|=4$. Ее и добавляем в W (рис. 33.1, *е*). Проверяем v_6: $|v_6|=7$; $|v_4|+|e_{46}|=6$ — уменьшаем $|v_6|$ до 6. Шаг 3: осталась единственная вершина v_6, поэтому заканчиваем алгоритм. В каждой вершине — кратчайший путь от v_1 до нее.

Правильность алгоритма Дейкстры легко доказывается по индукции, т. к. на каждой итерации путь до новой вершины находится таким образом, что получаются кратчайшие пути для всех вершин из W.

33.3. Алгоритм Флойда — Уоршелла

В этом алгоритме строится матрица кратчайших путей между всеми парами вершин орграфа. На входе нужно задать матрицу D размера $n \times n$, в каждом элементе которой d_{ij} находится вес дуги — длина пути из v_i в v_j. При этом диагональные элементы должны быть равны бесконечности: $\forall d_{ii}=\infty$. Весь алгоритм умещается в тройной цикл. Вот как он выглядит в синтаксисе MATLAB:

```
for j=1:n do,
  for i=setdiff([1:n],j) do,
    for k=setdiff([1:n],j) do,
      D(i,k)=min(D(i,k),D(i,j)+D(j,k));
    end
  end
end
```

Смысл этого тройного цикла показан на рис. 33.2: для каждой промежуточной вершины v_j мы проверяем все возможные начальные вершины v_i и конечные v_k, не совпадающие с v_j, но, возможно, совпадающие между собой. Если при каждой такой проверке сумма весов дуг $d_{ij}+d_{jk}$ окажется меньше d_{ik}, мы заменяем более длинный путь d_{ik} более коротким $d_{ij}+d_{jk}$. Эта операция называется операцией треугольника.

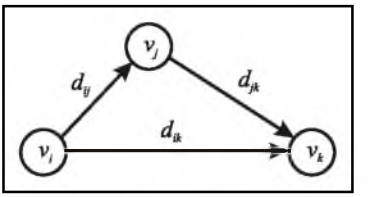


Рис. 33.2. Операция треугольника

Определение 33.1. *Операцией треугольника* для фиксированной вершины v_j называется операция:

$$\begin{cases} d_{ik} = \min\left(d_{ik}, d_{ij} + d_{jk}\right); \\ i \in [1, n]; \quad i \neq j; \\ k \in [1, n]; \quad k \neq j; \end{cases} \tag{33.1}$$

где допускается $i=k$. □

Таким образом, алгоритм Флойда — Уоршелла заключается в проведении операции треугольника для каждой вершины. Оказывается, этого вполне достаточно!

■ **ТЕОРЕМА 33.1.** Если все веса дуг неотрицательные, то после однократного проведения операции треугольника для всех вершин матрица D становится матрицей кратчайших путей.

Доказательство. Докажем по индукции, что после шага номер s матрица D станет матрицей кратчайших путей из любой вершины в любую по вершинам с номерами $j \le s$. Начнем с $s=1$. Если провести операцию (33.1) только для $j=1$, то пути из любой вершины в любую через v_1 будут кратчайшими. Значит, для $s=1$ теорема имеет место. Предположим, что теорема верна для $s=r-1$: пути из любой вершины в любую через вершины с номерами от 1 до $r-1$ минимальны по длине. Проведем операцию треугольника еще и для промежуточной вершины номер r: $d_{ik}=\min(d_{ik}, d_{ir}+d_{rk})$. Докажем, что в этом случае путь из любой вершины в любую, проходящий через промежуточные вершины v_1, v_2, ..., v_r, будет кратчайшим. Если реальный кратчайший путь на самом деле не проходит через новую вершину v_r, то минимальным будет пер-

вое число d_{ik}, и в этом случае теорема верна, т. к. добавление новой промежуточной вершины v_r ничего не меняет. Если же реальный кратчайший путь пройдет через новую вершину v_r, то d_{ik} будет заменено на меньшую величину $d_{ir}+d_{rk}$. Но каждое из слагаемых d_{ir} и d_{rk} является кратчайшим путем через все промежуточные вершины v_1, v_2, ..., v_{r-1}, поэтому $d_{ir}+d_{rk}$ действительно будет кратчайшим путем из v_i в v_k через промежуточные вершины v_1, v_2, ..., v_r. Таким образом, по индукции теорема доказана. □

33.4. Описание процедуры *ShortPath*

Рассмотрим реализацию алгоритма Флойда — Уоршелла на MATLAB. На вход будем подавать список дуг размером $m \times 2$ или $m \times 3$ (идентификатор E). Как и ранее, будем предполагать, что, если пользователь задал 2 столбца, то нужно решать невзвешенную задачу и искать путь из минимального количества дуг, а если 3 — то взвешенную, и в этом случае ищем пути минимального общего веса. На выходе процедура будет возвращать матрицу кратчайших путей размера $n \times n$. Вот как выглядит заголовок процедуры и справочная информация к ней:

```
function dSP=ShortPath(E)
% Функция dSP=ShortPath(E) решает задачу о кратчайшем пути
% между всеми вершинами орграфа.
% Входной параметр:
%   E(m,2) или (m,3) - дуги орграфа и их веса;
%     1-й и 2-й элементы каждой строки - это номера вершин;
%     3-й элемент каждой строки - это вес дуги;
%     m - количество дуг.
%     Если задан массив E(m,2), то все веса равны 1.
% Выходной параметр:
%   dSP(n,n) - матрица кратчайших путей.
%   Каждый элемент dSP(i,j) - это кратчайший путь
%   из вершины i в вершину j (может быть inf,
%   если вершина j не достижима из вершины i).
% Используется алгоритм Флойда-Уоршелла.
% Выражаю признательность проф. Жерару Био (Университет Монпелье-II,
% Франция) за тестирование этого алгоритма.
```

Проверка исходных данных организована так же, как и в других функциях пакета. Проверяем наличие данных, размерность и количество столбцов входного массива, положительность и целочисленность первых двух столбцов.

```
if nargin<1,
  error('Нет исходных данных!')
end
```

```
sz=size(E); % размеры массива E
if length(sz)~=2,
  error('Массив E должен быть двумерным!')
end
if (sz(2)<2),
  error('В массиве E должно быть 2 столбца!'),
end
if ~all(all(E(:,1:2)>0)),
  error('1-й и 2-й столбцы массива E должны быть положительными!')
end
if ~all(all((E(:,1:2)==round(E(:,1:2))))),
  error('1-й и 2-й столбцы массива E должны быть целыми!')
end
```

Находим размер и мощность орграфа. Если пользователь не задал веса дуг, задаем их единичными.

```
m=size(E,1); % количество дуг
n=max(max(E(:,1:2))); % количество вершин
if sz(2)==2, % веса дуг не заданы
  E(:,3)=1; % все веса =1
end
```

Для запуска алгоритма Флойда — Уоршелла нужно задать матрицу весов дуг размером $n \times n$. Если какой-либо дуги нет, в соответствующем элементе должно быть бесконечное число. Заполняем такую матрицу.

```
dSP=ones(n)*inf; % сначала - все элементы равны бесконечности
for k=1:m, % просматриваем каждую дугу
  dSP(E(k,1),E(k,2))=E(k,3); % заносим вес дуги на нужное место
end
```

И наконец, проводим основной цикл алгоритма Флойда — Уоршелла. Обратите внимание, как он организован:

```
for j=1:n,
  i=setdiff((1:n),j);
  dSP(i,i)=min(dSP(i,i),repmat(dSP(i,j),1,n-1)+repmat(dSP(j,i),n-1,1));
end
return
```

Первая функция `repmat` копирует j-й столбец нашей матрицы (с выброшенным j-м элементом) $n-1$ раз по горизонтали: получается матрица размером $(n-1)\times(n-1)$. Вторая функция `repmat` копирует j-ю строку (с выброшенным j-м элементом) также $n-1$ раз, но уже по вертикали. При этом также получа-

ется матрица размером $(n-1)\times(n-1)$. При их сложении мы фактически складываем каждый элемент с каждым. Далее мы сравниваем эту сумму с исходной матрицей, у которой выброшены j-я строка и j-й столбец. Следовательно, этот один оператор заменяет два внутренних цикла и полностью реализует операцию треугольника (33.1).

Подобные приемы программирования на MATLAB нужно стремиться применять везде, где только можно. Любая встроенная функция (в данном случае `repmat`) работает гораздо быстрее, чем цикл.

33.5. Пример обращения к процедуре *ShortPath*

Рассмотрим пример обращения к процедуре `ShortPath`. Возьмем тот же граф, что и в *главе 32*, но добавим веса дуг и будем считать его ориентированным. Нарисуем его (рис. 33.3).

```
clear all
V=[[0 0];[1 1];[1 0];[1 -1];...
   [2 1];[2 0];[2 -1];[3 1];...
   [3 0];[3 -1];[4 0]]; % координаты вершин
E=[[1 2 5];[1 3 5];[1 4 5];[2 3 2];[3 4 2];[2 5 3];...
   [2 6 2];[3 6 5];[3 7 2];[4 7 3];[6 5 1];[6 7 1];...
   [5 8 5];[6 8 2];[6 9 3];[7 9 2];[7 10 3];[8 9 2];...
   [9 10 2];[8 11 5];[9 11 4];[10 11 4]]; % ребра и их веса
PlotGraph(V,E,'o'); % рисуем орграф
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfИсходный орграф со взвешенными дугами')
```

Найдем матрицу кратчайших путей между любой парой вершин и напечатаем ее.

```
dSP=ShortPath(E);
disp('Кратчайшие пути между всеми вершинами:');
fprintf('    %2.0f',1:size(dSP,2));
fprintf('\n');
for k1=1:size(dSP,1),
  fprintf('%2.0f',k1)
  fprintf('%6.2f',dSP(k1,:))
  fprintf('\n')
end
```

```
Кратчайшие пути между всеми вершинами:
     1     2     3     4     5     6     7     8     9    10    11
 1   Inf  5.00  5.00  5.00  8.00  7.00  7.00  9.00  9.00 10.00 13.00
 2   Inf   Inf  2.00  4.00  3.00  2.00  3.00  4.00  5.00  6.00  9.00
 3   Inf   Inf   Inf  2.00  6.00  5.00  2.00  7.00  4.00  5.00  8.00
 4   Inf   Inf   Inf   Inf   Inf   Inf  3.00   Inf  5.00  6.00  9.00
 5   Inf   Inf   Inf   Inf   Inf   Inf   Inf  5.00  7.00  9.00 10.00
 6   Inf   Inf   Inf   Inf  1.00   Inf  1.00  2.00  3.00  4.00  7.00
 7   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf  2.00  3.00  6.00
 8   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf  2.00  4.00  5.00
 9   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf  2.00  4.00
10   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf  4.00
11   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
```

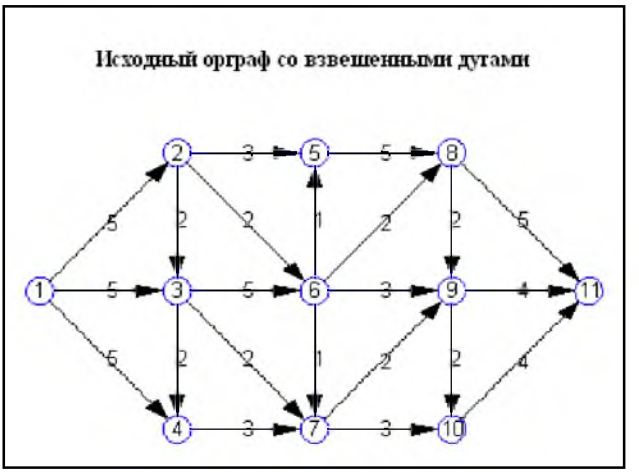


Рис. 33.3. Исходный орграф со взвешенными дугами, нарисованный с помощью MATLAB

33.6. Вопросы для самопроверки

1. Проведите формальное доказательство алгоритма Дейкстры.
2. Какова полиномиальная сложность алгоритмов Дейкстры и Флойда — Уоршелла?
3. Как в алгоритмах Дейкстры и Флойда — Уоршелла организовать вывод последовательности вершин, реализующих кратчайший путь?
4. Найдите на сайте [51] или реализуйте самостоятельно алгоритм Дейкстры и сравните его скорость со скоростью алгоритма Флойда — Уоршелла.

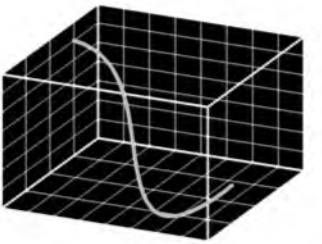
5. Замените последний фрагмент процедуры `ShortPath` на тройной цикл, приведенный в начале *раздела 33.3*. Сравните скорость работы обоих вариантов на примере большого орграфа с одними петлями[1]:

```
E=repmat([1:1000]',1,2);
ShortPath(E)
```

6. Сформулируйте задачу нахождения кратчайшего поту из v_i в v_j как задачу ЦЛП. Реализуйте этот алгоритм на MATLAB и сравните его скорость со скоростями алгоритмов Дейкстры и Флойда — Уоршелла.

[1] Этот пример принадлежит проф. Жерару Био (Университет Монпелье-II, Франция) и публикуется с его разрешения.

ГЛАВА 34



Разбиваем и упорядочиваем

Взглянем на орграф с несколько иных позиций. По *определению 28.19* любая дуга — это *упорядоченная* совокупность двух вершин (v_i, v_k), что можно рассматривать как бинарное отношение между вершинами. Поэтому определим вначале некоторые понятия, связанные с бинарными отношениями.

34.1. Бинарные отношения

Определение 34.1. *Декартовым произведением* $A \times B$ двух множеств A и B называется множество всех возможных упорядоченных пар элементов, в которых первый элемент $a \in A$, а второй $b \in B$:

$$A \times B = \{(a, b): (a \in A) \cap (b \in B)\}. \square \qquad (34.1)$$

Для конечных множеств с $|A| = n$, $|B| = m$ элементы $A \times B$ полностью заполняют клетки матрицы размером $n \times m$. Примером декартового произведения для однозначных целых чисел является хорошо известная из школы таблица умножения.

Определение 34.2. *Бинарным отношением* называется подмножество декартового произведения некоторого множества A на себя. $\square$

Бинарное отношение удобно обозначать знаком *между* элементами множества. Например, на множествах натуральных, целых, рациональных и действительных чисел определены бинарные отношения $=$, $\neq$, $>$, $<$, $\geq$, $\leq$. Поэтому запись $5 \geq 3$ фактически означает, что элемент декартового произведения $(5, 3)$ принадлежит бинарному отношению "больше или равно": $(3, 5) \in \geq$. Но запись $5 \geq 3$ более удобна, чем $(3, 5) \in \geq$, поэтому обычно пользуются именно ею. Обратное соотношение не имеет места, поэтому говорят, что $(3, 5) \notin \geq$

(элемент декартового произведения $(3,5)$ не принадлежит бинарному отношению "больше или равно").

С этой точки зрения множество дуг орграфа можно рассматривать как бинарное отношение на множестве вершин, которое мы назовем *достижимостью* и будем обозначать стрелкой $\rightarrow$ или $\leftarrow$. Когда мы говорим, что из вершины v_i достигается вершина v_j, то это эквивалентно тому, что существует дуга из v_i в v_j, или бинарное отношение $v_i \rightarrow v_j$. И наоборот, выражение "вершина v_j достижима из вершины v_i" означает существование дуги в v_j из v_i, или бинарное отношение $v_j \leftarrow v_i$. Мы будем считать, что оба эти выражения и отношения эквивалентны между собой.

Рассмотрим классификацию бинарных отношений [13]. Здесь для их обозначения используется символ $\asymp$, чтобы не привязываться ни к какому конкретному бинарному отношению. Элементы множества A обозначены a или a_1, a_2, a_3, ...

Определение 34.3. Бинарное отношение $\asymp \subseteq A \times A$ на A называется:

- *рефлексивным*, если $a \asymp a$ имеет место $\forall\, a \in A$;
- *транзитивным*, если из выполнения $a_1 \asymp a_2$, $a_2 \asymp a_3$ следует, что $a_1 \asymp a_3$ $\forall\, a_1, a_2, a_3 \in A$;
- *полным*, если $a_1 \asymp a_2$ или $a_2 \asymp a_1$ выполняется $\forall\, (a_1, a_2) \in A \times A$;
- *антисимметричным*, если из одновременного выполнения $a_1 \asymp a_2$, $a_2 \asymp a_1$ следует, что $a_1 = a_2$ (элементы эквивалентны в смысле некоторого определения, которое еще нужно ввести в рассмотрение);
- *асимметричным*, если из выполнения $a_1 \asymp a_2$ следует, что $a_2 \asymp a_1$ не выполняется никогда;
- *симметричным*, если из выполнения $a_1 \asymp a_2$ следует, что обязательно выполняется $a_2 \asymp a_1$;
- *квазиупорядоченным*, если оно рефлексивное и транзитивное;
- *эквивалентным*, если оно рефлексивное, транзитивное и симметричное;
- *предупорядоченным*, если оно рефлексивное, транзитивное и полное;
- *частично упорядоченным*, если оно рефлексивное, транзитивное и антисимметричное;
- *полностью упорядоченным*, если оно рефлексивное, транзитивное, полное и антисимметричное. □

Рассмотрим, какие из этих требований выполняются для орграфов и бинарного отношения $\rightarrow$ на множестве его вершин V.

Рефлексивность. Даже если в вершине v нет петель, мы вправе считать, что, если мы уже находимся в вершине v, то мы ее достигли. Поэтому будем считать, что $v \to v$ выполняется $\forall v$, и, значит, бинарное отношение $\to$ рефлексивно.

Транзитивность. Если из v_1 достигается v_2, а из v_2 — v_3, то мы можем утверждать, что v_3 достижима и из v_1 (уже не за один, а за два перехода по дугам). Значит, бинарное отношение $\to$ транзитивно.

Полнота. Не обязательно! Например, на рис. 28.1, *б* $v_1 \to v_4$, но $(v_4, v_1) \notin \to$.

Антисимметричность. Для проверки этого требования нужно ввести понятие эквивалентных вершин.

Определение 34.4. Вершины орграфа v_i и v_j называются *эквивалентными в смысле достижимости*, если существуют пути по дугам из v_i в v_j и из v_j в v_i. □

В смысле определения 34.4 бинарное отношение $\to$ является антисимметричным.

Асимметричность. Это требование не выполняется: могут существовать орграфы, у которых есть взаимно достижимые вершины.

Симметричность. Также не выполняется, т. к. могут существовать орграфы, у которых $v_i \to v_j$, но $(v_j, v_i) \notin \to$. Пример такого орграфа — на рис. 28.1, *б*.

Следовательно, множество вершин орграфа с введенным на нем бинарным отношением $\to$ является *частично упорядоченным*, т. к. оно рефлексивное, транзитивное и антисимметричное.

Из теории множеств известно, что, введя понятие эквивалентности, мы можем разбить любое множество (в данном случае множество вершин орграфа) на классы эквивалентности, в каждый из которых попадают только взаимно эквивалентные (взаимно достижимые) вершины. Другой важный вывод теории множеств: для частично упорядоченных множеств их классы эквивалентности также частично упорядочиваются.

ПРИМЕР 34.1. На рис. 34.1, *а* показан орграф с 5 вершинами. Они разбиваются на 4 класса эквивалентности, которые обведены замкнутыми штриховыми линиями. На рис. 34.1, *б* изображено их частичное упорядочение: классы c_i расположены линейно, и отношения достижимости между ними (стрелки) направлены только слева направо. Но упорядочение здесь действительно частичное: для классов c_2 и c_3 бинарное отношение $\to$ не определено ни в одну, ни в другую сторону. Их можно было бы поменять на рисунке местами, при этом упорядочение сохранилось бы. □

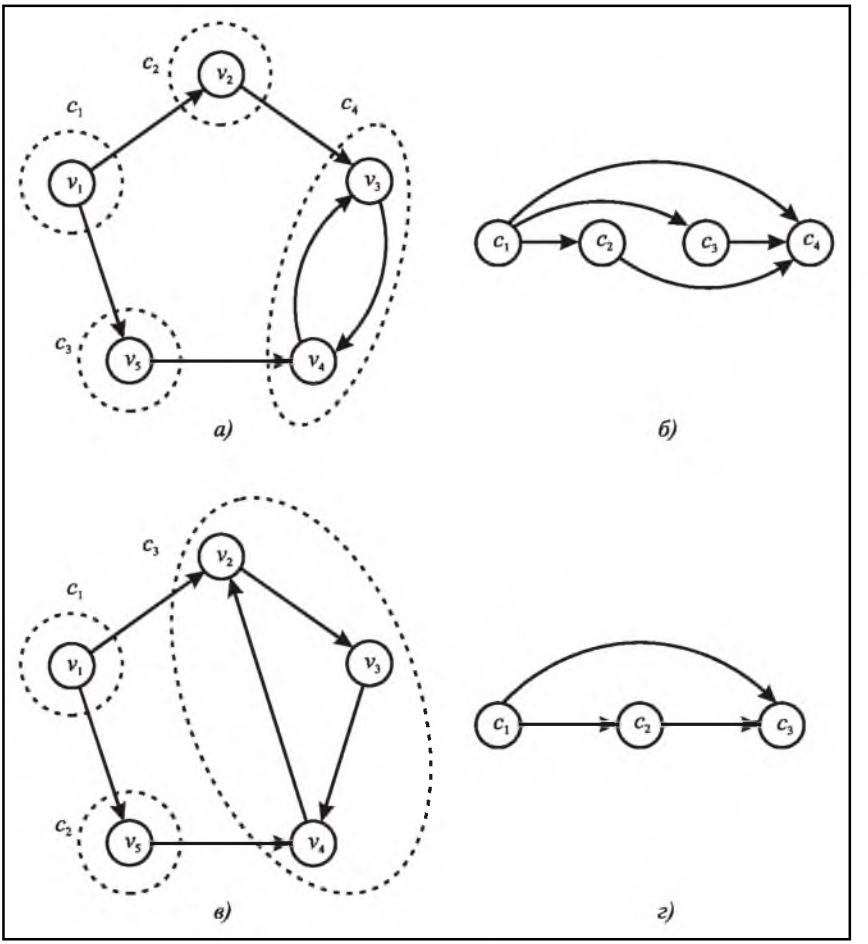


Рис. 34.1. Разбиение множества вершин V на классы эквивалентности и их частичное упорядочение

Упорядочение удобно задавать в виде булевой матрицы **R** размером $N \times N$, где N — количество классов эквивалентности ($N \le n$). Каждый ее элемент r_{ij} = true, если между классами c_i и c_j существует отношение достижимости. Наша задача — так перенумеровать классы, чтобы ниже главной диагонали остались лишь 0 (false). Так, для частичного упорядочения на рис. 34.1, *б* эта матрица имеет вид:

$$\mathbf{R} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \qquad (34.2)$$

Если мы теперь заменим дугу e_{43} на e_{42} (рис. 34.1, *в*), то вместо 4 классов эквивалентности получим только 3. Их упорядочение из частичного становится уже полным (рис. 34.1, *г*), а его матрица будет иметь вид:

$$\mathbf{R} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}. \qquad (34.3)$$

На основе разбиения множества вершин на классы эквивалентности и их частичного упорядочения можно решать различные задачи. Этот алгоритм можно использовать, например, в социологических исследованиях [13]. С его помощью легко выявляются лидеры и группы влияния в различных объединениях граждан: рабочих коллективах, школьных классах, студенческих группах, руководстве политических партий, клубах по интересам.

Другая интересная проблема, которая может быть решена на основе данного алгоритма, — это *стягивание* орграфа и уменьшение количества его вершин. Взаимно достижимые вершины объединяются в одну, а дуги между ними удаляются. Далее можно ставить задачу минимизации количества дуг с тем, чтобы оставить в силе имеющееся частичное упорядочение, и т. д. Но мы рассмотрим только два первых этапа: разбиение множества вершин на классы эквивалентности по достижимости и частичное упорядочение этих классов.

34.2. Разбиение на классы эквивалентности

Один из простейших алгоритмов разбиения множества вершин V на подмножества из взаимно достижимых вершин может быть таким. Пусть в нашем распоряжении есть булева матрица достижимости D размером $n \times n$, каждый элемент которой $d_{ij} = \text{true}$, если $v_i \to v_j$, и $d_{ij} = \text{false}$, если $(v_i, v_j) \notin \to$. Мы всегда можем построить такую матрицу с помощью процедуры `ShortPath`. В этой матрице $d_{ij} = d_{ji} = \text{true}$ тогда и только тогда, когда v_i и v_j взаимно достижимы (т. е. эквивалентны по определению 34.4). Мы должны объединить их в один класс. Такое объединение состоит в том, что все вершины, которые были достижимы из v_i или v_j, будут достижимыми и из их объединения. И наоборот, полученное объединение будет достижимым из всех вершин, из которых были достижимы и v_i, и v_j. Проще всего этого добиться путем сложения (объединения) соответствующих строк и столбцов матрицы D, как показано на рис. 34.2 в верхней части.

Как только мы обнаружим в матрице D пару симметричных истинных элементов $d_{ij} = d_{ji} = \text{true}$, нужно выполнить такие действия:

- прибавить к i-му столбцу матрицы D ее j-й столбец и удалить j-й столбец;
- прибавить к i-й строке матрицы D ее j-ю строку и удалить j-ю строку.

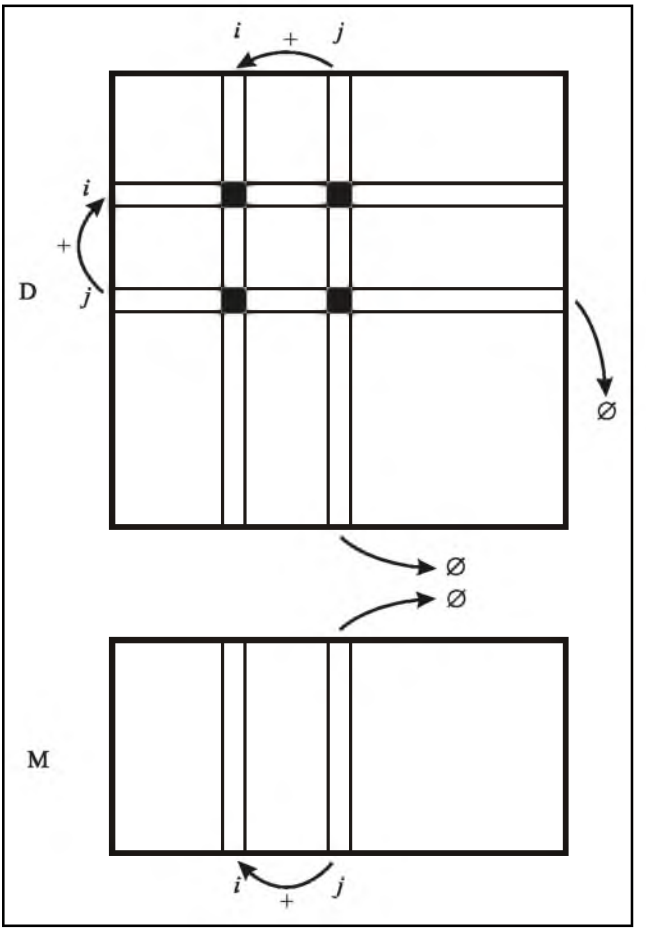


Рис. 34.2. Алгоритм разбиения V на подмножества взаимно достижимых вершин

После такой операции матрица достижимости уменьшит свои размеры на 1. Далее мы ищем следующую пару симметричных истинных элементов и повторяем процесс. Так поступаем до тех пор, пока удается найти $d_{ij} = d_{ji} = \text{true}$. Если все такие пары исчерпаны, итерационный процесс заканчивается: все взаимно достижимые вершины объединены в классы эквивалентности по достижимости.

Но это еще не все. Нам нужно где-то хранить номера вершин, входящих в тот или иной класс. Для этого перед итерациями создадим еще одну матрицу M — диагональную булеву матрицу размером $n \times n$. В каждом ее столбце истинные элементы будут соответствовать номерам вершин, входящих в данный класс. Перед началом итераций у нас есть n классов, в каждый из кото-

рых входит по одной вершине, поэтому исходная M — диагональная. Как только находится пара $d_{ij}=d_{ji}=$true, то кроме описанных выше двух действий выполняем еще и третье:

- прибавить к i-му столбцу матрицы M ее j-й столбец и удалить j-й столбец.

В результате такого действия в i-м столбце будут сохранены номера вершин, которые объединяются в один класс эквивалентности. Это действие показано в нижней части рис. 34.2.

После выполнения описанного выше итерационного процесса все вершины будут объединены в классы эквивалентности, а матрица достижимости вершин станет матрицей достижимости классов. Теперь нужно провести частичное упорядочение этих классов.

34.3. Алгоритмы упорядочения

Задача линейного упорядочения объектов интересна сама по себе. Наиболее простой и очевидный путь ее решения — это сравнивать каждый элемент с любым другим. Такой алгоритм требует $n(n-1)/2$ операций сравнения. Более экономно добавлять по одному элементу в уже упорядоченную последовательность. Если часть элементов упорядочена, то вновь добавляемый элемент не обязательно сравнивать со всеми предыдущими. Мы можем сравнивать его с 1-м, 2-м и т. д., пока не найдем его место в последовательности. В худшем случае будем иметь те же $n(n-1)/2$ операций сравнения, но "в среднем" их будет меньше. В идеальном случае вообще после 1-го же сравнения новый элемент сразу становится на место.

Но и этот алгоритм не наилучший. Можно применить дискретный вариант метода половинного деления. Когда в упорядоченную последовательность добавляется новый элемент, его в первую очередь сравнивают со средним элементом последовательности (или с одним из двух средних, если их число четно). Тем самым мы определяем половину, в которой будет находиться новый элемент. Далее процесс деления этой половины пополам продолжается до тех пор, пока положение нового, добавляемого элемента не будет определено однозначно.

К сожалению, у нас эти алгоритмы не будут работать, т. к. упорядочение у нас только частичное. Некоторые классы эквивалентности вообще нельзя сравнивать. Но, оказывается, нам вообще не нужно сравнивать между собой классы эквивалентности, т. к. результат сравнения у нас уже есть: это матрица достижимости D. Нам нужно только расставить классы в нужном порядке. Порядок определяется тем, что в матрице D не должно быть истинных элементов ниже главной диагонали.

Поэтому, если такой элемент d_{ij} обнаружится, выполняем такие действия:

- меняем местами i-й и j-й столбцы матрицы D;
- меняем местами i-ю и j-ю строки матрицы D;
- меняем местами i-й и j-й столбцы матрицы M.

В результате мы поменяем местами i-й и j-й классы эквивалентности и соответствующим образом скорректируем матрицу достижимости.

Этот процесс нужно продолжать до тех пор, пока в матрице D есть истинные элементы ниже главной диагонали.

34.4. Описание процедуры *DecompPartOrder*

Как всегда, начнем с заголовка. Для работы алгоритма нужен список дуг орграфа размером $m \times 2$ (веса здесь не нужны). Обозначим его идентификатором `E`. Это будет входной параметр. На выходе получаем две матрицы, которые в описаниях алгоритмов были обозначены M и D. Первая из них будет иметь размер $N \times n$, а вторая $N \times N$, где N — количество найденных классов эквивалентности. В процедуре они обозначены идентификаторами `Decomp` и `PartOrder` соответственно. Записываем заголовок и справочную информацию.

```
function [Decomp,PartOrder]=DecompPartOrder(E)
% Функция [Decomp,PartOrder]=DecompPartOrder(E) решает задачу
% разбиения орграфа на классы из взаимно достижимых вершин и
% их частичного упорядочения
% Входной параметр:
%   E(m,2) - список дуг орграфа;
%     1-й и 2-й элементы каждой строки - это номера вершин;
%     m - количество дуг.
% Выходные параметры:
%   Decomp(n,ns) - булевская матрица с номерами вершин.
%     n - количество вершин;
%     ns - количество классов из взаимно достижимых вершин.
%     В каждом столбце массива Decomp истинные значения имеют элементы
%     с номерами вершин этого класса.
%     Все столбцы частично упорядочены.
%   PartOrder(ns,ns) - булевская матрица частичного упорядочения классов.
%     Это - правая верхняя треугольная матрица.
%     Если PartOrder(i,j)=True, то из класса i
%     (вершины i-го столбца матрицы Decomp) достижим класс j
%     (вершины j-го столбца матрицы Decomp).
```

Вначале нам нужно построить матрицу достижимости. Для этого вызываем процедуру `ShortPath`, где заодно и проверим исходные данные. Затем конечные элементы матрицы кратчайших путей заменяем единицами, а бесконечные — нулями. Добавим также единицы на главной диагонали: это соответствует рефлексивности бинарного отношения $\rightarrow$.

```
dSP=ShortPath(E); % проверка данных и кратчайшие пути
PartOrder=(~isinf(dSP)+eye(size(dSP)))>0; % булева матрица достижимости
```

Еще нам нужно задать матрицу с номерами вершин, столбцы которой мы будем объединять.

```
n=size(PartOrder,1); % количество вершин
Decomp=eye(n); % начальные классы - по 1 вершине
```

Начинаем процесс объединения вершин в классы. Он оформлен в виде цикла `for`, т. к. максимальное количество итераций известно заранее. Если необходимо будет выйти из цикла раньше, мы воспользуемся командой `break`. Ищем первую попавшуюся пару симметричных истинных элементов. Если таких пар нет — выходим из цикла.

```
for it=1:n*(n-1)/2, % максимальное количество итераций
  Msym=triu(PartOrder+PartOrder',1);
  % прибавили транспонированную, выделили верхний правый треугольник
  [is,js,Mw]=find(Msym==2); % ищем пару симметричных истинных элементов
  if isempty(is), % нет такой пары
    break; % выходим из цикла for
  end
  i1=is(1); % номер строки первой пары
  j1=js(1); % номер столбца первой пары
```

Выполняем операции по объединению строк и столбцов матрицы D.

```
  PartOrder(i1,:)=PartOrder(i1,:)|PartOrder(j1,:); % прибавляем строку
  PartOrder(:,i1)=PartOrder(:,i1)|PartOrder(:,j1); % прибавляем столбец
```

Удаляем j-й столбец и j-ю строку из матрицы достижимости.

```
  cNv=size(PartOrder,1); % текущий размер матрицы
  PartOrder=PartOrder(setdiff([1:cNv],j1),setdiff([1:cNv],j1)); % удаляем
```

Теперь работаем с матрицей, в столбцах которой — список вершин. Объединяем два столбца и удаляем один из них.

```
  Decomp(:,i1)=Decomp(:,i1)+Decomp(:,j1); % добавляем столбец
  Decomp=Decomp(:,setdiff([1:cNv],j1)); % удаляем добавленный
end % окончание алгоритма объединения
```

На этом первая часть процедуры (объединение вершин в классы эквивалентности) заканчивается. Теперь переходим к сортировке. Как и в первой части процедуры, используем цикл `for` и выход из него по команде `break`. Находим истинные элементы в матрице достижимости ниже главной диагонали.

```
ns=size(PartOrder,1); % количество классов
for it=1:ns*(ns-1)/2, % итерации частичного упорядочения
  Mlow=tril(PartOrder,-1); % нижний левый треугольник
  [is,js,Mw]=find(Mlow); % ищем неупорядоченную пару классов
```

Если таких элементов нет — выходим из цикла: частичное упорядочение закончено.

```
  if isempty(is), % все классы упорядочены
    break; % выходим из цикла for
  end
```

Если же такие элементы есть, берем первый попавшийся и выполняем частичное упорядочение. Меняем местами строки и столбцы в матрице упорядочения, а также столбцы в матрице с номерами вершин.

```
  i1=is(1); % номер строки
  j1=js(1); % номер столбца
  tmp=PartOrder(i1,:);
  PartOrder(i1,:)=PartOrder(j1,:);
  PartOrder(j1,:)=tmp; % меняем строки в матрице упорядочения
  tmp=PartOrder(:,i1);
  PartOrder(:,i1)=PartOrder(:,j1);
  PartOrder(:,j1)=tmp; % меняем столбцы в матрице упорядочения
  tmp=Decomp(:,i1);
  Decomp(:,i1)=Decomp(:,j1);
  Decomp(:,j1)=tmp; % меняем столбцы в матрице с номерами вершин
end
return
```

34.5. Пример обращения к процедуре *DecompPartOrder*

Продемонстрируем работу процедуры на орграфе с 25 вершинами и 46 дугами, который мы использовали в *разделе 28.4*. Введем исходные данные и нарисуем орграф (рис. 34.3).

```
clear all
V=[[0 4];[1 4];[2 4];[3 4];[4 4];[0 3];[1 3];[2 3];[3 3];[4 3];...
   [0 2];[1 2];[2 2];[3 2];[4 2];[0 1];[1 1];[2 1];[3 1];[4 1];...
   [0 0];[1 0];[2 0];[3 0];[4 0]];
```

```
E=[[ 1  2];[ 3  2];[ 4  3];[ 5  4];[ 6  1];...
   [ 2  7];[ 8  2];[ 3  8];[ 9  4];[ 9  5];...
   [10  5];[ 7  6];[ 8  7];[ 8  9];[10  9];...
   [11  6];[ 7 12];[13  8];[14  9];[15 10];...
   [12 11];[13 12];[13 14];[14 13];[15 14];...
   [16 11];[12 17];[13 18];[20 15];[17 16];...
   [17 18];[18 17];[19 18];[19 20];[21 16];...
   [17 22];[18 22];[22 18];[18 23];[19 24];...
   [20 25];[21 22];[22 21];[23 24];[24 23];[24 25]];
PlotGraph(V,E,'o'); % рисуем орграф
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfИсходный орграф')
```

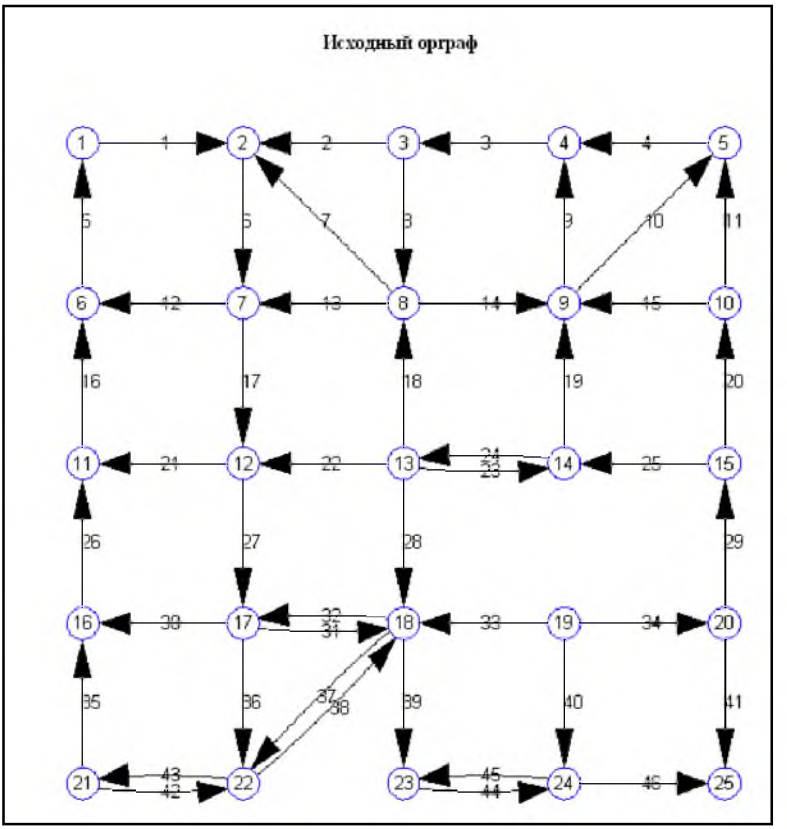


Рис. 34.3. Исходный орграф, нарисованный с помощью MATLAB

Вызовем процедуру `DecompPartOrder` и выведем результаты. Напечатаем номера вершин, попавших в каждый класс эквивалентности, и матрицу частич-

ного упорядочения. Нарисуем орграф, в котором вместо номеров вершин покажем номер класса, в который она попадает (рис. 34.4).

```
[Decomp,PartOrder]=DecompPartOrder(E); % решаем задачу
disp('Классы взаимно достижимых вершин:')
disp(' N    вершины')
for k1=1:size(Decomp,2), % печатаем номера вершин в каждом классе
  fprintf('%2.0f    ',k1)
  fprintf('%d  ',find(Decomp(:,k1)))
  fprintf('\n')
end
fprintf('Частичное упорядочение классов:\n  ')
fprintf('%3.0f',1:size(PartOrder,2))
fprintf('\n')
for k1=1:size(PartOrder,1), % матрица частичного упорядочения
  fprintf('%2.0f ',k1)
  fprintf(' %1.0f ',PartOrder(k1,:))
  fprintf('\n')
end
V1=V;
for k1=1:size(Decomp,2),
  V1(find(Decomp(:,k1)),3)=k1;
end
PlotGraph(V1,E,'o'); % рисуем орграф с номерами классов вершин
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfКлассы эквивалентности вершин')
```

```
Классы взаимно достижимых вершин:
 N    вершины
 1    19
 2    20
 3    15
 4    13  14
 5    10
 6    3  4  5  8  9
 7    1  2  6  7  11  12  16  17  18  21  22
 8    23  24
 9    25
Частичное упорядочение классов:
    1  2  3  4  5  6  7  8  9
 1  1  1  1  1  1  1  1  1  1
 2  0  1  1  1  1  1  1  1  1
 3  0  0  1  1  1  1  1  1  1
```

```
4  0  0  0  1  0  1  1  1  1
5  0  0  0  0  1  1  1  1  1
6  0  0  0  0  0  1  1  1  1
7  0  0  0  0  0  0  1  1  1
8  0  0  0  0  0  0  0  1  1
9  0  0  0  0  0  0  0  0  1
```

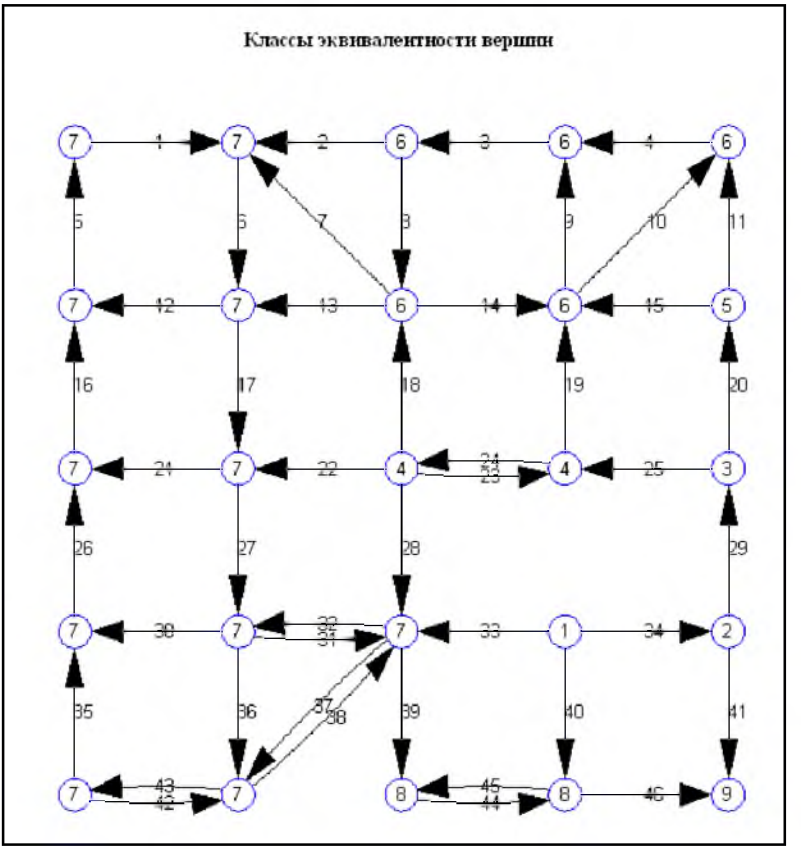


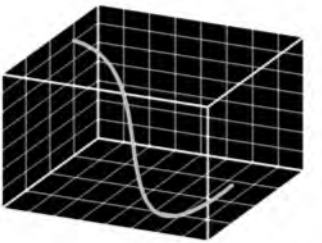
Рис. 34.4. Классы эквивалентности по достижимости и их частичное упорядочение

В данном примере множество из 25 вершин разбивается на 9 классов эквивалентности. В каждом классе любая вершина достижима из любой. Отношение достижимости между классами определяется матрицей частичного упорядочения. Здесь упорядочение оказалось почти полным: из любого класса с меньшим номером можно достичь класса с большим номером, за исключением 4-го и 5-го классов, между которыми нет упорядочения.

34.6. Вопросы для самопроверки

1. Какова полиномиальная сложность алгоритма упорядочения, основанного на половинном делении?
2. Реализуйте алгоритм стягивания орграфа на множестве вершин до орграфа на множестве классов эквивалентности.
3. Дополните этот алгоритм упрощением орграфа: оставьте только дуги минимально возможного общего веса, при которых сохраняется неизменным частичное упорядочение классов. Можно ли этот алгоритм применить для упрощения исходного орграфа?
4. Проведите социологическое исследование своей студенческой группы. Выявите лидеров и группы влияния. Добавьте веса дуг (силу влияния одного человека на другого), и в соответствии с [13] найдите неформальных лидеров.

ГЛАВА 35



Максимальный поток в сети

В этой главе мы решим еще одну интересную задачу на орграфах, имеющую широкое практическое применение.

35.1. Задача о максимальном потоке как задача линейного программирования

Определение 35.1. *Сетью* называется орграф со взвешенными дугами и с выделенными в нем двумя вершинами, одна из которых называются *источником* сети, а другая — *стоком*. □

Обычно вершину-источник обозначают буквой s, а сток — t. Как правило, из s только выходят дуги, а в t только входят, но это не обязательно. Из s в t ведут разные пути по дугам разного веса. Неотрицательные веса дуг будем обозначать c_k. В задаче о максимальном потоке вес каждой дуги рассматривается как ограничение на ее пропускную способность. Для каждой дуги e_k введем в рассмотрение ассоциированную с ней переменную, которую также обозначим e_k. Ее смысл: это реальная величина потока в данной дуге. Поток в дуге e_k неотрицательный и не превышает пропускную способность c_k в этой дуге:

$$\begin{cases} 0 \le e_k \le c_k; \\ k \in [1, m]. \end{cases} \tag{35.1}$$

В задаче о максимальном потоке, как следует из ее названия, требуется максимизировать суммарный поток, выходящий из источника s при условии, что во всех промежуточных вершинах (кроме s и t) сумма входящих и выходящих потоков должна быть равна нулю. Из общего баланса потоков следует, что поток в t будет отличаться от потока в s только знаком: ведь ни в дугах, ни в промежуточных вершинах ничего не задерживается!

Примером такой задачи является построение схемы развозки товара со склада s в магазин t по улицам с ограничением транспортного потока. По каким маршрутам и сколько машин направить, чтобы максимизировать количество доставленного за данное время товара? В этой задаче считается, что нигде на маршруте товар не задерживается: ни на улицах (дугах), ни на перекрестках (промежуточных вершинах).

Задача о максимальном потоке в сети с ограниченными пропускными способностями допускает формулировку в терминах линейного программирования. Мы уже ввели переменные e_k и ограничения на них (35.1). Введем также в рассмотрение матрицу инцидентности орграфа A. Она похожа на матрицу инцидентности графа (28.6), но в ней есть не только единицы, но и –1. А именно: будем полагать $a_{ik} = 1$, если из вершины v_i выходит дуга e_k, и $a_{ik} = -1$, если в вершину v_i входит дуга e_k. Например, матрица инцидентности орграфа с рис. 28.1, *б* имеет вид:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 & -1 \end{pmatrix}. \qquad (35.2)$$

В каждом столбце этой матрицы по одной единице и по одной –1, поэтому сумма всех строк — нулевая строка. Это соответствует общему балансу потоков.

Для матрицы инцидентности орграфа A обозначим:

- a_s — ее s-я строка (вектор-строка); в ней обязательно должны быть единицы, т. к. из источника должны выходить дуги;
- a_t — ее t-я строка (также вектор-строка); в ней обязательно должны быть –1, т. к. в сток должны входить дуги;
- A_{st} — матрица A с выброшенными из нее s-й и t-й строками.

Тогда максимизируемая целевая функция — общий поток из источника. Она может быть записана так:

$$z = \mathrm{a}_s \mathrm{e} \to \max, \qquad (35.3)$$

где e — вектор-столбец переменных e_k. А условие баланса потоков во всех промежуточных вершинах записывается в виде ограничения-равенства:

$$\mathrm{A}_{st} \mathrm{e} = 0. \qquad (35.4)$$

Вместе с ограничениями (35.1) выражения (35.3-4) определяют задачу линейного программирования, причем даже не целочисленную, а обычную, т. к. потоки в дугах могут быть дробными.

35.2. Описание процедуры *MaxFlows*

В инструментарии MATLAB [53] есть процедура `linprog` для решения задачи линейного программирования. Поэтому процедура `MaxFlows`, которую мы сейчас опишем, фактически является интерфейсом между формулировкой задачи и вызовом процедуры `linprog`. Для работы `MaxFlows` нужно задать входной параметр — список дуг орграфа и их веса. Как обычно, будем использовать массив размером $m \times 2$ или $m \times 3$, которому в процедуре соответствует идентификатор `E`. В третьем его столбце — веса дуг. Если пользователь их не задал, будем считать их единичными. Нужно также задать номера вершин источника и стока. На выходе мы должны получить вектор длины m потоков в дугах и суммарный максимальный поток из источника (вершины s). Пишем заголовок и справочную информацию.

```
function [v,mf]=MaxFlows(E,s,t)
% Функция [v,mf]=MaxFlows(E,s,t) решает задачу о максимальном потоке
% в сети с ограниченными пропускными способностями.
% Входные параметры:
%   E(m,2) или (m,3) - дуги орграфа и их веса;
%     1-й и 2-й элементы каждой строки - это номера вершин;
%     3-й элемент каждой строки - это вес дуги;
%     m - количество дуг.
%     Если задан массив E(m,2), то все веса равны 1.
%   s - источник сети (номер вершины);
%   t - сток сети (номер вершины).
% Выходные параметры:
%   v(m,1) - вектор-столбец потоков в дугах;
%   mf - общий максимальный поток в сети.
% Используется сведение к задаче линейного программирования.
```

Проверяем исходные данные. Вначале проверяем наличие данных, размерность и количество столбцов входного массива, положительность и целочисленность первых двух столбцов.

```
if nargin<3,
  error('Нет исходных данных!')
end
sz=size(E); % размеры массива E
if length(sz)~=2,
  error('Массив E должен быть двумерным!')
end
if (sz(2)<2),
  error('В массиве E должно быть 2 или 3 столбца!'),
end
```

```
if ~all(all(E(:,1:2)>0)),
  error('1-й и 2-й столбцы массива E должны быть положительными!')
end
if ~all(all((E(:,1:2)==round(E(:,1:2))))),
  error('1-й и 2-й столбцы массива E должны быть целыми!')
end
```

Из вершины с номером s обязательно должны выходить дуги, а в вершину с номером t — входить. Проверяем это.

```
if ~ismember(s,E(:,1)),
  error(['Из вершины номер s=' num2str(s) ' не выходит ни одной дуги!'])
end
if ~ismember(t,E(:,2)),
  error(['В вершину номер t=' num2str(t) ' не входит ни одна дуга!'])
end
```

Находим размер и мощность орграфа. Если пользователь не задал веса дуг, задаем их единичными.

```
m=size(E,1); % количество дуг
n=max(max(E(:,1:2))); % количество вершин
if sz(2)==2, % веса дуг не заданы
  E(:,3)=1; % все веса =1
end
```

Строим матрицу инцидентности орграфа вида (35.2). Вначале она состоит из одних нулей. Затем записываем 1 и –1 на нужные места.

```
A=zeros(n,m); % для матрицы инцидентности
for k=1:m, % просматриваем все дуги
  A(E(k,1),k)=1; % 1, если из вершины выходит дуга
  A(E(k,2),k)=-1; % -1, если в вершину входит дуга
end
```

Для системы ограничений-равенств (35.4) удаляем из матрицы инцидентности строки с номерами s и t.

```
Aeq=A(setdiff([1:n],[s t]),:); % баланс потоков в промежуточных вершинах
```

Вектор коэффициентов в целевой функции (35.3) — это s-я строка матрицы A. Мы берем ее со знаком "минус", т. к. процедура `linprog` решает задачу минимизации.

```
f=-A(s,:)'; % коэффициенты в целевой функции
```

Настраиваем опции для процедуры `linprog`. Нам не нужно выдавать на экран сообщения, поэтому подавляем вывод.

```
options=optimset('linprog'); % опции по умолчанию
options.Display='off'; % подавляем вывод
```

И наконец, решаем задачу линейного программирования и находим общий поток из источника.

```
v=linprog(f,[],[],Aeq,zeros(size(Aeq,1),1),...
  zeros(m,1),E(:,3),[],options); % решаем задачу ЛП
mf=-f'*v; % общий поток
return
```

35.3. Пример обращения к процедуре *MaxFlows*

Продемонстрируем порядок обращения к процедуре `MaxFlows`. Используем тот же орграф, который мы неоднократно брали и раньше, в частности, в *главах 32 и 33*. Нарисуем его со взвешенными дугами и невзвешенными вершинами (рис. 35.1).

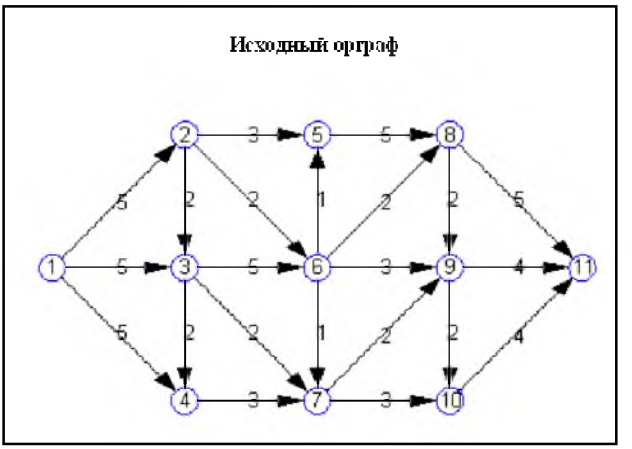


Рис. 35.1. Исходный орграф со взвешенными дугами

```
clear all
V=[[0 0];[1 1];[1 0];[1 -1];...
   [2 1];[2 0];[2 -1];[3 1];...
   [3 0];[3 -1];[4 0]]; % координаты вершин
E=[[1 2 5];[1 3 5];[1 4 5];[2 3 2];[3 4 2];[2 5 3];...
   [2 6 2];[3 6 5];[3 7 2];[4 7 3];[6 5 1];[6 7 1];...
   [5 8 5];[6 8 2];[6 9 3];[7 9 2];[7 10 3];[8 9 2];...
   [9 10 2];[8 11 5];[9 11 4];[10 11 4]]; % ребра и их веса
```

```
PlotGraph(V,E,'o'); % рисуем орграф
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfИсходный орграф')
```

Пусть источник s — это вершина v_1, а сток t — v_{11}. Зададим их. Решим задачу о максимальном потоке и напечатаем решение: потоки в дугах и общий максимальный поток из источника. На рисунке (рис. 35.2) изобразим наш орграф, у которого вместо весов дуг покажем потоки в них.

```
s=1; % источник сети
t=11; % сток сети
fprintf('Источник сети s=%d\nСток сети t=%d\n',s,t)
[v,mf]=MaxFlows(E,s,t); % решаем задачу
disp('Решение задачи о максимальном потоке в сети')
disp('  N дуги       поток')
fprintf('   %2.0f      %12.8f\n',[[1:length(v)];v'])
fprintf('Максимальный поток =%12.8f\n',mf)
PlotGraph(V(:,1:2),[E(:,1:2),v],'o'); % рисуем, пишем потоки в дугах
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfПотоки в дугах')
```

```
Источник сети s=1
Сток сети t=11
Решение задачи о максимальном потоке в сети
  N дуги       поток
    1        5.00000000
    2        5.00000000
    3        3.00000000
    4        0.63432629
    5        0.00000000
    6        2.79872259
    7        1.56695111
    8        4.12415344
    9        1.51017285
   10        3.00000000
   11        0.93404510
   12        0.17003857
   13        3.73276770
   14        1.85198251
   15        2.73503838
   16        1.86897174
   17        2.81123968
   18        0.58475020
```

```
    19         1.18876032
    20         5.00000000
    21         4.00000000
    22         4.00000000
Максимальный поток = 13.00000000
```

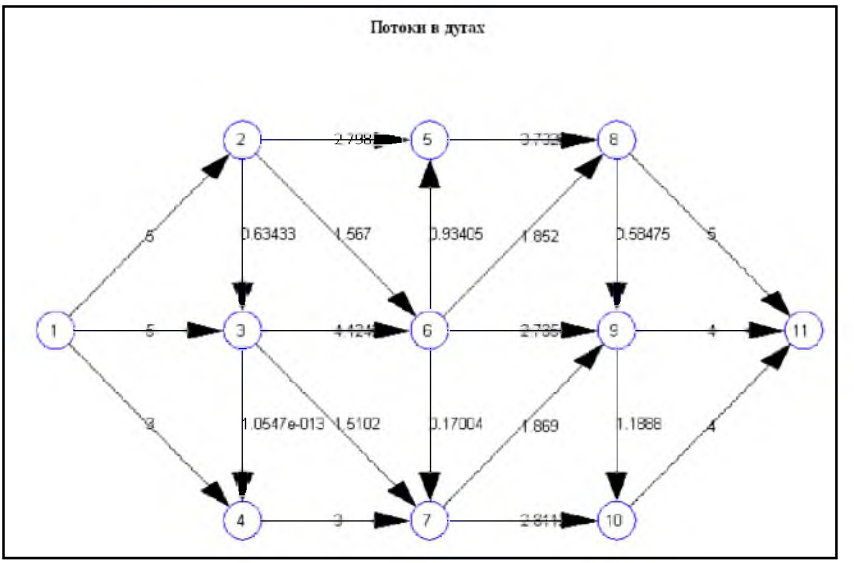
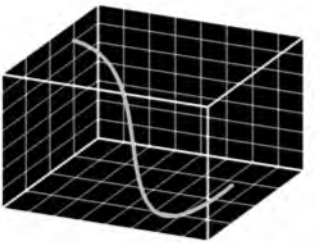


Рис. 35.2. Потоки в дугах, найденные с помощью MATLAB

35.4. Вопросы для самопроверки

1. Какие еще вы знаете методы решения задачи о максимальном потоке? Реализуйте их и сравните время решения и объем кода.
2. Решите задачу о минимальном разрезе. Используйте ее двойственность к задаче о максимальном потоке. Дополните процедуру `MaxFlows` так, чтобы она решала и эту задачу. Подсказка: множители Лагранжа одной из пары двойственных задач являются переменными другой задачи.

ГЛАВА 36



Задача коммивояжера

Мы завершаем книгу одной классической задачей комбинаторной оптимизации — несимметричной задачей коммивояжера. В общем случае это NP-полная задача, хотя для некоторых частных случаев доказано существование полиномиальных алгоритмов. Поскольку третья часть нашей книги посвящена теории графов, сформулируем задачу коммивояжера как задачу на орграфе и запишем ее в терминах ЦЛП.

36.1. Задача коммивояжера — задача ЦЛП

В задаче коммивояжера заданы n городов и матрица C расстояний между ними, в общем случае несимметричная. Размер этой матрицы $n \times n$, ее диагональные элементы могут быть произвольными, т. к. нигде в решении они не используются. Остальные элементы матрицы C по смыслу задачи неотрицательные. Требуется построить такой маршрут обхода всех n городов (по одному разу каждого), при котором его общая длина будет минимальной.

С точки зрения комбинаторной оптимизации имеем задачу оптимизации на перестановках P_n, т. к. любой замкнутый маршрут обхода всех n городов по одному разу взаимно однозначно характеризуется перестановкой n чисел — номеров городов.

Сформулируем задачу коммивояжера как задачу на орграфе. Рассмотрим ориентированную клику без петель K_n со взвешенными дугами. Вершины — это города, дуги — пути, вес c_{ik} каждой дуги e_{ik} — это расстояние от города v_i до города v_k. В общем случае $c_{ik} \neq c_{ki}$, т. к. мы рассматриваем несимметричную задачу. Требуется построить простой цикл из n вершин (и, соответственно, n дуг) минимального общего веса.

Как и многие другие задачи теории графов, эта задача допускает формулировку в терминах ЦЛП. Введем в рассмотрение целочисленные перемен-

ные e_{ik}, ассоциированные с дугами. Всего таких переменных будет столько, сколько и дуг: $m = n(n-1)$. Каждая из них может принимать только одно из двух возможных значений: 1 или 0 в зависимости от того, входит или нет дуга e_{ik} в искомый цикл:

$$\begin{cases} e_{ik} = 0 \vee 1; \\ i,\ k = \overline{1,n};\ i \neq k. \end{cases} \tag{36.1}$$

Целевая функция в данной задаче — это общий вес дуг, входящих в цикл:

$$z = \sum_{\substack{i,k=1 \\ i \neq k}}^{n} c_{ik} e_{ik} \to \min. \tag{36.2}$$

На переменные e_{ik}, помимо (36.1), наложены еще и такие ограничения. Из каждой вершины должна выходить ровно одна дуга:

$$\begin{cases} \sum\limits_{\substack{k=1 \\ k \neq i}}^{n} e_{ik} = 1; \\ i = \overline{1,n}. \end{cases} \tag{36.3}$$

Далее, в каждую вершину также должна входить одна дуга:

$$\begin{cases} \sum\limits_{\substack{i=1 \\ i \neq k}}^{n} e_{ik} = 1; \\ k = \overline{1,n}. \end{cases} \tag{36.4}$$

И, наконец, нужно еще, чтобы цикл был полным, т. е. состоял из n вершин и n дуг. Ведь ограничениям (36.3—36.4) удовлетворяет, например, система из нескольких меньших циклов, которые в сумме охватывают все n вершин. Чтобы устранить такие маленькие циклы (назовем их подциклами), введем в рассмотрение неограниченные действительные переменные v_i, ассоциированные с вершинами. В [33] приведена система ограничений, устраняющая подциклы. Эти ограничения похожи на (32.9) и имеют вид:

$$\begin{cases} v_i - v_k + (n-1)e_{ik} \leq n-2; \\ 2 \leq i \neq k \leq n. \end{cases} \tag{36.5}$$

Можно доказать, что любой цикл из n городов удовлетворяет условиям (36.5) при некоторых значениях v_i, и в то же время цикл из менее чем n городов противоречит (36.3—36.5).

■ **ТЕОРЕМА 36.1.** Любой простой цикл, не проходящий через v_1, противоречит системе (36.5) при любых v_i.

Доказательство. Предположим противное. Пусть есть последовательность h городов ($h<n$), не содержащая v_1. Их номера: $j_1, j_2, ..., j_h$. Докажем, что в этом случае система (36.5) всегда будет несовместной. В построенном подцикле есть дуги: $e_{j_1 j_2}$, $e_{j_2 j_3}$, ..., $e_{j_{h-1} j_h}$, $e_{j_h j_1}$. Значения этих переменных равны 1, а остальные $e_{ik}=0$. Рассмотрим систему (36.5) для переменных, входящих в построенный цикл:

$$\begin{cases} v_{j_r} - v_{j_{r+1}} + n - 1 \le n - 2; \; r = [1, h-1]; \\ v_{j_h} - v_{j_1} + n - 1 \le n - 2. \end{cases} \tag{36.6}$$

Сложим уравнения. Переменные v_i взаимно уничтожатся, и останется:

$$h(n-1) \le h(n-2), \tag{36.7}$$

что, очевидно, не имеет места. □

Из этой теоремы следует, что любой цикл, удовлетворяющий ограничениям (36.5), обязательно должен проходить через v_1. В совокупности с (36.3—36.4) это означает, что только полные циклы из n городов удовлетворяют (36.3—36.5). Но верно ли обратное? Любые ли циклы из n городов, удовлетворяющие (36.3—36.4), будут удовлетворять и (36.5)? Оказывается, да.

■ **ТЕОРЕМА 36.2.** Любой полный цикл из n городов, удовлетворяющий ограничениям (36.3—36.4), при некоторых v_i будет удовлетворять (36.5).

Доказательство. Рассмотрим любой полный цикл обхода всех n городов по одному разу: $j_1, j_2, ..., j_n$. При этом нумерацию вершин начнем с v_1: $j_1=1$. Переменные $e_{1 j_2}$, $e_{j_2 j_3}$, ..., $e_{j_{n-1} j_n}$, $e_{j_n 1}$ равны 1, а остальные — 0. Дадим переменным v_i значения от 1 до n в порядке обхода городов: $v_1=1$; $v_{j_2}=2$; ..., $v_{j_n}=n$. Проверим выполнение (36.5) для этих значений v_i и e_{ik}. Если переменная $e_{ik}=0$ (дуга e_{ik} не встречается в цикле), то должно выполняться:

$$\begin{cases} v_i - v_k \le n - 2; \\ 2 \le i \ne k \le n. \end{cases} \tag{36.8}$$

Это всегда выполняется при наших значениях v_i, т. к. $v_1=1$ здесь не участвует. Проверим теперь выполнение (36.5) на дугах, входящих в цикл:

$$\begin{cases} v_{j_r} - v_{j_{r+1}} + n - 1 \le n - 2; \\ r = [2, n-1]. \end{cases} \tag{36.9}$$

Подставляем наши значения v_i:

$$\begin{cases} r-(r+1)+n-1 \le n-2; \\ r=[2,\, n-1]; \end{cases} \tag{36.10}$$

что также всегда выполняется. □

Таким образом, имеем смешанную задачу ЦЛП. Смешанную — потому что часть переменных — целочисленные, а часть — непрерывные. Всего у нас $n(n-1)$ целочисленных переменных e_{ik} с ограничениями (36.1) и n непрерывных неограниченных переменных v_i. На них наложены ограничения (36.3—36.5). Целевая функция — (36.2).

Как и при правильной раскраске графа в *главе 32*, наличие ограничений (36.5) значительно усложняет процесс решения и требует применения мощной, но не очень быстро работающей процедуры `miqp` из [59]. Поэтому с помощью данного метода можно решать только задачи небольшого размера.

36.2. Описание процедуры *TravSale*

Для решения несимметричной задачи коммивояжера нужно задать квадратную матрицу расстояний C. На выходе получаем порядок обхода городов и общую длину маршрута. Вот как выглядит заголовок процедуры и справочная информация к ней.

```
function [pTS,fmin]=TravSale(C)
% Функция pTS=TravSale(C) решает несимметричную задачу коммивояжера.
% Входной параметр:
%   C(n,n) - матрица расстояний между городами, возможно, несимметричная.
%     n - количество городов.
% Выходные параметры:
%   pTS(n) - порядок обхода городов;
%   fmin - длина маршрута.
% Используется сведение к смешанной задаче ЦЛП.
% Смотри также: Miller C.E., Tucker A. W., Zemlin R. A.
% Integer Programming Formulation of Traveling Salesman Problems.
% J.ACM, 1960, Vol.7, p. 326-329.
% Необходимые другие функции: MIQP.M.
% Эта функция может быть свободно скопирована с сайта:
% http://control.ee.ethz.ch/~hybrid/miqp/
```

Простейшая проверка исходных данных заключается в том, что на входе должна быть квадратная матрица размером не менее 3×3.

```
if nargin<1,
  error('Нет исходных данных!')
end
s=size(C); % размер массива C
if length(s)~=2,
  error('Массив C должен быть двумерным!')
end
if s(1)~=s(2),
  error('Матрица C должна быть квадратной!')
end
if s(1)<3,
  error('Должно быть не менее 3 городов!')
end
```

Находим размеры задачи: количество городов (вершин) и дуг.

```
n=s(1); % количество городов
m=n*(n-1); % количество дуг
```

Начинаем формировать параметры обращения к процедуре `miqp`. Переменные будем нумеровать в таком порядке: e_{12}, e_{13}, ..., e_{1n}, e_{21}, ..., $e_{n(n-1)}$, v_2, v_3, ..., v_n. Общее количество неизвестных здесь $m+n-1=n^2-1$. Именно столько будет столбцов в матрицах ограничений. Вначале заполняем матрицу коэффициентов в ограничениях-равенствах (27.3—27.4). Сумма уравнений (27.3) совпадает с суммой (27.4), поэтому одно из уравнений в (27.3—27.4) лишнее, и всего будет $2n-1$ уравнений.

```
Aeq=[]; % для матрицы ограничений-равенств
for k1=1:n,
  z1=zeros(n);
  z1(k1,:)=1; % для (27.3)
  z2=[z1;eye(n)]; % для (27.4)
  Aeq=[Aeq z2([1:2*n-1],setdiff([1:n],k1))]; % удалили i=k
end
Aeq=[Aeq zeros(2*n-1,n-1)]; % добавили столбцы для переменных v(2:n)
```

Теперь строим матрицу коэффициентов при неизвестных в ограничениях-неравенствах (36.5).

```
A=[]; % для матрицы ограничений-неравенств
for k1=2:n,
  z1=[];
  for k2=1:n,
    z2=eye(n)*(n-1)*(k2==k1); % коэффициент при eik
    z1=[z1 z2(setdiff([2:n],k1),setdiff([1:n],k2))];
  end
```

```
  z2=-eye(n); % коэффициенты при vi и vk
  z2(:,k1)=z2(:,k1)+1;
  A=[A;[z1 z2(setdiff([2:n],k1),2:n)]];
end
```

Далее — правые части для обеих систем.

```
beq=ones(2*n-1,1); % правые части для ограничений-равенств
b=ones((n-1)*(n-2),1)*(n-2); % правые части для ограничений-неравенств
```

Чтобы построить вектор коэффициентов в целевой функции, нужно из матрицы C удалить главную диагональ, а оставшуюся часть развернуть по строкам в одномерный массив.

```
C1=C'+diag(ones(1,n)*NaN); % нечисловые значения на главной диагонали
C2=C1(:); % в одномерный массив по строкам
c=[C2(~isnan(C2));zeros(n-1,1)]; % удалили диагональ, добавили 0 для vi
```

Задаем границы по всем переменным и нулевой гессиан.

```
vlb=[zeros(m,1);-inf*ones(n-1,1)]; % нижние границы
vub=[ones(m,1);inf*ones(n-1,1)]; % верхние границы
H=zeros(n^2-1); % гессиан
```

Решаем смешанную задачу ЦЛП и начинаем формировать ответ. Округляем решения. Из линейного (одномерного) массива формируем квадратную матрицу, добавляя в нее главную диагональ.

```
[xmin,fmin]=miqp(H,c,A,b,Aeq,beq,[1:m],vlb,vub); % решаем задачу ЦЛП
eik=round(xmin(1:m)); % округляем решение - дуги найденного пути
e1=[zeros(1,n-1);reshape(eik,n,n-1)]; % дополняем 0 для главной диагонали
e2=[e1(:);0]; % еще один нуль в конце
e3=(reshape(e2,n,n))'; % матрица дуг n*n из 0 и 1
```

Теперь из полученной матрицы нужно извлечь ответ — список городов. Начинаем с города номер 1. Ищем, где в 1-й строке матрицы дуг встречается единица — это будет 2-й город. Затем ищем единицу в этой строке, и т. д., пока не вернемся в 1-й город. Тем самым мы построим список обхода. А длина пути у нас есть — это второй выходной параметр при обращении к процедуре `miqp`.

```
pTS=[1 find(e3(1,:))]; % начинаем строить список: первые 2 города
while pTS(end)>1, % пока последний в списке - не 1-й город
  pTS=[pTS find(e3(pTS(end),:))]; % добавляем в список
end
return
```

36.3. Пример обращения к процедуре *TravSale*

Обращение к процедуре `TravSale` очень простое. Вначале мы задаем матрицу расстояний между городами:

```
clear all
C=[[0 3 7 4 6 4];[4 0 3 7 8 5];[6 9 0 3 2 1];
   [8 6 3 0 9 8];[3 7 4 6 0 4];[4 5 8 7 2 0]];
disp('Расстояния между городами:')
fprintf('     %2.0f',1:size(C,2))
fprintf('\n')
for k1=1:size(C,1),
  fprintf('%2.0f',k1)
  fprintf('%7.2f',C(k1,:))
  fprintf('\n')
end
```

```
Расстояния между городами:
      1      2      3      4      5      6
 1   0.00   3.00   7.00   4.00   6.00   4.00
 2   4.00   0.00   3.00   7.00   8.00   5.00
 3   6.00   9.00   0.00   3.00   2.00   1.00
 4   8.00   6.00   3.00   0.00   9.00   8.00
 5   3.00   7.00   4.00   6.00   0.00   4.00
 6   4.00   5.00   8.00   7.00   2.00   0.00
```

Затем вызываем процедуру и печатаем результаты ее работы:

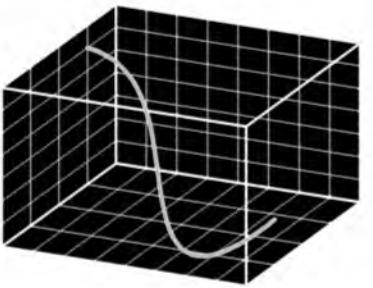
```
[pTS,fmin]=TravSale(C); % вызов процедуры
disp('Порядок обхода городов:')
fprintf('%d   ',pTS)
fprintf('\nМинимальная длина пути =%3.0f\n',fmin)
```

```
Порядок обхода городов:
1   4   2   3   6   5   1
Минимальная длина пути = 19
```

36.4. Вопросы для самопроверки

1. Какие вы знаете методы решения задачи коммивояжера?
2. Для каких частных случаев задача коммивояжера имеет полиномиальное время решения?

3. Какой алгоритм используется для решения задачи коммивояжера в справочной службе MATLAB? Можно ли его использовать для решения несимметричной задачи?
4. Какие еще проверки исходных данных можно провести в процедуре `TravSale`?
5. Будет ли процедура `TravSale` работать при отрицательных расстояниях?



ПРИЛОЖЕНИЯ

Приложение 1. Учимся работать в MATLAB

Приложение 2. Описание компакт-диска

ПРИЛОЖЕНИЕ 1

Учимся работать в MATLAB

Мы не ставим своей целью изучить MATLAB полностью. Если вы хотите более полно освоить систему инженерных и научных расчетов MATLAB, то начните, например, с [15, 16, 36]. Здесь мы разберем только те элементы программирования на MATLAB, которые понадобятся нам для решения задач в этой книге. В частности, мы рассмотрим символические вычисления, построение графиков, решение конечных и дифференциальных уравнений.

Заметим, что разные версии MATLAB имеют различный синтаксис некоторых команд, поэтому мы будем стараться подробно описывать каждую команду. Но если что-нибудь у вас не будет получаться, то справку по любой команде MATLAB можно получить, вызвав справочную службу или набрав в командном окне:

```
help [имя команды]
```

В том месте книги, где нужно будет что-то посчитать, вы встретите область ввода и область вывода. Вот как они выглядят:

```
Это - образец области ввода. Сюда вводятся команды MATLAB.
Это - образец области вывода.
Сюда MATLAB направляет результаты своей работы.
```

В электронном варианте книги эти области — "живые". Если в область ввода ввести какие-либо команды MATLAB (или изменить имеющиеся там) и нажать на кнопку "Посчитать", которая находится рядом с ними на Web-странице, то команды из области ввода будут пересланы в MATLAB, запущены на счет, а результаты возвращены в область вывода на Web-страницу.

Здесь же, в печатном варианте, эти области — лишь иллюстрация. Ими можно воспользоваться как шаблоном для составления своих программ. Если у вас нет диска с электронной версией книги, то можно поступить так. Запустите MATLAB на своем компьютере, откройте редактор-отладчик и введите

туда команды из каждой области ввода вручную. Потом посылайте их на счет.

П1.1. Символические вычисления

Вы, наверное, слышали, что современные математические пакеты позволяют проводить не только численные, но и аналитические расчеты: дифференцирование, интегрирование, вычисление пределов и суммирование рядов, умеют аналитически решать системы конечных и дифференциальных уравнений. Такой способностью обладает и MATLAB. Для этих целей в нем предусмотрены специальные переменные — символические [58]. Они являются объектами довольно сложной внутренней структуры. Если вы знакомы с объектно-ориентированным программированием, то символические переменные как раз и есть такие объекты. Они ориентированы на проведение определенных действий: в данном случае — символических вычислений.

Чтобы использовать такие переменные, вовсе не обязательно знать, как они устроены. Достаточно описать такую переменную, и тогда все другие переменные, которые ее используют, тоже будут символическими. Так можно составлять функции, зависящие от разных аргументов, и выполнять над ними различные действия.

Рассмотрим пример. Мы опишем символические переменные `x` и `y`, составим из них функцию двух переменных `z` и напечатаем ее.

Вначале мы с помощью команды `clear all` очищаем рабочую область MATLAB от предыдущих задач. Эту операцию рекомендуется проделывать в начале каждой программы (но не процедуры!). Зачем? А вот представьте: вы по ошибке используете какую-то переменную, которую не определили. Если вы предварительно очистили память, то MATLAB сразу выдаст вам сообщение об ошибке: используется неопределенная переменная. Если же память не очищена, то от предыдущих вычислений может остаться переменная с таким же именем, и она, как ни в чем не бывало, будет использоваться. В этом случае выловить ошибку будет значительно труднее. А вот в начале процедуры или функции команду очистки использовать не нужно: она сотрет все, в том числе и вызывающую программу.

Итак, мы очистили память. Теперь описываем символические переменные `x` и `y` с помощью команды `syms`. Строим функцию `z` по обычным правилам, и она автоматически станет символической.

Напечатаем нашу функцию. Для печати в MATLAB можно использовать два приема.

- Не ставить точку с запятой в конце выражения при вычислении какой-либо величины. Эта величина автоматически будет напечатана в команд-

ном окне MATLAB (в электронной версии — в области вывода). Точка с запятой подавляет вывод на экран.

- Использовать команды форматированной печати `fprintf`. Этот метод более универсальный, и мы чаще всего будем применять именно его.

Команда `fprintf`, к сожалению, не печатает непосредственно символические переменные. Для печати ее нужно представить в виде строки символов. Такое преобразование любых типов в строку в MATLAB выполняется с помощью команды `char`. Теперь полученную строку можно напечатать. Вот что у нас в итоге получилось (после символа процента в каждой строке записаны комментарии):

```
clear all % очистили память
syms x y % описали символические переменные
z=(x^2-y^2)/(x^2+y^2); % вычислили функцию
zch=char(z); % преобразовали в строку
fprintf('z=%s\n',zch); % напечатали
z=(x^2-y^2)/(x^2+y^2)
```

Теперь полученную символическую функцию можно аналитически дифференцировать и интегрировать. Производные вычисляются с помощью команды `diff`. Найдем частные производные $\partial z/\partial x$, $\partial z/\partial y$ и напечатаем их.

```
dzdy=diff(z,y); % вычисляем dz/dy
dzdx=diff(z,x); % вычисляем dz/dx
fprintf('dz/dx=%s\ndz/dy=%s\n',char(dzdx),char(dzdy)); % печатаем
dz/dx=2*x/(x^2+y^2)-2*(x^2-y^2)/(x^2+y^2)^2*x
dz/dy=-2*y/(x^2+y^2)-2*(x^2-y^2)/(x^2+y^2)^2*y
```

Рассмотрим теперь интегрирование символических выражений. Для этого используется функция `int`, которая позволяет вычислять и неопределенные, и определенные интегралы. Если пределы интегрирования не указаны, то вычисляется неопределенный интеграл, а если указаны — определенный. Найдем неопределенные интегралы $\int z(x,y)\,dx$; $\int z(x,y)\,dy$; определенный интеграл $\int_0^1 z(x,y)\,dx$; а затем повторный интеграл $\int_0^2 \left(\int_0^1 z(x,y)\,dx \right) dy$. Вычислим значение полученного символического выражения с помощью функции `eval`. Напечатаем результаты. Если строка с командой или оператором очень длинная, ее можно перенести на следующую строку с помощью многоточия.

```
udIzx=int(z,x); % неопределенный интеграл от z по x
udIzy=int(z,y); % неопределенный интеграл от z по y
dIzx=int(z,x,0,1); % определенный интеграл от z по x в [0;1]
```

```
dIz=int(dIzx,y,0,2); % еще и по y в [0;2]
fprintf('Неопределенный интеграл от z по x\nI=%s\n',char(udIzx));
fprintf('Неопределенный интеграл от z по y\nI=%s\n',char(udIzy));
fprintf('Определенный интеграл от z по x\nI=%s\n',char(dIzx));
fprintf('Проинтегрировали еще и по y\nI=%s=%8.5f\n',...
  char(dIz),eval(dIz));
Неопределенный интеграл от z по x
I=x-2*y*atan(x/y)
Неопределенный интеграл от z по y
I=-y+2*x*atan(y/x)
Определенный интеграл от z по x
I=1-2*y*atan(1/y)
Проинтегрировали еще и по y
I=-4*atan(1/2)+atan(2)=-0.74744
```

Используя этот принцип, мы можем вычислять кратные интегралы, сводя их к повторным определенным. Ведь пределы не обязательно должны быть константами: они могут быть символическими переменными.

П1.2. Построение графиков

Простейший двумерный график можно построить с помощью команды `plot`. В этой команде нужно задать массивы абсцисс и ординат одинаковой длины. График будет построен по точкам и автоматически масштабирован. На одном графике можно построить и несколько функций. Для этого нужно указать несколько массивов абсцисс и ординат. После построения графика можно оформить рисунок: написать заголовок (команда `title`), метки осей (команды `xlabel` и `ylabel`); отрегулировать масштаб (функция и команда `daspect`); установить пределы по координатным осям (команды `xlim` и `ylim`); выполнить другие действия.

Построим на одном рисунке два графика: функции $\sin x$ линией синего цвета и нашей функции $z(x,y)$ при $y=1$ линией красного цвета. Графики будем строить в интервале $[0,\pi]$.

Вначале командой `linspace` зададим линейный массив аргументов на интервале $[0,\pi]$. Первую функцию $\sin x$ мы просто вычислим. Чтобы сформировать вторую функцию, нужно в нашу функцию z подставить $y=1$. Такая подстановка осуществляется командой `subs`. Теперь в полученную функцию мы снова подставляем (опять-таки командой `subs`) вместо аргумента x наш массив. Рисуем командой `plot` два графика на одном рисунке.

Далее займемся оформлением полученного графика. Установим кириллизированный вариант шрифта Times New Roman размером 10 пт. Для этого

функцией `get` получим значение дескриптора текущего окна (функция `gcf`), и тут же командой `set` установим в этом окне новое значение параметров `FontName` и `FontSize`. Надписываем заголовок окна (команда `title`) и метки координатных осей (команды `xlabel`, `ylabel`). Выравниваем масштабы по осям координат. Функция `daspect` возвращает в переменной `da` текущие масштабы. Мы их выравниваем по минимуму, чтобы рисунок поместился в окне, и устанавливаем. И, наконец, командой `xlim` устанавливаем границы графика по оси *Ox*. По оси *Oy* границы пусть будут такими, как получаются: мы их не меняем. Результат показан на рис. П1.1.

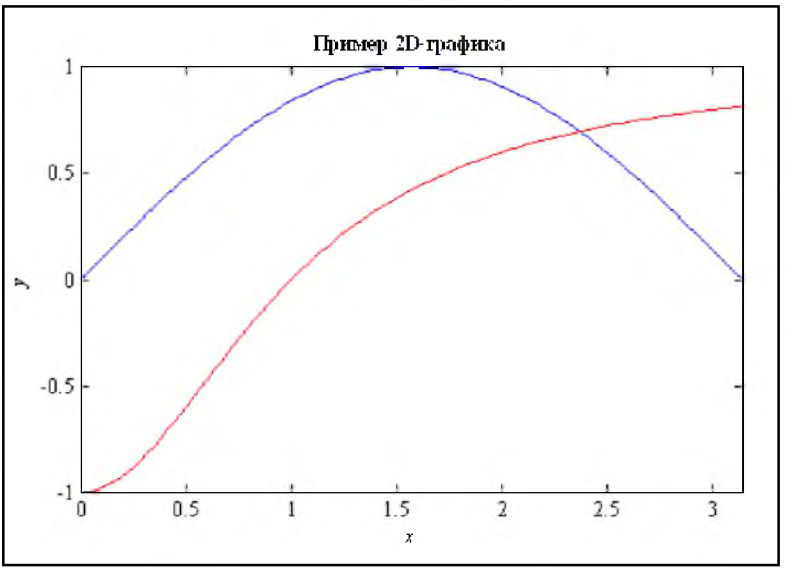


Рис. П1.1. Двумерный график

```
x1=linspace(0,pi); % задали массив x
y1=sin(x1); % 1-я функция
zy1=subs(z,y,1); % функция z при y=1
fprintf('z(x,1)=%s\n',char(zy1));
z1=subs(zy1,x,x1); % подставили x
plot(x1,y1,'b',x1,z1,'r') % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfПример 2D-графика') % заголовок
xlabel('\itx') % метка оси Ox
ylabel('\ity') % метка оси Oy
da=daspect; % получили текущий масштаб
da(1:2)=min(da(1:2)); % выравниваем масштаб
```

```
daspect(da); % установили новый масштаб
xlim([0 pi]); % пределы по оси OX
z(x,1)=(x^2-1)/(x^2+1)
```

Трехмерные графики рисуются примерно так же. Вместо команды `plot` используются ее трехмерные аналоги: команды `plot3`, `mesh` или `surf`. Они рисуют трехмерные графики в разных вариантах. Поэкспериментируйте с ними: попробуйте построить один и тот же график разными командами.

Нарисуем трехмерный график нашей функции $z(x, y)$ в области $[0.5; 5.5]\times[0.5; 5.5]$. Зададим линейный массив аргументов на этом интервале, при этом ограничимся 40 точками. Сформируем двумерную сетку с помощью команды `meshgrid`. Полученные аргументы подставим (команда `subs`) в нашу функцию — получим аппликаты графика. Рисуем график командой `surf`. Устанавливаем шрифт, надписываем заголовок и метки осей. Задаем границы графика по осям *Ox* и *Oy*. Выбираем удобную точку просмотра с помощью команды `view`. То, что у нас получилось — на рис. П1.2.

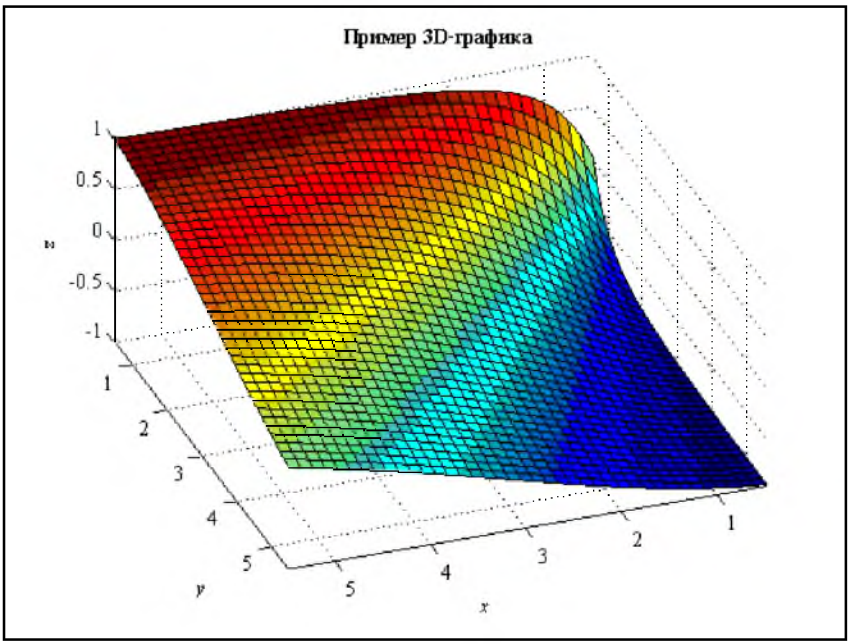


Рис. П1.2. Трехмерный график

```
x2=linspace(0.5,5.5,40); % массив из 40 точек
[X,Y]=meshgrid(x2,x2); % 2D-сетка
z2=subs(z,{x,y},{X,Y}); % вычислили z
surf(X,Y,z2); % нарисовали
```

```
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfПример 3D-графика') % заголовок
xlabel('\itx') % метка оси Ox
ylabel('\ity') % метка оси Oy
zlabel('\itz') % метка оси Oz
xlim([0.5 5.5]); % пределы по оси Ox
ylim([0.5 5.5]); % пределы по оси Oy
view(160,50); % установили точку просмотра
```

Если нужно нарисовать пространственную линию, поступаем так же. Нарисуем, например, 3 витка винтовой линии:

$$\begin{cases} x = \cos t; \\ y = \sin t; \\ z = \dfrac{t}{2\pi}. \end{cases} \qquad (П1.1)$$

Задаем линейный массив для параметра t. Вычисляем массивы абсцисс, ординат и аппликат. Рисуем график линии командой `plot3`. Устанавливаем шрифт окна. Надписываем заголовок и метки осей. С помощью команды `box on` включаем режим показа ограничивающего параллелепипеда, а командой `grid on` показываем сетку. Результат — на рис. П1.3.

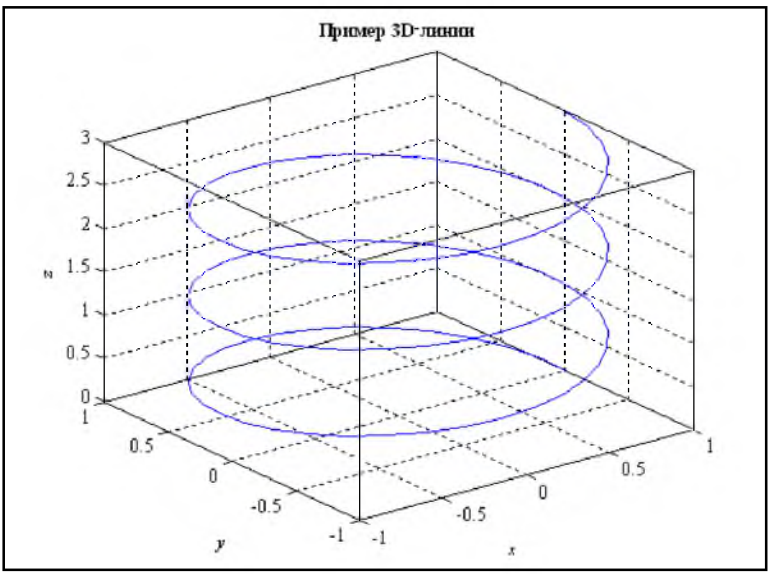


Рис. П1.3. Рисование пространственной линии

```
t=linspace(0,6*pi); % параметр для винтовой линии
x3=cos(t); % абсциссы
y3=sin(t); % ординаты
z3=t/(2*pi); % аппликаты
plot3(x3,y3,z3); % нарисовали
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfПример 3D-линии') % заголовок
xlabel('\itx') % метка оси Ox
ylabel('\ity') % метка оси Oy
zlabel('\itz') % метка оси Oz
box on % ограничивающий параллелепипед
grid on % сетка
```

П1.3. Решение конечных уравнений

Системы нелинейных уравнений в MATLAB можно решать аналитически или численно. Мы разберем оба варианта.

Для аналитического решения систем нелинейных уравнений служит команда `solve`. Посмотрите в справочной службе ее синтаксис. Сами уравнения нужно задавать в виде строк символов. Чтобы избежать различных неясностей, лучше задать и список переменных, относительно которых нужно решить систему: это также строка символов. Результат вернется в виде массива символических переменных (а не строк символов!)

Рассмотрим пример решения системы нелинейных уравнений:

$$\begin{cases} R\left(t-\sin t\right)=3; \\ R\left(1-\cos t\right)=2. \end{cases} \qquad \text{(П1.2)}$$

Вначале очищаем память. Определяем строковые переменные `eq1` и `eq2` с левыми частями нашей системы, печатаем уравнения. В команде `solve` задавать равенство нулю не обязательно: MATLAB сам его добавит. Решаем систему и помещаем результат в символический массив `Sol`. В логическом операторе `if...then...end` мы проверяем наличие решений. Функция `isempty` — логическая, она возвращает true, если массив пустой, и false, если не пустой. Если решений нет, то мы печатаем сообщение об этом с помощью команды `disp`. Команду `disp` удобно применять, когда нужно вывести только строку. Если же нужно отпечатать смешанную информацию: строка символов, значения переменных и т. д., то удобнее использовать оператор форматированной печати `fprintf`. Если решения есть (массив `Sol` не пустой), то мы с помощью оператора цикла `for...end` печатаем все решения. Массив `Sol` — это струк-

турный массив; доступ к его полям `R` и `t` мы получаем с помощью разделительной точки.

```
clear all % очистили память
eq1='R*(t-sin(t))-3'; % строка с 1-м уравнением
eq2='R*(1-cos(t))-2'; % строка со 2-м уравнением
fprintf('Система уравнений:\n%s=0\n%s=0\n',eq1,eq2);
Sol=solve(eq1,eq2,'R,t'); % решаем систему
if isempty(Sol), % нет решений
  disp('Решения не найдены');
else
  n=length(Sol); % количество решений
  fprintf('Количество найденных решений: %d\n',n);
  for k=1:n,
    fprintf('Решение %d:\nR=%12.7f\nt=%12.7f\n',...
            k,eval(Sol(k).R),eval(Sol(k).t));
  end
end
Система уравнений:
R*(t-sin(t))-3=0
R*(1-cos(t))-2=0
Количество найденных решений: 1
Решение 1:
R=   1.0013267
t=   3.0687767
```

Решим эту же систему численно. В инструментарии для решения задач оптимизации [53] есть команда `fsolve` для численного решения систем нелинейных уравнений. При вызове этой команды ей надо передать информацию о том, какую систему нужно решить. По правилам MATLAB такую информацию нужно записать в отдельный файл в виде функции, озаглавить этот файл так же, как называется функция (расширение дать m), и поместить этот файл в какой-либо каталог, доступный системе MATLAB. Эта функция (которую мы сами должны еще написать) должна по заданному вектору аргументов возвращать значения функций, которые должны обращаться в нуль. Так, для нашей системы (П1.2), если аргументы обозначить $R=x_1$; $t=x_2$; функция должна вычислять:

$$\begin{cases} y_1 = x_1\left(x_2 - \sin x_2\right) - 3; \\ y_2 = x_1\left(1 - \cos x_2\right) - 2. \end{cases} \qquad (П1.3)$$

Составим такую функцию и запишем ее в файл. Конечно же, мы это будем делать не вручную, а напишем программу на MATLAB. Заполним массив

строк s, в которых будет содержаться текст функции. В 1-й строке опишем заголовок функции. Во 2-й строке переименуем переменные. Вместо массива аргументов x нам удобнее использовать переменные R и t: у нас строки eq1 и eq2 записаны через эти переменные. Здесь же инициализируем массив функций y такого же размера, что и x. В 3-й и 4-й строках описываем вычисление функций (П1.3), используя готовые строки eq1 и eq2. Квадратные скобки для строковых переменных означают конкатенацию (соединение) строк. Строки eq1 и eq2 мы векторизуем командой vectorize: расставляем точки перед операциями умножения, деления и возведения в степень. Команда fsolve будет работать не с одним набором аргументов, а с массивом, и нам нужно, чтобы действия выполнялись поэлементно, а не как операции над матрицами. В конце каждой строки добавляем точку с запятой, чтобы подавить печать функций при каждом вызове нашей функции из команды fsolve.

Итак, мы подготовили функцию. Теперь нужно ее записать в каталог, доступный системе MATLAB, в файл с именем MyFunc.m (так мы назвали функцию, так же должен называться и файл). Выберем для записи текущий каталог: функция pwd возвращает путь к этому каталогу на вашем компьютере. Команда fullfile формирует полное имя файла из этого каталога и нашего имени. Открываем файл с таким именем (команда fopen). Записываем туда строки (команда fprintf с указанием дескриптора файла fid). После записи закрываем файл командой fclose. Обратите внимание: в текущем каталоге MATLAB нет файлов, начинающихся на My*.*, поэтому мы ничего не испортим.

```
s{1}='function y = MyFunc(x)'; % заголовок
s{2}='R=x(1); t=x(2); y=x;'; % переменные
s{3}=['y(1)=' vectorize(eq1) ';'];
s{4}=['y(2)=' vectorize(eq2) ';'];
filename=fullfile(pwd,'MyFunc.m'); % полное имя файла
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:});
fid=fopen(filename,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записали файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\MyFunc.m:
function y = MyFunc(x)
R=x(1); t=x(2); y=x;
y(1)=R.*(t-sin(t))-3;
y(2)=R.*(1-cos(t))-2;
```

Для вызова команды fsolve нужно еще задать начальное приближение. Мы его зададим в виде массива единиц нужного размера (функция ones). Решаем

систему, печатаем количество вызовов функции и полученные решения. В конце работы уберем за собой: сотрем файл MyFunc.m, записанный ранее в текущий каталог.

```
xinit=ones(2,1); % начальное приближение
[xzero,yzero,eflag,opt]=fsolve('MyFunc',xinit); % решаем
fprintf('Количество вычислений функций: %d\n',opt.iterations);
fprintf('Найденное решение:\nR=%12.7f\nt=%12.7f\n',xzero);
delete(filename) % удаляем файл
Количество вычислений функций: 7
Найденное решение:
R=   1.0013267
t=   3.0687767
```

П1.4. Решение дифференциальных уравнений

Здесь мы ограничимся рассмотрением обыкновенных дифференциальных уравнений. Решение уравнений в частных производных подробно рассмотрено в *главе 5*.

Системы дифференциальных уравнений, как и конечных, можно решать аналитически и численно. Для аналитического решения применяется команда `dsolve`. Она позволяет находить как общее, так и частное решение. Если начальные (или граничные) условия заданы, то команда находит частное решение, а если нет — то общее.

Сами дифференциальные уравнения, а также начальные или граничные условия должны быть заданы в виде строковых переменных. При этом применяются такие правила для обозначения производных: если искомая функция обозначена идентификатором `y`, то первая производная должна обозначаться `Dy`; вторая — `D2y`; третья — `D3y` и т. д. Желательно во избежание неясностей также указать, относительно какого аргумента решать уравнение. Результат возвращается в виде массива символических переменных. Длина массива — это количество найденных решений. Если искомых функций несколько, то каждый элемент массива является структурой, поля которой имеют такие же имена, как и искомые переменные.

Рассмотрим задачу Коши:

$$y'' + 2y' + y = x\sin 2x; \quad \begin{cases} y(0) = 1; \\ y'(0) = -1. \end{cases} \qquad (П1.4)$$

Очистим память. Введем строку с левой частью дифференциального уравнения (все перенесено налево). Введем начальные условия x_0, y_0 и y_0' — обычные действительные числа. Введем также x_k — конечную точку (она нам дальше будет нужна для численного решения). Выведем строки для граничных условий. Проще всего это сделать с помощью функции `sprintf`, которая очень похожа на оператор `fprintf`, но выдает результат не на печать, а в строку. Напечатаем полученные строки.

```
clear all % очистили память
eq='D2y+2*Dy+y-x*sin(2*x)'; % уравнение
x0=0; % начальная точка
xk=5; % конечная точка
y0=1; % начальное условие
y10=-1; % начальное условие
ic1=sprintf('y(%d)=%d',x0,y0); % строка с начальным условием
ic2=sprintf('Dy(%d)=%d',x0,y10); % строка с начальным условием
fprintf('Дифференциальное уравнение:\n%s=0\n',eq);
fprintf('Начальные условия:\n%s\n%s\n',ic1,ic2);
Дифференциальное уравнение:
D2y+2*Dy+y-x*sin(2*x)=0
Начальные условия:
y(0)=1
Dy(0)=-1
```

Решаем полученное уравнение относительно переменной `x` с помощью команды `dsolve`. Равенство левых частей нулю можно не задавать. Проверяем, есть ли решения. Если решения есть, то печатаем их все, а если нет — выдаем об этом сообщение.

```
Sol=dsolve(eq,ic1,ic2,'x'); % решаем
if isempty(Sol), % нет решений
  disp('Решения не найдены');
else
  n=length(Sol); % количество решений
  fprintf('Количество найденных решений: %d\n',n);
  for k=1:n,
    fprintf('Решение %d:\ny=%s\n',k,char(Sol(k)));
  end
end
Количество найденных решений: 1
Решение 1:
y=-4/25*cos(2*x)*x-4/125*cos(2*x)-3/25*x*sin(2*x)+22/125*sin(2*x)+
129/125*exp(-x)-4/25*exp(-x)*x
```

Теперь рассмотрим численное решение систем дифференциальных уравнений. В MATLAB есть много различных функций, которые называются решателями дифференциальных уравнений. Вот некоторые из них: `ode113`, `ode23`, `ode45` и др. Они применяются для разных систем (мягкие, жесткие) и используют различные схемы численного интегрирования. Но для любой из них обязательно нужно задать систему, которую надо решить, начальные условия, интервал и шаг поиска решения. Систему уравнений нужно привести к нормальному виду:

$$\begin{cases} y_1' = f_1(x, y_1, y_2, \ldots, y_n); \\ y_2' = f_2(x, y_1, y_2, \ldots, y_n); \\ \ldots \\ y_n' = f_n(x, y_1, y_2, \ldots, y_n); \end{cases} \qquad \text{(П1.5)}$$

и записать в файл функцию, которая по данным значениям аргумента x и всех функций y_1, y_2, ..., y_n вычисляет производные в этой точке. Этот файл нужно назвать так, как называется функция (расширение дать m), и поместить в какой-либо каталог, доступный MATLAB. Мы уже использовали такой прием раньше, при решении конечных уравнений.

Приведем наше уравнение 2-го порядка (П1.4) к системе двух уравнений 1-го порядка заменой:

$$\begin{cases} y_1 = y; \\ y_2 = y'. \end{cases} \qquad \text{(П1.6)}$$

Тогда наше уравнение преобразуется в систему:

$$\begin{cases} y_1' = y_2; \\ y_2' = -2y_2 + y_1 + x\sin 2x; \end{cases} \quad \begin{cases} y_1(0) = 1; \\ y_2(0) = -1. \end{cases} \qquad \text{(П1.7)}$$

Получим второе уравнение непосредственно из наших данных. Решаем уравнение (П1.4) относительно y'' (команда `solve`), и подставляем в полученное решение вместо `y` и `Dy` символические выражения `y(1)` и `y(2)` соответственно. Так как мы не описывали никаких символических переменных, мы их создаем командой `sym`. Это — обычное преобразование типов: от строки к символическому выражению. Ранее мы применяли обратную команду `char` для преобразования символического выражения в строку.

Формируем строки для функции вида (П1.7). Описываем заголовок функции. Создаем нулевой массив для результатов размером 2×1 (функция `zeros`). Заполняем строки командами вычисления правых частей системы дифференциальных уравнений.

Записываем полученные строки в файл под именем MyRightPart.m в текущий каталог MATLAB. Выводим их также на экран.

```
D2y=solve(eq,'D2y'); % решаем относительно D2y
f2=subs(D2y,{sym('y'),sym('Dy')},{sym('y(1)'),sym('y(2)')});
s{1}='function dydx=MyRightPart(x,y)';
s{2}='dydx=zeros(2,1);';
s{3}='dydx(1)=y(2);';
s{4}=['dydx(2)=' char(f2) ';'];
filename=fullfile(pwd,'MyRightPart.m'); % полное имя
disp(['Текст файла ' filename ':'])
fprintf('%s\n',s{:});
fid=fopen(filename,'w'); % открыли файл
fprintf(fid,'%s\n',s{:}); % записали файл
fclose(fid); % закрываем файл
Текст файла D:\Iglin\Matlab\MyRightPart.m:
function dydx=MyRightPart(x,y)
dydx=zeros(2,1);
dydx(1)=y(2);
dydx(2)=-2*y(2)-y(1)+x*sin(2*x);
```

Для численного решения системы дифференциальных уравнений, кроме функции вычисления правых частей, нужно задать также начальные условия и массив точек, в которых будет вычисляться решение. Разбиваем $[x_0, x_k]$ на 20 интервалов командой `linspace`. Начальные условия у нас есть — это переменные `y0` и `y10`. Решаем систему уравнений решателем `ode45`. В результате получаем: массив аргументов `xx` — это точки, в которых найдено решение, и `YY` — двумерный массив функций. Каждый столбец массива `YY` — это одна из переменных y_1, y_2, ..., y_n в точках `xx`. В нашем случае 1-й столбец `YY` — это функция $y(x)$, а 2-й — ее производная. Для построения графика нам нужен будет 1-й столбец массива `YY`.

Рисуем на одном графике (см. рис. П1.4) аналитическое решение линией синего цвета и численное красной линией. Задаем абсциссы для аналитического решения, и находим подстановкой ординаты. Рисуем графики. Надписываем заголовок и метки осей выбранным шрифтом. Устанавливаем границы по оси *Ox*. И, наконец, удаляем из текущего каталога MATLAB записанный туда файл.

```
xr=linspace(x0,xk,20); % 20 интервалов
yin=[y0;y10]; % вектор-столбец начальных условий
[xx,YY]=ode45('MyRightPart',xr,yin); % решаем
xpl=linspace(x0,xk); % абсциссы для решения
ypl=subs(Sol(1),sym('x'),xpl); % ординаты
```

```
plot(xpl,ypl,'b',xr,YY(:,1),'r');
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',10)
title('\bfРешение ODE') % заголовок
xlabel('\itx') % метка оси Ox
ylabel('\ity') % метка оси Oy
xlim([x0 xk]); % пределы по оси Ox
delete(filename);
```

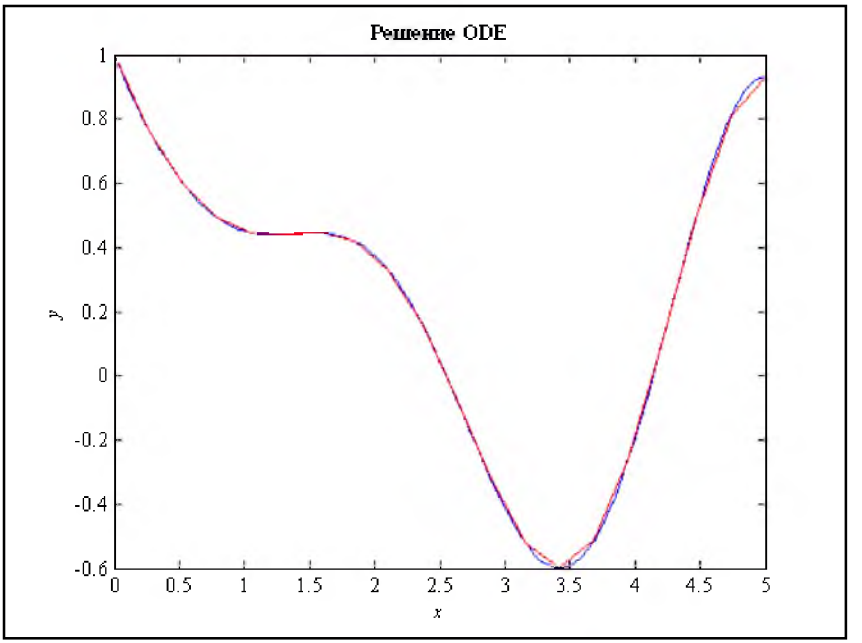


Рис. П1.4. Аналитическое и численное решение дифференциального уравнения

П1.5. Вопросы для самопроверки

1. Как описываются символические переменные?
2. Как выполняется в MATLAB дифференцирование? интегрирование? подстановка?
3. Как построить двумерный график?
4. Как построить трехмерный график поверхности? линии?
5. Какая функция решает аналитически систему конечных уравнений?

6. Какая функция применяется для численного решения системы конечных уравнений?
7. Как организуется запись в файл?
8. Как решить в аналитическом виде систему дифференциальных уравнений?
9. Как решить такую систему численно?
10. Почему на последнем графике синяя и красная линии совпадают только в отдельных точках? Свидетельствует ли это о плохой работе процедуры численного решения?

ПРИЛОЖЕНИЕ 2

Описание компакт-диска

Приложением к книге является компакт-диск, содержимое которого представляет собой интерактивный документ в формате HTML. Он доступен для просмотра через любой обозреватель Интернета, но все динамические эффекты будут доступны только при использовании Internet Explorer версии 4.0 или выше. Структура папок и описание файлов приведены в таблице.

Папки	Файлы	Описание
Корневая	index.html	Главный запускаемый файл
	ReadMe.txt	Справочная информация
\AllDocs		Вся остальная документация
\AllDocs\Part00	part00.html	Введение
	*.jpg	Рисунки для введения
\AllDocs\Part01	part01.html	Глава 1
	*.jpg, *.gif	Рисунки, формулы для главы 1
	*.emf	Графические результаты работы MATLAB для главы 1
...	...	...
\AllDocs\Part36	...	Все файлы главы 36
\AllDocs\Part37	...	Все файлы приложения 1
\AllDocs\Part38	...	Все файлы списка литературы
\AllDocs\OtherDocs	*.gif	Баннеры
	examples.zip	Примеры (упакованные текстовые файлы)
	MyStyle.css	Таблица стилей для всех Web-страниц
	*.vbs	Сценарии для всех Web-страниц

Папки для всех глав имеют одинаковую структуру. В каждой из них есть один файл part??.html, где ?? — номер главы, файлы рисунков и формул (*.jpg, *.gif) и графические результаты работы MATLAB (*.emf). Поэтому в таблице повторяющиеся элементы заменены многоточиями. В папке \AllDocs\OtherDocs находятся общие элементы для всех или некоторых страниц: баннеры, таблицы стилей, сценарии, примеры. Таким образом, на диске есть только текстовые (ReadMe.txt, *.html, MyStyle.css, *.vbs), упакованные текстовые (examples.zip) и графические (*.gif, *.jpg, *.emf) файлы. Никаких исполняемых файлов или их компонентов нет.

Документ можно загружать или непосредственно с компакт-диска, или предварительно переписав на жесткий диск компьютера. При копировании нужно сохранить структуру папок, иначе нарушится схема перекрестных ссылок.

После копирования никакой распаковки и инсталляции не требуется. При работе с документом никаких записей в реестр Windows не производится. Удаление документа не требует деинсталляции.

При совместной работе с MATLAB пользователь может изменять содержимое областей ввода и вывода. Эти изменения сохраняются в папке для временных файлов системы Windows на локальном компьютере пользователя. В ней создается папка \VarStatGraph, в которую помещаются папки \Part00, \Part01 и т. д. в зависимости от номера главы, в которой произведены изменения. В каждой из них сохраняется измененное содержимое областей ввода (текстовые файлы Inp*.txt), областей вывода (текстовые файлы Out*.txt) и созданные фигуры MATLAB (графические файлы Out*.emf). При каждой загрузке Web-страницы сценарий просматривает созданные временные файлы и, если они есть, загружает их вместо имеющихся по умолчанию. Поэтому при работе с документом пользователь видит все внесенные им изменения. Чтобы восстановить исходное содержимое какой-либо главы, нужно удалить папку с ее временными файлами. Если нужно восстановить исходное содержимое всех глав документа, можно удалить сразу всю папку \VarStatGraph.

Список литературы

1. Абрамовиц М., Стиган И. Справочник по специальным функциям. — М.: Наука, 1979.
2. Алгоритмы и программы решения задач на графах и сетях / Нечипуренко М. И., Попков В. К., Майнагашев С. М. и др. — Новосибирск: Наука, Сиб. отд-ние, 1990.
3. Ахиезер Н. И. Лекции по вариационному исчислению. — М.: Гостехиздат, 1955.
4. Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. Численные методы. — М.: Наука, 1987.
5. Берж К. Теория графов и ее применения. — М.: ИЛ, 1962.
6. Вентцель Е. С., Овчаров Л. А. Прикладные задачи теории вероятностей. — М.: Радио и связь, 1983.
7. Гельфанд И. М., Фомин С. В. Вариационное исчисление. — М.: Наука, 1969.
8. Гихман И. И., Скороход А. В., Ядренко М. И. Теория вероятностей и математическая статистика. — К.: Вища школа, 1979.
9. Гмурман В. Е. Руководство к решению задач по теории вероятностей и математической статистике. — М.: Высшая школа, 1998.
10. Гмурман В. Е. Теория вероятностей и математическая статистика. — М.: Высшая школа, 1998.
11. Гнеденко В. В. Курс теории вероятностей. — М.: Наука, 1988.
12. Говорухин В. Н., Цибулин В. Г. Компьютер в математическом исследовании: Maple, MATLAB, LaTeX. Учебный курс. — СПб.: Питер, 2001.
13. Грицак В. В., Грицак Ю. В., Иглин С. П. Стационарная алгоритмическая модель общественных отношений. — В журн.: Соционика, ментология и психология личности, № 1, 2002.

14. Гюнтер Н. М. Курс вариационного исчисления. — М.: Гостехиздат, 1941.

15. Дьяконов В. П. MATLAB 6/6.1/6.5 + Simulink 4/5. Основы применения. Полное руководство пользователя. — М.: Солон-Пресс, 2002.

16. Дьяконов В. П., Круглов В. В. Математические пакеты расширения MATLAB. Специальный справочник. — СПб.: Питер, 2001.

17. Зенкевич О. Метод конечных элементов в технике. — М.: Мир, 1975.

18. Зыков А. А. Основы теории графов. — М.: Наука, 1987.

19. Иглин С. П. Вариационное исчисление с применением MATLAB. Метод. пособие. — Харьков: ХПИ, 2001.

20. Иглин С. П. М-сайты: публикация М-книг в Интернете. — В журн.: Exponenta Pro. Математика в приложениях, № 2, 2003.

21. Иглин С. П. на сайте Exponenta. — **http://iglin.exponenta.ru**.

22. Иглин С. П. Решение некоторых задач теории графов в MATLAB. — В журн.: Exponenta Pro. Математика в приложениях, № 4, 2003.

23. Іглін С. П. Методичні вказівки до самостійної роботи студентів при виконанні типових розрахунків з курсу "Математична обробка результатів експерименту". — Харків, ХДПУ, 1999.

24. Калиткин Н. Н. Численные методы. — М.: Наука, 1978.

25. Кассандрова О. Н., Лебедев В. В. Обработка результатов наблюдений. — М.: Наука, 1970.

26. Кетков Ю. Л., Кетков А. Ю., Шульц М. М. MATLAB 6.x: программирование численных методов. — СПб.: БХВ-Петербург, 2004.

27. Корн Г., Корн Т. Справочник по математике для научных работников и инженеров. — М.: Наука, 1973.

28. Краснов Л. М., Макаренко Г. И., Киселев А. И. Вариационное исчисление. — М.: Наука, 1973.

29. Крылов В. И., Бобков В. В., Монастырный П. И. Вычислительные методы, в 2-х томах. — М.: Наука, 1977.

30. Лаврентьев М. А., Люстерник Л. А. Курс вариационного исчисления. — М.: Гостехиздат, 1936.

31. Математическая статистика: Учебник / Иванова В. М., Калинина В. Н., Нешумова Л. А. и др. — М.: Высшая школа, 1981.

32. Оре О. Теория графов. — М.: Наука, 1980.

33. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. — М.: Мир, 1985.

34. Петров Е. П., Иглин С. П. Комбинаторная оптимизация лопаточного аппарата с расстройкой // Вестник ХГПУ. Вып. 53. Динамика и прочность машин. — 1999.
35. Петрова К. Ю. Вариационное исчисление в пакете Maple. — В журн.: Exponenta Pro. Математика в приложениях, № 1, 2004.
36. Потемкин В. Г. Система MATLAB. Справочное пособие. — М.: Диалог-МИФИ, 1998.
37. Пугачев В. С. Теория вероятностей и математическая статистика. — М.: Наука, 1979.
38. Пустыльник Е. И. Статистические методы анализа и обработки наблюдений. — М.: Наука, 1968.
39. Рвачев В. Л., Курпа Л. В. R-функции в задачах теории пластин. — К.: Наукова думка, 1987.
40. Сегерлинд Л. Применение метода конечных элементов. — М.: Мир, 1979.
41. Смирнов В. И. Курс высшей математики. В 5 томах, том 4. — М.: Гостехиздат, 1957.
42. Список функций Statistics Toolbox. — **http://www.matlab.ru/statist/book2/index.asp**.
43. Татт У. Теория графов. — М.: Мир, 1988.
44. Тюрин Ю. Н. Непараметрические методы статистики. — М.: Знание, 1979.
45. Цлаф Л. Я. Вариационное исчисление и интегральные уравнения. — М.: Наука, 1970.
46. Чернова Н. И. Лекции по математической статистике. — **http://www.nsu.ru/mmf/tvims/chernova/ms/lec/ms.html**.
47. Щукин А. Н. Теория вероятностей и ее применение в инженерно-технических расчетах. — М.: Советское радио, 1974.
48. Эльсгольц Л. Э. Дифференциальные уравнения и вариационное исчисление. — М.: Наука, 1965.
49. Benker H. Statistik mit MATHCAD und MATLAB. — Springer, 2001.
50. Marques de Sá J. P. Applied Statistics Using SPSS, Statistica, and MATLAB. — Springer, 2003.
51. Mathworks File Exchange Central. — **http://www.mathworks.com/matlabcentral/fileexchange/loadCategory.do**.
52. Matlog: Logistics Engineering Matlab Toolbox. — **http://www.ie.ncsu.edu/kay/matlog/**.

53. Optimization Toolbox User's Guide. — © 1990—1997 by The MathWorks, Inc.
54. Partial Differential Equation Toolbox User's Guide. — © 1984—1997 by The MathWorks, Inc.
55. Schrafstetter E. Probabilités et Statistiques avec MATLAB et Maple. — Centre Universitaire de Formation Continue, 1999.
56. Software by Kevin Murphy. — **http://www.ai.mit.edu/~murphyk/Software/**.
57. Statistics Toolbox User's Guide. — © 1993—2001 by The MathWorks, Inc.
58. Symbolic Math Toolbox User's Guide. — © 1993—1997 by The MathWorks, Inc.
59. The Hybrid Systems Project. — **http://control.ee.ethz.ch/~hybrid/miqp/**.
60. TOMLAB Optimization Inc. — **http://tomlab.biz/**.

Предметный указатель

А

Алгоритм:
◊ Дейкстры 568
◊ жадный 530
◊ Флойда — Уоршелла 570
Аппроксимация:
◊ линейная функции 2-х переменных 432
◊ нелинейная 437
◊ полиномом для функции 2-х переменных 437
◊ степенными полиномами 424
◊ тригонометрическими полиномами 427
◊ функции нескольких переменных 431

Б

Базис цикловый 548
Бинарное отношение 577
◊ антисимметричное 578
◊ асимметричное 578
◊ достижимость 578
◊ квазиупорядоченное 578
◊ полное 578
◊ полностью упорядоченное 578
◊ предупорядоченное 578
◊ рефлексивное 578
◊ симметричное 578
◊ транзитивное 578
◊ частично упорядоченное 578
◊ эквивалентное 578

В

Вариационное исчисление:
◊ задача в параметрической форме 121
◊ изопериметрическая задача 235
◊ классические задачи 21
◊ основная задача 19, 21
◊ основная лемма 49
◊ элементарная задача 54
Вариационные принципы 27
◊ Остроградского — Гамильтона 28
◊ Ферма 25, 176, 187
Вероятность доверительная 337
Вершинное покрытие 520
◊ минимальное 520
◊ минимальное по включению 520
Выборка 314
◊ репрезентативная 317
◊ число степеней свободы 328

Г

Гиперграф 492
◊ гиперребро 493
◊ двойственный 497
◊ матрица инцидентности 498
Гистограмма 363
Граничные условия:
◊ Дирихле 109, 116
◊ естественные 126, 129
◊ матрица 115
◊ Неймана 109, 116
◊ файл 116

Граф 490
◊ вершина 490
◊ взвешенный 493
◊ двудольный 495
◊ двудольный соответствующий гиперграфу 496
◊ дерево 538
◊ инцидентность 493
◊ кратчайший путь в орграфе 567
◊ лес 538
◊ маршрут 532, 546
▫ простой 532
◊ мощность 491
◊ ориентированный (орграф) 498
◊ остовное дерево 533
◊ остовное дерево минимального веса (МОД) 534
◊ петля 492
◊ планарный 560
◊ полный (клика) 495
◊ правильная раскраска 560
◊ путь 532
◊ размер 490
◊ раскрашенный 494
◊ ребро 490
▫ кратное 492
◊ связный 532
◊ смежность 493
◊ сумма простых циклов 547
◊ цикл простой 534, 546
▫ независимый 548

Д

Декартово произведение 577
Дисперсионный анализ 399
◊ 1-факторный 402
◊ 2-факторный 405
◊ многофакторный 410
Дифференциальное уравнение:
◊ Эйлера 56
▫ первый интеграл 63, 67
▫ система 82
▫ частные случаи 61
◊ Эйлера — Остроградского 101
◊ Эйлера — Пуассона 92
◊ реализация 337
Достаточное условие экстремума 211
◊ Вейерштрасса 219
◊ Лежандра 225
Дробный факторный эксперимент 411, 437

Ж

Жадный алгоритм 530

З

Задача:
◊ о кратчайшем пути 567
◊ о максимальном потоке 591
◊ Дидоны 21
▫ взвешенная 23
▫ решение 239
◊ о брахистохроне 24
▫ решение 67, 78, 128, 140
◊ о геодезической линии 26
◊ о якорной цепи 26
▫ решение 238

И

Инструментарий
Partial Differential Equations Toolbox (PDE Toolbox) 109
Интервал доверительный 337

К

Квантиль 334
Контроль разрушающий 313
Корреляционный анализ 445, 452
Корреляция 447
◊ коэффициент 448
◊ коэффициент выборочный 452
Критерий:
◊ *F* (Фишера) 384
◊ Бартлета 390
◊ Кохрана 392
◊ согласия 362, 374
▫ Колмогорова 375
▫ Пирсона 378
▫ простейший 361

Л

Латинский квадрат 410
◊ 2-го порядка 410
◊ ортогональный 410

М

Математическая статистика 313
Матроид 532
◊ цикловый 548
Метод:
◊ выборочный 314
◊ конечных разностей 272
◊ конечных элементов 109
◊ матричной прогонки 258
◊ моментов 371
◊ наименьших квадратов 412
▫ взвешенный 415
◊ начальных параметров 255
◊ неопределенных множителей Лагранжа 227
◊ Сонина — Шмидта 424
◊ стрельбы 255
Методика текущих измерений 331
Мультиграф 492

Н

Неравенство Чебышева 320

О

Ограничения:
◊ голономные 227
◊ неголономные 234
◊ равенства 227
Операция треугольника 571
Оптимизация, принцип двойственности 238
Орграф 498
◊ достижимость 578
◊ дуга 498
◊ стрелка 498
◊ стягивание 581
Оценка 318
◊ асимметрии 329
◊ дисперсии 326
◊ интервальная 348
◊ математического ожидания 325
◊ несмещенность 322
◊ состоятельность 318
◊ точечная 317
◊ требования 318
◊ эксцесса 329
◊ эффективность 322

П

Параметр:
◊ выборочнный 317
◊ генеральный 317
Паросочетание 511
◊ максимальное 511
◊ максимальное по включению 511
Поле:
◊ собственное 211
◊ функция наклона 211
◊ центральное 213
◊ экстремалей 214
Полный факторный эксперимент 411, 433
Примитив 110
◊ круг 110
◊ многоугольник 110
◊ прямоугольник 110
◊ эллипс 111
Принцип:
◊ значимости 343
◊ максимума правдоподобия 315, 370
◊ практической достоверности 341, 343
◊ практической невозможности 341, 343
Псевдограф 492

Р

Разложенная матрица геометрии 111
Распределение:
◊ λ (Колмогорова) 376
◊ τ (максимального относительного отклонения) 360
◊ χ^2 (Пирсона) 354
◊ F (Фишера) 383
◊ g (Кохрана) 392
◊ t (Стьюдента) 350
◊ двухпараметрическое 368
◊ однопараметрическое 368, 369
Регрессионный анализ 415
Регрессия 415

С

Сеть 591
◊ источник 591
◊ сток 591
Система нормальных уравнений Гаусса 418

Событие:
◊ абсолютно достоверное 341
◊ практически достоверное 341
Совокупность:
◊ выборочная 314
◊ генеральная 314
Сравнение:
◊ двух дисперсий 382
◊ двух средних 386
◊ нескольких дисперсий 390
◊ нескольких средних 393
Статистическая гипотеза 342
◊ 0-гипотеза 345
◊ альтернативная 345
◊ двухсторонний критерий 345
◊ критическая область 343
◊ критические числа 343
◊ односторонний критерий 345
◊ основная 361
Стохастическая связь 446
◊ составляющая 446
▫ случайная 446

Т

Теорема:
◊ Гливенко — Кантелли 315
◊ Колмогорова 376

У

Уровень значимости 338
◊ жесткий 344
◊ мягкий 344
Условие трансверсальности 147, 153, 154
◊ в задаче:
▫ отражения экстремалей 176
▫ преломления экстремалей 186
▫ с односторонними вариациями 204

Ф

Функционал 19
◊ вариация 46
◊ варьируемый 45
◊ линейный 45
◊ максимум 39
◊ минимум 39
◊ непрерывный 42
◊ приращение 42
◊ строгий максимум 39
◊ строгий минимум 39
◊ экстремум k-го порядка 39
Функция 19
◊ δ-окрестность в классе C_0 31
◊ δ-окрестность в классе L_2 30
◊ базисная 416
▫ ортогональная 419
▫ ортонормированная 419
◊ близость:
▫ 0-го порядка 31
▫ 1-го порядка 32
▫ k-го порядка 34
◊ вариация 41
◊ Вейерштрасса 218
◊ внутренняя 48
◊ дифференцируемая 41
◊ зависящая от параметра 19
◊ класс 20
▫ C_0 31
▫ C_1 32
▫ C_k 33
▫ L_2 30
◊ Лагранжа 229
◊ максимум 29
▫ строгий 29
◊ минимум 29
▫ строгий 29
◊ непрерывная 41
◊ область определения 19
◊ однородная 123
◊ правдоподобия 370
◊ распределения
▫ выборочная 314

Ш

Шаблон 274

Э

Экстремаль 21, 57
Экстремум:
◊ необходимое условие 41, 48
◊ сильный 40
◊ слабый 40
▫ k-го порядка 40
Элементарный участок границы 111