Стивен Л. Брантон, Дж. Натан Куц

Анализ данных в науке и технике

Data-Driven Science and Engineering

Machine Learning, Dynamical Systems, and Control

Steven L. Brunton, J. Nathan Kutz

CAMBRIDGE
UNIVERSITY PRESS

Анализ данных в науке и технике

Машинное обучение, динамические системы и управление

Стивен Л. Брантон, Дж. Натан Куц

ДМК Пресс
издательство
Москва, 2021

УДК 001.5, 004.6
ББК 20, 32.97
Б87

Брантон С. Л., Куц Дж. Н.
Б87 Анализ данных в науке и технике / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2021. – 574 с.: ил.

ISBN 978-5-97060-931-6

Открытия, сделанные на основе анализа данных, совершили революцию в моделировании, прогнозировании поведения и управлении сложными системами. В этой книге приводятся сведения из машинного обучения, инженерной математики и математической физики с целью показать, как моделирование и управление динамическими системами сочетаются с современными методами науки о данных. Рассказывается о многих достижениях в области научных расчетов, которые позволяют применять управляемые данными методы к изучению разнообразных сложных систем, таких как турбулентность, науки о мозге, климатология, эпидемиология, финансы, робототехника и автономные системы.

Книга рассчитана на интересующихся студентов старших курсов и аспирантов первого года обучения инженерных и физических специальностей, в ней рассматривается широкий круг тем и методов на уровне от введения до недавних работ.

УДК 001.5, 004.6
ББК 20, 32.97

Copyright Original English language edition published by Cambridge University Press is part of the University of Cambridge. Russian language edition copyright © 2021 by DMK Press. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-1-108-42209-3 (англ.)
ISBN 978-5-97060-931-6 (рус.)

© Steven L. Brunton and J. Nathan Kutz, 2019
© Оформление, издание, перевод, ДМК Пресс, 2021

Содержание

От издательства 13

Об авторах 14

Предисловие 15

Общеупотребительные методы оптимизации, уравнения, символы и акронимы 20

Часть I. ПОНИЖЕНИЕ РАЗМЕРНОСТИ И ПРЕОБРАЗОВАНИЯ 31

Глава 1. Сингулярное разложение (SVD) 32

1.1. Общие сведения 33
Определение SVD 34
Вычисление SVD 35
Историческая справка 36
Использование в этой книге и предположения о подготовке читателей 37
1.2. Аппроксимация матриц 37
Усечение 38
Пример: сжатие изображения 38
1.3. Математические свойства и манипуляции 41
Интерпретация с привлечением доминирующих корреляций 41
Метод моментальных снимков 43
Геометрическая интерпретация 43
Инвариантность SVD относительно унитарных преобразований 45
Левые унитарные преобразования 46
Правые унитарные преобразования 46
1.4. Псевдообращение, метод наименьших квадратов и регрессия 47
Одномерная линейная регрессия 49
Полилинейная регрессия 51
Предостережение 53
1.5. Метод главных компонент (PCA) 53
Вычисление 54
Пример: данные с гауссовым шумом 55
Пример: данные о раке яичников 57
1.6. Пример: «собственные лица» 58
1.7. Отсечение и выравнивание 64
Оптимальный жесткий порог отсечения 64
Важность выравнивания данных 68

1.8. Рандомизированное сингулярное разложение 71
Рандомизированная линейная алгебра 71
Рандомизированный алгоритм SVD 72
Пример рандомизированного SVD 75
1.9. Тензорные разложения и *N*-мерные массивы данных 76
Рекомендуемая литература 81

Глава 2. Преобразование Фурье и вейвлет-преобразование 82
2.1. Ряд Фурье и преобразование Фурье 83
Скалярные произведения функций и векторов 83
Ряд Фурье 84
Преобразование Фурье 89
2.2. Дискретное преобразование Фурье (ДПФ) и быстрое преобразование Фурье (БПФ) 92
Дискретное преобразование Фурье 93
Быстрое преобразование Фурье 95
Пример БПФ: фильтрация шума 96
Пример БПФ: спектральные производные 98
2.3. Преобразование дифференциальных уравнений в частных производных 100
Уравнение теплопроводности 101
Одностороннее волновое уравнение 103
Уравнение Бюргерса 105
2.4. Преобразование Габора и спектрограмма 107
Дискретное преобразование Габора 108
Пример: сигнал с квадратичной частотной модуляцией 108
Пример: «Патетическая соната» Бетховена 110
Принцип неопределенности 112
2.5. Вейвлеты и многомасштабный анализ 113
Дискретное вейвлет-преобразование 115
2.6. Двумерные преобразования и обработка сигналов 116
Двумерное преобразование Фурье для изображений 116
Двумерное вейвлет-преобразование изображений 119
Рекомендуемая литература 122

Глава 3. Разреженность и сжатие измерений 123
3.1. Разреженность и сжатие 124
Пример: сжатие изображения 125
Почему сигналы допускают сжатие: просторность пространства изображений 127
3.2. Сжатое измерение 128
Заявление об отказе от ответственности 132
Другие формулировки 133
3.3. Примеры сжатых измерений 133
Норма ℓ_1 и разреженные решения недоопределенной системы 134

Восстановление звукового сигнала по разреженным измерениям 135
3.4. Геометрия сжатия 137
Свойство ограниченной изометрии (RIP) 139
Некогерентность и матрицы измерений 139
Плохие измерения 140
3.5. Разреженная регрессия 140
Отбрасывание выбросов и робастность 141
Отбор признаков и LASSO-регрессия 142
3.6. Разреженное представление 146
3.7. Робастный метод главных компонент (RPCA) 151
3.8. Разреженное размещение датчиков 153
Разреженное размещение датчиков для реконструкции 154
Разреженная классификация 158
Рекомендуемая литература 159

Часть II. МАШИННОЕ ОБУЧЕНИЕ И АНАЛИЗ ДАННЫХ 160

Глава 4. Регрессия и выбор модели 161

4.1. Классическая аппроксимация кривой 163
Методы наименьших квадратов 163
Линия наименьших квадратов 166
Линеаризация данных 167
4.2. Нелинейная регрессия и градиентный спуск 169
Градиентный спуск 170
Метод переменных направлений 175
4.3. Регрессия и уравнение $\mathbf{Ax} = \mathbf{b}$: переопределенные и недоопределенные системы 176
Переопределенные системы 176
Недоопределенные системы 180
4.4. Оптимизация как краеугольный камень регрессии 183
4.5. Парето-фронт и Lex Parsimoniae 188
Переобучение 190
4.6. Выбор модели: перекрестная проверка 191
k-групповая перекрестная проверка 195
Перекрестная проверка с контролем по *p* точкам 197
4.7. Выбор модели: информационный критерий 197
Информационные критерии: AIC и BIC 200
Вычисление AIC и BIC 201
Рекомендуемая литература 202

Глава 5. Кластеризация и классификация 203

5.1. Выделение признаков и добыча данных 204
5.2. Обучение с учителем и без учителя 210
5.3. Обучение без учителя: кластеризация методом *k* средних 214
5.4. Иерархическая кластеризация без учителя: дендрограмма 219

5.5. Смесовые модели и EM-алгоритм ... 223
5.6. Обучение с учителем и линейные дискриминанты ... 227
5.7. Метод опорных векторов (SVM) ... 233
Линейный SVM ... 233
Нелинейный SVM ... 235
Ядерные методы в сочетании с SVM ... 236
5.8. Решающие деревья и случайные леса ... 238
Случайные леса ... 243
5.9. 10 лучших алгоритмов по версии Data Mining 2008 ... 244
Алгоритм *k* средних ... 244
EM-алгоритм (смесовые модели) ... 245
Метод опорных векторов (SVM) ... 245
CART (Classification and Regression Tree – дерево классификации и регрессии) ... 245
Метод *k* ближайших соседей (kNN) ... 246
Наивная байесовская классификация ... 246
AdaBoost (ансамблевое обучение с усилением) ... 246
C4.5 (ансамблевое обучение решающих деревьев) ... 247
Алгоритм Apriori ... 247
PageRank ... 247
Рекомендуемая литература ... 248

Глава 6. Нейронные сети и глубокое обучение ... 249
6.1. Нейронные сети: однослойные сети ... 250
Однослойная сеть ... 252
6.2. Многослойные сети и функции активации ... 255
6.3. Алгоритм обратного распространения ... 260
6.4. Алгоритм стохастического градиентного спуска ... 264
6.5. Глубокие сверточные нейронные сети ... 267
Сверточные слои ... 268
Пулинговые слои ... 269
Полносвязные слои ... 269
Прореживание ... 270
6.6. Нейронные сети для динамических систем ... 272
6.7. Разнообразие нейронных сетей ... 277
Перцептрон ... 277
Сети прямого распространения (FF) ... 277
Рекуррентная нейронная сеть (RNN) ... 279
Автокодировщик (AE) ... 279
Марковская цепь (MC) ... 280
Сеть Хопфилда (HN) ... 280
Машина Больцмана (BM) ... 280
Ограниченная машина Больцмана (RBM) ... 281
Сеть глубокого доверия (DBN) ... 281
Глубокая сверточная нейронная сеть (DCNN) ... 281
Антисверточная сеть (DN) ... 281
Глубокая сверточная сеть обратной графики (DCIGN) ... 282

Порождающая состязательная сеть (GAN) 282
Машина неустойчивых состояний (LSM) 282
Машина экстремального обучения (ELM) 283
Сеть с эхо-состояниями (ESN) 283
Глубокая остаточная сеть (DRN) 283
Сеть Кохонена (KN) 284
Нейронная машина Тьюринга (NTM) 284
Рекомендуемая литература 284

Часть III. ДИНАМИЧЕСКИЕ СИСТЕМЫ И УПРАВЛЕНИЕ 285

Глава 7. Динамические системы, управляемые данными 286
7.1. Обзор, мотивация и проблемы 287
Динамические системы 287
Цели и проблемы современной теории динамических систем 291
7.2. Разложение по динамическим модам (DMD) 294
Алгоритм DMD 295
Пример и код 300
Расширения, приложения и ограничения 300
7.3. Разреженная идентификация нелинейной динамики (SINDy) 308
Нахождение дифференциальных уравнений в частных производных 314
Обобщение SINDy на рациональные нелинейности 316
Применение информационного критерия для выбора модели 319
7.4. Теория оператора Купмана 320
Математическая теория оператора Купмана 320
Разложение по модам Купмана и конечные представления 324
Примеры погружений Купмана 326
Аналитическое разложение собственных функций в ряд 329
История и недавние достижения 331
7.5. Управляемый данными анализ Купмана 332
Расширенный DMD 332
Аппроксимация собственных функций Купмана на основе данных 334
Управляемый данными анализ Купмана и запаздывающие координаты 336
Нейронные сети для погружений Купмана 340
Рекомендуемая литература 342

Глава 8. Теория линейного управления 344
Типы управления 345
8.1. Управление с замкнутым контуром обратной связи 346
Примеры преимуществ управления с обратной связью 348
8.2. Линейные стационарные системы 351
Линеаризация нелинейной динамики 351

Неуправляемая линейная система ... 352
Управляемая линейная система ... 354
Системы с дискретным временем ... 355
Пример: обратный маятник ... 356
8.3. Управляемость и наблюдаемость ... 357
Управляемость ... 357
Наблюдаемость ... 359
Критерий управляемости PBH ... 360
Теорема Кэли–Гамильтона и достижимость ... 361
Грамианы и степень управляемости и наблюдаемости ... 362
Стабилизируемость и распознаваемость ... 364
8.4. Оптимальное управление полным состоянием: линейно-квадратичный регулятор (ЛКР) ... 364
Вывод уравнения Риккати оптимального управления ... 366
8.5. Оптимальное оценивание полного состояния: фильтр Калмана ... 369
8.6. Оптимальное управление с использованием датчиков: линейно-квадратичное гауссово управление (ЛКГ) ... 372
8.7. Практический пример: обратный маятник на тележке ... 374
Управление маятником на тележке с обратной связью ... 376
Оценка полного состояния системы маятник–тележка ... 379
Управление с обратной связью системой маятник–тележка с использованием датчиков ... 382
8.8. Робастное управление и методы анализа в частотной области ... 384
Методы в частотной области ... 384
Качество управления и передаточная функция контура: чувствительность и дополнительная чувствительность ... 389
Обращение динамики ... 392
Робастное управление ... 393
Рекомендуемая литература ... 396

Глава 9. Сбалансированные модели, пригодные для управления ... 397

9.1. Упрощение модели и идентификация системы ... 397
9.2. Сбалансированное упрощение модели ... 399
Цель упрощения модели ... 399
Замена переменных в системах управления ... 401
Балансирующие преобразования ... 403
Сбалансирование усечения ... 407
Вычисление сбалансированных реализаций ... 408
Пример сбалансированного упрощения модели ... 413
9.3. Идентификация системы ... 415
Алгоритм реализации собственной системы ... 416
Идентификация наблюдателей с помощью фильтра Калмана ... 419
Комбинация ERA и OKID ... 423
Рекомендуемая литература ... 425

Глава 10. Управление на основе данных 426
10.1. Идентификация нелинейной системы для управления 427
DMD с управлением 428
Нелинейное управление с помощью оператора Купмана 430
SINDy с управлением 432
Пример управления на основе прогнозирующих моделей (MPC) 432
10.2. Управление с машинным обучением 436
Обучение с подкреплением 438
Управление с итеративным обучением 439
Генетические алгоритмы 439
Генетическое программирование 441
Пример: применение генетического алгоритма для настройки ПИД-регулятора 443
10.3. Адаптивное управление с поиском экстремума 448
Простой пример управления с поиском экстремума 452
Пример управления с поиском экстремума в сложной ситуации 455
Приложения управления с поиском экстремума 456
Рекомендуемая литература 458

Часть IV. МОДЕЛИ ПОНИЖЕННОГО ПОРЯДКА 460

Глава 11. Модели пониженного порядка (ROM) 461
11.1. POD для дифференциальных уравнений в частных производных 462
Разложение по модам Фурье 465
Специальные функции и теория Штурма–Лиувилля 466
Понижение размерности 467
11.2. Элементы оптимального базиса: собственное ортогональное разложение 468
Проекция Галеркина на POD-моды 470
Пример: гармонический осциллятор 471
11.3. POD и динамика солитонов 475
Упрощение солитона ($N = 1$) 477
Упрощение солитона ($N = 2$) 479
11.4. POD в непрерывной формулировке 480
Квадратурные правила для R: правило трапеций 482
Квадратурные правила более высокого порядка 483
POD-моды и квадратурные формулы 485
11.5. POD с симметриями: повороты и сдвиги 486
Сдвиг: распространение волн 486
Поворот: спиральные волны 488
Рекомендуемая литература 492

Глава 12. Интерполяция для ROM 494
12.1. Неполное POD 494
Разреженные измерения и реконструкция 496

Моды гармонического осциллятора 497
12.2. Ошибка и сходимость неполного POD 501
Случайная выборка и сходимость 501
Неполные измерения и качество реконструкции 503
12.3. Неполные измерения: минимизация числа обусловленности 504
Замены числа обусловленности 510
12.4. Неполные измерения: максимальная дисперсия 512
12.5. POD и дискретный эмпирический метод интерполяции (DEIM) 517
POD и DEIM 518
DEIM 519
12.6. Реализация алгоритма DEIM 521
Алгоритм QDEIM 523
12.7. Машинное обучение ROM 524
Выбор POD-моды 525
Пример: обтекание цилиндра 527
Рекомендуемая литература 529

Глоссарий 531

Список литературы 538

Предметный указатель 539

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com на странице с описанием соответствующей книги.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Springer очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об авторах

Стивен Л. Брантон – доцент факультета общего машиностроения в Вашингтонском университете. Также является внештатным сотрудником отделения прикладной математики и науки о данных Института eScience. Область его научных интересов охватывает применение науки о данных и машинного обучения к динамическим системам и управлению в области гидрогазодинамики, биомеханики, оптики, энергетических систем и производства. Является автором двух учебников, лауреатом премии армии и ВВС для молодых ученых, получил право преподавания в инженерном колледже Вашингтонского университета и удостоен премии для молодых преподавателей.

Дж. Натан Куц – профессор прикладной математики Вашингтонского университета, был деканом факультета до 2015 года. Также является внештатным профессором отделения электротехники и физики и старшим научным сотрудником отделения науки о данных в Институте eScience. Область научных интересов охватывает сложные системы и анализ данных, конкретно применение методов машинного обучения и динамических систем и управление в разнообразных приложениях. Автор двух учебников, лауреат премии Боинг за отличное преподавание прикладной математики, а также премии CAREER Национального научного фонда США.

Предисловие

Эта книга посвящена растущей области знаний на пересечении методов обработки больших данных, прикладной оптимизации и классических дисциплин инженерной математики и математической физики. Мы готовили данный материал на протяжении ряда лет, в основном для лекций, читаемых студентам старших курсов и аспирантам технических и физических факультетов.

Обычно такие студенты имеют подготовку в области линейной алгебры, дифференциальных уравнений и научных расчетов, а инженеры также знакомы с теорией управления и (или) дифференциальными уравнениями в частных производных. Однако в большинстве учебных программ научно-технических вузов методы обработки данных и (или) оптимизации освещаются слабо или не включены вовсе. С другой стороны, студенты, обучающиеся по специальностям «информатика» и «статистика», плохо знакомы с динамическими системами и теорией управления. Нашей целью было написать введение в прикладную науку о данных для обеих групп. Включенные в книгу методы отбирались по трем критериям: (1) релевантность, (2) простота и (3) общность. Мы стремились представить широкий круг тем от вводного материала до методов, реально применяемых в исследованиях.

Открытие на основе анализа данных революционизировало наши подходы к моделированию, прогнозированию поведения и управлению сложными системами. Самые насущные научно-технические задачи нашего времени не поддаются эмпирическим моделям и выводам, основанным на первопринципах. Все чаще исследователи обращаются к подходам на основе анализа данных при изучении широкого спектра сложных систем, как то: турбулентность, науки о мозге, климатология, эпидемиология, финансы, робототехника, автономные системы. Такие системы обычно являются нелинейными, динамическими, многомасштабными в пространстве и во времени, многомерными и имеют доминирующие паттерны, которые необходимо охарактеризовать и смоделировать, чтобы в конечном итоге обеспечить сбор данных, прогнозирование, оценку и управление. Благодаря современным математическим методам вкупе с невиданной ранее доступностью данных и располагаемыми вычислительными ресурсами мы теперь можем подступиться к неприступным до недавнего времени проблемам. Упомянем лишь малую толику новых методов: надежное восстановление изображения по разреженным и зашумленным измерениям случайных пикселей, управление турбулентностью с помощью машинного обучения, оптимальное размещение датчиков и приводов, идентификация допускающих интерпретацию нелинейных динамических систем на основе одних лишь данных и модели пониженного порядка, позволяющие ускорить изучение и оптимизацию систем со сложной многомасштабной физикой.

Движущим началом современной науки о данных является доступность больших и постоянно увеличивающихся объемов данных вследствие заме-

чательных инноваций в области разработки дешевых датчиков, возросших на порядки вычислительных мощностей и практически неограниченной емкости устройств хранения и скорости передачи. Такое изобилие данных открывает перед учеными и инженерами во всех областях новые возможности для изобретений на основе анализа данных; часто в этой связи говорят о четвертой парадигме научного открытия [245]. Эта четвертая парадигма представляет собой естественную кульминацию первых трех: эмпирического эксперимента, аналитического вывода и вычислительного исследования. Интеграция всех трех методик создает новаторскую платформу для новых открытий на основе данных. Этот процесс научного открытия не нов и по сути дела повторяет усилия титанов научной революции: Иоганна Кеплера (1571–1630) и сэра Исаака Ньютона (1642–1727). Оба сыграли ключевую роль в разработке теоретических принципов небесной механики на базе сочетания эмпирических подходов, основанных на анализе данных, и аналитических вычислений. Наука о данных не заменяет математическую физику и технику, но дополняет ее с учетом достижений XXI века, что больше напоминает возрождение, нежели революцию.

Наука о данных сама по себе не нова, ее предложил больше 50 лет назад Джон Тьюки, предвидевший появление науки, в центре внимания которой будет обучение на данных, или *анализ данных* [152]. С тех пор в науке о данных преобладают два разных подхода [78]. Сообщество *машинного обучения* состоит в основном из специалистов по информатике и интересуется в первую очередь разработкой быстрых, масштабируемых и качественных алгоритмов прогнозирования. Сообщество же *статистического обучения*, которое вовсе необязательно во всем противопоставлять первому, больше сосредоточено на факультетах математической статистики и занимается выводом интерпретируемых моделей. Обе методологии могут похвастаться значительными успехами и закладывают математические и вычислительные основания методов науки о данных. Целью ученых и инженеров должно стать использование этих методов для выведения из результатов наблюдений и обсчета моделей (чаще нелинейных), которые правильно улавливают динамику системы *и* количественно и качественно обобщаются на ненаблюдавшиеся области фазового, параметрического или прикладного пространства. А в этой книге нашей целью будет применение статистических методов и методов машинного обучения к решению технических задач.

Рассматриваемые вопросы

В книге обсуждается целый ряд ключевых тем. Во-первых, во многих сложных системах присутствуют доминирующие *низкоразмерные паттерны* данных, несмотря на быстрое увеличение разрешающей способности измерений и вычислений. Базовая структура открывает возможность эффективного размещения датчиков и компактного представления для моделирования и управления. Выделение паттернов тесно связано со второй темой: отысканием *преобразований координат*, позволяющих упростить систему.

Действительно, богатая история математической физики вращается вокруг преобразований координат (например, спектральные разложения, преобразование Фурье, обобщенные функции и т. д.), хотя эти методы в большинстве своем были ограничены простой идеализированной геометрией и линейной динамикой. Умение выводить преобразования *на основе данных* открывает возможность обобщить их на новые задачи с более сложной геометрией и граничными условиями. На протяжении всей книги мы будем интересоваться *динамическими системами и управлением*, т. е. применением методов, основанных на анализе данных, к моделированию и управлению систем, изменяющихся во времени. Красной нитью проходит тема *управляемой данными прикладной оптимизации*, поскольку едва ли не каждый рассматриваемый вопрос так или иначе связан с оптимизацией (например, нахождение *оптимальных* низкоразмерных паттернов, *оптимальное* расположение датчиков, *оптимизация* в машинном обучении, *оптимальное* управление и т. д.). И еще одна, даже более фундаментальная тема – большинство данных организовано в массивы для анализа, а широкое развитие численных инструментов линейной алгебры, начиная с 1960-х годов, лежит в основе математических методов матричных разложений и стратегий решения, встречающихся в этой книге.

Благодарности

Мы в долгу перед многими талантливыми студентами, сотрудниками и коллегами, которые делились ценными замечаниями и предложениями и оказывали нам поддержку. Особенно мы благодарны Джошуа Проктору (Joshua Proctor), который стоял у истоков этой книги и помогал при планировании ее структуры и организации. Мы также извлекли много полезного из бесед с Бингом Брантоном (Bing Brunton), Игорем Мезичем (Igor Mezić), Берндом Ноаком (Bernd Noack) и Сэмом Тайрой (Sam Taira). Эта книга была бы невозможна без помощи наших сотрудников и коллег, исследования которых отражены в тексте.

На протяжении работы над книгой и чтения соответствующих курсов мы получали чрезвычайно ценные отзывы и комментарии от наших замечательных студентов и постдоков: Трэвиса Эшкама (Travis Askham), Майкла Ау-Юнга (Michael Au-Yeung), Цзе Бая (Zhe Bai), Идо Брайта (Ido Bright), Кэтлин Чемпион (Kathleen Champion), Эмили Кларк (Emily Clark), Чарльза Делаханта (Charles Delahunt), Даниэля Дылевски (Daniel Dylewski), Бена Эричсона (Ben Erichson), Чарли Фислера (Charlie Fiesler), Синь Фу (Xing Fu), Чена Гонга (Chen Gong), Тарена Гормана (Taren Gorman), Джейкоба Гросека (Jacob Grosek), Сета Хирша (Seth Hirsh), Микала Джонсона (Mikala Johnson), Юрики Кайзер (Eurika Kaiser), Мейсона Кема (Mason Kamb), Джеймса Кьюнерта (James Kunert), Бетани Луш (Bethany Lusch), Педро Майа (Pedro Maia), Критики Манохара (Krithika Manohar), Найла Мэнгана (Niall Mangan), Арианы Мендибль (Ariana Mendible), Томаса Морена (Thomas Mohren), Меган Моррисон (Megan Morrison), Маркуса Куэйда (Markus Quade), Сэма Руди (Sam Rudy), Сюзанны Саргсян

(Susanna Sargsyan), Изабель Шерл (Isabel Scherl), Эли Шлизермана (Eli Shlizerman), Джорджа Степанянца (George Stepaniants), Бена Строма (Ben Strom), Чань Суна (Chang Sun), Роя Тэйлора (Roy Taylor), Мегханы Велагар (Meghana Velagar), Джейка Вехолта (Jake Weholt) и Мэтта Уильямса (Matt Williams). Наши студенты подвигли нас на написание этой книги, благодаря им мы каждый день приходим на работу с радостью и волнением.

Мы также благодарны руководителю издательской группы в Cambridge University Press Лоран Коулз (Lauren Cowles), на которую могли положиться на протяжении всего процесса работы.

Онлайновые материалы

Мы с самого начала предполагали, что к книге будут прилагаться обширные дополнительные материалы: код, данные, видео, домашние задания и рекомендуемые способы построения курса. Все эти материалы можно найти на сайте databookuw.com.

Код в сети полнее, чем в книге, в частности включен код генерации рисунков, пригодных для публикации. Визуализация данных была поставлена на первое место среди методов науки о данных в опросе «Состояние науки о данных и машинного обучения», проведенном на Kaggle 2017. Поэтому мы настоятельно рекомендуем читателям скачать код с сайта и в полной мере воспользоваться командами построения графиков и диаграмм.

Мы также записали и разместили на YouTube лекции по большинству тем, включенных в книгу. Есть дополнительные видео для студентов, желающих восполнить пробелы в подготовке по научным расчетам и основам прикладной математики. Этот текст задуман одновременно как справочное пособие и источник материалов к нескольким курсам, рассчитанным на студентов разного уровня. Большинство глав самостоятельны, на их основе можно разработать *курсы молодого бойца*, рассчитанные примерно на 10 часов каждый.

Как пользоваться этой книгой

Книга рассчитана на начинающих аспирантов и продвинутых студентов старших курсов научных и технических факультетов. Поэтому методы машинного обучения излагаются с азов, но при этом мы предполагаем, что студенты умеют моделировать физические системы с помощью дифференциальных уравнений и решать их с помощью таких программ, как **ode45**. Рассматриваются как начальные вопросы, так и актуальные исследовательские методы. Наша цель – представить цельный взгляд и математический инструментарий для решения научно-технических задач. Но книга может быть также полезна студентам, изучающим информатику и статистику, которые зачастую мало знают о динамических системах и теории управления. На основе представленного материала можно разработать несколько курсов,

программы некоторых из них имеются на сайте книги и включают домашние задания, наборы данных и код.

Прежде всего мы хотели, чтобы книга была интересной, чтобы она вдохновляла, открывала глаза и вооружала знаниями молодых ученых и инженеров. Мы пытались по возможности не слишком усложнять, не жертвуя при этом глубиной и широтой охвата, без которых не может быть никакой исследовательской работы. Многие главы можно было бы развернуть в целые книги, и такие книги есть. Однако мы также стремились к полноте в той мере, в какой этого можно ожидать от книги, посвященной столь обширной и быстро развивающей области. Мы надеемся, что книга придется вам по вкусу, что вы овладеете всеми описанными в ней методами и измените мир с помощью прикладной науки о данных!

Общеупотребительные методы оптимизации, уравнения, символы и акронимы

Наиболее распространенные стратегии оптимизации

Метод наименьших квадратов (обсуждается в главах 1 и 4) минимизирует сумму квадратов разностей (ошибок) между фактическими данными и предсказаниями модели. В случае линейного метода наименьших квадратов, когда данные аппроксимируются линейной функцией, имеется решение в замкнутой форме, которое можно найти, приравняв к нулю производную ошибки по каждому неизвестному. Этот подход широко используется в технике и прикладных науках для аппроксимации полиномиальными функциями. Применение нелинейного метода наименьших квадратов обычно требует итеративного уточнения путем аппроксимации нелинейного решения линейным на каждой итерации.

Градиентный спуск (обсуждается в главах 4 и 6) – основной метод выпуклой оптимизации в многомерных системах. Для минимизации ошибки вычисляется градиент аппроксимирующей функции. Решение обновляется итеративно путем *спуска с горы* в пространстве решений. Одномерным вариантом градиентного спуска является метод Ньютона–Рафсона. В многомерном пространстве метод часто находит только локальный минимум. Важнейшими инновациями в приложениях больших данных являются стохастический градиентный спуск и алгоритм обратного распространения, благодаря чему оптимизация сводится к самому вычислению градиента.

Чередующийся градиентный спуск (Alternating Descent Method – ADM) (обсуждается в главе 4) позволяет избежать вычисления градиента за счет того, что на каждом шаге производится оптимизация по одной неизвестной. Таким образом, все неизвестные переменные считаются постоянными, за исключением одной, по которой производится линейный поиск (невыпуклая оптимизация). Эта переменная обновляется, после чего фиксируется, и то же самое повторяется для другой переменной. На одном шаге итерации пере-

бираются все неизвестные, а сами итерации продолжаются до тех пор, пока не будет достигнута желаемая точность.

Расширенный метод Лагранжа (Augmented Lagrange Method – ALM) (обсуждается в главах 3 и 8) – класс алгоритмов для решения задач условной оптимизации. Они похожи на методы штрафования тем, что заменяют задачу оптимизации с ограничениями последовательностью задач без ограничений и прибавляют к целевой функции штрафной член, который играет роль множителя Лагранжа. Расширенный метод Лагранжа – не то же самое, что метод множителей Лагранжа.

Линейное программирование и симплекс-метод – безотказные алгоритмы выпуклой оптимизации. В линейном программировании целевая функция линейно зависит от неизвестных, а ограничениями являются линейные равенства и неравенства. Вычислив область допустимых решений – выпуклый политоп, – алгоритм линейного программирования находит в полиэдре точку, в которой функция принимает наименьшее (или наибольшее) значение, если таковая существует. Симплекс-метод – это конкретная итеративная процедура линейного программирования, которая по заданному опорному допустимому решению пытается найти другое опорное решение, для которого целевая функция принимает меньшее значение, и тем самым производит оптимизацию.

Наиболее употребительные уравнения и символы

Линейная алгебра

Линейная система уравнений

$$\mathbf{Ax} = \mathbf{b}. \tag{0.1}$$

Матрица $\mathbf{A} \in \mathbb{R}^{p \times n}$ и вектор $\mathbf{b} \in \mathbb{R}^p$ обычно известны, а вектор $\mathbf{x} \in \mathbb{R}^n$ неизвестен.

Уравнение для собственных значений

$$\mathbf{AT} = \mathbf{T\Lambda}. \tag{0.2}$$

Столбец $\boldsymbol{\xi}_k$ матрицы $\mathbf{T}$ является собственным вектором матрицы $\mathbf{A} \in \mathbb{C}^{n \times n}$, соответствующим собственному значению λ_k: $\mathbf{A}\boldsymbol{\xi}_k = \lambda_k \boldsymbol{\xi}_k$. Матрица $\boldsymbol{\Lambda}$ – диагональная матрица, содержащая эти собственные значения, в простейшем случае все n собственных значений различны.

Замена координат

$$\mathbf{x} = \boldsymbol{\Psi}\mathbf{a}. \tag{0.3}$$

Вектор $\mathbf{x} \in \mathbb{R}^n$ можно записать как $\mathbf{a} \in \mathbb{R}^n$ в системе координат, определенной столбцами матрицы $\boldsymbol{\Psi} \in \mathbb{R}^{n \times n}$.

Уравнение измерений

$$\mathbf{y} = \mathbf{C}\mathbf{x}. \tag{0.4}$$

Вектор $\mathbf{y} \in \mathbb{R}^p$ является измерением состояния $\mathbf{x} \in \mathbb{R}^n$ в результате применения матрицы измерений $\mathbf{C} \in \mathbb{R}^{p \times n}$.

Сингулярное разложение

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \approx \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*. \tag{0.5}$$

Матрицу $\mathbf{X} \in \mathbb{C}^{n \times m}$ можно разложить в произведение трех матриц $\mathbf{U} \in \mathbb{C}^{n \times n}$, $\mathbf{\Sigma} \in \mathbb{C}^{n \times m}$ и $\mathbf{V} \in \mathbb{C}^{m \times m}$. Матрицы $\mathbf{U}$ и $\mathbf{V}$ *унитарные*, т. е. $\mathbf{U}\mathbf{U}^* = \mathbf{U}^*\mathbf{U} = \mathbf{I}^{n \times n}$ и $\mathbf{V}\mathbf{V}^* = \mathbf{V}^*\mathbf{V} = \mathbf{I}^{m \times m}$, где * обозначает операцию комплексного сопряжения и транспонирования. Столбцы $\mathbf{U}$ (соответственно $\mathbf{V}$) ортогональны и называются левыми (соответственно правыми) *сингулярными векторами*. На главной диагонали диагональной матрицы $\mathbf{\Sigma}$ находятся убывающие неотрицательные элементы, называемые *сингулярными значениями*.

Часто $\mathbf{X}$ аппроксимируется матрицей низкого ранга $\tilde{\mathbf{X}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*$, где $\tilde{\mathbf{U}}$ и $\tilde{\mathbf{V}}$ содержат первые $r \ll n$ столбцов $\mathbf{U}$ и $\mathbf{V}$ соответственно, а $\tilde{\mathbf{\Sigma}}$ – левый верхний блок $\mathbf{\Sigma}$ размера $r \times r$. В контексте пространственных мод, моделей пониженного порядка и размещения датчиков матрица $\tilde{\mathbf{U}}$ часто обозначается буквой $\mathbf{\Psi}$.

Регрессия и оптимизация

Оптимизация переопределенных и недоопределенных линейных систем

$$\operatorname{argmin}_{\mathbf{x}}(\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 + \lambda g(\mathbf{x})) \text{ или} \tag{0.6a}$$

$$\operatorname{argmin}_{\mathbf{x}} g(\mathbf{x}) \text{ при условии } \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \leq \varepsilon. \tag{0.6b}$$

Здесь $g(\mathbf{x})$ – штраф регрессии (со штрафным параметром λ для переопределенных систем). Для переопределенных и недоопределенных систем линейных уравнений $\mathbf{A}\mathbf{x} = \mathbf{b}$, когда решений либо не существует, либо бесконечно много, для нахождения решения нужно задать ограничение или штраф; эта процедура называется *регуляризацией*.

Оптимизация переопределенных и недоопределенных линейных систем

$$\operatorname{argmin}_{\mathbf{x}}(f(\mathbf{A}, \mathbf{x}, \mathbf{b}) + \lambda g(\mathbf{x})) \text{ или} \tag{0.7a}$$

$$\operatorname{argmin}_{\mathbf{x}} g(\mathbf{x}) \text{ при условии } f(\mathbf{A}, \mathbf{x}, \mathbf{b}) \leq \varepsilon. \tag{0.7b}$$

Это обобщение линейной системы на нелинейную систему $f(\cdot)$ с регуляризацией $g(\cdot)$. Такие переопределенные и недоопределенные системы часто решаются методами градиентного спуска.

Композиционная оптимизация для нейронных сетей

$$\operatorname{argmin}_{\mathbf{A}_j}(f_M(\mathbf{A}_M, \cdots f_2(\mathbf{A}_2, (f_1(\mathbf{A}_1, \mathbf{x})) \cdots) + \lambda g(\mathbf{A}_j)). \tag{0.8}$$

Здесь $\mathbf{A}_k$ – матрицы весов связей между k-м и $(k+1)$-м слоями нейронной сети. Обычно это сильно недоопределенная система, которая регуляризи-

руется прибавлением $g(\mathbf{A}_j)$. Композиция и регуляризация весьма важны как для порождения выразительных представлений данных, так и для предотвращения переобучения.

Динамические системы и системы пониженного порядка

Нелинейное обыкновенное дифференциальное уравнение (динамическая система)

$$\frac{d}{dt}\mathbf{x}(\mathrm{t}) = \mathbf{f}(\mathbf{x}(t), t; \boldsymbol{\beta}). \tag{0.9}$$

Вектор $\mathbf{x}(t) \in \mathbb{R}^n$ описывает состояние системы, изменяющейся во времени t, $\boldsymbol{\beta}$ – вектор параметров, а $\mathbf{f}$ – векторное поле. В общем случае $\mathbf{f}$ – липшицева функция, что гарантирует существование и единственность решения.

Система с линейной зависимостью выхода от входа

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{0.10a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}. \tag{0.10b}$$

Состояние системы представлено вектором $\mathbf{x} \in \mathbb{R}^n$, входы (приводы) – вектором $\mathbf{u} \in \mathbb{R}^q$, а выходы (датчики) – вектором $\mathbf{y} \in \mathbb{R}^p$. Матрицы $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ определяют динамику, управляющее воздействие, стратегию работы датчиков и эффект сквозного управления соответственно.

Нелинейное отображение (динамические системы с дискретным временем)

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k). \tag{0.11}$$

Состояние системы на k-й итерации представлено вектором $\mathbf{x}_k \in \mathbb{R}^n$, а $\mathbf{F}$ – потенциально нелинейное отображение. Часто это отображение описывает продвижение итераций во времени, т. е. $\mathbf{x}_k = \mathbf{x}(k\Delta t)$; в таком случае потоковое отображение обозначается $\mathbf{F}_{\Delta t}$.

Операторное уравнение Купмана (с дискретным временем)

$$\mathcal{K}_t g = g \circ \mathbf{F}_t \Rightarrow \mathcal{K}_t \varphi = \lambda\varphi. \tag{0.12}$$

Линейный оператор Купмана $\mathcal{K}_t$ экстраполирует функции измерения состояния $g(\mathbf{x})$ с помощью потока $\mathbf{F}_t$. Собственные значения и собственные векторы $\mathcal{K}_t$ обозначаются λ и $\varphi(\mathbf{x})$ соответственно. Оператор $\mathcal{K}_t$ применяется к гильбертову пространству измерений.

Нелинейные дифференциальные уравнения в частных производных (УрЧП)

$$\mathbf{u}_t = \mathbf{N}(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, \cdots, x, t; \boldsymbol{\beta}). \tag{0.13}$$

Состояние УрЧП описывается вектором $\mathbf{u}$, $\mathbf{N}$ – нелинейный оператор эволюции, нижние индексы обозначают взятие частных производных, а x и t –

пространственная и временная переменные соответственно. УрЧП параметризуется значениями, собранными в векторе $\boldsymbol{\beta}$. Состояние УрЧП $\mathbf{u}$ может быть непрерывной функцией $u(x, t)$, а может быть дискретизировано в нескольких точках пространства, $\mathbf{u}(t) = [u(x_1, t)\ u(x_2, t) \cdots u(x_n, t)]^T \in \mathbb{R}^n$.

Разложение Галеркина
Непрерывное разложение Галеркина имеет вид:

$$u(x,t) \approx \sum_{k=1}^{r} a_k(t)\psi_k(x). \tag{0.14}$$

Функции $a_k(t)$ – коэффициенты, отражающие временную динамику, а $\psi_k(x)$ – пространственные моды. Для многомерного дискретизированного состояния разложение Галеркина принимает вид $\mathbf{u}(t) \approx \sum_{k=1}^{r} a_k(t)\boldsymbol{\psi}_k$. Пространственные моды $\boldsymbol{\psi}_k \in \mathbb{R}^n$ могут быть столбцами матрицы $\boldsymbol{\Psi} = \tilde{\mathbf{U}}$.

Список обозначений

Размерности

K количество ненулевых элементов K-разреженного вектора $\mathbf{s}$
m количество снимков данных (т. е. столбцов $\mathbf{X}$)
n размерность состояния $\mathbf{x} \in \mathbb{R}^n$
p размерность измерения, или выходной переменной $\mathbf{y} \in \mathbb{R}^p$
q размерность выходной переменной $\mathbf{u} \in \mathbb{R}^q$
r ранг усеченного сингулярного разложения или иной низкоранговой аппроксимации

Скаляры

s частота в лапласовой области
t время
δ скорость обучения в методе градиентного спуска
Δt временной шаг
x пространственная переменная
Δx пространственный шаг
σ сингулярное значение
λ собственное значение
λ параметр разреженности при разреженной оптимизации (раздел 7.3)
λ множитель Лагранжа (разделы 3.7, 8.4 и 11.4)
τ порог

Векторы

$\mathbf{a}$ вектор амплитуд мод $\mathbf{x}$ в базисе $\boldsymbol{\Psi}$, $\mathbf{a} \in \mathbb{R}^r$
$\mathbf{b}$ вектор измерений в линейной системе $\mathbf{Ax} = \mathbf{b}$
$\mathbf{b}$ вектор амплитуд мод в разложении по динамическим модам (раздел 7.2)
$\mathbf{Q}$ вектор, содержащий функцию потенциала в алгоритме PDE-FIND

$\mathbf{r}$ вектор невязок
$\mathbf{s}$ разреженный вектор $\mathbf{s} \in \mathbb{R}^n$
$\mathbf{u}$ регулируемая переменная (главы 8, 9, 10)
$\mathbf{u}$ вектор состояния УрЧП (главы 11 и 12)
$\mathbf{w}$ экзогенные входы
$\mathbf{w}_d$ возмущения системы
$\mathbf{w}_n$ шум измерений
$\mathbf{w}_r$ опорная траектория
$\mathbf{x}$ состояние системы $\mathbf{x} \in \mathbb{R}^n$
$\mathbf{x}_k$ снимок данных в момент t_k
$\mathbf{x}_j$ пример данных $j \in Z := \{1, 2, \cdots, m\}$ (главы 5 и 6)
$\tilde{\mathbf{x}}$ упрощенное состояние $\tilde{\mathbf{x}} \in \mathbb{R}^r$, т. е. $\mathbf{x} \approx \tilde{\mathbf{U}}\tilde{\mathbf{x}}$
$\hat{\mathbf{x}}$ оценка состояния системы
$\mathbf{y}$ вектор измерений $\mathbf{y} \in \mathbb{R}^p$
$\mathbf{y}_j$ метка данных $j \in Z := \{1, 2, \cdots, m\}$ (главы 5 и 6)
$\hat{\mathbf{y}}$ оценка измерения выхода
$\mathbf{z}$ преобразованное состояние $\mathbf{x} = \mathbf{T}\mathbf{z}$ (главы 8 и 9)
$\boldsymbol{\varepsilon}$ вектор ошибок
$\boldsymbol{\beta}$ бифуркационные параметры
$\boldsymbol{\xi}$ собственный вектор оператора Купмана (разделы 7.4 и 7.5)
$\boldsymbol{\xi}$ разреженный вектор коэффициентов (раздел 7.3)
$\boldsymbol{\varphi}$ мода в разложении по динамическим модам
$\boldsymbol{\psi}$ мода собственного ортогонального разложения (POD)
$\boldsymbol{\Upsilon}$ вектор измерений УрЧП в алгоритме PDE-FIND

Матрицы

$\mathbf{A}$ матрица системы уравнений, или динамики
$\tilde{\mathbf{A}}$ редуцированная динамика в r-мерном подпространстве POD
$\mathbf{A}_\mathbf{X}$ матричное представление линейной динамики с состоянием $\mathbf{x}$
$\mathbf{A}_\Upsilon$ матричное представление линейной динамики с наблюдаемыми переменными $\mathbf{y}$
$(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{B})$ матрицы системы с непрерывным пространством состояний
$(\mathbf{A}_d, \mathbf{B}_d, \mathbf{C}_d, \mathbf{B}_d)$ матрицы системы с дискретным пространством состояний
$(\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}, \hat{\mathbf{B}})$ матрицы пространства состояний системы в новых координатах $\mathbf{z} = \mathbf{T}^{-1}\mathbf{x}$
$(\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}, \tilde{\mathbf{B}})$ матрицы пространства состояний упрощенной системы ранга r
$\mathbf{B}$ матрица входных данных с приводов
$\mathbf{C}$ матрица линейных измерений состояний
$\mathcal{C}$ матрица управляемости
$\mathcal{F}$ дискретное преобразование Фурье
$\mathbf{G}$ матричное представление линейной динамики состояний и входов $[\mathbf{x}^T\mathbf{u}^T]^T$
$\mathbf{H}$ матрица Ганкеля
$\mathbf{H}'$ матрица Ганкеля с временным сдвигом
$\mathbf{I}$ единичная матрица
$\mathbf{K}$ матричная форма оператора Купмана (глава 7)
$\mathbf{K}$ коэффициент усиления системы управления с замкнутым контуром (глава 8)

$\mathbf{K}_f$ коэффициент усиления фильтра Калмана
$\mathbf{K}_r$ коэффициент усиления линейно-квадратичного регулятора (ЛКР)
$\mathbf{L}$ низкоранговая часть матрицы $\mathbf{X}$ (глава 3)
$\boldsymbol{O}$ матрица наблюдаемости
$\mathbf{P}$ унитарная матрица, применяемая к столбцам $\mathbf{X}$
$\mathbf{Q}$ весовая матрица стоимости отклонений от нулевого состояния в ЛКР (раздел 8.4)
$\mathbf{Q}$ ортогональная матрица в QR-разложении
$\mathbf{R}$ весовая матрица стоимости управляющих воздействий в ЛКР (раздел 8.4)
$\mathbf{R}$ верхнетреугольная матрица в QR-разложении
$\mathbf{S}$ разреженная часть матрицы $\mathbf{X}$ (глава 3)
$\mathbf{T}$ матрица собственных векторов (глава 8)
$\mathbf{T}$ замена координат (главы 8 и 9)
$\mathbf{U}$ левые сингулярные векторы $\mathbf{X}$, $\mathbf{U} \in \mathbb{R}^{n \times n}$
$\hat{\mathbf{U}}$ левые сингулярные векторы экономичного сингулярного разложения $\mathbf{X}$, $\hat{\mathbf{U}} \in \mathbb{R}^{n \times n}$
$\tilde{\mathbf{U}}$ левые сингулярные векторы (POD-моды) усеченного сингулярного разложения $\mathbf{X}$, $\tilde{\mathbf{U}} \in \mathbb{R}^{n \times r}$
$\mathbf{V}$ правые сингулярные векторы $\mathbf{X}$, $\mathbf{V} \in \mathbb{R}^{m \times m}$
$\tilde{\mathbf{V}}$ правые сингулярные векторы (POD-моды) усеченного сингулярного разложения $\mathbf{X}$, $\tilde{\mathbf{V}} \in \mathbb{R}^{m \times r}$
$\mathbf{\Sigma}$ матрица сингулярных значений $\mathbf{X}$, $\mathbf{\Sigma} \in \mathbb{R}^{n \times m}$
$\hat{\mathbf{\Sigma}}$ матрица сингулярных значений экономичного сингулярного разложения $\mathbf{X}$, $\hat{\mathbf{\Sigma}} \in \mathbb{R}^{m \times m}$
$\tilde{\mathbf{\Sigma}}$ матрица сингулярных значений усеченного сингулярного разложения $\mathbf{X}$, $\tilde{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$
$\mathbf{W}$ собственные векторы $\tilde{\mathbf{A}}$
$\mathbf{W}_c$ грамиан управляемости
$\mathbf{W}_o$ грамиан наблюдаемости
$\mathbf{X}$ матрица данных, $\mathbf{X} \in \mathbb{R}^{n \times m}$
$\mathbf{X}'$ матрица данных с временным сдвигом, $\mathbf{X}' \in \mathbb{R}^{n \times m}$
$\mathbf{Y}$ проекция матрицы $\mathbf{X}$ на ортогональный базис в рандомизированном сингулярном разложении (раздел 1.8)
$\mathbf{Y}$ матрица данных наблюдаемых величин, $\mathbf{Y} = \mathbf{g}(\mathbf{X})$, $\mathbf{Y} \in \mathbb{R}^{p \times m}$ (глава 7)
$\mathbf{Y}'$ матрица данных наблюдаемых величин со сдвигом, $\mathbf{Y}' = \mathbf{g}(\mathbf{X}')$, $\mathbf{Y}' \in \mathbb{R}^{p \times m}$ (глава 7)
$\mathbf{Z}$ эскиз матрицы для рандомизированного сингулярного разложения, $\mathbf{Z} \in \mathbb{R}^{n \times r}$ (раздел 1.8)
$\mathbf{\Theta}$ матрица измерений, умноженная на разреживающий базис, $\mathbf{\Theta} = \mathbf{C\Psi}$ (глава 3)
$\mathbf{\Theta}$ матрица функций-кандидатов для SINDy (раздел 7.3)
$\mathbf{\Gamma}$ матрица производных функций-кандидатов для SINDy (раздел 7.3)
$\mathbf{\Xi}$ матрица коэффициентов функций-кандидатов для SINDy (раздел 7.3)
$\mathbf{\Xi}$ матрица нелинейных снимков для DEIM (раздел 12.5)
$\mathbf{\Lambda}$ диагональная матрица собственных значений
$\mathbf{\Upsilon}$ матрица входных снимков, $\mathbf{\Upsilon} \in \mathbb{R}^{q \times m}$
$\mathbf{\Phi}$ матрица DMD-мод, $\mathbf{\Phi} \triangleq \mathbf{X}'\mathbf{V}'\mathbf{\Sigma}^{-1}\mathbf{W}$

$\boldsymbol{\Psi}$ ортонормированный базис (например, моды Фурье или POD-моды)

Тензоры

$(\mathcal{A}, \mathcal{B}, \mathcal{M})$ тензоры N-мерных массивов размера $I_1 \times I_2 \times \cdots \times I_N$

Нормы

$||\cdot||_0$ псевдонорма ℓ_0 вектора $\mathbf{x}$: количество ненулевых элементов $\mathbf{x}$

$||\cdot||_1$ норма ℓ_1 вектора $\mathbf{x}$: $||\mathbf{x}||_1 = \sum_{i=1}^{n} |x_i|$

$||\cdot||_2$ норма ℓ_2 вектора $\mathbf{x}$: $||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{n} (x_i^2)}$

$||\cdot||_2$ норма ℓ_2 матрицы $\mathbf{X}$: $||\mathbf{X}||_2 = \max_x \dfrac{||\mathbf{X}\mathbf{x}||_2}{||\mathbf{x}||_2}$

$||\cdot||_F$ норма Фробениуса матрицы $\mathbf{X}$: $||\mathbf{X}||_F = \sqrt{\sum_{i=1}^{n} \sum_{j-1}^{m} |X_{ij}|^2}$

$||\cdot||_*$ ядерная норма матрицы $\mathbf{X}$: $||\mathbf{X}||_* = \text{trace}\left(\sqrt{\mathbf{X}^*\mathbf{X}}\right) = \sum_{i=1}^{m} \sigma_i$ (для $m \leq n$)

$\langle \cdot, \cdot \rangle$ скалярное произведение. Для функций $\langle f(x), g(x) \rangle = \int_{-\infty}^{\infty} f(x) g^*(x) dx$

$\langle \cdot, \cdot \rangle$ скалярное произведение. Для векторов $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^* \mathbf{v}$

Операторы, функции и отображения

$\mathcal{F}$ преобразование Фурье

$\mathbf{F}$ отображение динамической системы с дискретным временем

$\mathbf{F}_t$ дискретное потоковое отображение динамической системы

$\mathbf{f}$ динамическая система с непрерывным временем

G преобразование Габора

$\mathbf{G}$ передаточная функция, отображающая входы на выходы (глава 8)

g скалярная функция измерения $\mathbf{x}$

$\mathbf{g}$ скалярная функция измерения $\mathbf{x}$

J функция стоимости для регулирования

ℓ функция потерь в методе опорных векторов (глава 5)

$\mathcal{K}$ оператор Купмана (с непрерывным временем)

$\mathcal{K}_t$ оператор Купмана, ассоциированный с потоковым отображением

$\mathcal{L}$ преобразование Лапласа

$\mathbf{L}$ передаточная функция контура (глава 8)

$\mathbf{L}$ линейное дифференциальное уравнение в частных производных (главы 11 и 12)

$\mathbf{N}$ нелинейное дифференциальное уравнение в частных производных

O порядок величины

$\mathbf{S}$ функция чувствительности (глава 8)

$\mathbf{T}$ дополнительная функция чувствительности (глава 8)

$\mathcal{W}$ вейвлет-преобразование

μ несогласованность между матрицей измерений $\mathbf{C}$ и базисом $\boldsymbol{\Psi}$

κ число обусловленности

φ собственная функция Купмана

∇ оператор градиента

$*$ оператор свертки

Наиболее употребительные акронимы

БПФ	быстрое преобразование Фурье
ГО	глубокое обучение
ОДУ	обыкновенное дифференциальное уравнение
СНС	сверточная нейронная сеть
УрЧП	дифференциальное уравнение в частных производных
DMD	разложение по динамическим модам (dynamic mode decomposition)
PCA	метод главных компонент (principal components analysis)
POD	собственное ортогональное разложение (proper orthogonal decomposition)
ROM	модель пониженного порядка (reduced order model)
SVD	сингулярное разложение (singular value decomposition)

Прочие акронимы

ДПФ	дискретное преобразование Фурье
ИНС	искусственная нейронная сеть
ЛДА	линейный дискриминантный анализ
НУШ	нелинейное уравнение Шрёдингера
ОПФ	оконное преобразование Фурье (short time Fourier transform)
ПИД	пропорционально-интегрально-дифференцирующий регулятор
РНС	рекуррентная нейронная сеть
СГС	стохастический градиентный спуск
ADM	метод переменных направлений (alternating directions method)
AIC	информационный критерий Акаике (Akaike information criterion)
ALM	расширенный метод множителей Лагранжа (augmented Lagrange multiplier)
ARMA	авторегрессионное скользящее среднее (autoregressive moving average)
ARMAX	авторегрессионное скользящее среднее с экзогенным входом (autoregressive moving average with exogenous input)
BIC	байесовский информационный критерий (Bayesian information criterion)
BPOD	сбалансированное собственное ортогональное разложение (balanced proper orthogonal decomposition)
CCA	канонический корреляционный анализ (canonical correlation analysis)
CFD	вычислительная гидродинамика (computational fluid dynamics)
CoSaMP	согласованное преследование со сжатой выборкой (compressive sampling matching pursuit)
CWT	непрерывное вейвлет-преобразование (continuous wavelet transform)
DCT	дискретное косинусное преобразование (discrete cosine transform)
DEIM	дискретный эмпирический метод интерполяции (discrete empirical interpolation method)
DMDc	разложение по динамическим модам с управлением (dynamic mode decomposition with control)

DMDc	разложение по динамическим модам с управлением (dynamic mode decomposition with control)
DNS	прямое численное моделирование (direct numerical simulation)
DWT	дискретное вейвлет-преобразование
ECOG	электрокортикография (electrocorticography)
eDMD	расширенное DMD (extended DMD)
EIM	эмпирический метод интерполяции (empirical interpolation method)
EM	математическое ожидание-максимизация (expectation maximization)
EOF	эмпирические ортогональные функции (empirical orthogonal functions)
ERA	алгоритм реализации собственной системы (eigensystem realization algorithm)
ESC	управление с поиском экстремума (extremum-seeking control)
GMM	модель гауссовой смеси (Gaussian mixture model)
HAVOK	ганкелево альтернативное представление оператора Купмана (Hankel alternative view of Koopman)
ICA	метод независимых компонент (independent component analysis)
JL	Джонсона–Линденштраусса (Johnson–Lindenstrauss)
KL	Кульбака–Лейблера (Kullback–Leibler)
KLT	преобразование Карунена–Лоэва (Karhunen–Loève transform)
LAD	наименьшее абсолютное отклонение (least absolute deviations)
LASSO	оператор наименьшего абсолютного сжатия и выборки (least absolute shrinkage and selection operator)
LQE	линейно-квадратичная модель оценки (linear quadratic estimator)
LQG	линейно-квадратичный гауссов регулятор (linear quadratic Gaussian controller)
LQR	линейно-квадратичный регулятор
LTI	линейная стационарная система (linear time invariant system)
MIMO	с несколькими входами и несколькими выходами (multiple input, multiple output)
MLC	управление на основе машинного обучения (machine learning control)
MPE	оценка отсутствующей точки (missing point estimation)
mrDMD	многомасштабное разложение по динамическим модам (multiresolution dynamic mode decomposition)
NARMAX	нелинейная авторегрессионная модель с экзогенными входами (nonlinear autoregressive model with exogenous inputs)
OKID	идентификация наблюдателей с помощью фильтра Калмана (observer Kalman filter identification)
PBH	критерий Попова–Белевича–Хаутуса (Popov–Belevitch–Hautus test)
PCP	преследование главных компонент (principal component pursuit)
PDE-FIND	функциональная идентификация нелинейной динамики с уравнениями в частных производных (partial differential equation functional identification of nonlinear dynamics)

PDF	функция распределения вероятностей (probability distribution function)
PIV	анемометрия по изображениям частиц (particle image velocimetry)
RIP	свойство ограниченной изометрии (restricted isometry property)
RKHS	гильбертово пространство с воспроизводящим ядром (reproducing kernel Hilbert space)
RPCA	робастный метод главных компонент (robust principal components analysis)
rSVD	рандомизированное SVD (randomized SVD)
SINDy	разреженная идентификация нелинейных систем (sparse identification of nonlinear dynamics)
SISO	с одним входом и одним выходом (single input, single output)
SRC	разреженное представление для классификации (sparse representation for classification)
SSA	анализ сингулярного спектра (singular spectrum analysis)
STLS	последовательный метод наименьших квадратов с порогом (sequential thresholded least-squares)
SVM	метод опорных векторов (support vector machine)
TICA	метод независимых компонент с задержкой (time-lagged independent component analysis)
VAC	вариационный подход к конформационной динамике (variational approach of conformation dynamics)

Часть I

ПОНИЖЕНИЕ РАЗМЕРНОСТИ И ПРЕОБРАЗОВАНИЯ

Глава 1

Сингулярное разложение (SVD)

Сингулярное разложение (SVD) – одно из самых важных разложений матриц, появившихся в компьютерную эру, оно лежит в основе почти всех методов обработки данных, рассматриваемых в этой книге. SVD – это численно устойчивое разложение матрицы, которое применимо для самых разных целей и при этом гарантированно существует. Мы будем использовать SVD для получения низкоранговых аппроксимаций матриц и для вычисления псевдообращения неквадратных матриц, т. е. нахождения решения системы уравнений вида $\mathbf{Ax} = \mathbf{b}$. Еще одно важное применение SVD находит как алгоритм, лежащий в основе метода главных компонент (principal component analysis – PCA), идея которого – выделение статистически значимых факторов из многомерных данных. Комбинация SVD+PCA применяется к решению широкого круга научно-технических задач.

В каком-то смысле SVD является обобщением быстрого преобразования Фурье (БПФ) – темы следующей главы. Учебники прикладной математики часто начинаются с описания БПФ, поскольку это основа многих классических аналитических и численных результатов. Однако БПФ работает в идеализированной постановке, тогда как SVD – более общая техника, основанная на анализе предъявленных данных. Поскольку эта книга посвящена данным, мы начнем с сингулярного разложения, которое можно рассматривать как базис, созданный *специально* под конкретные данные, в отличие от БПФ, предоставляющего *общий* базис.

Во многих предметных областях сложные системы порождают данные, которые естественно организуются в виде больших матриц или, более общо, массивов. Например, временной ряд, получающийся в результате эксперимента или имитационного моделирования, можно представить в виде матрицы, каждый столбец которой содержит все измерения в один момент времени. Если данные в каждый момент времени многомерные, как при построении трехмерной модели погоды высокой разрешающей способности, то их можно *разгладить*, преобразовав в длинный вектор-столбец, и собрать такие столбцы в большую матрицу. Аналогично значения пикселей полутонового изображения можно сохранить в виде матрицы или преобразовать в длинные векторы-столбцы, которые в совокупности образуют матрицу, представляющую кадры фильма. Примечательно, что данные, порождаемые

такими системами, обычно имеют низкий ранг. Это означает, что можно выделить несколько доминирующих паттернов, которые объясняют многомерные данные. SVD как раз и является численно устойчивым и эффективным методом обнаружения таких паттернов в данных.

1.1. Общие сведения

Сейчас мы вкратце опишем SVD и разовьем интуицию, которая поможет продемонстрировать SVD на ряде мотивирующих примеров. SVD лежит в основе многих других методов, описанных в этой книге, в т. ч. методов классификации в главе 5, разложения по динамическим модам (DMD) в главе 7 и собственного ортогонального разложения (POD) в главе 11. Подробно математические свойства обсуждаются в последующих разделах.

Высокая размерность – самая главная трудность при обработке данных, порождаемых сложными системами. Такие наборы данных могут включать аудио, изображения и видео. Данные могут также генерироваться физической системой, например записи электрической активности мозга или измерения скорости течения жидкости в модели либо в эксперименте. Замечено, что во многих естественно возникающих системах в данных присутствуют преобладающие паттерны, характеризуемые аттрактором или многообразием низкой размерности [252, 251].

Рассмотрим, например, типичное изображение – оно содержит много измерений (пикселей) и, стало быть, является точкой в многомерном векторном пространстве. Однако большинство изображений отлично сжимаются, т. е. существенную информацию можно представить подпространством гораздо более низкой размерности. Сжимаемость изображений будет подробно обсуждаться на страницах этой книги. Сложные гидрогазодинамические системы, например атмосфера Земли или турбулентная спутная струя за кормой автомобиля, также дают убедительные примеры низкоразмерной структуры, скрывающейся в пространстве состояний высокой размерности. Хотя в точных моделях поведения жидкости или газа количество степеней свободы исчисляется миллионами и миллиардами, часто в потоке можно выделить доминирующие когерентные структуры, например периодические завихрения за кормой автомобиля или ураганы в атмосфере.

SVD предлагает систематический способ нахождения низкоразмерной аппроксимации данных высокой размерности в терминах доминирующих паттернов. Эту технику можно назвать *управляемой данными*, поскольку для обнаружения паттернов нужны только данные, без привлечения экспертных знаний или интуиции. Метод SVD обладает численной устойчивостью и дает иерархическое представление данных в новой системе координат, определяемой доминирующими корреляциями внутри данных. Кроме того, гарантируется, что сингулярное разложение, в отличие от спектрального, существует для любой матрицы.

SVD имеет много полезных приложений, помимо понижения размерности данных. Этот метод используется для вычисления псевдообратной матрицы для неквадратных матриц и тем самым позволяет найти решения недоопре-

деленных или переопределенных матричных уравнений вида $\mathbf{Ax} = \mathbf{b}$. Также мы будем использовать SVD для очистки наборов данных от шумов. Не менее важно, что SVD позволяет охарактеризовать входную и выходную геометрию линейного отображения векторных пространств. Все эти приложения будут рассмотрены в данной главе и дадут нам возможность развить интуитивное понимание матриц и данных высокой размерности.

Определение SVD

В общем случае нас интересует анализ большого набора данных $\mathbf{X} \in \mathbb{C}^{n \times m}$:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix}. \tag{1.1}$$

Столбцы $\mathbf{x}_k \in \mathbb{C}^n$ могут представлять измерения, полученные в процессе моделирования или эксперимента. Например, это могут быть изображения, вытянутые в векторы-столбцы длиной, равной количеству пикселей в изображении. Векторы-столбцы могут также представлять состояние нестационарной физической системы, например скорости течения жидкости в выбранных точках, множество измерений электрической активности мозга или состояние метеорологической модели с разрешением 1 км.

Индекс k – это метка, обозначающая k-й набор измерений. Во многих примерах из этой книги $\mathbf{X}$ будет содержать *временные ряды* данных, а $\mathbf{x}_k = \mathbf{x}(k\Delta t)$. Зачастую *размерность состояния* n очень велика – порядка миллионов или миллиардов степеней свободы. Столбцы иногда называют (моментальными) *снимками*, а m обозначает количество снимков в $\mathbf{X}$. Для многих систем $n \gg m$, поэтому получается *высокая и тощая* матрица, в отличие от *низкой и толстой* в случае, когда $n \ll m$.

SVD – это однозначно определенное разложение, существующее для любой комплексной матрицы $\mathbf{X} \in \mathbb{C}^{n \times m}$:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \tag{1.2}$$

где $\mathbf{U} \in \mathbb{C}^{n \times n}$ и $\mathbf{V} \in \mathbb{C}^{m \times m}$ – *унитарные* матрицы[1] с ортонормированными столбцами, а $\mathbf{\Sigma} \in \mathbb{R}^{n \times m}$ – матрица, диагональные элементы которой вещественны и неотрицательны, а все остальные равны нулю. Здесь символ * обозначает операцию комплексного сопряжения и транспонирования[2]. Ниже мы увидим, что унитарность $\mathbf{U}$ и $\mathbf{V}$ важна и часто используется.

Если $n \geq m$, то на диагонали матрицы $\mathbf{\Sigma}$ расположено не более m ненулевых элементов, и ее можно записать в виде $\mathbf{\Sigma} = \begin{bmatrix} \hat{\mathbf{\Sigma}} \\ \mathbf{0} \end{bmatrix}$. Поэтому $\mathbf{X}$ можно *точно* представить, воспользовавшись *экономной* формой SVD:

---

[1] Квадратная матрица $\mathbf{U}$ называется унитарной, если $\mathbf{U}\mathbf{U}^* = \mathbf{U}^*\mathbf{U} = \mathbf{I}$.

[2] Для вещественных матриц эта операция совпадает с обычным транспонированием: $\mathbf{X}^* = \mathbf{X}^T$.

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* = \begin{bmatrix}\hat{\mathbf{U}} & \hat{\mathbf{U}}^\perp\end{bmatrix}\begin{bmatrix}\hat{\mathbf{\Sigma}} \\ \mathbf{0}\end{bmatrix}\mathbf{V}^* = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\mathbf{V}^*. \tag{1.3}$$

Полное и экономное SVD показаны на рис. 1.1. На столбцы матрицы $\hat{\mathbf{U}}^\perp$ натянуто векторное пространство, дополнительное и ортогональное к натянутому на столбцы $\hat{\mathbf{U}}$. Столбцы $\mathbf{U}$ называются *левыми сингулярными векторами* $\mathbf{X}$, а столбцы $\mathbf{V}$ – *правыми сингулярными векторами*. Диагональные элементы $\hat{\mathbf{\Sigma}} \in \mathbb{C}^{m\times m}$ называются *сингулярными значениями*, они расположены в порядке убывания. Ранг $\mathbf{X}$ равен количеству ненулевых сингулярных значений.

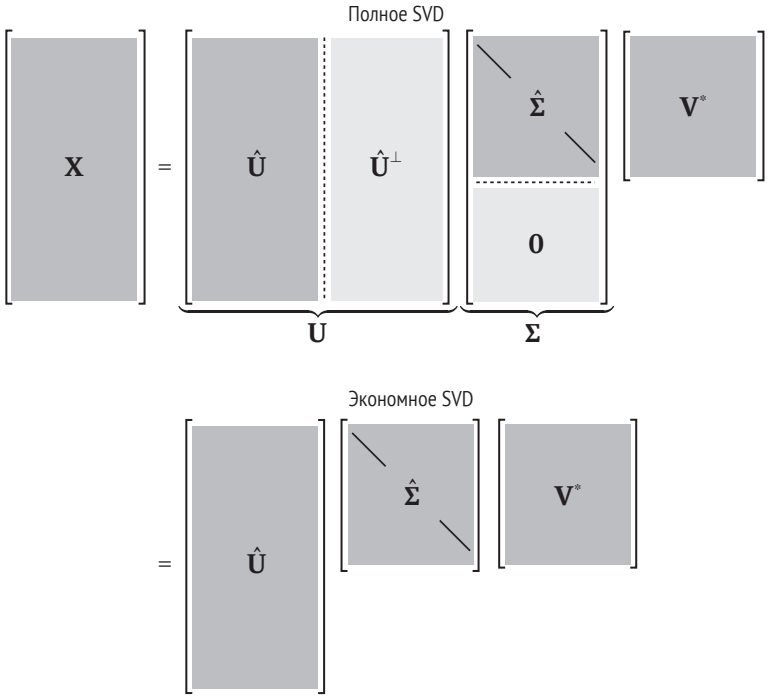

Рис. 1.1 ❖ Схема матриц в полном и экономном SVD

Вычисление SVD

SVD – краеугольный камень вычислительных методов в науке и технике, а численная реализация SVD важна и познавательна с математической точки зрения. Впрочем, большинство стандартных численных реализаций хорошо отработаны, к ним существует простой интерфейс из многих современных языков программирования, что позволяет нам абстрагироваться от деталей вычисления SVD. Как правило, мы будем просто использовать SVD как часть решения более крупной задачи и примем существование эффективных и устойчивых численных алгоритмов как данность. В последующих разделах мы продемон-

стрируем использование SVD на разных языках программирования, а также обсудим большинство стандартных стратегий вычисления и их ограничений. На тему вычисления SVD есть немало важных результатов [212, 106, 211, 292, 238]. Более полное обсуждение вычислительных аспектов можно найти в работе [214]. Для вычисления SVD очень больших матриц все чаще используют рандомизированные алгоритмы, этот вопрос обсуждается в разделе 1.8.

MATLAB. В MATLAB SVD вычисляется прямолинейно:

```
>>X = randn(5,3);   % создать случайную матрицу 5×3
>>[U,S,V] = svd(X); % сингулярное разложение
```

Для неквадратных матриц **X** экономное SVD эффективнее:

```
>>[Uhat,Shat,V] = svd(X,'econ'); % экономное SVD
```

Python

```
>>> import numpy as np
>>> X = np.random.rand(5, 3) % create random data matrix
>>> U, S, V = np.linalg.svd(X,full_matrices=True) % full SVD
>>> Uhat, Shat, Vhat = np.linalg.svd(X, full_matrices=False)
    % economy SVD
```

R

```
> X <- replicate(3, rnorm(5))
> s <- svd(X)
> U <- s$u
> S <- diag(s$d)
> V <- s$v
```

Mathematica

```
In:= X=RandomReal[{0,1},{5,3}]
In:= {U,S,V} = SingularValueDecomposition[X]
```

Другие языки

SVD реализовано и на других языках, например Fortran и C++. На самом деле большинство реализаций SVD основаны на библиотеке LAPACK (Linear Algebra Package) [13], написанной на Fortran. В LAPACK подпрограмма вычисления SVD называется **DGESVD**, а она уже обернута функциями C++ в библиотеках **Armadillo** и **Eigen**.

Историческая справка

SVD имеет давнюю и богатую историю, от ранних работ, в которых был заложен теоретический фундамент, до современных исследований по численной устойчивости и эффективности. Отличный исторический обзор приведен в работе Stewart [502], где описан общий контекст и многие важные детали. В этом обзоре много внимания уделено ранним теоретическим работам Бельтрами и Джордана (1873), Сильвестра (1889), Шмидта (1907) и Вейля (1912). Обсуждаются также более поздние работы, включая основополагаю-

щие труды по вычислительной стороне проблемы Голуба с сотрудниками [212, 211]. Кроме того, в современных учебниках [524, 17, 316] имеются прекрасно написанные главы, посвященные SVD.

Использование в этой книге и предположения о подготовке читателей

SVD – основа многих методов понижения размерности. К ним относятся метод главных компонент (PCA) в статистике [418, 256, 257], преобразование Карунена–Лоэва (KLT) [280, 340], эмпирические ортогональные функции (EOF) в изучении климата [344], собственное ортогональное разложение (POD) в гидродинамике [251] и канонический корреляционный анализ (CCA) [131]. Хотя многие из этих методов разрабатывались независимо для разных областей знания, различаются они только способами сбора и предварительной обработки данных. В работе Gerbrands [204] приведено превосходное обсуждение связи между SVD, KLT и PCA.

SVD широко используется для идентификации систем и в теории управления для получения моделей пониженного порядка, сбалансированных в том смысле, что состояния иерархически упорядочены в терминах доступности наблюдению и управляемости с помощью приводов [388].

В этой главе предполагается, что читатель знаком с линейной алгеброй и имеет некоторый опыт применения вычислительных методов. Если необходимо освежить познания, то существует много отличных книг по численной линейной алгебре, в т. ч. с обсуждением SVD, например [524, 17, 316].

1.2. Аппроксимация матриц

Пожалуй, самым полезным и определяющим свойством SVD является тот факт, что оно дает *оптимальную* низкоранговую аппроксимацию матрицы $\mathbf{X}$. На самом деле SVD позволяет построить *иерархию* низкоранговых аппроксимаций, поскольку для получения аппроксимации ранга r нужно просто оставить первые r сингулярных значений и векторов, а остальные отбросить.

Шмидт обобщил SVD на пространства функций и доказал теорему, устанавливающую, что усеченное SVD является оптимальной низкоранговой аппроксимацией исходной матрицы $\mathbf{X}$ [476]. Теорема Шмидта была заново открыта в работе Eckart and Young [170], поэтому иногда ее называют теоремой Эккарта–Янга.

Теорема 1 (Eckart–Young [170]). *Оптимальную в смысле наименьших квадратов аппроксимацию* $\mathbf{X}$ *ранга* r *дает усеченное SVD* $\tilde{\mathbf{X}}$ *ранга* r:

$$\underset{\tilde{\mathbf{X}} \text{ такое, что } \operatorname{rank}(\tilde{\mathbf{X}})=r}{\operatorname{argmin}} ||\mathbf{X} - \tilde{\mathbf{X}}||_F = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*. \tag{1.4}$$

Здесь $\tilde{\mathbf{U}}$ *и* $\tilde{\mathbf{V}}$ *обозначают матрицы, образованные первыми* r *столбцами* $\mathbf{U}$ *и* $\mathbf{V}$, *а* $\tilde{\mathbf{\Sigma}}$ *содержит левый верхний блок* $\mathbf{\Sigma}$ *размера* $r \times r$. $||\cdot||_F$ *– норма Фробениуса.*

Согласно этой нотации базис усеченного SVD (и аппроксимирующая матрица $\tilde{\mathbf{X}}$) обозначается $\tilde{\mathbf{X}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*$. Поскольку $\mathbf{\Sigma}$ – диагональная матрица, SVD-аппроксимация ранга r может быть представлена суммой r матриц ранга 1:

$$\tilde{\mathbf{X}} = \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^* = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^* + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^* + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^*. \tag{1.5}$$

Это так называемое *диадическое* суммирование. Для любого ранга r не существует лучшей аппроксимации $\mathbf{X}$ в смысле нормы ℓ_2, чем усеченная SVD-аппроксимация $\tilde{\mathbf{X}}$. Таким образом, данные высокой размерности хорошо описываются несколькими доминирующими паттернами, определяемыми столбцами $\tilde{\mathbf{U}}$ и $\tilde{\mathbf{V}}$.

Это важное свойство SVD, к которому мы будем много раз возвращаться. Есть многочисленные примеры наборов данных, содержащих измерения высокой размерности, которые приводят к большой матрице $\mathbf{X}$. Однако в данных часто присутствуют доминирующие паттерны низкой размерности, и базис усеченного SVD $\tilde{\mathbf{U}}$ определяет преобразование координат из пространства измерений высокой размерности в пространство паттернов низкой размерности. В результате уменьшается размер и размерность больших наборов данных, а значит, открывается возможность для визуализации и анализа. Наконец, многие системы, рассматриваемые в этой книге, *динамические* (см. главу 7), а базис SVD дает иерархию мод, характеризующую наблюдаемый аттрактор, на который мы можем спроецировать динамическую систему низкой размерности для получения моделей пониженного порядка (см. главу 12).

Усечение

Усеченное SVD показано на рис. 1.2, где $\tilde{\mathbf{U}}$, $\tilde{\mathbf{\Sigma}}$ и $\tilde{\mathbf{V}}$ обозначают усеченные матрицы. Если $\mathbf{X}$ – матрица неполного ранга, то некоторые сингулярные значения в $\tilde{\mathbf{\Sigma}}$ могут быть равны нулю, и тогда усеченное SVD остается точным. Однако если r меньше числа ненулевых сингулярных значений (т. е. ранга $\mathbf{X}$), то усеченное SVD является всего лишь аппроксимацией $\mathbf{X}$:

$$\mathbf{X} \approx \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*. \tag{1.6}$$

Есть много способов выбора ранга усеченной матрицы r, они обсуждаются в разделе 1.7. Если мы требуем, чтобы усеченная матрица содержала все ненулевые сингулярные значения, то равенство $\mathbf{X} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*$ точное.

Пример: сжатие изображения

Проиллюстрируем идею аппроксимации матриц на простом примере: сжатие изображения. Тема, которая красной нитью проходит через всю книгу, – наличие в больших наборах данных паттернов, благодаря чему возможны низкоранговые представления. Естественные изображения дают простой

и интуитивно понятный пример такой внутренне присущей *сжимаемости*. Полутоновое изображение можно рассматривать как вещественную матрицу $\mathbf{X} \in \mathbb{R}^{n \times m}$, где n и m – числа пикселей по вертикали и по горизонтали соответственно[1]. В зависимости от базиса представления (пространство пикселей, частотная область в смысле преобразования Фурье, преобразованные с помощью SVD координаты) изображение может иметь очень компактные аппроксимации.

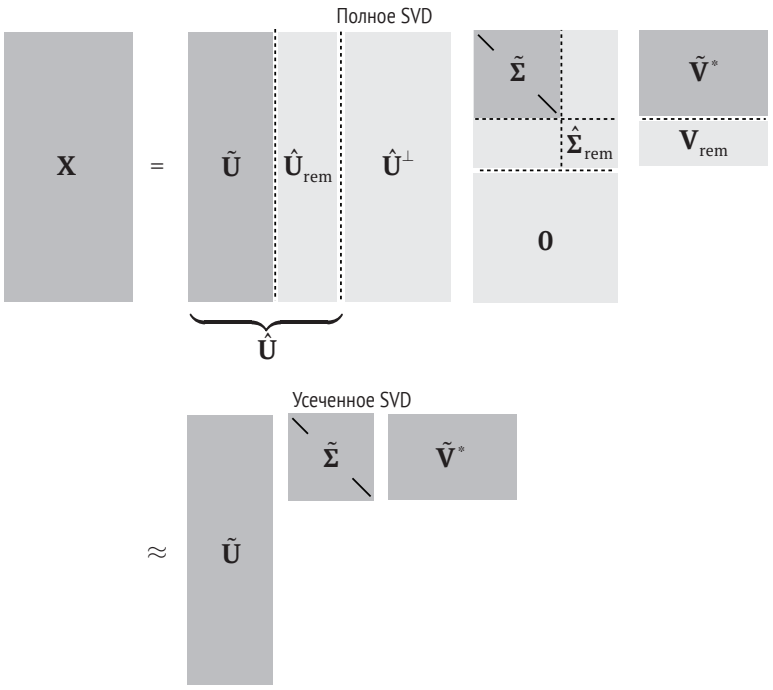

Рис. 1.2 ❖ Схема усеченного SVD.
Нижний индекс «rem» обозначает остаток $\hat{\mathbf{U}}, \hat{\mathbf{\Sigma}}$ или $\mathbf{V}$ после усечения

Рассмотрим изображение собаки по кличке Мордекай на снегу (рис. 1.3). Его размер 2000×1500 пикселей. Мы можем вычислить SVD этого изображения и нанести на график сингулярные значения (рис. 1.4). На рис. 1.3 показаны приближенные матрицы $\tilde{\mathbf{X}}$ для разных значений r. При $r = 100$ реконструированное изображение вполне точное, а сингулярные значения отражают почти 80 % неоднородности изображения. Усечение SVD приводит к сжатию исходного изображения, поскольку в $\tilde{\mathbf{U}}, \tilde{\mathbf{\Sigma}}$ и $\tilde{\mathbf{V}}$ нужно хранить только первые 100 столбцов $\mathbf{U}$ и $\mathbf{V}$ плюс первые 100 диагональных элементов $\mathbf{\Sigma}$.

---

[1] Размер изображения часто задают, указывая сначала размер по горизонтали, а затем по вертикали, т. е. $\mathbf{X}^T \in \mathbb{R}^{m \times n}$, но мы будем придерживаться противоположного соглашения, совпадающего с общепринятым порядком обозначения размера матрицы.

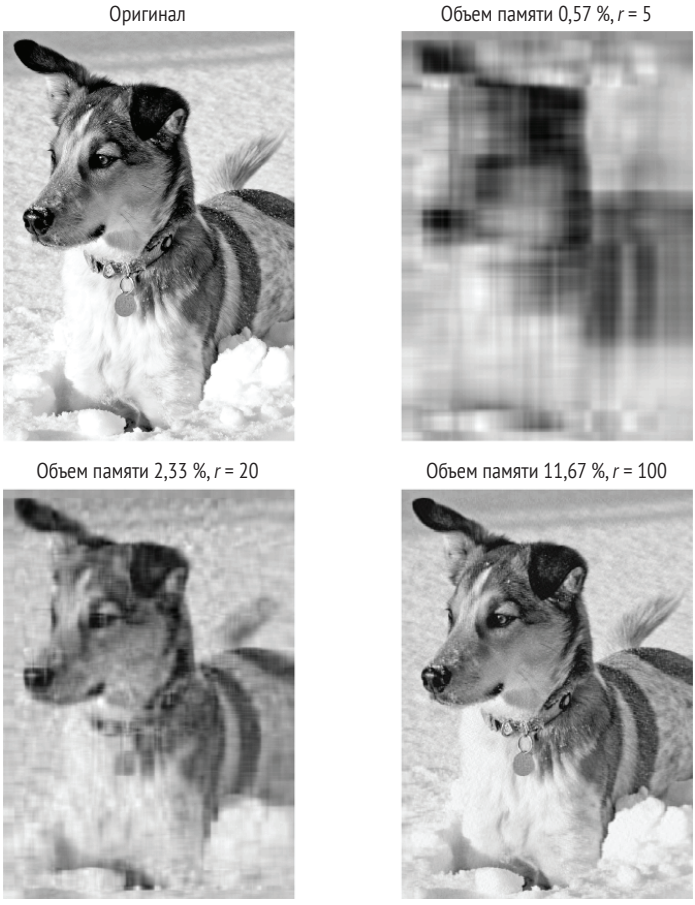

Рис. 1.3 ❖ Сжатие изображения собаки Мордекая на снегу путем усечения SVD с различными значениями ранга *r*. Разрешение исходного изображения 2000×1500

Сначала загрузим изображение:

```
A=imread('../DATA/dog.jpg');
X=double(rgb2gray(A)); % преобразовать RBG в полутоновое, 256 бит->double.
nx = size(X,1); ny = size(X,2);
imagesc(X), axis off, colormap gray
```

и вычислим SVD:

```
[U,S,V] = svd(X);
```

Затем вычислим приближенную матрицу, используя усеченные SVD с разными рангами (r = 5, 20, 100):

```
for r=[5 20 100]; % Truncation value
    Xapprox = U(:,1:r)*S(1:r,1:r)*V(:,1:r)'; % Approx. image
```

```
    figure, imagesc(Xapprox), axis off
    title(['r=',num2str(r,'%d'),']);
end
```

Наконец, построим графики сингулярных значений и суммарной энергии, изображенные на рис. 1.4.

```
subplot(1,2,1), semilogy(diag(S),'k')
subplot(1,2,2), plot(cumsum(diag(S))/sum(diag(S)),'k')
```

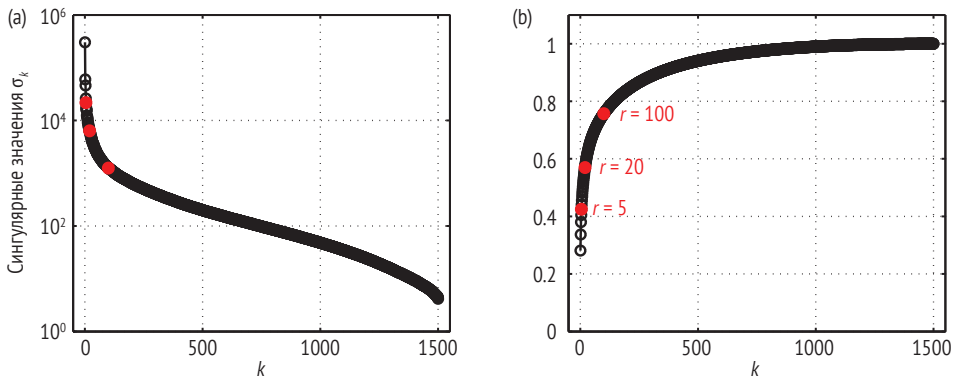

Рис. 1.4 ❖ (a) Сингулярные значения σ_k. (b) Суммарная энергия в первых k модах

1.3. Математические свойства и манипуляции

Опишем важные математические свойства SVD, в т. ч. геометрические интерпретации унитарных матриц **U** и **V**, а также обсудим SVD в терминах доминирующих корреляций, присутствующих в данных **X**. Связь между SVD и корреляциями данных мы объясним в разделе 1.5, посвященном методу главных компонент.

Интерпретация с привлечением доминирующих корреляций

Сингулярное разложение тесно связано с задачей о собственных значениях, в которой фигурируют корреляционные матрицы $\mathbf{XX}^*$ и $\mathbf{X}^*\mathbf{X}$, показанные на рис. 1.5 для конкретного изображения, а на рис. 1.6 и 1.7 для матриц общего вида. Подставив (1.3) в $\mathbf{XX}^*$ и $\mathbf{X}^*\mathbf{X}$, получаем:

$$\mathbf{XX}^* = \mathbf{U}\begin{bmatrix}\hat{\mathbf{\Sigma}}\\ \mathbf{0}\end{bmatrix}\mathbf{V}^*\mathbf{V}\begin{bmatrix}\hat{\mathbf{\Sigma}} & \mathbf{0}\end{bmatrix}\mathbf{U}^* = \mathbf{U}\begin{bmatrix}\hat{\mathbf{\Sigma}}^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{U}^*; \tag{1.7a}$$

$$\mathbf{X}^*\mathbf{X} = \mathbf{V}\begin{bmatrix}\hat{\mathbf{\Sigma}} & \mathbf{0}\end{bmatrix}\mathbf{U}^*\mathbf{U}\begin{bmatrix}\hat{\mathbf{\Sigma}} \\ \mathbf{0}\end{bmatrix}\mathbf{V}^* = \mathbf{V}\hat{\mathbf{\Sigma}}^2\mathbf{V}^*. \tag{1.7b}$$

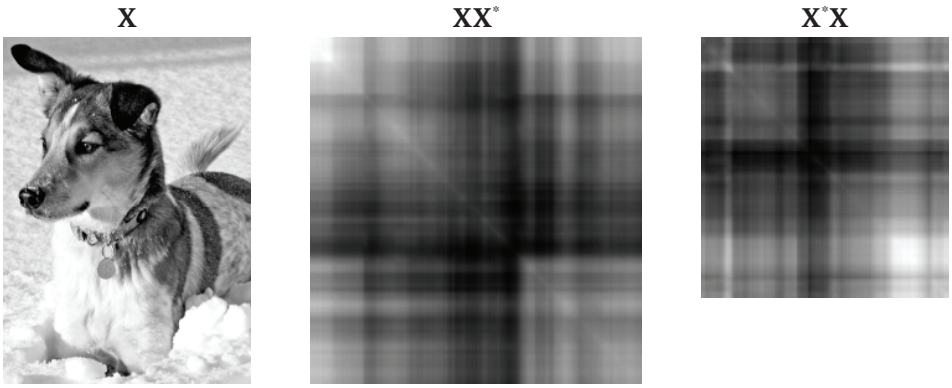

Рис. 1.5 ❖ Корреляционные матрицы **XX*** и **X*X** для матрицы **X**, полученной из изображения собаки. Заметим, что обе корреляционные матрицы симметричны

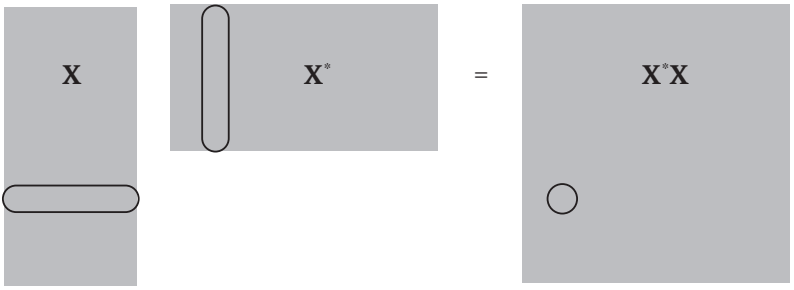

Рис. 1.6 ❖ Корреляционная матрица **XX*** получена взятием скалярных произведений строк **X**

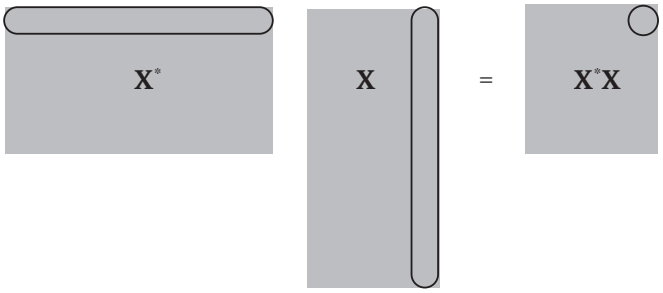

Рис. 1.7 ❖ Корреляционная матрица **X*X** получена взятием скалярных произведений столбцов **X**

Учитывая, что **U** и **V** унитарны, **U**, **Σ** и **V** являются решениями следующих задач на нахождение собственных значений:

$$\mathbf{X}\mathbf{X}^{*}\mathbf{U} = \mathbf{U}\begin{bmatrix}\hat{\mathbf{\Sigma}}^{2} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}; \tag{1.8a}$$

$$\mathbf{X}^{*}\mathbf{X}\mathbf{V} = \mathbf{V}\hat{\mathbf{\Sigma}}^{2}. \tag{1.8b}$$

Иными словами, любое ненулевое сингулярное значение **X** является положительным квадратным корнем из какого-то собственного значения матриц $\mathbf{X}^{*}\mathbf{X}$ и $\mathbf{X}\mathbf{X}^{*}$, которые имеют одинаковые множества собственных значений. Отсюда следует, что если **X** самосопряженная (т. е. $\mathbf{X} = \mathbf{X}^{*}$), то сингулярные значения **X** равны абсолютным величинам собственных значений **X**.

В результате мы получаем интуитивно понятную интерпретацию SVD: столбцы **U** являются собственными векторами корреляционной матрицы $\mathbf{X}\mathbf{X}^{*}$, а столбцы **V** – собственными векторами $\mathbf{X}^{*}\mathbf{X}$. Мы упорядочиваем сингулярные значения в порядке убывания абсолютной величины, поэтому порядок столбцов **U** отражает, какую часть корреляции между столбцами **X** они улавливают; аналогично **V** улавливает корреляцию между строками **X**.

Метод моментальных снимков

На практике часто невозможно построить матрицу $\mathbf{X}\mathbf{X}^{*}$, поскольку размерность состояния n слишком велика; что уж говорить о нахождении собственных значений. Если **X** состоит из миллиона элементов, то число элементов $\mathbf{X}\mathbf{X}^{*}$ равно триллиону. В 1987 году Сирович (Sirovich) заметил, что можно не вычислять эту большую матрицу, а найти первые m столбцов **U** с помощью метода, получившего название «метод (моментальных) снимков» [490].

Вместо того чтобы вычислять спектральное разложение $\mathbf{X}\mathbf{X}^{*}$ для получения левых сингулярных векторов **U**, мы вычисляем только спектральное разложение гораздо меньшей и простой для работы матрицы $\mathbf{X}^{*}\mathbf{X}$. Затем из (1.8b) находим **V** и $\hat{\mathbf{\Sigma}}$. Если у $\hat{\mathbf{\Sigma}}$ имеются нулевые сингулярные значения, то мы оставляем только ее часть $\tilde{\mathbf{\Sigma}}$, соответствующую r ненулевым значениям, и соответственные столбцы $\tilde{\mathbf{V}}$ матрицы **V**. Зная эти матрицы, можно следующим образом аппроксимировать матрицу $\tilde{\mathbf{U}}$, состоящую из первых r столбцов **U**:

$$\tilde{\mathbf{U}} = \mathbf{X}\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}. \tag{1.9}$$

Геометрическая интерпретация

Столбцы матрицы **U** образуют ортонормированный базис пространства столбцов **X**. Аналогично столбцы **V** образуют ортонормированный базис пространства строк **X**. Если столбцы **X** содержат пространственные измерения в разные моменты времени, то **U** кодирует пространственные паттерны, а **V** – временные паттерны.

Особенно полезным делает SVD тот факт, что $\mathbf{U}$ и $\mathbf{V}$ – *унитарные* матрицы, так что $\mathbf{UU}^* = \mathbf{U}^*\mathbf{U} = \mathbf{I}_{n\times n}$ и $\mathbf{VV}^* = \mathbf{V}^*\mathbf{V} = \mathbf{I}_{m\times m}$. Это означает, что для решения системы уравнений с матрицей $\mathbf{U}$ или $\mathbf{V}$ нужно просто умножить обе части на транспонированную матрицу. Сложность этой операции составляет $O(n^2)$, в отличие от традиционных методов обращения матрицы общего вида, имеющих сложность $O(n^3)$. Как отмечено в предыдущем разделе и в работе [57], SVD тесно связано со спектральными свойствами компактных самосопряженных операторов $\mathbf{XX}^*$ и $\mathbf{X}^*\mathbf{X}$.

Сингулярное разложение $\mathbf{X}$ можно геометрически интерпретировать, рассмотрев отображение гиперсферы $S^{n-1} \triangleq \{\mathbf{x} \mid \|\mathbf{x}\|_2 = 1\} \subset \mathbb{R}^n$ в эллипсоид, $\{\mathbf{y} \mid \mathbf{y} = \mathbf{Xx}$ для $\mathbf{x} \in S^{n-1}\} \subset \mathbb{R}^m$ посредством умножения на $\mathbf{X}$. Графически это показано на рис. 1.8 для сферы в $\mathbb{R}^3$ и отображения посредством умножения на $\mathbf{X}$ с тремя ненулевыми сингулярными значениями. Поскольку умножение на матрицу – линейное отображение, достаточно знать, как оно ведет себя на единичной сфере, чтобы вычислить образ любого вектора.

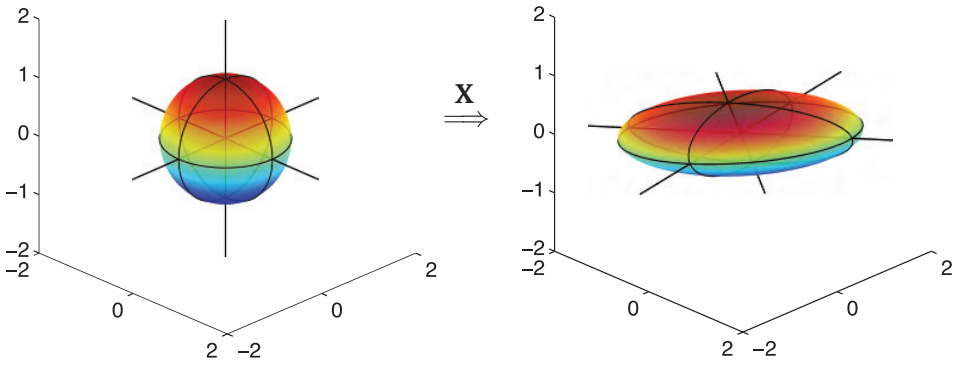

Рис. 1.8 ❖ Геометрическая иллюстрация SVD как отображения сферы в $\mathbb{R}^n$ в эллипсоид в $\mathbb{R}^m$

Для частного случая на рис. 1.8 мы строим матрицу $\mathbf{X}$ из трех матриц поворота $\mathbf{R}_x$, $\mathbf{R}_y$ и $\mathbf{R}_z$ и четвертой матрицы, описывающей растяжение вдоль главных осей:

$$\mathbf{X} = \underbrace{\begin{bmatrix}\cos(\theta_3) & -\sin(\theta_3) & 0\\ \sin(\theta_3) & \cos(\theta_3) & 0\\ 0 & 0 & 1\end{bmatrix}}_{\mathbf{R}_z}\underbrace{\begin{bmatrix}\cos(\theta_2) & 0 & \sin(\theta_2)\\ 0 & 1 & 0\\ -\sin(\theta_2) & 0 & \cos(\theta_2)\end{bmatrix}}_{\mathbf{R}_y}$$
$$\times\underbrace{\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\theta_1) & -\sin(\theta_1)\\ 0 & \sin(\theta_1) & \cos(\theta_1)\end{bmatrix}}_{\mathbf{R}_y}\begin{bmatrix}\sigma_1 & 0 & 0\\ 0 & \sigma_2 & 0\\ 0 & 0 & \sigma_3\end{bmatrix}.$$

В этом случае $\theta_1 = \pi/15$, $\theta_2 = -\pi/9$ и $\theta_3 = -\pi/20$, а $\sigma_1 = 3$, $\sigma_2 = 1$, $\sigma_3 = 0.5$. Матрицы поворота не коммутируют, поэтому порядок поворотов имеет значение. Если одно из сингулярных значений равно нулю, то соответствующая размерность исчезает и эллипсоид становится фигурой в подпространстве меньшей размерности. Произведение $\mathbf{R}_x\mathbf{R}_y\mathbf{R}_z$ является унитарной матрицей $\mathbf{U}$ в сингулярном разложении $\mathbf{X}$. Матрица $\mathbf{V}$ в данном случае единичная.

Листинг 1.1 ❖ Построение матриц поворота

```
theta = [pi/15; -pi/9; -pi/20];
Sigma = diag([3; 1; 0.5]);              % масштабировать x, y, z
Rx = [1 0 0;                            % поворот вокруг оси x
0 cos(theta(1)) -sin(theta(1));
0 sin(theta(1)) cos(theta(1))];
Ry = [cos(theta(2)) 0 sin(theta(2));    % поворот вокруг оси y
0 1 0;
-sin(theta(2)) 0 cos(theta(2))];
Rz = [cos(theta(3)) -sin(theta(3)) 0;   % поворот вокруг оси z
sin(theta(3)) cos(theta(3)) 0;
0 0 1];
X = Rz*Ry*Rx*Sigma;                     % повернуть и масштабировать
```

Листинг 1.2 ❖ Изображение сферы

```
[x,y,z] = sphere(25);
h1=surf(x,y,z);
```

Листинг 1.3 ❖ Отображение сферы путем умножения на **X** и изображение получившегося эллипсоида

```
xR = 0*x; yR = 0*y; zR = 0*z;
for i=1:size(x,1)
    for j=1:size(x,2)
        vecR = X*[x(i,j); y(i,j); z(i,j)];
        xR(i,j) = vecR(1);
        yR(i,j) = vecR(2);
        zR(i,j) = vecR(3);
    end
end
h2=surf(xR,yR,zR,z);      % координата z сферы задает цвет
```

Инвариантность SVD относительно унитарных преобразований

У SVD есть полезное свойство: если умножить матрицу данных $\mathbf{X}$ слева или справа на унитарную матрицу, то члены сингулярного разложения не изменятся, за исключением левой или правой унитарной матрицы $\mathbf{U}$ или $\mathbf{V}$ соответственно. У этого факта есть важные следствия, поскольку дискретное преобразование Фурье (ДПФ, см. главу 2) $\mathcal{F}$ является унитарным преобразо-

ванием, а это значит, что SVD матрицы $\hat{\mathbf{X}} = \mathcal{F}\mathbf{X}$ совпадает с SVD матрицы $\mathbf{X}$ с тем отличием, что моды $\hat{\mathbf{U}}$ будут дискретными преобразованиями Фурье мод $\mathbf{U}$: $\hat{\mathbf{U}} = \mathcal{F}\mathbf{U}$. Кроме того, инвариантность SVD относительно унитарных преобразований позволяет использовать сжатые измерения для реконструкции мод SVD, разреженных в некотором базисе преобразования (см. главу 3).

Инвариантность SVD относительно унитарных преобразований геометрически очевидна, поскольку унитарное преобразование лишь поворачивает векторы в пространстве, но не изменяет их скалярные произведения и структуру корреляций. Будем обозначать левое унитарное преобразование $\mathbf{C}$, так что $\mathbf{Y} = \mathbf{CX}$, а правое унитарное преобразование $\mathbf{P}^*$, так что $\mathbf{Y} = \mathbf{XP}^*$. SVD матрицы $\mathbf{X}$ будем обозначать $\mathbf{U}_\mathbf{X}\mathbf{\Sigma}_\mathbf{X}\mathbf{V}_\mathbf{X}^*$, а SVD матрицы $\mathbf{Y}$ будет равно $\mathbf{U}_\mathbf{Y}\mathbf{\Sigma}_\mathbf{Y}\mathbf{V}_\mathbf{Y}^*$.

Левые унитарные преобразования

Сначала рассмотрим левое унитарное преобразование $\mathbf{X}$: $\mathbf{Y} = \mathbf{CX}$. Вычисление корреляционной матрицы $\mathbf{Y}^*\mathbf{Y}$ дает

$$\mathbf{Y}^*\mathbf{Y} = \mathbf{X}^*\mathbf{C}^*\mathbf{CX} = \mathbf{X}^*\mathbf{X}. \tag{1.10}$$

Спроецированные данные имеют такой же спектральный состав, т. е. $\mathbf{V}_\mathbf{X}$ и $\mathbf{\Sigma}_\mathbf{X}$ не изменяются. Применив метод снимков для реконструкции $\mathbf{U}_\mathbf{Y}$, находим

$$\mathbf{U}_\mathbf{Y} = \mathbf{YV}_\mathbf{X}\mathbf{\Sigma}_\mathbf{X}^{-1} = \mathbf{CXV}_\mathbf{X}\mathbf{\Sigma}_\mathbf{X}^{-1} = \mathbf{CU}_\mathbf{X}. \tag{1.11}$$

Таким образом, $\mathbf{U}_\mathbf{Y} = \mathbf{CU}_\mathbf{X}$, $\mathbf{\Sigma}_\mathbf{Y} = \mathbf{\Sigma}_\mathbf{X}$ и $\mathbf{V}_\mathbf{Y} = \mathbf{V}_\mathbf{X}$. Тогда SVD матрицы $\mathbf{Y}$ равно:

$$\mathbf{Y} = \mathbf{CX} = \mathbf{CU}_\mathbf{X}\mathbf{\Sigma}_\mathbf{X}\mathbf{V}_\mathbf{X}^*. \tag{1.12}$$

Правые унитарные преобразования

Для правого унитарного преобразования $\mathbf{Y} = \mathbf{XP}^*$ корреляционная матрица $\mathbf{Y}^*\mathbf{Y}$ равна

$$\mathbf{Y}^*\mathbf{Y} = \mathbf{PX}^*\mathbf{XP}^* = \mathbf{PV}_\mathbf{X}\mathbf{\Sigma}_\mathbf{X}^2\mathbf{V}_\mathbf{X}^*\mathbf{P}^* \tag{1.13}$$

и имеет такое спектральное разложение:

$$\mathbf{Y}^*\mathbf{YPV}_\mathbf{X} = \mathbf{PV}_\mathbf{X}\mathbf{\Sigma}_\mathbf{X}^2. \tag{1.14}$$

Таким образом, $\mathbf{V}_\mathbf{Y} = \mathbf{PV}_\mathbf{X}$ и $\mathbf{\Sigma}_\mathbf{Y} = \mathbf{\Sigma}_\mathbf{X}$. Мы можем воспользоваться методом снимков для реконструкции $\mathbf{U}_\mathbf{Y}$:

$$\mathbf{U}_\mathbf{Y} = \mathbf{YPV}_\mathbf{X}\mathbf{\Sigma}_\mathbf{X}^{-1} = \mathbf{XV}_\mathbf{X}\mathbf{\Sigma}_\mathbf{X}^{-1} = \mathbf{U}_\mathbf{X}. \tag{1.15}$$

Следовательно, $\mathbf{U}_\mathbf{Y} = \mathbf{U}_\mathbf{X}$, и SVD матрицы $\mathbf{Y}$ можно записать в виде:

$$\mathbf{Y} = \mathbf{XP}^* = \mathbf{U}_\mathbf{X}\mathbf{\Sigma}_\mathbf{X}\mathbf{V}_\mathbf{X}^*\mathbf{P}^*. \tag{1.16}$$

1.4. Псевдообращение, метод наименьших квадратов и регрессия

Многие физические системы можно представить линейной системой уравнений

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1.17}$$

где матрица ограничений $\mathbf{A}$ и вектор $\mathbf{b}$ известны, а вектор $\mathbf{x}$ неизвестен. Если $\mathbf{A}$ – квадратная обратимая матрица (т. е. определитель $\mathbf{A}$ не равен нулю), то существует единственное решение $\mathbf{x}$ для любого $\mathbf{b}$. Но если $\mathbf{A}$ сингулярная или прямоугольная, то может существовать одно, ни одного или бесконечно много решений в зависимости от конкретного $\mathbf{b}$ и пространств столбцов и строк $\mathbf{A}$.

Сначала рассмотрим *недоопределенную систему*, т. е. случай, когда $\mathbf{A} \in \mathbb{C}^{n\times m}$ и $n \ll m$ ($\mathbf{A}$ – низкая и толстая матрица) – уравнений меньше, чем неизвестных. Такая система, скорее всего, будет иметь полный столбцовый ранг, поскольку число столбцов много больше, чем требуется для линейно независимого базиса[1]. В общем случае, если низкая и толстая матрица $\mathbf{A}$ имеет полный столбцовый ранг, то для каждого $\mathbf{b}$ существует бесконечно много решений $\mathbf{x}$. Такая система называется *недоопределенной*, потому что элементов $\mathbf{b}$ недостаточно, чтобы определить вектор $\mathbf{x}$ высокой размерности.

Точно так же рассмотрим *переопределенную систему*, когда $n \gg m$ (высокая и тощая матрица), т. е. уравнений больше, чем неизвестных. Эта матрица не может иметь полного столбцового ранга, поэтому гарантируется, что существуют такие векторы $\mathbf{b}$, для которых нет ни одного решения $\mathbf{x}$. На самом деле решение $\mathbf{x}$ существует, только если $\mathbf{b}$ принадлежит пространству столбцов $\mathbf{A}$, т. е. $\mathbf{b} \in \mathrm{col}(\mathbf{A})$.

Технически могут существовать $\mathbf{b}$, при которых имеется бесконечно много решений $\mathbf{x}$ для высокой и тощей матрицы $\mathbf{A}$, равно как и такие $\mathbf{b}$, при которых не существует ни одного решения для низкой и толстой матрицы. Пространство решений системы (1.17) определяется четырьмя фундаментальными подпространствами $\mathbf{A} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*$, где ранг r выбран так, чтобы все ненулевые сингулярные значения были включены:

- пространство столбцов, $\mathrm{col}(\mathbf{A})$, натянутое на столбцы $\mathbf{A}$; оно называется также *областью значений*. Пространство столбцов $\mathbf{A}$ совпадает с пространством столбцов $\tilde{\mathbf{U}}$;
- ортогональное дополнение к $\mathrm{col}(\mathbf{A})$ обозначается $\ker(\mathbf{A}^*)$ и совпадает с пространством столбцов матрицы $\hat{\mathbf{U}}^{\perp}$ на рис. 1.1;
- пространство строк, $\mathrm{row}(\mathbf{A})$, натянутое на строки $\mathbf{A}$ и совпадающее с линейной оболочкой столбцов $\tilde{\mathbf{V}}$. Имеет место равенство $\mathrm{row}(\mathbf{A}) = \mathrm{col}(\mathbf{A}^*)$;

[1] Легко построить вырожденные примеры низкой и толстой матрицы неполного столбцового ранга, например $\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$.

- ядерное пространство, $\ker(\mathbf{A})$, являющееся ортогональным дополнением к $\text{row}(\mathbf{A})$ и называемое также *нуль-пространством*, или *ядром*. Ядро – это подпространство, состоящее из векторов, которые $\mathbf{A}$ отображает в нуль, т. е. $\mathbf{Ax} = \mathbf{0}$; оно совпадает с $\text{col}(\hat{\mathbf{V}}^{\perp})$.

Точнее, если $\mathbf{b} \in \text{col}(\mathbf{A})$ и $\dim(\ker(\mathbf{A})) \neq 0$, то существует бесконечно много решений $\mathbf{x}$. Заметим, что условие $\dim(\ker(\mathbf{A})) \neq 0$ гарантированно выполняется для низкой и толстой матрицы. Аналогично, если $\mathbf{b} \notin \text{col}(\mathbf{A})$, то решений не существует, и система уравнений (1.17) называется *несовместной*.

Описанные выше фундаментальные подпространства обладают следующими свойствами:

$$\text{col}(\mathbf{A}) \oplus \ker(\mathbf{A}^*) = \mathbb{R}^n \tag{1.18a}$$

$$\text{col}(\mathbf{A}^*) \oplus \ker(\mathbf{A}) = \mathbb{R}^n. \tag{1.18b}$$

Замечание 1. *Имеется обширная литература по теории случайных матриц, в которой перечисленные выше утверждения почти всегда верны, т. е. верны с высокой вероятностью. Например, крайне маловероятно, что система* $\mathbf{Ax} = \mathbf{b}$ *имеет решение для случайно выбранных матрицы* $\mathbf{A} \in \mathbb{R}^{n \times m}$ и вектора $\mathbf{b} \in \mathbb{R}^n$, *где* $n \gg m$, *т. к. мало шансов, что* $\mathbf{b}$ *принадлежит пространству столбцов* $\mathbf{A}$. *Эти свойства случайных матриц будут играть важную роль в теории сжатых измерений (см. главу 3).*

В переопределенном случае, когда решений не существует, нам часто хотелось бы найти вектор $\mathbf{x}$, который минимизирует сумму квадратов ошибок $||\mathbf{Ax} - \mathbf{b}||_2^2$, он называется решением *с наименьшей среднеквадратической ошибкой*. Заметим, что такое решение минимизирует также величину $||\mathbf{Ax} - \mathbf{b}||_2$. В недоопределенном случае, когда существует бесконечно много решений, часто требуется найти решение $\mathbf{x}$, для которого $\mathbf{Ax} = \mathbf{b}$, а норма $||\mathbf{x}||_2$ минимальна; оно называется решением с *минимальной нормой*.

SVD – общепризнанный метод решения этих важных задач оптимизации. Прежде всего если подставить точное усеченное SVD $\mathbf{A} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*$ вместо $\mathbf{A}$, то каждую из матриц $\tilde{\mathbf{U}}$, $\tilde{\mathbf{\Sigma}}$ и $\tilde{\mathbf{V}}^*$ можно будет поочередно «обратить», что даст *левую псевдообратную матрицу* Мура–Пенроуза [425, 426, 453, 572] $\mathbf{A}^{\dagger}$:

$$\mathbf{A}^{\dagger} \triangleq \tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}^* \;\Rightarrow\; \mathbf{A}^{\dagger}\mathbf{A} = \mathbf{I}^{m \times m}. \tag{1.19}$$

Ее можно использовать для нахождения решений системы (1.17) с наименьшей среднеквадратической ошибкой и с минимальной нормой:

$$\mathbf{A}^{\dagger}\mathbf{A}\tilde{\mathbf{x}} = \mathbf{A}^{\dagger}\mathbf{b} \;\Rightarrow\; \tilde{\mathbf{x}} = \tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}^*\mathbf{b}. \tag{1.20}$$

Подставляя решение $\tilde{\mathbf{x}}$ обратно в (1.17), получаем:

$$\mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}^*\mathbf{b} \tag{1.21a}$$

$$\mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{U}}\tilde{\mathbf{U}}^*\mathbf{b}. \tag{1.21b}$$

Заметим, что $\tilde{\mathbf{U}}\tilde{\mathbf{U}}^*$ – необязательно единичная матрица, но является проекцией на пространство столбцов $\tilde{\mathbf{U}}$. Поэтому $\tilde{\mathbf{x}}$ будет точным решением системы (1.17), только если $\mathbf{b}$ принадлежит пространству столбцов $\tilde{\mathbf{U}}$, а значит, пространству столбцов $\mathbf{A}$.

Вычислить псевдообратную матрицу $\mathbf{A}^\dagger$ можно эффективно, если предварительно произвести затратное вычисление SVD. Обращение унитарных матриц $\tilde{\mathbf{U}}$ и $\tilde{\mathbf{V}}$ сводится к умножению на транспонированные матрицы, для чего требуется $O(n^2)$ операций. Обращение диагональной матрицы $\tilde{\mathbf{\Sigma}}$ еще эффективнее и требует всего $O(n)$ операций. С другой стороны, обращение плотной квадратной матрицы потребовало бы $O(n^3)$ операций.

Одномерная линейная регрессия

Регрессия – важный статистический инструмент установления связи между величинами на основе имеющихся данных [360]. Рассмотрим набор данных, изображенный на рис. 1.9. Точки, обозначенные красными крестиками, получены прибавлением гауссова белого шума к черной прямой, как показано в листинге 1.4. Мы предполагаем, что между данными имеется линейная связь, как в (1.17), и используем псевдообратную матрицу для нахождения решения с наименьшей среднеквадратической ошибкой – синей штриховой прямой с угловым коэффициентом x, – как показано в листинге 1.5.

$$\begin{bmatrix} | \\ \mathbf{b} \\ | \end{bmatrix} = \begin{bmatrix} | \\ \mathbf{a} \\ | \end{bmatrix} x = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^* x. \tag{1.22a}$$

$$\Rightarrow x = \tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}^*\mathbf{b}. \tag{1.22b}$$

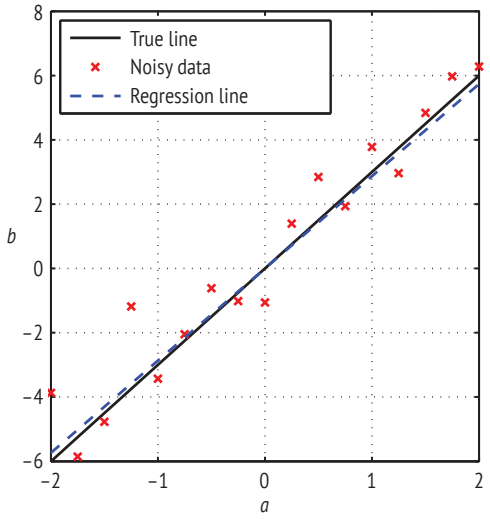

Рис. 1.9 ❖ Линейная регрессия с зашумленными данными

В (1.22b) $\tilde{\mathbf{\Sigma}} = ||\mathbf{a}||_2, \tilde{\mathbf{V}} = 1, \tilde{\mathbf{U}} = \mathbf{a}/||\mathbf{a}||_2$. Умножение на левую псевдообратную матрицу дает

$$x = \frac{\mathbf{a}^*\mathbf{b}}{||\mathbf{a}||_2^2}. \tag{1.23}$$

Это имеет физический смысл, если интерпретировать x как значение, которое дает наилучшее отображение нашего вектора **a** в вектор **b**. Такое наилучшее значение x единственно и получается в результате вычисления скалярного произведения **b** с нормированным вектором в направлении **a**. И мы добавляем второй нормировочный коэффициент $||\mathbf{a}||_2$, потому что **a** в формуле (1.22a) не нормирован.

Отметим, что если в (1.22) взять векторы-строки вместо векторов-столбцов, то будут происходить странные вещи. Кроме того, если величина шума велика по сравнению с угловым коэффициентом x, то в точности псевдообратной матрицы произойдет фазовый переход, связанный с жесткой пороговой обработкой, описанной ниже.

Листинг 1.4 ❖ Генерирование зашумленных данных для рис. 1.9

```
x = 3;                          % истинный угловой коэффициент
a = [-2:.25:2]';
b = a*x + 1*randn(size(a));     % добавить шум
plot(a,x*a,'k')                 % истинная связь
hold on, plot(a,b,'rx')         % зашумленные измерения
```

Листинг 1.5 ❖ Вычисление аппроксимации методом наименьших квадратов для рис. 1.9

```
[U,S,V] = svd(a,'econ');
xtilde = V*inv(S)*U'*b;         % аппроксимации методом наименьших квадратов
plot(a,xtilde*a,'b--')          % нарисовать график
```

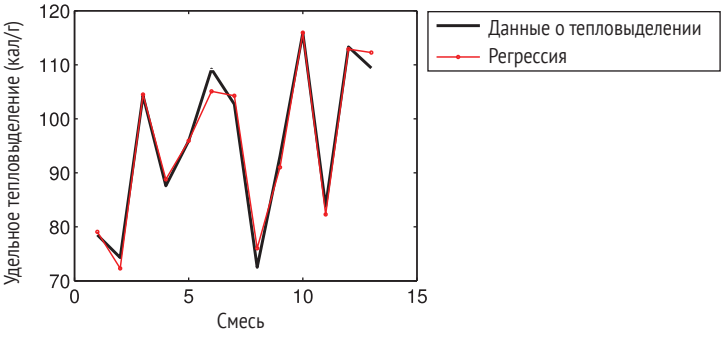

Рис. 1.10 ❖ Данные о тепловыделении для четырехкомпонентных цементных смесей

В статистике описанная выше процедура называется *линейной регрессией*. В MATLAB есть команда **regress**, а также команда **pinv**, которую тоже можно использовать.

Листинг 1.6 ❖ Альтернативные варианты метода наименьших квадратов в MATLAB

```
xtilde1 = V*inv(S)*U'*b
xtilde2 = pinv(a)*b
xtilde3 = regress(b,a)
```

Полилинейная регрессия

Пример 1: данные о тепловыделении цементных смесей

Начнем с простого встроенного в MATLAB набора данных, который описывает тепловыделение различных цементных смесей, составленных из четырех компонент. В этой задачи мы решаем систему (1.17) с $\mathbf{A} \in \mathbb{R}^{13\times 4}$, поскольку компонент четыре, а измерения производятся для 13 разных смесей. Наша цель – найти веса **x**, описывающие связь между четырьмя ингредиентами и тепловыделением. В листинге 1.7 показано, как найти решение с минимальной ошибкой. Исследуются варианты с использованием функций **regress** и **pinv**.

Листинг 1.7 ❖ Полилинейная регрессия для данных о тепловыделении цементных смесей

```
load hald;                  % загрузить набор данных Portlant Cement
A = ingredients;
b = heat;

[U,S,V] = svd(A,'econ');
x = V*inv(S)*U'*b;          % решить систему Ax=b методом SVD

plot(b,'k'); hold on        % построить график
plot(A*x,'r-o',);           % аппроксимация графика

x = regress(b,A);           % вариант 1 (regress)
x = pinv(A)*b;              % вариант 2 (pinv)
```

Пример 2: данные о недвижимости в Бостоне

В этом примере мы исследуем более крупный набор данных, чтобы определить, какие факторы лучше всего предсказывают цены на бостонском рынке недвижимости [234]. Этот набор можно скачать из репозитория машинного обучения UCI Machine Learning Repository [24].

Существует 13 атрибутов, коррелирующих с ценой дома, например: индекс преступности на душу населения и ставка налога на имущество. Мы построили регрессионную модель влияния этих факторов на цену и на рис. 1.11 на-

рисовали график, на котором лучшее предсказание цены сопоставляется с истинной стоимостью дома. На рис. 1.12 показаны коэффициенты регрессии. Хотя стоимость дома предсказана неточно, тренды хорошо согласуются. Часто бывает, что простая линейная аппроксимация плохо улавливает выбросы с наибольшими значениями; такую ситуацию мы наблюдаем и в этом примере.

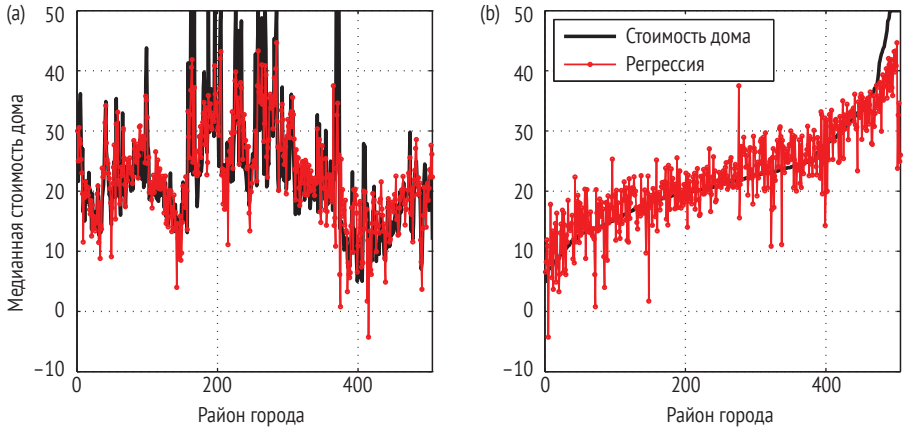

Рис. 1.11 ❖ Полилинейная регрессия цен домов с различными факторами: (a) неотсортированные данные; (b) данные отсортированы по стоимости дома

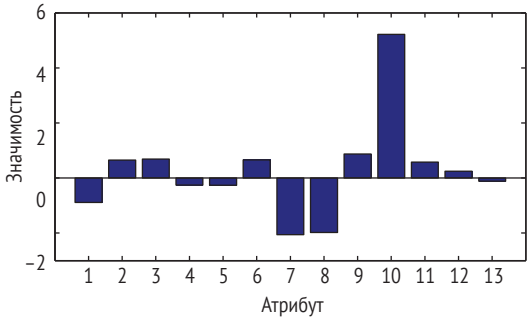

Рис. 1.12 ❖ Значимость различных атрибутов в регрессии

Этот набор данных содержит цены и атрибуты для 506 домов, так что матрица атрибутов имеет размер 506×13. Важно дополнить эту матрицу столбцом, состоящим из единиц, чтобы учесть возможность ненулевого постоянного смещения в формуле регрессии. В случае одномерной регрессии это соответствует свободному члену.

Листинг 1.8 ❖ Полилинейная регрессия для данных о стоимости домов в Бостоне

```
load housing.data

b = housing(:,14);                % стоимости домов в тысячах долларов
A = housing(:,1:13);              % прочие факторы
```

```
A = [A ones(size(A,1),1)];          % дополнить единичными свободными членами

x = regress(b,A);
plot(b,'k-o');
hold on, plot(A*x,'r-o');

[b sortind] = sort(housing(:,14));   % отсортированные стоимости
plot(b,'k-o')
hold on, plot(A(sortind,:)*x,'r-o')
```

Предостережение

Вообще говоря, матрица $\mathbf{U}$, столбцами которой являются левые сингулярные векторы $\mathbf{X}$, – унитарная квадратная матрица. Поэтому $\mathbf{U}^*\mathbf{U} = \mathbf{U}\mathbf{U}^* = \mathbf{I}_{n\times n}$. Однако чтобы вычислить псевдообратную матрицу для $\mathbf{X}$, мы должны вычислить $\mathbf{X}^\dagger = \tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}^*$, потому что только $\tilde{\mathbf{\Sigma}}$ обратима (если все сингулярные значения отличны от нуля), тогда как $\mathbf{\Sigma}$ в общем случае необратима (она даже в общем случае не является квадратной).

До сих пор мы предполагали, что $\mathbf{X} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*$ – точное SVD, так что ранг r включает все ненулевые сингулярные значения. Это гарантирует, что матрица $\tilde{\mathbf{\Sigma}}$ обратима.

Трудности начинаются при работе с усеченным базисом, образованным левыми сингулярными векторами $\tilde{\mathbf{U}}$. По-прежнему верно, что $\tilde{\mathbf{U}}^*\tilde{\mathbf{U}} = \mathbf{I}_{r\times r}$, где r – ранг $\mathbf{X}$. Однако $\tilde{\mathbf{U}}\tilde{\mathbf{U}}^* \neq \mathbf{I}_{n\times n}$, что легко проверить численно на простом примере. Предположение о том, что $\tilde{\mathbf{U}}\tilde{\mathbf{U}}^*$ равно единичной матрице, – одно из самых типичных заблуждений при использовании SVD[1].

```
>> tol = 1.e-16;
>> [U,S,V] = svd(X,'econ')
>> r = max(find(diag(S)>max(S(:))*tol));
>> invX = V(:,1:r)*S(1:r,1:r)*U(:,1:r)'; % только приближенно
```

1.5. Метод главных компонент (PCA)

Метод главных компонент (PCA) – одно из основных применений SVD, он позволяет, ориентируясь только на данные, построить систему координат для представления коррелированных данных высокой размерности. При этом используются корреляционные матрицы, описанные в разделе 1.3. Важно, что в методе PCA, перед тем как применять SVD, производится предварительная обработка данных: вычитание среднего и приведение к единичной дисперсии. Геометрия результирующей системы координат определяется главными компонентами (principal component – PC), которые не коррелируют между собой (ортогональны), но имеют максимальную корреляцию с результатами измерений. Теория была разработана в 1901 году Пирсоном

[1] Его не избежали и авторы, по ошибке включив это неверное тождество в ранний вариант работы [96].

[418] и независимо в 1930-х годах Хотеллингом [256, 257]. Работа Jolliffe [268] содержит хорошее справочное пособие.

Обычно в одном эксперименте производится несколько измерений, которые образуют вектор-строку. Эти измерения могут быть свойствами наблюдаемой величины, например демографическими признаками одного человека. Выполняется несколько экспериментов, и все векторы-строки собираются в большую матрицу **X**. Если говорить о демографии, то экспериментальные данные могут быть собраны путем опроса. Заметим, что такое соглашение – **X** состоит из строк признаков – отличается от соглашения, принятого в других частях этой главы, где отдельные «снимки» признаков расположены по столбцам. Но мы решили в этом разделе не вступать в противоречие с литературой по PCA. Матрица по-прежнему имеет размер $n \times m$ и может содержать больше строк, чем столбцов, или наоборот.

Вычисление

Теперь вычислим среднее по строкам $\bar{\mathbf{x}}$ (т. е. среднее всех строк) и вычтем его из **X**. Среднее $\bar{\mathbf{x}}$ определяется по формуле

$$\bar{\mathbf{x}}_j = \frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_{ij}, \tag{1.24}$$

а средняя матрица – по формуле

$$\bar{\mathbf{X}} = \begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix}\bar{\mathbf{x}}. \tag{1.25}$$

Вычитая $\bar{\mathbf{X}}$ из **X**, получаем матрицу **B** с нулевым средним:

$$\mathbf{B} = \mathbf{X} - \bar{\mathbf{X}}. \tag{1.26}$$

Ковариационная матрица строк **B** определяется формулой

$$\mathbf{C} = \frac{1}{n-1}\mathbf{B}^{*}\mathbf{B}. \tag{1.27}$$

Первая главная компонента $\mathbf{u}_1$ определяется так:

$$\mathbf{u}_1 = \underset{||\mathbf{u}_1||=1}{\operatorname{argmax}}\,\mathbf{u}_1^{*}\mathbf{B}^{*}\mathbf{B}\mathbf{u}_1. \tag{1.28}$$

Это собственный вектор $\mathbf{B}^{*}\mathbf{B}$, соответствующий наибольшему собственному значению. Теперь понятно, что $\mathbf{u}_1$ – левый сингулярный вектор **B**, соответствующий наибольшему сингулярному значению.

Главные компоненты можно получить, вычислив спектральное разложение **C**:

$$\mathbf{CV} = \mathbf{VD}, \tag{1.29}$$

которое гарантированно существует, потому что матрица **C** эрмитова.

Команда pca

В MATLAB имеются дополнительные команды **pca** и **princomp** (основанная на **pca**) для метода главных компонент:

```
>> [V,score,s2] = pca(X);
```

Матрица **V** эквивалентна матрице **V** из сингулярного разложения **X** с точностью до изменения знака столбцов. Вектор **s2** содержит собственные значения ковариационной матрицы **X**, которые называются также дисперсиями главных компонент; это квадраты сингулярных значений. Переменная **score** содержит координаты каждой строки **B** (данные после вычитания среднего) в направлениях главных компонент. Вообще говоря, мы часто предпочитаем использовать команду **svd** в сочетании с различными шагами постобработки, описанными ранее в этом разделе.

Пример: данные с гауссовым шумом

Рассмотрим зашумленное облако данных на рис. 1.13 (a), построенном программой в листинге 1.9. Данные генерируются путем выборки 10 000 векторов из двумерного нормального распределения с нулевым средним и единичной дисперсией. Эти векторы затем масштабируются в направлениях x и y с коэффициентами в табл. 1.1 и поворачиваются на угол $\pi/3$. Наконец, все облако данных подвергается параллельному переносу, так что его центр располагается в точке $\mathbf{x}_C = [2 \;\; 1]^T$.

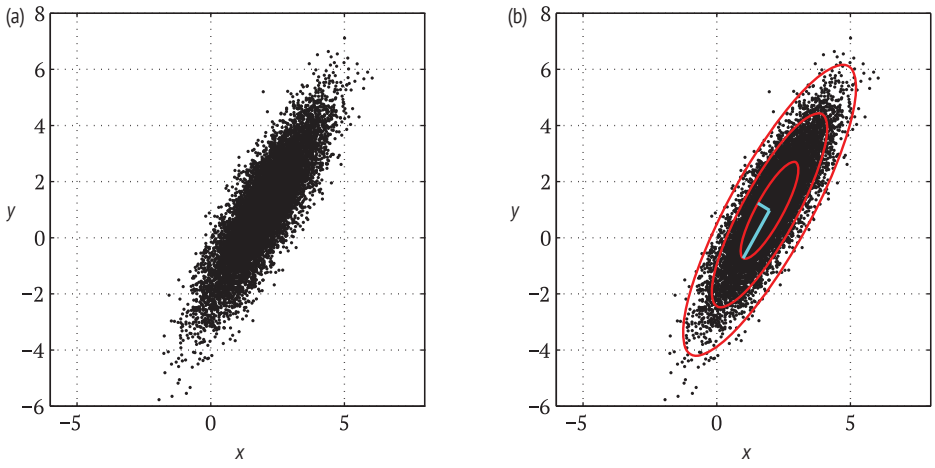

Рис. 1.13 ❖ Главные компоненты улавливают дисперсию гауссовых данных после вычитания среднего (a). На рисунке (b) показаны эллипсы для первых трех стандартных отклонений (красным цветом) и два левых сингулярных вектора, умноженных на сингулярные значения ($\sigma_1 u_1 + x_C$ и $\sigma_2 u_2 + x_C$, голубым цветом)

Таблица 1.1. Стандартное отклонение данных и нормированные сингулярные значения

	σ_1	σ_2
Данные	2	0.5
SVD	1.974	0.503

Программа в листинге 1.10 выполняет PCA и рисует на графике доверительные интервалы при нескольких стандартных отклонениях, как показано на рис. 1.13 (b). В табл. 1.1 также приведены сингулярные значения, соответствующие коэффициентам масштабирования. Матрица **U** из сингулярного разложения очень близка к матрице поворота с точностью до знака чисел в одном столбце:

$$\mathbf{R}_{\pi/3} = \begin{bmatrix} 0.5 & -0.8660 \\ 0.8660 & 0.5 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} -0.4998 & -0.8662 \\ -0.8662 & 0.4998 \end{bmatrix}.$$

Листинг 1.9 ❖ Генерирование зашумленного облака данных для иллюстрации PCA

```
xC = [2; 1;];                          % центр данных (среднее)
sig = [2; .5;];                        % главные оси

theta = pi/3;                          % повернуть облако на угол pi/3
R = [cos(theta) -sin(theta);           % матрица поворота
sin(theta) cos(theta)];

nPoints = 10000;                       % создать 10 000 точек
X = R*diag(sig)*randn(2,nPoints) + diag(xC)*ones(2,nPoints);
scatter(X(1,:),X(2,:),'k.','LineWidth',2)
```

Листинг 1.10 ❖ Вычислить главные компоненты и нарисовать доверительные интервалы

```
Xavg = mean(X,2);                      % вычислить среднее
B = X - Xavg*ones(1,nPoints);          % данные после вычитания среднего
[U,S,V] = svd(B/sqrt(nPoints),'econ'); % PCA методом SVD
scatter(X(1,:),X(2,:),'k.','LineWidth',2)  % нанести данные на график

theta = (0:.01:1)*2*pi;
Xstd = U*S*[cos(theta); sin(theta)];       % доверит. инт. для 1-го стд. откл.

plot(Xavg(1)+Xstd(1,:),Xavg(2) + Xstd(2,:),'r-')
plot(Xavg(1)+2*Xstd(1,:),Xavg(2) + 2*Xstd(2,:),'r-')
plot(Xavg(1)+3*Xstd(1,:),Xavg(2) + 3*Xstd(2,:),'r-')
```

Наконец, главные компоненты можно вычислить также командой **pca**:

```
>> [V,score,s2] = pca(X);
>> norm(V*score' - B)

ans =
   2.2878e-13
```

Пример: данные о раке яичников

Набор данных о раке яичников, встроенный в MATLAB, дает более реалистичный пример преимуществ PCA. Этот пример содержит генетические данные о 216 пациентах, из которых 121 болел раком яичников, а 95 – нет. Для каждого пациента имеется вектор данных, содержащий экспрессию 4000 генов. При обработке этого набора данных исследователь сталкивается с рядом трудностей, из которых основная – высокая размерность данных. Однако, как показано на рис. 1.14, значительная дисперсия улавливается уже несколькими первыми PCA-модами. Иными словами, данные о генах сильно коррелированы, так что у многих пациентов экспрессия генов демонстрирует значительное перекрытие. Возможность визуализировать паттерны и корреляции в данных высокой размерности – важная причина для использования PCA, и этот метод широко применяется для нахождения паттернов в многомерных биологических и генетических данных [448].

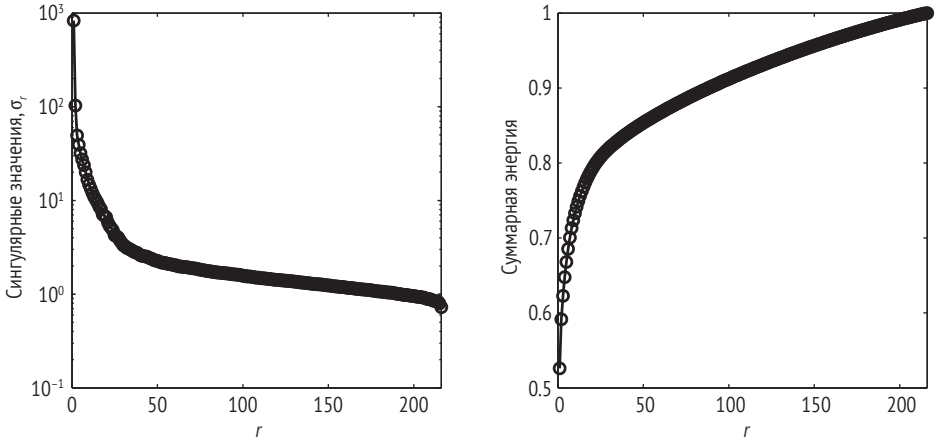

Рис. 1.14 ❖ Сингулярные значения для данных о раке яичников

Еще важнее тот факт, что пациенты с раком яичников, похоже, образуют кластер, отдельный от здоровых пациентов, если нарисовать их в пространстве, натянутом на первые три PCA-моды. Это видно на рис. 1.15, построенном программой из листинга 1.11. Подобная кластеризация данных в пространстве PCA по категориям – важнейший элемент машинного обучения и распознавания образов. Например, в разделе 1.6 мы увидим, что изображения лиц разных людей образуют кластеры в пространстве PCA. Использование этих кластеров будет подробно исследовано в главе 5.

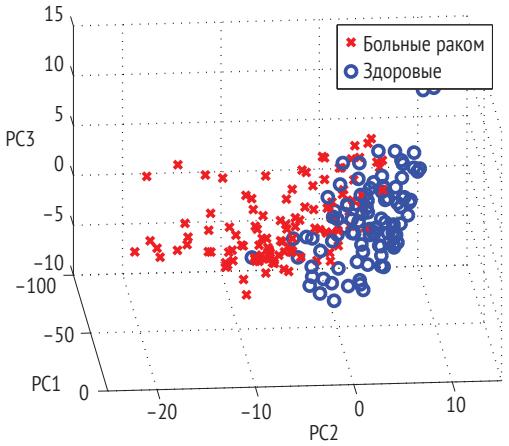

Рис. 1.15 ❖ Выборка распадается на два кластера по трем первым главным компонентам: здоровые пациенты и болеющие раком

Листинг 1.11 ❖ Вычисление PCA для данных о раке яичников

```
load ovariancancer;  % Загрузить данные о раке яичников
[U,S,V] = svd(obs,'econ');
for i=1:size(obs,1)
    x = V(:,1)'*obs(i,:)';
    y = V(:,2)'*obs(i,:)';
    z = V(:,3)'*obs(i,:)';
    if(grp{i}=='Cancer')
        plot3(x,y,z,'rx','LineWidth',2);
    else
        plot3(x,y,z,'bo','LineWidth',2);
    end
end
```

1.6. Пример: «собственные лица»

Одна из самых убедительных демонстраций комбинации SVD+PCA – так называемый алгоритм «собственных лиц» (eigenfaces). В этом случае PCA (т. е. SVD для данных после вычитания среднего) применяется к большой библиотеке изображений лиц, чтобы выделить доминирующие корреляции между изображениями. Результатом разложения является множество *собственных лиц*, определяющее новую систему координат. Изображения можно представить в этих координатах, вычислив скалярное произведение с каждой из главных компонент. В главе 5 будет показано, что изображения одного и того же человека кластеризуются в пространстве собственных лиц, поэтому такое преобразование полезно для распознавания и классификации лиц [510, 48]. Задача о собственных лицах впервые была изучена Сировичем и Кирби в 1987 году [491] и дополнительно в работе [291]. Ее применение к автома-

тизированному распознаванию лиц представлено в работе Turk and Pentland [537] в 1991 году.

Мы продемонстрируем этот алгоритм, используя расширенную базу данных. В базе данных о лицах Йельского университета (Extended Yale Face Database B) [203], которая состоит из кадрированных и выровненных изображений [327], 38 человек (28 из расширенной и 10 из оригинальной базы данных) в 9 положениях и при 64 условиях освещения[1]. Каждое изображение имеет размер 192 пикселя по вертикали на 168 пикселей по горизонтали. В отличие от примера из раздела 1.2, все изображения в библиотеке представлены в виде длинного вектора-столбца, содержащего $192 \times 168 = 32\,256$ элементов. Мы возьмем первые 36 человек (левая часть рис. 1.16) в качестве обучающих данных, а двух зарезервируем для тестирования. В правой части рисунка показаны все 64 изображения одного человека. Для загрузки и построения графиков была использована программа в листинге 1.12.

Рис. 1.16 ❖ (Слева) По одному изображению для каждого человека в Йельской базе данных; (справа) все изображения для одного человека. Левый рисунок сгенерирован программой в листинге 1.12

Листинг 1.12 ❖ Рисование изображения для каждого человека в Йельской базе данных (рис. 1.16 (а))

```
load ../DATA/allFaces.mat

allPersons = zeros(n*6,m*6);   % Создать массив, вмещающий все лица
count = 1;
for i=1:6     % сетка лиц 6×6
    for j=1:6
        allPersons(1+(i-1)*n:i*n,1+(j-1)*m:j*m) ...
             =reshape(faces(:,1+sum(nfaces(1:count-1))),n,m);
```

[1] Базу данных можно скачать по адресу http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/ExtYaleB.html.

```
        count = count + 1;
    end
end
imagesc(allPersons), colormap gray
```

Как уже отмечалось, каждое изображение представлено длинным вектором-столбцом. Затем вычисляется среднее лицо и результат вычитается из каждого вектора. Векторы, полученные после вычитания среднего, собираются в матрицу **X**, как показано на рис. 1.17. Для этой матрицы вычисляется сингулярное разложение, т. е. применяется метод PCA. Столбцами **U** являются собственные лица, которые можно преобразовать обратно в изображения 192×168. Соответствующий код приведен в листинге 1.13.

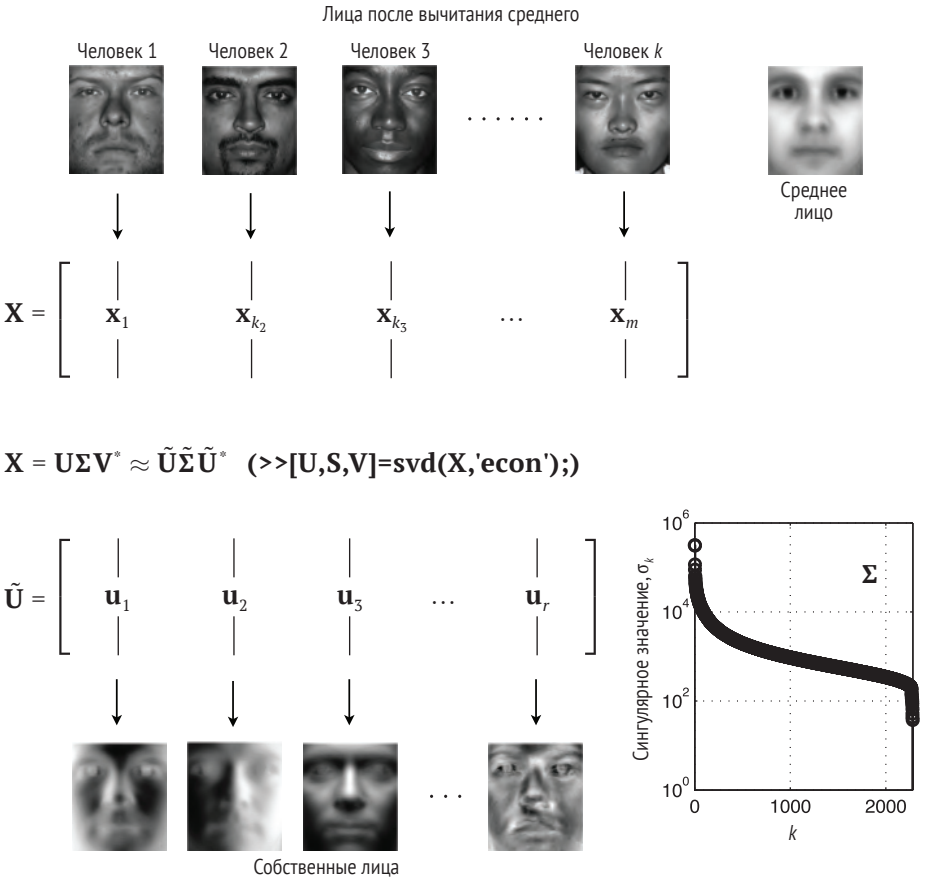

Рис. 1.17 ❖ Схема процедуры получения собственных лиц по лицам из библиотеки

Листинг 1.13 ❖ Вычисление собственных лиц по данным после вычитания среднего

```
% Мы используем первые 36 лиц в качестве обучающих данных
trainingFaces = faces(:,1:sum(nfaces(1:36)));
avgFace = mean(trainingFaces,2);
```

```
% Вычислить собственные лица по обучающим данным после вычитания среднего
X = trainingFaces-avgFace*ones(1,size(trainingFaces,2));
[U,S,V] = svd(X,'econ');

imagesc(reshape(avgFace,n,m)) % Нарисовать среднее лицо
imagesc(reshape(U(:,1),n,m))  % Нарисовать первое собственное лицо
```

С помощью матрицы $\tilde{\mathbf{U}}$, вычисленной этой программой, попытаемся приближенно представить изображение, которого не было среди обучающих данных. Напомним, что мы зарезервировали два лица (37-е и 38-е), теперь возьмем одно из них в качестве тестового изображения $\mathbf{x}_{\text{test}}$. Чтобы узнать, насколько хорошо SVD-базис ранга r аппроксимирует это изображение, воспользуемся следующей проекцией:

$$\tilde{\mathbf{x}}_{\text{test}} = \tilde{\mathbf{U}}\tilde{\mathbf{U}}^{*}\mathbf{x}_{\text{test}}.$$

На рис. 1.18 показана аппроксимация собственного лица для разных значений r, вычисленная программой в листинге 1.14. При $r \leq 200$ аппроксимация не очень хороша, но при $r > 400$ представление тестового изображения становится вполне приемлемым. Интересно отметить, что пространство собственных лиц полезно не только для представления человеческих лиц, но и для аппроксимации собаки (рис. 1.19) или чашки капучино (рис. 1.20). Так получается, потому что 1600 собственных лиц образуют базис большого подпространства 32 256-мерного пространства изображений, соответствующего широкому множеству гладких нелокализованных пространственных признаков: щек, лбов, ртов и т. д.

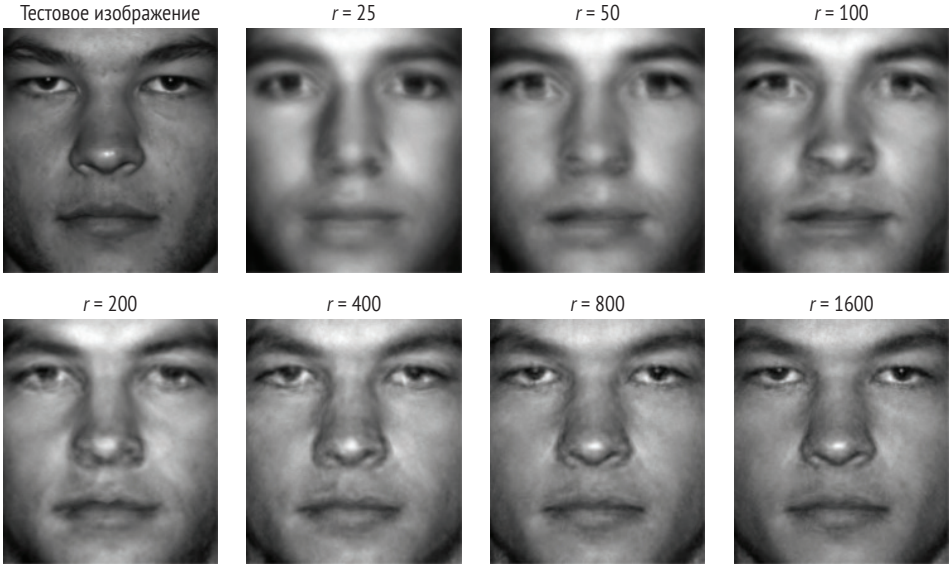

Рис. 1.18 ❖ Приближенное представление тестового изображения с использованием базиса собственных лиц разного порядка r. Тестовое изображение не входит в обучающий набор

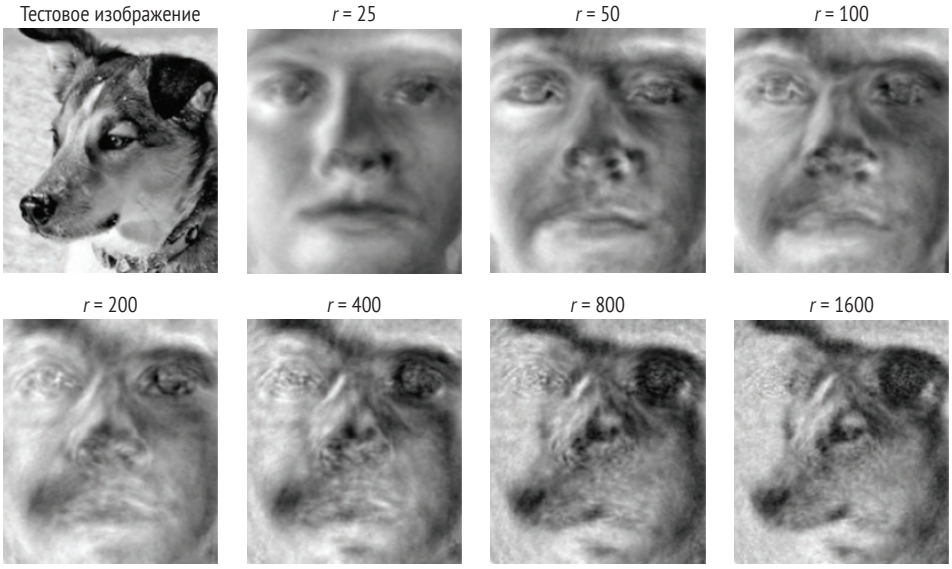

Рис. 1.19 ❖ Приближенное представление изображения собаки с помощью собственных лиц

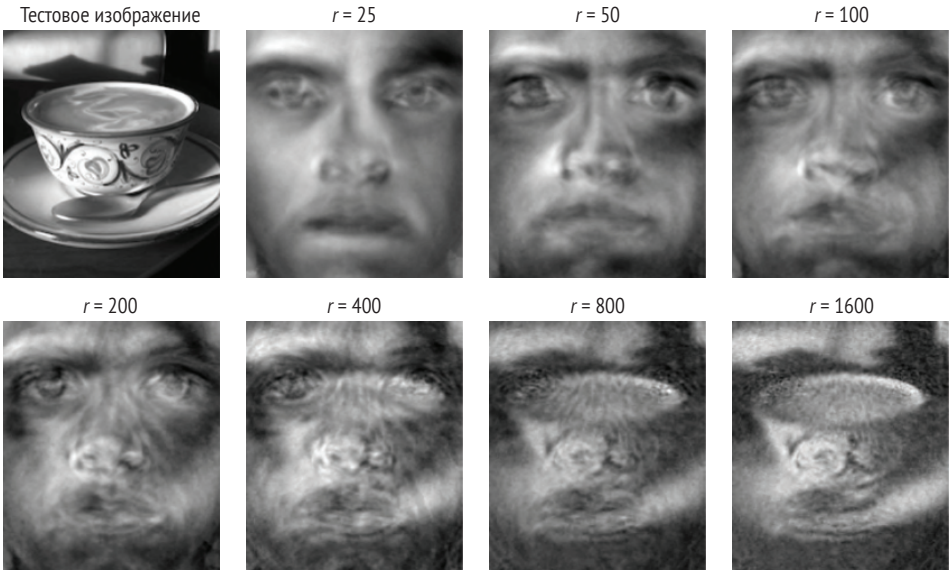

Рис. 1.20 ❖ Приближенное представление изображения чашки капучино с помощью собственных лиц

Листинг 1.14 ❖ Аппроксимация тестового изображения, не входящего в состав обучающих данных

```
testFaceMS = testFace - avgFace;
for r=[25 50 100 200 400 800 1600]
    reconFace = avgFace + (U(:,1:r)*(U(:,1:r)'*testFaceMS));
    imagesc(reshape(reconFace,n,m))
end
```

Продолжим исследование применения собственных лиц в качестве системы координат, определяющей пространство собственных лиц. Спроецировав изображение $\mathbf{x}$ на первые r PCA-мод, мы получим множество координат в этом пространстве: $\tilde{\mathbf{x}} = \tilde{\mathbf{U}}^{*}\mathbf{x}$. Некоторые главные компоненты улавливают типичные признаки, общие для всех человеческих лиц, тогда как другие более полезны для различения людей. Возможно, также существуют главные компоненты, улавливающие различия, вызванные углом освещения. На рис. 1.21, построенном программой в листинге 1.15, показаны координаты всех 64 изображений двух человек, спроецированные на 5-ю и 6-ю главные компоненты. Видно, что люди четко разделяются по этим координатам. Это наблюдение лежит в основе распознавания и классификации изображений, которое обсуждается в главе 5.

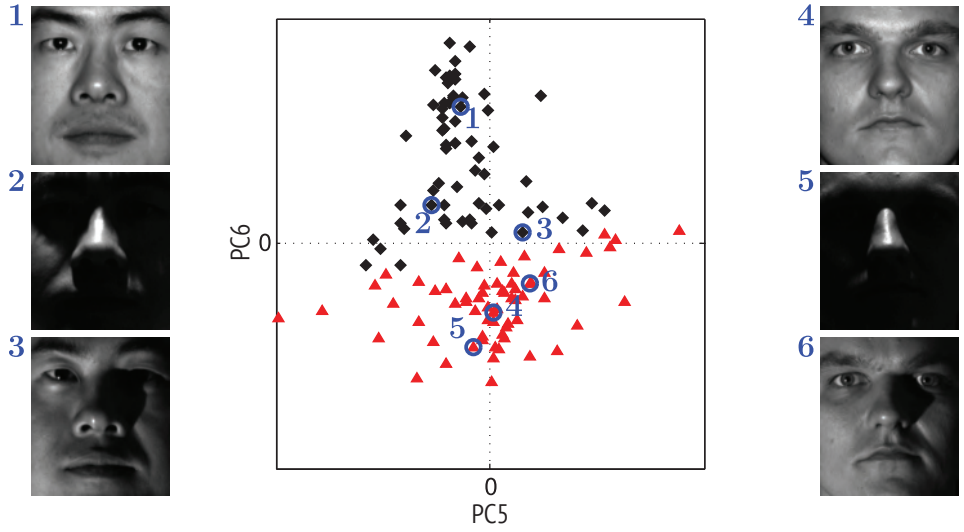

Рис. 1.21 ❖ Проекция всех изображений двух человек на 5-ю и 6-ю PCA-моды. Проекции изображений первого человека обозначены черными ромбами, а второго – красными треугольниками. По три примера для каждого человека обведены синей окружностью, и показаны соответствующие им изображения

Листинг 1.15 ❖ Проецирование изображений двух человек на 5-ю и 6-ю PCA-моды иллюстрирует потенциал автоматизированной классификации

```
P1num = 2; % человек под номером 2
P2num = 7; % человек под номером 7

P1 = faces(:,1+sum(nfaces(1:P1num-1)):sum(nfaces(1:P1num)));
P2 = faces(:,1+sum(nfaces(1:P2num-1)):sum(nfaces(1:P2num)));

P1 = P1 - avgFace*ones(1,size(P1,2));
P2 = P2 - avgFace*ones(1,size(P2,2));

PCAmodes = [5 6]; % спроецировать на PCA-моды 5 и 6
PCACoordsP1 = U(:,PCAmodes)'*P1;
PCACoordsP2 = U(:,PCAmodes)'*P2;

plot(PCACoordsP1(1,:),PCACoordsP1(2,:),'kd')
plot(PCACoordsP2(1,:),PCACoordsP2(2,:),'r^')
```

1.7. Отсечение и выравнивание

Решение о том, сколько оставлять сингулярных значений, т. е. где проходит порог отсечения, – одно из самых важных и неочевидных при использовании SVD. Существует много факторов, в т. ч. желаемый ранг системы, величина шума и распределение сингулярных значений. Иногда порогом отсечения является ранг r, при котором улавливается заранее заданная часть дисперсии или энергии исходных данных, скажем 90 % или 99 %. Хотя эта методика грубовата, она используется достаточно часто. Другой подход основан на идентификации «локтей» и «коленей» в распределении сингулярных значений, которые означают переход от значений, представляющих важные паттерны, к тем, что представляют шум. При этом порог отсечения задается жестко: сингулярные значения, большие τ, оставляются, остальные отбрасываются. В недавней работе Gavish and Donoho [200] предложен оптимальный порог отсечения при некоторых условиях, т. е. теоретически обоснованный подход к получению низкоранговых аппроксимаций матриц методом SVD.

На ранг SVD-аппроксимации также оказывает важное влияние выравнивание данных. SVD принципиально зависит от разнесения переменных по строкам и столбцам матрицы данных. Во многих ситуациях, например при анализе блуждающих волн или невыровненных данных, это предположение нарушается, что приводит к неоправданному завышению ранга.

Оптимальный жесткий порог отсечения

В недавнем прорывном теоретическом исследовании определен *оптимальный* жесткий порог τ отсечения сингулярных значений в предположении, что матрица действительно имеет низкий ранг, но он замаскирован гауссовым белым шумом [200]. Эта работа основана на обширной литературе, посвя-

щенной различным методам задания жестких и мягких порогов отсечения сингулярных значений. В данном разделе мы кратко опишем основные результаты и продемонстрируем задание порогов на различных примерах. Дополнительные сведения см. в тексте оригинальной статьи [200].

Прежде всего предположим, что матрица данных $\mathbf{X}$ является суммой истинной матрицы низкого или приближенно низкого ранга $\mathbf{X}_{\text{true}}$ и матрицы шума $\mathbf{X}_{\text{noise}}$:

$$\mathbf{X} = \mathbf{X}_{\text{true}} + \gamma \mathbf{X}_{\text{noise}}. \tag{1.30}$$

Предполагается, что элементы $\mathbf{X}_{\text{noise}}$ – независимые и одинаково распределенные гауссовы случайные величины с нулевым средним и единичной дисперсией. Величина шума характеризуется параметром γ, что отличается от обозначений, принятых в [200][1].

Если величина шума γ известна, то оптимальный жесткий порог τ можно вычислить в замкнутой форме.

1. Если $\mathbf{X} \in \mathbb{R}^{n \times n}$ – квадратная матрица, то

$$\tau = \left(4/\sqrt{3}\right)\sqrt{n}\gamma. \tag{1.31}$$

2. Если $\mathbf{X} \in \mathbb{R}^{n \times m}$ – прямоугольная матрица и $m \ll n$, то постоянная $4/\sqrt{3}$ заменяется функцией от отношения сторон $\beta = m/n$:

$$\tau = \lambda(\beta)\sqrt{n}y; \tag{1.32}$$

$$\lambda(\beta) = \left(2(\beta+1) + \frac{8\beta}{(\beta+1) + (\beta^2 + 14\beta + 1)^{1/2}}\right)^{1/2}. \tag{1.33}$$

Заметим, что это выражение сводится к (1.31) при $\beta = 1$. Если $n \ll m$, то $\beta = n/m$.

Если величина шума γ неизвестна, что более типично для реальных приложений, то ее можно оценить и масштабировать распределение сингулярных значений, умножив его на σ_{med} – *медианное* сингулярное значение. В этом случае выражения τ в замкнутой форме не существует, и его следует аппроксимировать численно.

3. Для неизвестного шума γ и прямоугольной матрицы $\mathbf{X} \in \mathbb{R}^{n \times m}$ оптимальный жесткий порог равен

$$\tau = \omega(\beta)\sigma_{\text{med}}. \tag{1.34}$$

Здесь $\omega(\beta) = \lambda(\beta)/\mu_\beta$, где μ_β – решение следующей задачи

$$\int_{(1-\beta)^2}^{\mu_\beta} \frac{\left[\left((1+\sqrt{\beta})^2 - t\right)\left(t - (1-\sqrt{\beta})^2\right)\right]^{1/2}}{2\pi t} dt = \frac{1}{2}.$$

[1] В работе [200] σ обозначает стандартное отклонение, а y_k – k-е сингулярное значение.

Решения необходимо искать численно. По счастью, к работе [200] прилагается код на MATLAB[1] [151] для нахождения приближенного значения μ_β.

Этот новый метод задания оптимального жесткого порога работает на удивление хорошо, как показывают примеры ниже.

Пример 1: модельная задача

В первом примере, показанном на рис. 1.22, мы искусственно строим матрицу ранга 2 (листинг 1.16) и прибавляем к сигналу гауссов белый шум (листинг 1.17). Затем очищаем сигнал от шума и понижаем размерность матрицы, задавая порог по формуле (1.31) (листинг 1.18) и используя для отсечения 90%-ный уровень энергии (листинг 1.19). Видно, что жесткий порог отфильтровывает шум более эффективно. Нанеся сингулярные значения на график, показанный на рис. 1.23 (листинг 1.20), мы увидим, что выше порога расположены два значения.

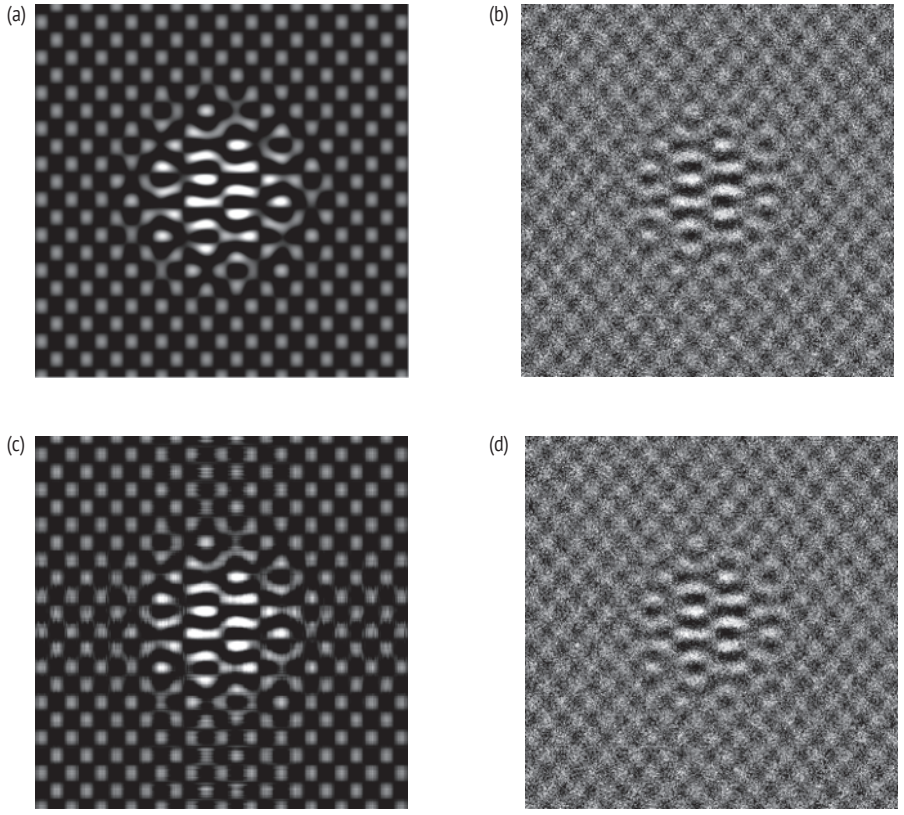

Рис. 1.22 ❖ (a) Исходная матрица ранга 2; (b) зашумленная матрица; (c) очищенная матрица после применения оптимального жесткого порога $(4/\sqrt{3})\sqrt{n}\sigma$; (d) отсечение на основе 90%-ной энергии

[1] http://purl.stanford.edu/vg705qn9070.

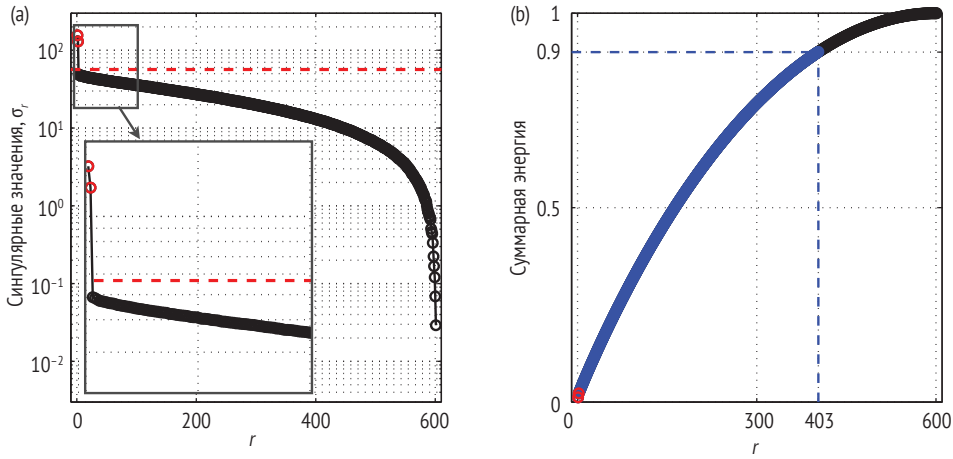

Рис. 1.23 ❖ (a) Сингулярные значения σ_r; (b) суммарная энергия, сосредоточенная в первых r модах. Оптимальный жесткий порог $\tau = (4/\sqrt{3})\sqrt{n}\sigma$ показан красной штриховой линией. В этом случае $n = 600$ и $\sigma = 1$, так что оптимальный порог отсечения приблизительно равен $\tau = 56.6$

Листинг 1.16 ❖ Вычисление истинного сигнала низкого ранга (рис. 1.22 (a))

```
clear all, close all, clc

t = (-3:.01:3)';

Utrue = [cos(17*t).*exp(-t.^2) sin(11*t)];
Strue = [2 0; 0 .5];
Vtrue = [sin(5*t).*exp(-t.^2) cos(13*t)];
X = Utrue*Strue*Vtrue';
figure, imshow(X);
```

Листинг 1.17 ❖ Добавить к сигналу шум (рис. 1.22 (b))

```
sigma = 1;
Xnoisy = X+sigma*randn(size(X));
figure, imshow(Xnoisy);
```

Листинг 1.18 ❖ Применить оптимальный жесткий порог отсечения (рис. 1.22 (c))

```
[U,S,V] = svd(Xnoisy);

N = size(Xnoisy,1);
cutoff = (4/sqrt(3))*sqrt(N)*sigma; % жесткий порог
r = max(find(diag(S)>cutoff)); % оставить моды, для которых сигнал > cutoff
Xclean = U(:,1:r)*S(1:r,1:r)*V(:,1:r)';
figure, imshow(Xclean)
```

Листинг 1.19 ❖ Отсечение по критерию 90%-ной энергии (рис. 1.22 (d))

```
cdS = cumsum(diag(S))./sum(diag(S)); % суммарная энергия
r90 = min(find(cdS>0.90)); % найти r, улавливающий 90 % энергии
```

```
X90 = U(:,1:r90)*S(1:r90,1:r90)*V(:,1:r90)';
figure, imshow(X90)
```

Листинг 1.20 ❖ Построение графика сингулярных значений для жесткого порога (рис. 1.23)

```
semilogy(diag(S),'-ok','LineWidth',1.5), hold on, grid on
semilogy(diag(S(1:r,1:r)),'or','LineWidth',1.5)
```

Пример 2: собственные лица

Во втором примере мы вернемся к задаче о собственных лицах из раздела 1.6. Этот пример более типичен, потому что матрица данных $\mathbf{X}$ прямоугольная с отношением сторон $\beta = 3/4$, а величина шума неизвестна. Также не ясно, зашумлены ли данные. Тем не менее метод определяет порог τ такой, что столбцы $\mathbf{U}$, соответствующие большим модам, описывают значимые признаки лиц, а соответствующие меньшим модам содержат в основном шум (рис. 1.24).

Важность выравнивания данных

Обсудим типичные проблемы, возникающие при применении SVD к невыровненным данным. Приведенный ниже пример иллюстрирует один из главных недостатков, свойственных использованию SVD в задаче о понижении размерности и выделении взаимосвязанных признаков из данных. Рассмотрим матрицу, содержащую нули во всех элементах, кроме прямоугольного блока, заполненного единицами. Выглядит это как белый прямоугольник на черном фоне (рис. 1.25 (a)). Если стороны прямоугольника в точности параллельны осям *x* и *y* рисунка, то с применением SVD не возникает никаких проблем: имеется только одно ненулевое сингулярное значение σ_1 (рис. 1.25 (c)), которому соответствуют сингулярные векторы $\mathbf{u}_1$ и $\mathbf{v}_1$, определяющие ширину и высоту белого прямоугольника.

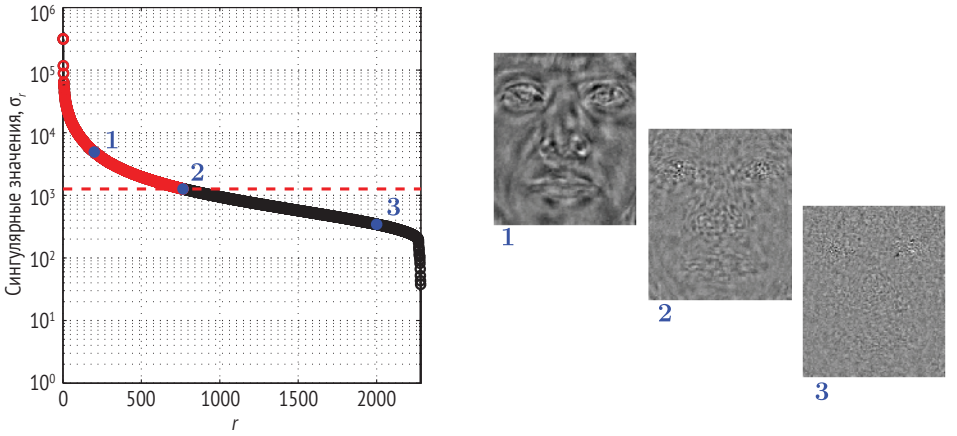

Рис. 1.24 ❖ Жесткий порог для примера с собственными лицами

Если повернуть внутренний прямоугольник, так что он перестанет быть выровненным с осями изображения, то в спектре появятся дополнительные ненулевые сингулярные значения (рис. 1.25 (b, d) и 1.26).

Листинг 1.21 ❖ Вычисление SVD для выровненного и повернутого квадрата (рис. 1.25)

```
n = 1000; % квадрат 1000×1000
X = zeros(n,n);
X(n/4:3*n/4,n/4:3:n/4) = 1;
imshow(X);

Y = imrotate(X,10,'bicubic'); % повернуть на 10 градусов
Y = Y - Y(1,1);
nY = size(Y,1);
startind = floor((nY-n)/2);
Xrot = Y(startind:startind+n-1, startind:startind+n-1);
imshow(Xrot);
[U,S,V] = svd(X);    % построить SVD выровненного квадрата
[U,S,V] = svd(Xrot); % построить SVD повернутого квадрата
semilogy(diag(S),'-ko')
semilogy(diag(S),'-ko')
```

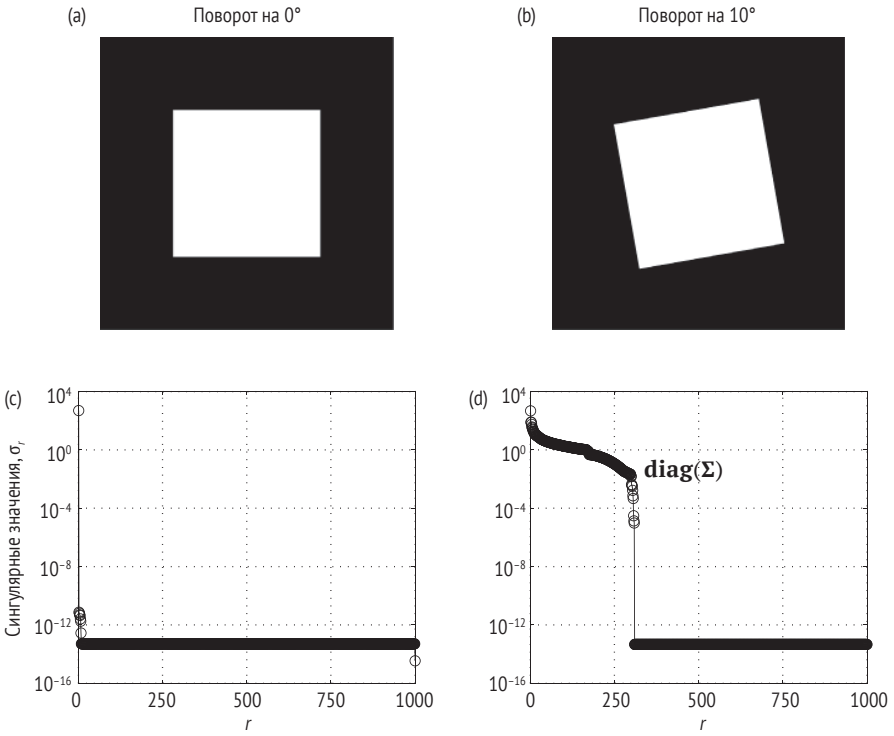

Рис. 1.25 ❖ (a) Матрица, содержащая нули всюду, кроме квадратного блока, содержащего единицы, и (c) ее SVD-спектр. Если повернуть блок на 10°, как в (b), то SVD-спектр (d) станет гораздо сложнее

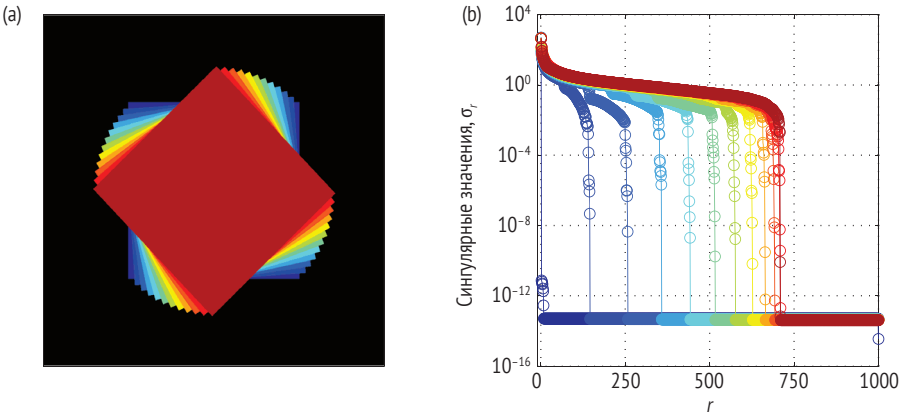

Рис. 1.26 ❖ (a) Матрица, содержащая нули всюду, кроме квадратного блока, содержащего единицы, и (b) ее SVD-спектр **diag(S)**

Причина проблемы заключается в том, что метод SVD по природе своей *геометрический*, т. е. зависит от системы координат, в которой представлены данные. Выше мы видели, что в общем случае сингулярное разложение инвариантно только относительно унитарных преобразований, т. е. преобразований, сохраняющих скалярное произведение. Это можно рассматривать и как достоинство, и как недостаток метода. С одной стороны, зависимость SVD от скалярного произведения очень важна для различных полезных геометрических интерпретаций. Кроме того, SVD присущи осмысленные единицы измерения и размерности. Но, с другой стороны, это делает SVD чувствительным к выравниванию данных. На самом деле ранг SVD резко возрастает при сдвиге, повороте или масштабировании объектов в столбцах, что сильно ограничивает его применимость к данным, не подвергнутым качественной предварительной обработке.

Например, в примере с собственными лицами использовалась библиотека изображений, которые были тщательно кадрированы, центрированы и выровнены по общему шаблону. Без такой предобработки способность к выделению признаков и кластеризации была бы совершенно неудовлетворительной.

Неспособность SVD улавливать сдвиг и поворот данных является серьезным ограничением. Например, SVD до сих пор остается предпочтительным методом для низкорангового разложения данных, полученных из дифференциальных уравнений в частных производных (УрЧП) (см. главы 11 и 12). Однако SVD по сути своей является методом управляемого данными разделения переменных, а это, как мы знаем, неверно для многих типов УрЧП, например для тех, которые описывают блуждающие волны. Поиск обобщенных разложений, которые сохраняли бы желательные свойства и были применимы к данным, содержащим разные виды симметрии, остается важной открытой проблемой.

Листинг 1.22 ❖ SVD для квадрата, повернутого на различные углы (рис. 1.26)

```
nAngles = 12; % перебрать 12 углов от 0 до 44 с шагом 4
Xrot = X;
for j=2:nAngles
    Y = imrotate(X,(j-1)*4,'bicubic'); % повернуть на угол (j-1)*4
    startind = floor((size(Y,1)-n)/2);
    Xrot1 = Y(startind:startind+n-1, startind:startind+n-1);
    Xrot2 = Xrot1 - Xrot1(1,1);
    Xrot2 = Xrot2/max(Xrot2(:));
    Xrot(Xrot2>.5) = j;

    [U,S,V] = svd(Xrot1);
    subplot(1,2,1), imagesc(Xrot), colormap([0 0 0; cm])
    subplot(1,2,2), semilogy(diag(S),'-o','color',cm(j,:))
end
```

1.8. Рандомизированное сингулярное разложение

Точное и эффективное разложение больших матриц – один из столпов современной вычислительной математики и науки о данных. Во многих случаях явно декларируется, что цель разложения – выявить в матрице доминирующую низкоранговую структуру, это проиллюстрировано на приведенных выше примерах. Недавно было показано, что если у матрицы $\mathbf{X}$ имеется низкоранговая структура, то для ее выявления можно применить чрезвычайно эффективные алгоритмы, основанные на случайной выборке; это тесно связано с идеей разреженности и многомерной геометрией разреженных векторов, которой мы займемся в главе 3. Эти так называемые *рандомизированные* численные методы потенциально способны преобразить вычислительную линейную алгебру, поскольку вычисляют точные разложения матриц многократно быстрее, чем детерминированные методы. Кроме того, хотя объем наблюдаемых данных постоянно растет (например, видео с разрешением 4K и 8K, интернет вещей и т. д.), часто оказывается, что *внутренний* ранг данных почти не увеличивается, хотя размерность пространства измерений и возрастает. Таким образом, экономия вычислений, достигаемая рандомизированными методами, в будущем, сулящем шквал данных, будет только возрастать.

Рандомизированная линейная алгебра

Рандомизированная линейная алгебра – гораздо более общая теория, чем представленное здесь применение в контексте SVD. Помимо рандомизированного SVD [464, 371], разработаны рандомизированные алгоритмы для ме-

тода главных компонент [454, 229], LUP-разложения [485], QRP-разложения [162] и разложения по динамическим модам [175]. Большинство рандомизированных алгоритмов разложения матриц можно разбить на несколько типичных шагов, описанных ниже. На эту тему имеется также несколько прекрасных обзоров: [354, 228, 334, 177]. Мы будем предполагать, что матрица высокая и тощая, т. е. $n > m$, хотя теория легко обобщается также на низкие и толстые матрицы.

Шаг 0: определить целевой ранг $r < m$.
Шаг 1: выполняя случайные проекции $\mathbf{P}$ для выборки из пространства столбцов, найти матрицу $\mathbf{Q}$, столбцы которой аппроксимируют пространство столбцов $\mathbf{X}$, т. е. $\mathbf{X} \approx \mathbf{QQ}^*\mathbf{X}$.
Шаг 2: спроецировать $\mathbf{X}$ на подпространство $\mathbf{Q}$, $\mathbf{Y} = \mathbf{Q}^*\mathbf{X}$ и вычислить разложение матрицы на $\mathbf{Y}$.
Шаг 3: реконструировать моды высокой размерности $\mathbf{U} = \mathbf{QU}_\mathbf{Y}$, используя $\mathbf{Q}$ и моды, вычисленные по $\mathbf{Y}$.

Рандомизированный алгоритм SVD

За последние двадцать лет было предложено несколько рандомизированных алгоритмов вычисления низкорангового сингулярного разложения, в т. ч. SVD методом *Монте-Карло* [190] и более робастные решения, основанные на случайных проекциях [464, 335, 371]. Эти методы были усовершенствованы путем включения структурированных выборочных матриц для ускорения матричного умножения [559]. Ниже мы воспользуемся рандомизированным алгоритмом SVD Халко, Мартинсона и Троппа [228], который объединил и дополнил предыдущие алгоритмы и обеспечивает хорошие оценки ошибок. Дополнительный анализ и детали численной реализации см. в работе Voronin and Martinsson [544]. Схематично алгоритм rSVD показан на рис. 1.27.

Шаг 1. Строим случайную матрицу проекции $\mathbf{P} \in \mathbb{R}^{m \times r}$ для выборки из пространства столбцов матрицы $\mathbf{X} \in \mathbb{R}^{n \times m}$:

$$\mathbf{Z} = \mathbf{XP}. \tag{1.35}$$

Матрица $\mathbf{Z}$ может быть гораздо меньше $\mathbf{X}$, особенно для низкоранговых матриц, когда $r \ll m$. Крайне маловероятно, что случайная матрица проекции $\mathbf{P}$ выбросит важные компоненты $\mathbf{X}$, поэтому $\mathbf{Z}$ аппроксимирует пространство столбцов $\mathbf{X}$ с высокой вероятностью. Таким образом, можно вычислить низкоранговое QR-разложение $\mathbf{Z}$, чтобы получить ортонормированный базис $\mathbf{X}$:

$$\mathbf{Z} = \mathbf{QR}. \tag{1.36}$$

Шаг 2. Имея низкоранговый базис $\mathbf{Q}$, мы можем спроецировать $\mathbf{X}$ на меньшее подпространство:

$$\mathbf{Y} = \mathbf{Q}^*\mathbf{X}. \tag{1.37}$$

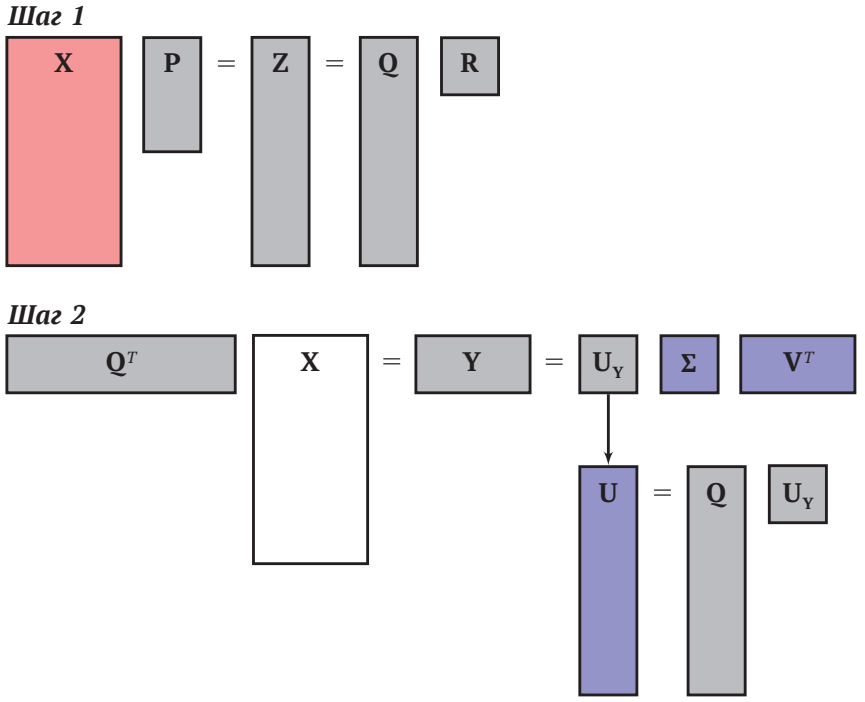

Рис. 1.27 ❖ Схема рандомизированного алгоритма SVD. Данные **X** высокой размерности изображены красным цветом, промежуточные шаги – серым, а результаты – синим. Этот алгоритм требует двух проходов по **X**

Отсюда также следует, что $\mathbf{X} \approx \mathbf{QY}$, причем аппроксимация будет точнее, если сингулярные значения σ_k быстро убывают при $k > r$.

Теперь можно вычислить сингулярное разложение **Y**:

$$\mathbf{Y} = \mathbf{U}_\mathbf{Y}\mathbf{\Sigma}\mathbf{V}^*. \tag{1.38}$$

Поскольку **Q** ортонормированная и аппроксимирует пространство столбцов **X**, матрицы **Σ** и **V** одинаковы что для **Y**, что для **X** (см. раздел 1.3).

Шаг 3. Наконец, мы можем реконструировать левые сингулярные векторы **U** высокой размерности, воспользовавшись $\mathbf{U}_\mathbf{Y}$ и **Q**:

$$\mathbf{U} = \mathbf{QU}_\mathbf{Y}. \tag{1.39}$$

Избыточная выборка

У большинства матриц **X** нет низкоранговой структуры точно с r модами. Сингулярные значения σ_k для $k > r$ обычно отличны от нуля, и столбцы матрицы-эскиза **Z** не образуют точный базис пространства столбцов **X**. В общем случае увеличение количества столбцов **P** с r до $r + p$ значительно улучшает результат, даже когда добавка p варьируется в диапазоне от 5 до 10 столбцов

[370]. Этот прием называется *избыточной выборкой*, а увеличение p уменьшает дисперсию спектра сингулярных значений матрицы-эскиза.

Степенные итерации

В рандомизированных алгоритмах встречается и другая проблема – медленное убывание спектра сингулярных значений, так что в отброшенных сингулярных значениях остается значительная доля дисперсии данных в $\mathbf{X}$. В таком случае можно предварительно обработать $\mathbf{X}$, выполнив *q степенных итераций* [454, 228, 224] с целью создать новую матрицу $\mathbf{X}^{(q)}$, сингулярные значения которой убывают быстрее:

$$\mathbf{X}(q) = (\mathbf{X}\mathbf{X}^*)^q\mathbf{X}. \tag{1.40}$$

Степенные итерации значительно улучшают качество рандомизированного разложения, поскольку спектр сингулярных значений $\mathbf{X}^{(q)}$ убывает быстрее:

$$\mathbf{X}^{(q)} = \mathbf{U}\mathbf{\Sigma}^{2q-1}\mathbf{V}^*. \tag{1.41}$$

Однако степенные итерации обходятся дорого, так как требуется q дополнительных проходов по данным $\mathbf{X}$. В некоторых экстремальных случаях данные $\mathbf{X}$ хранятся в распределенной системе, и тогда каждый дополнительный проход стоит очень дорого.

Гарантированные границы ошибки

Одно из самых важных свойств рандомизированного SVD – существование настраиваемых границ ошибки, которые представляют собой явно заданные функции от спектра сингулярных значений, желаемого ранга r, параметра избыточной выборки p и количества степенных итераций q. Наилучшая достижимая граница ошибки для детерминированного алгоритма имеет вид

$$||\mathbf{X} - \mathbf{Q}\mathbf{Y}||_2 \geq \sigma_{r+1}(\mathbf{X}). \tag{1.42}$$

Иными словами, для аппроксимации наилучшим возможным подпространством $\mathbf{Q}$ ранга r ошибка больше или равна первому отброшенному сингулярному значению $\mathbf{X}$. Для рандомизированных методов можно ограничить сверху *математическое ожидание* ошибки:

$$\mathbb{E}(||\mathbf{X} - \mathbf{Q}\mathbf{Y}||_2) \leq \left(1 + \sqrt{\frac{r}{p-1}} + \frac{e\sqrt{r+p}}{p}\sqrt{m-r}\right)^{\frac{1}{2q+1}}\sigma_{k+1}(\mathbf{X}), \tag{1.43}$$

где e – число Эйлера.

Выбор случайной матрицы P

Есть несколько способов выбрать случайную матрицу $\mathbf{P}$. Часто используются гауссовы случайные проекции (например, элементы $\mathbf{P}$ – независимые и одинаково распределенные гауссовы случайные величины), что связано с их удобными математическими свойствами и степенью полноты информации,

попадающей в матрицу-эскиз **Z**. В частности, крайне маловероятно, что гауссова случайная матрица **P** окажется настолько *плохой*, что исключит из **X** важную информацию. Однако гауссовы проекции дорого генерировать, хранить и вычислять. Равномерные случайные матрицы также применяются часто и имеют похожие ограничения. Существует несколько альтернатив, например матрицы Радемахера, элементы которых равны $+1$ или -1 с одинаковой вероятностью [532]. Структурированные случайные матрицы проекций могут порождать эффективные эскизы и сокращать вычислительную сложность до $O(nm\log(r))$ [559]. Еще один вариант – разреженная матрица проекции **P**, для которой стоимость хранения и вычислений оказывается ниже, но ценой включения в эскиз меньшего объема информации. В экстремальном случае, когда даже один проход по матрице **X** обходится недопустимо дорого, матрицу **P** можно построить из случайно выбранных столбцов единичной матрицы размера $m \times m$, т. е. эскиз **Z** будет состоять из случайно выбранных столбцов **X**. Это самый быстрый вариант, но использовать его следует с осторожностью, поскольку можно потерять информацию, если важная структура **X** локализована в небольшом подмножестве столбцов, не вошедших в выборку.

Пример рандомизированного SVD

Для демонстрации рандомизированного алгоритма SVD разложим изображение высокого разрешения. Эта конкретная реализация приведена только для иллюстрации, она не оптимизирована по быстродействию, объему передаваемых данных и точности. На практике подходить к реализации следует внимательно [228, 177].

Функция в листинге 1.23 вычисляет рандомизированное SVD матрицы **X**, а программа в листинге 1.24 использует эту функцию для получения аппроксимации ранга 400 изображения высокого разрешения, показанного на рис. 1.28.

Листинг 1.23 ❖ Рандомизированный алгоритм SVD

```
function [U,S,V] = rsvd(X,r,q,p);

% Шаг 1: выборка из пространства столбцов X с помощью матрицы P
ny = size(X,2);
P = randn(ny,r+p);
Z = X*P;
for k=1:q
    Z = X*(X'*Z);
end
[Q,R] = qr(Z,0);

% Шаг 2: вычисление SVD для проекции Y=Q'*X;
Y = Q'*X;
[UY,S,V] = svd(Y,'econ');
U = Q*UY;
```

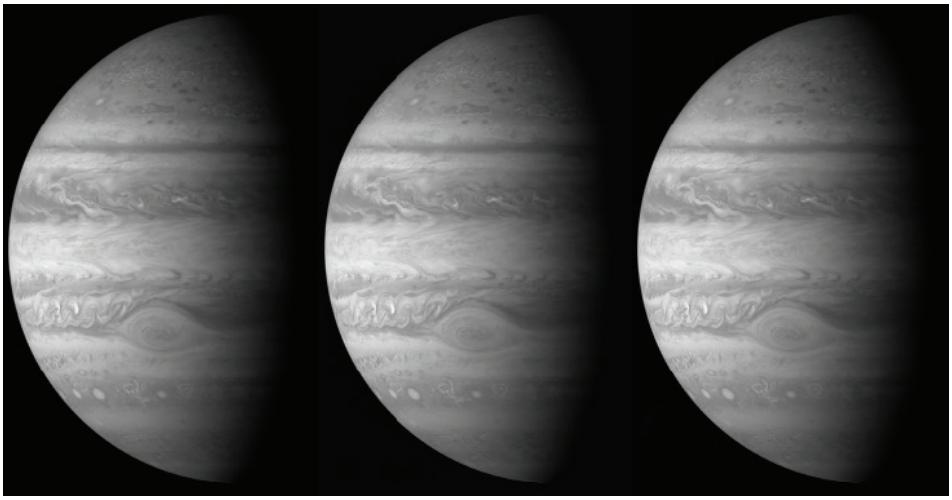

Рис. 1.28 ❖ Исходное изображение высокого разрешения (слева) и аппроксимации ранга 400, полученные методами SVD (в центре) и rSVD (справа)

Листинг 1.24 ❖ Вычисление рандомизированного SVD для изображения высокого разрешения

```
clear all, close all, clc
A=imread('jupiter.jpg');
X=double(rgb2gray(A));
[U,S,V] = svd(X,'econ'); % детерминированный SVD

r = 400;   % целевой ранг
q = 1;     % число степенных итераций
p = 5;     % параметр избыточной выборки
[rU,rS,rV] = rsvd(X,r,q,p); % рандомизированный SVD

%% Реконструкция
XSVD = U(:,1:r)*S(1:r,1:r)*V(:,1:r)';     % SVD-аппроксимация
errSVD = norm(X-XSVD,2)/norm(X,2);
XrSVD = rU(:,1:r)*rS(1:r,1:r)*rV(:,1:r)'; % rSVD-аппроксимация
errrSVD = norm(X-XrSVD,2)/norm(X,2);
```

1.9. Тензорные разложения и *N*-мерные массивы данных

Низкоранговые разложения можно обобщить с матриц на тензоры. Это важно, потому что для вычисления SVD необходимо упаковывать несопоставимые типы данных в один вектор для нахождения коррелирующих структур. Например, моментальные снимки (столбцы), сделанные в разные моменты времени, могут включать измерения таких разных величин, как температура, давление, концентрация вещества и т. д. Кроме того, могут присутствовать

и категориальные (дискретные) данные. Векторизация таких данных обычно не имеет смысла. Так что желательно сохранить данные разной структуры и типа в отдельных независимых направлениях. Матрицы можно обобщить, введя в рассмотрение N-мерные массивы, или тензоры, в которых данные можно организовать более естественно, не прибегая к разглаживанию (flattening).

Для построения тензоров нам придется вспомнить нотацию, применяемую для сложения и различных видов произведения тензоров [299]. Будем обозначать $\mathbf{a}_r$ r-й столбец матрицы $\mathbf{A}$. Если даны матрицы $\mathbf{A} \in \mathbb{R}^{I\times K}$ и $\mathbf{B} \in \mathbb{R}^{J\times K}$, то их произведение Хатри-Рао, обозначаемое $\mathbf{A} \odot \mathbf{B}$, определяется как матрица размера $IJ\times K$, составленная из произведений Кронекера соответственных столбцов

$$\mathbf{A} \odot \mathbf{B} = (\mathbf{a}_1 \otimes \mathbf{b}_1 \cdots \mathbf{a}_K \otimes \mathbf{b}_K).$$

Если $\mathcal{A}$ – N-мерный тензор размера $I_1\times I_2 \times \cdots \times I_N$, то его элемент $\mathbf{i} = (i_1, i_2, \cdots, i_N)$ обозначается $a_{\mathbf{i}}$.

Скалярное произведение двух N-мерных тензоров $\mathcal{A}$ и $\mathcal{B}$ совместимых размерностей определяется следующим образом:

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{\mathbf{i}} a_{\mathbf{i}} b_{\mathbf{i}}.$$

Нормой Фробениуса тензора $\mathcal{A}$, обозначаемой $||\mathcal{A}||_{\mathrm{F}}$, называется квадратный корень из скалярного произведения $\mathcal{A}$ с самим собой: $||\mathcal{A}||_{\mathrm{F}} = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$. Наконец, n-модовое разворачивание (matricization или unfolding) тензора $\mathcal{A}$ обозначается $\mathbf{mA}^{(n)}$.

Пусть $\mathcal{M}$ – N-мерный тензор размера $I_1 \times I_2 \times \cdots \times I_N$. Нас интересует R-компонентная факторная модель CANDECOMP/PARAFAC (CP) [124, 235, 299]

$$\mathcal{M} = \sum_{r=1}^{R} \lambda_r \mathbf{ma}_r^{(1)} \circ \cdots \circ \mathbf{ma}_r^{(N)}, \tag{1.44}$$

где $\circ$ обозначает внешнее произведение, а $\mathbf{ma}_r^{(n)}$ – r-й столбец *факторной матрицы* $\mathbf{mA}^{(n)}$ размера $I_n\times R$. CP-разложение – аббревиатура CANDECOMP/PARAFAC, в которой PARAFAC означает *параллельный факторный анализ*, а CANDECOMP – *каноническое разложение*. Каждый сомножитель мы будем называть *компонентой*. В предположении, что во всех факторных матрицах столбцы нормированы и имеют единичную евклидову длину, будем называть λ_r *весами*. Будем также использовать сокращенную нотацию $\lambda = (\lambda_1, \cdots, \lambda_R)^T$ [25]. Тензор, допускающий CP-разложение, иногда называют тензором Краскала.

Далее в этой главе будет рассматриваться 3-мерное тензорное CP-разложение (см. рис. 1.29), в котором две моды индексируют изменение состояния, а третья – изменение времени:

$$\mathcal{M} = \sum_{r=1}^{R} \lambda_r \mathbf{A}_r \circ \mathbf{B}_r \circ \mathbf{C}_r.$$

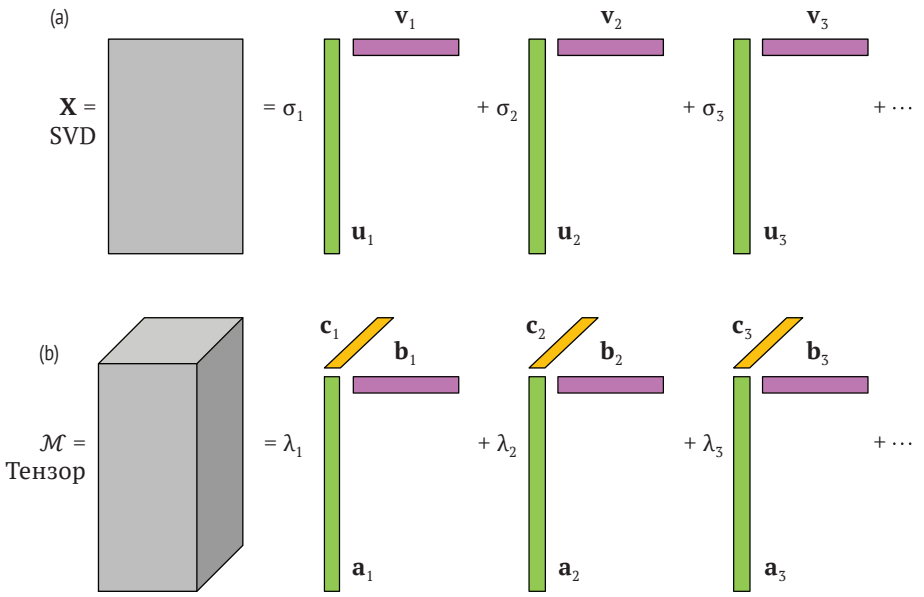

Рис. 1.29 ❖ Сравнение SVD и тензорного разложения. Оба метода порождают аппроксимацию исходной матрицы данных суммами внешних произведений. Но тензорное разложение обобщает идею SVD на *N*-мерные массивы без разглаживания (векторизации) данных

Обозначим $\mathbf{A} \in \mathbb{R}^{I_1 \times R}$ и $\mathbf{B} \in \mathbb{R}^{I_2 \times R}$ факторные матрицы, соответствующие двум модам состояния, а $\mathbf{C} \in \mathbb{R}^{I_3 \times R}$ факторную матрицу, соответствующую временной моде. На рис. 1.29 показано сравнение такого 3-мерного разложения с SVD.

Для иллюстрации тензорного разложения мы воспользуемся пакетом *N*-way для MATLAB, разработанным Расмусом Бро с сотрудниками [84, 15], который размещен на файлообменнике Mathworks. Этот простой в применении пакет содержит ряд инструментов для выполнения тензорных разложений и оценки получающихся факторных моделей. В примере ниже мы генерируем данные с помощью следующей пространственно-временной функции (рис. 1.30):

$$F(x, y, t) = \exp(-x^2 - 0.5y^2)\cos(2t) + \operatorname{sech}(x)\tanh(x)\exp(-0.2y^2)\sin(t). \qquad (1.45)$$

В этой модели имеются две пространственные моды с разными временными частотами, и, следовательно, двухфакторной модели должно быть достаточно для выделения пространственных и временных мод. Для построения функции в MATLAB используется следующая программа.

Листинг 1.25 ❖ Создание тензорных данных

```
x=-5:0.1:5; y=-6:0.1:6; t=0:0.1:10*pi;
[X,Y,T]=meshgrid(x,y,t);
A=exp(-(X.^2+0.5*Y.^2)).*(cos(2*T))+ ...
    (sech(X).*tanh(X).*exp(-0.2*Y.^2)).*sin(T);
```

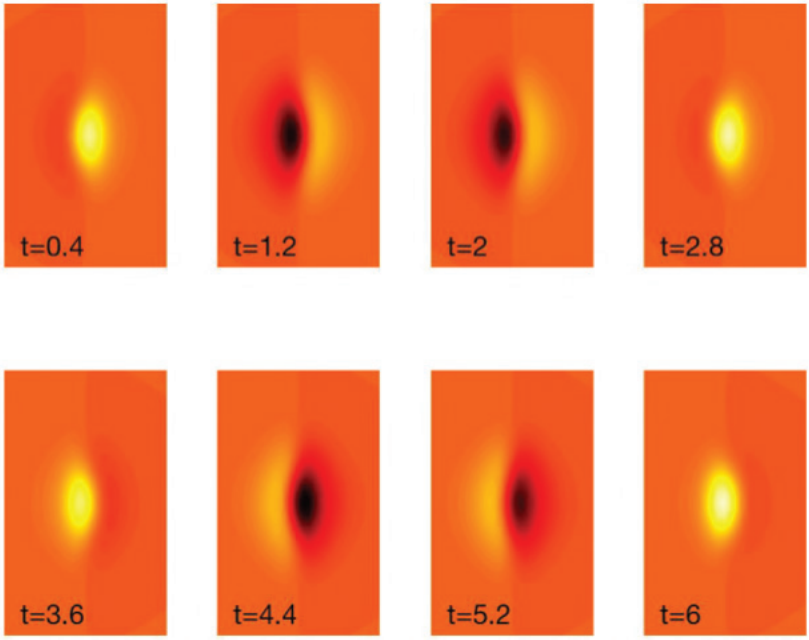

Рис. 1.30 ❖ Пример *N*-мерного набора данных, созданного функцией (1.45). Матрица данных $\mathbf{A} \in \mathbb{R}^{121\times101\times315}$. Тензорное CP-разложение можно использовать для выделения обеих исходных структур, породивших данные

Отметим, что команда **meshgrid** умеет генерировать *N*-мерные массивы. Вообще, в MATLAB очень легко задавать многомерные массивы и тензоры. В частности, команда **A** = **randn**(**10**, **10**, **10**, **10**, **10**) генерирует 5-мерный гиперкуб со случайными значениями по каждому из пяти измерений.

На рис. 1.30 показано восемь снимков функции (1.45), дискретизированной приведенной выше программой, которая порождает массив $\mathbf{A} \in \mathbb{R}^{121\times101\times315}$, содержащий приблизительно 10^6 элементов. Тензорное CP-разложение можно использовать для выделения из этого 3-мерного массива двухфакторной модели и, следовательно, порождения двух векторов в пространственных направлениях *x* и *y* и временном направлении *t*.

Пакет *N*-way обладает простой архитектурой, подходящей для тензорных разложений. Команда PARAFAC принимает входную функцию (1.45), дискретизирует ее с помощью приведенной выше программы и порождает двухфакторную модель. Следующая программа выводит результат в переменной **model**.

Листинг 1.26 ❖ Двухфакторная тензорная модель

```
model=parafac(A,2);
[A1,A2,A3]=fac2let(model);
subplot(3,1,1), plot(y,A1,'Linewidth',[2])
subplot(3,1,2), plot(x,A2,'Linewidth',[2])
subplot(3,1,3), plot(t,A3,'Linewidth',[2])
```

Заметим, что команда **fac2let** в этом коде преобразует факторы модели в соответствующие матрицы. Еще отметим, что организация данных для **meshgrid** отлична от предполагаемой в **parafac**, поскольку направления *x* и *y* переставлены местами.

На рис. 1.31 показаны результаты разложения *N*-мерного тензора в предписанную двухфакторную модель. Точнее, показаны оба вектора по каждому из трех направлений массива. В данном случае известен точный ответ, потому что данные генерировались моделью (1.45) ранга 2. Первый набор двух мод (вдоль исходного направления *y*) является гауссовым, как и предписано. Второй набор двух мод (вдоль исходного направления *x*) состоит из гауссианы в качестве первой функции и антисимметричной $\mathrm{sech}(x)\tanh(x)$ в качестве второй функции. Третий набор двух мод соответствует временной динамике обеих функций: $\cos(2t)$ и $\sin(t)$. Таким образом, двухфакторная модель, порожденная тензорным CP-разложением, возвращает ожидаемые низкоранговые функции, которые и порождали матрицу **A** высокой размерности.

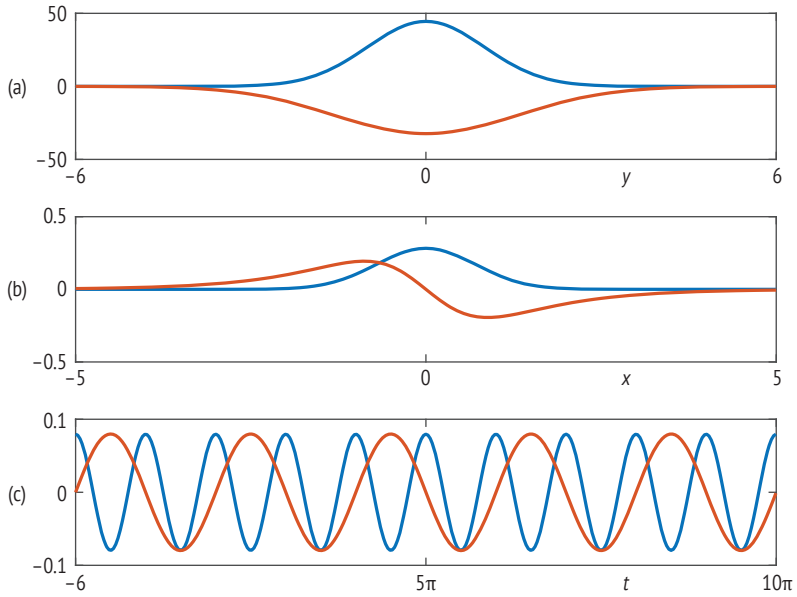

Рис. 1.31 ❖ Разложение 3-мерного тензора, сгенерированного функцией (1.45), дискретизированное так, что матрица данных $\mathbf{A} \in \mathbb{R}^{121\times101\times315}$. Тензорное CP-разложение можно использовать для выделения двух структур, порождающих эти данные. Первый фактор показан синим цветом, второй – красным. Три измерения данных (параллельные факторы) показаны на рисунках: (a) направление *y*, (b) направление *x*, (c) время *t*

Недавний теоретический и вычислительный прогресс в *N*-мерных разложениях открывает возможность производить тензорные разложения во многих областях. Но для больших *N* такие разложения могут оказаться практически невозможными из-за непомерного объема данных. Действительно,

даже в простом примере, показанном на рис. 1.30 и 1.31, имеется 10^6 точек. В общем, тензорное CP-разложение не очень хорошо масштабируется при увеличении размерности. Однако рандомизированные методы позволяют получать результат даже для очень больших наборов данных [158, 175]. Как и в случае SVD, в рандомизированных методах используется истинная низкоранговая структура данных для получения хорошей аппроксимации путем суммирования внешних произведений ранга 1. Кроме того, тензорные разложения можно сочетать с ограничениями на форму параллельных факторов, чтобы результаты было проще интерпретировать [348]. Тем самым мы получаем инфраструктуру для получения масштабируемых и допускающих интерпретацию вычислений с N-мерными массивами данных.

Рекомендуемая литература

Учебники

(1) Matrix computations, by G. H. Golub and C. F. Van Loan, 2012 [214].

Статьи и обзоры

(1) Calculating the singular values and pseudo-inverse of a matrix, by G. H. Golub and W. Kahan, *Journal of the Society for Industrial & Applied Mathematics, Series B: Numerical Analysis*, 1965 [212].

(2) A low-dimensional procedure for the characterization of human faces, by L. Sirovich and M. Kirby, *Journal of the Optical Society of America A*, 1987 [491].

(3) Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, by N. Halko, P.-G. Martinsson, and J. A. Tropp, *SIAM Review*, 2011 [230].

(4) A randomized algorithm for the decomposition of matrices, by P.-G. Martinsson, V. Rokhlin, and M. Tygert, *Applied and Computational Harmonic Analysis*, 2011 [371].

(5) The optimal hard threshold for singular values is $4/\sqrt{3}$, by M. Gavish and D. L. Donoho, *IEEE Transactions on Information Theory*, 2014 [200].

Глава 2

Преобразование Фурье и вейвлет-преобразование

В математической физике и инженерной математике центральное место занимает преобразование уравнений в систему координат, в которой выражения упрощаются, распадаются на части и становятся пригодны для вычислений и анализа. Эта сквозная тема данной книги, она возникает в самых разных контекстах: анализ данных (например, SVD), динамические системы (например, спектральное разложение по собственным значениям), теория управления (например, определение систем координат с учетом управляемости и наблюдаемости). Пожалуй, самое главное и повсеместно встречающееся преобразование координат было открыто Жаном-Батистом Жозефом Фурье в начале 1800-х годов в ходе исследований по теории тепла [185]. Согласно Фурье, синусы и косинусы возрастающей частоты образуют ортогональный *базис* пространства решений. В действительности синусы и косинусы, введенные Фурье, являются собственными функциями уравнения теплопроводности, конкретные частоты являются собственными значениями, определяемыми геометрией, а амплитуды определяются граничными условиями.

Первопроходческая работа Фурье заложила математические основания гильбертовых пространств, теории операторов, теории аппроксимации и последующей революции в аналитической и вычислительной математике. Пролетело две сотни лет – и быстрое преобразование Фурье стало краеугольным камнем вычислительной математики, именно благодаря ему возможно сжатие изображений и звука в режиме реального времени. Проще говоря, быстрое преобразование Фурье сыграло для формирования современного мира более важную роль, чем любой другой известный алгоритм.

Задачи, наборы данных и вычислительная геометрия постепенно усложнялись, и простые базисы Фурье, состоящие из синусов и косинусов, уступили место *специальным* базисам, таким как определяемое данными сингулярное разложение. На самом деле, как мы увидим ниже, SVD-базис можно рассматривать как прямой аналог базиса Фурье для решения УрЧП со сложной геометрией. Кроме того, для усовершенствованной обработки и сжатия сигналов были предложены родственные функции, названные вейвлетами. В этой главе мы продемонстрируем некоторые из многочисленных применений преобразования Фурье и вейвлет-преобразований.

2.1. Ряд Фурье и преобразование Фурье

Прежде чем описывать численное применение преобразования Фурье к векторам данных, мы дадим введение в аналитические ряды и преобразование Фурье для непрерывных функций. Понятно, что дискретная и непрерывная постановка должны совпадать в пределе, когда разрешение данных бесконечно велико. Ряд и преобразование Фурье тесно связаны с геометрией бесконечномерных пространств функций, или *гильбертовых* пространств, которые являются обобщением векторных пространств на множества функций с бесконечным числом степеней свободы. Поэтому начнем с введения в пространства функций.

Скалярные произведения функций и векторов

В этом разделе мы поговорим о скалярных произведениях и нормах функций. Точнее, мы будем использовать обычное эрмитово скалярное произведение функций $f(x)$ и $g(x)$, определенное для $x \in [a, b]$:

$$\langle f(x), g(x) \rangle = \int_a^b f(x)\bar{g}(x)dx, \tag{2.1}$$

где $\bar{g}$ обозначает комплексное сопряжение.

Такое определение скалярного произведения функций может поначалу показаться странным и произвольным, но оно обретает смысл, если рассмотреть скалярное произведение векторов. Дискретизируем функции $f(x)$ и $g(x)$, представив их векторами данных, как показано на рис. 2.1. Нам хотелось бы, чтобы скалярное произведение векторов сходилось к скалярному произведению функций при увеличении объема выборки. Скалярное произведение векторов $\mathbf{f} = [f_1 \ f_2 \ \cdots \ f_n]^T$ и $\mathbf{g} = [g_1 \ g_2 \ \cdots \ g_n]^T$ определяется следующим образом:

$$\langle \mathbf{f}, \mathbf{g} \rangle = \mathbf{g}^* \mathbf{f} = \sum_{k=1}^{n} f_k \bar{g}_k = \sum_{k=1}^{n} f(x_k)\bar{g}(x_k). \tag{2.2}$$

Абсолютная величина этого скалярного произведения возрастает при добавлении новых точек, т. е. при увеличении n. Поэтому нормируем его, умножив на $\Delta x = (b - a)/(n - 1)$:

$$\frac{b-a}{n-1}\langle \mathbf{f}, \mathbf{g} \rangle = \sum_{k=1}^{n} f(x_k)\bar{g}(x_k)\Delta x. \tag{2.3}$$

Это риманова аппроксимация скалярного произведения непрерывных функций. Теперь ясно, что если устремить $n \to \infty$ (т. е. перейти к бесконечному разрешению $\Delta x \to 0$), то скалярное произведение векторов будет сходиться к скалярному произведению функций, определенному формулой (2.1).

Это скалярное произведение также индуцирует норму в пространстве функций:

$$||f||_2 = (\langle f, f \rangle)^{1/2} = \sqrt{\langle f, f \rangle} = \left(\int_a^b f(x)\bar{g}(x)dx \right)^{1/2}. \quad (2.4)$$

Функции с ограниченной нормой образуют множество квадратично интегрируемых функций, обозначаемое $L^2([a, b])$; их также называют функциями, интегрируемыми по Лебегу. Интервал $[a, b]$ может быть и бесконечным (например, $(-\infty, \infty)$), полубесконечным (например, $[a, \infty)$) или периодическим (например, $[-\pi, \pi)$). Примером функции, принадлежащей $L^2([1, \infty))$, является $f(x) = 1/x$. Интеграл квадрата f на интервале от 1 до ∞ конечен, хотя интеграл самой функции расходится. Тело, полученное вращением этой функции вокруг оси x, называется рогом Гавриила; его объем конечен (поскольку определяется интегралом f^2), а площадь поверхности бесконечна (она определяется интегралом f).

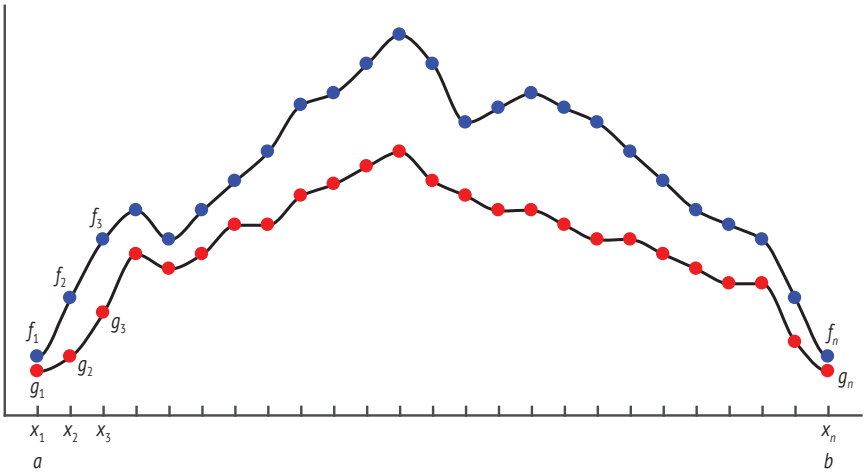

Рис. 2.1 ❖ Иллюстрация скалярного произведения на примере дискретизированных функций

Как и в конечномерных пространствах, скалярное произведение можно использовать для *проецирования* функции в новую систему координат, определенную базисом ортогональных функций. Представление функции f рядом Фурье как раз и является проекцией функции на ортогональное множество синусов и косинусов с целочисленным периодом на отрезке $[a, b]$. Это тема следующего раздела.

Ряд Фурье

Фундаментальный результат анализа Фурье заключается в том, что если функция $f(x)$ периодическая и кусочно-гладкая, то ее можно представить рядом Фурье, т. е. бесконечной суммой синусов и косинусов возрастающей частоты. В частности, если $f(x)$ периодична с периодом 2π, то ее можно записать в виде:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos(kx) + b_k \sin(kx)). \tag{2.5}$$

Коэффициенты a_k и b_k определяются по формулам:

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(kx)dx; \tag{2.6a}$$

$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(kx)dx, \tag{2.6b}$$

которые можно рассматривать как координаты, полученные проецированием функции на базис, образованный ортогональными синусами и косинусами $\{\cos(kx), \sin(kx)\}_{k=0}^{\infty}$. Иными словами, интегралы в формулах (2.6) можно переписать в терминах скалярного произведения:

$$a_k = \frac{1}{||\cos(kx)||^2}\langle f(x), \cos(kx)\rangle; \tag{2.7a}$$

$$b_k = \frac{1}{||\sin(kx)||^2}\langle f(x), \sin(kx)\rangle, \tag{2.7b}$$

где $||\cos(kx)||^2 = ||\sin(kx)||^2 = \pi$ – этот факт легко проверить, численно проинтегрировав $\cos^2(x)$ и $\sin^2(x)$ на отрезке от $-\pi$ до π.

Аналогично ряд Фурье для функции с периодом L на интервале $[0, L)$ имеет вид:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}\left(a_k \cos\left(\frac{2\pi kx}{L}\right) + b_k \sin\left(\frac{2\pi kx}{L}\right)\right), \tag{2.8}$$

где коэффициенты a_k и b_k равны:

$$a_k = \frac{2}{L}\int_0^L f(x)\cos\left(\frac{2\pi kx}{L}\right)dx; \tag{2.9a}$$

$$b_k = \frac{2}{L}\int_0^L f(x)\sin\left(\frac{2\pi kx}{L}\right)dx. \tag{2.9b}$$

Поскольку мы разлагаем функции по синусам и косинусам, естественно воспользоваться формулой Эйлера $e^{ikx} = \cos(kx) + i\sin(kx)$ и записать ряд Фурье в комплексной форме с комплексными коэффициентами $c_k = \alpha_k + i\beta_k$:

$$\begin{aligned} f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx} &= \sum_{k=-\infty}^{\infty}(\alpha_k + i\beta_k)(\cos(kx) + i\sin(kx)) \\ &= (\alpha_0 + i\beta_0) + \sum_{k=1}^{\infty}[(\alpha_{-k} + \alpha_k)\cos(kx) + (\beta_{-k} - \beta_k)\sin(kx)] \\ &\quad + i\sum_{k=1}^{\infty}[(\beta_{-k} + \beta_k)\cos(kx) - (\alpha_{-k} - \alpha_k)\sin(kx)]. \end{aligned} \tag{2.10}$$

Если $f(x)$ вещественная, то $\alpha_{-k} = \alpha_k$ и $\beta_{-k} = -\beta_k$, так что $c_{-k} = \bar{c}_k$.

Таким образом, функции $\psi_k = e^{ikx}$ для $k \in Z$ (т. е. для целых k) образуют базис пространства периодических комплексных функций на интервале $[0, 2\pi)$. Легко видеть, что эти функции ортогональны:

$$\langle \psi_j, \psi_k \rangle = \int_{-\pi}^{\pi} e^{ijx} e^{-ikx} dx = \int_{-\pi}^{\pi} e^{i(j-k)x} dx = \left[\frac{e^{i(j-k)x}}{i(j-k)} \right]_{-\pi}^{\pi} = \begin{cases} 0, & \text{если } j \neq k \\ 2\pi, & \text{если } j = k \end{cases}.$$

Итак, $\langle \psi_j, \psi_k \rangle = 2\pi\delta_{jk}$, где δ – символ Кронекера. Аналогично функции $e^{i2\pi kx/L}$ образуют базис пространства $L^2([0, L))$ квадратично интегрируемых функций, определенных на интервале $[0, L)$.

В принципе, ряд Фурье – это просто представление функции $f(x)$ координатами в бесконечномерном пространстве функций, натянутом на ортогональный базис, составленный из синусов и косинусов (т. е. $\psi_k = e^{ikx} = \cos(kx) + i\sin(kx)$):

$$f(x) = \sum_{k=-\infty}^{\infty} c_k \psi_k(x) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \langle f(x), \psi_k(x) \rangle \psi_k(x). \tag{2.11}$$

Коэффициенты вычисляются по формуле $c_k = (1/2\pi)\langle f(x), \psi_k(x) \rangle$. Множитель $1/2\pi$ нормирует проекцию на квадрат нормы ψ_k, т. е. $||\psi_k||^2 = 2\pi$. Это согласуется со стандартной заменой базиса в конечномерном пространстве, показанной на рис. 2.2. Вектор $\vec{f}$ можно записать в системе координат $(\vec{x}, \vec{y})$ или $(\vec{u}, \vec{v})$, спроецировав на соответствующие ортогональные базисы:

$$\vec{f} = \langle \vec{f}, \vec{x} \rangle \frac{\vec{x}}{||\vec{x}||^2} + \langle \vec{f}, \vec{y} \rangle \frac{\vec{y}}{||\vec{y}||^2} \tag{2.12a}$$

$$= \langle \vec{f}, \vec{u} \rangle \frac{\vec{u}}{||\vec{u}||^2} + \langle \vec{f}, \vec{v} \rangle \frac{\vec{v}}{||\vec{v}||^2}. \tag{2.12b}$$

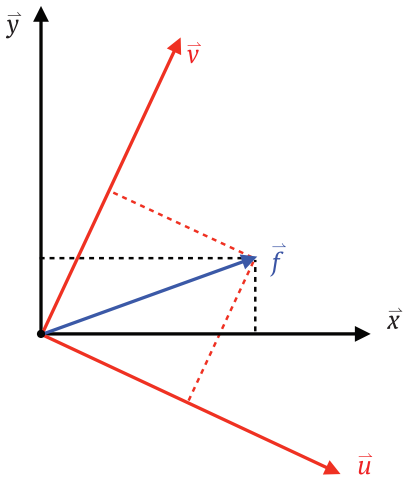

Рис. 2.2 ❖ Замена координат вектора в двумерном пространстве

Пример: ряд Фурье непрерывной треугольной функции

В качестве простого примера продемонстрируем использование ряда Фурье для аппроксимации непрерывной треугольной функции, определенной на интервале от $-\pi$ до π:

$$f(x) = \begin{cases} 0 & \text{для } x \in [-\pi, \pi/2] \\ 1 + 2x/\pi & \text{для } x \in [-\pi/2, 0] \\ 1 - 2x/\pi & \text{для } x \in [0, \pi/2] \\ 0 & \text{для } x \in [\pi/2, \pi] \end{cases}. \tag{2.13}$$

Поскольку это четная функция, ее можно аппроксимировать одними косинусами. На рис. 2.3 показано, как выглядят частичные суммы ряда Фурье для $f(x)$ при увеличении числа косинусов. На рис. 2.4 показаны коэффициенты a_k при четных косинусах, а также ошибка аппроксимации при увеличении количества мод. Коэффициенты b_k при нечетных синусах не показаны, поскольку для четной треугольной функции все они равны нулю.

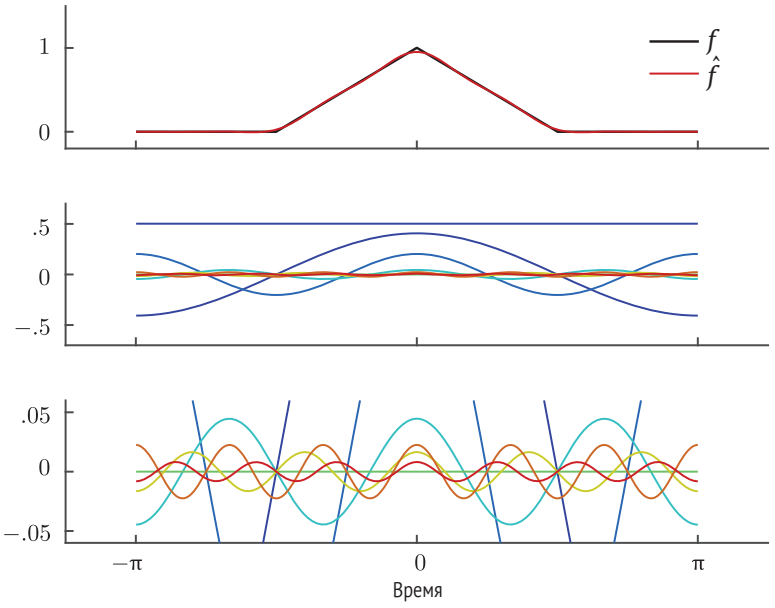

Рис. 2.3 ❖ (сверху) Треугольная функция и аппроксимация рядом Фурье, состоящим только из косинусов, при $n = 7$; (в середине) частичные суммы ряда Фурье, аппроксимирующие треугольную функцию; (снизу) увеличение масштаба мод при малой амплитуде и высокой частоте

Листинг 2.1 ❖ Аппроксимация треугольной функции рядом Фурье

```
% Задать область определения
dx = 0.001;
L = pi;
```

```
x = (-1+dx:dx:1)*L;
n = length(x); nquart = floor(n/4);

% Определить треугольную функцию
f = 0*x;
f(nquart:2*nquart) = 4*(1:nquart+1)/n;
f(2*nquart+1:3*nquart) = 1-4*(0:nquart-1)/n;
plot(x,f,'-k','LineWidth',1.5), hold on

% Вычислить ряд Фурье
CC = jet(20);
A0 = sum(f.*ones(size(x)))*dx;
fFS = A0/2;
for k=1:20
    A(k) = sum(f.*cos(pi*k*x/L))*dx; % скалярное произведение
    B(k) = sum(f.*sin(pi*k*x/L))*dx;
    fFS = fFS + A(k)*cos(k*pi*x/L) + B(k)*sin(k*pi*x/L);
    plot(x,fFS,'-','Color',CC(k,:),'LineWidth',1.2)
end
```

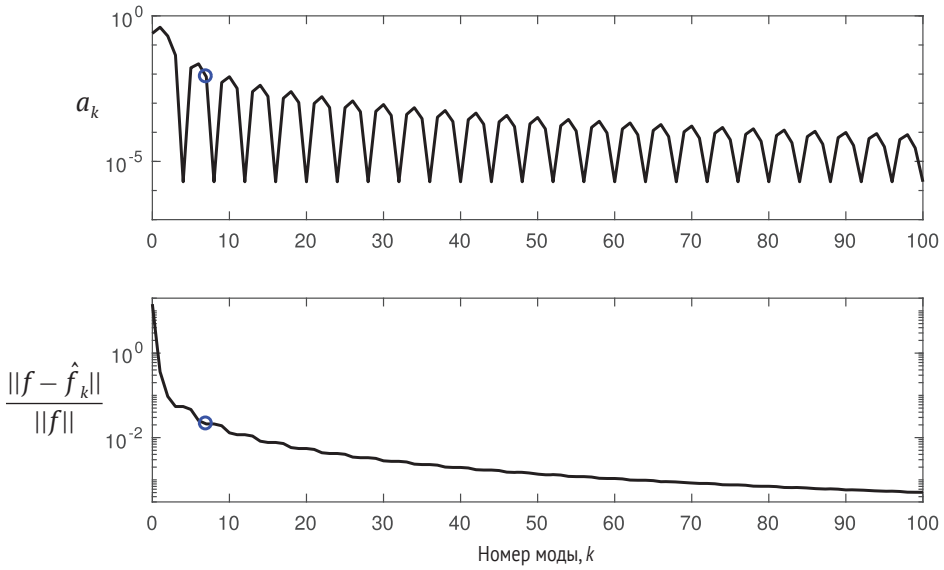

Рис. 2.4 ❖ Коэффициенты Фурье (сверху) и относительная погрешность аппроксимации рядом Фурье (снизу) для кусочно-постоянной функции на рис. 2.3. Аппроксимация, соответствующая $n = 7$, выделена синим кружком

Пример: ряд Фурье разрывной кусочно-постоянной функции

Теперь рассмотрим разрывную кусочно-постоянную функцию, определенную на интервале $[0, L)$ и показанную на рис. 2.5:

$$f(x) = \begin{cases} 0 & \text{для } x \in [0, L/4] \\ 1 & \text{для } x \in [L/4, 3L/4]. \\ 0 & \text{для } x \in [3L/4, L] \end{cases} \tag{2.14}$$

В этом случае наблюдаются осцилляции в окрестности точек разрыва, получившие название «явления Гиббса». Данный пример иллюстрирует проблемы, связанные с применением ряда Фурье к разрывным функциям.

```
dx = 0.01; L = 10;
x = 0:dx:L;
n = length(x); nquart = floor(n/4);

f = zeros(size(x));
f(nquart:3*nquart) = 1;

A0 = sum(f.*ones(size(x)))*dx*2/L;
fFS = A0/2;
for k=1:100
    Ak = sum(f.*cos(2*pi*k*x/L))*dx*2/L;
    Bk = sum(f.*sin(2*pi*k*x/L))*dx*2/L;
    fFS = fFS + Ak*cos(2*k*pi*x/L) + Bk*sin(2*k*pi*x/L);
end

plot(x,f,'k','LineWidth',2), hold on
plot(x,fFS,'r-','LineWidth',1.2)
```

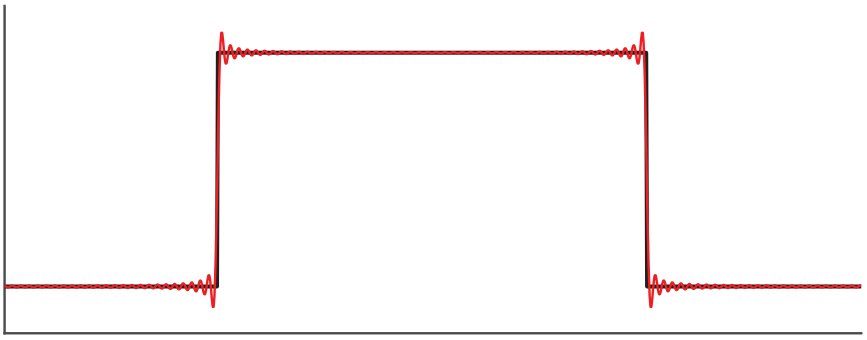

Рис. 2.5 ❖ Явление Гиббса – это высокочастотная осцилляция в окрестности точек разрыва. Черным цветом изображен график разрывной функции, а красным – его аппроксимация рядом Фурье

Преобразование Фурье

Ряд Фурье определен для периодических функций, которые бесконечно повторяют себя за пределами области определения. Интегральное преобразование Фурье по существу представляет собой предел ряда Фурье, когда длина области определения стремится к бесконечности. Это позволяет определить непериодическую функцию на всей оси $(-\infty, \infty)$, как показано на рис. 2.6. Будем рассматривать ряд Фурье на интервале $x \in [-L, L)$ и положим $L \to \infty$. В этой области определения ряд Фурье имеет вид:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}\left[a_k \cos\left(\frac{k\pi x}{L}\right) + b_k \sin\left(\frac{k\pi x}{L}\right)\right] = \sum_{k=-\infty}^{\infty} c_k e^{ik\pi x/L}, \tag{2.15}$$

где коэффициенты равны

$$c_k = \frac{1}{2L}\langle f(x), \psi_k \rangle = \frac{1}{2L}\int_{-L}^{L} f(x) e^{-ik\pi x/L} dx. \tag{2.16}$$

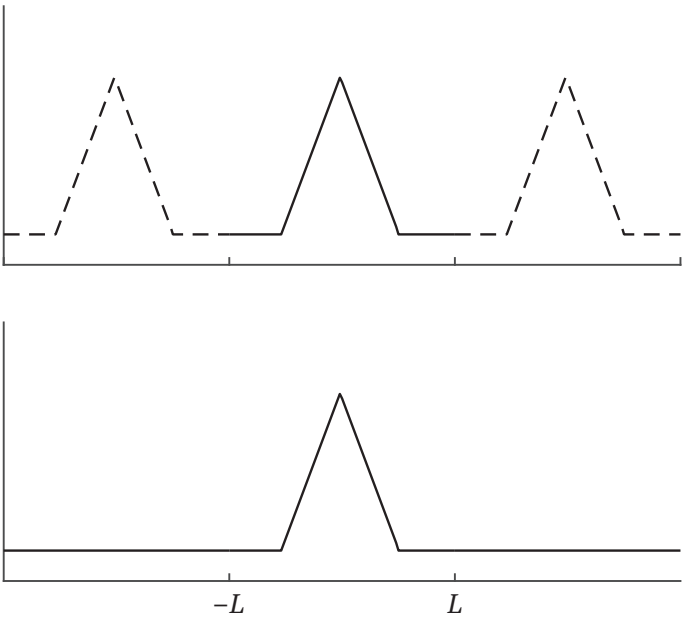

Рис. 2.6 ❖ (сверху) Ряд Фурье имеет смысл только для функции, периодически продолженной за пределы интервала $[-L, L)$ (снизу). Преобразование Фурье определено и для непериодических функций

Вспоминая предыдущие результаты, представим $f(x)$ суммой синусов и косинусов с дискретным набором частот $\omega_k = k\pi/L$. При переходе к пределу при $L \to \infty$ дискретный набор частот становится непрерывным диапазоном частот. Положим $\omega = k\pi/L$, $\Delta\omega = \pi/L$ и перейдем к пределу при $L \to \infty$, т. е. $\Delta\omega \to 0$:

$$f(x) = \lim_{\Delta\omega \to 0} \sum_{k=-\infty}^{\infty} \frac{\Delta\omega}{2\pi} \underbrace{\int_{-\pi/\Delta\omega}^{\pi/\Delta\omega} f(\xi) e^{-ik\Delta\omega\xi} d\xi}_{\langle f(x), \psi_k(x) \rangle} e^{ik\Delta\omega x}. \tag{2.17}$$

При переходе к пределу выражение $\langle f(x), \psi_k(x) \rangle$ становится преобразованием Фурье функции $f(x)$, обозначаемым $\hat{f}(\omega) \triangleq \mathcal{F}(f(x))$. А суммирование с весом $\Delta\omega$ становится интегралом Римана, т. е. мы получаем

$$f(x) = \mathcal{F}^{-1}\left(\hat{f}(\omega)\right) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(\omega) e^{i\omega x} d\omega; \tag{2.18a}$$

$$\hat{f}(\omega) = \mathcal{F}\left(f(\omega)\right) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx. \tag{2.18b}$$

Эти два интеграла называются *парой преобразований Фурье*. Оба интеграла сходятся при условии, что $\int_{-\infty}^{\infty}|f(x)|dx < \infty$ и $\int_{-\infty}^{\infty}|\hat{f}(\omega)|d\omega < \infty$, т. е. если обе функции интегрируемы по Лебегу, $f, \hat{f} \in L_1(-\infty, \infty)$.

Преобразование Фурье очень полезно в силу ряда свойств, в т. ч. линейности и поведения производных в частотной области. Эти свойства сплошь и рядом используются в анализе данных и научных расчетах (например, для точного и эффективного решения УрЧП), в чем мы убедимся ниже в этой главе.

Производные функций. Преобразование Фурье производной функции вычисляется следующим образом:

$$\mathcal{F}\left(\frac{d}{dx}f(x)\right) = \int_{-\infty}^{\infty}\overbrace{f'(x)}^{dv}\overbrace{e^{-i\omega x}}^{u}dx \tag{2.19a}$$

$$= \Big[\underbrace{f(x)e^{-i\omega x}}_{uv}\Big]_{-\infty}^{\infty} - \int_{-\infty}^{\infty}\underbrace{f(x)}_{v}\Big[\underbrace{-i\omega e^{-i\omega x}}_{du}\Big]dx \tag{2.19b}$$

$$= i\omega\int_{-\infty}^{\infty}f(x)e^{-i\omega x}dx \tag{2.19c}$$

$$= i\omega\mathcal{F}(f(x)). \tag{2.19d}$$

Это чрезвычайно важное свойство преобразования Фурье, поскольку оно позволяет преобразовывать дифференциальные уравнения в частных производных (УрЧП) в обыкновенные дифференциальные уравнения (ОДУ), что тесно связано с разделением переменных:

$$\underset{\text{(УрЧП)}}{u_{tt} = cu_{xx}} \quad \overset{\mathcal{F}}{\Longrightarrow} \quad \underset{\text{(ОДУ)}}{u_{tt} = -c\omega^2\hat{u}}. \tag{2.20}$$

Линейность преобразования Фурье. Преобразование Фурье – линейный оператор, т. е.

$$\mathcal{F}(\alpha f(x) + \beta g(x)) = \alpha\mathcal{F}(f) + \beta\mathcal{F}(g); \tag{2.21}$$

$$\mathcal{F}^{-1}(\alpha f(\omega) + \beta g(\omega)) = \alpha\mathcal{F}^{-1}(f) + \beta\mathcal{F}^{-1}(g). \tag{2.22}$$

Теорема Парсеваля

$$\int_{-\infty}^{\infty}|f(\omega)|^2d\omega = 2\pi\int_{-\infty}^{\infty}|f(x)|^2dx. \tag{2.23}$$

Иными словами, преобразование Фурье сохраняет норму L_2 с точностью до постоянной. Это свойство тесно связано с унитарностью, т. е. скалярное произведение двух функций до и после преобразования Фурье одинаково. Оно полезно также для аппроксимации и усечения, поскольку дает возможность ограничить ошибку при заданном усечении.

Свертка. Свертка двух функций особенно хорошо ведет себя в частотной области Фурье, поскольку переходит в произведение преобразованных функций. Определим свертку $f * g$ двух функций $f(x)$ и $g(x)$:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x-\xi)g(\xi)d\xi. \tag{2.24}$$

Полагая $\hat{f} = \mathcal{F}(f)$, $\hat{g} = \mathcal{F}(g)$, получаем:

$$\mathcal{F}^{-1}(\hat{f}\hat{g})(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(\omega)\hat{g}(\omega)e^{i\omega x}d\omega \tag{2.25a}$$

$$= \int_{-\infty}^{\infty} \hat{f}(\omega)e^{i\omega x}\left(\frac{1}{2\pi}\int_{-\infty}^{\infty} g(y)e^{-i\omega y}dy\right)d\omega \tag{2.25b}$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(y)\hat{f}(\omega)e^{i\omega(x-y)}d\omega dy \tag{2.25c}$$

$$= \int_{-\infty}^{\infty} g(y)\underbrace{\left(\frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(\omega)e^{i\omega(x-y)}d\omega\right)}_{f(x-y)}dy \tag{2.25d}$$

$$= \int_{-\infty}^{\infty} g(y)f(x-y)dy = g * f = f * g. \tag{2.25e}$$

Таким образом, умножению функций в частотной области соответствует свертка в пространственной области. Это особенно полезно в контексте систем управления, передаточных функций и преобразования Лапласа.

2.2. Дискретное преобразование Фурье (ДПФ) и быстрое преобразование Фурье (БПФ)

До сих пор мы рассматривали ряд Фурье и преобразование Фурье для непрерывных функций $f(x)$. Однако при работе с реальными данными необходимо аппроксимировать преобразование Фурье для дискретных векторов. Получающееся дискретное преобразование Фурье (ДПФ) – это по существу дискретизированный вариант ряда Фурье для векторов данных $\mathbf{f} = [f_1 \quad f_2 \quad f_3 \quad \cdots \quad f_n]^T$, полученных дискретизацией функции $f(x)$ с постоянным шагом Δx, как показано на рис. 2.7.

ДПФ весьма полезно для численной аппроксимации и вычислений, но оно плохо масштабируется на очень большие $n \gg 1$, поскольку при бесхитростном применении включает умножение на плотную матрицу $n \times n$, что требует $\mathcal{O}(n^2)$ операций. В 1965 году Джеймс У. Кули (James W. Cooley) из IBM и Джон У. Тьюки (John W. Tukey) из Принстонского университета разработали революционный алгоритм *быстрого* преобразования Фурье (БПФ) [137,

136] со сложностью $\mathcal{O}(n\log(n))$. При очень больших n логарифм $\log(n)$ растет медленно, так что время работы алгоритма близко к линейному. Предложенный ими алгоритм был основан на фрактальной симметрии преобразования Фурье, которая позволяет выполнять n-мерное ДПФ с помощью вычисления нескольких ДПФ меньшей размерности. Хотя различие в вычислительной сложности ДПФ и БПФ может показаться несущественным, именно оценка $\mathcal{O}(n\log(n))$ позволила столь успешно применять БПФ для реализации связи в реальном времени, основанной на сжатии звука и изображений [539].

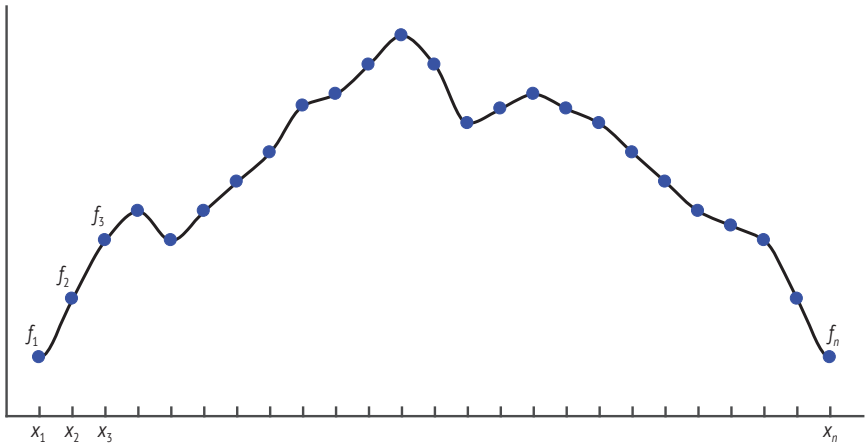

Рис. 2.7 ❖ Выборка данных для дискретного преобразования Фурье

Важно отметить, что Кули и Тьюки не открыли идею БПФ, потому что в предшествующие десятилетия было изучено много частных случаев, но именно они предложили общую формулировку, которая используется в настоящее время. Поразительно, что алгоритм БПФ был разработан еще Гауссом на 150 лет раньше, в 1805 году, когда для решения задачи о приближенном вычислении орбит астероидов Паллада и Юнона по данным измерений ему понадобилась очень точная схема интерполяции [239]. Поскольку Гаусс вычислял в уме и на бумаге, ему был нужен быстрый алгоритм, вот он и изобрел БПФ. Однако Гаусс не рассматривал это как серьезное достижение, и его выкладки были опубликованы только в 1866 году при разборе заметок [198]. Интересно, что открытие Гаусса было сделано еще до того, как Фурье в 1807 году объявил о разложении функций в тригонометрический ряд (опубликована эта работа была только в 1822 году [186]).

Дискретное преобразование Фурье

Хотя при вычислениях мы всегда будем использовать БПФ, поучительно начать с более простой формулировки ДПФ. Дискретное преобразование Фурье описывается формулой

$$\hat{f}_k = \sum_{j=0}^{n-1} f_j e^{-i2\pi jk/n}, \tag{2.26}$$

а обратное дискретное преобразование Фурье (оДПФ) – формулой

$$f_k = \frac{1}{n}\sum_{j=0}^{n-1} \hat{f}_j e^{i2\pi jk/n}. \tag{2.27}$$

Таким образом, ДПФ – линейный оператор (т. е. *матрица*), который отображает элементы $\mathbf{f}$ в частотную область $\hat{\mathbf{f}}$:

$$\{f_1,\ f_2,\ \cdots,\ f_n\} \overset{\text{ДПФ}}{\Rightarrow} \{\hat{f}_1,\ \hat{f}_2,\ \cdots,\ \hat{f}_n\}. \tag{2.28}$$

При заданном количестве точек n ДПФ представляет данные с помощью синусов и косинусов целых кратных основной частоты $\omega_n = e^{-2\pi i/n}$. ДПФ можно вычислить путем умножения на матрицу:

$$\begin{bmatrix} \hat{f}_1 \\ \hat{f}_2 \\ \hat{f}_3 \\ \vdots \\ \hat{f}_n \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)^2} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_n \end{bmatrix}. \tag{2.29}$$

Выходной вектор $\hat{\mathbf{f}}$ содержит коэффициенты Фурье для входного вектора $\mathbf{f}$, а матрица ДПФ $\mathbf{F}$ – это унитарная матрица Вандермонда. Матрица $\mathbf{F}$ комплексная, поэтому у выходного вектора $\hat{\mathbf{f}}$ есть абсолютная величина и фаза, и обе они имеют полезную физическую интерпретацию. Вещественная часть матрицы $\mathbf{F}$ показана на рис. 2.8 для $n = 256$. Для построения и изображения этой матрицы использовалась программа в листинге 2.2. По рисунку видно, что $\mathbf{F}$ обладает иерархической и в высшей степени симметричной многомасштабной структурой. Каждая строка и столбец – это косинус с увеличивающейся частотой.

Листинг 2.2 ❖ Создание матрицы дискретного преобразования Фурье

```
clear all, close all, clc
n = 256;
w = exp(-i*2*pi/n);

% Медленное
for i=1:n
    for j=1:n
        DFT(i,j) = w^((i-1)*(j-1));
    end
end

% Быстрое
```

```
[I,J] = meshgrid(1:n,1:n);
DFT = w.^((I-1).*(J-1));
imagesc(real(DFT))
```

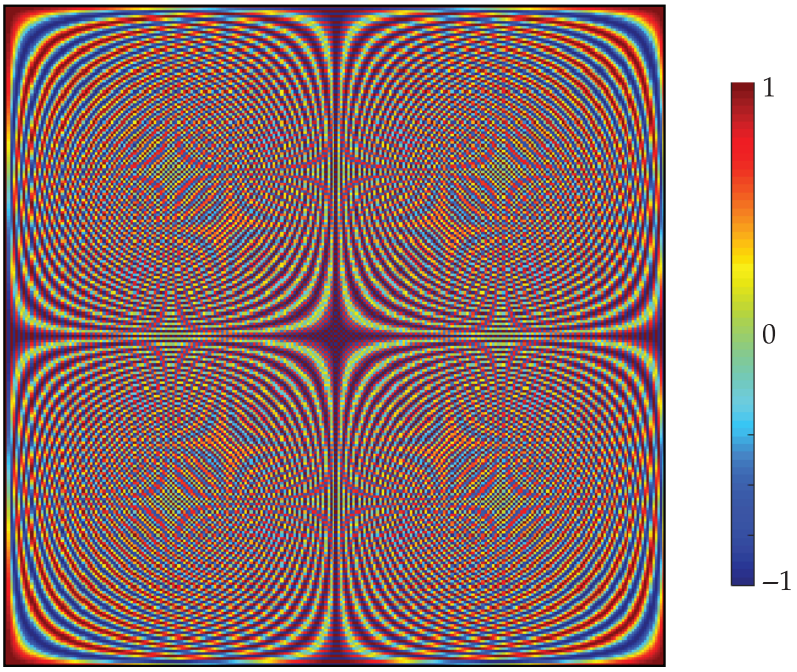

Рис. 2.8 ❖ Вещественная часть матрицы ДПФ при $n = 256$

Быстрое преобразование Фурье

Как было сказано выше, умножение на матрицу ДПФ **F** требует $\mathcal{O}(n^2)$ операций. Вычислительная сложность быстрого преобразования Фурье составляет $\mathcal{O}(n\log(n))$, что позволяет применять его в очень широком круге приложений, включая сжатие звука и изображений в форматах MP3 и JPG, потоковое видео, спутниковую связь и сотовые сети – и это лишь малая толика. Например, звук обычно дискретизируется с частотой 44.1 кГц, или 44 100 отсчетов в секунду. В случае 10-секундного звучания вектор **f** имел бы длину $n = 4.41\times10^5$. Вычисление ДПФ с использованием умножения на матрицу потребовало бы приблизительно 2×10^{11}, или 200 млрд операций умножения. С другой стороны, БПФ требует приблизительно 6×10^6 операций, т. е. налицо ускорение больше, чем в 30 000 раз. Поэтому, говоря о дискретном преобразовании Фурье, мы всегда имеем в виду БПФ, а библиотеки БПФ существуют практически для любого устройства и операционной системы, в которой производится цифровая обработка сигналов.

Чтобы воочию убедиться в колоссальном преимуществе БПФ, рассмотрим передачу, хранение и декодирование звукового сигнала. Ниже мы увидим, что многие сигналы отлично сжимаются в частотной области, т. е. большин-

ство коэффициентов Фурье $\hat{\mathbf{f}}$ настолько малы, что их можно отбросить. Тем самым обеспечивается гораздо более эффективное хранение и передача сжатого сигнала, поскольку нужно передавать лишь ненулевые коэффициенты. Но тогда должен быть способ быстро кодировать и декодировать сжатый сигнал путем вычисления БПФ и обратного БПФ. Для этого существуют такие команды:

```
>>fhat = fft(f);  % быстрое преобразование Фурье
>>f = ifft(fhat); % обратное быстрое преобразование Фурье
```

Основная идея БПФ заключается в том, что ДПФ можно реализовать гораздо эффективнее, если число точек n является степенью 2. Например, пусть $n = 1024 = 2^{10}$. В этом случае матрицу ДПФ $\mathbf{F}_{1024}$ можно записать в виде:

$$\hat{\mathbf{f}} = \mathbf{F}_{1024}\mathbf{f} = \begin{bmatrix} \mathbf{I}_{512} & -\mathbf{D}_{512} \\ \mathbf{I}_{512} & -\mathbf{D}_{512} \end{bmatrix} \begin{bmatrix} \mathbf{F}_{512} & 0 \\ 0 & \mathbf{F}_{512} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{\text{even}} \\ \mathbf{f}_{\text{odd}} \end{bmatrix}, \tag{2.30}$$

где $\mathbf{f}_{\text{even}}$ – элементы $\mathbf{f}$ с четными индексами, $\mathbf{f}_{\text{odd}}$ – элементы с нечетными индексами, $\mathbf{I}_{512}$ – единичная матрица 512×512, а $\mathbf{D}_{512}$ имеет вид

$$\mathbf{D}_{512} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \omega & 0 & \cdots & 0 \\ 0 & 0 & \omega^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \omega^{511} \end{bmatrix}. \tag{2.31}$$

Это выражение можно вывести путем перегруппировки членов формул (2.26) и (2.29). Если $n = 2^p$, то процедуру можно повторить и выразить $\mathbf{F}_{512}$ через $\mathbf{F}_{256}$ и так далее: $\mathbf{F}_{128} \to \mathbf{F}_{64} \to \mathbf{F}_{32} \to \cdots$. Если $n \neq 2^p$, то вектор можно дополнить нулями до длины, равной степени двойки. Таким образом, БПФ сводится к разделению элементов $\mathbf{f}$ с четными и нечетными индексами и вычислению нескольких меньших ДПФ.

Пример БПФ: фильтрация шума

Чтобы познакомиться с использованием и интерпретацией БПФ, рассмотрим для начала простой пример, в котором БПФ применяется для очистки сигнала от шума. Мы возьмем функцию времени $f(t)$:

$$f(t) = \sin(2\pi f_1 t) + \sin(2\pi f_2 t) \tag{2.32}$$

с частотами $f_1 = 50$ и $f_2 = 120$. Затем прибавим к этому сигналу сильный гауссов белый шум, как показано в верхней части рис. 2.9.

Вычислить быстрое преобразование Фурье зашумленного сигнала можно с помощью команды **fft**. Спектральная плотность мощности (СПМ, англ. PSD) – это нормированная квадратичная величина $\hat{\mathbf{f}}$, она показывает, какая часть мощности сигнала приходится на каждую частоту. Из средней части

рис. 2.9 видно, что зашумленный сигнал содержит два острых пика на частотах 50 Гц и 120 Гц. Мы можем обнулить компоненты, мощность которых ниже пороговой, и тем самым очистить сигнал от шума. После обратного преобразования профильтрованного сигнала обнаруживается, что чистый и профильтрованный временной ряд хорошо согласуются (рис. 2.9, снизу). Программа в листинге 2.3 выполняет все описанные шаги и строит графики.

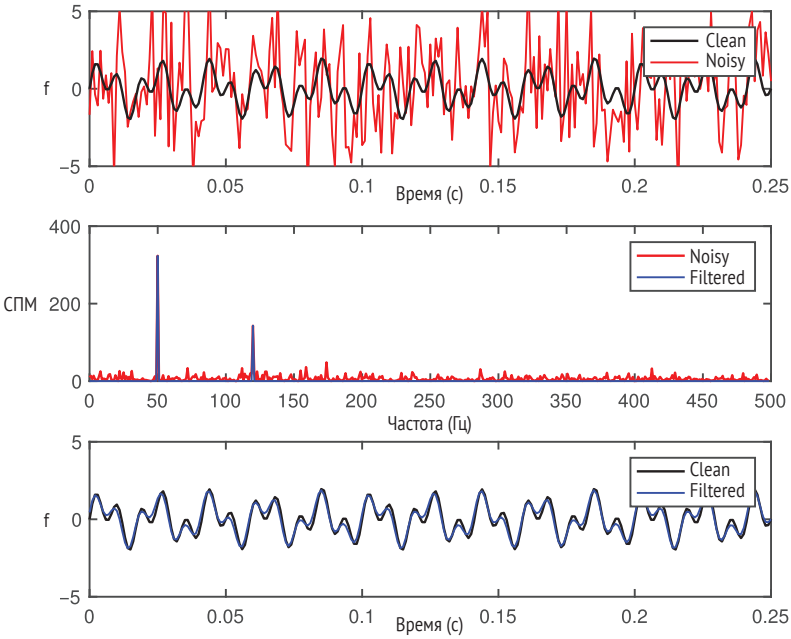

Рис. 2.9 ❖ Очистка от шума с помощью БПФ: (сверху) добавлен шум к простому сигналу в виде суммы двух синусоид; (в середине) в частотной области можно выделить острые пики и отфильтровать шум; (снизу) очищенный сигнал получается применением обратного преобразования Фурье к двум доминирующим пикам

Листинг 2.3 ❖ Применение быстрого преобразования Фурье для очистки сигнала от шума

```
dt = .001;
t = 0:dt:1;
f = sin(2*pi*50*t) + sin(2*pi*120*t); % сумма двух частот
f = f + 2.5*randn(size(t)); % добавить шум

%% Вычисление быстрого преобразования Фурье
n = length(t);
fhat = fft(f,n);   % вычислить быстрое преобразование Фурье
PSD = fhat.*conj(fhat)/n; % спектр мощности (части мощности в частотах)
freq = 1/(dt*n)*(0:n); % построить ось x - частота в Гц
L = 1:floor(n/2); % нанести на график только первую половину частот

%% Использовать СПМ для фильтрации шума
```

```
indices = PSD>100; % найти частоты с большой мощностью
PSDclean = PSD.*indices; % обнулить все остальные
fhat = indices.*fhat; % обнулить малые коэффициенты Фурье
ffilt = ifft(fhat); % обратное БПФ для профильтрованного сигнала

%% Графики
subplot(3,1,1)
plot(t,f,'r','LineWidth',1.2), hold on
plot(t,f,'k','LineWidth',1.5)
legend('Noisy','Clean')

subplot(3,1,2)
plot(t,f,'k','LineWidth',1.5), hold on
plot(t,ffilt,'b','LineWidth',1.2)
legend('Clean','Filtered')

subplot(3,1,3)
plot(freq(L),PSD(L),'r','LineWidth',1.5), hold on
plot(freq(L),PSDclean(L),'-b','LineWidth',1.2)
legend('Noisy','Filtered')
```

Пример БПФ: спектральные производные

В следующем примере мы продемонстрируем применение БПФ для быстрого и точного вычисления производных. Из формулы (2.19) следует, что непрерывное преобразование Фурье обладает свойством $\mathcal{F}(df/dx) = i\omega\mathcal{F}(f)$. Аналогично численную производную вектора дискретизированных данных можно хорошо аппроксимировать, умножив каждую компоненту дискретного преобразования Фурье $\mathbf{f}$ на $i\kappa$, где $\kappa = 2\pi k/n$ – дискретное волновое число, ассоциированное с этой компонентой. Благодаря точности и эффективности вычисления спектральной производной она весьма полезна при решении дифференциальных уравнений в частных производных, как мы увидим в следующем разделе.

Для демонстрации так называемой *спектральной* производной возьмем функцию $f(x)$, для которой производную можно вычислить аналитически:

$$f(x) = \cos(x)e^{-x^2/25} \quad \Rightarrow \quad \frac{df}{dx}(x) = -\sin(x)e^{-x^2/25} - \frac{2}{25}xf(x). \tag{2.33}$$

На рис. 2.10 показаны результаты сравнения спектральной производной с вычисленной аналитически и методом прямых конечных разностей Эйлера с $n = 128$ точками дискретизации:

$$\frac{df}{dx}(x_k) \approx \frac{f(x_{k+1}) - f(x_k)}{\Delta x}. \tag{2.34}$$

Погрешность обеих схем дифференцирования можно уменьшить, увеличив n, это то же самое, что уменьшение Δx. Однако погрешность спектральной производной с ростом n уменьшается быстрее, чем в конечноразностной схеме, что показано на рис. 2.11. Известно, что метод прямых конечных

разностей Эйлера неточен, и его погрешность пропорциональна $\mathcal{O}(\Delta x)$, но даже увеличение порядка конечноразностной схемы не дает такого же повышения точности, как в случае спектральной производной, при вычислении которой фактически используется информация из всей области определения. Программа в листинге 2.4 вычисляет и сравнивает обе схемы дифференцирования.

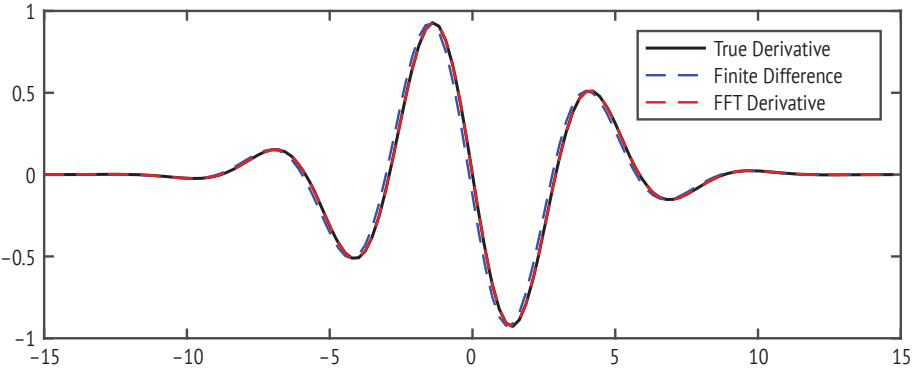

Рис. 2.10 ❖ Сравнение спектральной производной, вычисленной методом БПФ, и производной, вычисленной методом конечных разностей

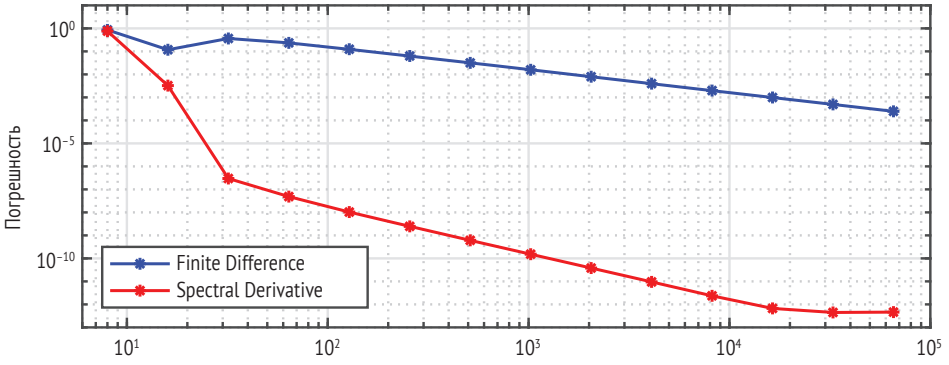

Рис. 2.11 ❖ Поведение спектральной производной при изменении степени дискретизации данных

Листинг 2.4 ❖ Применение быстрого преобразования Фурье для вычисления производных

```
n = 128;
L = 30;
dx = L/(n);
x = -L/2:dx:L/2-dx;
f = cos(x).*exp(-x.^2/25); % функция
df = -(sin(x).*exp(-x.^2/25) + (2/25)*x.*f); % производная

%% Аппроксимация производной методом конечных разностей
```

```
for kappa=1:length(df)-1
    dfFD(kappa) = (f(kappa+1)-f(kappa))/dx;
end
dfFD(end+1) = dfFD(end);

%% Вычисление производной методом БПФ (спектральная производная)
fhat = fft(f);
kappa = (2*pi/L)*[-n/2:n/2-1];
kappa = fftshift(kappa); % переупорядочить частоты БПФ
dfhat = i*kappa.*fhat;
dfFFT = real(ifft(dfhat));

%% Построение графиков
plot(x,df,'k','LineWidth',1.5), hold on
plot(x,dfFD,'b--','LineWidth',1.2)
plot(x,dfFFT,'r--','LineWidth',1.2)
legend('True Derivative','Finite Diff.','FFT Derivative')
```

Если производная функции разрывна, то в спектральной производной будет иметь место явление Гиббса, как показано на рис. 2.12.

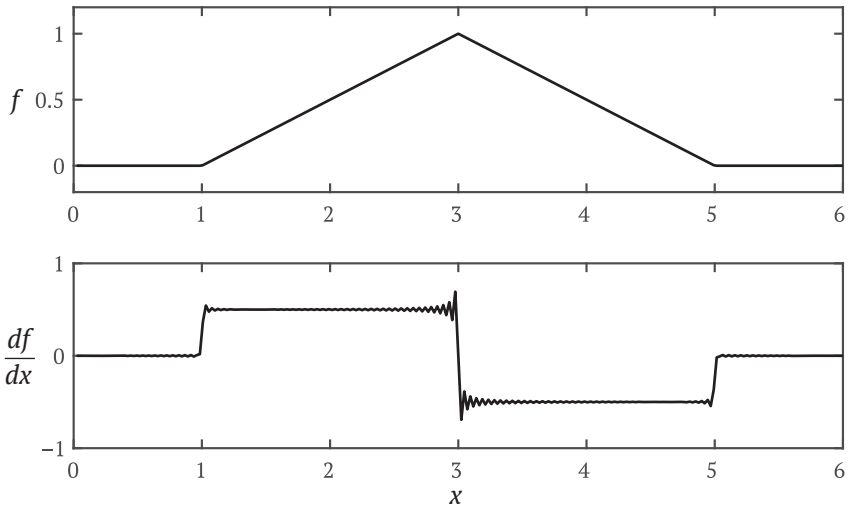

Рис. 2.12 ❖ Явление Гиббса
для спектральной производной функции с разрывной производной

2.3. Преобразование дифференциальных уравнений в частных производных

Преобразование Фурье изначально было предложено в 1800-х годах как замена координат в уравнении теплопроводности, при которой динамика распадается на части. В общем случае преобразование Фурье полезно для пре-

образования дифференциальных уравнений в частных производных (УрЧП) обыкновенными дифференциальными уравнениями (ОДУ), как в формуле (2.20). В этом разделе мы продемонстрируем применение БПФ к численному решению ряда УрЧП. Великолепное изложение спектральных методов для УрЧП см. в книге Trefethen [523]; существуют также обобщения на жесткие УрЧП [282].

Уравнение теплопроводности

Базис преобразования Фурье идеально подходит для решения уравнения теплопроводности. Одномерное уравнение теплопроводности имеет вид

$$u_t = \alpha^2 u_{xx}, \tag{2.35}$$

где $u(t, x)$ – распределение температуры во времени и пространстве. После пространственного преобразования Фурье получаем $\mathcal{F}(u(t, x)) = \hat{u}(t, \omega)$. Теперь УрЧП (2.35) принимает вид:

$$\hat{u}t = -\alpha^2\omega^2\hat{u}, \tag{2.36}$$

поскольку две пространственные производные привносят множитель $(i\omega)^2 = -\omega^2$ в частотную область. Таким образом, после выполнения преобразования Фурье УрЧП PDE (2.35) становится ОДУ для каждой частоты ω. Решение этого уравнения имеет вид:

$$\hat{u}(t, \omega) = e^{-\alpha^2\omega^2 t}\hat{u}(0, \omega). \tag{2.37}$$

Функция $\hat{u}(0, \omega)$ – это преобразование Фурье начального распределения температуры $u(0, x)$. Теперь видно, что чем больше частота, т. е. значение ω, тем быстрее убывает решение со временем, поэтому острые углы в распределении температуры быстро сглаживаются. Для вычисления обратного преобразования Фурье можно воспользоваться свойством свертки (2.24):

$$u(t,x) = \mathcal{F}^{-1}(\hat{u}(t,\omega)) = \mathcal{F}^{-1}\left(e^{-\alpha^2\omega^2 t}\right) * u(0,x) = \frac{1}{2\alpha\sqrt{\pi t}} e^{-\frac{x^2}{4\alpha^2 t}} * u(0,x). \tag{2.38}$$

При численном моделировании УрЧП проще и точнее сначала перейти в частотную область с помощью БПФ. В данном случае (2.36) принимает вид

$$\hat{u}_t = -\alpha^2\kappa^2\hat{u}, \tag{2.39}$$

где κ – дискретизированная частота. По принятым в MATLAB соглашениям, необходимо вызвать команду **fftshift** для переупорядочения волновых чисел.

Программа в листинге 2.5 моделирует одномерное уравнение теплопроводности с применением БПФ, результаты показаны на рис. 2.13 и 2.14. В данном примере УрЧП линейное, поэтому с помощью **ode45** можно продолжить систему непосредственно в частотной области, пользуясь векторным полем в листинге 2.6. Наконец, в листинге 2.7 приведены команды построения графиков.

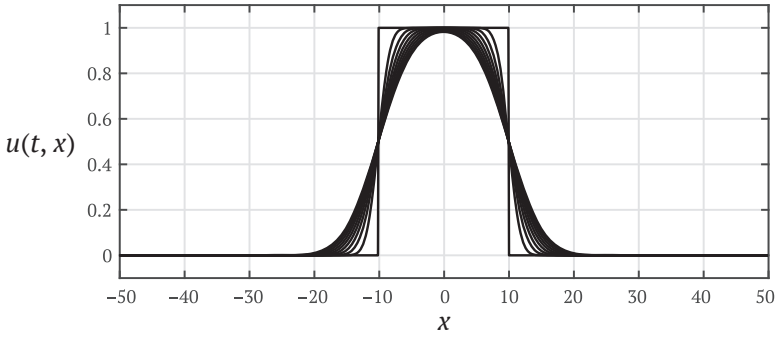

Рис. 2.13 ❖ Зависимость решения одномерного уравнения теплопроводности от времени при начальном условии, заданном кусочно-постоянной функцией. С течением времени острые углы быстро сглаживаются и решение стремится к гауссовой функции

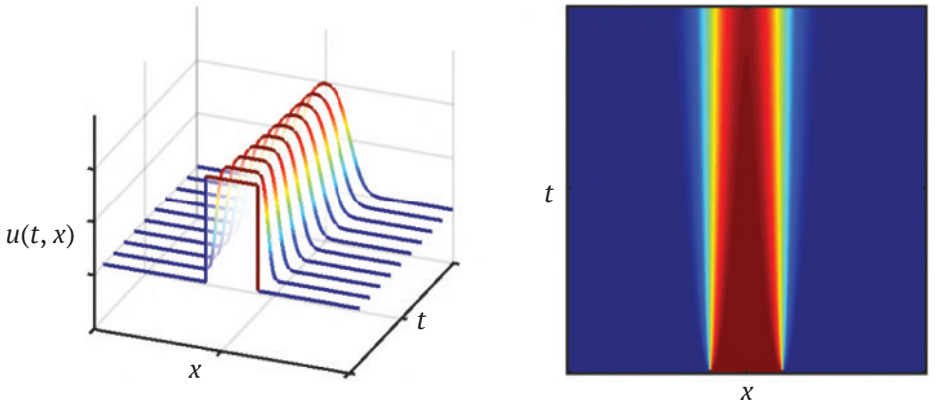

Рис. 2.14 ❖ Изменение решения одномерного уравнения теплопроводности в зависимости от времени. Представлено водопадной диаграммой (слева) и *x*-*t* диаграммой (справа)

На рис. 2.13 и 2.14 показано несколько представлений распределения температуры $u(t, x)$ в зависимости от времени. На рис. 2.13 мы видим несколько кривых, соответствующих разным моментам времени на одном графике. Те же данные показаны на рис. 2.14 в виде водопадной диаграммы (слева) и *x*-*t* диаграммы (справа). На всех рисунках хорошо видно, что острые углы быстро сглаживаются, поскольку соответствуют наибольшим волновым числам. В конечном итоге изменение наименьших волновых чисел тоже сходит на нет, и температура выходит на стационарное распределение, являющееся решением уравнения Лапласа $u_{xx} = 0$. При решении этого УрЧП методом БПФ мы неявно предполагаем, что область определения решения конечна, так что правая и левая границы точно определены, но периодически продолжается в обе стороны. Однако если область определения достаточно велика, то влияние границ мало.

Листинг 2.5 ❖ Программа моделирования одномерного уравнения теплопроводности с помощью преобразования Фурье

```
a = 1;              % постоянный коэффициент тепловой диффузии
L = 100;            % длина области определения
N = 1000;           % количество точек дискретизации
dx = L/N;
x = -L/2:dx:L/2-dx; % задать область определения по x

% Определить дискретные волновые числа
kappa = (2*pi/L)*[-N/2:N/2-1];
kappa = fftshift(kappa);  % переупорядочить волновые числа БПФ

% Начальное условие
u0 = 0*x;
u0((L/2 - L/10)/dx:(L/2 + L/10)/dx) = 1;

% Моделирование в частотной области
t = 0:0.1:10;
[t,uhat]=ode45(@(t,uhat)rhsHeat(t,uhat,kappa,a),t,fft(u0));

for k = 1:length(t) % обПФ для возврата в пространственную область
    u(k,:) = ifft(uhat(k,:));
end
```

Листинг 2.6 ❖ Правая часть одномерного уравнения теплопроводности в частотной области, *dû/dt*

```
function duhatdt = rhsHeat(t,uhat,kappa,a)
duhatdt = -a^2*(kappa.^2)'.*uhat; % Linear and diagonal
```

Листинг 2.7 ❖ Программа построения графиков решений одномерного уравнения теплопроводности

```
figure, waterfall((u(1:10:end,:)));
figure, imagesc(flipud(u));
```

Одностороннее волновое уравнение

В качестве второго примера рассмотрим простое линейное УрЧП – одностороннее волновое уравнение:

$$u_t + cu_x = 0. \tag{2.40}$$

Любое начальное условие $u(0, x)$ просто распространяется вправо во времени со скоростью c, поскольку $u(t, x) = u(0, x - ct)$ является решением. Программа в листинге 2.8 моделирует это УрЧП для начального условия, заданного в виде колоколообразного импульса. Это уравнение можно проинтегрировать в частотной области, как и раньше, воспользовавшись векторным полем, которое строит программа в листинге 2.9. Однако его можно проинтегрировать и в пространственной области, просто воспользовавшись

БПФ для вычисления производных, а затем выполнив обратное преобразование, как в листинге 2.10. Решение $u(t, x)$ изображено на рис. 2.15 и 2.16.

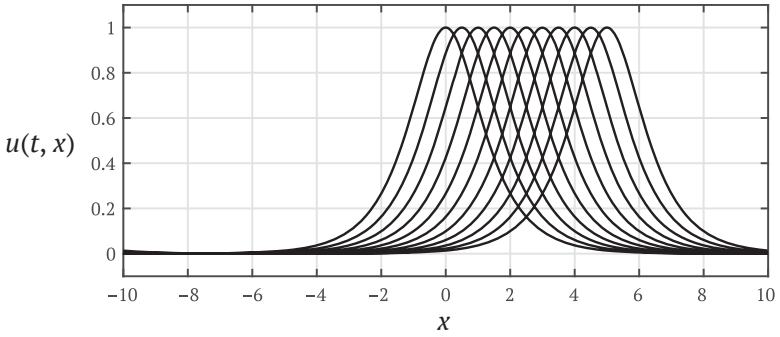

Рис. 2.15 ❖ Зависимость решения одностороннего волнового уравнения от времени. С течением времени начальное условие распространяется слева направо с постоянной скоростью волны

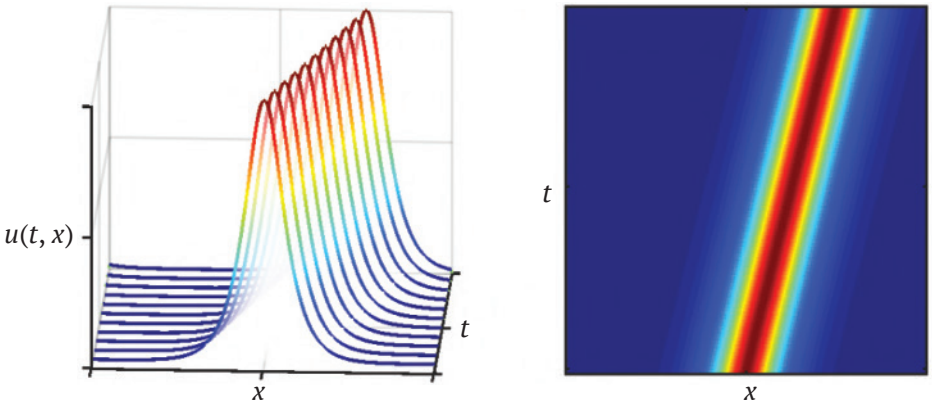

Рис. 2.16 ❖ Изменение решения одностороннего волнового уравнения в зависимости от времени. Представлено водопадной диаграммой (слева) и *x*-*t* диаграммой (справа)

Листинг 2.8 ❖ Программа моделирования одностороннего волнового уравнения с помощью преобразования Фурье

```
c = 2;                  % скорость волны
L = 20;                 % длина области определения
N = 1000;               % количество точек дискретизации
dx = L/N;
x = -L/2:dx:L/2-dx; % задать область определения по x

% Определить дискретные волновые числа
kappa = (2*pi/L)*[-N/2:N/2-1];
```

```
kappa = fftshift(kappa'); % переупорядочить волновые числа БПФ

% Начальное условие
u0 = sech(x);
uhat0 = fft(u0);

% Моделирование в частотной области
dt = 0.025;
t = 0:dt:100*dt;
[t,uhat] = ode45(@(t,uhat)rhsWave(t,uhat,kappa,c),t,uhat0);

% Альтернативно - моделирование в пространственной области
[t,u] = ode45(@(t,u)rhsWaveSpatial(t,u,kappa,c),t,u0);
```

Листинг 2.9 ❖ Правая часть одностороннего волнового уравнения в частотной области

```
function duhatdt = rhsWave(t,uhat,kappa,c)
duhatdt = -c*i*kappa.*uhat;
```

Листинг 2.10 ❖ Правая часть одностороннего волнового уравнения в пространственной области

```
function dudt = rhsWaveSpatial(t,u,kappa,c)
uhat = fft(u);
duhat = i*kappa.*uhat;
du = ifft(duhat);
dudt = -c*du;
```

Уравнение Бюргерса

И напоследок рассмотрим нелинейное уравнение Бюргерса

$$u_t + uu_x = \nu u_{xx}, \tag{2.41}$$

описывающее простую одномерную нелинейную конвекцию и диффузию, которая приводит к ударным волнам в жидкостях [253]. Из-за нелинейной конвекции uu_x передняя кромка волны становится круче, поскольку для частей u с большими амплитудами конвекция развивается быстрее, что приводит к формированию фронта ударной волны.

Программа в листинге 2.11 моделирует уравнение Бюргерса и строит визуализации, показанные на рис. 2.17 и 2.18. Уравнение Бюргерса – интересный пример применения БПФ, поскольку из-за нелинейности мы должны переходить в частотную область и обратно на каждом временном шаге, как показывает векторное поле в листинге 2.12. В этом примере мы переходим в частотную область для вычисления u_x и u_{xx}, а затем возвращаемся в пространственную область для вычисления произведения uu_x. На рис. 2.17 и 2.18 ясно видно, что эффект увеличения крутизны приводит к формированию ударной волны. Не будь демпфирующего члена u_{xx}, ударная волна стала бы бесконечно крутой, но благодаря его наличию ширина остается конечной.

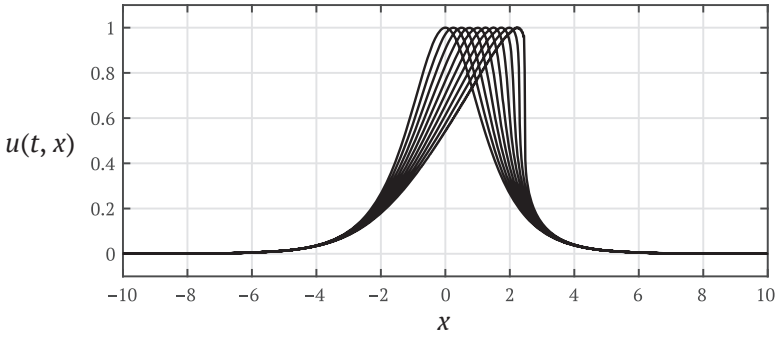

Рис. 2.17 ❖ Зависимость решения уравнения Бюргерса от времени. С течением времени передняя кромка колоколообразного начального условия становится круче, формируя фронт ударной волны

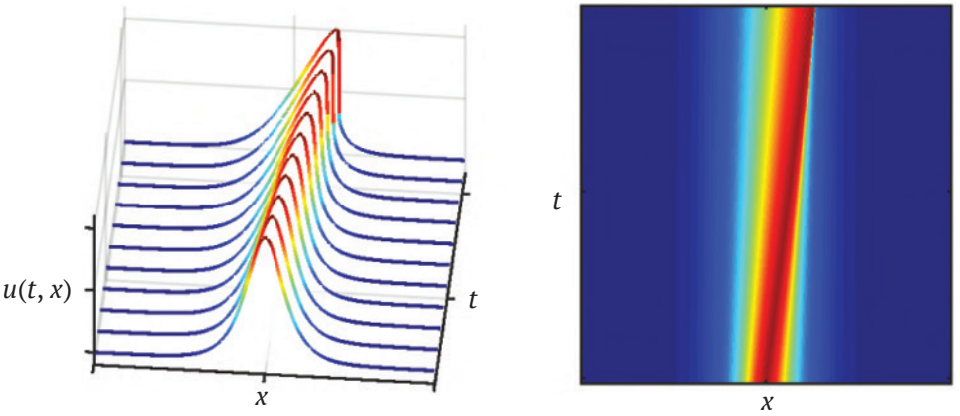

Рис. 2.18 ❖ Изменение решения уравнения Бюргерса в зависимости от времени. Представлено водопадной диаграммой (слева) и *x*-*t* диаграммой (справа)

Листинг 2.11 ❖ Программа моделирования уравнения Бюргерса с помощью преобразования Фурье

```
clear all, close all, clc
nu=0.001; % постоянная диффузии

% Определить пространственную область
L = 20;               % длина области опеределения
N = 1000;             % количество точек дискретизации
dx = L/N;
x = -L/2:dx:L/2-dx;   % задать область определения по x

% Определить дискретные волновые числа
kappa = (2*pi/L)*[-N/2:N/2-1];
kappa = fftshift(kappa'); % переупорядочить волновые числа БПФ

% Начальное условие
u0 = sech(x);
```

```
% Моделирование в пространственной области
dt = 0.025;
t = 0:dt:100*dt;
[t,u] = ode45(@(t,u)rhsBurgers(t,u,kappa,nu),t,u0);
```

Листинг 2.12 ❖ Правая часть уравнения Бюргерса в частотной области

```
function dudt = rhsBurgers(t,u,kappa,nu)
uhat = fft(u);
duhat = i*kappa.*uhat;
dduhat = -(kappa.^2).*uhat;
du = ifft(duhat);
ddu = ifft(dduhat);
dudt = -u.*du + nu*ddu;
```

2.4. Преобразование Габора и спектрограмма

Хотя преобразование Фурье дает детальную информацию о частотном составе данного сигнала, оно ничего не говорит о том, в какие моменты времени эти частоты возникают. Преобразование Фурье может охарактеризовать только истинно периодические и стационарные сигналы, поскольку в результате интегрирования в (2.18а) время исключается. Для сигнала с нестационарным частотным составом, например музыкальной композиции, важно одновременно иметь характеристику частотного состава и его изменения во времени.

Преобразование Габора, известное также как оконное преобразование Фурье (ОПФ), вычисляет БПФ в скользящем окне [437, 262, 482], как показано на рис. 2.19. ОПФ позволяет локализовать частотный состав во времени и получить *спектрограмму*, т. е. график зависимости частоты от времени, показанный на рис. 2.21 и 2.22. ОПФ вычисляется по формуле

$$\mathcal{G}(f)(t,\omega) = \hat{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)e^{-i\omega\tau}\overline{g}(\tau - t)d\tau = \langle f, g_{t,\omega}\rangle, \tag{2.42}$$

где $g_{t,\omega}(\tau)$ определено как

$$g_{t,\omega}(\tau) = e^{i\omega\tau}g(\tau - t). \tag{2.43}$$

Функция $g(t)$ называется ядром, в этом качестве часто используется гауссова функция:

$$g(t) = e^{-(t-\tau)^2/a^2}. \tag{2.44}$$

Параметр a определяет ширину скользящего окна преобразования Фурье, а τ – его центр.

Обратное ОПФ описывается формулой:

$$f(t) = \mathcal{G}^{-1}\left(\hat{f}_g(t,\omega)\right) = \frac{1}{2\pi||g||^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\hat{f}_g(t,\omega)g(t-\tau)e^{i\omega t}d\omega dt. \tag{2.45}$$

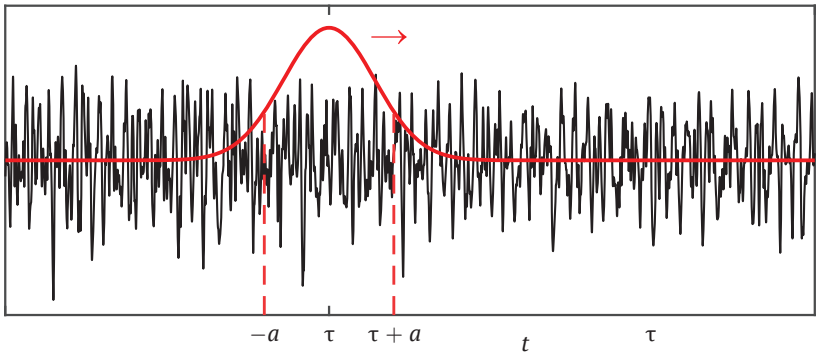

Рис. 2.19 ❖ Преобразование Габора,
или оконное преобразование Фурье со скользящим гауссовым окном

Дискретное преобразование Габора

Чаще всего преобразование Габора применяется к дискретным сигналам, как и БПФ. Но в этом случае необходимо дискретизировать одновременно время и частоту:

$$\nu = j\Delta\omega, \tag{2.46}$$

$$\tau = k\Delta t. \tag{2.47}$$

Дискретизированная ядерная функция принимает вид:

$$g_{j,k} = e^{i2\pi j\Delta\omega t} g(t - k\Delta t), \tag{2.48}$$

а дискретное преобразование Габора –

$$\hat{f}_{j,k} = \langle f, g_{j,k} \rangle = \int_{-\infty}^{\infty} f(\tau)\bar{g}_{j,k}(\tau)d\tau. \tag{2.49}$$

Этот интеграл можно затем аппроксимировать конечной римановой суммой дискретизированных функций f и $\bar{g}_{j,k}$.

Пример: сигнал с квадратичной частотной модуляцией

В качестве простого примера построим осциллирующую функцию, определяемую как косинус с частотой колебаний, квадратично зависящей от времени:

$$f(t) = \cos(2\pi t\omega(t)), \text{ где } \omega(t) = \omega_0 + (\omega_1 - \omega_0)t^2/3t_1^2. \tag{2.50}$$

Частота изменяется от ω_0 при $t = 0$ до ω_1 при $t = t_1$.

На рис. 2.20 показана спектральная плотность мощности, полученная в результате применения БПФ к сигналу с квадратичной частотной модуляцией.

Несмотря на отчетливый пик на частоте 50 Гц, мы не имеем никакой информации об изменении частоты во времени. Программа построения спектрограммы приведена в листинге 2.13, а результирующая спектрограмма показана на рис. 2.21, где видно, что частотный состав сдвигается во времени.

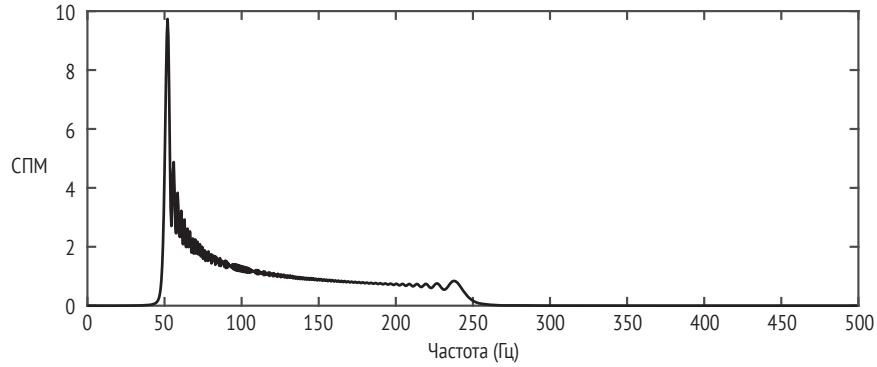

Рис. 2.20 ❖ Спектральная плотность мощности сигнала с квадратичной частотной модуляцией

Листинг 2.13 ❖ Спектрограмма сигнала с квадратичной частотной модуляцией, показанная на рис. 2.21

```
t = 0:0.001:2;
f0 = 50;
f1 = 250;
t1 = 2;
x = chirp(t,f0,t1,f1,'quadratic');
x = cos(2*pi*t.*(f0 + (f1-f0)*t.^2/(3*t1^2)));
% В документации по MATLAB опечатка...
% ... поделить на 3, чтобы амплитуда производной соответствовала частоте
spectrogram(x,128,120,128,1e3,'yaxis')
```

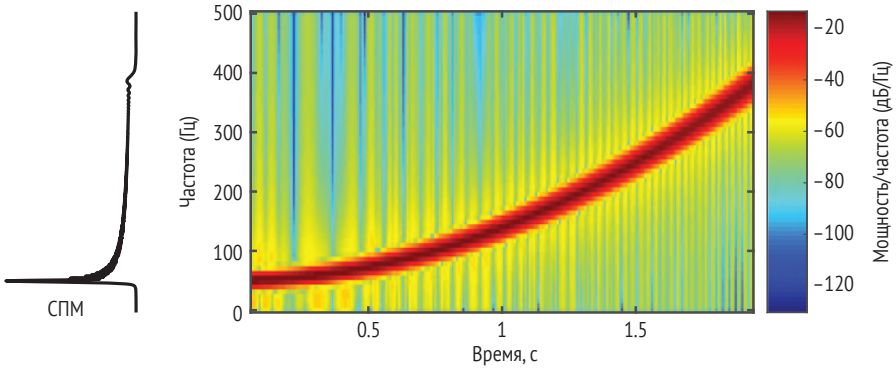

Рис. 2.21 ❖ Спектрограмма сигнала с квадратичной частотной модуляцией. Слева показана СПМ, соответствующая мощности, просуммированной по строкам спектрограммы

Пример: «Патетическая соната» Бетховена

С помощью спектрограммы мы можем проанализировать более сложные сигналы, например «Патетическую сонату» Бетховена, ноты которой показаны на рис. 2.22. Спектрограммы широко используются для анализа музыки и недавно были включены в алгоритм программы Shazam, которая ищет основные признаки в спектрограмме музыкального произведения и быстро идентифицирует его по коротким фрагментам записи [545].

На рис. 2.22 показано начало «Патетической сонаты» Бетховена и его спектрограмма. На спектрограмме отчетливо видны различные аккорды и гармоники. Увеличив масштаб частоты, мы увидим две октавы, а также начало и окончание отдельных нот. Программа в листинге 2.14 загружает данные, вычисляет спектрограмму и выводит результат.

Листинг 2.14 ❖ Вычисление спектрограммы «Патетической сонаты» Бетховена (рис. 2.22)

```
% Скачать mp3read с http://www.mathworks.com/matlabcentral/
% fileexchange/13852-mp3read-and-mp3write
[Y,FS,NBITS,OPTS] = mp3read('beethoven.mp3');

%% Построить спектрограмму командой ‘spectrogram'
T = 40;      % 40 секунд
y=Y(1:T*FS); % первые 40 секунд
spectrogram(y,5000,400,24000,24000,'yaxis');

%% Спектрограмма с помощью оконного преобразования Фурье ‘stft'
wlen = 5000; % ширина окна
h=400;       % перекрытие равно wlen - h
% Произвести анализ зависимости частоты от времени
[S,f,t_stft] = stft(y, wlen, h, FS/4, FS); % y axis 0-4000HZ

imagesc(log10(abs(S))); % нарисовать спектрограмму (в логарифм. масштабе)
```

Чтобы обратить спектрограмму и воссоздать оригинальное звучание, напишем:

```
[x_istft, t_istft] = istft(S, h, FS/4, FS);
sound(x_istft,FS);
```

Художники, например Эйфекс Твин, использовали обращение спектрограмм изображений для генерирования музыки. Частоту любой клавиши фортепиано тоже легко вычислить. Например, частота 40-й клавиши вычисляется так:

```
freq = @(n)(((2^(1/12))^(n-49))*440);
freq(40)  % частота 40-й клавиши = до
```

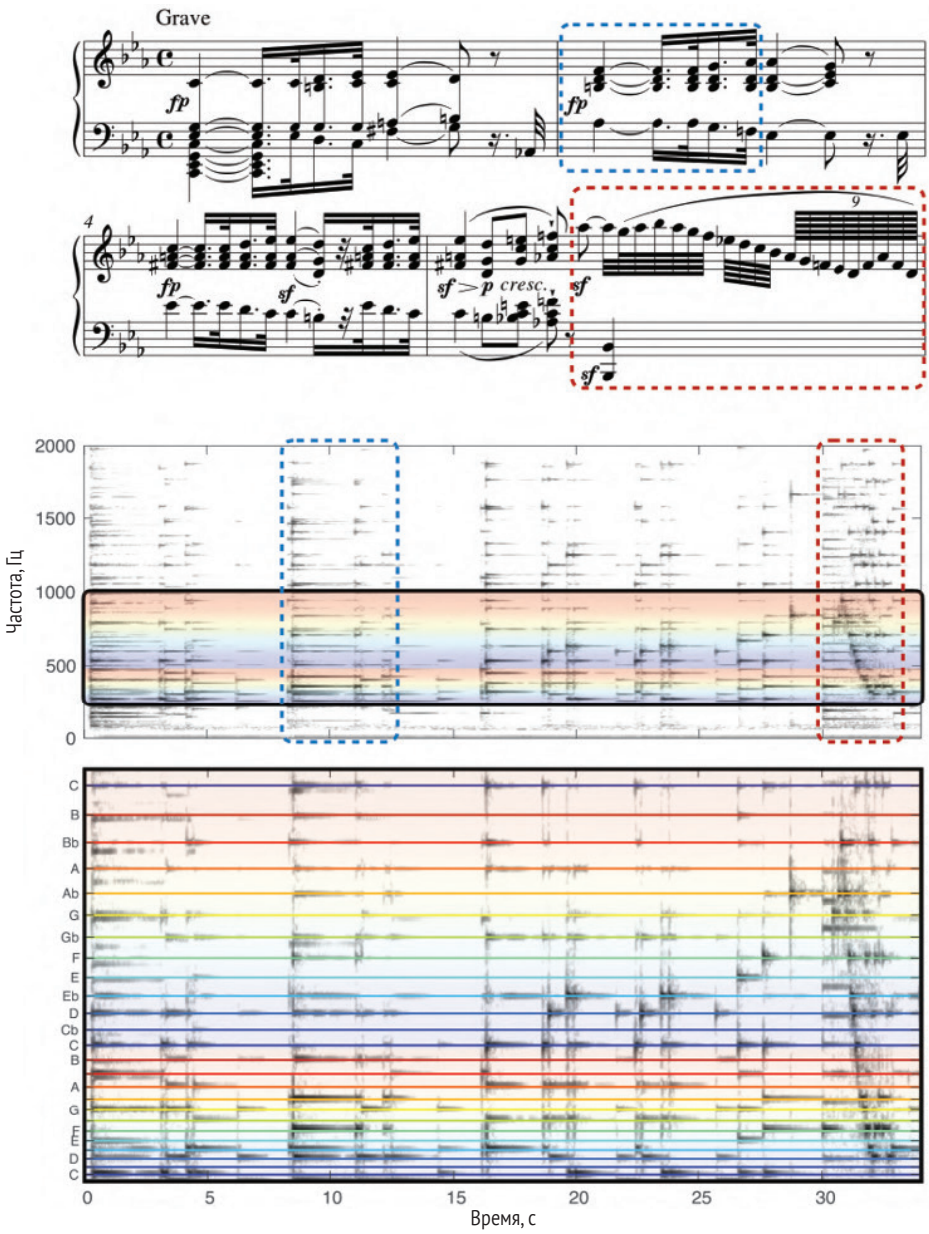

Рис. 2.22 ❖ Начало «Патетической сонаты» Бетховена (соната для фортепиано № 8 до минор, оп. 13) и ее аннотированная спектрограмма

Принцип неопределенности

В частотно-временном анализе имеет место фундаментальный принцип неопределенности, ограничивающий возможность достижения высокой точности одновременно в частотной и временной областях. В его крайнем проявлении мы имеем временной ряд, который обладает идеальным разрешением по времени, но не дает никакой информации о частотном составе, и преобразование Фурье, которое идеально определяет частотный состав, но ничего не говорит о том, в какой момент возникает та или иная частота. Спектрограмма дает информацию и о частоте, и о времени, но разрешение в обеих областях низкое, как показано на рис. 2.23. Альтернативный подход, основанный на многомасштабном анализе, будет рассмотрен в следующем разделе.

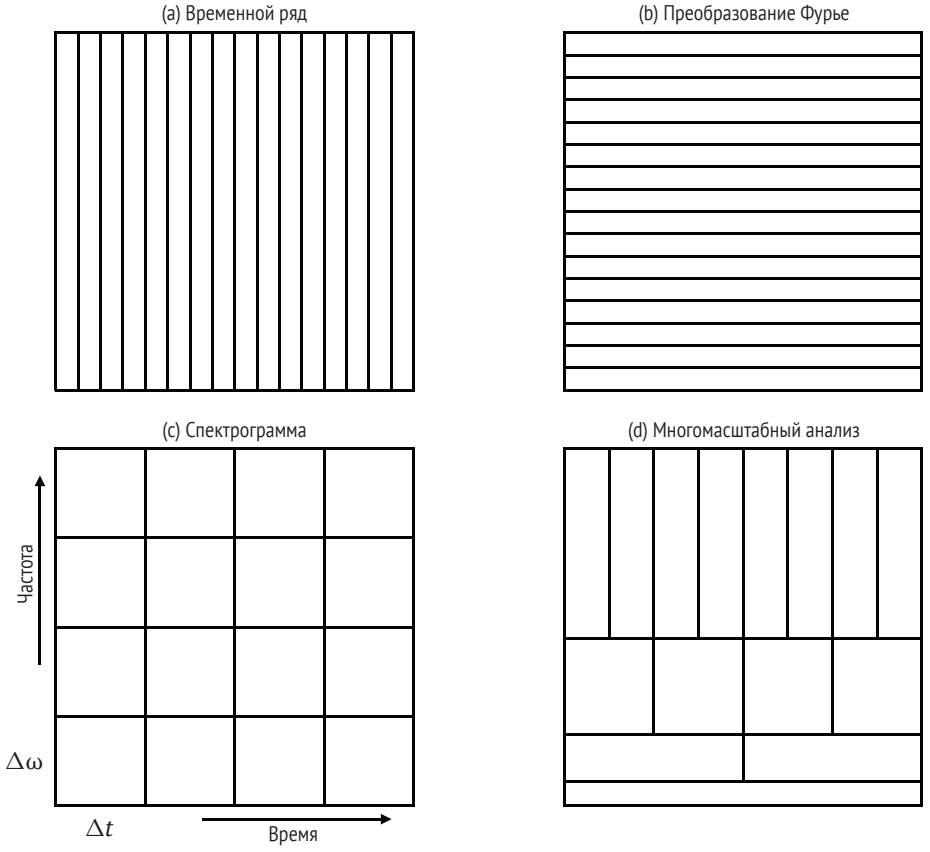

Рис. 2.23 ❖ Иллюстрация ограничения на разрешение и принципа неопределенности в частотно-временном анализе

Математически принцип частотно-временной неопределенности [429] можно сформулировать в виде:

$$\left(\int_{-\infty}^{\infty} x^2|f(x)|^2dx\right)\left(\int_{-\infty}^{\infty} \omega^2|\hat{f}(\omega)|^2d\omega\right) \geq \frac{1}{16\pi^2}. \tag{2.51}$$

Это неравенство справедливо, если $f(x)$ абсолютно непрерывна и $xf(x)$ и $f'(x)$ квадратично интегрируемы. Функция $x^2|f(x)|^2$ описывает рассеивание в окрестности $x = 0$. Для вещественных функций это второй момент, который измеряет дисперсию, если $f(x)$ – гауссова функция. Иными словами, функцию $f(x)$ и ее преобразование Фурье невозможно локализовать одновременно с произвольной точностью. Если функция f стремится к дельта-функции, то преобразование Фурье обязательно становится широкополосным, и наоборот. Это имеет прямое отношение к принципу неопределенности Гейзенберга [240], поскольку волновые функции положения и импульса являются парой Фурье.

В частотно-временном анализе принцип неопределенности налагает ограничение на возможности локализации преобразования Фурье во времени. Это ограничение называется пределом Габора. Если частотный состав сигнала разрешается с большей точностью, то мы теряем информацию о том, когда происходят события, и наоборот. Таким образом, принципиально невозможно одновременно достичь высокого разрешения в частотной и временной областях. Еще одно следствие заключается в том, что функция f и ее преобразование Фурье не могут одновременно иметь конечный носитель, т. е. функция не может быть одновременно ограниченной в диапазоне времени и в диапазоне частоты, как утверждает теорема Бенедика [8, 51].

2.5. Вейвлеты и многомасштабный анализ

Вейвлеты [359, 145] распространяют идеи анализа Фурье на более общие ортогональные базисы и позволяют частично преодолеть вышеупомянутый принцип неопределенности за счет использования многомасштабного разложения, как показано на рис. 2.23 (d). Этот подход дает возможность иметь разную точность по времени и по частоте в различных частотных диапазонах, что особенно полезно при разложении сложных сигналов, возникающих в многомасштабных процессах, встречающихся в климатологии, нейронауках, эпидемиологии, финансах и изучении турбулентности. Изображения и звуковые сигналы также поддаются вейвлет-анализу, который в настоящее время является основным методом сжатия изображений [16], о чем мы еще будем говорить ниже в этой и следующих главах. Кроме того, вейвлет-преобразования можно вычислять с помощью похожих быстрых методов [58], благодаря чему они масштабируются на данные высокой размерности. Существует несколько отличных книг по вейвлетам [521, 401, 357], помимо основных учебников [359, 145].

Основная идея вейвлет-анализа – начать с функции $\psi(t)$, называемой *материнским* вейвлетом, и построить семейство, получающееся применением к этой функции сдвига и масштабирования:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right). \tag{2.52}$$

Параметры a и b определяют величину масштабирования и сдвига функции ψ соответственно. Например, можно выбрать a и b, так чтобы масштабированная и сдвинутая функция помещалась в один сегмент на рис. 2.23 (d). Если эти функции ортогональны, то базис можно использовать для проецирования, как в преобразовании Фурье.

Простейший и самый ранний пример вейвлета – вейвлет *Хаара*, предложенный в 1910 году [227]:

$$\psi(t) = \begin{cases} 1 & 0 \le t \le 1/2 \\ -1 & 1/2 \le t < 1 \\ 0 & \text{в противном случае} \end{cases}. \tag{2.53}$$

Все три вейвлета Хаара, $\psi_{1,0}$, $\psi_{1/2,0}$ и $\psi_{1/2,1/2}$, показаны на рис. 2.24. Они представляют первые два уровня масштабов на рис. 2.23 (d). Заметим, что если уровень каждой следующей частоты получается делением предыдущего уровня пополам, то результирующие вейвлеты Хаара будут ортогональны и, значит, образуют иерархический базис сигнала.

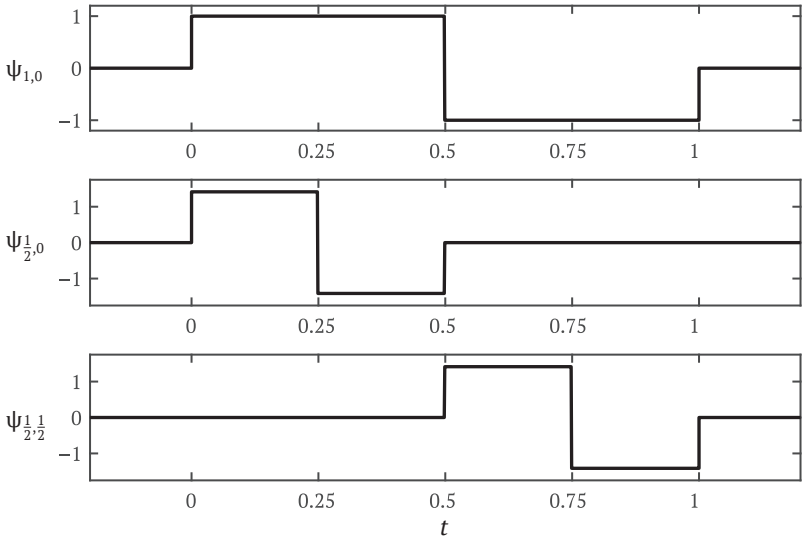

Рис. 2.24 ❖ Три вейвлета Хаара для первых двух уровней масштаба на рис. 2.23 (d)

Описанное выше свойство ортогональности вейвлетов критически важно для разработки дискретного вейвлет-преобразования (discrete wavelet transform – DWT) ниже. Но начнем мы с непрерывного вейвлет-преобразования (continuous wavelet transform – CWT), определяемого формулой

$$\mathcal{W}_{\psi}(f)(a,b) = \langle f, \psi_{a,b} \rangle = \int_{-\infty}^{\infty} f(t)\bar{\psi}_{a,b}(t)dt, \tag{2.54}$$

где $\bar{\psi}_{a,b}$ обозначает комплексное сопряжение $\psi_{a,b}$. Это справедливо только для функций $\psi(t)$, удовлетворяющих свойству ограниченности:

$$C_{\psi} = \int_{-\infty}^{\infty} \frac{|\bar{\psi}(\omega)|^2}{|\omega|} d\omega < \infty. \tag{2.55}$$

Обратное вейвлет-преобразование (iCWT) вычисляется по формуле:

$$f(t) = \frac{1}{C_{\psi}} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathcal{W}_{\psi}(f)(a,b)\psi_{a,b}(t)\frac{1}{a^2} da\, db. \tag{2.56}$$

Новые вейвлеты можно также генерировать применением свертки $\psi * \varphi$ при условии, что ψ – вейвлет, а φ – ограниченная интегрируемая функция. Помимо вейвлета Хаара, есть много других материнских вейвлетов ψ, обладающих различными свойствами. Например, вейвлет «сомбреро» (Mexican hat wavelet) определен следующим образом:

$$\psi(t) = (1 - t^2)e^{-t^2/2}; \tag{2.57a}$$

$$\hat{\psi}(\omega) = \sqrt{2\pi}\omega^2 e^{-\omega^2/2}. \tag{2.57b}$$

Дискретное вейвлет-преобразование

Как и в случае преобразований Фурье и Габора, при вычислении вейвлет-преобразования данных необходима дискретизированная версия. Дискретное вейвлет-преобразование (DWT) определяется по формуле:

$$\mathcal{W}_{\psi}(f)(j,k) = \langle f, \psi_{j,k} \rangle = \int_{-\infty}^{\infty} f(t)\,\bar{\psi}_{j,k}(t)dt, \tag{2.58}$$

где $\psi_{j,k}(t)$ – дискретное семейство вейвлетов:

$$\psi_{j,k}(t) = \frac{1}{a^j}\psi\left(\frac{t - kb}{a^j}\right). \tag{2.59}$$

И снова, если это семейство ортогонально, как описанные выше вейвлеты Хаара, то существует и единственное разложение функции $f(t)$ по этому базису:

$$f(t) = \sum_{j,k=-\infty}^{\infty} \langle f(t), \psi_{j,k}(t) \rangle \psi_{j,k}(t). \tag{2.60}$$

Явное вычисление DWT устроено довольно сложно, ему посвящено несколько прекрасных статей и учебников [359, 145, 521, 401, 357]. Но наша цель – не вдаваться в детали вычисления, а дать общее представление о том, чего позволяет достичь вейвлет-преобразование. Путем масштабирования

и сдвига заданной фигуры поперек сигнала мы можем эффективно выделять многомасштабные структуры и строить из них иерархию, обеспечивающую оптимальный компромисс между временным и пространственным разрешениями. Эта общая процедура широко применяется в обработке изображений и звука, сжатии информации, научных расчетах и машинном обучении – и это далеко не все примеры.

2.6. Двумерные преобразования и обработка сигналов

Хотя мы анализировали преобразование Фурье и вейвлет-преобразование для одномерных сигналов, оба метода легко обобщаются на двумерные и трехмерные сигналы. Преобразование Фурье и вейвлет-преобразование оказали огромное влияние на обработку и сжатие изображений, и это достаточная побудительная причина для исследования преобразований в пространствах высокой размерности.

Двумерное преобразование Фурье для изображений

Для двумерного преобразования Фурье матрицы $\mathbf{X} \in \mathbb{R}^{n \times m}$ нужно сначала применить одномерное преобразование Фурье к каждой строке матрицы, а затем к каждому столбцу получившейся промежуточной матрицы. Это последовательное преобразование строк и столбцов показано на рис. 2.25. Если изменить порядок, т. е. сначала применить преобразование Фурье к строкам, а потом к столбцам, то результат не изменится.

Листинг 2.15 ❖ Выполнение двумерного преобразования Фурье путем применения одномерного БПФ к строкам и столбцам

```
A = imread('../../CH01_SVD/DATA/dog.jpg');
B = rgb2gray(A); % преобразовать в полутоновое изображение
subplot(1,3,1), imagesc(B); % нарисовать изображение

for j=1:size(B,1); % применить БПФ к строкам
    Cshift(j,:) = fftshift(fft(B(j,:)));
    C(j,:) = (fft(B(j,:)));
end
subplot(1,3,2), imagesc(log(abs(Cshift)))

for j=1:size(C,2); % применить БПФ к столбцам
   D(:,j) = fft(C(:,j));
end
subplot(1,3,3), imagesc(fftshift(log(abs(D))))

D = fft2(B); % гораздо эффективнее воспользоваться функцией fft2
```

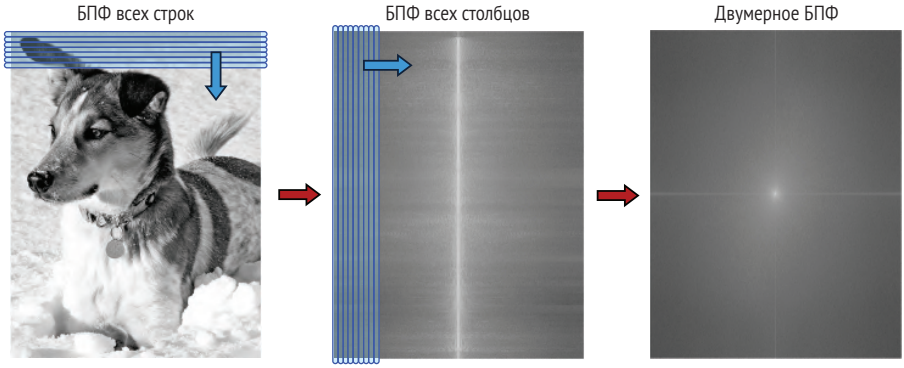

Рис. 2.25 ❖ Схема двумерного БПФ.
Сначала вычисляется БПФ для каждой строки,
затем для каждого столбца промежуточной матрицы

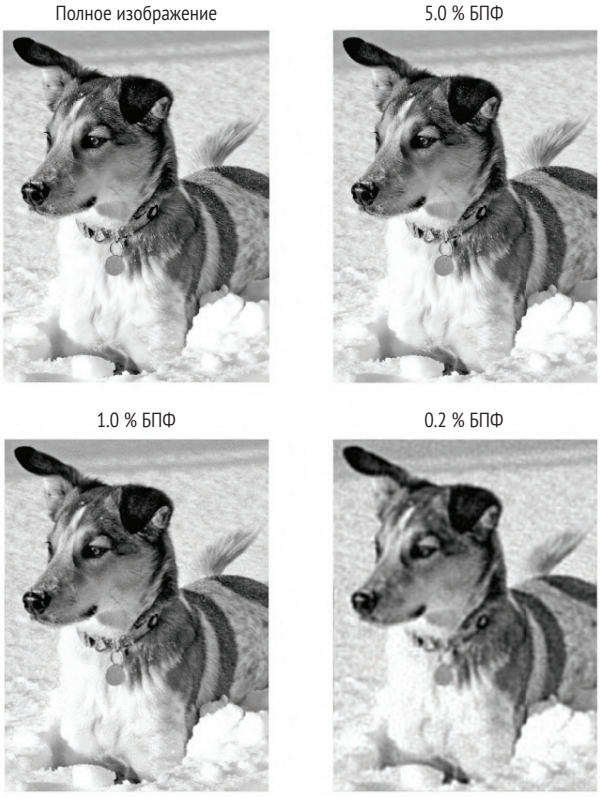

Рис. 2.26 ❖ Изображение, сжатое с различными порогами:
5 %, 1 % и 0.2 % – наибольших коэффициентов Фурье

Двумерное БПФ эффективно для сжатия изображений, потому что многие коэффициенты Фурье малы и ими можно пренебречь без потери качества. А хранить и передавать нужно лишь немногие большие коэффициенты Фурье.

Листинг 2.16 ❖ Сжатие изображений посредством БПФ

```
Bt=fft2(B); % B - полутоновое изображение, построенное выше
Btsort = sort(abs(Bt(:))); % отсортировать по абсолютной величине

% Отбросить малые коэффициенты и обратить преобразование
for keep=[.1 .05 .01 .002];
    thresh = Btsort(floor((1-keep)*length(Btsort)));
    ind = abs(Bt)>thresh;      % найти малые индексы
    Atlow = Bt.*ind;           % отбросить малые индексы
    Alow=uint8(ifft2(Atlow));  % сжатое изображение
    figure, imshow(Alow)       % нарисовать результат реконструкции
end
```

Наконец, БПФ часто применяется для очистки от шума и фильтрации сигналов, поскольку выделить и обработать конкретные частотные полосы очень просто. В листинге 2.17 и на рис. 2.27 демонстрируется применение порогового фильтра для очистки изображения, к которому был добавлен гауссов шум. В этом примере шум особенно заметен на верхних частотах, поэтому мы отбрасываем коэффициенты Фурье, не принадлежащие заданному диапазону нижних частот.

Листинг 2.17 ❖ Очистка изображения от шума с помощью БПФ

```
Bnoise = B + uint8(200*randn(size(B))); % добавить шум
Bt=fft2(Bnoise);
F = log(abs(Btshift)+1); % перевести БПФ в логарифмический масштаб

subplot(2,2,1), imagesc(Bnoise) % нарисовать исходное изображение
subplot(2,2,2), imagesc(F)      % нарисовать результат БПФ

[nx,ny] = size(B);
[X,Y] = meshgrid(-ny/2+1:ny/2,-nx/2+1:nx/2);
R2 = X.^2+Y.^2;
ind = R2<150^2;

Btshiftfilt = Btshift.*ind;
Ffilt = log(abs(Btshiftfilt)+1); % перевести БПФ в логарифмический масштаб
subplot(2,2,4), imagesc(Ffilt)   % нарисовать БПФ результата фильтрации
Btfilt = ifftshift(Btshiftfilt);

Bfilt = ifft2(Btfilt);
subplot(2,2,3), imagesc(uint8(real(Bfilt))) % профильтрованное изображение
```

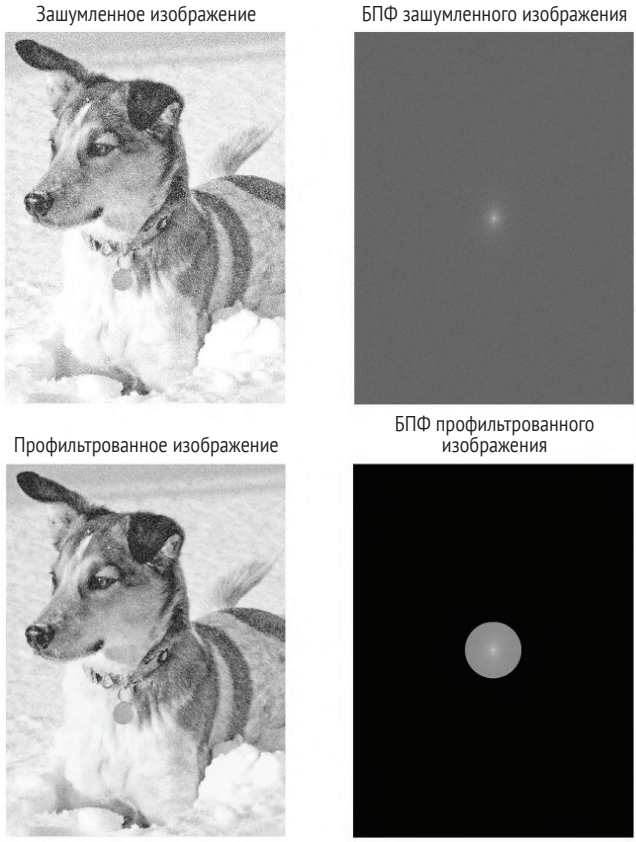

Рис. 2.27 ❖ Очистка изображения от шума путем отбрасывания высокочастотных коэффициентов Фурье, оказавшихся вне заданного диапазона (справа внизу)

Двумерное вейвлет-преобразование изображений

Как и БПФ, дискретное вейвлет-преобразование широко применяется для обработки и сжатия изображений. Программа в листинге 2.18 вычисляет вейвлет-преобразование изображения, а на рис. 2.28 показаны первые три уровня. Хорошо видна иерархическая природа разложения по вейвлетам. В верхнем левом углу правого рисунка показано то же изображение в низком разрешении, а последующие признаки добавляют более точные детали.

Листинг 2.18 ❖ Пример двухуровневого разложения по вейвлетам

```
%% Разложение по вейвлетам (двухуровневое)
n = 2; w = 'db1'; [C,S] = wavedec2(B,n,w);
```

```
% Уровень 1
A1 = appcoef2(C,S,w,1); % аппроксимация
[H1 V1 D1] = detcoef2('a',C,S,k); % детали
A1 = wcodemat(A1,128);
H1 = wcodemat(H1,128);
V1 = wcodemat(V1,128);
D1 = wcodemat(D1,128);

% Уровень 2
A2 = appcoef2(C,S,w,1); % аппроксимация
[H2 V2 D2] = detcoef2('a',C,S,k); % детали
A2 = wcodemat(A2,128);
H2 = wcodemat(H2,128);
V2 = wcodemat(V2,128);
D2 = wcodemat(D2,128);

dec2 = [A2 H2; V2 D2];
dec1 = [imresize(dec2,size(H1)) H1 ; V1 D1];
image(dec1);
```

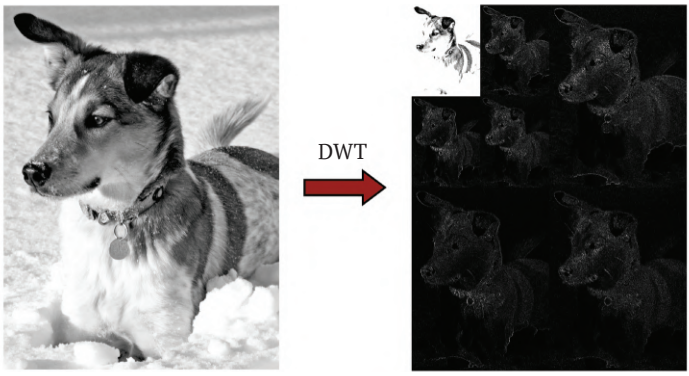

Рис. 2.28 ❖ Три уровня дискретного вейвлет-преобразования

На рис. 2.29 показано несколько вариантов сжатого изображения при различных коэффициентах сжатия, вычисленных программой в листинге 2.19. Иерархическое представление данных при вейвлет-преобразовании идеально для сжатия изображений. Даже при агрессивном сжатии, когда остается только 0.5 % коэффициентов DWT, основные признаки изображения сохраняются. Поэтому даже при передаче данных по каналу ограниченной пропускной способности, когда много информации отбрасывается, большинство важных признаков данных передаются.

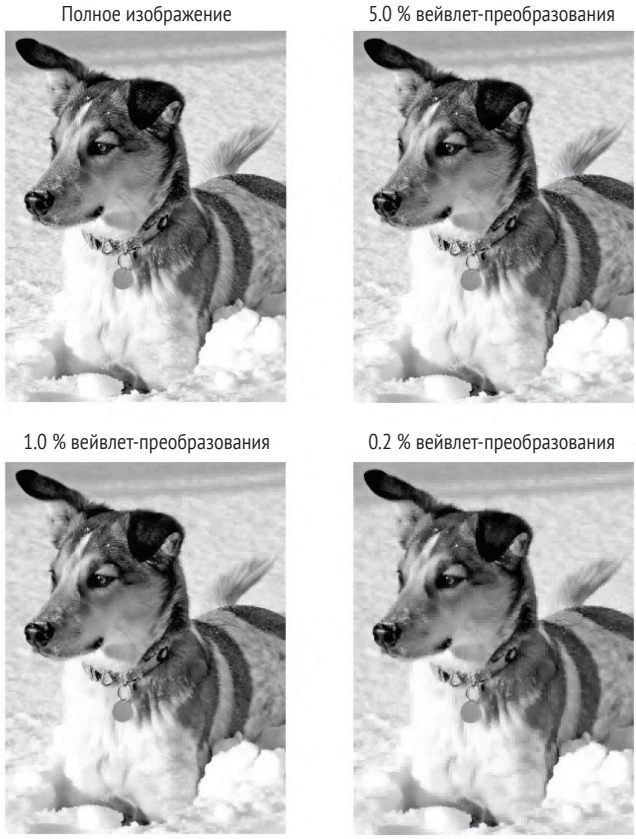

Рис. 2.29 ❖ Изображение, сжатое с различными порогами: 5 %, 1 % и 0.5 % – наибольших коэффициентов вейвлет-преобразования

Листинг 2.19 ❖ Применение вейвлет-преобразования для сжатия изображений

```
[C,S] = wavedec2(B,4,'db1');
Csort = sort(abs(C(:))); % отсортировать по абсолютной величине

for keep = [.1 .05 .01 .005]
    thresh = Csort(floor((1-keep)*length(Csort)));
    ind = abs(C)>thresh;
    Cfilt = C.*ind; % отбросить малые индексы

    % Нарисовать реконструкцию
    Arecon=uint8(waverec2(Cfilt,S,'db1'));
    figure, imagesc(uint8(Arecon))
end
```

Рекомендуемая литература

Книги

(1) The analytical theory of heat, by J.-B. J. Fourier, 1978 [185].
(2) A wavelet tour of signal processing, by S. Mallat, 1999 [357].
(3) Spectral methods in MATLAB, by L. N. Trefethen, 2000 [523].

Статьи и обзоры

(1) An algorithm for the machine calculation of complex Fourier series, by J. W. Cooley and J. W. Tukey, *Mathematics of Computation*, 1965 [137].
(2) The wavelet transform, time-frequency localization and signal analysis, by I. Daubechies, *IEEE Transactions on Information Theory*, 1990 [145].
(3) An industrial strength audio search algorithm, by A. Wang et al., *Ismir*, 2203 [545].

Глава 3

Разреженность и сжатие измерений

Наблюдаемая структура естественных данных позволяет предположить, что данные допускают разреженное представление в подходящей системе координат. Иными словами, если выразить естественные данные в удачно выбранном базисе, то понадобится лишь небольшое число параметров, чтобы описать, какие моды активны и в какой пропорции. Сжатие данных целиком и полностью зависит от разреженности, когда сигнал более эффективно представляется разреженным вектором коэффициентов в базисе некоторого общего преобразования, например в базисе Фурье или в вейвлетном базисе. Недавние фундаментальные математические результаты перевернули эту парадигму вверх тормашками. Вместо того чтобы собирать данные измерений высокой размерности, а затем сжимать их, теперь можно сжимать данные *в процессе* измерения и искать самый разреженный многомерный сигнал, согласующийся с этими измерениями. Это так называемое *сжатие измерений* (compressed sensing) – новый полезный подход, имеющий приложения и к сложным инженерным системам, потенциально он способен совершить революцию в сборе и обработке данных. В этой главе мы на конкретных примерах обсудим фундаментальные принципы разреженности и сжатия, а также математическую теорию, лежащую в основе сжатия измерений.

Обсуждение разреженности и сжатия измерений не может обойтись без некоторых важнейших вопросов оптимизации и статистики. Разреженность – полезный стимул для развития *экономных* моделей, которые избегают переобучения и сохраняют возможность интерпретации, поскольку для объяснения данных в них используется минимальное количество параметров. Это одно из проявлений принципа бритвы Оккама, утверждающего, что самое простое объяснение обычно оказывается правильным. Разреженная оптимизация полезна также для повышения устойчивости к выбросам и недостающим данным – проблемам, которые искажают результаты регрессии методом наименьших квадратов, например SVD. Вопросы, рассматриваемые в данной главе, тесно связаны с рандомизированной линейной алгеброй, обсуждавшейся в разделе 1.8, и будут также использованы в нескольких последующих главах. Разреженную регрессию мы будем изучать и в главе 4, а в разделе 7.3 применим для идентификации интерпретируемых и эконом-

ных моделей нелинейных динамических систем, построенных на основе анализа данных.

3.1. Разреженность и сжатие

Большинство естественных сигналов, в т. ч. изображения и звук, очень хорошо поддаются сжатию. Это означает, что если записать сигнал в подходящем базисе, то активно будет небольшое число мод, поэтому количество значений, которые нужно хранить для точного представления, уменьшается. Иначе говоря, сжимаемый сигнал $\mathbf{x} \in \mathbb{R}^n$ можно записать в виде разреженного вектора $\mathbf{s} \in \mathbb{R}^n$ (содержащего в основном нули) в базисе преобразования $\mathbf{\Psi} \in \mathbb{R}^{n \times n}$:

$$\mathbf{x} = \mathbf{\Psi s}. \tag{3.1}$$

Точнее, вектор $\mathbf{s}$ называется K-разреженным в $\mathbf{\Psi}$, если он содержит ровно K ненулевых элементов. Если $\mathbf{\Psi}$ – базис общего вида, например Фурье или вейвлетный, то для реконструкции исходного сигнала $\mathbf{x}$ требуется лишь небольшое число активных элементов $\mathbf{s}$, что уменьшает объем данных, необходимых для хранения или передачи сигнала.

Изображения и звуковые сигналы допускают сжатие в базисе Фурье и в вейвлетном базисе, поскольку после выполнения соответствующего преобразования большинство коэффициентов мало, и их можно считать нулевыми почти без потери качества. Поэтому вместо исходного многомерного сигнала достаточно хранить и передавать лишь немногие активные коэффициенты. Затем для реконструкции исходного сигнала в объемлющем пространстве (например, в пространстве пикселей в случае изображения) нужно будет просто выполнить обратное преобразование. В главе 2 мы говорили, что быстрое преобразование Фурье – именно та технология, которая открывает возможность эффективной реконструкции изображения $\mathbf{x}$ по разреженным коэффициентам $\mathbf{s}$. Она лежит в основе алгоритмов JPEG и MP3, предназначенных соответственно для сжатия изображений и звука.

Моды Фурье и вейвлеты являются *общими*, или *универсальными*, базисами в том смысле, что почти все естественные изображения и звуковые сигналы оказываются в этих базисах разреженными. Поэтому после сжатия сигнала нужно хранить или передавать только вектор $\mathbf{s}$, а не матрицу $\mathbf{\Psi}$ целиком, поскольку программно-аппаратные реализации преобразования Фурье и вейвлет-преобразования уже имеются на большинстве машин. В главе 1 мы узнали, что сигналы можно сжимать также с помощью сингулярного разложения, порождающего *специальный* базис. На самом деле использовать SVD для сжатия изображений можно двумя способами: 1) вычислять SVD изображения непосредственно и сохранять только доминирующие столбцы $\mathbf{U}$ и $\mathbf{V}$ (раздел 1.2) или 2) представить изображение в виде линейной комбинации *собственных* изображений, как в примере с собственными лицами (раздел 1.6). Первый вариант менее эффективен, потому что необходимо хранить

векторы базиса **U** и **V**. Во втором же варианте специальный базис **U** можно вычислить и сохранить один раз, а затем использовать для сжатия целого класса изображений, например человеческих лиц. У такого специального базиса есть дополнительное преимущество – моды допускают интерпретацию в виде корреляционных признаков, которые могут оказаться полезны для обучения. Важно отметить, что и базис Фурье $\mathcal{F}$, и SVD-базис **U** – унитарные преобразования, это будет важно в последующих разделах.

Хотя основные достижения теории сжатия связаны с обработкой звука, изображений и видео, у нее есть много применений в технических системах. Решение многомерной системы дифференциальных уравнений обычно располагается на многообразии меньшей размерности, что указывает на существование когерентных структур, допускающих разреженное представление. Даже такие широкополосные явления, как турбулентность, в каждый момент времени можно охарактеризовать разреженным представлением. Это оказало большое влияние на способы измерения и вычислений, о чем мы будем говорить в этой и последующих главах.

Пример: сжатие изображения

Сжатие изображений реализовать сравнительно просто, мы говорили об этом в разделе 2.6 и повторим сейчас (см. рис. 3.1). Сначала мы загружаем изображение, преобразуем его в полутоновое и рисуем:

```
A=imread('jelly', 'jpeg');   % загрузить изображение
Abw=rgb2gray(A);             % преобразовать в полутоновое
imshow(Abw).                 % нарисовать изображение
```

Затем выполним быстрое преобразование Фурье и построим график коэффициентов в логарифмическом мастшабе:

```
At=fft2(Abw);
F = log(abs(fftshift(At))+1); % перевести БПФ в логарифмический масштаб
imshow(mat2gray(F),[]);
```

Для сжатия изображения мы сначала сортируем коэффициенты Фурье по абсолютной величине и решаем, какой процент оставлять (в данном случае 5 %). Тем самым задается порог отсечения:

```
Bt = sort(abs(At(:)));
keep = 0.05;
thresh = Bt(floor((1-keep)*length(Bt)));
ind = abs(At)>thresh;
Atlow = At.*ind;
```

Наконец, рисуем сжатое изображение, предварительно выполнив обратное БПФ (оБПФ):

```
Alow=uint8(ifft2(Atlow));
imshow(Alow)
```

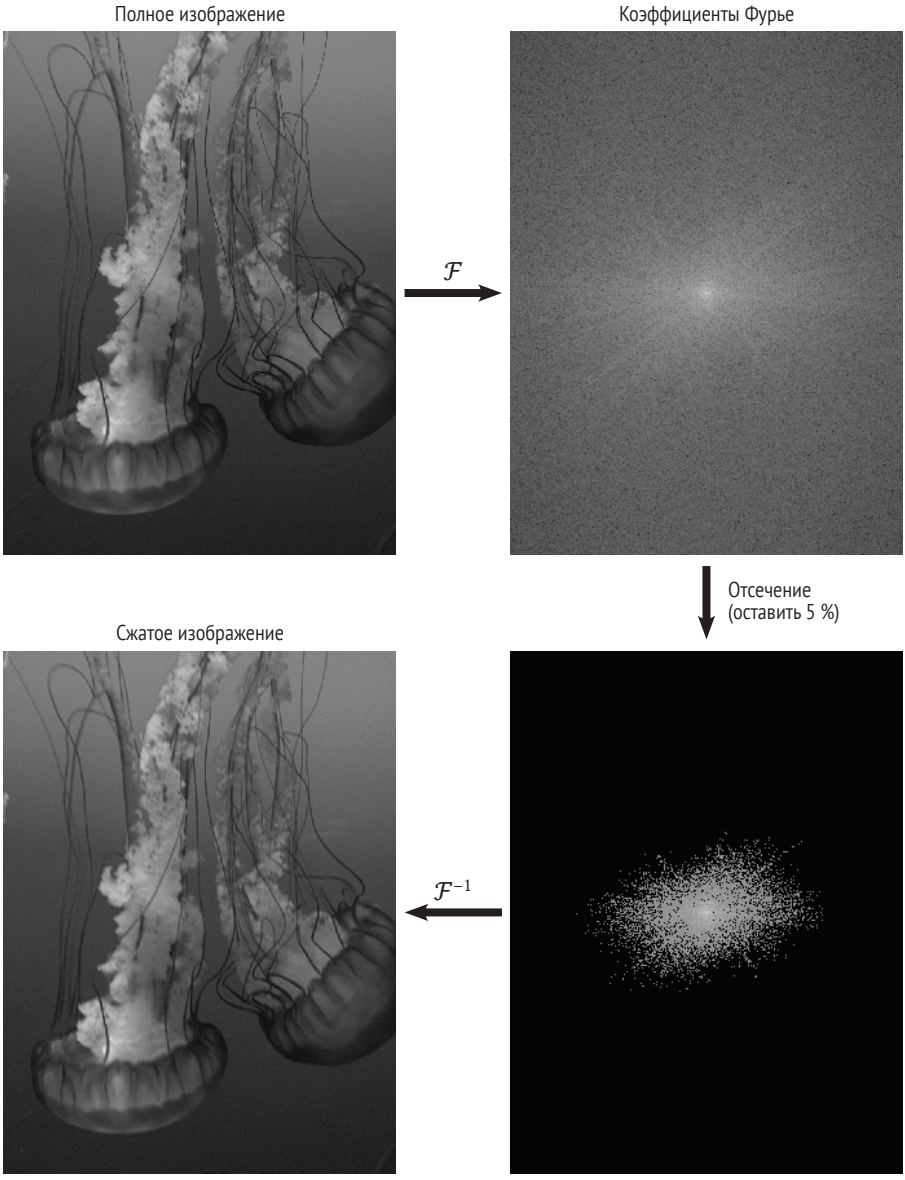

Рис. 3.1 ❖ Сжатие с помощью быстрого преобразования Фурье (БПФ) $\mathcal{F}$

Чтобы понять, какую роль играют разреженные коэффициенты Фурье в сжатом изображении, полезно рассмотреть изображение как поверхность, на которой высота точки определяется яркостью соответствующего пикселя. Это показано на рис. 3.2. Мы видим, что поверхность сравнительно простая и может быть представлена в виде суммы небольшого числа пространственных мод Фурье.

```
Anew = imresize(Abw,.2);
surf(double(Anew));
shading flat, view(-168,86)
```

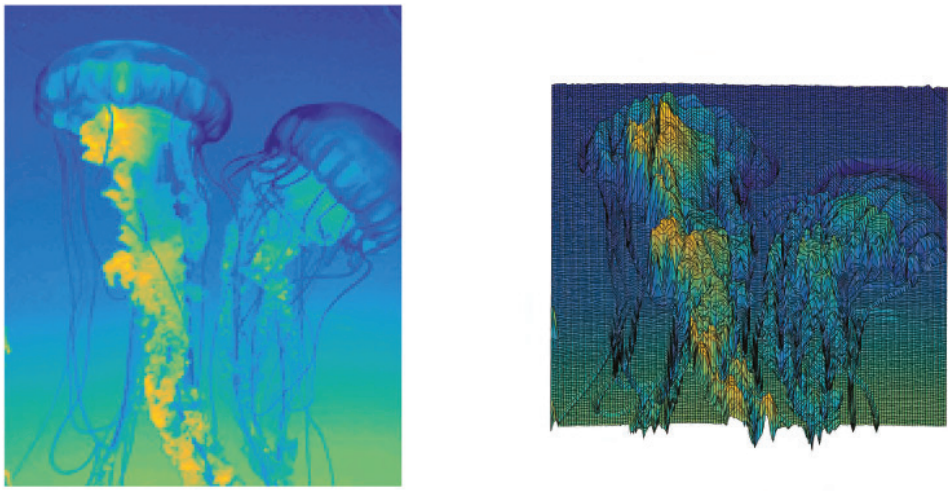

Рис. 3.2 ❖ Сжатое изображение (слева);
оно же, рассматриваемое как поверхность (справа)

Почему сигналы допускают сжатие: просторность пространства изображений

Важно отметить, что сжимаемость изображений тесно связана с огромной размерностью пространства изображений. Даже для простых черно-белых изображений размера 20×20 число возможных изображений равно 2^{400}, это больше, чем количество нуклонов в известной Вселенной. Для изображений высокого разрешения с большей глубиной цвета громадность пространства поражает еще сильнее.

В пространстве мегапиксельных изображений (порядка 1000×1000 пикселей) существуют изображения всех когда-либо живших людей, меня, набирающего это предложение, и вас в процессе его чтения. Но как бы обширно ни было множество этих *естественных* изображений, оно занимает лишь крохотную, едва заметную часть всего пространства изображений. Большинство элементов этого пространства – случайный шум, напоминающий статические помехи на экране телевизора. Для простоты будем рассматривать полутоновые изображения и представим, что значение каждого полутонового пикселя выбрано случайным образом. С очень высокой вероятностью получившееся изображение будет выглядеть как бессмысленный шум. Вы могли

бы рисовать такие случайные изображения всю свою жизнь и ни разу не наткнуться на изображение горы, человека или еще чего-нибудь узнаваемого[1].

Иными словами, естественные изображения встречаются на просторах пространства изображений чрезвычайно редко (см. рис. 3.3). Поскольку так много изображений носят случайный характер и лишены всякой структуры, большинство измерений, используемых для кодирования изображений, только для этих случайных наборов пикселей и нужны. Эти измерения избыточны, если нас интересует только кодирование естественных изображений. Отсюда вытекает важное следствие – интересные нам изображения (естественные) должны очень хорошо сжиматься, если удастся найти подходящий преобразованный базис, в котором легко идентифицировать избыточные измерения.

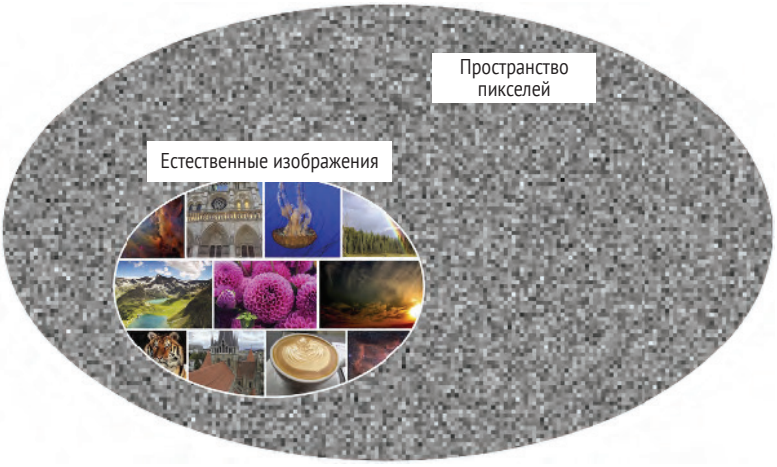

Рис. 3.3 ❖ Иллюстрации просторности пространства изображений (пикселей). Естественные изображения занимают исчезающе малую часть этого пространства

3.2. Сжатое измерение

Несмотря на значительные успехи, достигнутые сжатием в реальных приложениях, оно все еще нуждается в доступе к полноразмерным измерениям. Недавнее изобретение сжатого измерения [150, 112, 111, 113, 115, 109, 39, 114,

[1] Обширность пространства сигналов описана в рассказе Борхеса «Вавилонская библиотека» (1944 год), где речь идет о библиотеке, содержащей все книги, которые когда-либо могли быть написаны; среди них связные тексты занимают ничтожно малую долю [69]. В библиотеке Борхеса есть миллионы экземпляров этой книги, различающихся лишь в данном предложении. Есть еще одна известная вариация на эту тему – если обезьяна будет достаточно долго бить по клавишам печатной машинки, то рано или поздно напечатает собрание сочинений Шекспира. Одно из самых старых описаний этих комбинаторно больших пространств восходит еще к Аристотелю.

40] полностью перевернуло наши представления о сжатии. Стоит ли собирать данные высокой размерности, лишь для того чтобы потом их сжать? Ведь можно вместо этого произвести на удивление мало *сжатых*, или *случайных*, измерений, а затем на их основе вывести, как выглядит сжатое представление в преобразованном базисе. Идею сжатых измерений довольно просто выразить математически, но до недавнего времени нахождение наиболее разреженного вектора, согласующегося с измерениями, было задачей неполиномиальной сложности (NP-трудной). Стремительное внедрение сжатых измерений в инженерные и прикладные науки покоится на прочном математическом основании[1] – условиях, при которых с высокой вероятностью возможна реконструкция полного сигнала с помощью выпуклых алгоритмов.

Математически сжатое измерение использует разреженность сигнала в общем базисе, чтобы обеспечить полную реконструкцию сигнала по неожиданно малому набору измерений. Если сигнал $\mathbf{x}$ K-разреженный в базисе $\boldsymbol{\Psi}$, то вместо того чтобы измерять $\mathbf{x}$ непосредственно (n измерений), а затем сжимать, мы можем произвести гораздо меньше случайно выбранных, или *сжатых*, измерений, а затем найти ненулевые элементы $\mathbf{s}$ в преобразованной системе координат. Измерения $\mathbf{y} \in \mathbb{R}^p$, где $K < p \ll n$, описываются уравнением

$$\mathbf{y} = \mathbf{C}\mathbf{x}. \tag{3.2}$$

Матрица измерений[2] $\mathbf{C} \in \mathbb{R}^{p \times n}$ представляет множество p линейных измерений состояния $\mathbf{x}$. Выбор матрицы измерений $\mathbf{C}$ критически важен и обсуждается в разделе 3.4. Как правило, измерения – это случайные проекции состояния, и в таком случае элементами $\mathbf{C}$ являются случайные величины с распределением Гаусса или Бернулли. Можно также измерять отдельные элементы $\mathbf{x}$ (т. е. одиночные пиксели, если $\mathbf{x}$ – изображение), тогда $\mathbf{C}$ состоит из случайных строк единичной матрицы.

Зная разреженный вектор $\mathbf{s}$, можно реконструировать сигнал $\mathbf{x}$ по формуле (3.1). Таким образом, цель сжатого измерения – найти самый разреженный вектор $\mathbf{s}$, согласующийся с измерениями $\mathbf{y}$:

$$\mathbf{y} = \mathbf{C}\boldsymbol{\Psi}\mathbf{s} = \boldsymbol{\Theta}\mathbf{s}. \tag{3.3}$$

Система уравнений (3.3) недоопределенная, потому что у нее бесконечно много согласующихся решений $\mathbf{s}$. *Наиболее разреженный* вектор $\hat{\mathbf{s}}$ является решением следующей задачи оптимизации:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmin}} \|\mathbf{s}\|_0 \quad \text{при условии} \quad \mathbf{y} = \mathbf{C}\boldsymbol{\Psi}\mathbf{s}, \tag{3.4}$$

---

[1] Интересно, что невероятно плодотворное сотрудничество между Эммануэлем Кандесом (Emmanuel Candès) и Терренсом Тао (Terrance Tao) началось с обсуждения странных свойств реконструкции сигнала, когда они забирали детей из садика.

[2] В литературе по сжатым измерениям матрица измерений часто обозначается $\boldsymbol{\Phi}$; мы же выбрали букву $\mathbf{C}$, чтобы сохранить единство обозначений с уравнением выхода в теории управления. Кроме того, буква $\boldsymbol{\Phi}$ уже используется для обозначения DMD-мод в главе 7.

где $||\cdot||_0$ обозначает псевдонорму ℓ_0, равную числу ненулевых элементов; по-другому она называется мощностью **s**.

Задача оптимизации (3.4) невыпуклая, и в общем случае решение можно найти только перебором, сложность которого комбинаторно зависит от n и K. В частности, нужно проверить все возможные K-разреженные векторы в $\mathbb{R}^n$; если точная величина разреженности K неизвестна, то поиск еще шире. Поскольку поиск комбинаторный, задача (3.4) неразрешима даже для сравнительно небольших n и K, а перспектива нахождения решения более крупных задач не становится радужнее даже с учетом закона Мура об экспоненциальном возрастании вычислительной мощности.

По счастью, при определенных условиях на матрицу измерений **C** задачу оптимизации (3.4) можно свести к выпуклой минимизации по норме ℓ_1 [112, 150]:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmin}} ||\mathbf{s}||_1 \quad \text{при условии} \quad \mathbf{y} = \mathbf{C}\Psi\mathbf{s}, \tag{3.5}$$

где норма $||\cdot||_1$ определяется следующим образом:

$$||\mathbf{s}||_1 = \sum_{k=1}^{n} |s_k|. \tag{3.6}$$

Норма ℓ_1 называется еще манхэттенской нормой, потому что равна расстоянию между двумя точками, которое проезжает такси по линиям прямоугольной сетки. Выполнение сжатых измерений схематически показано на рис. 3.4. Решение с минимальной нормой ℓ_1 разрежено, а решение с минимальной нормой ℓ_2 – нет, как видно по рис. 3.5.

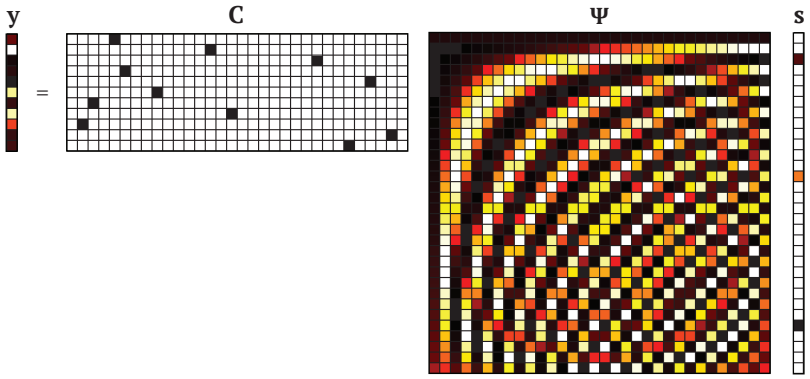

Рис. 3.4 ❖ Схема сжатых измерений

Существуют весьма специфические условия, при которых ℓ_1-минимизация в задаче (3.5) с высокой вероятностью сходится к самому разреженному решению задачи (3.4) [109, 111, 39]. Они будут подробно рассмотрены в разделе 3.4, а пока просто сформулируем их.

1. Матрица измерений **C** должна быть *некогерентной* относительно базиса **Ψ**, т. е. строки **C** не должны коррелировать со столбцами **Ψ**.

2. Количество измерений p должно быть достаточно велико, порядка

$$p \approx \mathcal{O}(K \log(n/K)) \approx k_1 K \log(n/K). \tag{3.7}$$

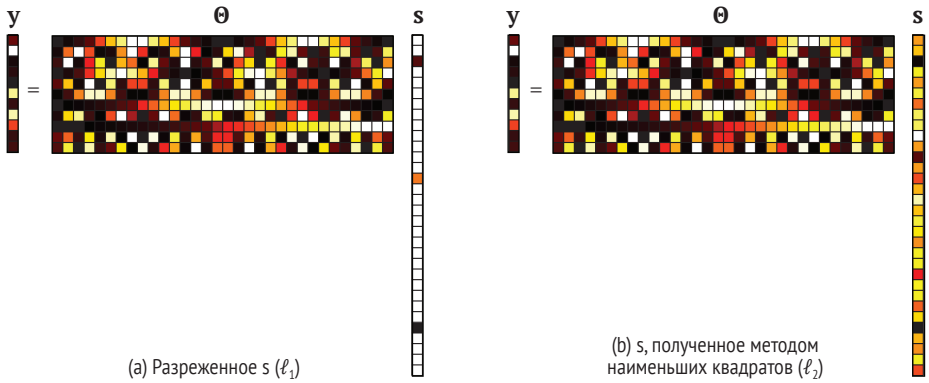

Рис. 3.5 ❖ Решения задачи о сжатом измерении с минимальными нормами ℓ_1 и ℓ_2. Различия в решениях этой задачи регрессии далее рассматриваются в главе 4

Постоянный множитель k_1 зависит от того, насколько некогерентны **C** и **Ψ**. Грубо говоря, эти два условия гарантируют, что матрица **CΨ** играет роль унитарного преобразования K-разреженных векторов **s**, сохраняющего относительные расстояния между векторами и обеспечивающего почти точную реконструкцию сигнала с минимизацией по норме ℓ_1. Это очень точно формулируется в терминах свойства ограниченной изометрии (restricted isometry property – RIP) в разделе 3.4.

Идея сжатого измерения может поначалу показаться противоречащей интуиции, особенно если принять во внимание классические результаты, касающиеся требований к выборке, при которых возможна точная реконструкция сигнала. Так, теорема Котельникова[1] [486, 409] утверждает, что для точного восстановления сигнала необходимо, чтобы частота дискретизации в два раза превышала наибольшую присутствующую в спектре частоту. Однако этот результат дает строгую границу требуемой частоты дискретизации только для сигналов с широкополосным спектром. Как правило, единственными по-настоящему широкополосными являются уже сжатые сигналы. Поскольку несжатый сигнал в общем случае является разреженным в базисе преобразования, теорему Котельникова можно ослабить, так что сигнал можно будет реконструировать по гораздо меньшему числу измерений, чем удвоенная максимальная частота. Но хотя число измерений можно уменьшить, техника сжатого измерения все равно требует точного *хронометража* измерений, как будет показано ниже. Кроме того, восстановление сигнала по результатам сжатых измерений, строго говоря, не гарантируется, утверждается лишь, что вероятность этого весьма высока, так что теория по сути своей статис-

[1] В англоязычной литературе ее называют теоремой отсчетов Шеннона–Найквиста. – *Прим. перев.*

тическая. Впрочем, для задач умеренного размера вероятность успешного восстановления необычайно велика.

Заявление об отказе от ответственности

Грубая схема сжатого измерения показана на рис. 3.6. Но эта схема – всего лишь инсценировка, не основанная на реальном вычислении сжатых измерений, поскольку применение сжатых измерений для реконструкции изображения практически нереализуемо из-за гигантского объема вычислений. Важно отметить, что в большинстве приложений, связанных с обработкой изображений, сжатые измерения практически неосуществимы. Однако изображения все же часто используются, чтобы мотивировать и объяснить суть сжатых измерений, в силу простоты манипуляций и нашего интуитивного восприятия картинок. Мы должны признать свою вину в том, что ведем читателя в неверном направлении.

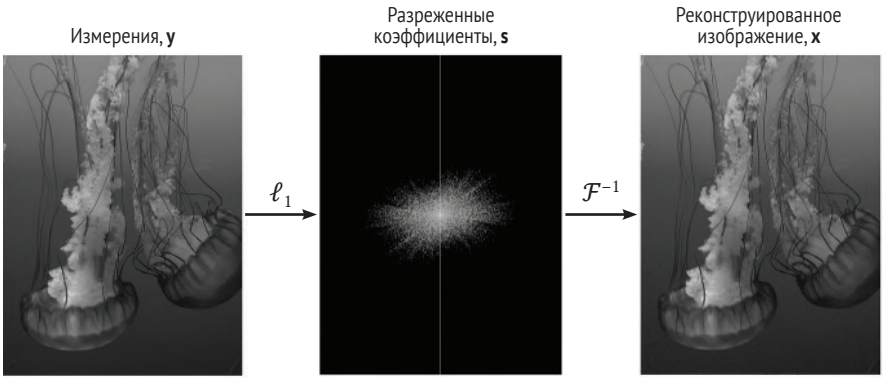

Рис. 3.6 ❖ Схематическая иллюстрация сжатого измерения с минимизацией по норме ℓ_1. Отметим, что это инсценировка, которая не основана на настоящем вычислении сжатых измерений. В действительности для сжатого измерения типичного изображения нужно очень много измерений, и вычисление практически нереализуемо

При ближайшем рассмотрении примера выясняется, что мы анализируем изображение, состоящее из 1024×768 пикселей, и что для сжатия, обеспечивающего точную реконструкцию, требуется приблизительно 5 % коэффициентов Фурье. Таким образом, уровень разреженности $K = 0.05 \times 1024 \times 768 \approx 40\,000$. Следовательно, согласно грубой оценке (3.7) с постоянным множителем $k_1 = 3$, получается, что нужно $p \approx 350\,000$ измерений, или примерно 45 % исходного числа пикселей. Но даже если бы мы имели доступ к этим 45 % случайных измерений, вычислить правильный разреженный вектор коэффициентов Фурье практически невозможно, во всяком случае несоизмеримо труднее, чем эффективное сжатие изображения с помощью БПФ, описанное в разделе 3.1.

Сжатое измерение изображений обычно используется только в особых случаях, когда удается добиться существенного уменьшения числа измерений. Например, одним из первых применений технологии сжатых измерений стала МРТ (магниторезонансная томография) младенцев, когда сокращение времени, в течение которого ребенок должен быть неподвижен, помогало снизить потребность в опасном глубоком наркозе.

Однако легко видеть, что число измерений p уменьшается с увеличением уровня разреженности K, т. е. чем *более* разрежен сигнал, тем меньше требуется измерений. Так что математическое достижение, заключающееся в сведении комбинаторно сложных ℓ_0-задач к выпуклым ℓ_1-задачам, все же ценно и может использоваться далеко не только для сжатого измерения изображений.

Другие формулировки

Помимо минимизации по норме ℓ_1 в задаче (3.5), есть и другие подходы, основанные на *жадных алгоритмах* [525, 526, 528, 527, 530, 243, 529, 207, 531, 205, 398, 206], которые находят разреженное решение (3.3) путем итеративного согласованного преследования. Например, алгоритм согласованного преследования со сжатой выборкой (compressed sensing matching pursuit – CoSaMP) [398] вычислительно эффективен, прост в реализации и свободно доступен.

Если в измерениях $\mathbf{y}$ присутствует аддитивный шум, скажем белый шум с амплитудой ε, то существуют более робастные варианты (3.5):

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmin}} ||\mathbf{s}||_1 \quad \text{при условии} \quad ||\mathbf{C\Psi s} - \mathbf{y}||_2 < \mu. \tag{3.8}$$

Родственная задача выпуклой оптимизации имеет вид:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmin}} ||\mathbf{C\Psi s} - \mathbf{y}||_2 + \lambda ||\mathbf{s}||_1, \tag{3.9}$$

где $\lambda \geq 0$ – весовой параметр, определяющий важность разреженности. Уравнения (3.8) и (3.9) тесно связаны [528].

3.3. Примеры сжатых измерений

В этом разделе мы рассмотрим конкретные примеры сжатых измерений для восстановления разреженного сигнала. Первый пример показывает, что норма ℓ_1 поощряет разреженность при решении общей недоопределенной системы уравнений, а во втором рассматривается восстановление разреженного двухтонового звукового сигнала с помощью сжатых измерений.

Норма ℓ_1 и разреженные решения недоопределенной системы

Чтобы увидеть положительный эффект нормы ℓ_1, рассмотрим общую недоопределенную систему уравнений. Мы построим систему уравнений $\mathbf{y} = \mathbf{\Theta s}$, в которой матрица $\mathbf{\Theta}$ состоит из $p = 200$ строк (измерений) и $n = 1000$ столбцов (неизвестных). В общем случае существует бесконечно много решений $\mathbf{s}$, удовлетворяющих этим уравнениям, если только мы не окажемся настолько невезучими, что строки линейно зависимы, а измерения с ними несовместны. На самом деле это отличный пример вероятностного мышления, которое в технологии сжатых измерений формулируется более общо: случайная система линейных уравнений, в которой количество неизвестных значительно превышает количество уравнений, *с высокой вероятностью* имеет бесконечно много решений.

В MATLAB не составляет проблемы решить эту недоопределенную систему с условием минимизации как нормы ℓ_1, так и нормы ℓ_2. Решение с минимальной нормой ℓ_2 получается псевдообращением матрицы (связанным с методом SVD, описанным в главах 1 и 4). А решение с минимальной нормой ℓ_1 получается применением пакета оптимизации **cvx** (ConVeX). На рис. 3.7 показано, что решение с минимальной нормой ℓ_1 фактически разреженное (большинство элементов близки к нулю), а решение с минимальной нормой ℓ_2 *плотное*, т. е. какая-то часть энергии приходится на каждый элемент вектора.

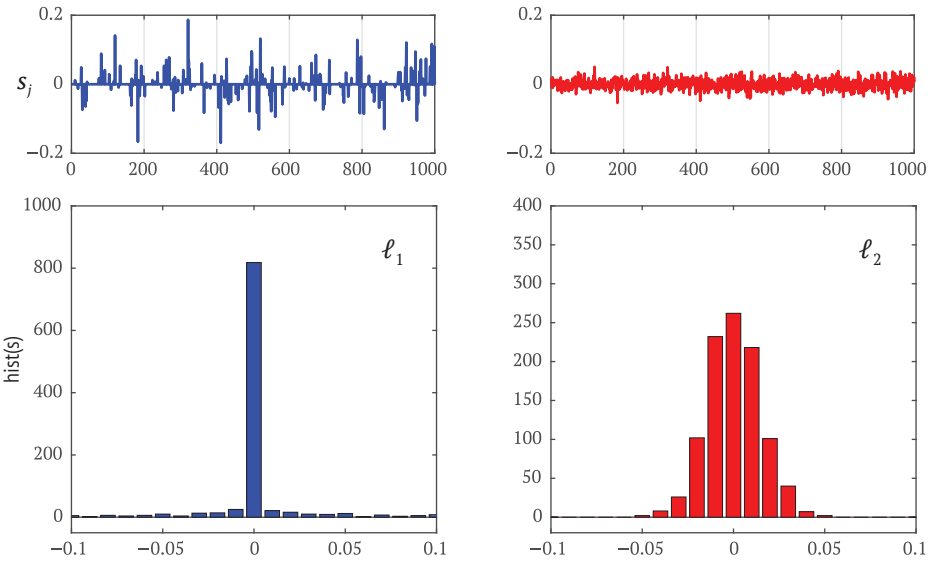

Рис. 3.7 ❖ Сравнение решений недоопределенной линейной системы с минимальной нормой ℓ_1 (синее, слева) и с минимальной нормой ℓ_2 (красное, справа)

Листинг 3.1 ❖ Решения недоопределенной линейной системы **y** = **Θs**

```
% Решить систему y = Theta * s относительно "s"
n = 1000; % размерность s
p = 200;  % количество измерений, dim(y)
Theta = randn(p,n);
y = randn(p,1);

% Решение с минимальной нормой L1,s_L1
cvx_begin;
variable s_L1(n);
minimize( norm(s_L1,1) );
subject to
Theta*s_L1 == y;
cvx_end;

s_L2 = pinv(Theta)*y; % Решение с минимальной нормой L2,s_L2
```

Восстановление звукового сигнала по разреженным измерениям

Чтобы проиллюстрировать использование сжатых измерений для восстановления многомерного сигнала по разреженному набору случайных измерений, рассмотрим двухтоновый звуковой сигнал:

$$x(t) = \cos(2\pi \times 97t) + \cos(2\pi \times 777t). \tag{3.10}$$

Очевидно, что этот сигнал разрежен в частотной области, поскольку является суммой ровно двух косинусоидальных волн. Наибольшая частота равна 777 Гц, поэтому, по теореме Котельникова, частота дискретизации должна быть равна 1554 Гц. Однако, воспользовавшись разреженностью сигнала в частотной области, мы можем точно реконструировать сигнал по случайным отборам, производимым со средней частотой 128 Гц, что намного ниже, чем диктует теорема Котельникова. На рис. 3.8 показан результат сжатого измерения, реализованного программой в листинге 3.2. В этом примере полный сигнал генерируется в промежутке времени от $t = 0$ до $t = 1$ с разрешением $n = 4096$, а затем производится случайная выборка в $p = 128$ моментов времени. Для нахождения разреженного вектора коэффициентов в базисе дискретного косинусного преобразования (discrete cosine transform – DCT) применяется метод согласованного преследования.

Листинг 3.2 ❖ Реконструкция двухтонового звукового сигнала по сжатым измерениям

```
%% Генерация сигнала, DCT сигнала
n = 4096; % число точек для генерации сигнала с высоким разрешением
t = linspace(0, 1, n);
x = cos(2* 97 * pi * t) + cos(2* 777 * pi * t);
xt = fft(x); % сигнал после преобразования Фурье
PSD = xt.*conj(xt)/n; % спектральная плотность мощности
```

```
%% Случайная выборка из сигнала
p = 128; % количество случайных отборов, p = n/32
perm = round(rand(p, 1) * n);
y = x(perm); % сжатое измерение

%% Решение задачи о сжатом измерении
Psi = dct(eye(n, n)); % построить матрицу Psi
Theta = Psi(perm, :); % измерить строки Psi
s = cosamp(Theta,y',10,1.e-10,10); % СИ согласованным преследованием
xrecon = idct(s);     % реконструировать полный сигнал
```

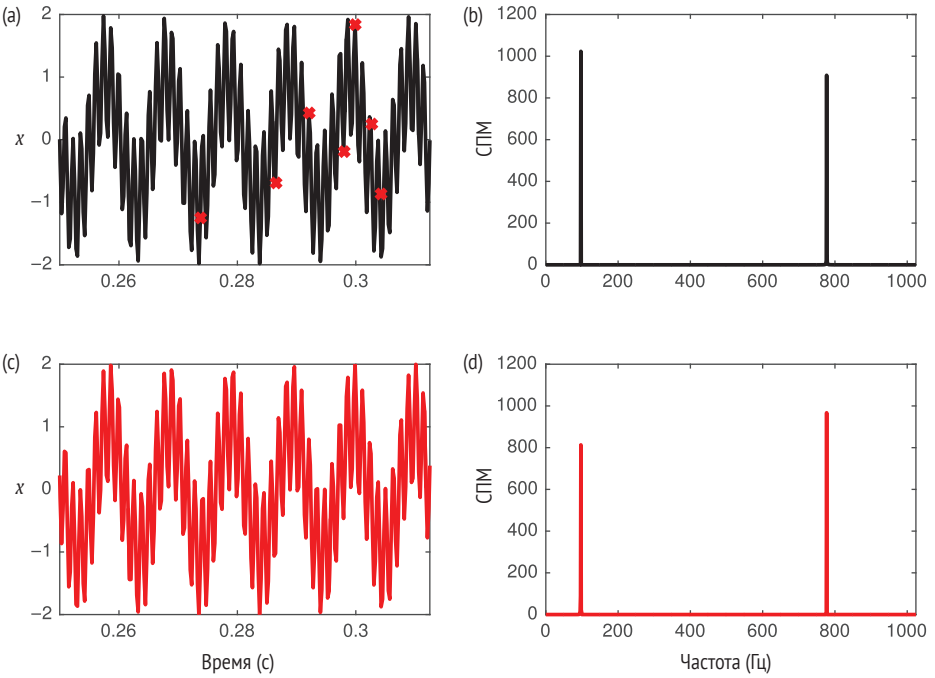

Рис. 3.8 ❖ Реконструкция двухтонового звукового сигнала $x(t) = \cos(2\pi \times 97t) + \cos(2\pi \times 777t)$ по сжатым измерениям. Полный сигнал и спектральная мощность сигнала показаны на рисунках (a) и (b) соответственно. Сигнал измерен в случайные разреженные моменты времени, обозначенные красными точками на рисунке (a), а затем эти измерения использованы для построения оценки, показанной на рисунках (c) и (d). Временные ряды на рисунках (a) и (c) – масштабированное изображение полного временного диапазона от $t = 0$ до $t = 1$

Важно отметить, что $p = 128$ измерений случайно выбраны из сигнала с разрешением 4096 точек. Таким образом, нам точно известен набор разреженных измерений с разрешением, значительно превышающим нашу частоту отборов. Если бы $p = 128$ измерений были равномерно распределены по времени, то алгоритм сжатых измерений не сработал бы. Точнее, если бы мы вычисляли СПМ непосредственно по таким равномерным измерениям, то появились бы гармоники высокочастотного сигнала, дающие ложные частотные пики.

Наконец, мы можем заменить алгоритм согласованного преследования

```
s = cosamp(Theta,y',10,1.e-10,10); % СИ согласованным преследованием
```

минимизацией по норме ℓ_1, воспользовавшись пакетом **CVX** [218]:

```
%% минимизация L1 с помощью CVX
cvx_begin;
    variable s(n);
    minimize( norm(s,1) );
    subject to
        Theta*s == y';
cvx_end;
```

Решая задачу о сжатых измерениях методом согласованного преследования (CoSaMP), мы должны задать желаемый уровень разреженности K, но эта величина заранее не всегда известна. Для минимизации по норме ℓ_1 априорное знание уровня разреженности не требуется, хотя для сходимости к самому разреженному решению необходимо достаточное число измерений p, а эта величина косвенно зависит от K.

3.4. Геометрия сжатия

Суть разреженного сжатия можно изложить довольно просто: заданный сигнал, если он достаточно разрежен в известном базисе, можно реконструировать (с высокой вероятностью), используя значительно меньше измерений, чем длительность сигнала, при условии что произведено довольно много измерений и эти измерения в достаточной степени случайны. Каждую часть этого утверждения можно математически строго сформулировать в рамках стройной концепции, описывающей геометрию разреженных векторов и их преобразование в результате случайных измерений. Конкретно, наличие достаточного количества хороших измерений означает существование матрицы

$$\mathbf{\Theta} = \mathbf{C\Psi}, \tag{3.11}$$

которая сохраняет расстояния и скалярные произведения разреженных векторов $\mathbf{s}$. Иными словами, мы ищем матрицу измерений $\mathbf{C}$ такую, что отображение $\mathbf{\Theta}$ является почти изометрией на множестве разреженных векторов. Слово «изометрия» означает *одинаковое расстояние*, оно тесно связано с унитарностью, т. е. сохранением не только расстояний, но и углов между векторами. Если $\mathbf{\Theta}$ является почти изометрией, то можно из следующего уравнения найти самый разреженный вектор $\mathbf{s}$, применив выпуклую ℓ_1-минимизацию:

$$\mathbf{y} = \mathbf{\Theta s}. \tag{3.12}$$

Далее в этом разделе мы опишем условия на матрицу измерений $\mathbf{C}$, при которых $\mathbf{\Theta}$ с высокой вероятностью является почти изометрическим отображением. Геометрические свойства различных норм показаны на рис. 3.9.

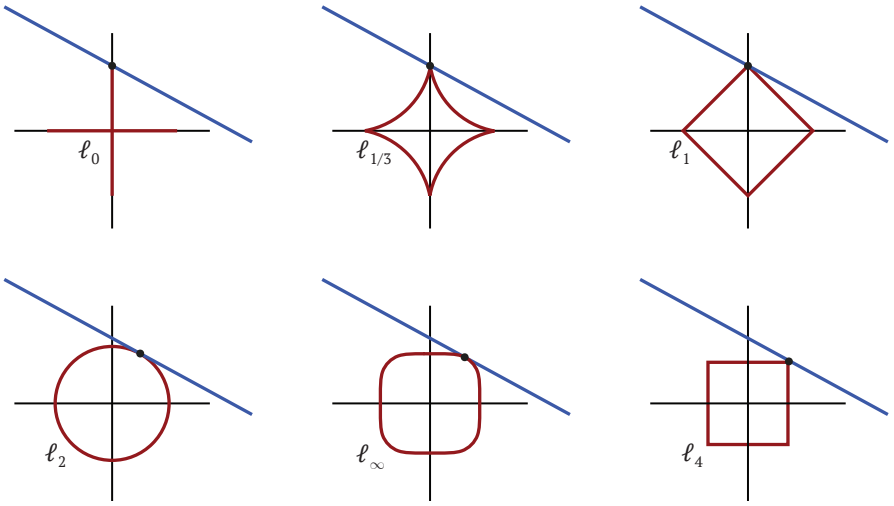

Рис. 3.9 ❖ Точка с минимальной нормой на прямой для разных норм ℓ_p. Синяя прямая представляет множество решений недоопределенной системы уравнений, а красные кривые – линии уровня минимальных норм, пересекающие эту синюю прямую. Для норм от ℓ_0 до ℓ_1 решение с минимальной нормой соответствует самому разреженному решению, и активна только одна координата. Для норм, начиная с ℓ_2, решение с минимальной нормой не является разреженным, но активны все координаты

Определить, сколько нужно измерений, сравнительно просто. Если сигнал является K-разреженным в базисе $\mathbf{\Psi}$, т. е. имеется не более K ненулевых коэффициентов, то по порядку величины число измерений $p \sim \mathcal{O}(K\log(n/K)) = k_1 K\log(n/K)$, как в (3.7). Постоянный множитель k_1, определяющий, сколько *точно* измерений необходимо произвести, зависит от качества измерений. Грубо говоря, измерения хороши, если они *некогерентны* относительно столбцов разреживающего базиса. Это означает, что скалярные произведения строк $\mathbf{C}$ со столбцами $\mathbf{\Psi}$ малы. Если измерения когерентны со столбцами разреживающего базиса, то измерение дает мало информации, если только данная мода базиса не окажется ненулевой в $\mathbf{s}$. Напротив, на некогерентное измерение оказывает влияние почти любая активная мода, что позволяет определить активные моды. Дельта-функции некогерентны относительно мод Фурье, поскольку возбуждают ответ в широкой полосе частот. Чем более *некогерентны* измерения, тем меньшее их количество p необходимо.

Некогерентность измерений $\mathbf{C}$ и базиса $\mathbf{\Psi}$ описывается функцией $\mu(\mathbf{C}, \mathbf{\Psi})$:

$$\mu(\mathbf{C},\mathbf{\Psi}) = \sqrt{n}\max_{j,k}\left|\langle \mathbf{c}_k, \boldsymbol{\psi}_j\rangle\right|, \tag{3.13}$$

где $\mathbf{c}_k$ – k-я строка матрицы $\mathbf{C}$, а $\boldsymbol{\psi}_j$ – j-й столбец матрицы $\mathbf{\Psi}$. Значения μ, соответствующие когерентности, лежат в диапазоне от 1 до $\sqrt{n}$.

Свойство ограниченной изометрии (RIP)

Если измерения некогерентны, то матрица $\mathbf{C\Psi}$ обладает *свойством ограниченной изометрии* (RIP) для разреженных векторов $\mathbf{s}$

$$(1-\delta_K)||\mathbf{s}||_2^2 \leq ||\mathbf{C\Psi s}||_2^2 \leq (1+\delta_K)||\mathbf{s}||_2^2,$$

где δ_K – постоянная ограниченной изометрии [114]. Постоянная δ_K определяется как наименьшее число, удовлетворяющее неравенству выше для *всех* K-разреженных векторов $\mathbf{s}$. Если δ_K мало, то $\mathbf{C\Psi}$ действует как почти изометрия на множестве K-разреженных векторов $\mathbf{s}$. На практике трудно вычислить δ_K непосредственно; более того, матрица измерений $\mathbf{C}$ может выбираться случайно, поэтому лучше бы делать статистические выводы о границах δ_K для семейства матриц измерений $\mathbf{C}$, а не вычислять δ_K для конкретной $\mathbf{C}$. Вообще говоря, при увеличении количества измерений постоянная δ_K уменьшается, что улучшает способность $\mathbf{C\Psi}$ играть роль изометрии на множестве разреженных векторов. При наличии достаточного числа некогерентных измерений (см. выше) можно точно определить K ненулевых элементов вектора $\mathbf{s}$ длины n. В таком случае существует граница постоянной δ_K, гарантирующая точность реконструкции сигнала по незашумленным данным. Углубленное обсуждение некогерентности и свойства RIP имеется в работах [39, 114].

Некогерентность и матрицы измерений

Еще один значительный результат теории сжатых измерений – существование общих выборочных матриц $\mathbf{C}$, которые в достаточной степени некогерентны относительно почти всех базисов преобразований. Точнее, матрицы случайных измерений с распределениями Бернулли и Гаусса с высокой вероятностью удовлетворяют свойству RIP для базиса $\mathbf{\Psi}$ общего вида [113]. Имеются также дополнительные результаты, связанные с обобщением RIP и исследованием некогерентности разреженных матриц [205].

Во многих инженерных приложениях выгодно представить сигнал $\mathbf{x}$ в общем базисе, например Фурье или вейвлетном. Ключевое преимущество заключается в том, что одноточечные измерения некогерентны относительно этих базисов, т. е. вызывают отклик в широкой полосе частот. Отбор сигнала в случайных точках представляет особый интерес в приложениях, где индивидуальные измерения обходятся дорого, например при мониторинге океана. На рис. 3.10 показаны примеры матриц случайных измерений: однопиксельная, гауссова, Бернулли и случайная разреженная.

Особенно полезный базис преобразования для сжатых измерений получается в результате сингулярного разложения[1], дающего специальный базис, в котором данные оптимально разрежены [316, 80, 81, 31, 98]. Базис

[1] SVD дает оптимальную низкоранговую аппроксимацию матрицы, которая используется в методе главных компонент (PCA) и собственном ортогональном разложении (POD).

усеченного SVD может дать более эффективное восстановление сигнала по меньшему количеству измерений. Удалось достигнуть некоторого прогресса в разработке сжатых SVD и PCA на основе леммы Джонсона–Линденштраусса [267, 187, 436, 206]. Эта лемма тесно связана с RIP и говорит, когда можно погрузить векторы высокой размерности в пространство низкой размерности с сохранением спектральных свойств.

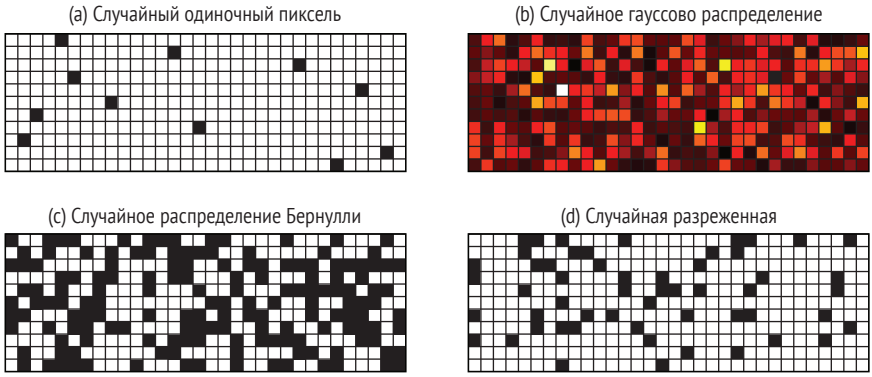

Рис. 3.10 ❖ Примеры матриц хороших случайных измерений **C**

Плохие измерения

До сих пор мы описывали, как производить *хорошие* сжатые измерения. На рис. 3.11 показан пример особенно неудачного выбора измерений **C**, соответствующий последним p столбцам разреживающего базиса $\mathbf{\Psi}$. В этом случае произведение $\mathbf{\Theta} = \mathbf{C\Psi}$ является единичной матрицей размера $p \times p$, дополненной нулями слева. При таком выборе **C** любой сигнал **s**, не активный в последних p столбцах $\mathbf{\Psi}$, попадает в ядро $\mathbf{\Theta}$ и оказывается вообще невидим для измерений **y**. В результате эти измерения приводят к значительной потере информации для многих разреженных векторов.

3.5. Разреженная регрессия

Норма ℓ_1 стала применяться для поощрения разреженности гораздо раньше, чем появилась технология сжатых измерений. На самом деле многие достоинства нормы ℓ_1 были хорошо известны и часто использовались в статистике в течение многих десятилетий. В этом разделе мы покажем, что норму ℓ_1 можно использовать для *регуляризации* статистической регрессии – как с целью штрафовать статистические выбросы, так и с целью поощрять *экономные* статистические модели с как можно меньшим числом факторов. Сравнение норм ℓ_2 и ℓ_1 в регрессии обсуждается далее в главе 4.

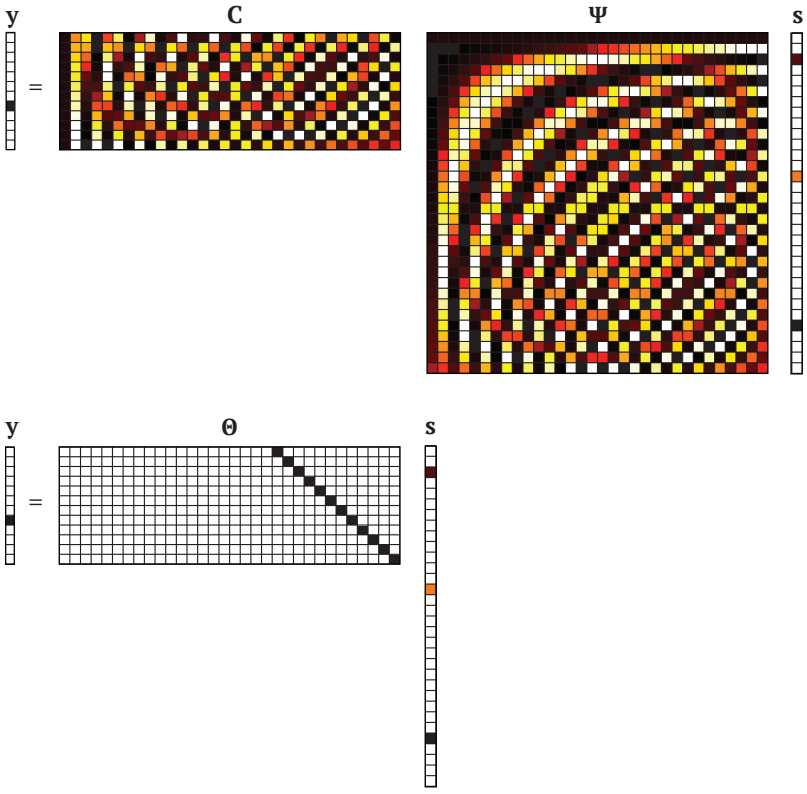

Рис. 3.11 ❖ Примеры плохих матриц измерений **C**

Отбрасывание выбросов и робастность

Регрессия методом наименьших квадратов является, наверное, самой употребительной статистической моделью, применяемой для аппроксимации данных. Однако хорошо известно, что аппроксимация может быть сколь угодно сильно искажена присутствием в данных всего лишь одного сильного выброса; в случае регрессии методом наименьших квадратов выбросам назначаются большие веса, поскольку их расстояние до аппроксимирующей прямой возводится в квадрат. Схематически это показано на рис. 3.12.

С другой стороны, если оптимизация производится по норме ℓ_1, то всем экспериментальным точкам назначаются одинаковые веса, поэтому решение получается более робастным по отношению к выбросам и поврежденным данным. Эта процедура называется также регрессией наименьших абсолютных отклонений (least absolute deviations – LAD). В листинге 2.3 приведена программа, демонстрирующая применения регрессии наименьших квадратов (ℓ_2) и LAD (ℓ_1) для набора данных с выбросом.

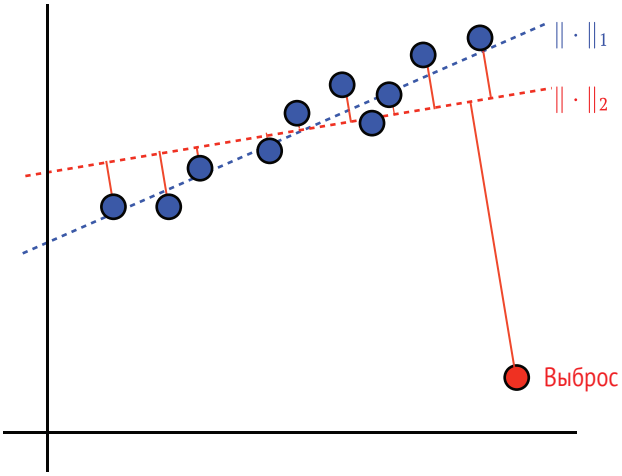

Рис. 3.12 ❖ Регрессия методом наименьших квадратов чувствительна к выбросам (красная прямая), а регрессия наименьших абсолютных отклонений робастна (синяя прямая)

Листинг 3.3 ❖ Использование нормы ℓ_1 для робастной статистической регрессии

```
x = sort(4*(rand(25,1)-.5)); % случайные данные из отрезка [-2,2]
b = .9*x + .1*randn(size(x)); % прямая y = 0.9x без шума
atrue = x\b; % угловой коэффициент по методу наим. квадратов (без выбросов)

b(end) = -5.5; % добавить выброс
acorrupt = x\b; % новый угловой коэффициент

cvx_begin; % L1-оптимизация для исключения выброса
    variable aL1; % aL1 - подлежащий оптимизации угловой коэффициент
    minimize( norm(aL1*x-b,1) ); % aL1 робастный
cvx_end;
```

Отбор признаков и LASSO-регрессия

Возможность интерпретации – важное свойство статистических моделей, поскольку их потребителями часто являются люди без технической подготовки, в т. ч. руководители предприятий и политики. Обычно модель регрессии интерпретируется тем лучше, чем меньше в ней параметров, влияющих на результат. И это еще одна причина желать разреженности.

Оператор наименьшего абсолютного сжатия и выборки (least absolute shrinkage and selection operator – LASSO) – это метод регрессии со штрафом по метрике ℓ_1, представляющий собой компромисс между сложностью модели и описательной способностью [518]. Это стремление к *экономичности* модели также является проявлением принципа бритвы Оккама, который гласит, что из всех возможных описаний простейшая правильная модель обычно является верной. С момента открытия в работе Tibshirani в 1996 году [518] метод LASSO стал краеугольным камнем статистического моделирования,

сейчас у него много вариантов и родственных методов [236, 558, 264]. LASSO тесно связан с более ранним методом неотрицательной гарроты (nonnegative garrote) Бреймена [76] и с еще более ранним алгоритмом сравнения с мягким порогом Donoho and Johnstone [153, 154]. LASSO можно рассматривать как регрессию, поощряющую разреженность, которая обращает на пользу устойчивость гребневой регрессии, регуляризированной по норме ℓ_2 [249], известной также как регуляризация по Тихонову. В качестве метода регрессии часто используется также эластичная сеть, в которой комбинируются штрафы по нормам ℓ_1 и ℓ_2 из LASSO и гребневой регрессии [573]. Разреженную регрессию мы будем подробно изучать в главе 4.

Если заданы наблюдаемые значения параметров прогностической модели (предикторов) и выходы системы, организованные в виде строк матрицы **A** и вектора **b** соответственно, то цель регрессии – найти связь между столбцами **A**, которая лучше всего согласуется с результатами **b**. Математически это можно записать в виде уравнения

$$\mathbf{Ax} = \mathbf{b}. \tag{3.14}$$

Регрессия методом наименьших квадратов обычно находит вектор **x**, все элементы которого ненулевые, т. е. для предсказания **b** необходимо использовать все *столбцы* **A**. Однако нам часто хотелось бы, чтобы статистическая модель была *проще*, а это значит, что вектор **x** должен быть разреженным. Метод LASSO добавляет штраф по норме ℓ_1 для *регуляризации* задачи регрессии методом наименьших квадратов с целью предотвратить переобучение модели:

$$\mathbf{x} = \underset{\mathbf{x}'}{\operatorname{argmin}} ||\mathbf{Ax}' - \mathbf{b}||_2 + \lambda ||\mathbf{x}||_1. \tag{3.15}$$

Обычно параметр λ выбирается из некоторого диапазона значений, а качество аппроксимации *контролируется* на тестовом наборе зарезервированных данных. Если данных недостаточно для формирования представительного обучающего и тестового наборов, то зачастую модель несколько раз обучают и тестируют на случайной выборке данных (обычно 80 % выборки отводится под обучающий набор, а 20 % под тестирующий) – это называется *перекрестной проверкой*. Процедура перекрестной проверки позволяет найти экономную модель, в которой сравнительно немного параметров, и избежать переобучения.

Многие статистические системы являются переопределенными, поскольку наблюдений больше, чем возможных предикторов. Поэтому воспользоваться стандартным методом сжатых измерений не получится, т. к. из-за шума измерений гарантированно не будет существовать точного разреженного решения, которое минимизировало бы $||\mathbf{Ax} - \mathbf{b}||_2$. Однако LASSO-регрессия хорошо работает в переопределенных задачах, вследствие чего принята в качестве общего метода регрессии. Заметим, что ранняя версия геометрической иллюстрации на рис. 3.9, которая объясняет поощряющую разреженность природу нормы ℓ_1, была приведена в статье Тибширани 1996 года [518].

LASSO-регрессия часто применяется для построения статистических моделей заболеваний, например рака и сердечной недостаточности, поскольку существует много разных предикторов, в т. ч. демографическая информация, стиль жизни, биометрические и генетические данные. Таким образом, LASSO – это умный вариант подхода *кухонной мойки*, когда чуть ли не вся потенциально применимая для прогнозирования информация сваливается в одну кучу, а затем просеивается в поисках действительно релевантных предикторов.

В качестве простого примера рассмотрим искусственный набор данных, состоящий из 100 наблюдений и результата, представленного вектором $\mathbf{b} \in \mathbb{R}^{100}$. Каждый результат в **b** описывается комбинацией ровно 2 из 10 потенциальных предикторов, наблюдения которых представлены строками матрицы $\mathbf{A} \in \mathbb{R}^{100\times10}$:

```
A = randn(100,10); % матрица возможных предикторов
x = [0; 0; 1; 0; 0; 0; -1; 0; 0; 0]; % 2 ненулевых предиктора
b = A*x + 2*randn(100,1); % наблюдения (с шумом)
```

Вектор **x** разреженный по построению, поскольку в нем только два ненулевых элемента, а к наблюдениям в **b** мы прибавили шум. Регрессия методом наименьших квадратов дает:

```
>>xL2 = pinv(A)*b
xL2 = -0.0232
      -0.3395
       0.9591
      -0.1777
       0.2912
      -0.0525
      -1.2720
      -0.0411
       0.0413
      -0.0500
```

Обратите внимание, что все коэффициенты ненулевые.

Реализация алгоритма LASSO с 10-групповой перекрестной проверкой в MATLAB выполняется одной командой:

```
[XL1 FitInfo] = lasso(A,b,'CV',10);
```

Команда **lasso** пробегает по диапазону значений λ, и получающиеся векторы **x** сохраняются в виде столбцов матрицы **XL1**. Чтобы выбрать самую экономную модель, описывающую данные и при этом не страдающую от переобучения, мы можем нарисовать ошибку перекрестной проверки в виде функции от λ, как показано на рис. 3.13:

```
lassoPlot(XL1,FitInfo,'PlotType','CV')
```

Зеленая точка соответствует значению λ, которое доставляет минимум среднеквадратической ошибке перекрестной проверки, а синяя точка отстоит от минимума ошибки перекрестной проверки на одно стандартное отклонение. Результирующая модель находится в элементе **FitInfo.Index1SE**:

```
>> xL1 = XL1(:,FitInfo.Index1SE)

  xL1 =    0
           0
      0.7037
           0
           0
           0
     -0.4929
           0
           0
           0
```

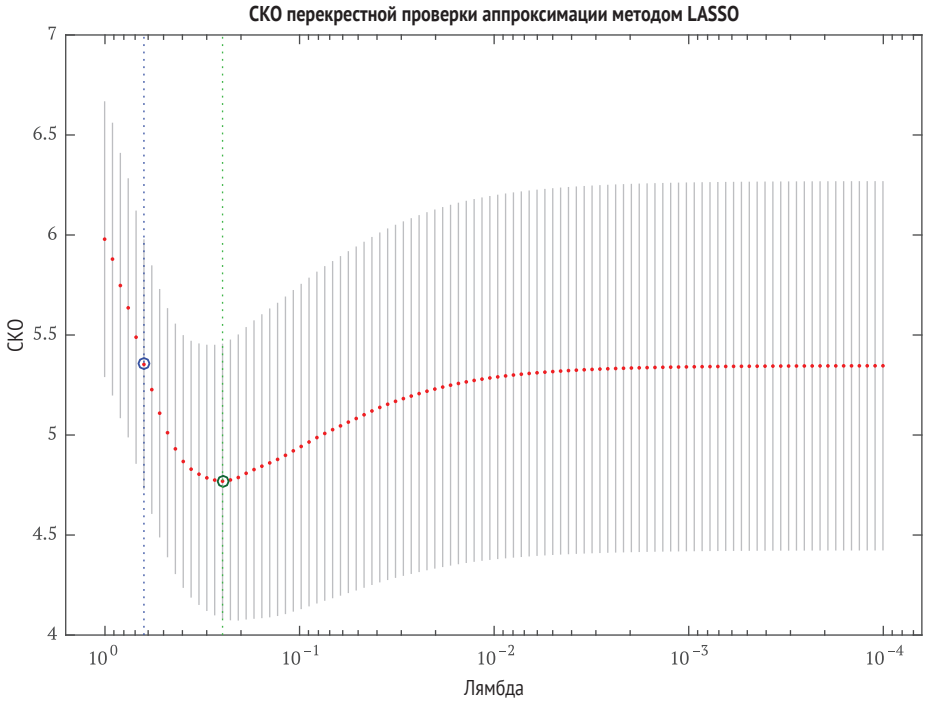

Рис. 3.13 ❖ Результат команды **lassoPlot**, визуализирующей зависимость среднеквадратической ошибки (СКО) перекрестной проверки от λ

Отметим, что результирующая модель разреженная и активны в ней правильные члены. Однако значения регрессии для этих членов не точны, поэтому может оказаться необходимым *исключить систематическую ошибку* LASSO, дополнительно применив регрессию методом наименьших квадратов к выявленным ненулевым коэффициентам:

```
>>xL1DeBiased = pinv(A(:,abs(xL1)>0))*b
  xL1DeBiased =  1.0980
                -1.0671
```

3.6. Разреженное представление

В нашем обсуждении разреженности неявно предполагалось, что если сигнал высокой размерности в действительности обладает низкоразмерной структурой, то он допускает *разреженное представление* в подходящем базисе, или словаре. Помимо базиса SVD или Фурье, сигнал может быть разреженным в сверхполном словаре, столбцы которого состоят из самих обучающих данных. По сути дела, тестовый сигнал может быть не только разрежен в библиотеке общих признаков $\mathbf{U}$, участвующей в сингулярном разложении $\mathbf{X} = \mathbf{U\Sigma V}^*$, но иметь и разреженное представление в словаре $\mathbf{X}$.

В работе Wright et al. [560] продемонстрирована эффективность разреженного представления в словаре тестовых сигналов для робастной классификации человеческих лиц в присутствии значительного шума и преград. Так называемое разреженное представление для классификации (sparse representation for classification – SRC) уже давно и широко используется в обработке изображений, а сравнительно недавно еще и для классификации динамических режимов в нелинейных дифференциальных уравнениях [98, 433, 191, 308].

На рис. 3.14 показана базовая схема SRC, где библиотека изображений лиц применяется для построения сверхполной библиотеки $\mathbf{\Theta}$. В этом примере для каждого из 20 людей в базе данных Yale B используется 30 изображений, так что число столбцов в матрице $\mathbf{\Theta}$ равно 600. Чтобы воспользоваться сжатыми измерениями, т. е. ℓ_1-минимизацией, $\mathbf{\Theta}$ должна быть недоопределенной, поэтому мы понижаем разрешение с 192×168 до 12×10, так что сериализованное изображение представляется вектором длины 120. Алгоритм понижающей передискретизации влияет на точность классификации. Новое тестовое изображение $\mathbf{y}$, соответствующее классу c, также передискретизируется, чтобы соответствовать столбцам $\mathbf{\Theta}$, а затем с помощью алгоритма сжатых измерений строится его разреженное представление в виде суммы столбцов $\mathbf{\Theta}$. Результирующий вектор коэффициентов $\mathbf{s}$ должен быть разреженным, и в идеале коэффициенты должны быть велики преимущественно в областях библиотеки, соответствующих правильному человеку, принадлежащему классу c. На последнем этапе классификации вычисляется ошибка ℓ_2-реконструкции по коэффициентам в векторе $\mathbf{s}$ отдельно для каждой категории. Та категория, для которой ошибка ℓ_2-реконструкции минимальна, признается правильной для тестового изображения.

Листинг 3.4 ❖ Загрузка лиц из Йельской базы данных и построение обучающего и тестового наборов

```
load ../../CH01_SVD/DATA/allFaces.mat
X = faces;
%% Построить обучающий и тестовый наборы
nTrain = 30; nTest = 20; nPeople = 20;
Train = zeros(size(X,1),nTrain*nPeople);
Test = zeros(size(X,1),nTest*nPeople);
for k=1:nPeople
    baseind = 0;
```

```
    if(k>1) baseind = sum(nfaces(1:k-1));
    end
    inds = baseind + (1:nfaces(k));
    Train(:,(k-1)*nTrain+1:k*nTrain)=X(:,inds(1:nTrain));
    Test(:,(k-1)*nTest+1:k*nTest)=X(:,inds(nTrain+1:nTrain+nTest));
end
```

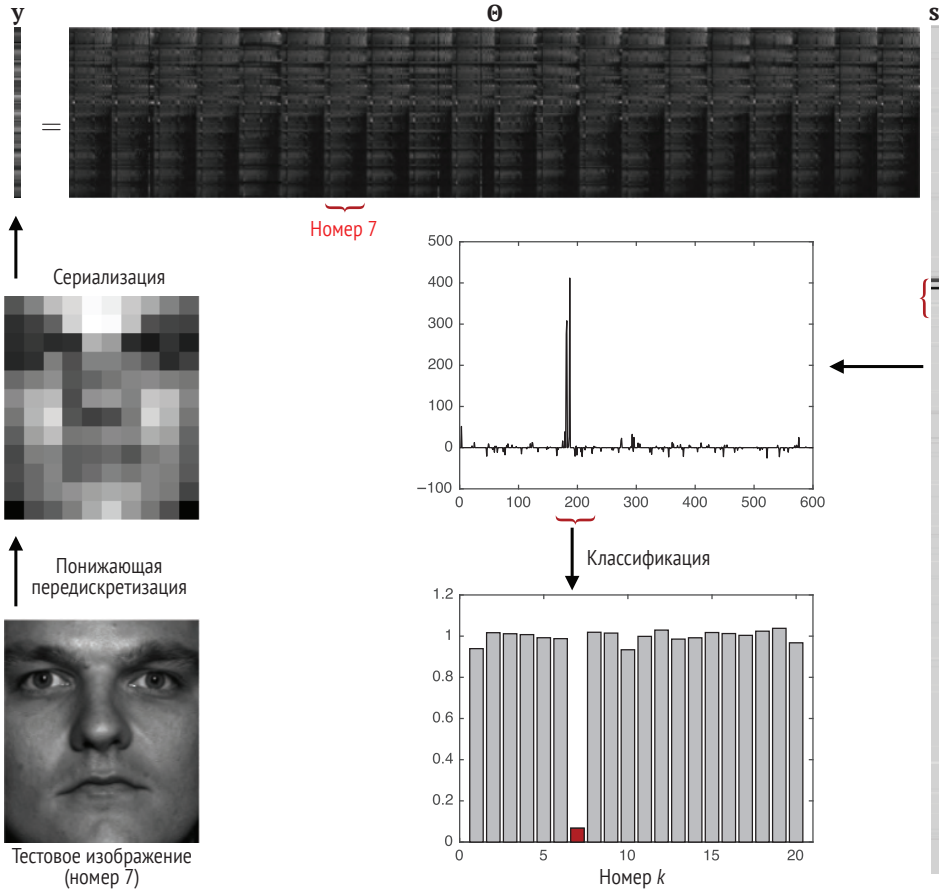

Рис. 3.14 ❖ Схематическое описание разреженного представления для классификации

Листинг 3.5 ❖ Понижающая передискретизация обучающих изображений для построения библиотеки **Θ**

```
M = size(Train,2);
Theta = zeros(120,M);
for k=1:M
    temp = reshape(Train(:,k),n,m);
    tempSmall = imresize(temp,[12 10],'lanczos3');
    Theta(:,k) = reshape(tempSmall,120,1);
end
```

```
for k=1:M % нормировать столбцы Theta
    Theta(:,k) = Theta(:,k)/norm(Theta(:,k));
end
```

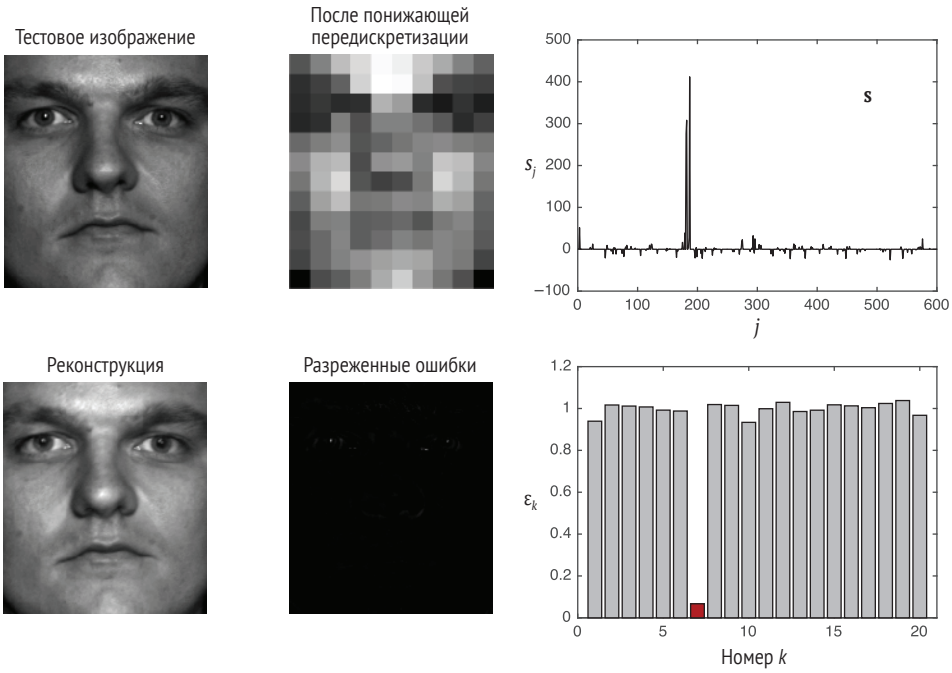

Рис. 3.15 ❖ Разреженное представление для классификации на примере библиотеки лиц. Чистое тестовое изображение правильно отождествлено с человеком номер 7 из библиотеки

Листинг 3.6 ❖ Построение тестовых изображений и их понижающая передискретизация для получения **y**

```
x1 = Test(:,126); % чистое изображение
mustache = double(rgb2gray(imread('mustache.jpg')))/255);
x2 = Test(:,126).*reshape(mustache,n*m,1); % усы
randvec = randperm(n*m);
first30 = randvec(1:floor(.3*length(randvec)));
vals30 = uint8(255*rand(size(first30)));
x3 = x1;
x3(first30) = vals30; % 30 % выпавших пикселей
x4 = x1 + 50*randn(size(x1)); % случайный шум

%% Понижающая передискретизация тестовых изображений
X = [x1 x2 x3 x4];
Y = zeros(120,4);
for k=1:4
    temp = reshape(X(:,k),n,m);
    tempSmall = imresize(temp,[12 10],'lanczos3');
    Y(:,k) = reshape(tempSmall,120,1);
end

%% L1-поиск, очистка
```

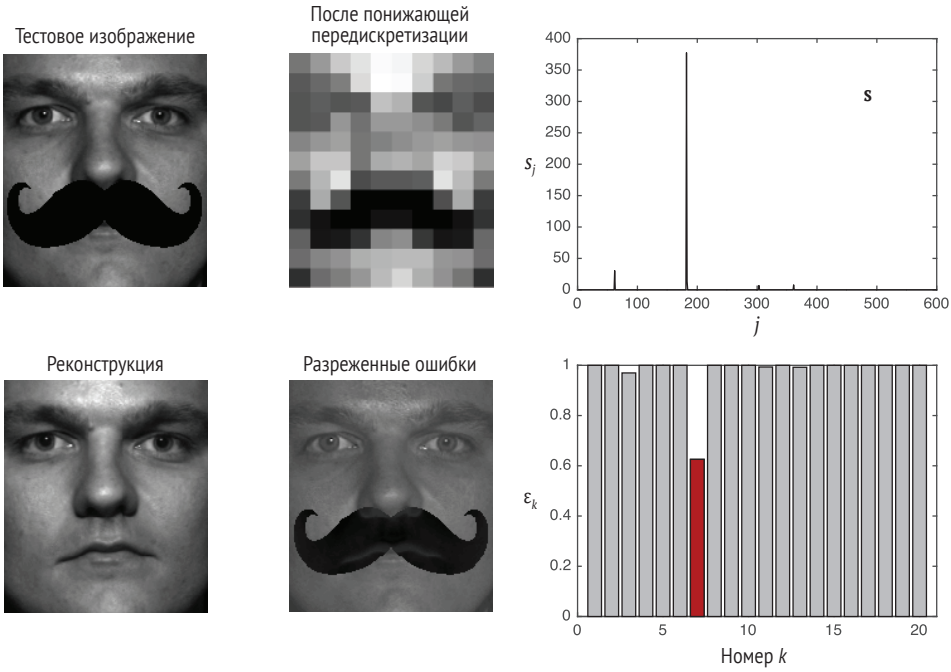

Рис. 3.16 ❖ Разреженное представление для классификации на примере лица номер 7, загороженного накладными усами

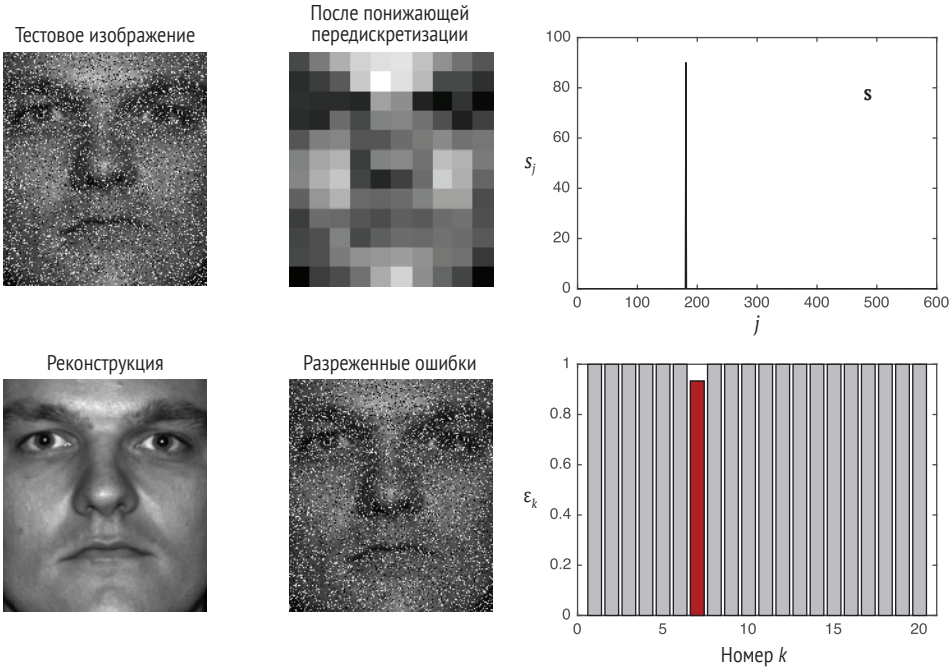

Рис. 3.17 ❖ Разреженное представление для классификации на примере лица с 30 % выпавших пикселей (случайных и равномерно распределенных)

Листинг 3.7 ❖ Поиск разреженного представления тестового изображения. Один и тот же код используется для всех тестовых изображений от $\mathbf{y}_1$ до $\mathbf{y}_4$

```
y1 = Y(:,1);
eps = .01;
cvx_begin;
    variable s1(M); % разреженный вектор коэффициентов
    minimize( norm(s1,1) );
    subject to
        norm(Theta*s1 - y1,2) < eps;
cvx_end;

plot(s1)
imagesc(reshape(Train*(s1./normTheta'),n,m))
imagesc(reshape(x1-(Train*(s1./normTheta')),n,m))

binErr = zeros(nPeople,1);
for k=1:nPeople
    L = (k-1)*nTrain+1:k*nTrain;
    binErr(k)=norm(x1-(Train(:,L)*(s1(L)./normTheta(L)')))/norm(x1)
end
bar(binErr)
```

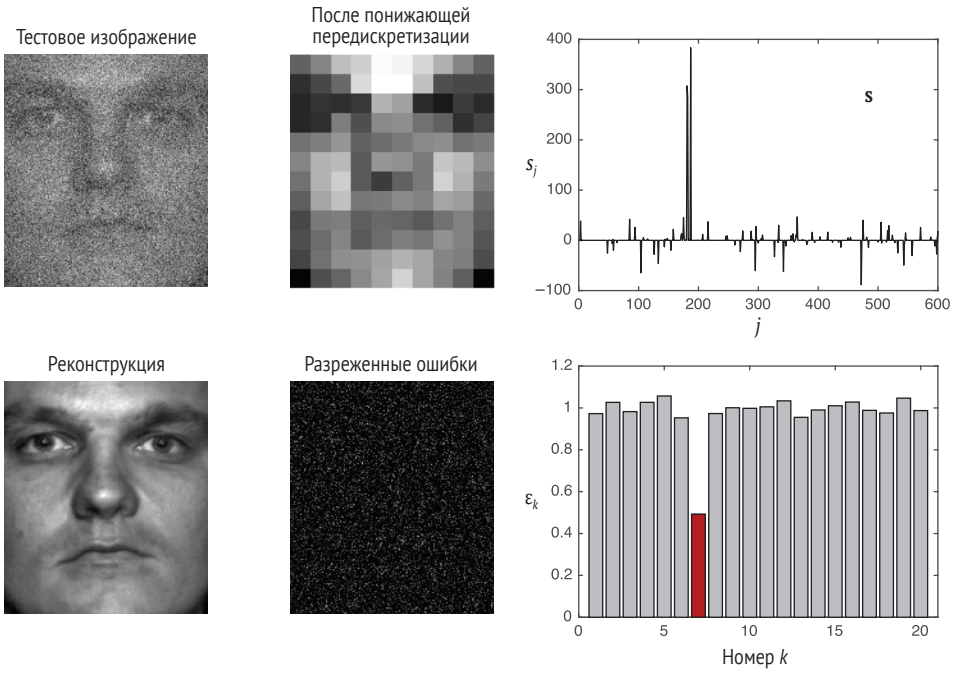

Рис. 3.18 ❖ Разреженное представление для классификации на примере лица с добавленным шумом

3.7. Робастный метод главных компонент (RPCA)

Как уже отмечалось в разделе 3.5, модели регрессии методом наименьших квадратов очень чувствительны к выбросам и загрязненным данным. Методу главных компонент (PCA) эта слабость также присуща, т. е. он является *хрупким* по отношению к выбросам. Чтобы ослабить эту чувствительность, в работе Candès et al. [110] был разработан робастный метод главных компонент (RPCA), цель которого – разложить матрицу данных $\mathbf{X}$ в структурную низкоранговую матрицу $\mathbf{L}$ и разреженную матрицу $\mathbf{S}$, содержащую выбросы и загрязненные данные:

$$\mathbf{X} = \mathbf{L} + \mathbf{S}. \tag{3.16}$$

Главные компоненты $\mathbf{L}$ являются *робастными* относительно выбросов и загрязненных данных в $\mathbf{S}$. Это разложение имеет важные следствия для многих современных задач, включая видеонаблюдение (когда объекты на заднем плане сосредоточены в $\mathbf{L}$, а объекты на переднем плане – в $\mathbf{S}$), распознавание лиц (собственные лица – в $\mathbf{L}$, а тени, преграды и т. д. – в $\mathbf{S}$), обработку естественного языка и латентное семантическое индексирование, а также задачи ранжирования[1].

Математически цель состоит в том, чтобы найти такие матрицы $\mathbf{L}$ и $\mathbf{S}$, что:

$$\min_{\mathbf{L},\mathbf{S}} \operatorname{rank}(\mathbf{L}) + ||\mathbf{S}||_0 \quad \text{при условии} \quad \mathbf{L} + \mathbf{S} = \mathbf{X}. \tag{3.17}$$

Однако члены $\operatorname{rank}(\mathbf{L})$ и $||\mathbf{S}||_0$ не являются выпуклыми, поэтому задача оптимизации практически неразрешима. Но, как и в случае сжатых измерений, оптимальные $\mathbf{L}$ и $\mathbf{S}$ можно найти *с высокой вероятностью*, воспользовавшись выпуклым ослаблением задачи (3.17):

$$\min_{\mathbf{L},\mathbf{S}} \operatorname{rank}||\mathbf{L}||_* + \lambda||\mathbf{S}||_1 \quad \text{при условии} \quad \mathbf{L} + \mathbf{S} = \mathbf{X}. \tag{3.18}$$

Здесь $||\cdot||_*$ обозначает ядерную норму, равную сумме сингулярных значений, выступающую в роли замены ранга. Примечательно, что решение задачи (3.18) сходится к решению (3.17) с высокой вероятностью, если $\lambda = 1/\sqrt{\max(n,m)}$, где n и m – размеры $\mathbf{X}$, при условии что $\mathbf{L}$ и $\mathbf{S}$ удовлетворяют следующим условиям:

1. $\mathbf{L}$ не является разреженной.

---

[1] Задачу ранжирования можно рассмотреть на примере приза, присужденного компанией Netflix за решение задачи о пополнении матрицы. В этом конкурсе строится большая матрица предпочтений, строки которой соответствуют пользователям, а столбцы – фильмам. Матрица разреженная, поскольку большинство пользователей оценивают лишь небольшую долю фильмов. Задача заключается в том, чтобы правильно заполнить отсутствующие элементы матрицы и тем самым узнать, какую оценку, скорее всего, поставил бы пользователь фильму, который не смотрел.

2. **S** не является матрицей низкого ранга; мы предполагаем, что элементы случайно распределены, поэтому размерность пространства столбцов не мала.

Выпуклая задача (3.18) называется *преследованием главных компонент* (principal component pursuit – PCP), ее можно решить расширенным методом множителей Лагранжа (augmented Lagrange multiplier – ALM). Точнее, строится расширенный лагранжиан:

$$\mathcal{L}(\mathbf{L},\mathbf{S},\mathbf{Y}) = ||\mathbf{L}||_* + \lambda||\mathbf{S}||_1 + \langle \mathbf{Y}, \mathbf{X} - \mathbf{L} - \mathbf{S}\rangle + \frac{\mu}{2}||\mathbf{X} - \mathbf{L} - \mathbf{S}||_F^2. \tag{3.19}$$

Общий алгоритм такой: найти $\mathbf{L}_k$ и $\mathbf{S}_k$, которые доставляют минимум $\mathcal{L}$, обновить множители Лагранжа $\mathbf{Y}_{k+1} = \mathbf{Y}_k + \mu(\mathbf{X} - \mathbf{L}_k - \mathbf{S}_k)$ и продолжать итерации до сходимости. Однако в этой конкретной системе метод переменных направлений (alternating directions method – ADM) [337, 566] дает простую процедуру нахождения **L** и **S**.

Сначала строится оператор сжатия $\mathcal{S}_\tau(x) = \text{sign}(x)\max(|x| - \tau, 0)$ (в MATLAB функция **shrink**, показанная ниже):

```
function out = shrink(X,tau)
    out = sign(X).*max(abs(X)-tau,0);
end
```

Затем строится оператор порога сингулярных значений $\text{SVT}_\tau(\mathbf{X}) = \mathbf{U}\mathcal{S}_\tau(\mathbf{\Sigma})\mathbf{V}^*$ (в MATLAB функция **SVT**, показанная ниже):

```
function out = SVT(X,tau)
    [U,S,V] = svd(X,'econ');
    out = U*shrink(S,tau)*V';
end
```

И наконец, мы можем итеративно использовать операторы $\mathcal{S}_\tau$ и SVT для нахождения **L** и **S**:

Листинг 3.8 ❖ Алгоритм RPCA с применением метода переменных направлений (ADM)

```
function [L,S] = RPCA(X)
[n1,n2] = size(X);
mu = n1*n2/(4*sum(abs(X(:))));
lambda = 1/sqrt(max(n1,n2));
thresh = 1e-7*norm(X,'fro');

L = zeros(size(X));
S = zeros(size(X));
Y = zeros(size(X));
count = 0;
while((norm(X-L-S,'fro')>thresh)&&(count<1000))
    L = SVT(X-S+(1/mu)*Y,1/mu);
    S = shrink(X-L+(1/mu)*Y,lambda/mu);
    Y = Y + mu*(X-L-S);
    count = count + 1
end
```

Продемонстрируем использование этой функции на примере с собственными лицами:

```
load allFaces.mat
X = faces(:,1:nfaces(1));
[L,S] = RPCA(X);
```

На рис. 3.19 показаны исходные столбцы **X**, а также низкоранговая и разреженная компоненты. Заметим, что в этом примере RPCA успешно заполняет затемненные области, соответствующие теням. В низкоранговой компоненте **L** тени удалены и заполнены наиболее подходящими низкоранговыми признаками из собственных лиц. Эту технику можно использовать и для устранения других преград, например фальшивых усов, солнечных очков или шума.

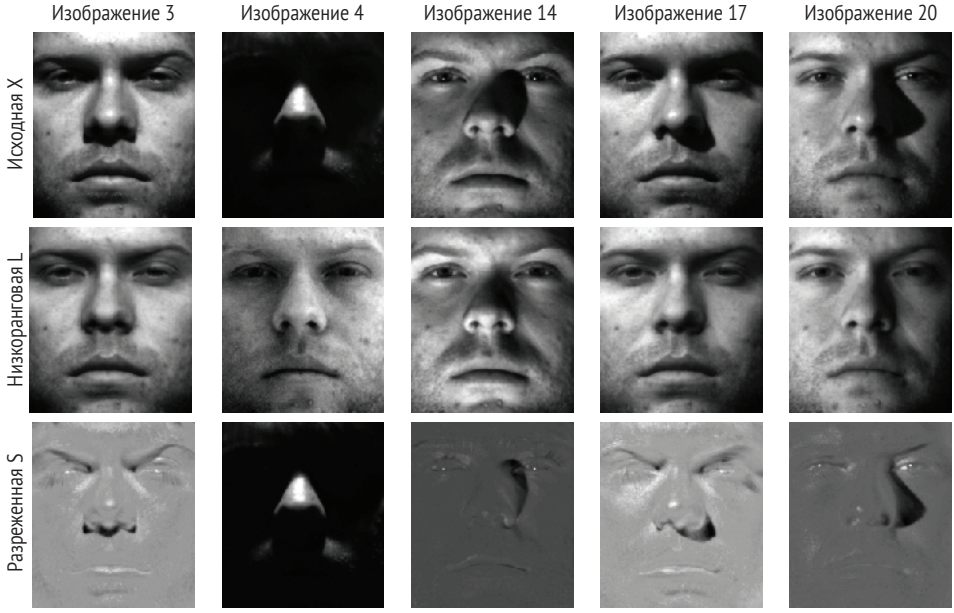

Рис. 3.19 ❖ Результат работы RPCA для изображений из базы данных Yale B

3.8. Разреженное размещение датчиков

До сих пор мы изучали реконструкцию сигнала по случайным измерениям в общем базисе, например Фурье или вейвлетном. Это обеспечивает значительную гибкость, поскольку априори не предполагается никакой структуры, за исключением того, что сигнал является разреженным в известном базисе. Например, метод сжатых измерений одинаково хорошо восстанавливает изображение горы, лица или чашки кофе. Но если мы заранее знаем, что будем реконструировать изображение человеческого лица, то можем сильно уменьшить число датчиков, необходимых для реконструкции или классифи-

кации, оптимизировав размещение датчиков для конкретной библиотеки признаков $\mathbf{\Psi}_r = \tilde{\mathbf{U}}$, построенной с помощью сингулярного разложения.

Таким образом, можно спроектировать *специальные* датчики для конкретной библиотеки – в отличие от ранее описанного подхода, когда датчики выбирались случайно из общей библиотеки. Близкое к оптимальному размещение датчиков можно найти с помощью жадных процедур, которые хорошо масштабируются на сигналы высокой размерности, например QR-разложения матрицы. Обсуждение ниже следует статьям Manohar et al. [366] и B. Brunton et al. [89], читателю рекомендуется прочитать их для восполнения деталей. Похожие подходы будут использоваться для эффективной выборки из моделей пониженного порядка в главе 12, где они называются *гиперредукцией*. Существуют также обобщения, ориентированные на размещение датчиков и приводов в системах управления [365], основанные на балансирующих преобразованиях (см. главу 9).

Оптимизация размещения датчиков важна почти во всех задачах, решаемых позже: классификации, предсказания, оценивания, моделирования и управления. Однако нахождение оптимальных мест включает полный перебор комбинаторно большого множества, его мощность равна числу способов размещения p датчиков в n возможных местах. Благодаря недавно предложенным жадным и разреженным методам этот поиск стал практически осуществимым и масштабируется на большие задачи. Уменьшение числа датчиков посредством теоретически обоснованного размещения может оказаться критически важным, если датчики стоят дорого, а также позволяет ускорить оценку состояния в системах управления с низкой задержкой и широкой полосой пропускания.

Разреженное размещение датчиков для реконструкции

Цель оптимизированного размещения датчиков для специальной библиотеки $\mathbf{\Psi}_r \in \mathbb{R}^{n \times r}$ – найти такую разреженную матрицу измерений $\mathbf{C} \in \mathbb{R}^{p \times n}$, чтобы обращение линейной системы уравнений

$$\mathbf{y} = \mathbf{C}\mathbf{\Psi}_r\mathbf{a} = \boldsymbol{\theta}\mathbf{a} \tag{3.20}$$

было настолько хорошо обусловлено, насколько возможно. Иными словами, мы ищем $\mathbf{C}$, минимизирующую число обусловленности $\mathbf{C}\mathbf{\Psi}_r = \boldsymbol{\theta}$, так чтобы ее можно было обратить и найти низкоранговые коэффициенты $\mathbf{a}$ при заданных зашумленных измерениях $\mathbf{y}$. Числом обусловленности матрицы $\boldsymbol{\theta}$ называется отношение ее максимального сингулярного значения к минимальному, оно показывает, насколько умножение на матрицу или ее обращение чувствительно к ошибкам входных данных. Чем больше число обусловленности, тем хуже качество обращения зашумленного сигнала. Число обусловленности – это мера ошибки в худшем случае, когда сигнал $\mathbf{a}$ направлен так же, как сингулярный вектор, соответствующий минимальному сингулярному значению $\boldsymbol{\theta}$, а добавленный шум направлен так же, как максимальный сингулярный вектор:

$$\boldsymbol{\theta}(\mathbf{a} + \varepsilon_{\mathbf{a}}) = \sigma_{\min}\mathbf{a} + \sigma_{\max}\varepsilon_{\mathbf{a}}. \tag{3.21}$$

Таким образом, отношение сигнала к шуму уменьшается кратно числу обусловленности после применения отображения $\boldsymbol{\theta}$. Поэтому мы хотим минимизировать число обусловленности путем разумного выбора $\mathbf{C}$. Схематически это показано на рис. 3.20 для $p = r$.

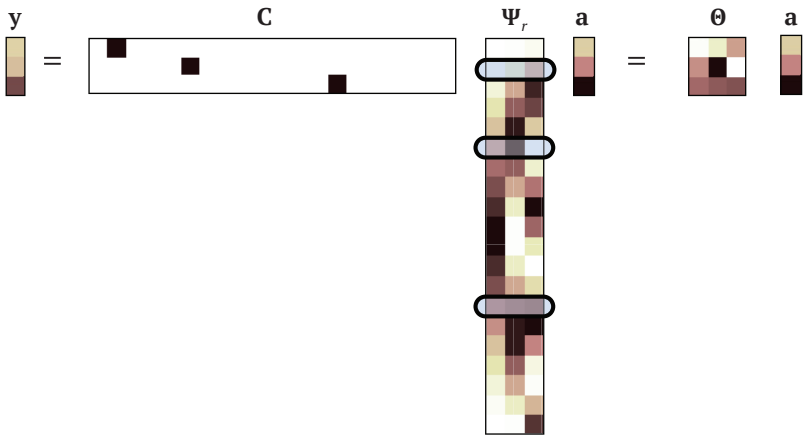

Рис. 3.20 ❖ Метод наименьших квадратов с r разреженными датчиками дает единственное решение $\mathbf{a}$, а значит, и $\mathbf{x}$.
Взято из работы Manohar et al. [366] с разрешения авторов

Если число датчиков равно рангу библиотеки, т. е. $p = r$, то $\boldsymbol{\theta}$ – квадратная матрица, и мы выбираем $\mathbf{C}$ так, чтобы эта матрица была настолько хорошо обусловленной для обращения, насколько возможно. Если $p > r$, то мы стараемся улучшить условие $\mathbf{M} = \boldsymbol{\theta}^T\boldsymbol{\theta}$, участвующее в вычислении псевдообратной матрицы. Можно выписать критерий оптимизации для минимизации сингулярного значения, следа или определителя $\boldsymbol{\theta}$ (соответственно $\mathbf{M}$). Однако все эти задачи NP-трудные и требуют комбинаторного перебора всех возможных конфигураций датчиков. Существуют итеративные методы решения этой задачи, например выпуклая оптимизация и полуопределенное программирование [74, 269], но они дорогие, т. к. включают итеративное разложение матриц размера $n \times n$. Вместо них обычно применяются жадные алгоритмы, позволяющие приближенно оптимизировать размещение датчиков. Эти методы *неполного собственного ортогонального разложения* (gappy POD) [179] первоначально основывались на случайной субдискретизации. Однако удалось добиться значительного прогресса, продемонстрировав использование разумных стратегий дискретизации для моделей пониженного порядка (ROM) в гидродинамике [555] и моделировании океана [565]. Предложенные недавно варианты *метода эмпирической интерполяции* (EIM, DEIM и Q-DEIM) [41, 127, 159] позволяют достичь близкой к оптимальной дискретизации для интерполяционной реконструкции нелинейных членов ROM.

Случайные датчики. В общем случае для оценки вектора коэффициентов при модах **a** можно использовать случайно размещенные датчики. Однако если $p = r$ и число датчиков равно числу мод, то число обусловленности обычно очень велико. На самом деле матрица $\mathbf{\Theta}$ часто оказывается численно вырожденной, так как число обусловленности близко к 10^{16}. Избыточная выборка, описанная в разделе 1.8, резко улучшает число обусловленности, и уже при $p = r + 10$ качество реконструкции обычно получается гораздо выше.

QRP-разложение для разреженных датчиков. Жадное QR-разложение с перестановкой столбцов (column pivoting), или QRP-разложение матрицы $\mathbf{\Psi}_r^T$, изученное в работе Drmac and Gugercin [159] для моделей пониженного порядка, дает особенно простую и эффективную оптимизацию размещения датчиков. Метод QRP-разложения быстро работает, легко реализуется и дает почти оптимальные датчики, соответствующие конкретному базису SVD/POD. QR-разложение оптимизировано в большинстве библиотек для научных расчетов, в т. ч. MATLAB, LAPACK и NumPy. Кроме того, алгоритм QR можно ускорить, если заканчивать процедуру после первых p перестановок.

Редуцированное QR-разложение с перестановкой столбцов матрицы $\mathbf{A} \in \mathbb{R}^{m \times n}$ – это совокупность унитарной матрицы $\mathbf{Q}$, верхнетреугольной матрицы $\mathbf{R}$ и матрицы перестановки столбцов $\mathbf{C}^T$ таких, что $\mathbf{AC}^T = \mathbf{QR}$. Процедура перестановки предлагает приближенный жадный метод минимизации объема матрицы[1], равного абсолютной величине ее определителя. Перестановка столбцов увеличивает объем подматрицы, построенной из переставленных столбцов путем выбора нового опорного столбца с максимальной нормой ℓ_2 с последующим вычитанием из каждого столбца, кроме опорного, его ортогональной проекции на опорный столбец.

Следовательно, QRP-разложение дает r точечных датчиков (опор), которые определяют наилучшую выборку r базисных мод $\mathbf{\Psi}_r$

$$\mathbf{\Psi}_r^T \mathbf{C}^T = \mathbf{QR}. \tag{3.22}$$

Основываясь на том же принципе QRP-разложения, т. е. контроле над числом обусловленности путем минимизации объема, мы можем обработать случай избыточной выборки, выполнив QRP-разложение матрицы $\mathbf{\Psi}_r \mathbf{\Psi}_r^T$:

$$(\mathbf{\Psi}_r \mathbf{\Psi}_r^T)\mathbf{C}^T = \mathbf{QR}. \tag{3.23}$$

Ниже приведен код для обоих случаев.

```
if (p==r) % размещение датчиков: QR, p = r
    [Q,R,pivot] = qr(Psi_r','vector');
elseif (p>r) % размещение датчиков с избыточной выборкой: QR, p > r
    [Q,R,pivot] = qr(Psi_r*Psi_r','vector');
end
C = zeros(p,n);
for j=1:p
    C(j,pivot(j))=1;
end
```

[1] Имеется в виду объем параллелепипеда, натянутого на векторы-столбцы матрицы. – *Прим. перев.*

Пример: реконструкция лица с помощью разреженных датчиков

Для демонстрации идеи реконструкции сигнала в специальном базисе спроектируем оптимизированные разреженные датчики в библиотеке собственных лиц из раздела 1.6. На рис. 3.21 показаны QR-размещение датчиков и реконструкция, а также результат реконструкции при случайном размещении датчиков. Мы использовали $p = 100$ датчиков в библиотеке с $r = 100$ модами. В программе предполагается, что лица уже загружены, а сингулярные векторы находятся в матрице **U**. Оптимизированные методом QRP-разложения датчики дают более точную реконструкцию, ошибка реконструкции примерно в три раза меньше. Кроме того, число обусловленности на несколько порядков меньше, чем при случайно размещенных датчиках. Результаты как для QR, так и для случайного размещения можно улучшить с помощью избыточной выборки. В программе ниже вычисляется размещение датчиков (методом QRP-разложения) и производится приближенная реконструкция по этим датчикам.

```
R = 100; p = 100; % # число мод r, число датчиков p
Psi = U(:,1:r);
[Q,R,pivot] = qr(Psi','vector');
C = zeros(p,n*m);
for j=1:p
    C(j,pivot(j))=1;
end
%
Theta = C*Psi;
y = faces(pivot(1:p),1);  % измерить в точке размещения датчика
a = Theta\y;              % оценить коэффициенты
faceRecon = U(:,1:r)*a;   % реконструировать лицо
```

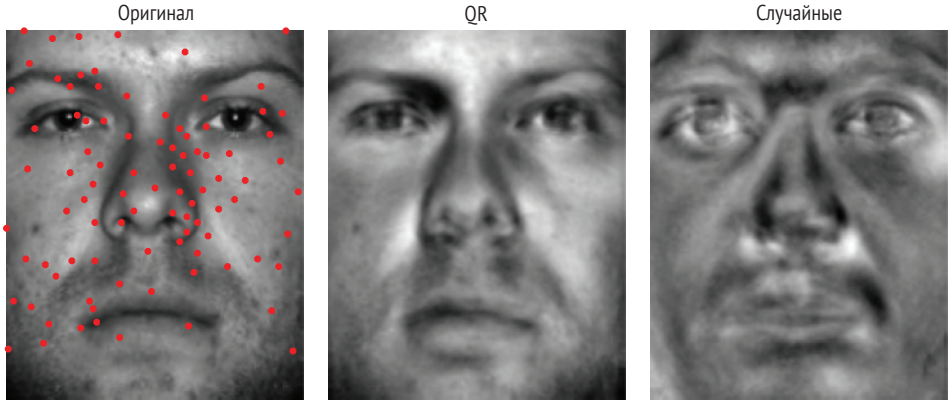

Рис. 3.21 ❖ (слева) Исходное изображение и $p = 100$ датчиков, вычисленных методом QRP-разложения, в библиотеке с $r = 100$ модами; (в середине) результат реконструкции по QR-датчикам; (справа) результат реконструкции по случайным датчикам

Разреженная классификация

Для классификации изображений нужно даже меньше датчиков, чем для реконструкции. Например, можно выбрать разреженные датчики, содержащие наиболее характерную информацию для отнесения к одной из двух категорий данных [89]. Если дана библиотека r SVD-мод $\mathbf{\Psi}_r$, то зачастую можно найти вектор $\mathbf{w} \in \mathbb{R}^r$, который лучше всего разделяет эти категории; это описано в разделе 5.6 и показано на рис. 3.22. Разреженные датчики $\mathbf{s}$, которые проецируют на это различающее направление, отбрасывая всю остальную информацию, ищутся посредством решения задачи

$$\mathbf{s} = \underset{\mathbf{s}'}{\operatorname{argmin}} ||\mathbf{s}'||_1 \quad \text{при условии} \quad \mathbf{\Psi}_r^T \mathbf{s}' = \mathbf{w}. \tag{3.24}$$

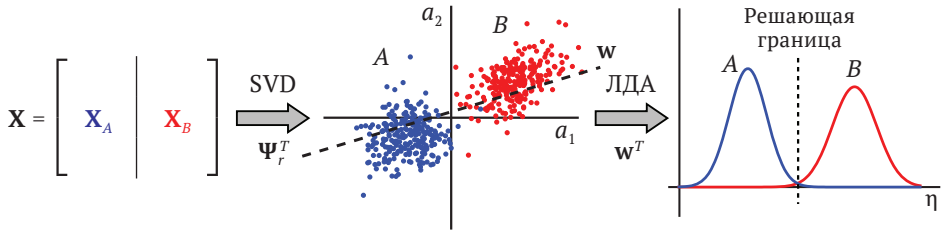

Рис. 3.22 ❖ Схематическая иллюстрация применения SVD для выделения признаков с последующим ЛДА для автоматической классификации с двумя классами A и B.
Взято из работы Bai et al. [29] с разрешения авторов

Эта оптимизация разреженного размещения датчиков для классификации (sparse sensor placement optimization for classification – SSPOC) показана на рис. 3.23 для задачи различения собак и кошек. Библиотека $\mathbf{\Psi}_r$ содержит первые *r собственных питомцев*, а вектор $\mathbf{w}$ идентифицирует ключевые различия между собаками и кошками. Заметим, что этому вектору все равно, сколько степеней свободы характеризуют различные признаки внутри кластеров

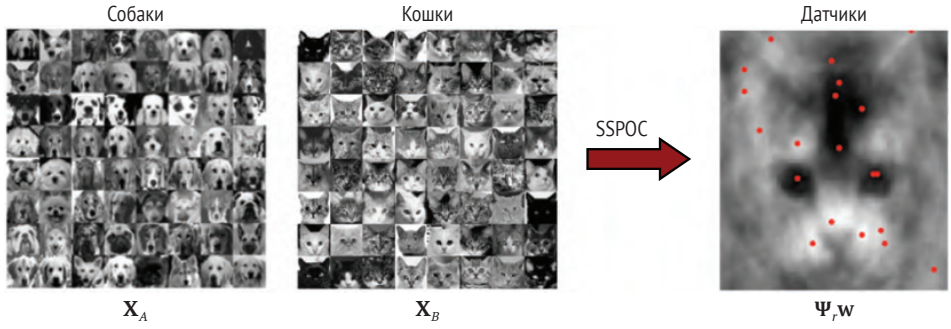

Рис. 3.23 ❖ Оптимизация разреженного размещения датчиков для классификации (SSPOC) на примере размещения датчиков для классификации собак и кошек.
Взято из работы Brunton et al. [89] с разрешения авторов

собак и кошек, ему важны лишь различия между двумя классами. Оптимизированные датчики совмещены с представляющими интерес областями: глаза, нос, рот, уши.

Рекомендуемая литература

Статьи и обзоры

(1) **Regression shrinkage and selection via the lasso**, by R. Tibshirani, *Journal of the Royal Statistical Society B*, 1996 [518].

(2) **Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information**, by E. J. Candès, J. Romberg, and T. Tao, *IEEE Transactions on Automatic Control*, 2006 [111].

(3) **Compressed sensing**, by D. L. Donoho, *IEEE Transactions on Information Theory*, 2006 [150].

(4) **Compressive sensing**, by R. G. Baraniuk, *IEEE Signal Processing Magazine*, 2007 [39].

(5) **Robust face recognition via sparse representation**, by J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009 [560].

(6) **Robust principal component analysis?**, by E. J. Candès, X. Li, Y. Ma, and J. Wright, *Journal of the ACM*, 2011 [110].

(7) **Signal recovery from random measurements via orthogonal matching pursuit**, by J. A. Tropp and A. C. Gilbert, *IEEE Transactions on Information Theory*, 2007 [529].

(8) **Data-driven sparse sensor placement**, by K. Manohar, B. W. Brunton, J. N. Kutz, and S. L. Brunton, *IEEE Control Systems Magazine*, 2018 [366].

Часть II

МАШИННОЕ ОБУЧЕНИЕ И АНАЛИЗ ДАННЫХ

Глава 4

Регрессия и выбор модели

Все машинное обучение крутится вокруг оптимизации. Сюда входят схемы регрессии и выбора модели, цель которых – подобрать подходящую экономную и допускающую интерпретацию модель данных [266]. Аппроксимация кривой – самая простая техника регрессии, полиномиальная и экспоненциальная аппроксимации ищутся путем решения линейной системы уравнений

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{4.1}$$

Если модель не предписана, то для выбора наилучшей модели применяются методы оптимизации. Математически аппроксимация функции сводится к переопределенной или недоопределенной задаче оптимизации для линейных систем:

$$\underset{\mathbf{x}}{\operatorname{argmin}}(||\mathbf{A}\mathbf{x} - \mathbf{b}||_2 + \lambda g(\mathbf{x})) \text{ или} \tag{4.2a}$$

$$\underset{\mathbf{x}}{\operatorname{argmin}}\, g(\mathbf{x}) \quad \text{при условии} \quad ||\mathbf{A}\mathbf{x} - \mathbf{b}||_2 \leq \mu, \tag{4.2b}$$

где $g(\mathbf{x})$ – заданная функция штрафования (с параметром λ для переопределенных систем). В случае переопределенных и недоопределенных систем линейных уравнений (4.1), которые либо не имеют решений, либо имеют бесконечно много решений, для получения решения необходимо выбрать ограничение или штраф, этот процесс называется *регуляризацией*. Например, можно потребовать, чтобы решение недоопределенной системы минимизировало норму ℓ_2, т. е. $\min g(\mathbf{x}) = \min||\mathbf{x}||_2$. В более общем случае, когда рассматриваются модели нелинейной регрессии, математическая задача оптимизации принимает вид

$$\underset{\mathbf{x}}{\operatorname{argmin}}(f(\mathbf{A}, \mathbf{x}, \mathbf{b}) + \lambda g(\mathbf{x})) \text{ или} \tag{4.3a}$$

$$\underset{\mathbf{x}}{\operatorname{argmin}}\, g(\mathbf{x}) \quad \text{при условии} \quad f(\mathbf{A}, \mathbf{x}, \mathbf{b}) \leq \mu, \tag{4.3b}$$

а ее решение часто ищется методами градиентного спуска. Именно эта общая постановка является предметом алгоритмов глубокого обучения.

Помимо стратегий оптимизации, один из главных вопросов науки о данных – понять, является ли предложенная модель переобученной или недообученной. Для оценки модели применяются стратегии *перекрестной проверки*. Во всех подробностях мы будем обсуждать перекрестную проверку ниже, но основную идею можно понять из рис. 4.1. Имеющийся набор данных необходимо разделить на обучающий, тестовый и контрольный. Модель строится на основе обучающих и контрольных данных, а затем проверяется на тестовом. В случае переобучения увеличение сложности модели или количества эпох (итераций) обучения уменьшает ошибку на обучающем наборе, но увеличивает на тестовом. На рис. 4.1 (a) показано каноническое поведение переобученной модели, это наводит на мысль, что во избежание переобучения сложность модели и (или) количество эпох обучения следует ограничить. С другой стороны, недообучение не позволяет прийти к хорошей модели, что показано на рис. 4.1 (b). Однако не всегда ясно, в чем причина: то ли имеет место недообучение, то ли нужно улучшить саму модель. Важность перекрестной проверки настолько велика, что она автоматически включается в большинство алгоритмов машинного обучения в MATLAB. Запомните следующую мантру: *кто пренебрегает перекрестной проверкой, тот дурак*.

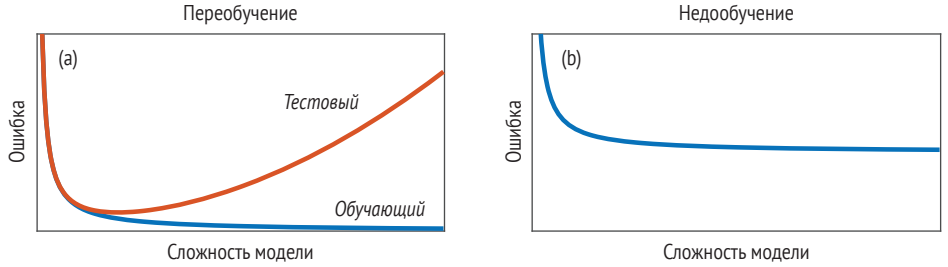

Рис. 4.1 ❖ Типичное поведение переобученной и недообученной моделей: (a) в случае переобучения увеличение сложности модели или количества эпох (итераций) обучения уменьшает ошибку на обучающем наборе, но увеличивает на тестовом; (b) в случае недообучения ошибку не удается в достаточной мере снизить вследствие ограниченной сложности модели. Эти канонические графики постоянно встречаются в науке о данных и необычайно важны при оценке модели

В следующих нескольких главах мы опишем, как оптимизация и перекрестная проверка возникают на практике и какие ограничения и структуры имеет смысл применять к $g(\mathbf{x})$, чтобы получить интерпретируемые решения. В действительности целевая функция (потеря) $f(\cdot)$ и регуляризация $g(\cdot)$ одинаково важны для определения вычислительно реализуемых стратегий оптимизации. Часто функции потерь и регуляризации выбираются так, чтобы найти не точное, а приближенное решение задачи оптимизации. Выбор сильно зависит от прикладной области и рассматриваемых данных.

4.1. Классическая аппроксимация кривой

Аппроксимация кривой – один из самых базовых инструментов науки о данных. Уже на заре развития инженерных и физических наук для понимания преобладающих тенденций, присутствующих в реальных данных, рекомендовалось применять полиномиальную аппроксимацию методом наименьших квадратов. Адриен-Мари Лежандр использовал метод наименьших квадратов еще в 1805 году для аппроксимации астрономических данных [328], а Гаусс полностью разработал теорию этого метода для решения задачи оптимизации в своей основополагающей работе 1821 года [197]. Аппроксимация кривых в астрономических приложениях оказалась весьма эффективной, поскольку планеты и кометы обращаются по простым эллиптическим орбитам (описываемым квадратичными функциями). Таким образом, можно высказать мнение, что наука о данных уже давно является краеугольным камнем научных достижений. И действительно, только потому, что Кеплер имел доступ к актуальным данным астрономических наблюдений Тихо Браге, он сумел, после одиннадцати лет исследований, сформулировать основные законы движения планет, постулировав эллиптическую форму орбит, поскольку именно такая форма наилучшим образом аппроксимировала имеющиеся данные [285].

Более широкий взгляд на аппроксимацию кривых, которого мы будем придерживаться в этой книге, называется *регрессией*. Как и в случае аппроксимации кривых, целью регрессии является оценка связи между величинами с применением различных статистических инструментов. Конкретно, мы можем рассмотреть общую связь между независимыми переменными $\mathbf{X}$, зависимыми переменными $\mathbf{Y}$ и некоторыми неизвестными параметрами $\boldsymbol{\beta}$:

$$\mathbf{Y} = f(\mathbf{X}, \boldsymbol{\beta}), \tag{4.4}$$

где функция регрессии $f(\cdot)$ обычно задана заранее, а параметры $\boldsymbol{\beta}$ ищутся путем оптимизации *согласия* этой функции с данными. Далее мы будем рассматривать аппроксимацию кривой как частный случай регрессии. Важно, что и регрессия, и аппроксимация кривой находят связи между переменными с помощью оптимизации. В широком смысле машинное обучение все посвящено методам регрессии, а те, в свою очередь, строятся вокруг оптимизации, основанной на данных. Таким образом, математически машинное обучение и наука о данных нацелены на решение задачи оптимизации. Разумеется, успех самой оптимизации в огромной степени зависит от определения подлежащей оптимизации *целевой функции*.

Методы наименьших квадратов

Для иллюстрации идеи регрессии рассмотрим классическую полиномиальную аппроксимацию методом наименьших квадратов для нахождения трендов в данных. Идея прямолинейная и бесхитростная: использовать простую функцию для описания тренда и минимизировать квадратическую ошибку

между выбранной функцией $f(\cdot)$ и данными. Классическая задача аппроксимации кривой ставится как поиск решения системы уравнений $\mathbf{Ax} = \mathbf{b}$.

Рассмотрим множество n точек

$$(x_1, y_1), (x_2, y_2), (x_3, y_3), \cdots, (x_n, y_n). \tag{4.5}$$

Пусть требуется найти прямую, которая наилучшим образом аппроксимирует эти точки. Прямую можно описать уравнением

$$f(x) = \beta_1 x + \beta_2, \tag{4.6}$$

где постоянные β_1 и β_2, образующие вектор параметров $\boldsymbol{\beta}$ в формуле (4.4), выбираются так, чтобы минимизировать некоторую ошибку аппроксимации. Аппроксимация точек прямой определяет модель *линейной регрессии* $\mathbf{Y} = f(\mathbf{A}, \boldsymbol{\beta}) = \beta_1 \mathbf{X} + \beta_2$. То есть функция описывает линейную модель, аппроксимирующую данные, а ошибка аппроксимации в каждой точке находится из равенства

$$f(x_k) = y_k + E_k, \tag{4.7}$$

где y_k – истинное значение данных, а E_k – отклонение от этого значения.

При аппроксимации заданной функцией $f(x)$ можно минимизировать разные метрики. Часто для этой цели выбирают метрики, основанные на нормах ℓ_2 (метод наименьших квадратов), ℓ_1 и ℓ_∞. Ошибки тогда определяются следующим образом:

$$E_\infty(f) = \max_{1<k<n} |f(x_k) - y_k| \qquad \text{максимальная ошибка } (\ell_\infty); \tag{4.8a}$$

$$E_1(f) = \frac{1}{n}\sum_{k=1}^{n} |f(x_k) - y_k| \qquad \text{средняя абсолютная ошибка } (\ell_1); \tag{4.8b}$$

$$E_2(f) = \left(\frac{1}{n}\sum_{k=1}^{n} |f(x_k) - y_k|^2\right)^{1/2} \qquad \text{среднеквадратическая ошибка } (\ell_2). \tag{4.8c}$$

Подобные метрики ошибок для регрессии уже рассматривались в главе 1, но сейчас мы рассмотрим их снова в контексте выбора модели. Помимо вышеперечисленных норм, стоит рассмотреть ошибку, основанную на норме общего вида ℓ_p

$$E_p(f) = \left(\frac{1}{n}\sum_{k=1}^{n} |f(x_k) - y_k|^p\right)^{1/p}. \tag{4.9}$$

Наилучшая прямая зависит от значения p. В большинстве случаев различия невелики. Но если в данных присутствуют выбросы, то выбор нормы может иметь решающее значение.

При подгонке кривой к данным часто стараются минимизировать среднеквадратическую ошибку (СКО) (4.8c). Это называется *аппроксимацией методом наименьших квадратов*. На рис. 4.2 изображены три прямые, минимизирующие ошибки E_∞, E_1 и E_2 соответственно. На ошибку E_∞ сильное влияние оказывает единственная точка, выпадающая из общего тренда. Прямые, со-

ответствующие ошибкам E_1 и E_2, хорошо аппроксимируют основной массив данных, хотя их угловые коэффициенты заметно отличаются от прямых, соответствующих случаю без выбросов. Линейные модели для всех трех метрик строятся командой MATLAB **fminsearch**. Общий для них код приведен в листинге ниже.

Листинг 4.1 ❖ Линейная регрессия

```
% Данные
x=[1 2 3 4 5 6 7 8 9 10]
y=[0.2 0.5 0.3 3.5 1.0 1.5 1.8 2.0 2.3 2.2]

p1=fminsearch('fit1',[1 1],[],x,y);
p2=fminsearch('fit2',[1 1],[],x,y);
p3=fminsearch('fit3',[1 1],[],x,y);

xf=0:0.1:11
y1=polyval(p1,xf); y2=polyval(p2,xf); y3=polyval(p3,xf);

subplot(2,1,2)
plot(xf,y1,'k'), hold on
plot(xf,y2,'k--','Linewidth',[2])
plot(xf,y3,'k','Linewidth',[2])
plot(x,y,'ro','Linewidth',[2]), hold on
```

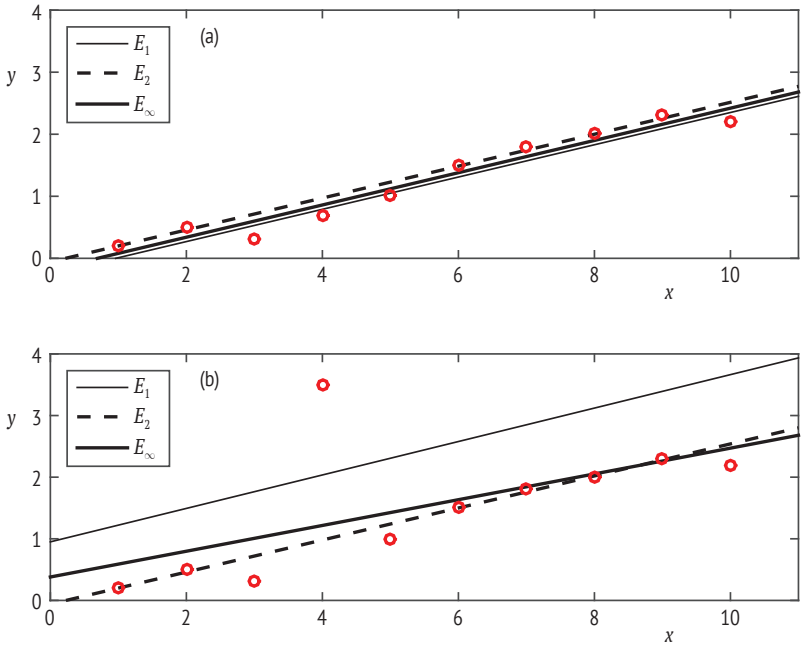

Рис. 4.2 ❖ Аппроксимация прямой с тремя разными метриками: E_∞, E_1 и E_2. В случае (a) выбросов нет, поэтому все три линейные модели дают хотя и разные, но очень похожие прямые. При наличии выбросов, как в случае (b), предсказания значительно различаются

Для каждой метрики необходимо произвести вычисления по одной из формул (4.8). Команде **fminsearch** нужно сообщить имя минимизируемой целевой функции. Функции, соответствующие трем рассматриваемым метрикам ошибок, приведены ниже.

Листинг 4.2 ❖ Максимальная ошибка ℓ_∞

```
function E=fit1(x0,x,y)
E=max(abs( x0(1)*x+x0(2)-y ));
```

Листинг 4.3 ❖ Средняя абсолютная ошибка ℓ_1

```
function E=fit2(x0,x,y)
E=sum(abs( x0(1)*x+x0(2)-y ));
```

Листинг 4.4 ❖ Среднеквадратическая ошибка ℓ_2

```
function E=fit3(x0,x,y)
E=sum(abs( x0(1)*x+x0(2)-y ).^2 );
```

Наконец, добавим в данные выброс, чтобы проиллюстрировать влияние выбранной метрики на модель линейной регрессии.

Листинг 4.5 ❖ Данные с выбросом

```
x=[1 2 3 4 5 6 7 8 9 10]
y=[0.2 0.5 0.3 0.7 1.0 1.5 1.8 2.0 2.3 2.2]
```

Линия наименьших квадратов

Линейная аппроксимация методом наименьших квадратов обладает важными преимуществами по сравнению с другими нормами и метриками. Конкретно, оптимизация обходится дешево, поскольку ошибку можно вычислить аналитически. Чтобы убедиться в этом, рассмотрим применение критерия наименьших квадратов к точкам (x_k, y_k), где $k = 1, 2, 3, \cdots, n$. Для аппроксимации этих данных прямой

$$f(x) = \beta_1 x + \beta_2 \tag{4.10}$$

необходимо минимизировать сумму

$$E_2(f) = \sum_{k=1}^{n} |f(x_k) - y_k|^2 = \sum_{k=1}^{n} (\beta_1 x_k + \beta_2 - y_k)^2. \tag{4.11}$$

Для минимизации этой суммы нам понадобится дифференцирование. Постоянные β_1 и β_2 выбираются так, чтобы достигался минимум. Следовательно, мы требуем, чтобы $\partial E_2/\partial\beta_1 = 0$ и $\partial E_2/\partial\beta_2 = 0$. Заметим, что обращение производной в нуль может означать как минимум, так и максимум, но в данном случае мы знаем, что это минимум, потому что максимума ошибки не существует – всегда можно выбрать прямую, для которой ошибка больше. Условие минимизации дает:

$$\frac{\partial E_2}{\partial \beta_1} = 0: \quad \sum_{k=1}^{n} 2(\beta_1 x_k + \beta_2 - y_k)x_k = 0; \tag{4.12a}$$

$$\frac{\partial E_2}{\partial \beta_2} = 0: \quad \sum_{k=1}^{n} 2(\beta_1 x_k + \beta_2 - y_k) = 0. \tag{4.12b}$$

После перегруппировки членов получаем систему двух линейных уравнений с двумя неизвестными:

$$\begin{pmatrix} \sum_{k=1}^{n} x_k^2 & \sum_{k=1}^{n} x_k \\ \sum_{k=1}^{n} x_k & n \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{n} x_k y_k \\ \sum_{k=1}^{n} y_k \end{pmatrix} \quad \rightarrow \quad \mathbf{Ax} = \mathbf{b}. \tag{4.13}$$

Эту систему можно решить, воспользовавшись командой \ в MATLAB. Поэтому процедура оптимизации не нужна, т. к. решение можно найти точно в явном виде.

Этот метод легко обобщается на аппроксимацию полиномами более высокой степени. В частности, для аппроксимации набора данных параболой нужно найти функцию

$$f(x) = \beta_1 x_2 + \beta_2 x + \beta_3, \tag{4.14}$$

т. е. теперь уже три постоянных β_1, β_2 и β_3. Они находятся из системы трех уравнений с тремя неизвестными, которая получается из условий минимизации ошибки $E_2(\beta_1, \beta_2, \beta_3)$ – приравниванием к нулю частных производных:

$$\frac{\partial E_2}{\partial \beta_1} = 0; \tag{4.15a}$$

$$\frac{\partial E_2}{\partial \beta_2} = 0; \tag{4.15b}$$

$$\frac{\partial E_2}{\partial \beta_3} = 0. \tag{4.15c}$$

На самом деле аппроксимация полиномом любой степени k сводится к решению системы линейных уравнений $\mathbf{Ax} = \mathbf{b}$ с матрицей размера $(k + 1)\times(k + 1)$.

Линеаризация данных

Каким бы мощным ни был метод минимизации, в общем случае аппроксимации функцией произвольного вида могут получиться уравнения, решить которые нетривиально. Рассмотрим, например, аппроксимацию данных экспоненциальной функцией

$$f(x) = \beta_2 \exp(\beta_1 x). \tag{4.16}$$

Подлежащая минимизации ошибка имеет вид

$$E_2(\beta_1, \beta_2) = \sum_{k=1}^{n} (\beta_2 \exp(\beta_1 x_k - y_k)^2. \tag{4.17}$$

Условия минимизации записываются в виде

$$\frac{\partial E_2}{\partial \beta_1} = 0: \quad \sum_{k=1}^{n} 2(\beta_2 \exp(\beta_1 x_k) - y_k)\beta_2 x_k \exp(\beta_1 x_k) = 0; \tag{4.18a}$$

$$\frac{\partial E_2}{\partial \beta_2} = 0: \quad \sum_{k=1}^{n} 2(\beta_2 \exp(\beta_1 x_k) - y_k)\exp(\beta_1 x_k) = 0, \tag{4.18b}$$

что, в свою очередь, приводит к системе двух уравнений

$$\beta_2 \sum_{k=1}^{n} x_k \exp(2\beta_1 x_k) - \sum_{k=1}^{n} x_k y_k \exp(\beta_1 x_k) = 0; \tag{4.19a}$$

$$\beta_2 \sum_{k=1}^{n} \exp(2\beta_1 x_k) - \sum_{k=1}^{n} y_k \exp(\beta_1 x_k) = 0. \tag{4.19b}$$

Эта система уравнений нелинейна и не допускает явного решения. Более того, она может даже не иметь решения. Или иметь много решений. В разделе 4.2 описывается одна из возможных итеративных процедур решения этой нелинейной системы – метод градиентного спуска.

Чтобы обойти трудности решения нелинейной системы, экспоненциальную аппроксимацию можно *линеаризовать*, применив преобразование

$$Y = \ln(y); \tag{4.20a}$$

$$X = x; \tag{4.20b}$$

$$\beta_3 = \ln \beta_2. \tag{4.20c}$$

Тогда аппроксимирующую функцию

$$f(x) = y = \beta_2 \exp(\beta_1 x) \tag{4.21}$$

можно линеаризовать, взяв натуральный логарифм обеих частей

$$\ln y = \ln(\beta_2 \exp(\beta_1 x)) = \ln \beta_2 + \ln(\exp(\beta_1 x)) = \beta_3 + \beta_1 x \Rightarrow Y = \beta_1 X + \beta_3. \tag{4.22}$$

Благодаря переходу к натуральному логарифму данных y

$$(x_i, y_i) \to (x_i, \ln y_i) = (X_i, Y_i) \tag{4.23}$$

мы свели задачу аппроксимации экспоненциальной функцией к задаче линейной аппроксимации, которая легко решается. Таким образом, если существует преобразование, линеаризующее данные, то для решения результирующей линейной системы $\mathbf{Ax} = \mathbf{b}$ можно применить стандартные методы полиномиальной аппроксимации.

4.2. Нелинейная регрессия и градиентный спуск

Полиномиальная и экспоненциальная аппроксимации кривой допускают аналитическое решение методом наименьших квадратов. Но это очень частные случаи, а для решения широкого круга задач необходим более общий математический аппарат. Допустим, к примеру, что нужно аппроксимировать набор данных нелинейной функцией вида $f(x) = \beta_1 \cos(\beta_2 x + \beta_3) + \beta_4$. Аппроксимация функцией общего вида приводит к системе нелинейных уравнений. В общей теории нелинейной регрессии предполагается, что аппроксимирующая функция имеет вид

$$f(x) = f(x, \boldsymbol{\beta}), \tag{4.24}$$

где для минимизации ошибки требуется подобрать вектор параметров $\boldsymbol{\beta} \in \mathbb{R}^m$, $m < n$. Тогда среднеквадратическая ошибка определяется следующим образом:

$$E_2(\beta) = \sum_{k=1}^{n} (f(x_k, \beta) - y_k)^2, \tag{4.25}$$

и для решения задачи минимизации следует рассмотреть систему m уравнений с m неизвестными, получающуюся приравниванием к нулю частных производных по каждому параметру β_j:

$$\frac{\partial E_2}{\partial \beta_j} = 0, \quad j = 1, 2, \cdots, m. \tag{4.26}$$

В общем случае эта система уравнений *нелинейна*:

$$\sum_{k=1}^{n} (f(x_k, \beta) - y_k) \frac{\partial f}{\partial \beta_j} = 0, \quad j = 1, 2, 3, \cdots, m. \tag{4.27}$$

Не существует общих методов решения таких нелинейных систем. Нелинейная система может вообще не иметь решений, иметь несколько решений или даже бесконечно много решений. Большинство попыток решить нелинейную систему основано на итеративных схемах. Вообще говоря, общая процедура аппроксимации проста и позволяет построить наилучшую аппроксимацию данных. Однако для быстрой сходимости к решению с минимальной ошибкой необходимо удачно выбрать начальное приближение.

На рис. 4.3 показано два примера минимизируемых функций. Первая функция выпуклая (рис. 4.3 (а)). Выпуклые функции в этом смысле идеальны, потому что для многих алгоритмов существуют гарантии сходимости, и для таких функций градиентный спуск работает отлично. Вторая функция невыпуклая, и на ее примере мы видим типичные проблемы градиентного спуска, в т. ч. тот факт, что у функции имеется несколько локальных мини-

мумов, а также плоские участки, на которых вычислить градиент трудно, поскольку он близок к нулю. Для оптимизации этой невыпуклой функции нужно правильно выбрать начальные условия алгоритма градиентного спуска, хотя существуют варианты алгоритма, предусматривающие его перезапуск и гарантирующие, что поиск не застрянет в локальном минимуме. Недавние алгоритмы обучения глубоких нейронных сетей далеко продвинулись в деле усовершенствования градиентного спуска. Мы рассмотрим их в главе 6, посвященной нейронным сетям.

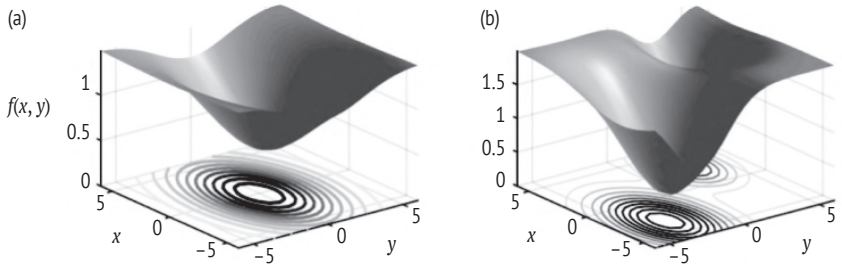

Рис. 4.3 ❖ Две поверхности, соответствующие (a) выпуклой и (b) невыпуклой целевой функциям. Для выпуклых функций имеется много гарантий сходимости, тогда как для невыпуклых существуют многочисленные проблемы, мешающие градиентному спуску. Конкретно, оптимизацию затрудняет наличие локальных минимумов и невозможность вычислить градиент в некоторых направлениях (частные производные близки к нулю)

Градиентный спуск

Для систем высокой размерности понятие минимума и максимума, т. е. экстремума функции нескольких переменных $f(\mathbf{x})$, осложняется. В точке экстремума градиент должен обращаться в нуль:

$$\nabla f(\mathbf{x}) = \mathbf{0}. \tag{4.28}$$

Поскольку в многомерных пространствах существуют седловые точки, необходимо проверять, достигается ли в точке экстремума минимум или максимум. Идея градиентного, или наискорейшего, спуска заключается в том, чтобы использовать информацию о производных в качестве основы итеративного алгоритма, сходящегося к точке локального минимума $f(\mathbf{x})$.

Чтобы проиллюстрировать, как это работает на практике, рассмотрим двумерную поверхность

$$f(x, y) = x^2 + 3y^2, \tag{4.29}$$

у которой имеется единственный минимум в начале координат $(x, y) = 0$. Градиент этой функции равен

$$\nabla f(\mathbf{x}) = \frac{\partial f}{\partial x}\hat{\mathbf{x}} + \frac{\partial f}{\partial y}\hat{\mathbf{y}} = 1x\hat{\mathbf{x}} + 6y\hat{\mathbf{y}}, \tag{4.30}$$

где $\hat{\mathbf{x}}$ и $\hat{\mathbf{y}}$ – единичные векторы вдоль осей x и y.

На рис. 4.4 показан алгоритм градиентного спуска. В качестве начального приближения вычисляем градиент $\nabla f(\mathbf{x})$. Он дает направление наискорейшего спуска к точке минимума $f(\mathbf{x})$, т. е. минимум находится в направлении, определяемом вектором $-\nabla f(\mathbf{x})$. Заметим, что градиент не указывает на минимум, он лишь определяет локально скорейший путь к минимизации $f(\mathbf{x})$. Геометрические соображения подсказывают, как построить алгоритм: на следующей итерации выбирается точка в направлении наискорейшего спуска, так что

$$\mathbf{x}_{k+1}(\delta) = \mathbf{x}_k - \delta\nabla f(\mathbf{x}_k), \tag{4.31}$$

где параметр δ определяет, как далеко продвинуться в направлении градиента. Эта формула является обобщением метода Ньютона, в котором величина производной используется для обновления на каждой итерации. В методе градиентного спуска решающую роль играет вопрос о величине шага вдоль вычисленного градиента, при котором *спуск по холму* производится оптимально. Для этого нужно выбрать правильное значение δ.

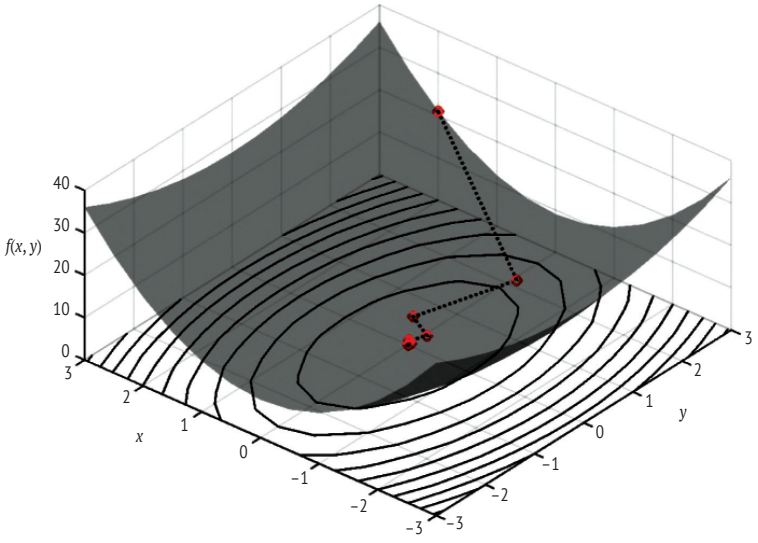

Рис. 4.4 ❖ Применение алгоритма градиентного спуска к функции $f(x, y) = x^2 + 3y^2$. В верхней части нарисованы линии уровня последовательных значений (x, y) в итеративном алгоритме с начальным приближением $(x, y) = (3, 2)$. Обратите внимание на ортогональность последовательных направлений градиента в алгоритме наискорейшего спуска. В нижней части демонстрируется быстрая сходимость к минимуму (оптимальному решению) и ошибка (E)

Для вычисления δ построим новую функцию

$$F(\delta) = f(\mathbf{x}_{k+1}(\delta)), \tag{4.32}$$

которую теперь следует минимизировать, как функцию от δ. Для этого приравняем к нулю $\partial F/\partial\delta$. Имеем:

$$\frac{\partial F}{\partial \delta} = -\nabla f(\mathbf{x}_{k+1})\nabla f(\mathbf{x}_k) = 0. \tag{4.33}$$

Геометрически этот результат интерпретируется следующим образом: $\nabla f(\mathbf{x}_k)$ – это направление градиента на текущей итерации, а $\nabla f(\mathbf{x}_{k+1})$ – на следующей итерации. Стало быть, δ выбирается так, чтобы эти направления были ортогональны.

Проведем эти вычисления для нашего примера, когда $f(x, y) = x^2 + 3y^2$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \delta\nabla f(\mathbf{x}_k) = (1 - 2\delta)x\hat{\mathbf{x}} + (1 - 6\delta)y\hat{\mathbf{y}}. \tag{4.34}$$

Это выражение используется для вычисления функции

$$F(\delta) = f(\mathbf{x}_{k+1}(\delta)) = (1 - 2\delta)^2x^2 + 3(1 - 6\delta)^2y^2, \tag{4.35}$$

производная которой по δ равна

$$F'(\delta) = -4(1 - 2\delta)x^2 - 36(1 - 6\delta)y^2. \tag{4.36}$$

Приравнивание $F'(\delta)$ к нулю дает

$$\delta = \frac{x^2 + 9y^2}{2x^2 + 54y^2}. \tag{4.37}$$

Это и есть оптимальная длина шага при спуске. Заметим, что длина δ изменяется по ходу выполнения алгоритма. Это вся информация, необходимая для поиска минимума заданной функции методом градиентного спуска.

Листинг 4.6 ❖ Пример градиентного спуска

```
x(1)=3; y(1)=2; % начальное приближение
f(1)=x(1)^2+3*y(1)^2; % начальное значение функции
for j=1:10
    del=(x(j)^2 +9*y(j)^2)/(2*x(j)^2 + 54*y(j)^2);
    x(j+1)=(1-2*del)*x(j); % обновить значения
    y(j+1)=(1-6*del)*y(j);
    f(j+1)=x(j+1)^2+3*y(j+1)^2;

    if abs(f(j+1)-f(j))<10^(-6) % проверить сходимость
        break
    end
end
```

Ясно, что этот алгоритм спуска, основанный на информации о производной, похож на метод Ньютона нахождения корня уравнения в одномерном

и многомерном случаях. На рис. 4.4 видно, как быстро этот алгоритм сходится к минимуму для данной выпуклой функции. Алгоритм градиентного спуска лежит в основе более развитых итеративных методов решения, например метода бисопряженных градиентов (**bicgstab**) и обобщенного метода минимальных невязок (**gmres**) [220].

В примере выше градиент можно было бы вычислить аналитически. В общем случае, когда имеются только сами данные, градиент вычисляется численно. Для вычисления локальных и глобальных градиентов применяется команда **gradient**. На рис. 4.5 показаны частные производные $\partial f/\partial x$ и $\partial f/\partial y$ для обеих функций рис. 4.3. Эти важнейшие для алгоритма градиентного спуска величины вычисляются так:

```
[dfx,dfy]=gradient(f,dx,dy);
```

где $f(x, y)$ – функция двух переменных, вычисляемая по известной формуле или непосредственно по данным. Результатом являются матрицы, содержащие значения $\partial f/\partial x$ и $\partial f/\partial y$ в дискретизированной области определения. Затем их можно использовать для аппроксимации локальных или глобальных градиентов в методе градиентного спуска. В следующей программе, результаты которой показаны на рис. 4.6, используется функция **interp2**, которая вычисляет значения функции рис. 4.3 (b) и ее градиента.

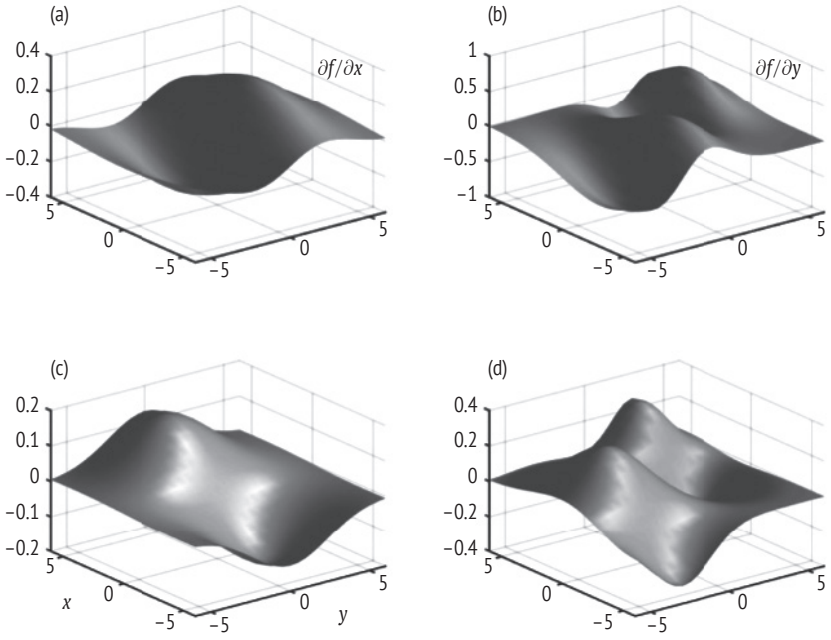

Рис. 4.5 ❖ Вычисление градиента обеих функций рис. 4.3. На левых рисунках вычисляются производные (a) $\partial f/\partial x$ и (c) $\partial f/\partial y$ для функции рис. 4.3 (a), а на правых – они же для функции рис. 4.3 (b). Для численного вычисления градиента служит команда **gradient**

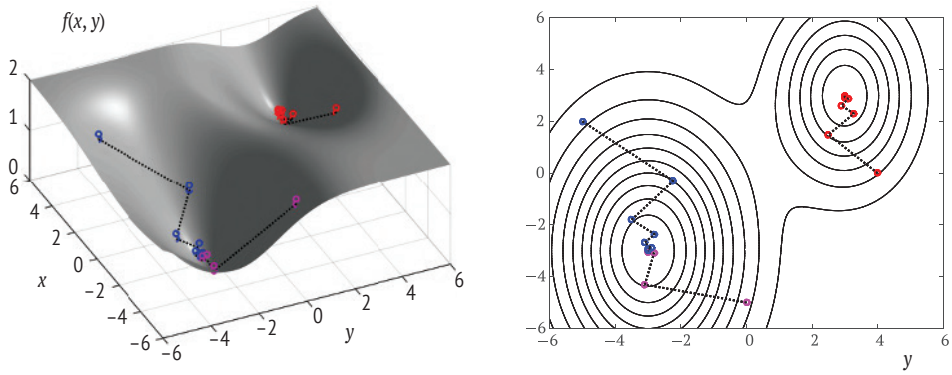

Рис. 4.6 ❖ Применение градиентного спуска в функции рис. 4.3 (b). Показаны три начальных приближения: (x_0, y_0) = {(4, 0), (0,–5), (–5, 2)}. В первом случае (красные точки) алгоритм застревает в локальном минимуме, а в остальных (синие и сиреневые точки) находит глобальный минимум. Для обновления на каждой итерации используется интерполяция частных производных, показанных на рис. 4.5

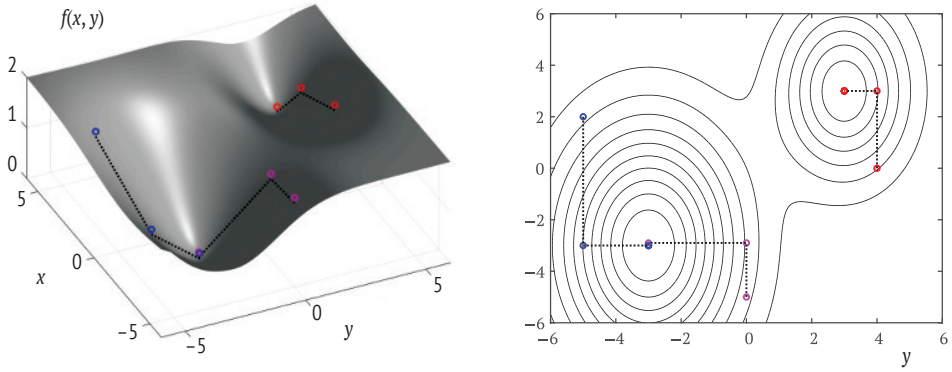

Рис. 4.7 ❖ Применение метода переменных направлений к функции рис. 4.3 (b). Показаны три начальных приближения: (x_0, y_0) = {(4, 0), (0, –5), (–5, 2)}. В первом случае (красные точки) алгоритм застревает в локальном минимуме, а в остальных (синие и сиреневые точки) находит глобальный минимум. При обновлении градиенты не используются. Обратите внимание на более высокую скорость сходимости по сравнению с рис. 4.6

Листинг 4.7 ❖ Пример градиентного спуска с интерполяцией

```
x(1)=x0(jj); y(1)=y0(jj);
f(1)=interp2(X,Y,F,x(1),y(1));
dfx=interp2(X,Y,dFx,x(1),y(1));
dfy=interp2(X,Y,dFy,x(1),y(1));

for j=1:10
    del=fminsearch('delsearch',0.2,[],x(end),y(end),dfx,dfy,X,Y,
        F); % оптимальное тау
    x(j+1)=x(j)-del*dfx; % обновить x, y, f
```

```
    y(j+1)=y(j)-del*dfy;
    f(j+1)=interp2(X,Y,F,x(j+1),y(j+1));
    dfx=interp2(X,Y,dFx,x(j+1),y(j+1));
    dfy=interp2(X,Y,dFy,x(j+1),y(j+1));

    if abs(f(j+1)-f(j))<10^(-6) % проверить сходимость
        break
    end
end
```

В этой программе для нахождения правильного значения δ используется команда **fminsearch**. За оптимизацию величины шага на каждой итерации отвечает функция

```
function mindel=delsearch(del,x,y,dfx,dfy,X,Y,F)
x0=x-del*dfx;
y0=y-del*dfy;
mindel=interp2(X,Y,F,x0,y0);
```

Мы только что обсудили самые основы градиентного спуска. Имеется целый ряд усовершенствований с целью ускорить эту основную процедуру нелинейной оптимизации, в т. ч. методы переменных направлений. Некоторые из них будут обсуждаться ниже в главе о нейронных сетях, где градиентный спуск играет важнейшую роль при обучении сети. А пока отметим, что с этой процедурой нелинейной оптимизации связан ряд проблем, в т. ч. выбор начального приближения, определение размера шага δ и эффективное вычисление градиента.

Метод переменных направлений

Еще одна широко распространенная техника оптимизации нелинейных функций нескольких переменных – *метод переменных направлений* (alternating direction method – ADM). Вместо того чтобы вычислять градиенты по нескольким переменным, оптимизация производится итеративно, по одной переменной за раз. В рассмотренном выше примере это сделало бы вычисление градиента ненужным. Основная стратегия проста: искать минимум по одной переменной, зафиксировав остальные. Перебрав все переменные, повторить процесс и поступать так до тех пор, пока не будет достигнута сходимость. В следующей программе показана часть итеративной процедуры для примера на рис. 4.6. Она заменяет вычисление градиента в процессе обновления на каждой итерации.

Листинг 4.8 ❖ Применение метода переменных направлений для обновления решения

```
fx=interp2(X,Y,F,xa(1),y); xa(2)=xa(1); [~,ind]=min(fx); ya(2)=y(ind);
fy=interp2(X,Y,F,x,ya(2)); ya(3)=ya(2); [~,ind]=min(fy); xa(3)=x(ind);
```

Заметим, что в методе переменных направлений на каждом шаге производится линейный поиск по одной переменной, что потенциально ускоряет

вычисления. К тому же метод не требует вычисления производных, что во многих приложениях делает его привлекательной альтернативой.

4.3. Регрессия и уравнение $\mathbf{Ax} = \mathbf{b}$: переопределенные и недоопределенные системы

Аппроксимация кривой, как показано в двух предыдущих разделах, приводит к задаче оптимизации. Во многих случаях оптимизацию можно математически оформить как решение системы линейных уравнений $\mathbf{Ax} = \mathbf{b}$. Но перед тем как переходить к обсуждению выбора модели и различных имеющихся методов оптимизации, полезно отметить, что в современной науке о данных линейная система $\mathbf{Ax} = \mathbf{b}$ чаще всего сильно переопределена или недоопределена. В переопределенной системе число ограничений (уравнений) больше числа неизвестных, а в недоопределенной – наоборот. Таким образом, в первом случае линейная система обычно не имеет точного решения, а вместо него ищется приближенное, минимизирующее заданную ошибку. Во втором случае решений бесконечно много, и необходимо наложить какое-то ограничение, чтобы осталось единственное решение, отвечающее нашим целям. В этом разделе мы расскажем о двух нормах (ℓ_2 и ℓ_1), которые используются при поиске оптимального решения переопределенных и недоопределенных систем вида $\mathbf{Ax} = \mathbf{b}$. Какое именно решение будет найдено, сильно зависит от выбора нормы.

Прежде чем двигаться дальше, отметим, что рассматриваемая система $\mathbf{Ax} = \mathbf{b}$ – это частный случай уравнения (4.4) $\mathbf{Y} = f(\mathbf{X}, \boldsymbol{\beta})$. Поэтому решение $\mathbf{x}$ содержит *нагрузки*, или *оценки рычагов*, связывающие входные данные $\mathbf{A}$ с выходными данными $\mathbf{b}$. В простом решении этой линейной задачи используется псевдообратная матрица Мура–Пенроуза $\mathbf{A}^\dagger$, которую мы рассматривали в разделе 1.4:

$$\mathbf{x} = \mathbf{A}^\dagger\mathbf{b}. \tag{4.38}$$

В MATLAB этот оператор вычисляется командой **pinv(A)**. Но это решение слишком ограничительное, а нам хотелось бы иметь большую гибкость при вычислении решений. Конкретно в данном разделе наша цель – продемонстрировать совместное использование норм ℓ_1 и ℓ_2 при решении переопределенных и недоопределенных систем.

Переопределенные системы

На рис. 4.8 показана общая структура переопределенной системы. Как уже отмечалось, в общем случае система $\mathbf{Ax} = \mathbf{b}$ неразрешима. Поэтому при ре-

шении задачи оптимизации приходится минимизировать ошибку, например среднеквадратическую ошибку E_2, т. е. искать такое значение $\hat{\mathbf{x}}$, что:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} ||\mathbf{A}\mathbf{x} - \mathbf{b}||^2. \tag{4.39}$$

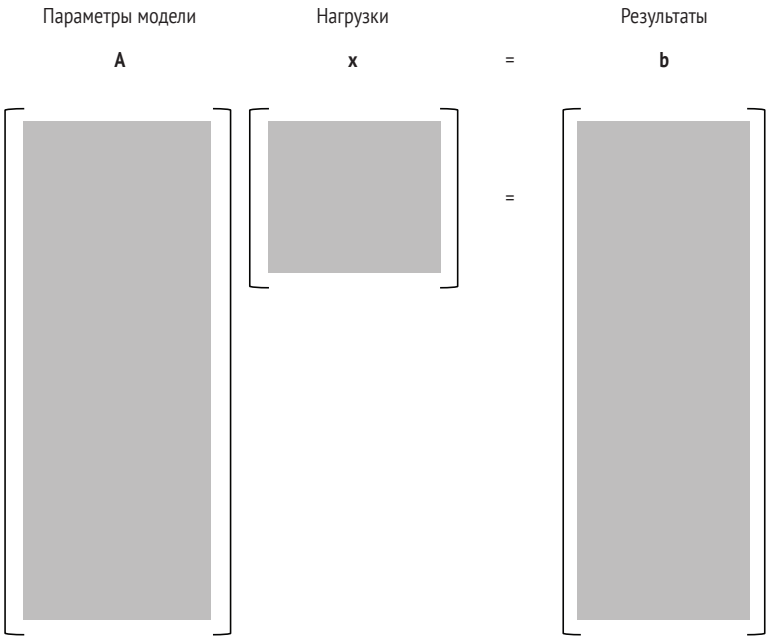

Рис. 4.8 ❖ Регрессия для переопределенных систем. В этом случае у системы **Ax** = **b** обычно нет точных решений. Поэтому решение сводится к минимизации, например, среднеквадратической ошибки $||\mathbf{Ax} - \mathbf{b}||_2$ при каком-то ограничении на **x**, скажем минимальности ℓ_2-нормы $||\mathbf{x}||_2$

В этой базовой архитектуре нет никаких явных ограничений на нагрузки **x**. Чтобы одновременно минимизировать ошибку и наложить ограничение, базовую архитектуру можно модифицировать следующим образом:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} ||\mathbf{A}\mathbf{x} - \mathbf{b}||^2 + \lambda_1 ||\mathbf{x}||_1 + \lambda_2 ||\mathbf{x}||_2, \tag{4.40}$$

где параметры λ_1 и λ_2 определяют штрафование по норме ℓ_1 и ℓ_2 соответственно. Теперь мы налагаем ограничения на сам вектор решения, а не только на ошибку. Идея штрафования путем добавления регуляризирующих ограничений критически важна для понимания того, как производится *выбор модели*.

В примерах ниже особое внимание будет уделено роли нормы ℓ_1. Как было показано в главе 3, эта норма поощряет разреженность, при которой многие нагрузки в решении **x** равны нулю. Это будет играть важную роль при выборе переменных и модели в следующем разделе. А пока займемся решением задачи оптимизации (4.40) при $\lambda_2 = 0$. Для нахождения решения мы

воспользуемся пакетом выпуклой оптимизации **cvx** с открытым исходным кодом для MATLAB [218]. В следующей программе вычисляются решения переопределенной системы с 500 ограничениями и 100 неизвестными при различных значениях штрафа по норме ℓ_1.

Листинг 4.9 ❖ Решение переопределенной системы

```
n=500; m=100;
A=rand(n,m);
b=rand(n,1);
xdag=pinv(A)*b;

lam=[0 0.1 0.5];
for j=1:3

    cvx_begin;
    variable x(m)
    minimize( norm(A*x-b,2) + lam(j)*norm(x,1) );
    cvx_end;
    subplot(4,1,j),bar(x)
    subplot(4,3,9+j), hist(x,20)
end
```

На рис. 4.9 показаны результаты процесса оптимизации в виде функции от параметра λ_1. Отметим, что решение при $\lambda_1 = 0$ эквивалентно решению **xdag**, получающемуся вычислением псевдообратной матрицы для **A**. Заметим также, что норма ℓ_1 поощряет нахождение разреженных решений, в которых многие элементы вектора **x** равны нулю. Гистограммы элементов **x** на рис. 4.9 (d)–(f) особенно показательны, поскольку демонстрируют процесс увеличения разреженности с ростом λ_1.

Регрессию для переопределенных систем можно обобщить на матричные системы, как показано на рис. 4.8. В этом случае в команде **cvx** только изменяется размер матрицы **b** и матрицы решения **x**. Рассмотрим два решения переопределенной системы, найденные следующей программой.

Листинг 4.10 ❖ Решения переопределенной матричной системы

```
n=300; m=60; p=20;
A=rand(n,m); b=rand(n,p);
lam=[0 0.1];
for j=1:2
    cvx_begin;
    variable x(m,p)
    minimize(norm(A*x-b,2) + lam(j)*norm(x,1));
    cvx_end;
    subplot(2,1,j), pcolor(x.'), colormap(hot), colorbar
end
```

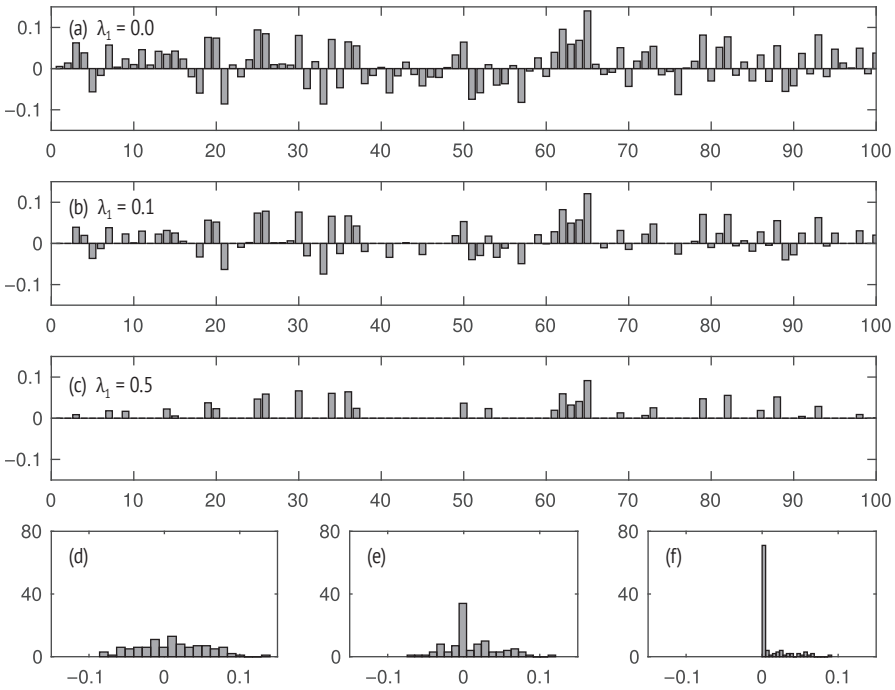

Рис. 4.9 ❖ Решения переопределенной системы с 500 ограничениями и 100 неизвестными. На рисунках (a)–(c) показаны столбчатые диаграммы нагрузок в векторе **x**. Заметим, что вместе с увеличением штрафа по норме ℓ_1 от (a) λ1 = 0 до (b) λ1 = 0.1 и (c) λ1 = 0.5 увеличивается и количество ненулевых элементов вектора, т. е. вектор становится более разреженным. Гистограммы нагрузок, соответствующие рисункам (a)–(c), показаны на рисунках (d)–(f) соответственно. Это иллюстрирует роль нормы ℓ_1 в поощрении разреженности решения

На рис. 4.10 показаны результаты решения этой переопределенной матричной системы при двух значениях штрафа по норме ℓ_1. Заметим, что добавление штрафа повышает разреженность решения, так что в результирующей матрице много нулевых элементов. Примеры на рис. 4.9 и 4.10 демонстрируют важную роль норм ℓ_2 и ℓ_1 в нахождении решений разных типов. Далее в этой книге мы воспользуемся этими нормами для порождения экономных моделей на основе данных.

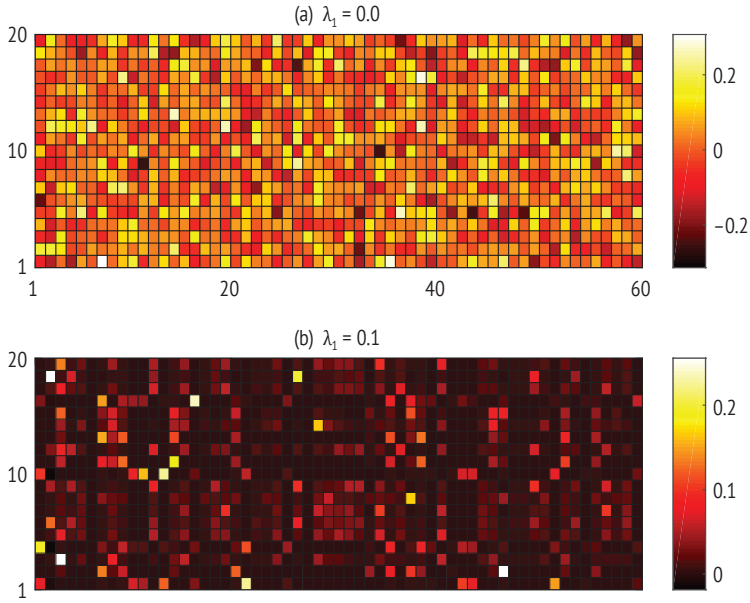

Рис. 4.10 ❖ Решения переопределенной системы **Ax** = **b** с 300 ограничениями и 60×20 неизвестными. На рисунках (a) и (b) показаны значения нагрузок в матрице **x**, когда штраф по норме ℓ_1 равен (a) $\lambda_1 = 0$ и (b) $\lambda_1 = 0.1$

Недоопределенные системы

У недоопределенной системы $\mathbf{Ax} = \mathbf{b}$ решений бесконечно много. В этом случае наша цель – наложить дополнительное ограничение или набор ограничений, так чтобы из бесконечного множества можно было выбрать единственное решение. Базовая математическая структура показана на рис. 4.11. Решение недоопределенной системы можно сформулировать как задачу оптимизации

$$\min||\mathbf{x}||_p \quad \text{при условии} \quad \mathbf{Ax} = \mathbf{b}, \tag{4.41}$$

где p обозначает норму ℓ_p вектора **x**. Для простоты мы рассматриваем только нормы ℓ_2 и ℓ_1. Как уже было показано для переопределенных систем, норма ℓ_1 поощряет разреженность решения.

Снова воспользуемся пакетом выпуклой оптимизации **cvx**. Программа в листинге 4.11 вычисляет решения недоопределенной системы (4.41) с 20 ограничениями и 100 неизвестными для норм ℓ_2 и ℓ_1.

Листинг 4.11 ❖ Решения недоопределенной матричной системы

```
n=20; m=100
A=rand(n,m); b=rand(n,1);
```

```
cvx_begin;
variable x2(m)
minimize( norm(x2,2) );
subject to
A*x2 == b;
cvx_end;

cvx_begin;
variable x1(m)
minimize( norm(x1,1) );
subject to
A*x1 == b;
cvx_end;
```

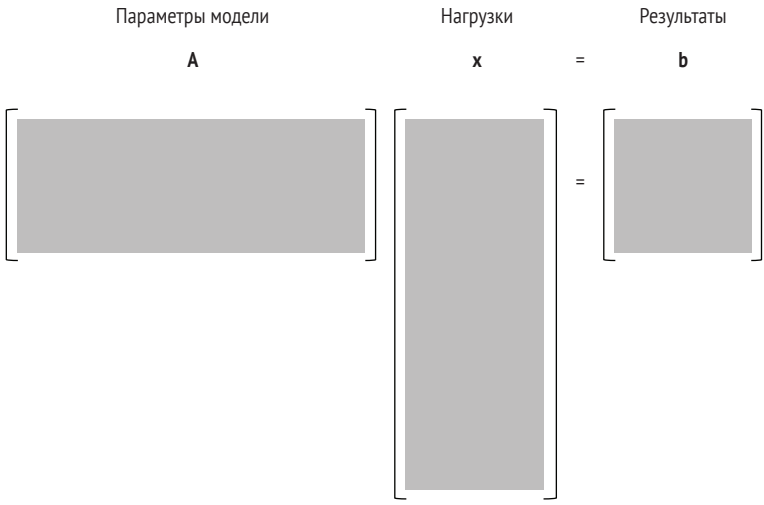

Рис. 4.11 ❖ Регрессия для недоопределенных систем. В этом случае уравнение **Ax** = **b** имеет бесконечно много решений. Поэтому для выделения единственного решения необходимо наложить ограничения в виде некоторого условия минимизации. Например, из бесконечного множества решений можно выбрать то, для которого норма $||\mathbf{x}||_2$ минимальна

Эта программа вычисляет два вектора решений **x2** и **x1** с минимальными нормами ℓ_2 и ℓ_1 соответственно. Обратите внимание, как именно **cvx** позволяет налагать ограничения в процедуре оптимизации. На рис. 4.12 показаны столбчатая диаграмма и гистограмма обоих решений. Как и раньше, норма ℓ_1, поощряющая разреженность, приводит к вектору, содержащему много нулей. На самом деле в этом случае нулей ровно 80, потому что имеется всего 20 ограничений на 100 неизвестных.

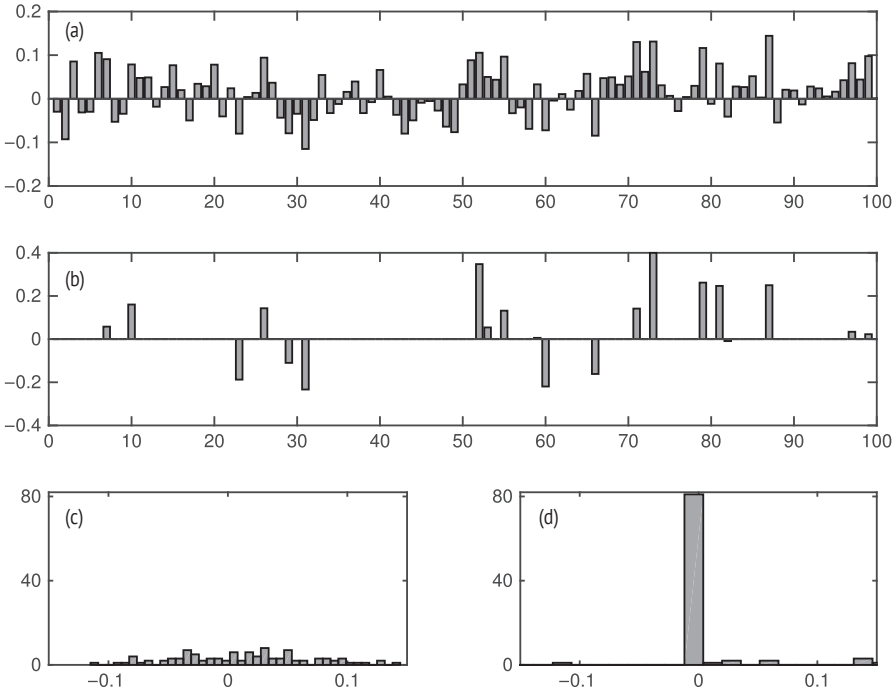

Рис. 4.12 ❖ Решения недоопределенной системы с 20 ограничениями и 100 неизвестными. На рисунках (a) и (b) показаны столбчатые диаграммы нагрузок в векторе **x**. В первом случае ищется решение с минимальной нормой ℓ_2, а во втором – с минимальной нормой ℓ_1. Отметим, что штраф по норме ℓ_1 поощряет нахождение разреженного вектора. Гистограммы нагрузок, соответствующие рисункам (a) и (b), показаны на рисунках (c) и (d) соответственно

Как и в случае переопределенных систем, процедуру оптимизации можно модифицировать для работы с более общими недоопределенными матричными уравнениями, как показано на рис. 4.11. Пакет **cvx** можно использовать и в этом случае. Программный движок способен также работать с общими нормами ℓ_p и производить одновременную минимизацию норм ℓ_1 и ℓ_2. В такой общей постановке вместо задачи (4.41) рассматривается такая:

$$\min(\lambda_1||\mathbf{x}||_1 + \lambda_2||\mathbf{x}||_2) \quad \text{при условии} \quad \mathbf{Ax} = \mathbf{b}. \tag{4.42}$$

Здесь выбор весов λ_1 и λ_2 позволяет контролировать степень разреженности решения. Применение различных стратегий оптимизации – обычное дело, мы рассмотрим этот вопрос ниже.

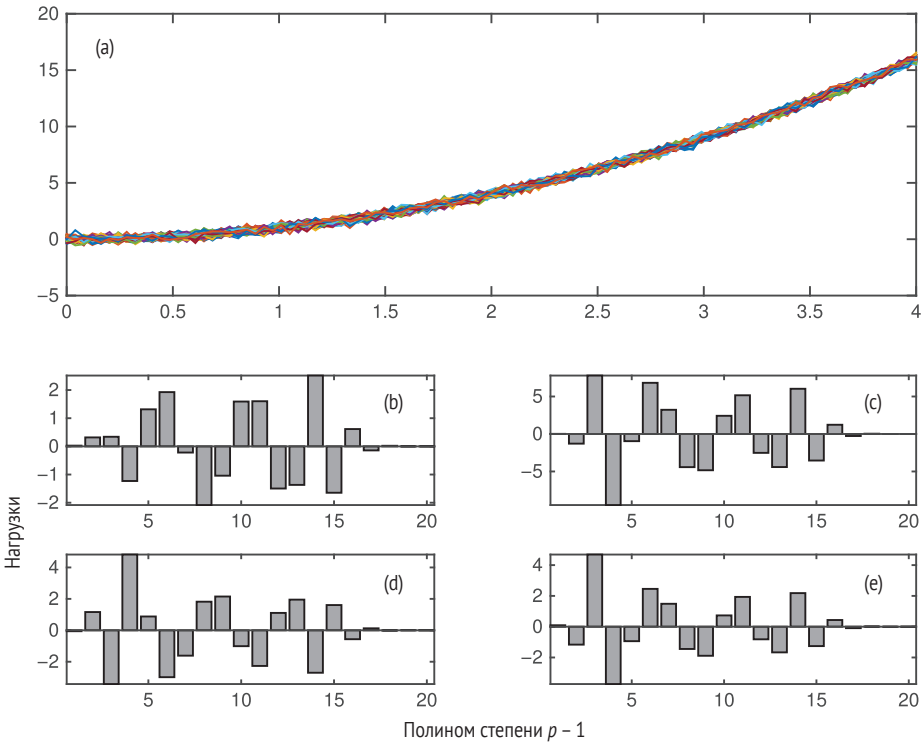

Рис. 4.13 ❖ (a) Сто реализаций параболической функции (4.43) с добавленным белым шумом с параметром $\sigma = 0.1$. Хотя шум мал, метод наименьших квадратов демонстрирует значительную изменчивость при аппроксимации полиномом двадцатой степени. На рисунках (b)–(e) показаны нагрузки (коэффициенты) при разных коэффициентах полинома для четырех реализаций шума. Это демонстрирует зависимость результатов от выбора модели

4.4. Оптимизация как краеугольный камень регрессии

В двух предшествующих разделах этой главы аппроксимирующая функция $f(x)$ была задана. Например, иногда желательно найти аппроксимирующую прямую вида $f(x) = \beta_1 x + \beta_2$. Зная общий вид, мы затем находим коэффициенты, применяя описанные выше методы регрессии и оптимизации. Теперь нашей целью будет разработка методов, которые позволят находить объективно хорошую модель для аппроксимации данных, будь то квадратичная или кубическая. Одна лишь метрика ошибки не определяет хороший выбор модели, поскольку чем больше в модели параметров, тем больше возможностей снизить ошибку, хотя дополнительные параметры могут не иметь никакого смысла и не допускают интерпретации.

Стратегии оптимизации играют главную роль при получении интерпретируемых результатов и осмысленных моделей на основе имеющихся данных. Как уже было показано в предыдущих разделах, результат оптимизации сильно зависит от взаимодействия между нормами ℓ_1 и ℓ_2. Чтобы еще больше подчеркнуть роль оптимизации и разнообразие возможных исходов, рассмотрим простой пример данных, сгенерированных по зашумленным измерениям параболической зависимости

$$f(x) = x^2 + \mathcal{N}(0, \sigma), \tag{4.43}$$

где $\mathcal{N}(0, \sigma)$ – нормально распределенная случайная величина с нулевым средним и стандартным отклонением σ. На рис. 4.13 (а) приведен пример 100 случайных измерений функции (4.43). Параболическая структура ясно просматривается, несмотря на добавленный шум. Аппроксимировать результаты параболой тривиально – нужно только воспользоваться методом наименьших квадратов, описанным в первом разделе этой главы.

Но наша цель – *выявить* наилучшую модель для имеющихся данных. На практике модель априори неизвестна, и мы должны найти соответствующую функцию. Можно начать с полиномиальных моделей регрессии. Точнее, рассмотрим задачу выбора модели $\mathbf{Y} = f(\mathbf{X}, \boldsymbol{\beta})$ (см. 4.4) как решение следующей системы $\mathbf{Ax} = \mathbf{b}$:

$$\begin{bmatrix} | & | & | & \cdots & | \\ 1 & x_j & x_j^2 & \cdots & x_j^{p-1} \\ | & | & | & \cdots & | \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{100}) \end{bmatrix}, \tag{4.44}$$

где матрица $\mathbf{A}$ содержит полиномиальные модели степени не более $p - 1$: каждая строка представляет одно измерение, β_k – коэффициенты полинома, а матрица $\mathbf{b}$ содержит результаты (данные) $f(x_j)$. Далее мы будем рассматривать случай, когда имеется 100 измерений, а для аппроксимации используется полином 19-й степени (с 20 членами). Таким образом, матричная система уравнений $\mathbf{Ax} = \mathbf{b}$ порождает переопределенную систему, как показано на рис. 4.8.

Следующая программа решает переопределенную систему (4.44) методом наименьших квадратов с помощью функции **pinv**. Мы в цикле решаем систему четыре раза, чтобы проиллюстрировать влияние, оказываемое небольшим шумом на процедуру регрессии.

Листинг 4.12 ❖ Аппроксимация зашумленной параболы полиномом методом наименьших квадратов

```
n=100; L=4;
x=linspace(0,L,n);
f=(x.^2).';             % парабола со 100 точками

M=20;                   % степень полинома
for j=1:M
```

```
phi(:,j)=(x.').^(j-1); % построить матрицу A
end

for j=1:4
    fn=(x.^2+0.1*randn(1,n)).';
    an=pinv(phi)*fn; fna=phi*an; % метод наименьших квадратов
    En=norm(f-fna)/norm(f);
    subplot(4,2,4+j),bar(an)
end
```

На рис. 4.13 (b)–(e) показаны четыре типичные нагрузки **β**, вычисленные процедурой регрессии. Заметим, что хотя добавленный шум мал, нагрузки значительно различаются. Стало быть, различные реализации шума порождают совершенно разные модели для объяснения данных.

Из-за сильной изменчивости результатов регрессии выбор модели оказывается проблематичным. Получается, что даже небольшой шум измерений может привести к абсолютно различным выводам об истинной модели. Ниже мы количественно охарактеризуем эту изменчивость, а заодно рассмотрим несколько процедур регрессии для решения переопределенной линейной системы $\mathbf{Ax} = \mathbf{b}$. Будет описано пять стандартных методов: регрессия методом наименьших квадратов (**pinv**), оператор \, LASSO (оператор наименьшего абсолютного сжатия и выборки) (**lasso**), робастная аппроксимация (**robustfit**) и гребневая регрессия (**ridge**). Для оформления математической архитектуры этих методов решения системы $\mathbf{Ax} = \mathbf{b}$ полезно вернуться к предыдущему разделу и конкретно к формуле (4.40). Псевдообращение Мура–Пенроуза (**pinv**) решает задачу (4.40) с $\lambda_1 = \lambda_2 = 0$. Команда \ решает переопределенную линейную систему с помощью QR-разложения [524]. Метод LASSO (**lasso**) решает (4.40) с $\lambda_1 > 0$ и $\lambda_2 = 0$. Гребневая регрессия (**ridge**) решает (4.40) с $\lambda_1 = 0$ и $\lambda_2 > 0$. Однако современная реализация гребневой регрессии в MATLAB имеет нюансы. Популярный алгоритм эластичной сети взвешивает оба штрафа – по нормам ℓ_2 и ℓ_1, т. е. реализует гибридную модель регрессии, среднюю между гребневой и LASSO. Робастная аппроксимация (**robustfit**) решает (4.40) методом взвешенных наименьших квадратов. Кроме того, она позволяет воспользоваться робастными статистическими методами и штрафованием по норме Хьюбера, поощряющей удаление выбросов [260]. Правда, в рассматриваемых ниже данных нет выбросов, поэтому мощь робастной аппроксимации в полной мере не задействована. Но как бы то ни было, остановиться на этой важной технике необходимо.

На рис. 4.14 показано несколько диаграмм размаха для 100 наборов случайных данных. Хорошо видно, что результат зависит от метода регрессии. Кроме того, налицо принципиальные различия стратегий оптимизации, основанных на нормах ℓ_1 и ℓ_2. С точки зрения выбора модели, аппроксимация методом наименьших квадратов дает сильную изменчивость нагрузок **β**, как видно на рис. 4.14 (a), (b) и (e). Результат аппроксимации методом наименьших квадратов был получен соответственно псевдообращением Мура–Пенроуза и QR-разложением. Если допускается штраф по норме ℓ_1 (регуляризация), то, как показывают рис. 4.14 (c), (d) и (f), выбирается более экономная модель с низкой изменчивостью. Это ожидаемо, поскольку норма

ℓ_1 поощряет разреженность вектора нагрузок $\boldsymbol{\beta}$. И действительно, стандартная LASSO-регрессия правильно определяет, что основной вклад в данные привносит квадратичный полином. Для построения этих диаграмм использовалась программа в листинге 4.13.

Листинг 4.13 ❖ Сравнение методов регрессии

```
lambda=0.1; phi2=phi(:,2:end);
for jj=1:100
    f=(x.^2+0.2*randn(1,n)).';
    a1=pinv(phi)*f; f1=phi*a1; E1(jj)=norm(f-f1)/norm(f);
    a2=phi\f; f2=phi*a2; E2(jj)=norm(f-f2)/norm(f);
    [a3,stats]=lasso(phi,f,'Lambda',lambda); f3=phi*a3; E3(jj)=
        norm(f-f3)/norm(f);
    [a4,stats]=lasso(phi,f,'Lambda',lambda,'Alpha',0.8); f4=phi*a4;
        E4(jj)=norm(f-f4)/norm(f);
    a5=robustfit(phi2,f);f5=phi*a5;E5(jj)=norm(f-f5)/norm(f);
    a6=ridge(f,phi2,0.5,0);f6=phi*a6;E6(jj)=norm(f-f6)/norm(f);

    A1(:,jj)=a1;A2(:,jj)=a2;A3(:,jj)=a3;A4(:,jj)=a4;A5(:,jj)=a5;A6
        (:,jj)=a6;
    plot(x,f), hold on
end
Err=[E1; E2; E3; E4; E5; E6];
Err2=[E1; E2; E3; E4; E5];
```

Этот код также порождает все 100 реализаций, показанных на рис. 4.13 (a).

Несмотря на сильную изменчивость большинства нагрузок, продемонстрированную на рис. 4.14 для разных методов регрессии, разброс ошибок аппроксимации невелик. Разные методы порождают регрессию с сопоставимыми ошибками. Таким образом, при всех различиях методов оптимизации ошибка аппроксимации малочувствительна к выбранному методу. Отсюда следует, что использование одной лишь ошибки в качестве критерия выбора модели потенциально проблематично, поскольку почти любой метод может дать надежную модель с низкой ошибкой. На рис. 4.15 (a) показана диаграмма размаха ошибки, полученной в результате применения методов регрессии на рис. 4.14. Все методы порождают сравнительно небольшую ошибку, слабо зависящую от выбранной стратегии.

И в завершение этого раздела мы можем рассмотреть процедуру регрессии как функцию от степени полиномов в формуле (4.44). Мы брали полиномы степени не больше 20. Но если вместо этого изменять степень полинома, то обнаруживаются интересные и важные вещи, показанные на рис. 4.15 (b)–(c). А именно ошибка регрессии уменьшается до 10^{-3} после добавления квадратичного члена, как видно на рисунке (b). Это вполне ожидаемо, поскольку модель с самого начала была квадратичной функцией с добавленным слабым шумом. Но удивительно, что при увеличении степени полинома ошибка регрессии возрастает, как явствует из рисунка (c). Следовательно, простое добавление новых членов не улучшает ошибку, что, на первый взгляд, противоречит интуиции. Ниже приведена программа, с помощью которой были получены эти результаты.

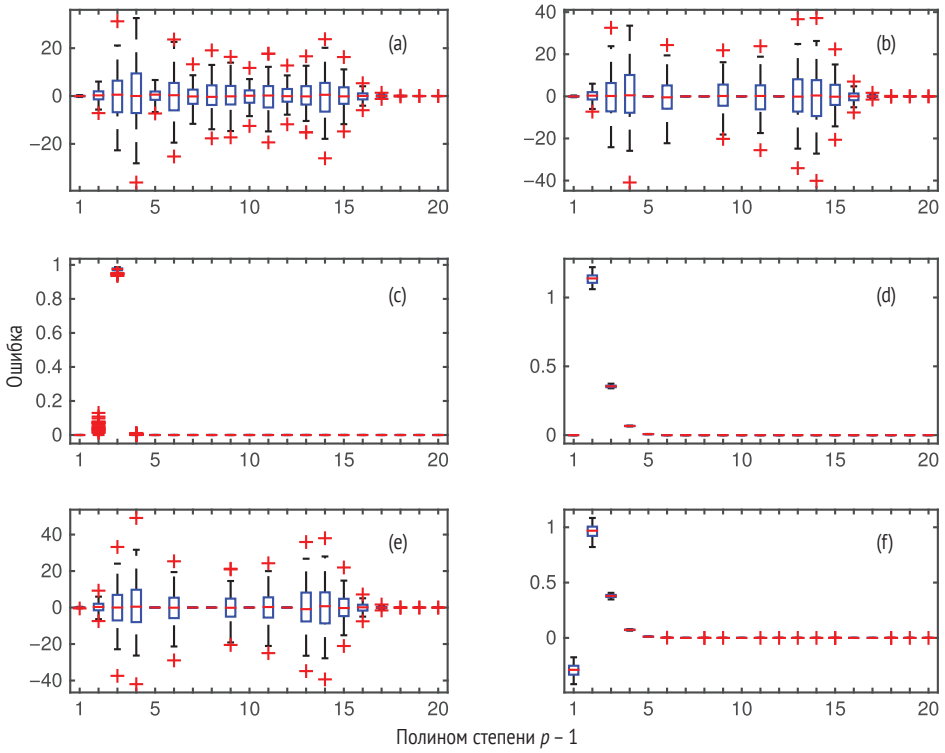

Рис. 4.14 ❖ Сравнение методов регрессии для решения переопределенной системы линейных уравнений **Ax** = **b**. Все 100 реализаций данных сгенерированы на основе простой параболы (4.43), аппроксимируемой полиномом 20-й степени согласно (4.44). На диаграммах размаха показаны: (a) регрессия наименьших квадратов посредством псевдообращения Мура–Пенроуза (**pinv**), (b) команда \, (c) LASSO-регрессия (**lasso**), (d) LASSO-регрессия с другим соотношением штрафов по нормам ℓ_2 и ℓ_1, (e) робастная аппроксимация, (f) гребневая регрессия. Отметим значительную изменчивость значений нагрузок для методов, основанных только на ℓ_2 ((a), (b) и (e)), и низкую изменчивость для методов, основанных на норме ℓ_1 ((c), (d) и (f)). Только стандартная LASSO-регрессия (c) выявляет преобладание параболического члена

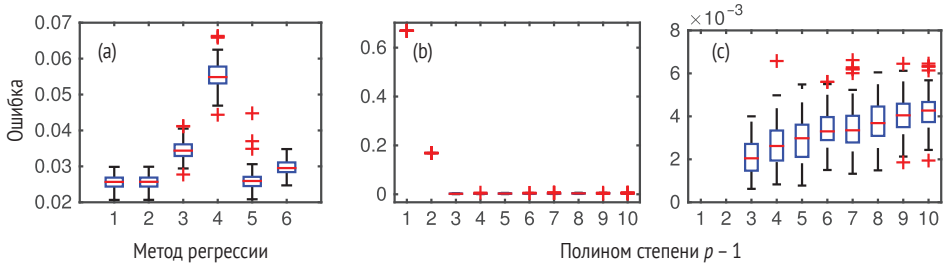

Рис. 4.15 ❖ (a) Сравнение ошибки для шести методов регрессии на рис. 4.14. Несмотря на изменчивость, присущую методам оптимизации, все они находят решения с малой ошибкой; (b) ошибка регрессии методом наименьших квадратов как функция возрастающей степени полинома. Ошибка быстро уменьшается после добавления квадратичного члена; (c) детальное исследование ошибки, показывающее, что она немного увеличивается при увеличении степени аппроксимирующего полинома

Листинг 4.14 ❖ Аппроксимация модели полиномами разной степени

```
En=zeros(100,M);
for jj=1:M
    for j=1:jj
        phi(:,j)=(x.').^(j-1);
    end
    f=(x.^2).';
    for j=1:100
        fn=(x.^2+0.1*randn(1,n)).';
        an=pinv(phi)*fn; fna=phi*an;
        En(j,jj)=norm(f-fna)/norm(f);
    end
end
```

Отметим, что мы рассматривали только полиномы степени не больше 10. Еще отметим, что рисунок (c) на рис. 4.15 – детализация рисунка (b). Ошибка аппроксимации простой параболой примерно в два раза меньше, чем при аппроксимации полиномом степени 10. Эти результаты помогут нам формализовать выбор модели в следующих разделах.

4.5. Парето-фронт и Lex Parsimoniae

В предыдущих главах мы показали, что регрессия – это несколько больше, чем простой выбор модели и аппроксимация методом наименьших квадратов. Мало того что существуют различные нормы для ограничения множества решений, так еще и саму модель необходимо тщательно выбирать, чтобы добиться описания данных, допускающего лучшую интерпретацию. Соображения о выборе подходящей модели восходят еще к Уильяму Оккаму (ок. 1287–1347), английскому францисканскому монаху, философу-схоласту и теологу. Оккам предложил свой принцип экономии (лат. *lex parsimoniae*), более известный под названием «бритва Оккама», утверждающий, что среди нескольких возможных гипотез следует предпочесть ту, что требует меньше допущений, а из двух теорий, приводящих к одинаковым предсказаниям, правильна, скорее всего, более простая. Идея бритвы Оккама широко использовалась в физике и биологии для вывода уравнений, описывающих наблюдаемые явления.

Экономия играет также главную роль в математических трудах итальянца Вильфредо Парето (ок. 1848–1923). Парето был инженером, социологом, экономистом, политологом и философом. Он внес весомый вклад в экономику, особенно в области распределения доходов и анализа индивидуального выбора. Также благодаря ему получил распространение термин «элита» в социологии. Позже получило известность его знаменитое правило 80/20, которое качественно показано на рис. 4.16. В 1941 году консультант по вопросам управления Джозеф М. Джуран назвал это правило принципом Парето. Простыми словами, этот принцип в применении к бизнесу означает, например, что 80 % продаж приходится на 20 % покупателей. Эта идея была популяризирована в книге Ричарда Коха «The 80/20 Principle»[1] [294] (и в не-

[1] *Кох Р.* Принцип 80/20. Бомбора, 2020.

скольких ее продолжениях [295, 296, 297]), где описывается ряд практических применений принципа Парето в управлении бизнесом и в жизни.

Парето и Оккам по существу исповедовали одну и ту же философию: объяснять большую часть наблюдаемых данных экономной моделью. Важно, что выбор модели не сводится к уменьшению ошибки – требуется, чтобы модель допускала понятную интерпретацию, была обобщаемой и обладала предсказательной способностью. На рис. 4.16 показана базовая идея *Парето-фронта* и решения, *оптимального по Парето*. Конкретно, для каждой рассматриваемой модели вычисляются параметры и ошибка аппроксимации данных. Решения с наименьшей ошибкой при заданном числе параметров определяют Парето-фронт. Эти экономные решения, в которых ошибка оптимально сбалансирована со сложностью, находятся в серой области и называются оптимальными по Парето. В теории игр оптимальным по Парето решением является стратегия, которая не может показать лучшие результаты в состязании с одной стратегией без ухудшения результатов в состязании с другой (в данном случае речь идет об ошибке и сложности). В экономике оно описывает ситуацию, когда выгода одной стороны не может быть увеличена без уменьшения выгоды другой. Наша цель – опираясь на научные принципы, выбрать наилучшую модель из множества оптимальных по Парето решений. Для этого мы будем использовать информационные критерии, обсуждаемые в последующих разделах.

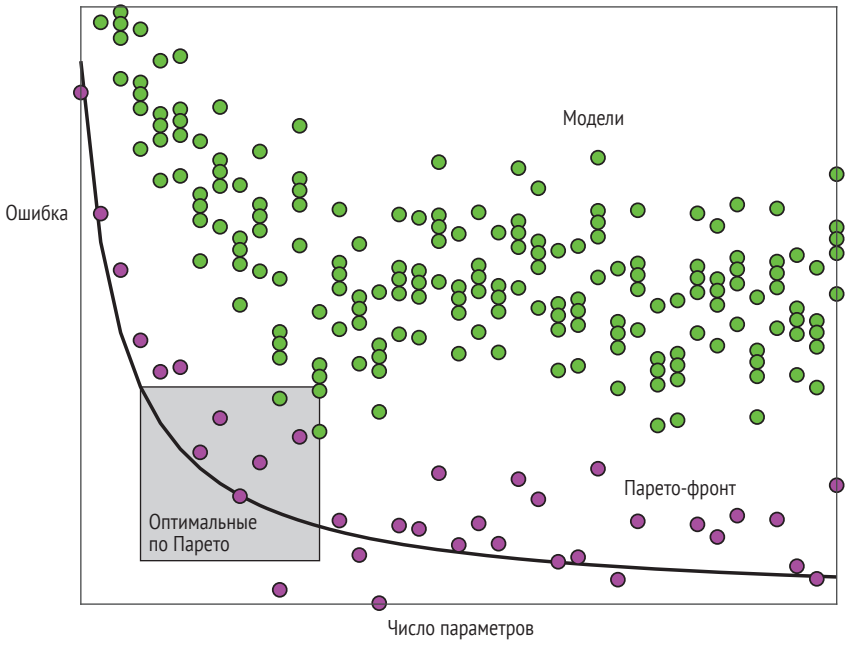

Рис. 4.16 ❖ При выборе модели ищется компромисс между критерием верности (малой ошибкой) и экономностью. Может существовать много моделей с одинаковым числом параметров (зеленые и сиреневые точки), но *Парето-фронт* (сиреневые точки) определяется как огибающая моделей, дающих наименьшую ошибку при заданном числе параметров. Сплошная кривая аппроксимирует Парето-фронт. Решения, *оптимальные по Парето* (серая область), – это те модели, которые дают верные результаты, оставаясь при этом экономичными

Переобучение

Идею Парето необходимо доработать для применения к реальным данным. А именно при построении моделей с большим числом параметров, как часто бывает в машинном обучении, очень легко переобучить модель, слишком тесно подогнав ее к данным. Это хорошо видно по рис. 4.15 (c), где ошибка возрастает с увеличением сложности модели. Таким образом, в противоречии с тем, что изображено на рис. 4.16, где ошибка уменьшается, когда число параметров модели растет, в действительности ошибка может быть тем больше, чем больше число параметров. Чтобы найти правильную модель, применяются различные алгоритмы перекрестной проверки и выбора модели.

Для иллюстрации переобучения рассмотрим простой пример из предыдущего раздела. В нем мы пытаемся найти правильную параболическую модель для зашумленных данных (4.43). Результаты на рис. 4.15 (b) и 4.15 (c) уже указывают, что для полиномиальных моделей выше второй степени имеет место переобучение. Следующий пример в MATLAB проясняет эффекты переобучения. Программа в листинге 4.15 генерирует обучающий и тестовый наборы для параболы (4.43). Обучающий набор берется из отрезка $x \in [0, 4]$, а тестовый – из экстраполируемой области $x \in [4, 8]$.

Листинг 4.15 ❖ Обучающие и тестовые данные для параболической модели

```
n=200; L=8;
x=linspace(0,L,n);
x1=x(1:100);   % обучающий
x2=x(101:200); % тестовый
n1=length(x1);
n2=length(x2);
ftrain=(x1.^2).'; % обучить модель на отрезке [0,4]
ftest=(x2.^2).';  % проверить модель на отрезке [4,8]
figure(1), subplot(3,1,1),
plot(x1,ftrain,'r',x2,ftest,'b','Linewidth',[2])
```

Эта программа порождает идеальную модель в двух непересекающихся областях: $x \in [0, 4]$ и $x \in [4, 8]$. Если добавить к данным шум, то полином с вычисленными параметрами уже не будет точно совпадать с данными. Для зашумленных измерений мы можем вычислить как ошибку интерполяции в области $x \in [0, 4]$, так и ошибку экстраполяции в области $x \in [4, 8]$. В этом примере выполнялась регрессия методом наименьших квадратов, для чего вычислялась псевдообратная матрица командой MATLAB **pinv**.

Листинг 4.16 ❖ Переобучение квадратичной модели

```
M=30; % число параметров модели
Eni=zeros(100,M); Ene=zeros(100,M);
for jj=1:M
    for j=1:jj
        phi_i(:,j)=(x1.').^(j-1); % ключ интерполяции
        phi_e(:,j)=(x2.').^(j-1); % ключ экстраполяции
    end
```

```
    f=(x.^2).';
    for j=1:100
        fni=(x1.^2+0.1*randn(1,n1)).'; % интерполяция
        fne=(x2.^2+0.1*randn(1,n2)).'; % экстраполяция

        ani=pinv(phi_i)*fni; fnai=phi_i*ani;
        Eni(j,jj)=norm(ftrain-fnai)/norm(ftrain);

        fnae=phi_e*ani; % использовать нагрузки для [0,4]
        Ene(j,jj)=norm(ftest-fnae)/norm(ftest);
    end
end
```

В этом примере продемонстрированы некоторые из основных и наиболее характерных черт переобучения моделей. Прежде всего переобученная модель не обобщается. Рассмотрим результаты этой программы, показанные на рис. 4.17. В этом примере для нахождения нагрузок вычислялась псевдообратная матрица для данных в диапазоне $x \in [0, 4]$. Соответствующая ошибка интерполяции показана на рис. 4.17 (b) и (c). Обратите внимание на то, как проявляется переобученность полиномов. Ошибка интерполяции увеличивается, когда степень полинома превышает 2. Ошибки экстраполяции переобученной модели очень велики. На рис. 4.17 (d) и (e) показан рост ошибки как функции от степени полиномиальной модели p. На рис. 4.17 (d) использован логарифмический масштаб, потому что ошибка возрастает до 10^{13}. Это ясно указывает, что переобученная модель не обобщается на область $x \in [4, 8]$. На самом деле только экономная полиномиальная модель степени 2 легко обобщается на область $x \in [4, 8]$ без увеличения ошибки.

Этот пример показывает, что если мы хотим, чтобы модель обобщалась за пределы собранных данных, то необходим какой-то систематический способ выбора экономной модели. Машинное обучение по сути своей включает две вещи: (i) использование данных для порождения прогностических моделей и (ii) методы перекрестной проверки для устранения наиболее пагубных эффектов переобучения. Пренебрежение перекрестной проверкой почти наверняка приведет к необобщаемой модели типа изображенной на рис. 4.17. Далее мы рассмотрим некоторые стандартные стратегии построения разумных моделей.

4.6. Выбор модели: перекрестная проверка

В предыдущем разделе мы видели фундаментальную проблему регрессии, а именно тот факт, что модель легко переобучить, так что она окажется необобщаемой и непригодной для экстраполяции. Это особенно опасно при обучении глубоких нейронных сетей. Для преодоления последствий переобучения предложены различные способы выбора экономной модели с небольшим числом параметров. Это приводит к новой интерпретации Парето-фронта на рис. 4.16. Именно, ошибка резко увеличивается с ростом числа параметров из-за переобучения, особенно когда модель используется для экстраполяции.

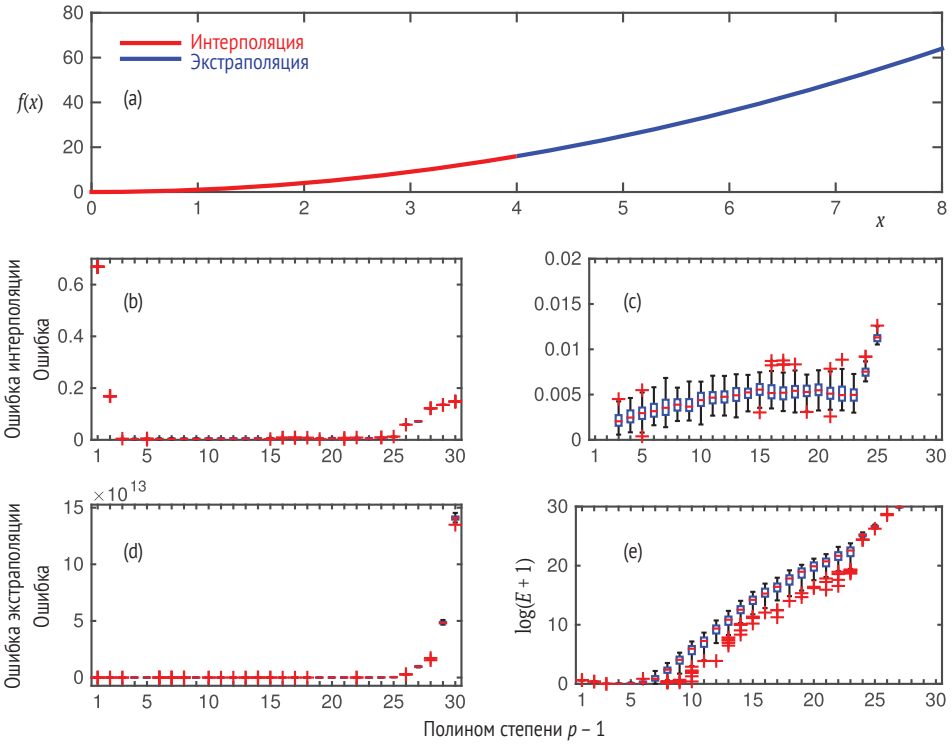

Рис. 4.17 ❖ (a) Идеальная модель $f(x) = x^2$ в области $x \in [0, 8]$. По данным, собранным в области $x \in [0, 4]$, строится полиномиальная модель регрессии (4.44) с полиномами увеличивающейся степени. В режиме интерполяции, т. е. на отрезке [0, 4], модель остается ограниченной, хотя ошибка увеличивается с ростом степени полинома. Эта ошибка показана на рисунке (b) и в увеличенном масштабе на рисунке (c). Но при экстраполяции на отрезок [4, 8] ошибка экспоненциально возрастает. На рисунке (d) видно, что ошибка увеличивается до 10^{13}. На рисунке (e) показана величина $\log(E + 1)$ в увеличенном логарифмическом масштабе (единица прибавлена, чтобы нулевая ошибка отображалась как 0), чтобы экспоненциальный рост был наглядно виден. Очевидно, что модель, обученная на отрезке [0, 4], не обобщается (не экстраполируется) на отрезок [4, 8]. Этот пример должен послужить серьезным предупреждением об опасностях, связанных с аппроксимацией моделей

Существует две общие математические стратегии, позволяющие избегать переобучения моделей: *перекрестная проверка* и вычисление *информационного критерия*. В этом разделе рассматривается первая стратегия, а в следующем – вторая. Стратегии перекрестной проверки, пожалуй, встречаются наиболее часто и применяются почти во всех алгоритмах машинного обучения. Идея проста: взять случайно отобранную часть данных и построить модель. Проделать это k раз и усреднить параметры модели (нагрузки регрессии). Проверить предсказания модели на зарезервированных данных (экстраполировать) и оценить, действительно ли модель хороша. Такая стратегия называется k-групповой перекрестной проверкой. Она проста, интуитивно понятна и порождает статистически обоснованную модель для оценки.

Для иллюстрации перекрестной проверки снова рассмотрим аппроксимацию простой функции $f(x) = x^2$ полиномиальными моделями (см. рис. 4.18). Выше мы уже детально обсуждали эту задачу как с точки зрения различных методов регрессии (псевдообращение матрицы, LASSO, робастная аппроксимация и т. д.), так и с точки зрения способности модели точно интерполировать и экстраполировать данные. В приведенной ниже программе для MATLAB демонстрируется применение трех методов регрессии (псевдообращения с помощью наименьших квадратов, оператора \, основанного на QR-разложении, и метода LASSO, поощряющего разреженность) для k-групповой перекрестной проверки ($k = 2$, 20 и 100). В этом случае k снимков данных можно рассматривать как испытания. Как и следовало ожидать, результаты тем лучше, чем больше k, и для построения окончательной модели усреднялись данные по $k = 100$ испытаниям.

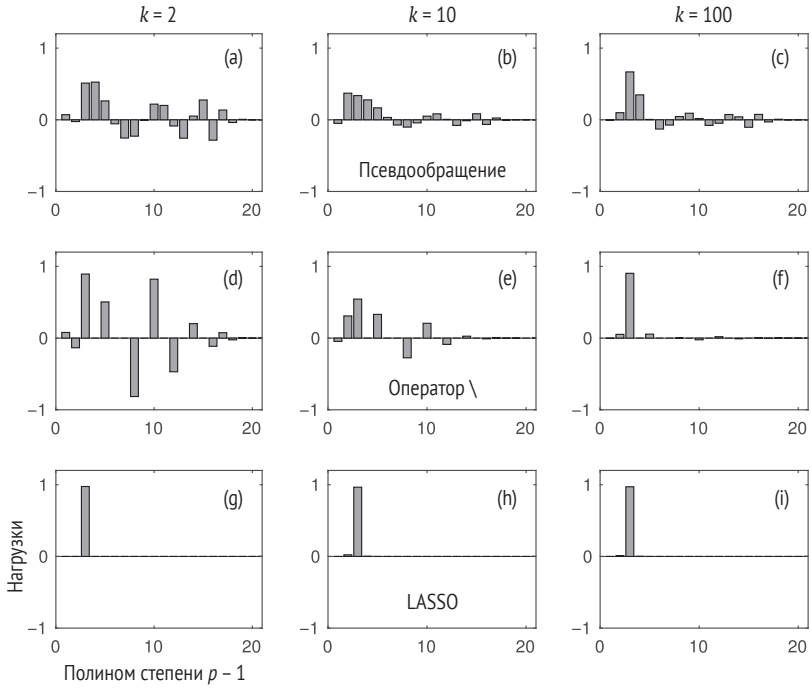

Рис. 4.18 ❖ Перекрестная проверка с использованием k-групповой стратегии при k = 2, 20 и 100 (левая, средняя и правая колонки соответственно). Перекрестной проверке подвергаются три стратегии регрессии: псевдообращение методом наименьших квадратов, оператор \, основанный на QR-разложении, и метод LASSO, поощряющий разреженность. Заметим, что в этом примере LASSO порождает квадратичную модель даже при 1- и 2-групповой проверке. У оператора \ после 100-групповой перекрестной проверки наличествует выраженный пик, а аппроксимация методом наименьших квадратов наводит на мысль, что имеют значение и квадратичные, и кубические члены – даже после 100-групповой перекрестной проверки

Листинг 4.17 ❖ *k*-групповая перекрестная проверка со 100 группами

```
n=100; L=4;
x=linspace(0,L,n);
f=(x.^2).'; % 100 точек, лежащих на параболе

M=21; % степень полинома
for j=1:M
    phi(:,j)=(x.').^(j-1); % построить матрицу A
end

trials=[2 10 100];
for j=1:3
    for jj=1:trials(j)
        f=(x.^2+0.2*randn(1,n)).';
        a1=pinv(phi)*f; f1=phi*a1; E1(jj)=norm(f-f1)/norm(f);
        a2=phi\f; f2=phi*a2; E2(jj)=norm(f-f2)/norm(f);
        [a3,stats]=lasso(phi,f,'Lambda',0.1); f3=phi*a3; E3(jj)=
            norm(f-f3)/norm(f);
        A1(:,jj)=a1; A2(:,jj)=a2; A3(:,jj)=a3;
    end
    A1m=mean(A1.'); A2m=mean(A2.'); A3m=mean(A3.');
    Err=[E1; E2; E3];

    subplot(3,3,j), bar(A1m), axis([0 21 -1 1.2])
    subplot(3,3,3+j), bar(A2m), axis([0 21 -1 1.2])
    subplot(3,3,6+j), bar(A3m), axis([0 21 -1 1.2])
end
```

На рис. 4.18 показаны результаты *k*-групповой перекрестной проверки. Благодаря поощрению разреженности (экономности) LASSO находит желаемую квадратичную модель уже при $k = 1$ (т. е. даже без перекрестной проверки). Напротив, в случае регрессии методом наименьших квадратов (псевдообращения) и регрессии на основе QR-разложения требуется много проходов, чтобы выделить доминирующий квадратичный член. Регрессия методом наименьших квадратов даже при $k = 100$ группах все еще включает и квадратичный, и кубический член.

На последнем этапе выбора модели с помощью *k*-групповой перекрестной проверки часто производится *пороговая фильтрация* малых членов. Приведенная выше программа демонстрирует три стратегии регрессии. LASSO выглядит почти идеально, но все-таки остается небольшая линейная компонента. Стратегия QR-разложения порождает несколько малых компонент, соответствующих разным степеням. Регрессия методом наименьших квадратов выделяет доминирующие квадратичный и кубический члены и содержит еще много ненулевых коэффициентов. Если подвергнуть коэффициенты пороговой фильтрации, то LASSO и оператор \ дадут в точности квадратичную модель, а метод наименьших квадратов – квадратично-кубическую модель. Результирующую модель можно затем оценить в областях интерполяции и экстраполяции, как на рис. 4.19.

Листинг 4.18 ❖ Сравнение моделей, подвергнутых перекрестной проверке

```
Atot=[A1m; A2m; A3m];   % средние нагрузки для трех методов
Atot2=(Atot>0.2).*Atot; % порог
Atot3=[Atot; Atot2];    % нагрузки до и после фильтрации

figure(3), bar3(Atot.')
figure(4), bar3(Atot2.')

n=200; L=8;
x=linspace(0,L,n);
x1=x(1:100);   % обучение (интерполяция)
x2=x(101:200); % проверка (экстраполяция)

ftrain=(x1.^2).'; % интерполированная парабола на отрезке [0,4]
ftest=(x2.^2).';  % экстраполированная парабола на отрезке [4,8]

for j=1:M
    phi_i(:,j)=(x1.').^(j-1); % ключ интерполяции
    phi_e(:,j)=(x2.').^(j-1); % ключ экстраполяции
end

for jj=1:6 % вычислить ошибки интерполяции и экстраполяции
    ani=Atot3(jj,:).';
    fnai=phi_i*ani;
    Eni(jj)=norm(ftrain-fnai)/norm(ftrain);
    fnae=phi_e*ani;
    Ene(jj)=norm(ftest-fnae)/norm(ftest);
end
```

Результаты на рис. 4.19 показывают, что качество модели очень сильно зависит от процесса выбора и от применяемого метода регрессии. Кроме того, видно, что, несмотря на *k*-групповую перекрестную проверку, ошибка экстраполяции, или обобщаемость модели, может оказаться никуда не годной. Хорошей считается модель, которая дает небольшую ошибку и при этом хорошо обобщается, как метод LASSO в нашем примере.

k-групповая перекрестная проверка

Процесс *k*-групповой перекрестной проверки иллюстрируется на рис. 4.20. Идея в том, чтобы разбить имеющийся набор данных на два: обучающий и тестовый. Тестовый, или зарезервированный, набор не участвует в процедуре обучения модели. Именно на тестовом наборе проверяется способность модели к экстраполяции, что, как показывают рисунки в двух предыдущих разделах, является нетривиальной задачей. При *k*-групповой перекрестной проверке обучающие данные случайным образом разбиваются на k групп одинакового размера. Например, при стандартной 10-групповой перекрестной проверке обучающие данные разбиваются на 10 групп. На основе каждой группы строится модель регрессии $\mathbf{Y}_j = f(\mathbf{X}_j, \boldsymbol{\beta}_j)$, $j = 1, 2, \cdots, 10$. Окончательная модель строится путем усреднения нагрузок $\bar{\beta} = (1/k)\sum_{j=1}^{k} \beta_j$ и подстановки этого значения в модель регрессии $\mathbf{Y} = f(\mathbf{X}, \bar{\boldsymbol{\beta}})$. Затем на зарезервирован-

ных данных проверяется способность этой модели к экстраполяции, т. е. ее обобщаемость. Величина ошибки на тестовом наборе и определяет качество модели. Существуют и другие методы выбора модели, например просто выбор лучшей из k моделей, построенных в ходе перекрестной проверки. Что касается разбиения исходного набора данных, то часто направляют 70 % на обучение, 20 % – на контроль в процессе обучения и 10 % – на тестирование обученной модели. Если набор данных очень велик, то контрольный и тестовый наборы можно уменьшить, при условии что данных все же будет достаточно для оценки построенной модели.

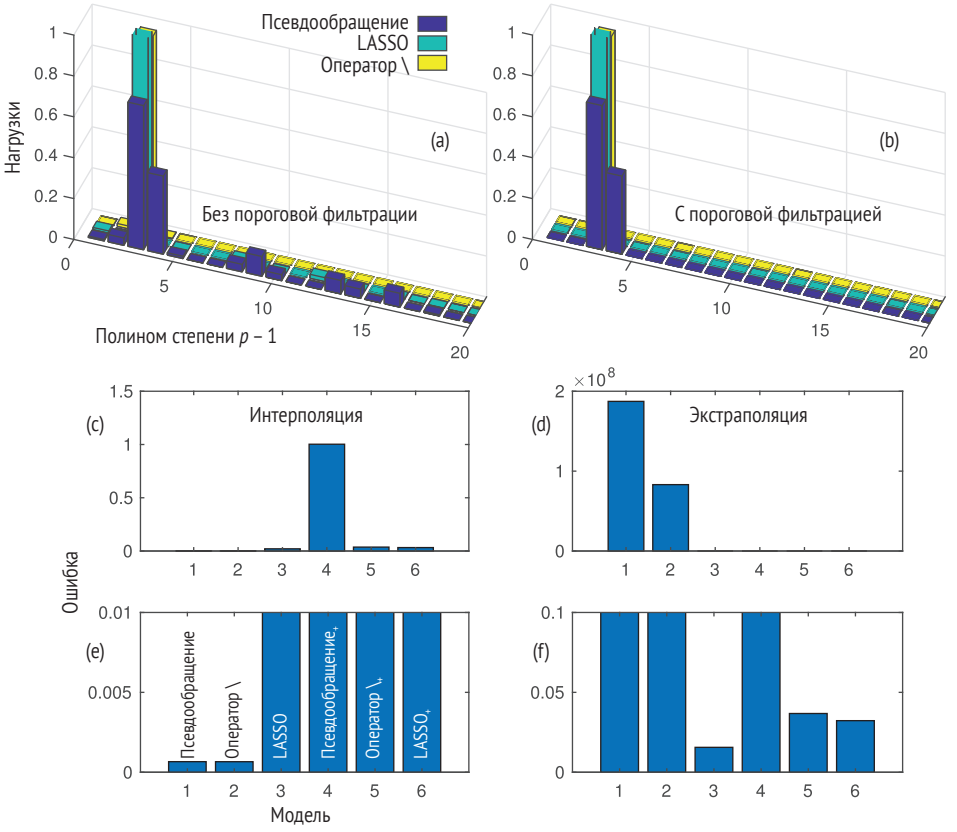

Рис. 4.19 ❖ Ошибка и нагрузки для 100-групповой перекрестной проверки. Нагрузки (с пороговой фильтрацией на рисунке (b) и без нее на рисунке (a)) показаны для аппроксимации методом наименьших квадратов посредством псевдообращения матрицы, для QR-разложения и для метода LASSO, поощряющего разреженность (см. рис. 4.18). На рисунке (c) (и более подробно на рисунке (e)) показана ошибка интерполяции, а на рисунке (d) (и более подробно на рисунке (f)) – ошибка экстраполяции. Нижний индекс + обозначает результаты с пороговой фильтрацией. LASSO хорошо работает и для интерполяции, и для экстраполяции, тогда как метод наименьших квадратов экстраполируется плохо. Рассмотрено шесть моделей: 1) псевдообращение; 2) оператор \; 3) LASSO; 4) псевдообращение с пороговой фильтрацией; 5) оператор \ с пороговой фильтрацией; 6) LASSO с пороговой фильтрацией

Перекрестная проверка с контролем по *p* точкам

Еще один стандартный метод – *перекрестная проверка с контролем по p точкам* (leave p-out cross validation – LpO CV). В этом случае из обучающих данных исключаются случайно выбранные p точек, которые играют роль контрольного набора. Модель строится по оставшимся обучающим данным, а затем тестируется на отложенной выборке. Затем процесс повторяется с другой выборкой объема p – и так до тех пор, пока все обучающие данные не побывают элементами контрольного набора. Затем модели, полученные при всех разбиениях, усредняются, и верность окончательной модели проверяется на тестовом наборе.

4.7. Выбор модели: информационный критерий

Этот подход к выбору модели отличается от описанных в предыдущем разделе стратегий перекрестной проверки. В начале 1950-х годов под выбор модели был подведен строгий математический фундамент. Расхождение Кульбака–Лейблера (КЛ) [314] измеряет расстояние между двумя плотностями распределения вероятности (или наборами данных, представляющими истинное положение вещей и модель). Расхождение КЛ имеет глубокие математические связи со статистическими методами, характеризующими энтропию, которые были разработаны Людвигом Э. Больцманом (1844–1906), а также с теорией информации, разработанной Клодом Шенноном [486]. Выбор модели – развитая дисциплина с обширной литературой, большая часть которой описана в блестящем обзоре Burnham and Anderson [105]. Ниже мы лишь вкратце осветим некоторые стандартные методы.

Расхождение КЛ между двумя моделями $f(\mathbf{X}, \boldsymbol{\beta})$ и $g(\mathbf{X}, \boldsymbol{\mu})$ определяется по формуле

$$I(f,g) = \int f(\mathbf{X}, \boldsymbol{\beta}) \log\left[\frac{f(\mathbf{X}, \boldsymbol{\beta})}{g(\mathbf{X}, \boldsymbol{\mu})}\right] d\mathbf{X}, \tag{4.45}$$

где $\boldsymbol{\beta}$ и $\boldsymbol{\mu}$ – векторы параметров моделей $f(\cdot)$ и $g(\cdot)$ соответственно. С точки зрения теории информации, величина $I(f, g)$ измеряет потерю информации в результате использования g для представления f. Заметим, что если $f = g$, то логарифм обращается в нуль ($\log(1) = 0$) и $I(f, g) = 0$, поэтому никакой потери информации нет. На практике f представляет *истину*, т. е. экспериментальные измерения, а g – модель, предложенную для описания f.

В отличие от регрессии и перекрестной проверки, при вычислении расхождения КЛ модель необходимо задать. Напомним, что ранее мы использовали перекрестную проверку, чтобы сгенерировать модель с применением различных стратегий регрессии (см., например, рис. 4.20). Здесь выдвигается несколько гипотез, и для каждой гипотетической модели $g_j(\mathbf{X}, \boldsymbol{\mu}_j)$, $j = 1, 2, \cdots$,

M, вычисляется расхождение КЛ $I_j(f, g_j)$. Модель, для которой потеря информации наименьшая, признается лучшей.

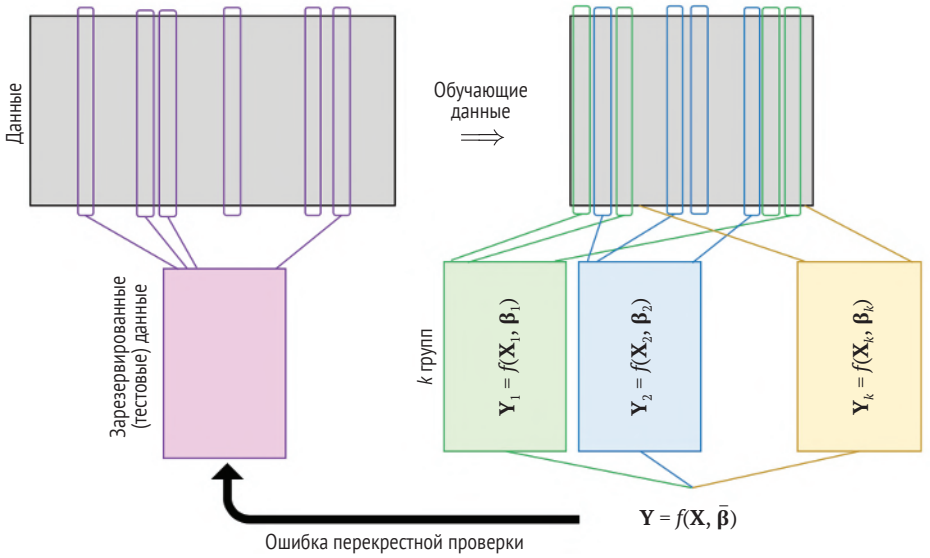

Рис. 4.20 ❖ Процедура k-групповой перекрестной проверки моделей. Вначале данные разбиваются на два набора: обучающий и тестовый (зарезервированный). Обычно в тестовый набор включается случайная выборка из всего набора данных. Обучающие данные случайным образом разбиваются на k групп, и для каждой строится модель регрессии $\mathbf{Y}_j = f(\mathbf{X}_j, \boldsymbol{\beta}_j)$. Важно, что для каждой модели генерируются свои параметры (нагрузки) $\boldsymbol{\beta}_j$. После обучения k моделей порождается наилучшая модель $\mathbf{Y} = f(\mathbf{X}, \bar{\boldsymbol{\beta}})$. Есть разные способы ее построения, один из них – усреднение параметров модели, т. е. $\bar{\beta} = (1/k)\sum_{j=1}^{k} \beta_j$. Или же можно просто выбрать модель с лучшими параметрами из k обученных. В любом случае качество окончательной модели проверяется на тестовом наборе

В качестве простого примера рассмотрим рис. 4.21, на котором три разные модели сравниваются с истинными данными. Этот рисунок был сгенерирован программой в листинге 4.19. Показаны также результаты вычисления расхождения КЛ. Чтобы избежать деления на нуль, к каждому распределению вероятностей прибавлено постоянное смещение. Истинные данные $f(x)$ представляют собой простую нормально распределенную случайную величину. Три модели – варианты нормально и равномерно распределенных величин.

Листинг 4.19 ❖ Вычисление расхождения КЛ

```
n=10000;
x1=randn(n,1); % "истинная" модель (данные)
x2=0.8*randn(n,1)+1; % модель 1
x3=0.5*randn(n,1)-1; % компоненты модели 3
```

```
x4=0.7*randn(n,1)-3;
x5=5*rand(n,1)-0.5;
x=-6:0.01:6; % диапазон данных

f=hist(x1,x)+0.01; % сгенерировать ФПВ
g1=hist(x2,x)+0.01;
g2a=hist(x3,x); g2b=hist(x4,x); g2=g2a+0.3*g2b+0.01;
g3=hist(x5,x)+0.01;

f=f/trapz(x,f); % нормировать данные
g1=g1/trapz(x,g1); g2=g2/trapz(x,g2); g3=g3/trapz(x,g3);
plot(x,f,x,g1,x,g2,x,g3,'Linewidth',[2])

% вычислить подынтегральное выражение
Int1=f.*log(f./g1); Int2=f.*log(f./g2); Int3=f.*log(f./g3);

% использовать при необходимости
%Int1(isinf(Int1))=0; Int1(isnan(Int1))=0;
%Int2(isinf(Int2))=0; Int2(isnan(Int2))=0;

% расхождение КЛ
I1=trapz(x,Int1); I2=trapz(x,Int2); I3=trapz(x,Int3);
```

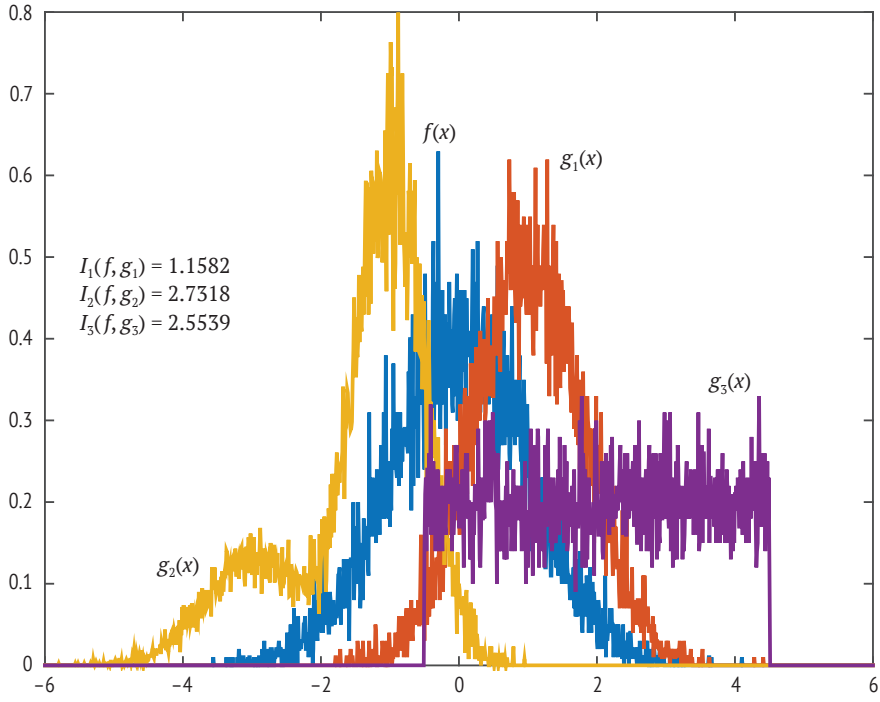

Рис. 4.21 ❖ Сравнение трех моделей $g_1(x)$, $g_2(x)$ и $g_3(x)$ с истинной функцией $f(x)$. Вычисляется расхождение КЛ $I_j(f, g_j)$ для каждой модели. Выясняется, что модель $g_1(x)$ статистически ближе всего к истинным данным

Информационные критерии: AIC и BIC

Этот простой пример демонстрирует базовую идею, стоящую за выбором модели: вычислить расстояние между предложенной моделью $g_j(x)$ и истинными измерениями $f(x)$. В начале 1970-х годов Хиротугу Акаике объединил вычисление максимального правдоподобия Фишера [183] с расхождением Кульбака–Лейблера и получил то, что теперь называется *информационным критерием Акаике* (Akaike Information Criterion – AIC) [7]. Впоследствии его модифицировал Гидеон Шварц, в результате чего получился *байесовский информационный критерий* (Bayesian Information Criterion – BIC) [480], который дает информационную оценку, гарантированно сходящуюся к правильной модели при наличии достаточно большого объема данных, если только правильная модель была включена в состав моделей-кандидатов.

Мы рассмотрим первопроходческую работу Акаике [7]. Акаике знал, что расхождение КЛ невозможно вычислить на практике, потому что для этого требуется знание полной статистики истинной модели $f(x)$ и всех параметров предлагаемых моделей $g_j(x)$. Поэтому он предложил альтернативный способ оценки расхождения КЛ, основанный на максимальном значении эмпирической функции логарифмического правдоподобия. Оно допускает практическое вычисление, и это стало краеугольным камнем строгих методов выбора модели. Технические аспекты работы Акаике, связавшей оценки логарифмического правдоподобия с расхождением КЛ [7, 105], стали математическим достижением, ознаменовавшим смену парадигмы. Получившийся в результате AIC имеет вид:

$$\mathrm{AIC} = 2K - 2\log[\mathcal{L}(\hat{\boldsymbol{\mu}}|\mathbf{x})], \tag{4.46}$$

где K – число параметров модели, $\hat{\boldsymbol{\mu}}$ – оценка лучших параметров модели $g(\mathbf{X}, \boldsymbol{\mu})$ (для которой расхождение КЛ минимально), вычисленное по *оценке максимального правдоподобия* (maximum likelihood estimate – MLE), а $\mathbf{x}$ – независимая выборка из подлежащих аппроксимации данных. Таким образом, вместо прямого вычисления расстояния между двумя моделями AIC предлагает оценку относительного расстояния между аппроксимирующей моделью и истинными данными. При возрастании числа параметров предложенной модели AIC растет с угловым коэффициентом $2K$ и, следовательно, штрафует неэкономные модели. Важно, что, будучи относительной мерой, AIC всегда находит объективно «лучшую» модель с наименьшей оценкой, что не мешает ей быть никуда не годной с точки зрения предсказания и реконструкции данных.

На сегодня AIC является одним из стандартных критериев выбора модели. Но есть и другие. Выше уже упоминался критерий BIC – модификация AIC, предложенная Гидеоном Шварцем [480]. BIC отличается от AIC только штрафом за количество параметров. Точнее, BIC определяется формулой

$$\mathrm{BIC} = \log(n)K - 2\log[\mathcal{L}(\hat{\boldsymbol{\mu}}|\mathbf{x})], \tag{4.47}$$

где n – количество точек данных, или размер выборки. У этого немного отличающегося варианта информационного критерия есть одно важное свойство. Заслуга Шварца – доказательство того, что если правильная модель включена

в состав моделей-кандидатов, то на основе критерия BIC она гарантированно будет признана наилучшей, если набор данных достаточно велик. Этим он отличается от критерия AIC, который в некоторых патологических случаях может выбирать неправильную модель.

Вычисление AIC и BIC

MATLAB позволяет непосредственно вычислить критерии AIC и BIC командой **aicbic**. Это средство включено в комплект эконометрических инструментов и дает возможность сравнивать между собой различные модели. Сравнение производится на основе оценки максимального правдоподобия модели. Число сравниваемых моделей произвольно.

Рассмотрим конкретный пример – истинные данные построены на основе авторегрессионной модели

$$x_n = -4 + 0.2x_{n-1} + 0.5x_{n-2} + \mathcal{N}(0, 2), \tag{4.48}$$

где x_n – значение временного ряда в момент t_n, а $\mathcal{N}(0, 2)$ – белый шум с нулевым средним и дисперсией 2. Мы попытаемся аппроксимировать данные тремя авторегрессионными интегрированными моделями скользящего среднего (ARIMA) порядка 1, 2 и 3. Показанная ниже программа вычисляет их логарифмическое правдоподобие и соответствующие критерии AIC и BIC.

Листинг 4.20 ❖ Вычисление AIC и BIC

```
T = 100; % объем выборки
DGP = arima('Constant',-4,'AR',[0.2, 0.5],'Variance',2);
y = simulate(DGP,T);

EstMdl1 = arima('ARLags',1);
EstMdl2 = arima('ARLags',1:2);
EstMdl3 = arima('ARLags',1:3);

logL = zeros(3,1); % выделить память для вектора правдоподобий
[~,~,logL(1)] = estimate(EstMdl1,y,'print',false);
[~,~,logL(2)] = estimate(EstMdl2,y,'print',false);
[~,~,logL(3)] = estimate(EstMdl3,y,'print',false);
[aic,bic] = aicbic(logL, [3; 4; 5], T*ones(3,1))
```

Заметим, что лучшей моделью, для которой AIC и BIC минимальны, является вторая модель порядка 2. Этого следовало ожидать, т. к. вторая модель соответствует истинным данным. Печатается следующий результат:

```
aic =
  381.7732
  358.2422
  358.8479
bic =
  389.5887
  368.6629
  371.8737
```

Наименьшие значения AIC и BIC равны соответственно 358.2422 и 368.6629. Заметим, что хотя была выбрана правильная модель, AIC показывает лишь небольшое различие между моделями второго и третьего порядка.

Рекомендуемая литература

Учебники

(1) Model selection and multimodel inference, by K. P. Burnham and D. R. Anderson [105].
(2) Multivariate analysis, by R. A. Johnson and D. Wichern, 2002 [266].
(3) An introduction to statistical learning, by G. James, D. Witten, T. Hastie and R. Tibshirani, 2013 [264].

Статьи и обзоры

(1) On the mathematical foundations of theoretical statistics, by R. A. Fischer, *Philosophical Transactions of the Royal Society of London*, 1922 [183].
(2) A new look at the statistical model identification, by H. Akaike, *IEEE Transactions on Automatic Control*, 1974 [7].
(3) Estimating the dimension of a model, by G. Schwarz et al., *The annals of statistics*, 1978 [480].
(4) On information and sufficiency, by S. Kullback and R. A. Leibler, *The annals of statistics*, 1951 [314].
(5) A mathematical theory of communication, by C. Shannon, *ACM SIGMOBILE Mobile Computing and Communications Review*, 2001 [480].

Глава 5

Кластеризация и классификация

Машинное обучение основано на методах оптимизации. Цель состоит в том, чтобы найти низкоранговое подпространство для оптимального погружения данных, а также методы регрессии для кластеризации и классификации данных по типам. Поэтому машинное обучение предоставляет набор строгих математических методов для выделения осмысленных признаков из данных, т. е. для добычи данных (data mining), а также для распределения данных по классам на основе обнаруженных паттернов, с тем чтобы затем использовать их при принятии решений. То есть модель обучается на данных и потом делает предсказания. В бизнес-приложениях это часто называется *прогнозной аналитикой* и является передним краем современной науки о принятии решений на основе данных. В интегрированной системе, каковой является, например, автономный робот, различные компоненты машинного обучения (в т. ч. обработка зрительных и осязательных раздражителей) объединены и образуют то, что теперь называется *искусственным интеллектом* (ИИ). Уточним: ИИ основан на интегрированных алгоритмах машинного обучения, которые, в свою очередь, неразрывно связаны с оптимизацией.

Существуют две широкие категории машинного обучения: *с учителем* и *без учителя*. В первом случае алгоритму предъявляется помеченный набор данных. Обучающие данные, о которых шла речь при описании перекрестной проверки в предыдущей главе, помечаются учителем (экспертом). Таким образом, примеры входов и выходов искомой модели задаются явно, а для поиска наилучшей модели, соответствующей помеченным данным, используется регрессия, основанная на оптимизации. Затем построенная модель используется для предсказания и классификации новых данных. Существуют важные разновидности методов обучения с учителем, в т. ч. *обучение с частичным привлечением учителя* (semisupervised learning), когда обучающие данные неполны, т. е. для некоторых входов отсутствуют выходы. Еще один подкласс обучения с учителем – *активное обучение*, когда алгоритм может получить обучающие метки только для подмножества примеров, ограниченного бюджетом, и должен оптимизировать выбор объектов, для которых будет получать метки. В интерактивной системе эти примеры могут быть предъявлены пользователю для пометки. Наконец, при *обучении с подкреплением*

вознаграждение или наказание играют роль обучающих меток, которые позволяют организовать архитектуру регрессии для построения оптимальной модели. С другой стороны, в алгоритмах обучения без учителя никаких меток нет вообще. Они должны самостоятельно искать паттерны и определять, каким образом кластеризовать данные и генерировать метки для предсказания и классификации новых данных. Цель обучения без учителя – найти паттерны, образующие низкоранговые подпространства, чтобы можно было воспользоваться *конструированием признаков* (feature engineering), или *выделением признаков* (feature extraction), для построения подходящей модели.

В этой главе мы рассмотрим некоторые из наиболее распространенных методов машинного обучения с учителем и без учителя. Мы увидим, как добыча данных позволяет выделить важные признаки для последующего построения модели. Мы покажем, что методы машинного обучения можно использовать для кластеризации и классификации, а также для построения моделей регрессии для предсказания. Принципиально важной чертой любой архитектуры машинного обучения является нахождение низкоранговых пространств признаков, которые были бы информативны и допускали интерпретацию.

5.1. Выделение признаков и добыча данных

Чтобы данные можно было использовать для диагностики, предсказания и управления, необходимо выделить в них доминирующие признаки. В первой главе мы познакомились с методами SVD и PCA, позволяющими выявлять доминирующие коррелированные структуры в наборе данных. Так, в примере с собственными лицами из раздела 1.6 доминирующие признаки были выделены из большого набора кадрированных изображений лиц. Эти собственные лица, упорядоченные по способности распознавать схожесть (корреляцию) с лицами в базе данных, гарантированно образуют наилучший набор r признаков для реконструкции лица в смысле нормы ℓ_2. Моды, определяемые собственными лицами, представляют собой ясные и допускающие интерпретацию признаки для идентификации лиц, включающие (вполне ожидаемо) области глаз, носа и рта. Важно, что пространство признаков позволяет заменить работу с пространством измерений высокой размерности рассмотрением значительно меньшего подпространства, по которому можно производить диагностику.

Цель добычи данных и машинного обучения – построить и использовать низкоранговое пространство признаков, внутренне присущих набору данных. Это пространство признаков можно найти алгоритмически в процессе обучения без учителя или построить явно с привлечением знаний и эксперта, и (или) имеющихся корреляций между данными. В случае собственных лиц признаками являются моды PCA, сгенерированные с помощью SVD. Таким образом, размерность каждой моды PCA высока, но единственное, что важно в пространстве признаков, – вес конкретной моды в представлении данного лица. Если произвести усечение ранга r, то любое лицо будет представле-

но в пространстве признаков только *r* признаками. В итоге мы получаем низкоранговое погружение данных в допускающий интерпретацию набор *r* признаков, который можно использовать для диагностики, предсказания, реконструкции или управления.

Мы приведем несколько примеров, иллюстрирующих построение пространства признаков, и начнем со стандартного набора данных, включенного в MATLAB. Набор данных Фишера об ирисах включает результаты обмера 150 ирисов трех видов: щетинистый (setosa), разноцветный (versicolor) и виргинский (virginica). Для 50 образцов каждого вида имеются следующие результаты измерений (в сантиметрах): длина чашелистика, ширина чашелистика, длина лепестка и ширина лепестка. Для этого набора данных уже определены четыре признака в терминах интерпретируемых ботанических свойств. Для простоты визуализации на рис. 5.1 показаны только первые три из этих признаков. Следующая программа обращается к набору данных Фишера.

Листинг 5.1 ❖ Признаки ирисов Фишера

```
load fisheriris;
x1=meas(1:50,:);    % щетинистый
x2=meas(51:100,:);  % разноцветный
x3=meas(101:150,:); % виргинский
plot3(x1(:,1),x1(:,2),x1(:,4),'go'), hold on
plot3(x2(:,1),x2(:,2),x2(:,4),'mo')
plot3(x3(:,1),x3(:,2),x3(:,4),'ro')
```

На рис. 5.1 показано, что измеренные свойства можно использовать в качестве хорошего набора признаков для кластеризации и классификации. Три разновидности ирисов разделены в пространстве признаков. Особенно четко выделяется ирис щетинистый, тогда как между разноцветным и виргинским ирисом есть небольшое перекрытие. Чтобы разработать приемлемую схему классификации для этого набора данных, машинное обучение, очевидно, не нужно. Однако в общем случае данные не сводятся так легко к двух- или трехмерным визуальным характеристикам. Решения о кластеризации приходится принимать в пространстве признаков гораздо более высокой размерности, поэтому необходимо прибегать к вычислительным методам.

В качестве второго примера на рис. 5.2 показана выборка из базы данных изображений собак и кошек, содержащей по 80 изображений тех и других. Цель этого набора данных – разработать метод автоматизированной классификации, с помощью которого компьютер мог бы отличить собаку от кошки. В этом случае каждый пример представляет собой изображение в пространстве пикселей 64×64. То есть каждое изображение имеет 4096 измерений, в отличие от 4 измерений в наборе данных об ирисах. Как и в случае собственных лиц, мы будем использовать SVD для выделения доминирующих корреляций между изображениями. Программа в листинге 5.2 загружает данные, вычитает среднее и производит сингулярное разложение. SVD порождает упорядоченный набор мод, характеризующий корреляцию между всеми изображениями собак и кошек. На рис. 5.3 показаны первые четыре SVD-моды всех 160 изображений (80 собак и 80 кошек).

Рис. 5.1 ❖ Набор данных Фишера об ирисах, содержащий результаты 150 обмеров трех видов ириса (щетинистый, разноцветный и виргинский) по 50 измерений для каждого. Для каждого цветка измерялись длина чашелистика, ширина чашелистика, длина лепестка и ширина лепестка. Показаны первые три признака, и, как видно, этих простых ботанических характеристик достаточно для количественного разделения видов

Листинг 5.2 ❖ Признаки собак и кошек

```
load dogData.mat
load catData.mat
CD=double([dog cat]);
[u,s,v]=svd(CD-mean(CD(:)),'econ');
```

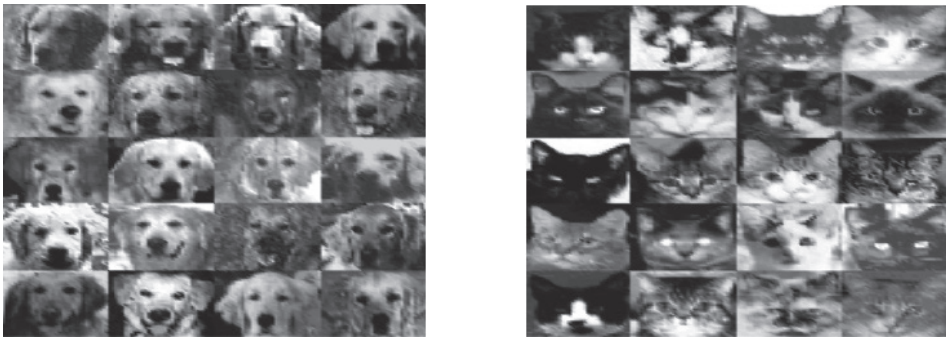

Рис. 5.2 ❖ Примеры изображений собак (слева) и кошек (справа). Наша цель – построить пространство признаков, по которому можно было бы эффективно производить автоматическую классификацию этих изображений

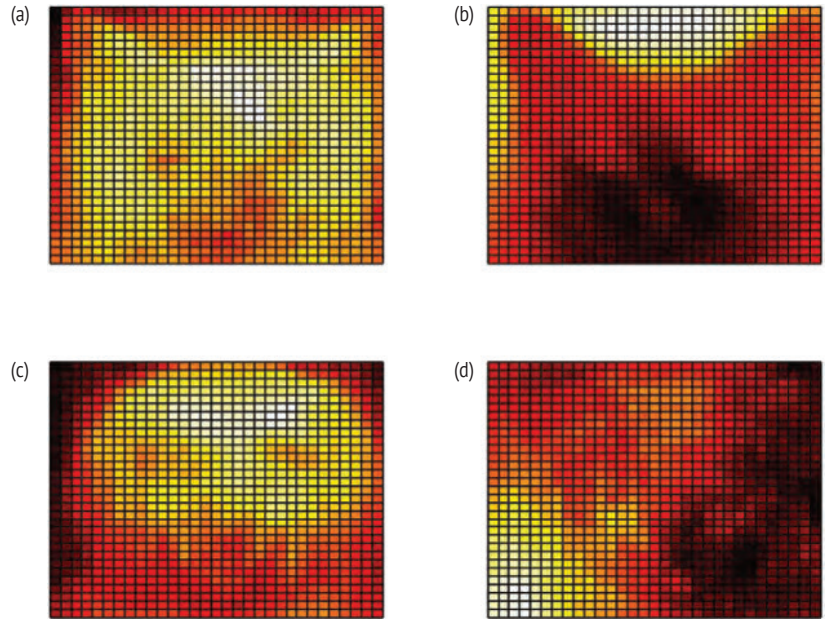

Рис. 5.3 ❖ Первые четыре признака, сгенерированных на основе сингулярного разложения 160 изображений собак и кошек на рис. 5.2. Заметим, что моды (a) и (b) показывают, что треугольные уши – важный признак. Он, безусловно, отличает кошек, а для собак выражен гораздо слабее. Таким образом, в пространстве признаков для кошек этот доминирующий признак обычно присутствует, а для собак отсутствует

Исходное пространство изображений, т. е. пространство пикселей, – лишь одно из потенциально возможных представлений набора данных. Данные можно преобразовать в вейвлет-представление, где выделены присутствующие в изображениях границы. Следующая программа загружает вейвлет-представление изображений и вычисляет другое низкоранговое подпространство.

Листинг 5.3 ❖ Вейвлетные признаки собак и кошек

```
load catData_w.mat
load dogData_w.mat
CD2=[dog_wave cat_wave];
[u2,s2,v2]=svd(CD2-mean(CD2(:)),'econ');
```

Эквивалентом рис. 5.3 в вейвлетном пространстве является рис. 5.4. Отметим, что вейвлет-представление выделяет такие ключевые признаки, как глаза, нос и уши, что может помочь в принятии решений о классификации. Порождение пространства признаков, облегчающего классификацию, критически важно для построения эффективных алгоритмов машинного обучения.

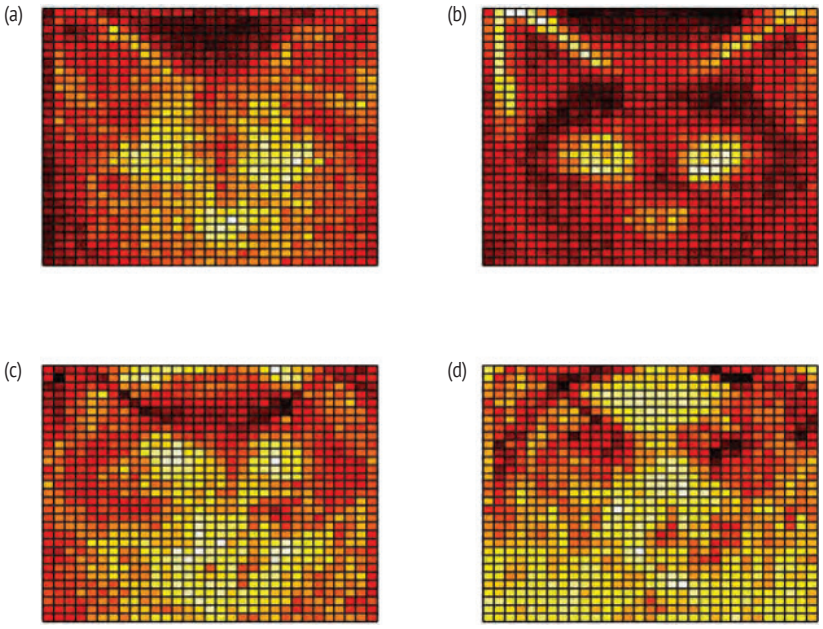

Рис. 5.4 ❖ Первые четыре признака, сгенерированных на основе сингулярного разложения 160 изображений собак и кошек в вейвлет-представлении. Как и раньше, первые две моды (a) и (b) показывают важность треугольных ушей. Это альтернативное представление собак и кошек, которое, возможно, дает лучшую классификацию

Какое бы пространство признаков – из исходных изображений или их вейвлет-представлений – ни использовалось, наша цель состоит в том, чтобы спроецировать данные на это пространство. Хорошее пространство признаков помогает находить отличительные признаки, благодаря чему можно решать самые разные задачи, в т. ч. кластеризации, классификации и предсказания. Важность каждого признака применительно к конкретному изображению дает матрица **V** в сингулярном разложении. Точнее, каждый столбец **V** определяет нагрузку, или вес признака, в данном изображении. Гистограммы этих нагрузок можно затем использовать для визуализации того, в какой мере собаки и кошки различимы по каждому признаку (см. рис. 5.5). Следующая программа строит гистограмму распределения нагрузок для собак и кошек.

Листинг 5.4 ❖ Гистограмма признаков собак и кошек

```
xbin=linspace(-0.25,0.25,20);
for j=1:4
    subplot(4,2,2*j-1)
    pdf1=hist(v(1:80,j),xbin)
    pdf2=hist(v(81:160,j),xbin)
    plot(xbin,pdf1,xbin,pdf2,'Linewidth',[2])
end
```

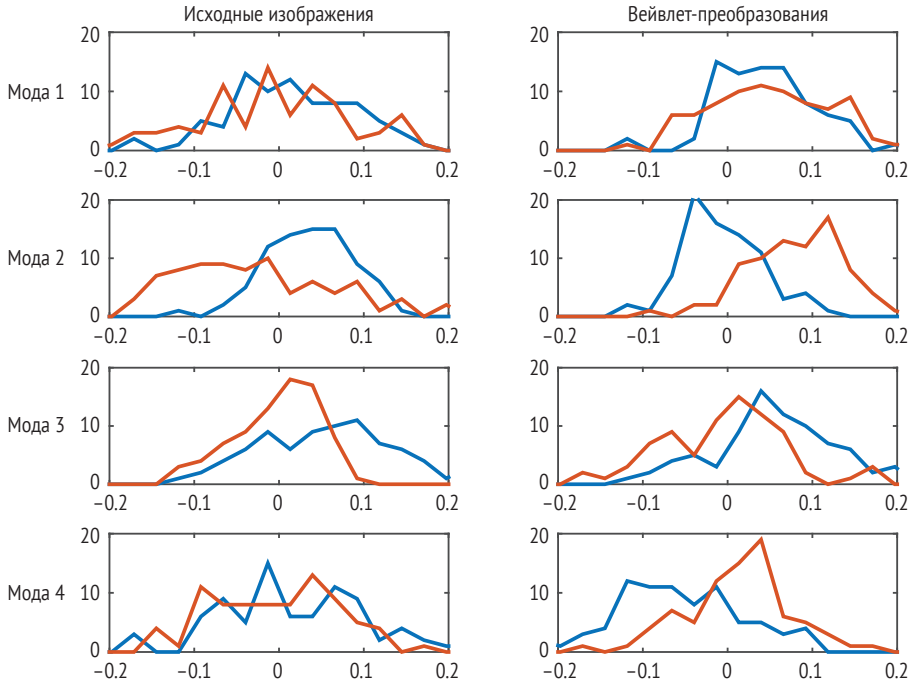

Рис. 5.5 ❖ Гистограмма распределения нагрузок для собак (синяя линия) и кошек (красная линия) по первым четырем доминирующим SVD-модам. На левом рисунке показаны распределения для исходных изображений (см. рис. 5.3), а на правом – для вейвлет-преобразований (см. рис. 5.4). Нагрузки берутся из столбца матрицы **V** сингулярного разложения. Отметим, что собаки и кошки хорошо разделяются по второй моде

На рис. 5.5 показано распределение нагрузок для первых четырех мод как для исходных изображений, так и для их вейвлет-преобразований. В обоих случаях видно, что собаки и кошки отчетливо различаются по второй моде. Вейвлет-преобразования изображения также разделяются по четвертой моде. Заметим, что первая мода практически не позволяет различить ни исходные изображения, ни их вейвлет-преобразования, поэтому она не особенно полезна для классификации и кластеризации.

Признаки, допускающие четкое разделение между данными разных типов (например, изображениями собак и кошек), как правило, используются для решения задач машинного обучения. Этот простой пример показывает, что конструирование признаков – процесс начального исследования данных, позволяющий определить потенциально перспективные методы предобработки. Затем компьютер может использовать признаки, обладающие высокой различительной способностью, в пространстве более высокой размерности для уверенной кластеризации, классификации и предсказания. И в заключение рассмотрим рис. 5.6, на котором показаны проекции данных о собаках и кошках на первые три моды PCA (SVD-моды), найденные в результате анализа исходных изображений и их вейвлет-преобразований. Как мы увидим

ниже, вейвлет-преобразования обеспечивают более высокую степень разделимости, а значит, улучшенную классификацию.

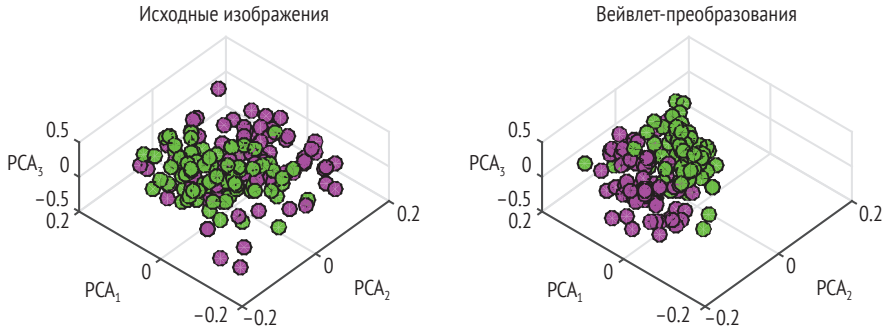

Рис. 5.6 ❖ Проекции собак (зеленые точки) и кошек (сиреневые точки) на пространство признаков. Заметим, что исходные изображения и их вейвлет-преобразования порождают разные погружения данных. В обоих случаях имеет место кластеризация, и это используется в алгоритмах обучения, которые будут описаны ниже. Вейвлет-преобразования особенно хороши для кластеризации и классификации, поскольку в этом пространстве признаков данные лучше разделены

5.2. Обучение с учителем и без учителя

Как уже отмечалось, цель добычи данных и машинного обучения – построить и использовать низкоранговое пространство признаков, внутренне присущее имеющемуся набору данных. Хорошие алгоритмы конструирования и выделения признаков можно затем использовать для обучения классификаторов и предикторов. Существует две основные парадигмы обучения на данных: *с учителем* и *без учителя*. Методу обучения с учителем предъявляется обучающий набор данных, помеченный экспертом. Это значит, что алгоритму подаются вход и выход искомой модели, для нахождения наилучшей модели применяются методы регрессии, выполняющие оптимизацию на основе помеченных данных. Затем эта модель используется для предсказания и классификации на новых данных. У этой базовой архитектуры есть варианты: обучение с частичным привлечением учителя, активное обучение и обучение с подкреплением. Алгоритму обучения без учителя не предъявляется помеченных данных, он должен самостоятельно выявить в данных паттерны и решить, как будет кластеризовать и классифицировать новые данные. Цель обучения без учителя – выявить паттерны, образующие низкоранговые подпространства, чтобы можно было воспользоваться конструированием или выделением признаков для построения подходящей модели.

Для иллюстрации различий между обучением с учителем и без учителя рассмотрим рис. 5.7. На нем показана диаграмма рассеяния двух гауссовых распределений. В одном случае данные четко разделены, т. е. их средние находятся достаточно далеко друг от друга и наблюдаются два непересе-

кающихся кластера. В другом случае два распределения близки, поэтому разделить данные – трудная задача. Цель обучения без учителя – выявить кластеры в данных. Эта задача тривиально решается визуальной инспекцией, при условии что распределения в достаточной степени разделены. Иначе выделить кластеры очень трудно. В случае обучения с учителем все данные или их часть снабжены метками. Можно сказать, что точки либо зеленые, либо сиреневые, а наша задача – классифицировать непомеченные точки (серые), отнеся их к зеленым или сиреневым. Как и в архитектуре без учителя, если статистические распределения, из которых выбраны данные, четко разделены, то наличие меток в обучающем наборе позволяет без труда классифицировать непомеченные данные. Алгоритмы обучения с учителем также работают плохо, если распределения данных сильно перекрываются.

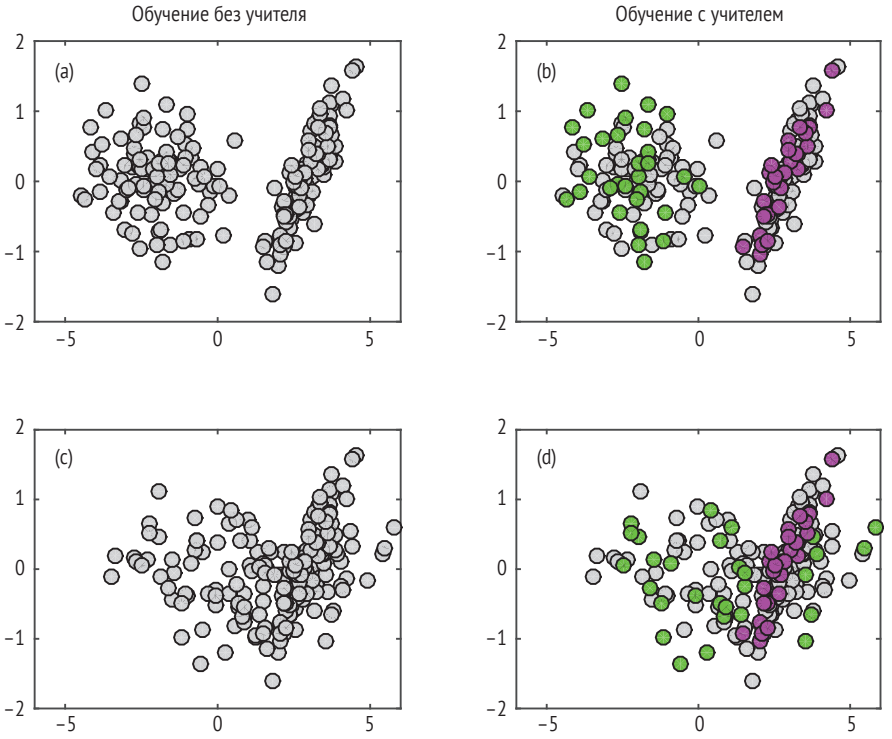

Рис. 5.7 ❖ Сравнение обучения с учителем и без учителя. На левых рисунках (a) и (c) алгоритм обучения без учителя пытается найти кластеры и разделить данные на две группы. Если данные четко разделены, как на рисунке (a), то задача проста и данным можно легко присвоить метки. Если же данные перекрываются, как на рисунке (c), то эта задача очень трудна. На правых рисунках (b) и (d) алгоритм обучения с учителем получает помеченные данные: зеленые и сиреневые точки. А затем должен классифицировать непомеченные данные: раскрасить в зеленый или сиреневый цвет. Для четко разделенных данных (b) задача пометки решается легко, а для перекрывающихся – с трудом

Задачи обучения с учителем и без учителя можно поставить математически. Пусть

$$\mathcal{D} \subset \mathbb{R}^n \tag{5.1}$$

– открытое ограниченное множество в n-мерном пространстве. Пусть далее

$$\mathcal{D}' \subset \mathcal{D}. \tag{5.2}$$

Цель классификации – построить классификатор, который способен пометить все данные в $\mathcal{D}$, зная, как помечены данные в $\mathcal{D}'$.

Чтобы уточнить постановку задачи, рассмотрим множество точек $\mathbf{x}_j \in \mathbb{R}^n$ и метки $\mathbf{y}_j$ для каждой точки, где $j = 1, 2, \cdots, m$. Метки данных могут иметь различную форму: от числовых значений, в т. ч. целочисленных, до текстовых строк. Для простоты ограничимся бинарными метками, так что $\mathbf{y}_j \in \{\pm 1\}$.

Тогда с задачей обучения классификатора без учителя ассоциируются следующие входы и выходы.

Вход

данные $\{\mathbf{x}_j \in \mathbb{R}^n, j \in Z := \{1, 2, \cdots, m\}\}$. (5.3a)

Выход

метки $\{\mathbf{y}_j \in \{\pm 1\}, j \in Z\}$. (5.3b)

Таким образом, математически обучение без учителя означает порождение меток $\mathbf{y}_j$ для всех данных. В общем случае данные $\mathbf{x}_j$, используемые для обучения классификатора, берутся из $\mathcal{D}'$. Затем классификатор применяется к более широкому множеству, т. е. обобщается на открытую ограниченную область $\mathcal{D}$. Если данные, используемые для построения классификатора, являются лишь небольшой выборкой из гораздо большего множества, то часто бывает, что классификатор обобщается плохо.

В случае обучения с учителем на этапе обучения предъявляются метки. Входы и выходы для задачи классификации выглядят так:

Вход

данные $\{\mathbf{x}_j \in \mathbb{R}^n, j \in Z := \{1, 2, \cdots, m\}\}$; (5.4a)

метки $\{\mathbf{y}_j \in \{\pm 1\}, j \in Z' \subset Z\}$. (5.4b)

Выход

метки $\{\mathbf{y}_j \in \{\pm 1\}, j \in Z)$. (5.4c)

В этом случае помечается подмножество данных, а оставшимся данным метки должен назначить алгоритм. Технически это обучение с частичным привлечением учителя, потому что для некоторых обучающих данных метки отсутствуют. В случае обучения с учителем должны быть известны все метки, чтобы построить классификатор на $\mathcal{D}'$. Затем этот классификатор применяется к $\mathcal{D}$. Как и при обучении без учителя, если данные, используемые для построения классификатора, являются лишь небольшой выборкой из гораздо большего множества, то часто бывает, что классификатор обобщается плохо.

Для наборов данных, описанных в разделе о выделении признаков и добыче данных, можно детально рассмотреть ключевые компоненты, необходимые для построения модели классификации: $\mathbf{x}_j$, $\mathbf{y}_j$, $\mathcal{D}$ и $\mathcal{D}'$. Набор данных Фишера об ирисах на рис. 5.1 – классический пример, для которого эти величины можно расписать полностью.

$$\mathbf{x}_j = \{\text{длина чашелистика, ширина чашелистика, длина лепестка, ширина лепестка}\}. \tag{5.5}$$

Таким образом, каждое измерение состоит из четырех полей, или признаков. Метки могут принимать следующие значения:

$$\mathbf{y}_j = \{\text{щетинистый, разноцветный, виргинский}\}. \tag{5.6}$$

В данном случае метки – текстовые строки, а всего их три. Заметим, что в нашей постановке задачи обучения с учителем и без учителя предполагалось только два выхода (бинарные метки): ± 1. Вообще говоря, меток может быть много, и часто это строки.

Наконец, имеется множество данных, а именно:

$$\mathcal{D}' \in \{\text{150 образцов ирисов: 50 щетинистых, 50 разноцветных, 50 виргинских}\} \tag{5.7}$$

и

$$\mathcal{D} \in \{\text{все щетинистые, разноцветные и виргинские ирисы}\}. \tag{5.8}$$

Аналогично мы можем представить данные о кошках и собаках:

$$\mathbf{x}_j = \{\text{изображение } 64 \times 64 = 4096 \text{ пикселей}\}, \tag{5.9}$$

а каждое изображение снабдить меткой вида

$$\mathbf{y}_j = \{\text{собака, кошка}\} = \{1, -1\}. \tag{5.10}$$

В этом примере метки – это текстовые строки, которые легко преобразовать в числовые значения. Это согласуется с нашей формулировкой обучения с учителем и без учителя, в которой фигурируют только две метки: ± 1. Наконец, рассмотрим такое множество данных:

$$\mathcal{D}' \in \{\text{160 изображений: 80 собак и 80 кошек}\} \tag{5.11}$$

и

$$\mathcal{D} \in \{\text{все изображения собак и кошек}\}. \tag{5.12}$$

Цель методов обучения с учителем и без учителя – создать алгоритмы для классификации, кластеризации и регрессии. Выше обсуждалась общая стратегия классификации, а в предыдущей главе – архитектуры регрессии. В обеих задачах требуется построить модель по данным в $\mathcal{D}'$, которая обобщалась бы на $\mathcal{D}$. Как уже было показано в предыдущей главе, обобщаемость может оказаться очень трудно достижимой, поэтому чрезвычайно важны стратегии

перекрестной проверки. Создание надежных обобщаемых схем – передний край исследований в области машинного обучения.

Некоторые трудности обобщения иллюстрируются на рис. 5.8. С первого взгляда видно, что эти наборы данных легко кластеризуются и классифицируются. Но для многих схем регрессии и классификации они представляют большие трудности. Суть дела в том, что граница между кластерами – нелинейное многообразие, которое зачастую нелегко охарактеризовать. Кроме того, если выборочные данные $\mathcal{D}'$ содержат только часть многообразия, то модель классификации или регрессии почти наверняка не сможет охарактеризовать $\mathcal{D}$. И это всего лишь двумерная задача классификации. Нетрудно представить, насколько сложными могут быть такие погружения данных в пространствах более высокой размерности. Визуализация в этом случае практически невозможна, и остается только полагаться на то, что алгоритмы смогут провести осмысленные границы, разделяющие данные разных классов. В этой и в следующей главах мы опишем методы классификации и регрессии по данным из $\mathcal{D}'$ – с метками и без меток. Спектр математических методов для решения подобных задач довольно широк.

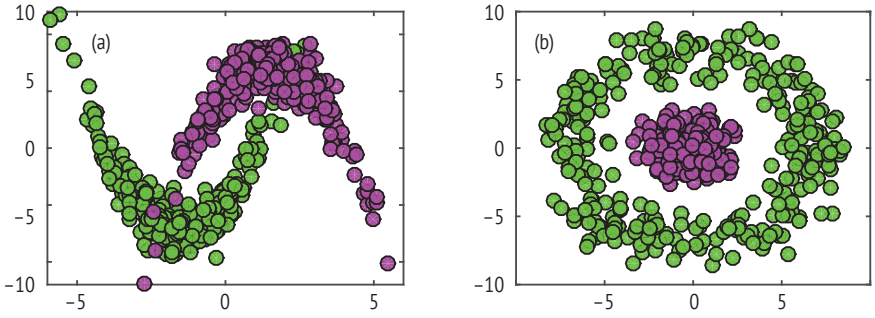

Рис. 5.8 ❖ Построение модели классификации и регрессии данных может оказаться затруднительным, если данные разделены нелинейной функцией. В данном случае выделить границу, отделяющую зеленые точки от сиреневых, нелегко. Кроме того, если выборочные данные $\mathcal{D}'$ содержат лишь малую часть разделительной границы, то, возможно, не удастся построить обобщаемую на $\mathcal{D}$ модель. На левом рисунке (a) показаны два серповидных множества, занимающих одну и ту же область в пространстве, а на рисунке (b) для разделения концентрических областей данных необходима окружность. Найти такие границы алгоритмически трудно

5.3. Обучение без учителя: кластеризация методом *k* средних

В этой главе будет описано несколько алгоритмов обучения с учителем и без. Начнем с одного из самых популярных на сегодняшний день алгоритмов обучения без учителя: кластеризации методом *k* средних. В алгоритме *k* сред-

них предполагается, что задано множество векторных данных, а цель – выделить во множестве m наблюдений k кластеров. Каждое наблюдение включается в кластер с ближайшей средней точкой, которая играет роль представителя этого кластера. В результате получается разбиение пространства данных на ячейки диаграммы Вороного.

Хотя число наблюдений и размерность системы известны, количество кластеров k обычно неизвестно и должно быть определено. Альтернативно пользователь может просто задать количество кластеров. Алгоритм k средних итеративный. Сначала каким-то образом выбираются средние для каждого кластера, а затем они обновляются, пока алгоритм не сойдется. На рис. 5.9 показано правило обновления в алгоритме k средних. Алгоритм работает следующим образом: (i) зная начальные значения k различных средних, вычислить расстояния между каждым наблюдением $\mathbf{x}_j$ и каждым из k средних; (ii) отнести каждое наблюдение к ближайшему среднему; (iii) после того как процедура отнесения к средним (пометка) завершена, вычислить *центр масс* (среднее) каждой группы одинаково помеченных точек. Эти центры становятся новыми средними, и алгоритм переходит к шагу (i). Данный эвристический алгоритм впервые был предложен Стюартом Ллойдом в 1957 году [339], но опубликован он был только в 1982 году.

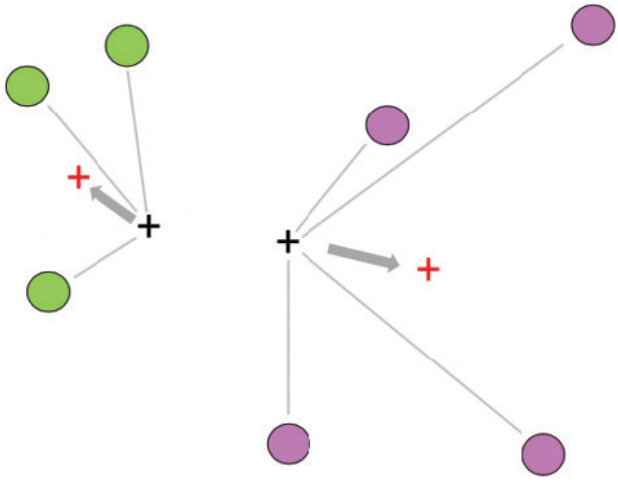

Рис. 5.9 ❖ Иллюстрация алгоритма k средних для $k = 2$. Заданы два начальных положения центров (черные +). Каждая точка помечается как отнесенная к одному из двух центров. Таким образом, зеленые точки отнесены к левому кластеру, а сиреневые – к правому. После пометки точек средние обоих кластеров пересчитываются (красные +). Процесс повторяется, пока средние не сойдутся

Алгоритм k средних можно формально описать как задачу оптимизации, а именно:

$$\underset{\boldsymbol{\mu}_j}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{\mathbf{x}_j \in \mathcal{D}_j'} ||\mathbf{x}_j - \boldsymbol{\mu}_j||^2, \tag{5.13}$$

где $\boldsymbol{\mu}_j$ – среднее j-го кластера, а $\mathcal{D}_j^i$ – часть множества данных, ассоциированная с этим кластером. Это задача минимизации внутрикластерной суммы квадратов расстояний. В общем случае она является *NP*-трудной, т. е. вычислительно неразрешимой. Но существует ряд эвристических алгоритмов, которые хорошо работают, хотя и не дают гарантии сходимости к глобально оптимальному решению.

Для определения качества алгоритма необходима перекрестная проверка, как всегда в машинном обучении. Поскольку метки отсутствуют, эта процедура имеет свои особенности – ведь нет объективной истины, с которой можно было бы сравнить результаты. Однако методы k-групповой перекрестной проверки, описанные в предыдущей главе, все же можно использовать для тестирования устойчивости классификатора к выборке различных подмножеств данных. Приведенный ниже код содержит реализацию алгоритма кластеризации методом k средних Ллойда. Сначала создадим два кластера и разделим данные на обучающий и тестовый наборы.

Листинг 5.5 ❖ Генерирование данных для метода k средних

```
% размеры обучающего и тестового наборов
n1=100;  % размер обучающего набора
n2=50;   % размер тестового набора

% случайный эллипс 1 с центром в точке (0,0)
x=randn(n1+n2,1); y=0.5*randn(n1+n2,1);

% случайный эллипс 2 с центром в точке (1,-2), повернутый на угол theta
x2=randn(n1+n2,1)+1; y2=0.2*randn(n1+n2,1)-2; theta=pi/4;
A=[cos(theta) -sin(theta); sin(theta) cos(theta)];
x3=A(1,1)*x2+A(1,2)*y2; y3=A(2,1)*x2+A(2,2)*y2;
subplot(2,2,1)
plot(x(1:n1),y(1:n1),'ro'), hold on
plot(x3(1:n1),y3(1:n1),'bo')

% обучающий набор: первые 200 из 240 точек
X1=[x3(1:n1) y3(1:n1)];
X2=[x(1:n1) y(1:n1)];
Y=[X1; X2]; Z=[ones(n1,1); 2*ones(n1,1)];

% тестовый набор: остальные 40 точек
x1test=[x3(n1+1:end) y3(n1+1:end)];
x2test=[x(n1+1:end) y(n1+1:end)];
```

На рис. 5.10 показаны сгенерированные данные, выбранные из двух разных нормальных распределений. В данном случае у нас имеются истинные данные для проверки кластеризации методом k средних. Но, вообще говоря, это не так. Алгоритм Ллойда наугад выбирает количество кластеров и места расположения средних каждого кластера, но во многих его современных версиях реализованы теоретически обоснованные стратегии инициализации.

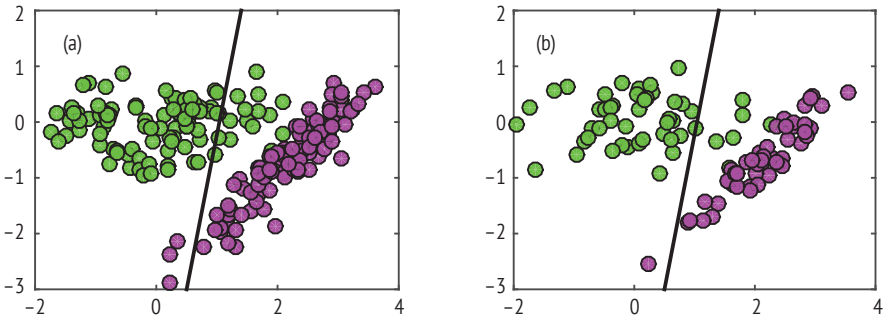

Рис. 5.10 ❖ Кластеризация методом *k* средних с использованием команды MATLAB **kmeans**. Необходимо задать только сами данные и количество кластеров. (а) Обучающие данные используются для построения решающей границы (черная прямая), разделяющей кластеры. Заметим, что граница, очевидно, не оптимальна. Затем верность модели можно протестировать на зарезервированных данных. При этом одна из 50 сиреневых точек и шесть из 50 зеленых помечены неправильно

Листинг 5.6 ❖ Алгоритм *k* средних Ллойда

```
g1=[-1 0]; g2=[1 0]; % начальная гипотеза
for j=1:4
    class1=[]; class2=[];
    for jj=1:length(Y)
        d1=norm(g1-Y(jj,:));
        d2=norm(g2-Y(jj,:));
        if d1<d2
            class1=[class1; [Y(jj,1) Y(jj,2)]];
        else
            class2=[class2; [Y(jj,1) Y(jj,2)]];
        end
    end
    g1=[mean(class1(1:end,1)) mean(class1(1:end,2))];
    g2=[mean(class2(1:end,1)) mean(class2(1:end,2))];
end
```

На рис. 5.11 показана итеративная процедура кластеризации методом *k* средних. Выбранные наугад центры используются для первоначальной пометки всех точек (рис. 5.11 (а)). Затем вычисляются новые средние и метки данных пересчитываются. Алгоритм сходится всего после четырех итераций. Этот код написан с нуля, специально чтобы показать, как быстро сходятся итерации пометки данных без учителя. В MATLAB уже имеется реализация алгоритма *k* средних, которой нужно передать только матрицу данных и желаемое количество кластеров. Она проста в использовании и представляет собой ценный диагностический инструмент. В программе ниже используется команда MATLAB **kmeans** и заодно строится *решающая граница*, разделяющая два кластера.

Листинг 5.7 ❖ Метод *k* средних в MATLAB

```
% вызов kmeans
[ind,c]=kmeans(Y,2);
plot(c(1,1),c(1,2),'k*','Linewidth',[2])
plot(c(2,1),c(2,2),'k*','Linewidth',[2])

midx=(c(1,1)+c(2,1))/2; midy=(c(1,2)+c(2,2))/2;
slope=(c(2,2)-c(1,2))/(c(2,1)-c(1,1));
b=midy+(1/slope)*midx;
xsep=-1:0.1:2; ysep=-(1/slope)*xsep+b;

figure(1), subplot(2,2,1), hold on
plot(xsep,ysep,'k','Linewidth',[2]),axis([-2 4 -3 2])

% ошибка на тестовых данных
figure(1), subplot(2,2,2)
plot(x(n1+1:end),y(n1+1:end),'ro'), hold on
plot(x3(n1+1:end),y3(n1+1:end),'bo')
plot(xsep,ysep,'k','Linewidth',[2]), axis([-2 4 -3 2])
```

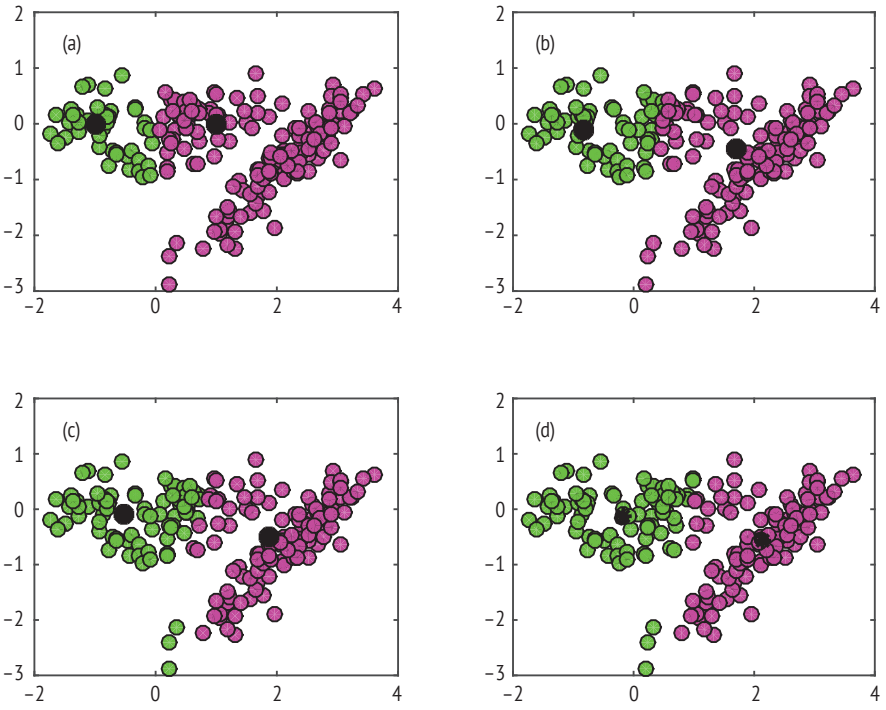

Рис. 5.11 ❖ Итеративная процедура в алгоритме Ллойда [339]. Ищутся два кластера, так что $k = 2$. Начальная гипотеза (черные точки на рисунке (a)) используется для первоначальной пометки всех данных в соответствии с расстоянием до среднего. Затем средние пересчитываются по только что помеченным данным. В данном случае этот двухэтапный эвристический алгоритм сходится после четырех итераций

На рис. 5.10 показаны результаты алгоритма *k* средних и прямая, разделяющая два кластера. Зеленые и сиреневые точки обозначают истинные метки данных. Видно, что найденная прямая разделяет метки не совсем правильно. На самом деле алгоритм обучения с учителем лучше согласуется с истинными данными, как будет показано ниже в этой главе. Впрочем, и этот алгоритм правильно пометил большую часть данных.

Популярность эвристического алгоритма *k* средних объясняется двумя причинами: (i) не требуется вмешательство учителя, и (ii) он работает быстро. Приведенный пример показывает, что алгоритм не особенно точен, но так часто бывает с алгоритмами обучения без учителя, потому что их знания о данных ограничены. Перекрестная проверка, в частности *k*-групповая, может улучшить модель и повысить точность обучения без учителя, но в общем случае оно все равно будет уступать обучению с учителем, в ходе которого предъявляются помеченные данные.

5.4. Иерархическая кластеризация без учителя: дендрограмма

Еще один часто используемый алгоритм кластеризации данных без учителя называется *дендрограммой*. Как и в случае кластеризации методом *k* средних, дендрограммы создаются простым иерархическим алгоритмом, который допускает эффективную визуализацию, когда кластеризуемые данные не помечены. Мы применим этот иерархический подход к данным, показанным на рис. 5.12, для которых известна объективная истина. Методы иерархической кластеризации бывают двух видов: нисходящие и восходящие, хотя в данном случае применяется специальная терминология.

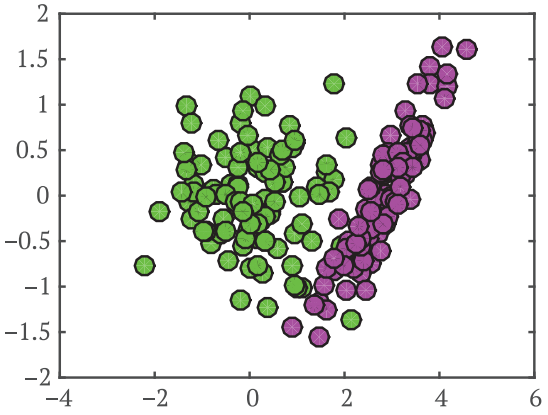

Рис. 5.12 ❖ Пример данных, используемых для построения дендрограммы. Данные выбраны из двух нормальных распределений (по 50 точек из каждого), так чтобы их было легко различить визуально. Алгоритм порождает иерархию, которая идеально отделяет зеленые точки от сиреневых

Агломеративная кластеризация. Первоначально каждая точка $\mathbf{x}_j$ находится в своем собственном кластере. Затем кластеры объединяются в пары. Объединение кластеров прекращается, после того как все данные окажутся в суперкластере. Это восходящий подход к иерархической кластеризации.

Дивизивная кластеризация. Первоначально все точки принадлежат одному гигантскому кластеру. Затем производится рекурсивное разбиение на меньшие кластеры. Разбиение продолжается, когда будет выполнено заданное пользователем условие. Дивизивный метод может продолжать разбиение до тех пор, пока каждая точка не окажется в отдельном кластере.

В общем случае объединение и разбиение данных производятся эвристическим жадным алгоритмом, который легко реализовать с вычислительной точки зрения. Результаты иерархической кластеризации обычно представляются в виде дендрограммы.

В этом разделе нас будет интересовать агломеративная кластеризация и команда MATLAB **dendrogram**. Как и алгоритм *k* средних Ллойда, построение дендрограммы опирается на вычисление расстояния между точками. Обычно мы используем евклидово расстояние, но существуют и другие метрики, уместность которых определяется типом данных. Приведем наиболее распространенные:

$$\text{евклидово расстояние } ||\mathbf{x}_j - \mathbf{x}_k||_2; \tag{5.14a}$$

$$\text{квадрат евклидова расстояния } ||\mathbf{x}_j - \mathbf{x}_k||_2^2; \tag{5.14b}$$

$$\text{манхэттенское расстояние } ||\mathbf{x}_j - \mathbf{x}_k||_1; \tag{5.14c}$$

$$\text{максимальное расстояние } ||\mathbf{x}_j - \mathbf{x}_k||_\infty; \tag{5.14d}$$

$$\text{расстояние Махаланобиса } \sqrt{(\mathbf{x}_j - \mathbf{x}_k)^T(\mathbf{x}_j - \mathbf{x}_k)}, \tag{5.14e}$$

где $\mathbf{C}$ – ковариационная матрица. Как уже отмечалось в предыдущей главе, выбор нормы может оказать огромное влияние на выявление паттернов в данных, а значит, на успешность кластеризации и классификации.

Дендрограммы показаны на рис. 5.13. Алгоритм работает следующим образом: (i) вычисляются попарные расстояния между всеми m точками $\mathbf{x}_j$, (ii) две ближайшие точки объединяются в одну, расположенную посередине между ними, (iii) процедура повторяется для новых $m - 1$ точек. Алгоритм продолжает работать, пока не останется одна точка.

Программа в листинге 5.8 выполняет иерархическую кластеризацию с помощью команды **dendrogram**. Мы использовали те же данные, что при кластеризации методом *k* средних, они показаны на рис. 5.12. Визуально отчетливо видны два кластера. Нас интересует, сумеет ли их выделить алгоритм дендрограммы.

Листинг 5.8 ❖ Кластеризация без учителя методом дендрограммы

```
Y3=[X1(1:50,:); X2(1:50,:)];
Y2 = pdist(Y3,'euclidean');
Z = linkage(Y2,'average');
thresh=0.85*max(Z(:,3));
[H,T,O]=dendrogram(Z,100,'ColorThreshold',thresh);
```

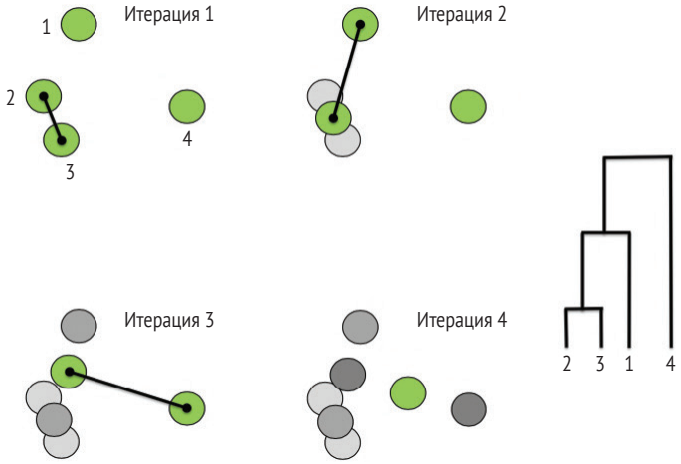

Рис. 5.13 ❖ Агломеративная иерархическая кластеризация в применении к четырем точкам. Алгоритм вычисляет евклидово расстояние между точками. В начальный момент минимальным является расстояние между точками 2 и 3. Эти точки объединяются в одну, расположенную посередине между ними, после чего расстояния пересчитываются. Дендрограмма справа показывает, как развивался процесс иерархической кластеризации. Заметим, что длины ветвей дерева зависят от расстояния между объединенными точками

На рис. 5.14 показана дендрограмма, соответствующая данным на рис. 5.12. Видно, какие точки объединялись, а также расстояния между точками. Задавая порог `thresh`, мы можем управлять тем, сколько кластеров будет построено. В программе ниже дендрограмма используется, чтобы показать, как данные разделены на классы. Напомним, что первые 50 точек принадлежат зеленому кластеру, а вторые – сиреневому.

Листинг 5.9 ❖ Дендрограмма классифицирует зеленые и сиреневые точки

```
bar(0), hold on
plot([0 100],[50 50],'r:','Linewidth',2)
plot([50.5 50.5],[0 100],'r:','Linewidth',2)
```

На рис. 5.15 показано, как были кластеризованы данные в дендрограмме. Если бы удалось добиться идеальной кластеризации, то первые 50 точек оказались бы ниже горизонтальной пунктирной линии, а вторые 50 точек – выше нее. Вертикальная красная пунктирная линия разделяет зеленые точки слева от нее и сиреневые справа.

Приведенный ниже код показывает, как можно увеличить число кластеров, изменив порог в команде **dendrogram**. Это то же самое, что в алгоритме *k* средних задать более двух кластеров. Напомним, что истинная картина, с которой можно было бы сравнить результат кластеризации без учителя, редко бывает известна, поэтому так важно правильно настроить порог.

```
thresh=0.25*max(Z(:,3));
[H,T,O]=dendrogram(Z,100,'ColorThreshold',thresh);
```

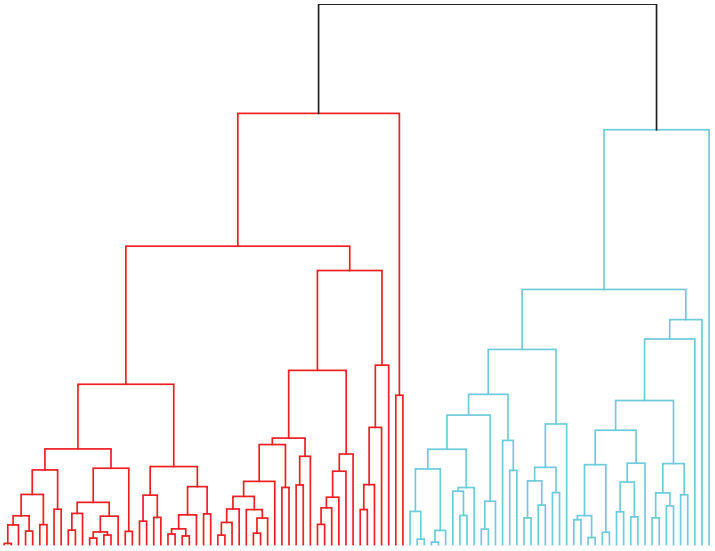

Рис. 5.14 ❖ Дендрограмма, построенная по данным на рис. 5.12. Она показывает не только, какие точки объединялись, но и расстояния между ними. При заданном пороге сгенерировано два кластера

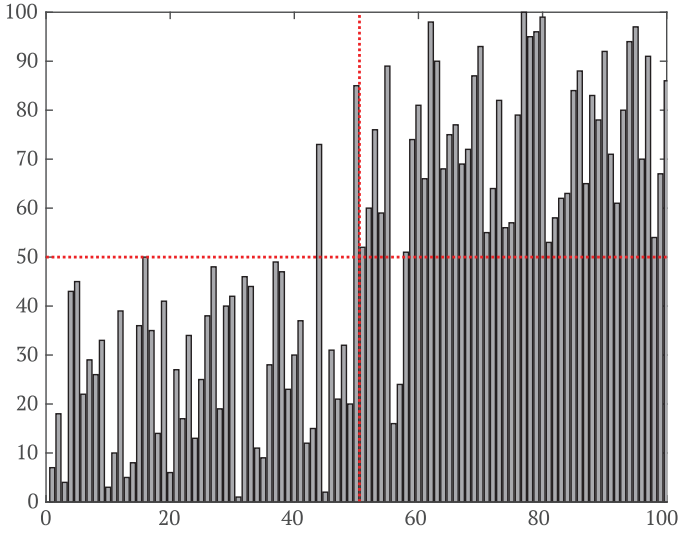

Рис. 5.15 ❖ Результат кластеризации командой **dendrogram**. Эта диаграмма получена на основе рис. 5.14 и показывает, как точки отнесены к кластерам, исходя из расстояний между ними. Горизонтальная красная пунктирная прямая показывает, где должно было бы проходить идеальное разделение. Первые 50 точек (зеленые точки на рис. 5.12) должны были бы быть сгруппированы так, чтобы оказаться ниже горизонтальной линии в левом нижнем квадранте. Вторые 50 точек (сиреневые точки на рис. 5.12) должны были бы оказаться выше горизонтальной линии в правом верхнем квадранте. Как выясняется, дендрограмма неправильно классифицировала только две зеленые и две сиреневые точки

На рис. 5.16 показана дендрограмма с другим порогом. Отметим, что в этом случае алгоритм создал более дюжины кластеров. Отсюда следует, что величина порога – такой же важный параметр иерархической кластеризации, как количество кластеров в методе k средних. Резюмируя, можно сказать, что и метод k средних, и иерархическая кластеризация позволяют автоматически разбить данные на кластеры. Это отправная точка для интерпретации и анализа в добыче данных.

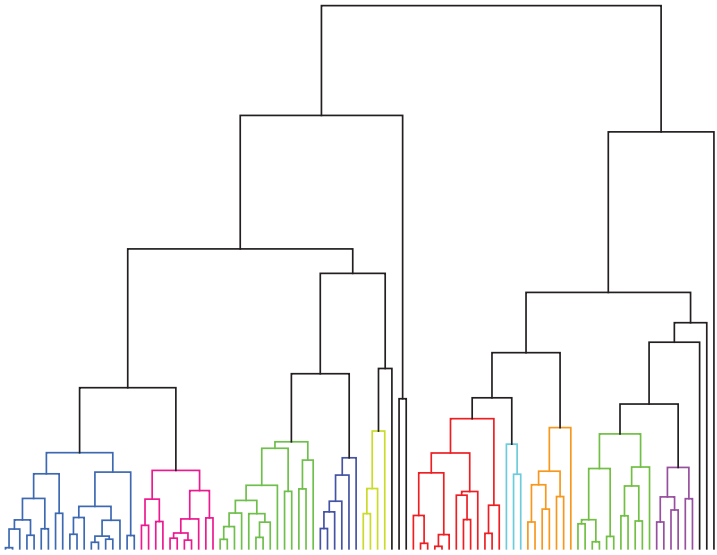

Рис. 5.16 ❖ Дендрограмма, построенная по данным на рис. 5.12, с порогом, отличным от использованного при построении рис. 5.14. Видно, какие точки объединялись, а также расстояния между точками. В этом случае сгенерировано больше дюжины кластеров

5.5. Смесовые модели и EM-алгоритм

Третий метод обучения без учителя называется *конечные смесовые модели*. Часто предполагается, что модель описывается смесью гауссовых распределений, и тогда этот метод называется *моделью гауссовой смеси* (Gaussian mixture models – GMM). Основное предположение заключается в том, что наблюдения $\mathbf{x}_j$ описываются смесью k процессов. Подобно методам k средних и иерархической кластеризации, для аппроксимации данных GMM-моделью необходимо задать количество распределений в смеси и статистические свойства каждого распределения, лучше всего отвечающие данным. GMM полезны, потому что из предположения о том, что все компоненты смеси гауссовы, следует, что их можно полностью охарактеризовать двумя параметрами: средним и дисперсией.

Алгоритм, лежащий в основе GMM, вычисляет максимальное правдоподобие, пользуясь знаменитым EM-алгоритмом (Expectation-Maximization),

описанным в работе Dempster, Laird and Rubin [148]. EM-алгоритм разработан для нахождения максимально правдоподобных параметров статистических моделей. Он итеративно ищет локальное максимальное правдоподобие для оценки истинных параметров, вычислить которые напрямую невозможно. Как обычно, наблюдаемые данные могут содержать много латентных, т. е. не измеряемых непосредственно, величин и неизвестных параметров. Но итеративный алгоритм с чередующимися шагами тем не менее позволяет найти наилучшие параметры при заданной начальной гипотезе. EM-алгоритм, как и алгоритм *k* средних, исходит из начальных предположений о средних и дисперсиях *k* гипотетических распределений. Затем алгоритм обновляет веса смесей и пересчитывает их параметры. Выполнение чередующихся шагов продолжается, пока не будет достигнута сходимость.

В любой подобной итеративной схеме заранее не очевидно, сойдется ли алгоритм к решению и будет ли это решение хорошим, поскольку алгоритм может застрять в локальном максимуме правдоподобия. Но можно доказать, что именно данный алгоритм сходится и что производная правдоподобия может быть сделана сколь угодно близкой к нулю, т. е. предел действительно является либо максимумом, либо седловой точкой [561]. В общем случае может существовать несколько максимумов, и нет никакой гарантии, что будет найден глобальный максимум. К тому же функция правдоподобия может иметь особые точки, т. е. максимумы, не имеющие смысла. Например, одна из компонент найденной EM-алгоритмом смеси может иметь нулевую дисперсию и среднее, совпадающее с одной из точек. Перекрестная проверка зачастую обнаруживает типичные проблемы, возникающие в случае неудачной инициализации параметров.

Главное предположение в смесовой модели – то, что функция плотности вероятности (ФПВ) для наблюдений $\mathbf{x}_j$ является взвешенной суммой нескольких неизвестных распределений

$$f(\mathbf{x}_j, \boldsymbol{\Theta}) = \sum_{p=1}^{k} \alpha_p f_p(\mathbf{x}_j, \boldsymbol{\Theta}), \tag{5.15}$$

где $f(\cdot)$ – измеренная ФПВ, $f_p(\cdot)$ – ФПВ j-го компонента смеси, а k – общее число компонент смеси. ФПВ $f_j(\cdot)$ имеет вес $\alpha_p(\alpha_1 + \alpha_2 + \cdots + \alpha_k = 1)$ и параметризуется неизвестным вектором параметров $\boldsymbol{\Theta}_p$. Теперь сформулируем цель смесовой модели более точно: *при заданной наблюдаемой ФПВ* $f(\mathbf{x}_j, \boldsymbol{\Theta})$ *оценить веса* α_p *и параметры распределений* $\boldsymbol{\Theta}_p$. Отметим, что вектор $\boldsymbol{\Theta}$ содержит все параметры $\boldsymbol{\Theta}_p$. Немного упрощает задачу тот факт, что мы предполагаем известной форму ФПВ $f_p(\cdot)$.

Для GMM известно, что вектор $\boldsymbol{\Theta}_p$ содержит всего два параметра: среднее μ_p и дисперсию σ_p. Кроме того, распределение $f_p(\cdot)$ нормально, т. е. формула (5.15) принимает вид

$$f(\mathbf{x}_j, \boldsymbol{\Theta}) = \sum_{p=1}^{k} \alpha_p \mathcal{N}_p(\mathbf{x}_j, \mu_p, \sigma_p). \tag{5.16}$$

В такой постановке задача уже не кажется неразрешимой, потому что множество параметров ограничено. Коль скоро мы предположили, что смесь состоит из k компонентов, требуется определить α_p, μ_p и σ_p для каждого компонента. Следует сказать, что, помимо гауссова, существует много других распределений, но модель GMM так широко распространена, потому что без априорной информации предположение о нормальности распределения наиболее естественно.

Оценку вектора параметров $\boldsymbol{\Theta}$ можно вычислить с помощью *оценки максимального правдоподобия* (ОМП) Фишера. При этом $\boldsymbol{\Theta}$ вычисляется как решение уравнения

$$\frac{\partial L(\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} = 0, \tag{5.17}$$

где функция максимального правдоподобия L имеет вид

$$L(\boldsymbol{\Theta}) = \sum_{j=1}^{n} \log f(\mathbf{x}_j|\boldsymbol{\Theta}), \tag{5.18}$$

а суммирование производится по всем n векторам $\mathbf{x}_j$. Решение этой задачи оптимизации дает локальный максимум. Его можно найти с помощью EV-алгоритма, поскольку вычислить производные явно, не зная аналитической формы функции, невозможно.

В начале EM-алгоритма делается начальное предположение о векторе параметров $\boldsymbol{\Theta}$. Это предположение можно использовать для вычисления оценки

$$\tau_p(\mathbf{x}_j, \boldsymbol{\Theta}) = \frac{\alpha_p f_p(\mathbf{x}_j, \boldsymbol{\Theta})}{f(\mathbf{x}_j, \boldsymbol{\Theta})}, \tag{5.19}$$

т. е. апостериорной вероятности того, что $\mathbf{x}_j$ принадлежит p-й компоненте смеси. На E-шаге EM-алгоритма эта апостериорная вероятность используется для вычисления принадлежностей. В случае GMM алгоритм работает следующим образом: зная начальные значения $\boldsymbol{\Theta}$ и α_p, вычислить

$$\tau_p^{(k)}(\mathbf{x}_j) = \frac{\alpha_p^{(k)} \mathcal{N}_p(\mathbf{x}_j, \mu_p^{(k)}, \sigma_p^{(k)})}{\mathcal{N}(\mathbf{x}_j, \boldsymbol{\Theta}^{(k)})}. \tag{5.20}$$

Имея оценку апостериорной вероятности, алгоритм на M-шаге обновляет параметры и веса компонентов смеси:

$$\alpha_p^{(k+1)} = \frac{1}{n} \sum_{j=1}^{n} \tau_p^{(k)}(\mathbf{x}_j); \tag{5.21a}$$

$$\mu_p^{(k+1)} = \frac{\sum_{j=1}^{n} \mathbf{x}_j \tau_p^{(k)}(\mathbf{x}_j)}{\sum_{j=1}^{n} \tau_p^{(k)}(\mathbf{x}_j)}; \tag{5.21b}$$

$$\Sigma_p^{(k+1)} = \frac{\sum_{j=1}^{n} \tau_p^{(k)}(\mathbf{x}_j)\left(\mathbf{x}_j - \mu_p^{(k+1)}\right)\left(\mathbf{x}_j - \mu_p^{(k+1)}\right)^T}{\sum_{j=1}^{n} \tau_p^{(k)}(\mathbf{x}_j)}, \tag{5.21c}$$

где $\Sigma_p^{(k+1)}$ – ковариационная матрица, содержащая дисперсии. E-шаг и M-шаг чередуются, пока не будет достигнута сходимость с заданной точностью. Напомним, что на этапе инициализации алгоритма необходимо задать число компонентов смеси k и начальные гипотезы о параметрах распределений. Это похоже на алгоритм k средних, в котором число кластеров k задается заранее и высказывается гипотеза о положениях центров кластеров.

Алгоритм GMM популярен, потому что просто аппроксимирует данные смесью k гауссовых распределений, что выглядит разумно в случае обучения без учителя. Но у него более основательная теоретическая база, чем у большинства методов обучения без учителя, в частности k средних и иерархической кластеризации, которые всего лишь определены алгоритмически. Основное предположение в GMM – количество кластеров и форма распределения $f(\cdot)$.

Программа ниже строит модель GMM по второй и четвертой главным компонентам вейвлет-преобразований изображений собак и кошек (см. рис. 5.4–5.6). Таким образом, признаками являются второй и четвертый столбцы правого сингулярного вектора SVD. Для построения смесовой модели служит команда **fitgmdist**.

Листинг 5.10 ❖ Модель гауссовой смеси для разделения собак и кошек

```
dogcat=v(:,2:2:4);
GMModel=fitgmdist(dogcat,2)
AIC= GMModel.AIC

subplot(2,2,1)
h=ezcontour(@(x1,x2)pdf(GMModel,[x1 x2]));
subplot(2,2,2)
h=ezmesh(@(x1,x2)pdf(GMModel,[x1 x2]));
```

Результаты работы алгоритма визуализируются, печатаются также параметры каждого распределения. Точнее, выводится доля каждого распределения в смеси, а еще средние по обоим измерениям пространства признаков.

```
Component 1:
Mixing proportion: 0.355535
Mean: -0.0290 -0.0753

Component 2:
Mixing proportion: 0.644465
Mean: 0.0758 0.0076

AIC =
  -792.8105
```

Программа также выводит информационный критерий Акаике (AIC), показывающий, насколько хорошо гауссова смесь объясняет данные. Это тео-

ретическая основа для перекрестной проверки с целью определения подходящего для описания данных количества компонент смеси.

На рис. 5.17 показаны результаты аппроксимации методом GMM вместе с исходными данными о собаках и кошках. Показаны также вычисленные гауссианы. Команду **fitgmdist** можно использовать вместе с командой **cluster** для пометки данных, полученных в результате разделения признаков, найденного GMM.

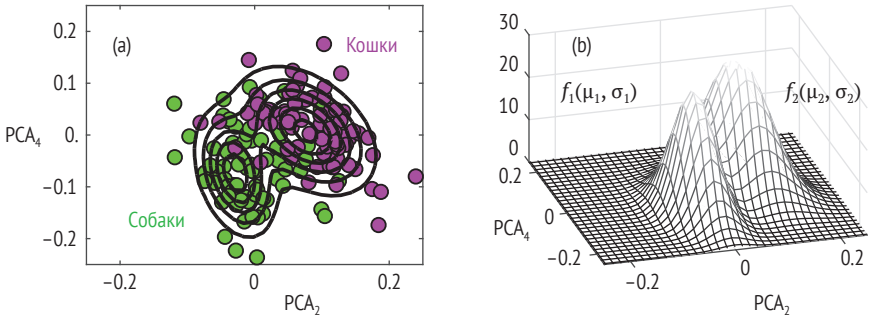

Рис. 5.17 ❖ Аппроксимация второй и четвертой главных компонент вейвлет-преобразований изображений собак и кошек с помощью GMM. На рисунке (a) видно, что одно гауссово распределение сконцентрировано в области признаков собак, а второе – кошек. ФПВ выделенных компонент гауссовой смеси показаны на рисунке (b) в условных единицах измерения

5.6. Обучение с учителем и линейные дискриминанты

Теперь обратимся к методам обучения с учителем. Один из самых первых таких методов классификации данных был предложен Фишером в 1936 году в контексте таксономии [182]. Его *линейный дискриминантный анализ* (ЛДА) до сих пор является одним из стандартных методов классификации. В 1948 году он был обобщен К. Р. Рао на случай более двух классов [446]. Цель этих алгоритмов – найти линейную комбинацию признаков, которая характеризует или разделяет два или более классов объектов либо событий, присутствующих в данных. Важно, что это метод обучения с учителем, т. е. мы помечаем данные, помогая тем самым алгоритму классификации. На рис. 5.18 иллюстрируется идея поиска оптимального погружения данных в пространство меньшей размерности для последующей классификации. Алгоритм ЛДА решает задачу оптимизации – поиск подпространства, в котором помеченные данные лучше всего различимы. Впоследствии это упрощает классификацию, т. к. уже выбрано оптимальное пространство признаков.

В архитектуре обучения с учителем имеется обучающий и тестовый наборы данных. Данные из тестового набора никогда не используются для обучения классификатора. С другой стороны, обучающие данные можно разделить

на k групп, например, чтобы улучшить модель классификации. В предыдущей главе мы подробно рассказывали о том, как следует использовать перекрестную проверку. Сейчас наша цель – обучить алгоритм, использующий пространство признаков, принимать решение о том, как классифицировать данные. На рис. 5.18 схематично показана основная идея ЛДА. Рассматриваются два набора данных, которые проецируются на разные подпространства. В левой проекции данные перемешаны, что не позволяет их различить. На правом же рисунке показан идеальный результат ЛДА, когда проекции данных со средними μ_1 и μ_2 четко разделены. Таким образом, ЛДА имеет двоякую цель: *найти такую проекцию, которая максимизирует расстояние между данными разных классов и одновременно минимизирует расстояние между данными одного класса*.

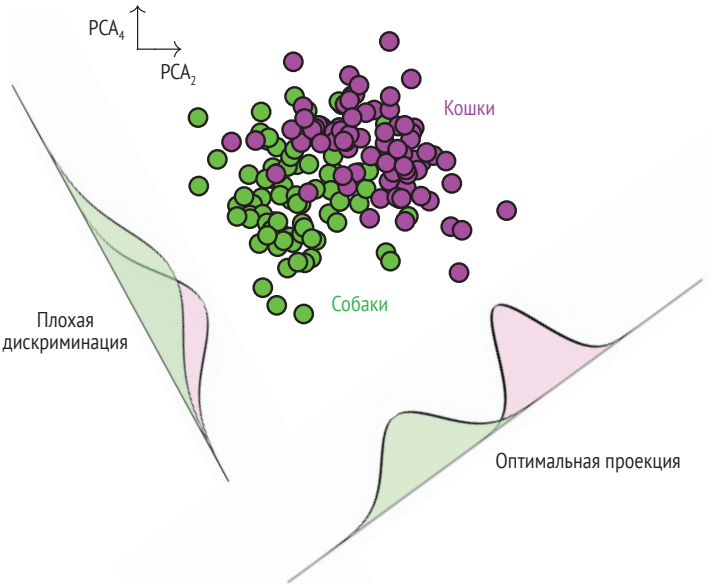

Рис. 5.18 ❖ Линейный дискриминантный анализ (ЛДА). Метод оптимизации ЛДА находит оптимальную проекцию для классификации. На рисунке показана проекция второй и четвертой главных компонент вейвлет-преобразования данных о собаках и кошках (см. рис. 5.4). Без оптимизации проекция могла бы породить очень плохую дискриминацию данных. Но ЛДА разделяет функции распределения вероятностей оптимальным образом

Для ЛДА с двумя классами это математически формулируется следующим образом. Построить проекцию $\mathbf{w}$ такую, что

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}, \tag{5.22}$$

где матрицы рассеяния для межклассовых ($\mathbf{S}_B$) и внутриклассовых ($\mathbf{S}_W$) данных имеют вид:

$$\mathbf{S}_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T; \tag{5.23}$$

$$\mathbf{S}_W = \sum_{j=1}^{2} \sum_{x} (\mathbf{x} - \mu_j)(\mathbf{x} - \mu_j)^T. \tag{5.24}$$

Эти величины измеряют дисперсию наборов данных и дисперсию разностей средних. Критерий (5.22) известен как обобщенный коэффициент Рэлея и может быть найден путем решения обобщенного уравнения собственных значений:

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w}, \tag{5.25}$$

где максимальное собственное значение λ и соответствующий ему собственный вектор дают интересующую нас величину и базис проекции. Таким образом, после построения матриц рассеяния в MATLAB можно найти обобщенные собственные векторы.

Выполнить ЛДА в MATLAB просто. Нужно только организовать данные в виде обучающего набора с метками, а затем проверить результат на тестовом наборе. Получив набор $\mathbf{x}_j$, $j = 1, 2, \cdots, m$, и соответствующие метки $\mathbf{y}_j$, алгоритм найдет оптимальное для классификации пространство, как показано на рис. 5.18. Затем можно проверить качество, классифицировав набор $\mathbf{x}_k$, $k = m + 1, m + 2, \cdots, m + n$. Мы проиллюстрируем это на примере классификации данных о собаках и кошках, описанных в разделе этой главы о признаках. Точнее, рассмотрим изображения собак и кошек в вейвлетной области и пометим их так, что $\mathbf{y}_j = 1$ для собак и $\mathbf{y}_j = -1$ для кошек. Программа в листинге 5.11 обучается на первых 60 изображениях собак и кошек, а затем тестирует классификатор на остальных 20 изображениях. Для простоты мы проводим обучение на второй и четвертой главных компонентах, поскольку они лучше всего различают собак и кошек (см. рис. 5.5).

Листинг 5.11 ❖ ЛДА изображений собак и кошек

```
load catData_w.mat
load dogData_w.mat
CD=[dog_wave cat_wave];
[u,s,v]=svd(CD-mean(CD(:)));

xtrain=[v(1:60,2:2:4); v(81:140,2:2:4)];
label=[ones(60,1); -1*ones(60,1)];
test=[v(61:80,2:2:4); v(141:160,2:2:4)];

class=classify(test,xtrain,label);
truth=[ones(20,1); -1*ones(20,1)];
E=100-sum(0.5*abs(class-truth))/40*100
```

Заметим, что команда **classify** в MATLAB принимает все три матрицы: обучающие данные, тестовые данные и метки для обучающих данных. Возвращает же она метки для тестовых данных. От этой команды можно также получить найденную прямую для использования в интерактивных програм-

мах. На рис. 5.19 показаны результаты классификации 40 тестовых примеров. Напомним, что классификация производится только по второй и четвертой модам PCA, которые кластеризуются, как показано на рис. 5.18. Возвращаются метки ± 1. Правильные метки для тестовых данных – сначала 20 значений $+1$ (собаки), затем 20 значений -1 (кошки). Точность классификации в этом примере составила 82.5 % (2 из 20 кошек и 5 из 20 собак классифицированы неправильно). Сравнивая вейвлет-изображения с исходными, мы видим, что отбирать признаки из исходных изображений было бы хуже. В частности, для тех же двух главных компонент неправильно классифицированы 9 из 20 кошек и 4 из 20 собак. Конечно, данных маловато, и для оценки классификатора всегда нужно применять перекрестную проверку. В следующей программе команда **classify** прогоняется 100 раз на 60 случайно выбираемых изображениях собак кошек, а затем классификатор тестируется на остальных 20 изображениях.

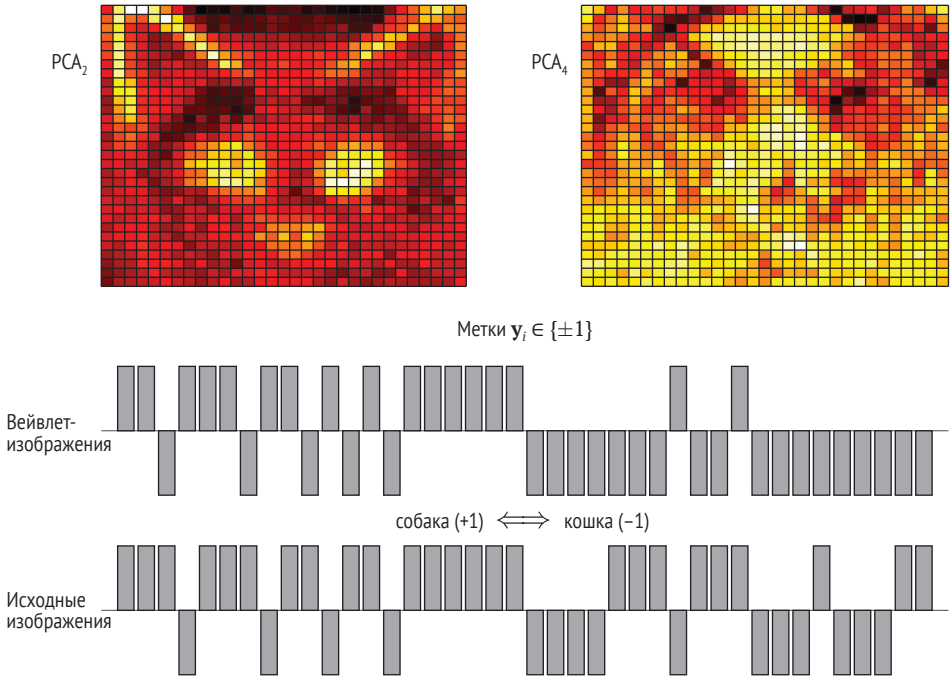

Рис. 5.19 ❖ Достигнутое качество классификации на второй и четвертой главных компонентах. На верхних рисунках показаны моды PCA (признаки), по которым строился классификатор. Метки $\mathbf{y}_j$ принимают значения {±1}. Правильным результатом в этом случае будет вектор, содержащий сначала 20 единиц, а потом 20 минус единиц

Листинг 5.12 ❖ Перекрестная проверка после ЛДА

```
for jj=1:100;
    r1=randperm(80); r2=randperm(80);
    ind1=r1(1:60); ind2=r2(1:60)+60;
```

```
    ind1t=r1(61:80); ind2t=r2(61:80)+60;

    xtrain=[v(ind1,2:2:4); v(ind2,2:2:4)];
    test=[v(ind1t,2:2:4); v(ind2t,2:2:4)];

    label=[ones(60,1); -1*ones(60,1)];
    truth=[ones(20,1); -1*ones(20,1)];
    class=classify(test,xtrain,label);
    E(jj)=sum(abs(class-truth))/40*100;
end
```

На рис. 5.20 показаны результаты перекрестной проверки по 100 испытаниям. Обратите внимание, что результаты испытаний различаются в довольно широких пределах. Иногда качество достигает 100 %, а иногда опускается до 40 %, т. е. хуже, чем классификация путем подбрасывания монеты. В главе, посвященной регрессии, мы уже говорили, что перекрестная проверка критически важна для тестирования и повышения робастности модели. Напомним, что методы построения классификатора основаны на оптимизации и регрессии, поэтому все рассмотренные выше способы перекрестной проверки переносятся на задачи кластеризации и классификации.

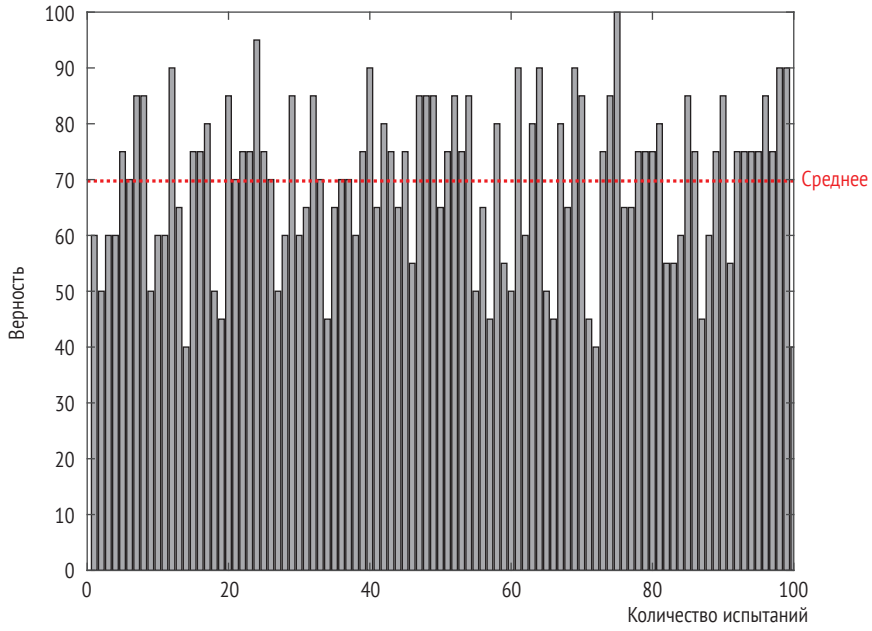

Рис. 5.20 ❖ Качество ЛДА в 100 испытаниях. Отметим изменчивость результатов, зависящую от того, какие данные выбирались для обучения и тестирования. Это подчеркивает важность перекрестной проверки для построения робастного классификатора

Для разделения данных можно найти не только линейный, но и квадратичный дискриминант. Вообще, команда MATLAB **classify** позволяет не толь-

ко построить классификатор, но и получить линию, разделяющую данные разных классов. Программа ниже порождает метки новых данных и линию, разделяющую собак и кошек.

Листинг 5.13 ❖ Построение линейного и квадратичного дискриминантов

```
subplot(2,2,1)
[class,~,~,~,coeff]=classify(test,xtrain,label);
K = coeff(1,2).const;
L = coeff(1,2).linear;
f = @(x,y) K + [x y]*L;
h2 = ezplot(f,[-.15 0.25 -.3 0.2]);
subplot(2,2,2)
[class,~,~,~,coeff]=classify(test,xtrain,label,'quadratic');
K = coeff(1,2).const;
L = coeff(1,2).linear;
Q = coeff(1,2).quadratic;
f = @(x,y) K + [x y]*L + sum(([x y]*Q) .* [x y], 2);
h2 = ezplot(f,[-.15 0.25 -.3 0.2]);
```

На рис. 5.21 показаны данные о собаках и кошках, а также разделяющие их линейный и квадратичный дискриминанты. Коэффициенты этой линейной или квадратичной функции находятся в структурной переменной **coeff**, возвращаемой командой **classify**. Квадратичная линия часто дает немного больше гибкости при описании границ классов данных. Основное достоинство методов, основанных на ЛДА, – простота интерпретации и вычислений. Поэтому они широко применяются во многих науках для классификации данных.

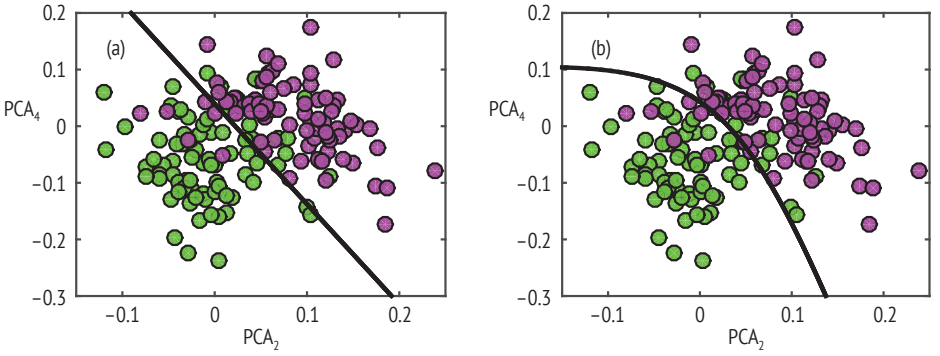

Рис. 5.21 ❖ Линия, разделяющая проекции собак и кошек на вторую и четвертую главные компоненты, в случае (a) линейного дискриминантного анализа (ЛДА) и (b) квадратичного дискриминантного анализа (КДА). Это двумерное пространство признаков позволяет хорошо разделить данные

5.7. Метод опорных векторов (SVM)

Одним из самых успешных современных методов добычи данных является *метод опорных векторов* (support vector machine – SVM). Он широко используется в науке и технике и часто дает лучшие результаты по сравнению с альтернативными методами. Наряду с алгоритмом *случайного леса* он остается одним из столпов машинного обучения на протяжении последних десятилетий. В наши дни SVM можно заменить глубокими нейронными сетями, если только имеется достаточное количество обучающих данных. Если же это не так, то SVM в сочетании со случайным лесом чаще всего применяются в приложениях, где необходима лучшая классификация из возможных.

Первоначальный алгоритм SVM был предложен Вапником и Червоненкисом в статье по статистическому обучению в 1963 году. Требовалось найти оптимальную гиперплоскость, разделяющую данные на два кластера. Спустя почти тридцать лет Бозер, Гийон и Вапник разработали нелинейные классификаторы, применив ядерный трюк для построения гиперплоскости с максимальным зазором [70]. Современная версия, ставшая стандартной (с мягким зазором), предложена Кортесом и Вапником в середине 1990-х годов [138].

Линейный SVM

Идея линейного метода опорных векторов заключается в том, чтобы построить гиперплоскость

$$\mathbf{w}\cdot\mathbf{x}+b=0, \tag{5.26}$$

параметризованную вектором $\mathbf{w}$ и скаляром b. На рис. 5.22 показаны две потенциальные гиперплоскости, разделяющие данные, – с разными $\mathbf{w}$ и b. Задача оптимизации в алгоритме SVM состоит в том, чтобы найти границу, при которой количество ошибок классификации минимально, и притом такую, чтобы зазор между данными, изображенный на рис. 5.22 серой полосой, был максимальным. Векторы, определяющие положение разделяющей границы, называются *опорными векторами*. Если известна гиперплоскость (5.26), то для классификации новой точки $\mathbf{x}_j$ нужно просто вычислить знак $(\mathbf{w}\cdot\mathbf{x}_j+b)$. Метка $\mathbf{y}_j\in\{\pm 1\}$, определяющая, лежит точка слева или справа от гиперплоскости, вычисляется по правилу

$$\mathbf{y}_j=\operatorname{sign}(\mathbf{w}\cdot\mathbf{x}_j+b)=\begin{cases}+1 & \text{сиреневая точка}\\ -1 & \text{зеленая точка}\end{cases}. \tag{5.27}$$

Таким образом, классификатор $\mathbf{y}_j$ явным образом зависит от положения точки $\mathbf{x}_j$.

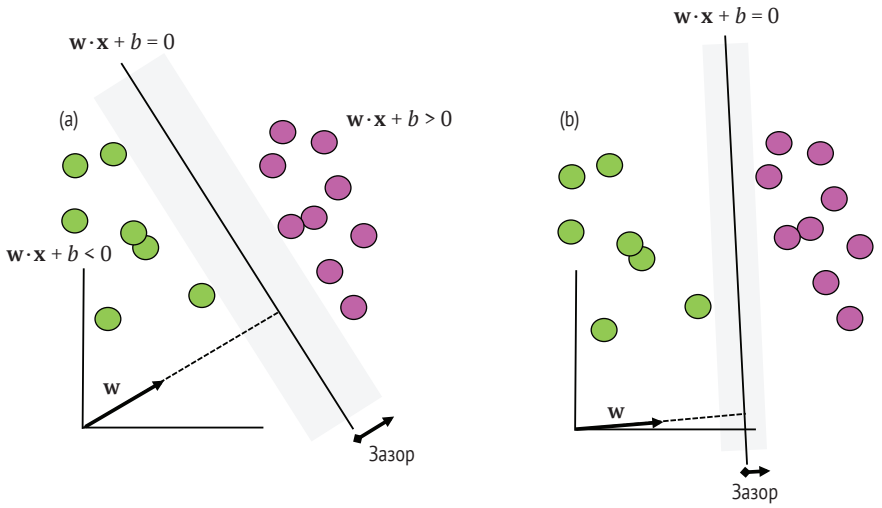

Рис. 5.22 ❖ В методе опорных векторов строится гиперплоскость $\mathbf{w}\cdot\mathbf{x} + b = 0$, оптимально разделяющая помеченные данные. Ширина зазора максимальна на рисунке (a) и гораздо меньше на рисунке (b). Цель оптимизации – найти вектор $\mathbf{w}$ и скаляр b. Заметим, что для точек справа от гиперплоскости $\mathbf{w}\cdot\mathbf{x} + b > 0$, а для точек слева от нее $\mathbf{w}\cdot\mathbf{x} + b < 0$. Поэтому классификация, т. е. вычисление меток $\mathbf{y}_j \in \{\pm 1\}$, определяющих, по какую сторону от гиперплоскости находится точка, сводится к вычислению $\mathbf{y}_j = \text{sign}(\mathbf{w}\cdot\mathbf{x}_j + b)$. Векторы, определяющие границы серой области, называются *опорными векторами*

Для успешной работы SVM нужен теоретически обоснованный способ определения $\mathbf{w}$ и b. Как и во всех методах машинного обучения, следует сформулировать задачу оптимизации. Цель оптимизации – минимизировать число неправильно классифицированных точек и при этом обеспечить максимальный зазор. Для построения целевой функции определим функцию потерь

$$\ell(\mathbf{y}_j, \bar{\mathbf{y}}_j) = \ell(\mathbf{y}_j, \text{sign}(\mathbf{w}\cdot\mathbf{x}_j + b)) = \begin{cases} 0, & \text{если } \mathbf{y}_j = \text{sign}(\mathbf{w}\cdot\mathbf{x}_j + b) \\ +1, & \text{если } \mathbf{y}_j \neq \text{sign}(\mathbf{w}\cdot\mathbf{x}_j + b) \end{cases}. \tag{5.28}$$

Проще говоря,

$$\ell(\mathbf{y}_j, \bar{\mathbf{y}}_j) = \begin{cases} 0, & \text{если точка помечена правильно} \\ +1, & \text{если точка помечена неправильно} \end{cases}. \tag{5.29}$$

Таким образом, каждая неправильно помеченная точка увеличивает потерю на 1. Ошибка обучения на m точках равна сумме потерь $\ell(\mathbf{y}_j, \bar{\mathbf{y}}_j)$.

Кроме минимизации функции потерь, наша цель – добиться максимального зазора. Поэтому задача оптимизации в линейном методе опорных векторов ставится так:

$$\underset{\mathbf{w},b}{\text{argmin}} \sum_{j=1}^{m} \ell(\mathbf{y}_j, \bar{\mathbf{y}}_j) + \frac{1}{2}||\mathbf{w}||^2 \quad \text{при условии} \quad \min_j |\mathbf{x}_j \cdot \mathbf{w}| = 1. \tag{5.30}$$

Несмотря на лаконичную постановку задачи оптимизации, решить ее очень трудно, потому что функция потерь дискретна. Обычно задачи оптимизации решаются с помощью какого-то варианта градиентного спуска, но для этого нужно, чтобы целевая функция была гладкой, иначе невозможно вычислить градиенты и обновлять решение на каждой итерации. Поэтому чаще встречается такая постановка:

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{j=1}^{m} H(\mathbf{y}_j, \overline{\mathbf{y}}_j) + \frac{1}{2}||\mathbf{w}||^2 \quad \text{при условии} \quad \min_j |\mathbf{x}_j \cdot \mathbf{w}| = 1, \tag{5.31}$$

где α – вес функции потерь, а $H(z) = \max(0, 1 - z)$ – кусочно-линейная функция потерь (hinge loss function). Эта кусочно-гладкая функция подсчитывает количество ошибок и допускает дифференцирование, поэтому становятся применимы стандартные методы оптимизации.

Нелинейный SVM

Несмотря на простоту интерпретации, ценность линейных классификаторов ограничена. Они попросту не способны разделить данные в пространстве высокой размерности, которые могут быть устроены, как показано на рис. 5.8. Чтобы построить более сложные разделяющие границы, необходимо обогатить пространство признаков для SVM. Это делается путем включения нелинейных признаков и построения гиперплоскости в этом новом пространстве. Для этого мы отображаем данные в нелинейное многомерное пространство

$$\mathbf{x} \mapsto \mathbf{\Phi}(\mathbf{x}). \tag{5.32}$$

$\mathbf{\Phi}(\mathbf{x})$ можно назвать новой *наблюдаемой величиной*. Теперь алгоритм SVM находит гиперплоскость, оптимально разделяющую данные на кластеры в новом пространстве, т. е. рассматривается функция

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}) + b \tag{5.33}$$

и соответствующие метки $\mathbf{y}_j \in \{\pm 1\}$ для каждой точки $f(\mathbf{x}_j)$.

Эта нехитрая идея обогащения пространства признаков путем определения новых функций $\mathbf{x}$ оказалась исключительно мощной и полезной для кластеризации и классификации. В качестве простого примера рассмотрим двумерные данные $\mathbf{x} = (x_1, x_2)$. Для обогащения пространства можно рассмотреть полиномиальные функции.

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1, x_2, x_1^2 + x_2^2). \tag{5.34}$$

Это дает новый набор координат, в которых можно представить данные. И понятно, зачем это делается: ведь в пространстве более высокой размерности проще найти гиперплоскость, разделяющую данные. Рассмотрим, к примеру, данные на рис. 5.8 (b). Очевидно, что никакой линейный классификатор (гиперплоскость) в плоскости x_1–x_2 не может разделить данные.

Но погружение (5.34) отображает данные в трехмерное пространство, где их легко разделить гиперплоскостью, как показано на рис. 5.23.

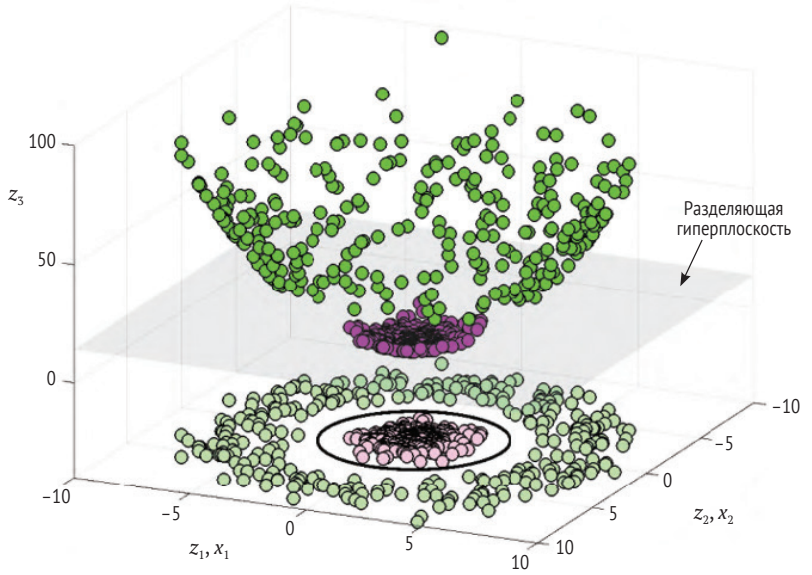

Рис. 5.23 ❖ Нелинейное погружение рис. 5.8 (b) с помощью преобразования $(x_1, x_2) \mapsto (z_1, z_2, z_3) \coloneqq (z_1, z_2, z_3) \coloneqq (x_1, x_2, x_1^2 + x_2^2)$. Теперь легко провести гиперплоскость, разделяющую зеленые и сиреневые точки, т. е. для выполнения линейной классификации нужно просто обогатить пространство измерений. Даже на глаз видно, что почти оптимальное разбиение дает плоскость $z_3 \approx 14$ (показана серым цветом). В исходной системе координат ей соответствует разделяющая окружность (показана черным цветом на плоскости $x_1 - x_2$) радиусом $r = \sqrt{z_3} = \sqrt{x_1^2 + x_2^2} \approx \sqrt{14}$. Из этого примера очевидно, как гиперплоскость в пространстве большей размерности может порождать криволинейные границы в исходном пространстве данных

Способность SVM к нелинейному погружению в пространство большей размерности делает его одним из самых успешных алгоритмов машинного обучения. Сам алгоритм оптимизации (5.31) остается неизменным, только прежняя функция пометки $\bar{\mathbf{y}}_j = \text{sign}(\mathbf{w} \cdot \mathbf{x}_j + \text{b})$ теперь заменяется такой:

$$\bar{\mathbf{y}}_j = \text{sign}(\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}_j) + b). \tag{5.35}$$

Функция $\mathbf{\Phi}(\mathbf{x})$ определяет обогащенное пространство наблюдаемых величин. Как правило, чем больше признаков, тем лучше классификация.

Ядерные методы в сочетании с SVM

Вопреки обещанию, метод SVM построения линейных классификаторов путем погружения в пространство большей размерности приводит к вычисли-

тельно нереализуемой оптимизации. Дело в том, что большое число дополнительных признаков ведет к *проклятию размерности*. Вычисление вектора **w** обходится слишком дорого, и возможно даже, что не хватит памяти. Эту проблему решает *ядерный трюк*. Вектор **w** представляется в виде

$$\mathbf{w} = \sum_{j=1}^{m} \alpha_j \mathbf{\Phi}(\mathbf{x}_j), \tag{5.36}$$

где параметры α_j – веса нелинейных наблюдаемых функций $\mathbf{\Phi}(\mathbf{x}_j)$. Таким образом, вектор **w** разлагается по наблюдаемым функциям. Тогда (5.33) можно обобщить следующим образом:

$$f(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j \mathbf{\Phi}(\mathbf{x}_j) \cdot \mathbf{\Phi}(\mathbf{x}) + b. \tag{5.37}$$

Теперь определим *ядерную функцию* [479]:

$$K(\mathbf{x}_j, \mathbf{x}) = \mathbf{\Phi}(\mathbf{x}_j) \cdot \mathbf{\Phi}(\mathbf{x}). \tag{5.38}$$

С новым определением **w** задача оптимизации (5.31) принимает вид

$$\underset{\boldsymbol{\alpha}, b}{\operatorname{argmin}} \sum_{j=1}^{m} H(\mathbf{y}_j, \bar{\mathbf{y}}_j) + \frac{1}{2} \left\| \sum_{j=1}^{m} \alpha_j \mathbf{\Phi}(\mathbf{x}_j) \right\|^2 \quad \text{при условии} \quad \min_j |\mathbf{x}_j \cdot \mathbf{w}| = 1, \tag{5.39}$$

где **α** – вектор коэффициентов α_j, который следует определить в процессе решения. Есть разные соглашения по поводу представления задачи минимизации. Но в описанной постановке минимизации подлежит **α** вместо **w**.

Ядерная функция $K(\mathbf{x}_j, \mathbf{x})$ позволяет компактно представить разложение в ряд Тейлора функции от большого (бесконечного) числа наблюдаемых переменных [479]. Благодаря ядерной функции мы можем работать в пространстве признаков высокой размерности, не вычисляя координат точек в нем, а только скалярные произведения между парами точек. Приведем два примера наиболее употребительных ядерных функций:

радиально-базисная функция (РБФ): $K(\mathbf{x}_j, \mathbf{x}) = \exp(-\gamma ||\mathbf{x}_j - \mathbf{x}||^2)$; (5.40a)

полиномиальное ядро: $K(\mathbf{x}_j, \mathbf{x}) = (\mathbf{x}_j \cdot \mathbf{x} + 1)^N$, (5.40b)

где N – степень полинома, слишком большая для вычислений без использования ядерного трюка, а γ – ширина гауссова ядра, измеряющая расстояние между отдельными точками $\mathbf{x}_j$ и разделяющей границей. Эти функции дифференцируемы и потому могут использоваться в решении задачи оптимизации (5.39).

Это и есть теоретический фундамент метода SVM, позволяющий конструировать пространства более высокой размерности с использованием наблюдаемых величин, сгенерированных ядерной функцией. К тому же вычисления в алгоритме оптимизации обозримы и реализуемы практически. В следующей программе иллюстрируется принцип работы ядерной функции на примере классификации собак и кошек. Сначала используется стандартный линейный SVM, а затем SVM с радиально-базисным ядром.

Листинг 5.14 ❖ Классификация методом SVM

```
load catData_w.mat
load dogData_w.mat
CD=[dog_wave cat_wave];
[u,s,v]=svd(CD-mean(CD(:)));

features=1:20;
xtrain=[v(1:60,features); v(81:140,features)];
label=[ones(60,1); -1*ones(60,1)];
test=[v(61:80,features); v(141:160,features)];
truth=[ones(20,1); -1*ones(20,1)];

Mdl = fitcsvm(xtrain,label);
test_labels = predict(Mdl,test);

Mdl = fitcsvm(xtrain,label,'KernelFunction','RBF');
test_labels = predict(Mdl,test);
CMdl = crossval(Mdl);       % перекрестная проверка модели
classLoss = kfoldLoss(CMdl) % вычислить потерю
```

Отметим, что здесь продемонстрированы диагностические средства реализации SVM в MATLAB, в т. ч. перекрестная проверка и вычисление функции потерь при обучении. Наше изложение SVM было весьма поверхностным. Вообще говоря, SVM – один из самых продвинутых инструментов машинного обучения в MATLAB, и существует много средств для настройки качества, перекрестной проверки и получения метрик верности.

5.8. Решающие деревья и случайные леса

Решающие деревья очень часто встречаются в бизнесе. Они определяют блок-схему алгоритма принятия решений на основе критериев, которые считаются значимыми и напрямую влияющими на желаемый результат. Часто решающие деревья строятся экспертами в предметной области. *Обучение решающего дерева* – систематический метод построения прогностической модели для классификации или регрессии, основанный на данных. Наряду с SVM деревья классификации и регрессии принадлежит к числу основных алгоритмов машинного обучения и добычи данных с доказанной эффективностью. В работе Лео Бреймана с сотрудниками [79] заложены теоретические основания решающих деревьев, используемые и по сей день.

Решающее дерево – это иерархическая структура, цель которой – найти оптимальные для робастной классификации и регрессии способы разделения данных. Это противоположность ранее описанной дендрограмме иерархической кластеризации, которая строится без учителя. В данном случае мы не поднимаемся вверх в процессе кластеризации, а спускаемся вниз, стремясь на каждом шаге создать наилучшее разделение данных. Это алгоритм обучения с учителем, т. е. данные предварительно помечены, и это учитывается при разделении.

У решающих деревьев для классификации и регрессии есть ряд важных преимуществ: (i) результаты допускают интерпретацию даже неспециалистом и могут быть отображены графически; (ii) они одинаково хорошо работают с числовыми и категориальными данными; (iii) они допускают статистическую проверку, т. е. надежность модели можно оценить; (iv) они хорошо масштабируются на большие данные; (v) алгоритм моделирует принятие решений человеком, что делает его еще более интерпретируемым и полезным.

Как и следовало ожидать, успехи в обучении решающих деревьев стали причиной появления большого числа инноваций и алгоритмов наилучшего разделения данных. Мы не сможем здесь осветить эту тему подробно, но расскажем о базовых принципах разделения данных и построения дерева. Напомним, что у нас есть:

$$\text{данные } \{\mathbf{x}_j \in \mathbb{R}^n,\ j \in Z := \{1, 2, \cdots, m\}\}; \tag{5.41a}$$

$$\text{метки } \{\mathbf{y}_j \in \{\pm 1\},\ j \in Z' \subset Z\}. \tag{5.41b}$$

Принципиально алгоритм построения решающего дерева прост. (i) Просмотреть все элементы x_k ($k = 1, 2, \cdots, n$) (признаки) вектора $\mathbf{x}_j$ и найти такой x_j, при котором предсказательная способность $\mathbf{y}_j$ максимальна. (ii) Сравнить точность предсказания для разделения по каждому признаку x_j. Выбрать признак, дающий наилучшее разделение дерева. (iii) Повторить процесс для обеих вновь созданных ветвей дерева. Алгоритм заканчивает работу, когда в каждом кластере останется одна точка, называемая *листом* дерева. По существу, алгоритм противоположен построению дендрограммы.

В качестве конкретного примера рассмотрим набор данных Фишера об ирисах на рис. 5.1. Каждый цветок характеризуется четырьмя признаками (ширина и длина чашелистика и ширина и длина лепестка) и тремя метками (ирис щетинистый, разноцветный или виргинский). Всего имеется по 50 примеров каждого вида, т. е. 150 точек. Таким образом, вектор $\mathbf{x}_j$ для этих данных содержит четыре элемента:

$$x_1 = \text{ширина чашелистика}; \tag{5.42a}$$

$$x_2 = \text{длина чашелистика}; \tag{5.42b}$$

$$x_3 = \text{ширина лепестка}; \tag{5.42c}$$

$$x_4 = \text{длина лепестка}. \tag{5.42d}$$

Алгоритм построения решающего дерева перебирает эти четыре признака, пытаясь решить, как лучше всего разделить данные. На рис. 5.24 показан процесс разделения в пространстве четырех переменных $x_1, \cdots, x_4$. Представлены только две плоскости: x_1–x_2 (рисунок (b)) и x_3–x_4 (рисунок (a)). Видно, что наилучшее разделение дает признак x_3 (длина лепестка). И действительно, первое разделение данных производится по значению $x_3 = 2.35$. Для предсказания ириса щетинистого больше никаких разделений и не нужно. Следующим наиболее перспективным является разделение по значению $x_4 = 1.75$. Наконец, третьим выполняется разделение по $x_3 = 4.95$. Показано только три разделения. Как видим, процедура разделения интуитивно понятна, потому что точки оптимального разделения видны невооруженным глазом. Кроме

того, по признакам x_1 и x_2 (ширина и длина чашелистика) разделение не производится, поскольку это было бы малоэффективно. На рис. 5.25 показано дерево, построенное процессом на рис. 5.24.

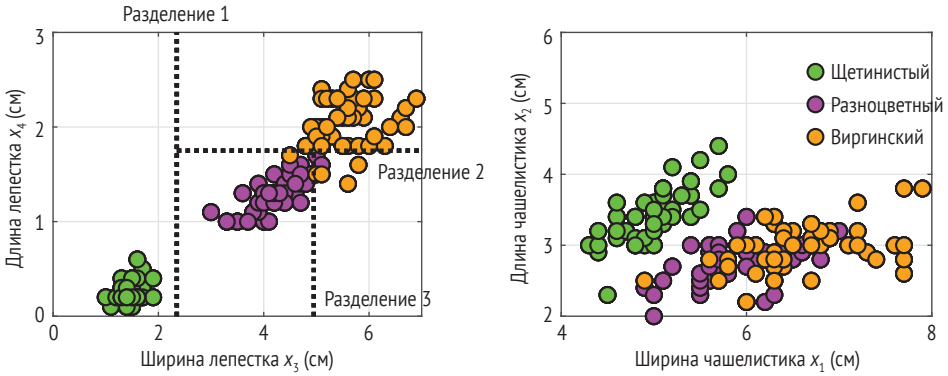

Рис. 5.24 ❖ Процедура разделения при обучении решающего дерева на наборе данных Фишера об ирисах. Просматривается каждый из признаков x_1, ..., x_4, чтобы найти наилучшее разделение, дающее правильную классификацию помеченных данных. Первое разделение производится по значению $x_3 = 2.35$. Второе разделение производится по $x_4 = 1.75$, а третье – по $x_3 = 4.95$. Показано только три разделения. Решающее дерево после трех разделений представлено на рис. 5.25. Заметим, что хотя данные об ирисе щетинистом на плоскости x_1–x_2 лучше всего разделяются по диагонали, решающее дерево может разделять данные только по горизонтали или по вертикали

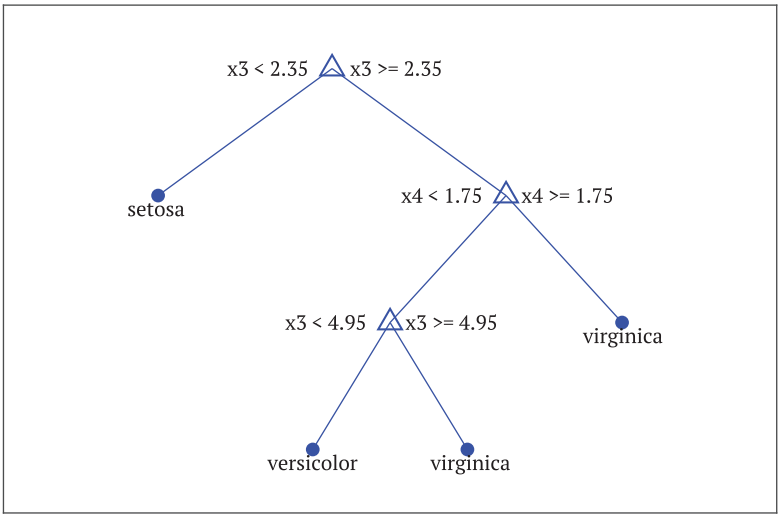

Рис. 5.25 ❖ Дерево, построенное командой MATLAB **fitctree**. Заметим, что произведено только три разделения, и в результате получилось дерево, которое дает ошибку классификации 4.67 %

Следующая программа строит дерево для набора данных Фишера. Заметим, что у команды **fitctree** много параметров, в т. ч. признак выполнения перекрестной проверки (задан в коде) и настройка параметров (в коде не используется).

Листинг 5.15 ❖ Построение решающего дерева для классификации данных Фишера об ирисах

```
load fisheriris;
tree=fitctree(meas,species,'MaxNumSplits',3,'CrossVal','on')
view(tree.Trained{1},'Mode','graph');
classError = kfoldLoss(tree)

x1=meas(1:50,:);    % щетинистый
x2=meas(51:100,:);  % разноцветный
x3=meas(101:150,:); % виргинский
```

Результаты процедуры разделения приведены на рис. 5.25. Команда **view** создает окно, в котором показана структура дерева. Дерево можно подвергать обрезке и интерактивно выполнять с ним другие действия. Ошибка классификации на наборе данных Фишера составляет 4.67 %.

В качестве второго примера построим решающее дерево для классификации собак и кошек по ранее сгенерированным вейвлет-преобразованиям изображений. Следующая программа загружает и разделяет данные.

Листинг 5.16 ❖ Построение решающего дерева для классификации данных о собаках и кошках

```
load catData_w.mat
load dogData_w.mat
CD=[dog_wave cat_wave];
[u,s,v]=svd(CD-mean(CD(:)));

features=1:20;
xtrain=[v(1:60,features); v(81:140,features)];
label=[ones(60,1); -1*ones(60,1)];
test=[v(61:80,features); v(141:160,features)];
truth=[ones(20,1); -1*ones(20,1)];

Mdl = fitctree(xtrain,label,'MaxNumSplits',2,'CrossVal','on');
classError = kfoldLoss(Mdl)
view(Mdl.Trained{1},'Mode','graph');
classError = kfoldLoss(Mdl)
```

На рис. 5.26 показано получившееся решающее дерево. Заметим, что алгоритм его обучения определил, что первые два разбиения производятся по признакам x_2 и x_4 соответственно. Эти два признака уже рассматривались выше, потому что гистограммы показывают, что они обеспечивают лучшее различение данных, чем другие компоненты PCA (см. рис. 5.5). Это разделение было подвергнуто перекрестной проверке, и оказалось, что ошибка клас-

сификации составляет примерно 16 %, что неплохо по сравнению с ошибкой 30 % в методе ЛДА.

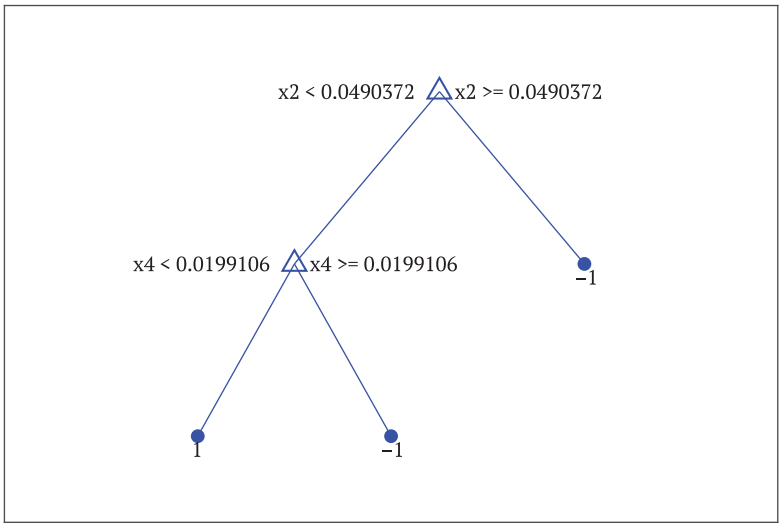

Рис. 5.26 ❖ Дерево, построенное командой MATLAB **fitctree** для классификации собак и кошек. Заметим, что произведено только два разделения, и в результате получилось дерево, которое дает ошибку классификации приблизительно 16 %

И напоследок рассмотрим данные переписи, включенные в MATLAB. Программа в листинге 5.17 демонстрирует некоторые важные особенности применения решающих деревьев для классификации и регрессии. В частности, включенные признаки можно использовать для создания ассоциаций между связями. В данном случае для предсказания величины заработной платы используются различные данные. Можно также вычислить, насколько важен каждый признак для величины зарплаты (см. рис. 5.27).

Листинг 5.17 ❖ Построение решающего дерева для данных переписи

```
load census1994
X = adultdata(:,{'age','workClass','education_num',
    'marital_status','race','sex','capital_gain',...
    'capital_loss','hours_per_week','salary'});

Mdl = fitctree(X,'salary','PredictorSelection','curvature',
    'Surrogate','on');

imp = predictorImportance(Mdl);

bar(imp,'FaceColor',[.6 .6 .6],'EdgeColor','k');
title('Predictor Importance Estimates');
ylabel('Estimates'); xlabel('Predictors'); h = gca;
h.XTickLabel = Mdl.PredictorNames;
h.XTickLabelRotation = 45;
```

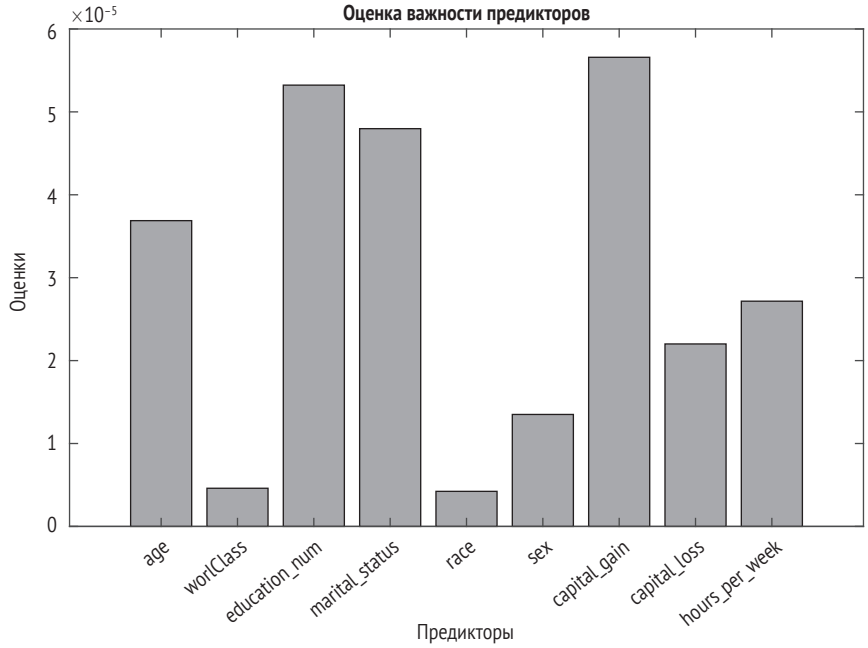

Рис. 5.27 ❖ Относительная важность признаков для предсказания величины заработной платы по результатам переписи США 1994 года. Архитектура решающих деревьев позволяет проводить нетривиальный анализ данных, в т. ч. оценивать статистическое влияние каждого признака на результат классификации

Как и для алгоритма SVM, имеется много настроечных параметров, а наше изложение поверхностно. В общем и целом решающие деревья – один из самых продвинутых инструментов машинного обучения в MATLAB, и существует много средств для настройки качества, перекрестной проверки и получения метрик верности.

Случайные леса

Прежде чем закончить этот раздел, упомянем о *случайных лесах* Бреймана [77] как способе усовершенствовать решающие деревья. Случайный лес – это ансамблевый метод обучения для классификации и регрессии. Это важное достижение, потому что решающие деревья, созданные в результате разделения данных, обычно недостаточно робастны по отношению к различным выборкам данных. Поэтому, взяв две выборки, можно получить два совершенно разных дерева классификации. Это создает серьезные проблемы для перекрестной проверки. В процессе ансамблевого обучения создается несколько решающих деревьев. Случайный лес исправляет склонность решающих деревьев к переобучению и, следовательно, является более робастной системой классификации.

Существует много вариантов алгоритма случайных лесов, в т. ч. *усиление* (*бустинг*) и *бэггинг*. Мы не будем их рассматривать, а только упомянем, что

в команде MATLAB **figctree** многие из них можно задать с помощью параметров. На ансамблевое обучение можно смотреть как на способ создания робастных решающих деревьев. При этом зачастую алгоритм уделяет больше внимания трудным для классификации данным и меньше – простым для классификации. Случайные леса, бэггинг и усиление – обширные самостоятельные темы, но они уже включены в основные программы, умеющие строить решающие деревья.

5.9. 10 лучших алгоритмов по версии Data Mining 2008

В этом разделе мы продемонстрировали колоссальное разнообразие методов обучения с учителем и без учителя, применяемых для анализа данных. Хотя эти алгоритмы сегодня реализованы во многих пакетах коммерческих и свободных программ, трудность в том, чтобы понять, какой метод или методы следует использовать в данной задаче. В декабре 2006 года специалисты по машинному обучению, собравшиеся на Международной конференции IEEE по добыче данных (ICDM), назвали 10 лучших алгоритмов для добычи данных [562], а именно: C4.5, *k* средних, SVM, Apriori, EM, PageRank, AdaBoost, kNN, наивный байесовский классификатор и CART. В то время они считались самыми авторитетными в исследовательских кругах. В обзорной статье был кратко описан каждый алгоритм, его влияние и потенциальные пути развития. Эти 10 алгоритмов относились к классификации, кластеризации, статистическому обучению, ассоциативному анализу и анализу связей – наиболее важным направлениям исследований и разработок в области добычи данных. Интересно, что глубокое обучение и нейронные сети – тема следующей главы – в статье не были упомянуты. Ландшафт науки о данных сильно изменился в 2012 году с появлением набора данных ImageNET, с тех пор глубокие нейронные сети стали по точности превосходить едва ли не все остальные методы классификации и регрессии.

В этом разделе мы приведем краткое описание 10 лучших алгоритмов и реализующих их команд MATLAB. Многие из них были рассмотрены выше в этой главе. Список не является исчерпывающим, и мы перечисляем их в том порядке, в каком они были включены в исходный список, не пытаясь ранжировать дополнительно. Наша цель – просто рассказать, какие инструменты добычи данных считались сообществом наиболее актуальными и перспективными в 2008 году. Начнем с алгоритмов, которые мы уже рассматривали.

Алгоритм *k* средних

Один из основных алгоритмов обучения без учителя. Как уже было сказано, его цель – выделить во множестве *k* точек кластеры, исходя из близости. Пересчитывая принадлежность точек к кластерам на основе расстояния до ближайшего среднего, алгоритм итеративно находит кластеры. Реализует алгоритм команда MATLAB

```
[labels,centers]=kmeans(X,k)
```

Команда **means** принимает данные **X** и предполагаемое количество кластеров *k*, возвращает метки для каждой точки **labels** и положение центров кластеров **centers**.

EM-алгоритм (смесовые модели)

Смесовые модели – второй из наиболее употребительных алгоритмов обучения без учителя. В его основе лежит предположение о том, что данные порождаются смесью функций распределения вероятностей с неизвестными весами. Параметры оцениваются с помощью EM-алгоритма (математическое ожидание–максимизация). Его реализует команда MATLAB

```
Model=fitgmdist(X,k)
```

где **fitgmdist** по умолчанию аппроксимирует *k* кластеров в данных **X** смесью гауссовых распределений. Возвращаемая структурная переменная **Model** содержит информацию о распределениях вероятностей (среднее, дисперсию и т. д.), а также точность приближения.

Метод опорных векторов (SVM)

Один из самых мощных и гибких алгоритмов обучения с учителем на протяжении 1990-х и 2000-х годов, SVM прекрасно справляется с классификацией и регрессией. Его основная идея – отобразить данные в пространство большей размерности и там разделить их гиперплоскостью. Практическое применение этой идеи неразрывно связано с ядерным трюком, который позволяет эффективно вычислять скалярные произведения в пространстве более высокой размерности. В MATLAB реализуется командой

```
Model = fitcsvm(xtrain,label);
test_labels = predict(Model,test);
```

Команда **fitcsvm** принимает помеченные обучающие данные в виде массивов **train** и **label** и возвращает структурную переменную **Model**. Эту переменную можно передать команде **predict** вместе с тестовыми данными **test** и получить в ответ метки (**test_labels**). У команды **fitcsvm** много настроечных и других параметров, что делает ее одной из лучших готовых реализаций метода SVM.

CART (Classification and Regression Tree – дерево классификации и регрессии)

Этот алгоритм мы обсуждали в предыдущем разделе, где показали, что это еще один эффективный метод обучения с учителем. Идея в том, чтобы систематически подойти к разделению данных и в результате получить до-

пускающее интерпретацию разбиение на кластеры. Алгоритм реализован в виде итеративной процедуры, на каждом шаге которой выбирается, по какой переменной производить разделение. В MATLAB представлен командой

```
tree = fitctree(xtrain,label);
```

которая принимает помеченные обучающие данные **train** и **label** и возвращает структурную переменную **tree**. У команды **fitctree** много настроечных и других параметров, что делает ее одной из лучших готовых реализаций решающих деревьев.

Метод *k* ближайших соседей (kNN)

Пожалуй, один из самых простых для понимания и выполнения алгоритмов обучения с учителем, результаты которого легко интерпретировать. Получив непомеченную точку $\mathbf{x}_k$, алгоритм находит k ближайших к ней соседей $\mathbf{x}_j$ с метками $\mathbf{y}_j$. Новая точка получает ту метку, которой помечено большинство соседей. В MATLAB реализован командой

```
label = knnsearch(Mdl,test)
```

которая использует помеченные данные **Mdl** для пометки тестовых данных **test**.

Наивная байесовская классификация

Наивный байесовский классификатор – интуитивно понятный метод обучения с учителем. Он не требует сложной оценки параметров, как SVM или решающие деревья. При этом на практике он дает великолепные и прекрасно интерпретируемые результаты. Метод основан на формуле Байеса и вычислении условных вероятностей. Метку новой точки можно оценить, зная априорное распределение вероятностей помеченных данных. В MATLAB наивный байесовский классификатор реализован командой

```
Model = fitNaiveBayes(xtrain,label)
```

принимающей помеченные обучающие данные **train** и **label** и возвращающей структурную переменную **Model**, которую можно передать команде **predict** для пометки тестовых данных **test**.

AdaBoost (ансамблевое обучение с усилением)

AdaBoost – пример алгоритма ансамблевого обучения [188]. Грубо говоря, это вариант случайного леса [77], в котором рассматривается ансамбль решающих деревьев. Как и все алгоритмы усиления, AdaBoost сначала назначает всем обучающим данным $\mathbf{x}_j$ одинаковые веса. В процессе усиления важность данных изменяется в зависимости от того, насколько трудно их классифици-

ровать. Таким образом, алгоритм уделяет особое внимание данным, классификация которых затруднена. Поэтому семейство слабых обучаемых можно обучить так, что в результате получится более сильный обучаемый [470]. Теоретическое обоснование эта идея получила в основополагающей работе Kearns and Valiant [283]. В MATLAB алгоритм реализован командой

```
ada = fitcensemble(xtrain,label,'Method','AdaBoostM1')
```

которая, помимо AdaBoost, умеет выполнять и другие алгоритмы ансамблевого обучения, в т. ч. робастное и градиентное усиления. Градиентное усиление – один из самых мощных методов [189].

C4.5 (ансамблевое обучение решающих деревьев)

Этот алгоритм – еще один вариант обучения решающих деревьев, предложенный в работах J. R. Quinlan [443, 444]. Разделение данных производится согласно оценке информационной энтропии. Последние версии поддерживают усиление и многие другие известные приемы повышения качества. В общем, можно считать, что это усовершенствованный вариант алгоритма CART. Команда **fitcensemble**, упомянутая при рассмотрении AdaBoost, реализует общую архитектуру ансамблевого обучения, включающую и различные алгоритмы обучения решающих деревьев, в т. ч. C4.5.

Алгоритм Apriori

Последние два метода акцентируют внимание на специальных вопросах машинного обучения. Цель алгоритма Apriori – найти часто встречающиеся в данных предметные наборы. Задача кажется тривиальной, но на деле таковой не является, потому что набор данных может быть очень большим, а ввиду комбинаторной природы алгоритмов вычисления оказываются NP-трудными. Алгоритм Apriori предлагает эффективный способ поиска частых предметных наборов, основанный на архитектуре генерирования кандидатов [4]. Его можно использовать для быстрого нахождения ассоциативных правил в данных.

PageRank

Компания Google, основанная Сергеем Брином и Ларри Пейджем, начиналась с алгоритма PageRank [82]. Этот алгоритм производит статическое ранжирование переменных, например веб-страниц. Значения переменных, не зависящие от поисковых запросов, вычисляются в автономном режиме. Алгоритм PageRank связывают с теорией графов, поскольку первоначально гиперссылка с одной страницы на другую интерпретировалась как поданный голос. Исходя из этого, можно вычислить оценку важности каждой переменной и построить

упорядоченный список. Впоследствии алгоритм многократно модернизировался. Точное ранжирование переменных (веб-страниц) по важности остается темой активных исследований.

Рекомендуемая литература

Учебники

(1) **Machine learning: a probabilistic perspective**, by K. P. Murphy, 2012 [396].
(2) **Pattern recognition and machine learning**, by C. M. Bishop, 2006 [64].
(3) **Pattern classification**, by R. O. Duda, P. E. Hart, and D. G. Stork, 2000 [161].
(4) **An introduction to statistical learning**, by G. James, D. Witten, T. Hastie and R. Tibshirani, 2013 [264].
(5) **Learning with kernels: support vector machines, regularization, optimization, and beyond**, by B. Schölkopf and A. J. Smola, 2002 [479].
(6) **Classification and regression trees**, by L. Breiman, J. Friedman, C. J. Stone and R. A. Olshen, 1984 [79].
(7) **Random forests**, by L. Breiman, 2001 [77].

Статьи и обзоры

(1) **Top 10 algorithms in data mining**, by X. Wu et al., *Knowledge and information systems*, 2008 [562].
(2) **The strength of weak learnability**, by R. E. Schapire, *Machine Learning*, 1990 [470].
(3) **Greedy function approximation: a gradient boosting machine**, by J. H. Friedman, *Annals of Statistics*, 2001 [189].

Глава 6

Нейронные сети и глубокое обучение

Стимулом для разработки нейронных сетей (НС) стала работа лауреатов Нобелевской премии Хьюбела и Визеля по первичной зрительной коре головного мозга кошек [259]. Их первопроходческие эксперименты показали, что нейронные сети организованы как слои клеток, обрабатывающих визуальные стимулы. Первая математическая модель НС, названная неокогнитроном, была разработана в 1980 году [193] и обладала многими характеристиками современных глубоких сверточных нейронных сетей (СНС), в том числе многослойная структура, свертка, max-пулинг и нелинейные динамические блоки. Недавние успехи, сопутствовавшие применению СНС в компьютерном зрении, стали возможны благодаря двум факторам: (i) беспрестанный рост вычислительной мощности и (ii) появление чрезвычайно больших помеченных наборов данных, что позволяет задействовать всю мощь *глубокой* многослойной архитектуры. Действительно, хотя теоретическому обоснованию НС исполнилось уже почти сорок лет, лишь анализ набора данных ImageNet в 2012 году [310] стал переломным событием в истории НС и глубокого обучения [324]. До этого существовало несколько наборов данных, содержащих порядка десяти тысяч помеченных изображений. В ImageNet было свыше 15 млн помеченных изображений высокого разрешения, разбитых на более чем 22 000 категорий. СНС (а это лишь один из многих типов нейронных сетей) с тех пор преобразили область компьютерного зрения, заняв доминирующее положение почти во всех значимых задачах классификации и идентификации изображений.

Хотя ImageNet оказала решающее влияние на отрасль, еще в начале 1990-х годов НС уделялось внимание в учебниках, правда, количество слоев обычно было невелико. Было показано, что важнейшие методы машинного обучения, например метод главных компонент (PCA), тесно связаны с сетями, включавшими обратное распространение. Важно, что были сделаны открытия, благодаря которым многослойные сети с прямым распространением выделились в отдельный класс универсальных аппроксиматоров [255]. За последние пять лет мы стали свидетелями колоссального прогресса в архитектурах НС, многие из них были специально разработаны для конкретных предметных областей. Причиной прогресса стали достижения в области соз-

дания алгоритмов, которые позволили резко повысить производительность в самых разных областях. К ним относятся предварительное обучение, прореживание, объединяющий модуль (inception module), пополнение данных виртуальными примерами, пакетная нормировка и остаточное обучение (см. детальное описание НС в книге [216]). Это лишь неполный перечень алгоритмических инноваций, из которого, однако, видно, что в отрасли происходит непрерывный и быстрый прогресс. Примечательно, что НС даже не были включены в список 10 лучших алгоритмов добычи данных, опубликованный в 2008 году [562]. Но спустя всего десять лет неоспоримый и растущий перечень достижений при анализе трудных наборов данных поставил эту технологию на первое место среди инструментов добычи данных, которыми должно овладеть новое поколение ученых и инженеров.

В двух предыдущих главах было показано, что по сути своей машинное обучение сводится к оптимизации. В случае НС целью является оптимизация композиции функций

$$\underset{\mathbf{A}_j}{\operatorname{argmin}}(f_M,(\mathbf{A}_M,\cdots,f_2(\mathbf{A}_2,f_1(\mathbf{A}_1,\mathbf{x}))\cdots)+\lambda g(\mathbf{A}_j)). \tag{6.1}$$

Эта задача часто решается с помощью стохастического градиентного спуска и обратного распространения. Каждая матрица $\mathbf{A}_k$ содержит веса связей между k-м и $(k+1)$-м слоями нейронной сети. Эта сильно недоопределенная система регуляризируется функцией $g(\mathbf{A}_j)$. Композиция и регуляризация отвечают за выразительность представления данных и предотвращение переобучения соответственно. Эта общая задача оптимизации является основой глубокого обучения, и в данной главе мы будем рассматривать, как она решается. Важно отметить, что НС весьма склонны к переобучению, поэтому нужно тщательно продумывать порядок перекрестной проверки. Напомним, что *кто пренебрегает перекрестной проверкой, тот дурак*.

6.1. Нейронные сети: однослойные сети

Общая архитектура многослойной нейронной сети показана на рис. 6.1. В задачах классификации цель НС – правильно отобразить входные данные $\mathbf{x}_j$ в метки $\mathbf{y}_j$. Множеством входных данных на рис. 6.1 является $\mathbb{R}^n$, а размерность выходного слоя определяется задачей классификации. Построение выходного слоя мы будем обсуждать ниже.

Сразу понятно, что в связи с проектированием НС возникает масса вопросов. Сколько должно быть слоев? Каковы должны быть размерности слоев? Как проектировать выходной слой? Каким должно быть отображение между слоями: *линейным* или *нелинейным*? Как и в случае настройки SVM и решающих деревьев, НС имеет много параметров, которые можно настраивать для повышения производительности.

Для начала рассмотрим отображение между слоями на рис. 6.1. Мы обозначаем промежуточные слои $\mathbf{x}^{(k)}$, где k – номер слоя. Если отображение между слоями линейно, то имеют место соотношения

$$\mathbf{x}^{(1)} = \mathbf{A}_1\mathbf{x}; \tag{6.2a}$$

$$\mathbf{x}^{(2)} = \mathbf{A}_2\mathbf{x}^{(1)}; \tag{6.2b}$$

$$\mathbf{y} = \mathbf{A}_3\mathbf{x}^{(2)}. \tag{6.2c}$$

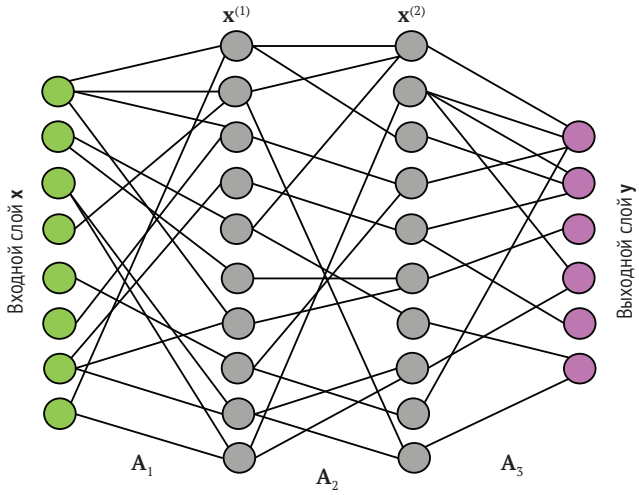

Рис. 6.1 ❖ Архитектура нейронной сети: входной слой **x** отображается в выходной слой **y**. Промежуточные (скрытые) слои обозначаются $\mathbf{x}^{(j)}$, где j – порядковый номер слоя. Матрицы $\mathbf{A}_j$ содержат коэффициенты отображения предыдущего слоя в следующий. Размерность входного слоя известна, $\mathbf{x} \in \mathbb{R}^n$, но при выборе размерностей промежуточных слоев и выходного слоя имеется значительная гибкость. Количество слоев и способ отображения между ними также выбирает пользователь. Эта гибкая архитектура предоставляет большую свободу для построения хорошего классификатора

Они описывают композиционную структуру отображения между входом и выходом:

$$\mathbf{y} = \mathbf{A}_3\mathbf{A}_2\mathbf{A}_1\mathbf{x}. \tag{6.3}$$

Эта базовая архитектура масштабируется на M слоев, так что в общем случае линейной НС отображение входных данных в выходной слой имеет вид:

$$\mathbf{y} = \mathbf{A}_M\mathbf{A}_{M-1}\cdots\mathbf{A}_2\mathbf{A}_1\mathbf{x}. \tag{6.4}$$

Это сильно недоопределенная система, поэтому для нахождения единственного решения необходимо наложить ограничения. Одно ограничение очевидно: отображение должно порождать M матриц оптимальным образом. Отметим также, что с помощью линейных отображений, даже с учетом композиционной структуры, можно породить только ограниченное множество функциональных откликов – ограничение линейности не дает развернуться.

Нелинейные отображения также возможны, и именно они чаще всего используются при построении НС. На самом деле нелинейные функции активации дают более широкий простор для формирования функциональных

откликов, чем линейные. В данном случае связи между слоями описываются соотношениями

$$\mathbf{x}^{(1)} = f_1(\mathbf{A}_1, \mathbf{x}); \tag{6.5a}$$

$$\mathbf{x}^{(2)} = f_2(\mathbf{A}_2, \mathbf{x}^{(1)}); \tag{6.5b}$$

$$\mathbf{y} = f_3(\mathbf{A}_3, \mathbf{x}^{(2)}). \tag{6.5c}$$

Заметим, что нелинейные функции $f_j(\cdot)$ для разных слоев различны. Зачастую используется единственная функция, но никто не заставляет так поступать. В таком случае отображение входного слоя в выходной описывается композицией

$$\mathbf{y} = f_M(\mathbf{A}_M, \cdots, f_2(\mathbf{A}_2, f_1(\mathbf{A}_1, \mathbf{x}))\cdots), \tag{6.6}$$

которую можно сравнивать с общей задачей оптимизации (6.1) для построения НС. Поскольку эта система недоопределенная, необходимо наложить ограничения. Для приложений, работающих с большими данными, например ImageNET и задач компьютерного зрения, оптимизация в такой постановке обходится дорого из-за большого количества переменных. Но для сетей умеренного размера ее можно выполнить даже на рабочей станции или на ноутбуке. Современные алгоритмы стохастического градиентного спуска и обратного распространения позволяют проводить такую оптимизацию, и мы рассмотрим их в следующих разделах.

Однослойная сеть

Чтобы понять, как строится НС, рассмотрим однослойную сеть для построения классификатора, различающего собак и кошек. Этот пример неоднократно использовался в предыдущей главе. Напомним, что даны изображения собак и кошек или их вейвлет-преобразования. Построение представлено на рис. 6.2. Чтобы максимально упростить пример, будем считать, что выходной слой сети совсем простой:

$$\mathbf{y} = \{\text{собака, кошка}\} = \{+1, -1\}, \tag{6.7}$$

т. е. каждому входному вектору сопоставляется метка $\mathbf{y} \in \{\pm 1\}$. Тогда выходной слой состоит из одного блока. А наша задача, как и во всех алгоритмах обучения с учителем, – найти такое отображение, при котором любой вектор $\mathbf{x}_j$ правильно помечается меткой $\mathbf{y}_j$.

Проще всего построить линейное отображение между входными изображениями $\mathbf{x}_j \in \mathbb{R}^n$ и выходным слоем. В результате получаем линейную систему $\mathbf{AX} = \mathbf{Y}$ вида

$$\mathbf{AX} = \mathbf{Y} \;\rightarrow\; [a_1 \;\; a_2 \;\; \cdots \;\; a_n] \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_p \\ | & | & & | \end{bmatrix} = [+1 \;\; +1 \;\; \cdots \;\; -1 \;\; -1], \tag{6.8}$$

в которой каждый столбец матрицы **X** является изображением собаки или кошки, а столбцы **Y** – соответствующими им метками. Поскольку выходной слой состоит из единственного блока, матрицы **A** и **Y** сводятся к векторам. Наша цель – определить матрицу (вектор) **A** с элементами a_j. Проще всего вычислить псевдообращение матрицы **X**:

$$\mathbf{A} = \mathbf{Y}\mathbf{X}^{\dagger}. \tag{6.9}$$

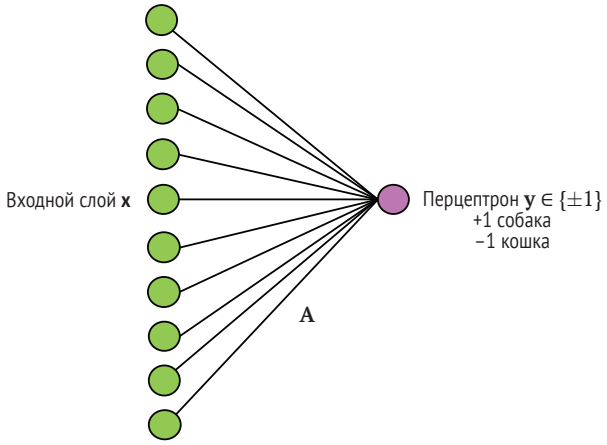

Рис. 6.2 ❖ Однослойная сеть для классификации изображений собак и кошек. Выходным слоем в этом случае является перцептрон с **y** ∈ {±1}. Линейное отображение между пространством входов и выходным слоем можно построить, обучив сеть решать уравнение **A** = **YX**†. Матрица **A** является тогда результатом регрессии методом наименьших квадратов

Таким образом, при наличии всего одного слоя мы можем построить НС, применив аппроксимацию методом наименьших квадратов. Разумеется, эту систему можно решить и другими способами, в т. ч. поощряющими разреженность. Следующая программа решает ее методами наименьших квадратов (**pinv**) и LASSO.

Листинг 6.1 ❖ Однослойная линейная нейронная сеть

```
load catData_w.mat; load dogData_w.mat; CD=[dog_wave cat_wave];
train=[dog_wave(:,1:60) cat_wave(:,1:60)];
test=[dog_wave(:,61:80) cat_wave(:,61:80)];
label=[ones(60,1); -1*ones(60,1)].';

A=label*pinv(train); test_labels=sign(A*test);
subplot(4,1,1), bar(test_labels)
subplot(4,1,2), bar(A)
figure(2), subplot(2,2,1)
A2=flipud(reshape(A,32,32)); pcolor(A2), colormap(gray)

figure(1), subplot(4,1,3)
A=lasso(train.',label.','Lambda',0.1).';
```

```
test_labels=sign(A*test);
bar(test_labels)
subplot(4,1,4)
bar(A)
figure(2), subplot(2,2,2)
A2=flipud(reshape(A,32,32)); pcolor(A2), colormap(gray)
```

На рис. 6.3 и 6.4 показаны результаты этой линейной однослойной ЕС с одним блоком в выходном слое. В четырех строках на рис. 6.3 показано, как выглядит выходной слой на тестовом наборе данных для псевдообращения матрицы и метода LASSO, а также столбчатые диаграммы весов в матрице **A** размера 32×32 (1024 пикселя). Заметим, что в случае псевдообращения все элементы матрицы ненулевые, тогда как LASSO выделяет небольшое число пикселей, которые могут классифицировать изображение так же верно, как все пиксели. На рис. 6.4 показаны матрицы **A** для обоих решений, представленные в виде изображений 32×32. Отметим, что веса в матрице **A**, полученной псевдообращением, свидетельствуют о большом числе признаков, отличающих собак от кошек. В случае же LASSO нужно всего несколько пикселей, сосредоточенных в области глаз и ушей. Таким образом, для данной однослойной сети интерпретировать результаты можно, взглянув на веса в построенной матрице **A**.

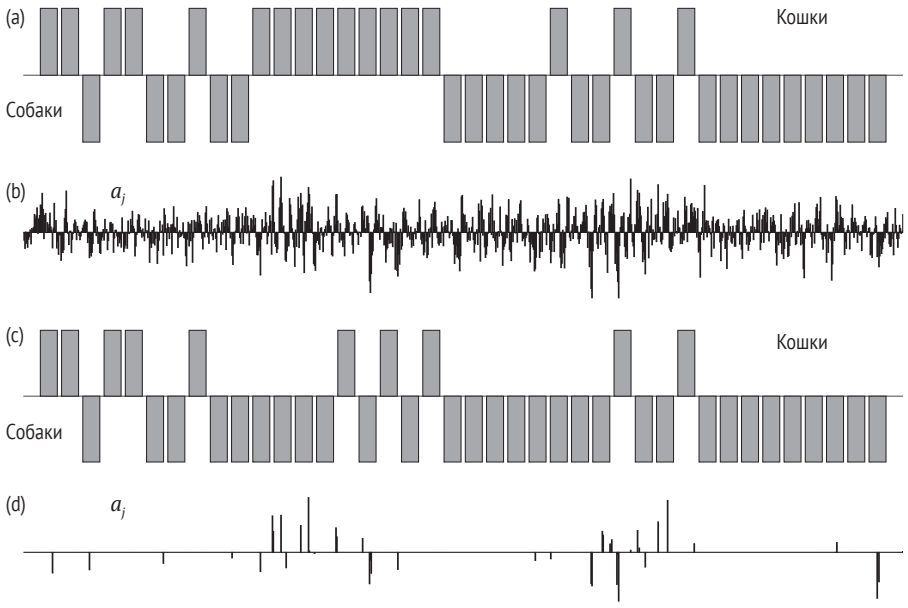

Рис. 6.3 ❖ Классификация на тестовом наборе с помощью обученной однослойной сети с линейным отображением входов (пространства пикселей) и одним блоком в выходном слое: (a) и (c) столбчатые диаграммы результатов классификации тестовых данных, полученные при обучении методами псевдообращения и LASSO соответственно. Результаты показывают, что в обоих случаях собаки неправильно распознаются чаще, чем кошки; (b) и (d) коэффициенты в матрице **A** для псевдообращения и LASSO соответственно. Отметим, что в случае LASSO количество ненулевых элементов мало, т. е. НС сильно разрежена

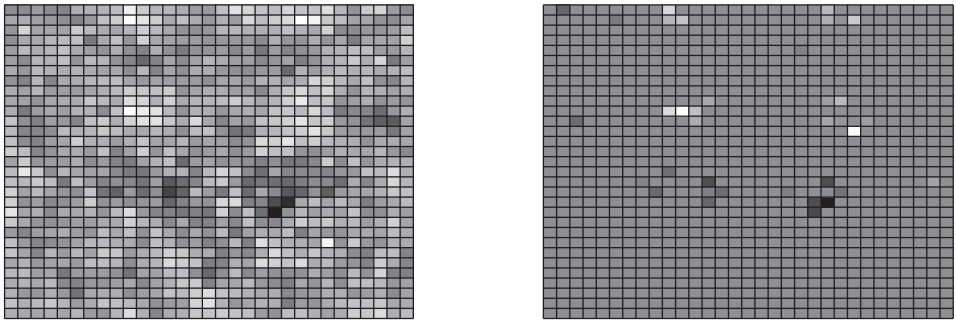

Рис. 6.4 ❖ Веса в матрице **A**, представленные в виде массивов 32×32. Слева показана матрица, вычисленная с помощью регрессии методом наименьших квадратов (псевдообращение), а справа – методом LASSO. Обе матрицы дают схожие результаты классификации на тестовых данных. Они легко интерпретируются – видно, что при псевдообращении число признаков собак и кошек велико, тогда как метод LASSO показывает, что для различения тех и других достаточно измерений в области глаз и ушей

6.2. Многослойные сети и функции активации

В предыдущем разделе мы построили простейшую НС – линейную, однослойную, с единственным нейроном в выходном слое. Количество возможных обобщений бесконечно, но в этом разделе мы сосредоточимся на двух. Первое обобщение связано с линейностью – ранее мы предполагали, что отображение пространства изображений в выходной слой линейно: $\mathbf{Ax} = \mathbf{y}$ в формуле (6.8). Здесь же мы рассмотрим нелинейные отображения вида

$$\mathbf{y} = f(\mathbf{A}, \mathbf{x}), \tag{6.10}$$

где $f(\cdot)$ – заданная *функция активации* (передаточная функция).

Использованное выше линейное отображение при всей своей простоте не обеспечивает достаточной гибкости и качества классификации. Перечислим другие стандартные функции активации:

$$f(x) = x \quad \text{линейная;} \tag{6.11a}$$

$$f(x) = \begin{cases} 0 & x \le 0 \\ 1 & x > 0 \end{cases} \quad \text{ступенчатая;} \tag{6.11b}$$

$$f(x) = \frac{1}{1 + \exp(-x)} \quad \text{логистическая (гладкая ступенька);} \tag{6.11c}$$

$$f(x) = \tanh(x) \quad \text{гиперболический тангенс;} \tag{6.11d}$$

$$f(x) = \begin{cases} 0 & x \le 0 \\ x & x > 0 \end{cases} \quad \text{ректифицированный линейный блок (ReLU).} \tag{6.11e}$$

Есть и другие возможности, но эти чаще всего применяются на практике, и для наших целей их достаточно. Важно, что функция $f(x)$ должна быть диф-

ференцируемой, т. к. будет использоваться в алгоритме градиентного спуска. Все вышеперечисленные функции гладкие или кусочно-гладкие. Пожалуй, чаще всего используется функция активации ReLU.

Если функция активации $f(x)$ нелинейна, а слоев больше одного, то стандартные процедуры линейной оптимизации, в частности псевдообращение матрицы и LASSO, неприменимы. Сейчас это может показаться неважным, но все же напомним, что мы производим оптимизацию в пространстве высокой размерности и в результате должны определить все элементы матрицы $\mathbf{A}$. Даже для скромных по размеру задач без специальных методов оптимизации эта проблема может оказаться вычислительно неразрешимой. По счастью, два основных алгоритма оптимизации, используемых при обучении НС, – стохастический градиентный спуск и обратное распространение – включены в состав функций работы с нейронными сетями в MATLAB. Поскольку эти методы критически важны, мы подробно рассмотрим их в следующих двух разделах.

Сеть может состоять и из нескольких слоев, как в (6.4) и (6.5c). В таком случае процедура оптимизации должна найти все промежуточные матрицы весов $\mathbf{A}_1, \mathbf{A}_2, \cdots \mathbf{A}_M$, тогда как в линейном случае ищется только одна матрица $\bar{\mathbf{A}} = \mathbf{A}_1 \cdots \mathbf{A}_2\mathbf{A}_M$. При многослойной структуре размер задачи оптимизации сильно увеличивается, т. к. нужно определить все элементы M матриц. Даже когда есть всего один слой, команда оптимизации **fminsearch** подвергается суровым испытаниям, если функция активации нелинейна, поэтому приходится переходить к алгоритму на основе градиентного спуска.

Включенный в MATLAB инструментарий для работы с нейронными сетями, как TensorFlow в случае Python, располагает обширным набором возможностей, что делает его исключительно мощным и удобным средством для построения НС. Приведенная ниже программа обучает NN классифицировать собак и кошек. Но теперь функция активации единственного слоя, отображающая входные данные в выходные метки, нелинейна. Выходной слой модифицируется следующим образом:

$$\mathbf{y} = \begin{bmatrix}1\\0\end{bmatrix} = \{\text{собака}\} \quad \text{и} \quad \mathbf{y} = \begin{bmatrix}0\\1\end{bmatrix} = \{\text{кошка}\}. \tag{6.12}$$

Половина данных используется для обучения, другая половина – для проверки результатов. Следующая программа строит сеть классификации изображений и обучает ее с помощью команды **train**.

Листинг 6.2 ❖ Нейронная сеть с нелинейными функциями активации

```
load catData_w.mat; load dogData_w.mat;
CD=[dog_wave cat_wave];

x=[dog_wave(:,1:40) cat_wave(:,1:40)];
x2=[dog_wave(:,41:80) cat_wave(:,41:80)];
label=[ones(40,1) zeros(40,1);
       zeros(40,1) ones(40,1)].';

net = patternnet(2,'trainscg');
net.layers{1}.transferFcn = 'tansig';

net = train(net,x,label);
```

```
view(net)
y = net(x);
y2= net(x2);
perf = perform(net,label,y);
classes2 = vec2ind(y);
classes3 = vec2ind(y2);
```

Здесь команда **patternnet** строит сеть с двумя выходами (6.12). Сеть оптимизируется в режиме **trainscg** – *масштабированный метод сопряженных градиентов с обратным распространением* (scaled conjugate gradient backpropagation). Параметр **net.layers** служит для задания функции активации, в данном случае – гиперболического тангенса (6.11d). Команда **view**(net) открывает диагностическое окно, показанное на рис. 6.5, в котором представлены архитектура сети и результаты оптимизации.

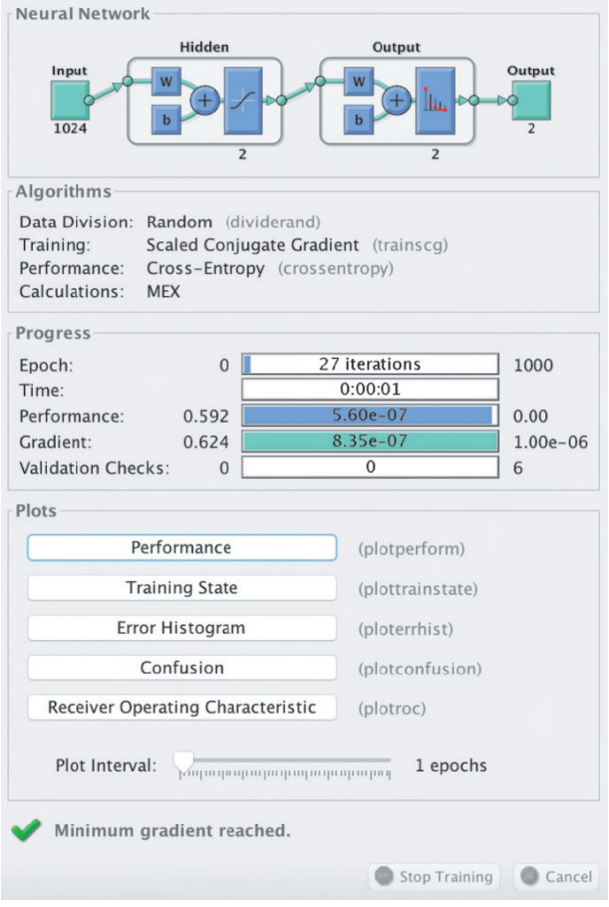

Рис. 6.5 ❖ Средство визуализации нейронной сети в MATLAB. Показано, в частности, количество итераций и достигнутое качество. При нажатии кнопок **Performance**, **Error Histogram** и **Confusion** открываются соответственно окна, показанные на рис. 6.7–6.9

На рис. 6.6 показаны результаты классификации на обучающем наборе с перекрестной проверкой и на тестовом наборе. Выходами являются векторы (6.12). Показаны компоненты векторов на 80 обучающих изображениях (40 собак и 40 кошек) и на 80 тестовых (40 собак и 40 кошек). На обучающем наборе достигнута идеальная классификация при использовании однослойной сети с гиперболическим тангенсом в качестве функции активации (6.11d). На тестовом наборе неправильно классифицированы 6 из 40 собак и кошек, т. е. верность составила ≈ 85 %.

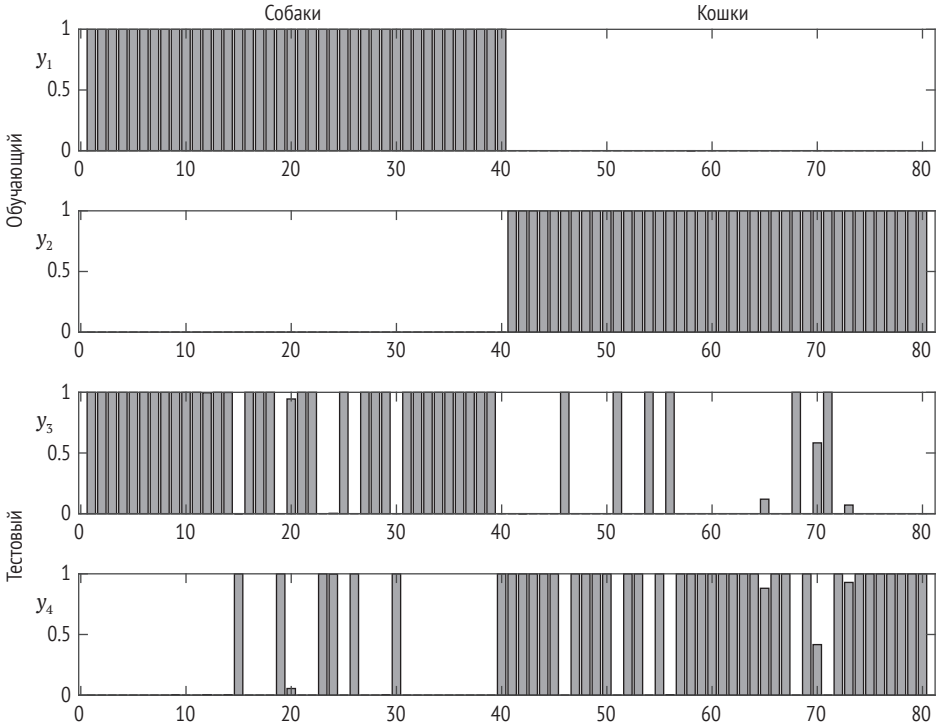

Рис. 6.6 ❖ Обученные выходные векторы $\mathbf{y} = [y_1 \quad y_2]^T$ на обучающем наборе идеально совпадают с истинными. На стадии обучения НС создается прошедший перекрестную проверку классификатор, достигающий 100%-ной верности на обучающих данных (верхние два рисунка, соответствующие 40 собакам и 40 кошкам). В применении к тестовому набору достигается верность 85 % (нижние два рисунка)

Диагностическое окно на рис. 6.5 дает доступ к ряду факторов, важных для оценки НС. На рис. 6.7 представлена информация о качестве НС. Алгоритм обучения автоматически разделяет данные на обучающий, контрольный и тестовый наборы. Затем метод стохастического градиентного спуска с обратным распространением выполняет несколько итераций обучения (эпох), пока ошибка перекрестной проверки не достигнет минимума. В данном случае хватило 22 эпох. Ошибка на тестовом наборе значительно выше, чем

удалось достичь в процессе перекрестной проверки. В данном случае объем данных для обучения был ограничен (40 собак и 40 кошек), поэтому достичь хорошего качества не удалось. Но все равно обученную сеть можно использовать для классификации новых данных, как показано на рис. 6.6.

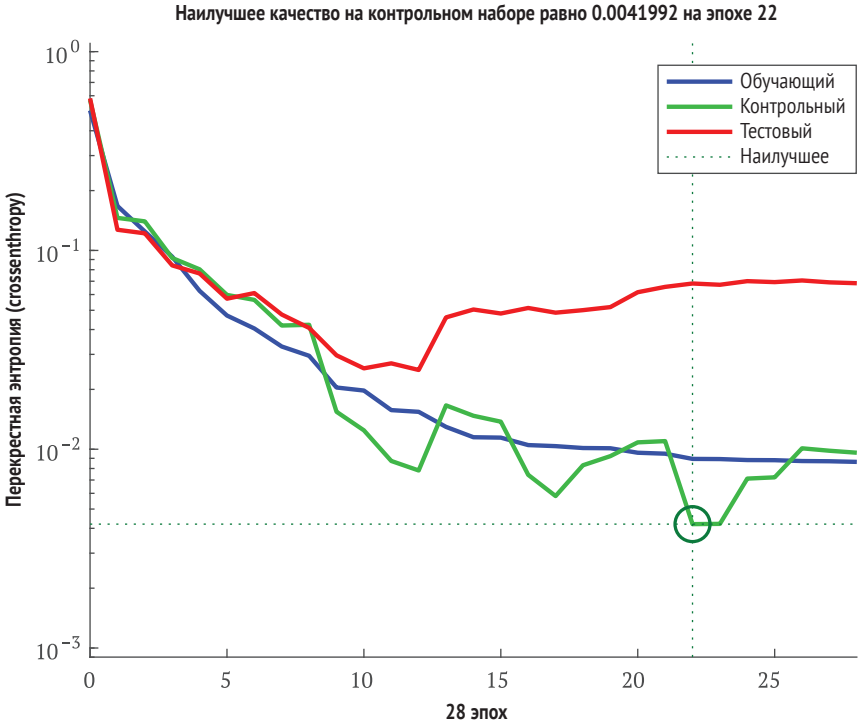

Рис. 6.7 ❖ Обучение НС на протяжении нескольких эпох. Алгоритм автоматически разделяет имеющиеся данные на обучающий, контрольный и тестовый наборы. Обучение продолжается (но не более 1000 эпох), пока кривая ошибки на контрольном наборе не достигнет минимума. Затем обученной сети подается тестовый набор для оценки качества. В данном случае НС обучена на небольшом объеме данных (40 собак и 40 кошек), поэтому качество невысокое. Чтобы увидеть этот рисунок на экране, нужно нажать кнопку **Performance** в интерактивном окне на рис. 6.6

В диагностическом окне НС (рис. 6.5) есть еще два полезных инструмента. На рис. 6.8 показана гистограмма ошибок обученной сети. Как и на рис. 6.7, данные разделены на обучающий, контрольный и тестовый наборы. Это дает общее представление о качестве классификации, достижимом с помощью алгоритма обучения НС. Еще один взгляд на качество дают матрицы неточностей для обучающих, контрольных и тестовых данных, показанные на рис. 6.9. В общем и целом рис. 6.7–6.9 являются отличными диагностическими инструментами, позволяющими оценить, насколько хорошо НС справляется с задачей классификации. На них также видны пределы качества.

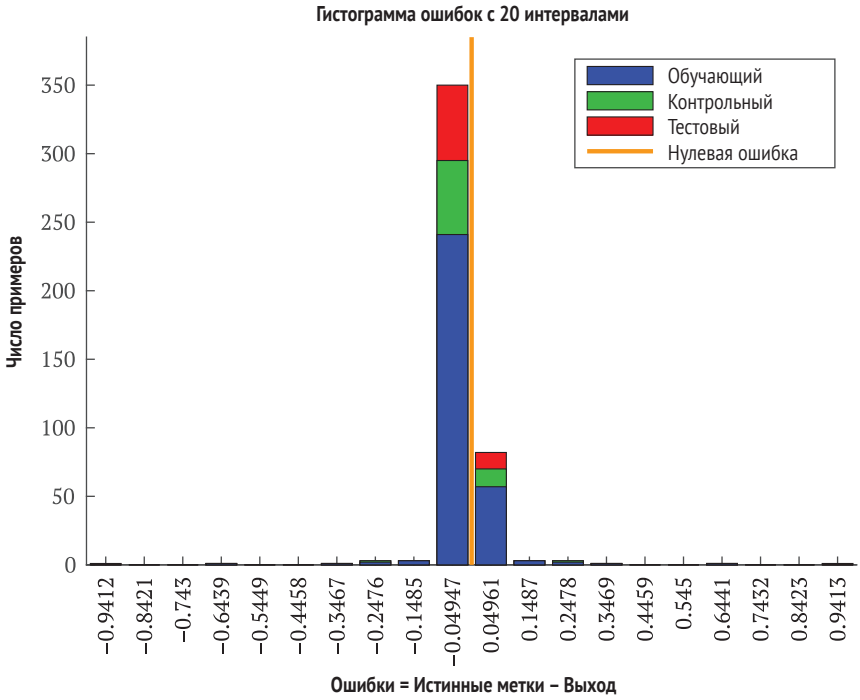

Рис. 6.8 ❖ Оценка качества архитектуры НС на обучающем, контрольном и тестовом наборах данных. Чтобы увидеть этот рисунок на экране, нужно нажать кнопку **Error Histogram** в интерактивном окне на рис. 6.6

6.3. Алгоритм обратного распространения

В предыдущих разделах было показано, что для определения весов сети нужны обучающие данные. Веса подбираются так, чтобы наилучшим образом различать изображения собак и кошек. В однослойной сети для этого использовалась регрессия методом наименьших квадратов и LASSO. Это доказывает, что по существу определение весов – не что иное, как оптимизация целевой функции, в ходе которой требуется свести к минимуму число неправильно классифицированных изображений. Но целевую функцию можно модифицировать, добавив регуляризатор или ограничения, например штраф по норме ℓ_1 в LASSO.

На практике для оптимизации берется не истинная целевая функция, а некий заместитель. Заместители выбираются в основном из соображений вычислительно реализуемой дифференцируемости. Существует много целевых функций, предназначенных для разных задач. Но для аппроксимации настоящей целевой функции часто подбирают подходящую функцию потерь. Главным фактором при обучении НС является простота вычислений.

В алгоритме обратного распространения используется композиционная природа НС, чтобы поставить задачу определения весов сети как задачу оптимизации. Точнее, задача ставится так, чтобы можно было применить стандартный метод градиентного спуска (см. раздел 4.2). Обратное распространение опирается на простой математический принцип: правило дифференцирования сложной функции. Можно доказать, что время вычислений при оценке градиента не более чем в пять раз превышает время вычисления самой функции [44]. Это утверждение известно как теорема Баура–Штрассена. На рис. 6.10 приведен простейший пример обратного распространения и применения градиентного спуска. Связь между входом и выходом этого единственного блока сети с одним скрытым слоем описывается формулой

$$y = g(z, b) = g(f(x, a), b). \tag{6.13}$$

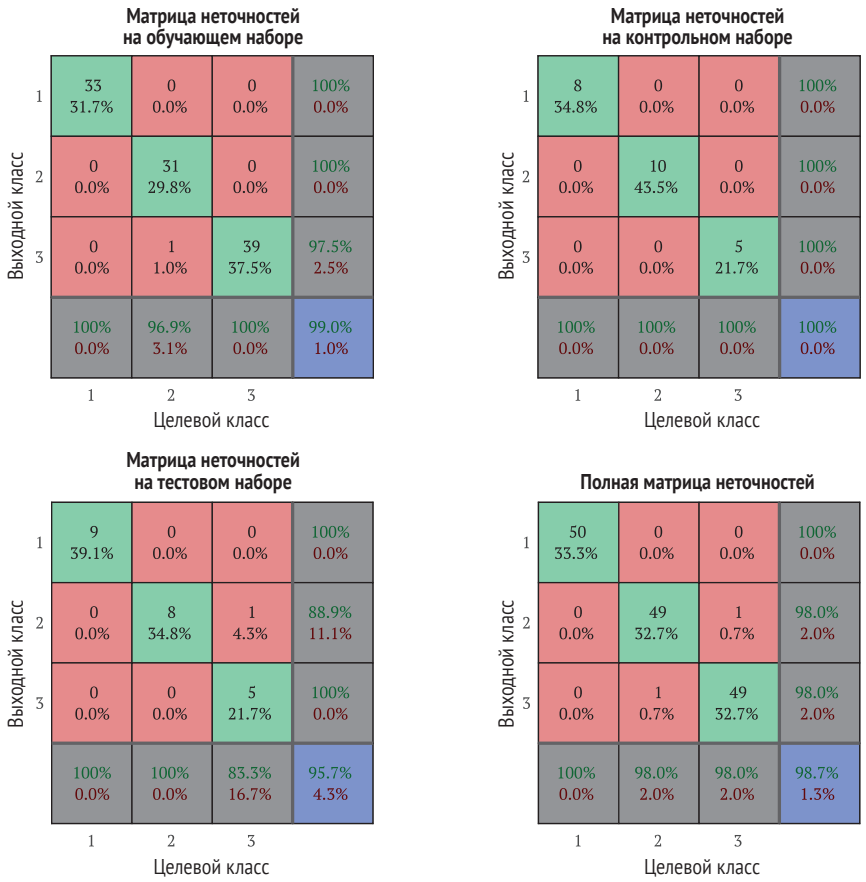

Рис. 6.9 ❖ Оценка качества сети по матрицам неточностей на обучающем, контрольном и тестовом наборах данных. Чтобы увидеть этот рисунок на экране, нужно нажать кнопку **Confusion** в интерактивном окне на рис. 6.6

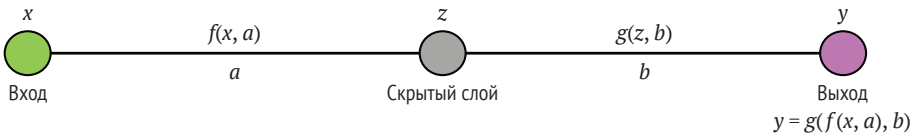

Рис. 6.10 ❖ Алгоритм обратного распространения в сети с одним блоком в одном скрытом слое. Благодаря композиционной природе сети между входом и выходом имеет место связь $y = g(z, b) = g(f(x, a), b)$. Для минимизации ошибки между выходом y и истинной меткой y_0, в силу правила дифференцирования сложной функции должно выполняться соотношение (6.15), которое используется для обновления весов. Заметим, что ошибка распространяется по сети в обратном направлении

Зная функции $f(\cdot)$ и $g(\cdot)$ с весами a и b, мы можем сравнить ошибку на выходе сети с объективной истиной:

$$E = \frac{1}{2}(y_0 - y)^2, \tag{6.14}$$

где y_0 – правильный результат, а y – его приближение, найденное сетью. Наша цель – найти a и b, доставляющие минимум ошибке. Для этого нужно, чтобы выполнялось равенство

$$\frac{\partial E}{\partial a} = -(y_0 - y)\frac{dy}{dz}\frac{dz}{da} = 0. \tag{6.15}$$

Благодаря композиционной природе сети вкупе с правилом дифференцирования сложной функции ошибка распространяется по сети в обратном направлении. Как это происходит, видно на примере члена $dy/dz\ dz/da$. Зная функции $f(\cdot)$ и $g(\cdot)$, легко явно вычислить производные, фигурирующие в правиле дифференцирования сложной функции.

Обратное распространение ошибки приводит к итеративному правилу обновления в алгоритме градиентного спуска

$$a_{k+1} = a_k + \delta\frac{\partial E}{\partial a_k}; \tag{6.16a}$$

$$b_{k+1} = b_k + \delta\frac{\partial E}{\partial b_k}, \tag{6.16b}$$

где δ – так называемая скорость обучения, а $\partial E/\partial a$ и $\partial E/\partial b$ можно вычислить явно по формуле (6.15). Этот итеративный алгоритм выполняется, пока не будет достигнута сходимость. Как и при любой итеративной оптимизации, для нахождения хорошего решения за разумное время необходимо хорошее начальное приближение.

Алгоритм обратного распространения выполняется в следующем порядке. (i) Задается НС и помеченный обучающий набор. (ii) Начальным весам сети присваиваются случайные значения. Важно не инициализировать веса нулями, как иногда делается в других алгоритмах машинного обучения. Если веса

равны нулю, то после каждого обновления исходящие веса каждого нейрона будут одинаковы, поскольку одинаковы градиенты. Кроме того, НС часто застревает в локальном оптимуме, где градиент равен нулю, а инициализация весов случайными значениями дает возможность бороться с этим путем многократного запуска алгоритма с разными начальными весами. (iii) На вход сети подаются обучающие данные, и сеть порождает выход **y**, который в идеале должен совпадать с метками $\mathbf{y}_0$. Затем вычисляются производные по каждому весу по формуле обратного распространения (6.15). (iv) Веса обновляются по формуле (6.16), в которую подставляется заданная скорость обучения δ. (v) Производится возврат к шагу (iii), и итерации продолжаются, пока не будет достигнута сходимость или исчерпано максимальное число итераций.

В качестве примера рассмотрим линейную функцию активации

$$f(\xi, \alpha) = g(\xi, \alpha) = \alpha\xi. \tag{6.17}$$

Тогда в обозначениях рис. 6.10 имеем:

$$z = ax; \tag{6.18a}$$

$$y = bz. \tag{6.18b}$$

Теперь можно явно вычислить градиенты по формуле (6.15):

$$\frac{\partial E}{\partial a} = -(y_0 - y)\frac{dy}{dz}\frac{dz}{da} = -(y_0 - y)\cdot b\cdot x; \tag{6.19a}$$

$$\frac{\partial E}{\partial b} = -(y_0 - y)\frac{dy}{db} = -(y_0 - y)z = -(y_0 - y)\cdot a\cdot x. \tag{6.19b}$$

Итак, зная текущие значения a и b, пару вход-выход x и y, а также целевую метку y_0, можно вычислить обе частные производные. Этой информации достаточно для выполнения обновления (6.16).

Обратное распространение для более глубокой сети производится аналогично. Рассмотрим сеть с M скрытыми слоями от z_1 до z_m и весом связи между x и z_1, равным a. В результате обобщения рис. 6.10 и формулы (6.15) получаем

$$\frac{\partial E}{\partial a} = -(y_0 - y)\frac{dy}{dz_m}\frac{dz_m}{dz_{m-1}}\cdots\frac{dz_2}{dz_1}\frac{dz_1}{da}. \tag{6.20}$$

Каскад производных, образующихся в результате применения правила дифференцирования сложной функции, иллюстрирует обратное распространение ошибки в процессе минимизации ошибки классификации.

Для полного обобщения метода обратного распространения нужно рассмотреть сеть с несколькими слоями и несколькими блоками в каждом слое. Общий случай показан на рис. 6.1. Наша цель – определить элементы каждой матрицы $\mathbf{A}_j$. Поэтому в процессе градиентного спуска необходимо обновить много параметров сети. Зачастую обучить сеть оказывается невозможно из-за гигантского объема вычислений, хотя правила обновления отдельных весов просты. Это означает, что НС подвержены проклятию размерности, т. к. для

обновления каждой матрицы, описывающей переход от одного слоя к другому, нужно пересчитать n^2 коэффициентов в предположении, что входной слой и оба связанных слоя имеют размерность n.

Обозначив $\mathbf{w}_j$ вектор весов, содержащий все элементы одной матрицы $\mathbf{A}_j$, получаем

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \delta \nabla E, \tag{6.21}$$

где ∇E обозначает градиент ошибки, получаемый по правилу дифференцирования сложной функции. Эта формула и выражает алгоритм обратного распространения для обновления весов и минимизации ошибки. Расписав ее поэлементно, имеем

$$w_{k+1}^j = w_k^j + \delta \frac{\partial E}{\partial w_k^j}. \tag{6.22}$$

Это уравнение имеет место для j-го элемента вектора $\mathbf{w}$. Член $\partial E/\partial w_j$ описывает обратное распространение по правилу дифференцирования сложной функции, т. е. порождает последовательность функций, подлежащих вычислению по формуле (6.20). Методы, позволяющие ускорить решение этой задачи оптимизации или хотя бы сделать вычисление практически осуществимым, остаются предметом активных исследований. Самым важным из них, пожалуй, является стохастический градиентный спуск, который мы рассмотрим в следующем разделе.

6.4. Алгоритм стохастического градиентного спуска

Обучение нейронных сетей обходится дорого из-за размера сети. Обучить НС даже умеренного размера может оказаться невозможным, если процедуры оптимизации плохо продуманы. Есть два основных алгоритма обучения НС: стохастический градиентный спуск (СГС) и обратное распространение. Алгоритм обратного распространения отвечает за эффективное вычисление градиента целевой функции, а СГС обеспечивает быстрое вычисление оптимальных весов. Есть и другие методы обучения НС, стремящиеся более экономно организовать вычисления, но здесь мы подробно рассмотрим только обратное распространение и СГС, чтобы дать читателю представление о базовой архитектуре НС.

Градиентный спуск рассматривался в разделе 4.2. Напомним, что этот алгоритм разрабатывался для нелинейной регрессии общего вида

$$f(x) = f(x, \boldsymbol{\beta}), \tag{6.23}$$

где $\boldsymbol{\beta}$ – параметры аппроксимации, которые нужно подобрать так, чтобы минимизировать ошибку. В случае НС параметры $\boldsymbol{\beta}$ – это веса сети, поэтому мы можем переписать формулу в виде

$$f(\mathbf{x}) = f(\mathbf{x}, \mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M), \tag{6.24}$$

где $\mathbf{A}_j$ – матрицы связей между соседними слоями НС, а всего слоев M.

Цель обучения НС – минимизировать ошибку между предсказаниями сети и фактическими данными. Стандартная среднеквадратическая ошибка в этом случае определяется так:

$$\underset{\mathbf{A}_j}{\operatorname{argmin}} E(\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) = \underset{\mathbf{A}_j}{\operatorname{argmin}} \sum_{k=1}^{n} (f(\mathbf{x}_k, \mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) - \mathbf{y}_k)^2. \tag{6.25}$$

Для решения этой задачи минимизации нужно приравнять к нулю частные производные по каждому элементу матрицы – $\partial E / \partial (a_{ij})_k = 0$, где $(a_{ij})_k$ – элемент на пересечении i-й строки и j-го столбца k-й матрицы ($k = 1, 2, \cdots, M$). Напомним, что обращение производной в нуль означает минимум, потому что максимальной ошибки не существует. В результате получаем градиент $\nabla f(\mathbf{x})$ функции по параметрам НС. Заметим далее, что функция $f(\cdot)$ вычисляется в каждой из n точек.

Как было показано в разделе 4.2, это приводит к итеративному методу Ньютона–Рафсона для нахождения минимума:

$$\mathbf{x}_{j+1}(\delta) = \mathbf{x}_j - \delta \nabla f(\mathbf{x}_j), \tag{6.26}$$

где параметр δ определяет величину шага в направлении градиента. В НС этот параметр называется *скоростью обучения*. В отличие от стандартного градиентного спуска, вычислить оптимальную скорость обучения практически невозможно из-за объема вычислений, пусть даже формально поставить задачу оптимизации для (6.25) нетрудно. Тому есть две причины: (i) количество весовых параметров в каждой матрице $\mathbf{A}_j$ очень велико и (ii) количество примеров данных n тоже очень велико.

Чтобы тем не менее решить задачу, алгоритм СГС не пытается оценить градиент в формуле (6.26) для всех n примеров. Вместо этого случайно выбирается одна точка или подмножество точек для *пакетного градиентного спуска* с целью приблизительно вычислить градиент на каждой итерации. В данном случае аппроксимацию методом наименьших квадратов (6.25) можно переформулировать в виде

$$E(\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) = \sum_{k=1}^{n} E_k(\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) \tag{6.27}$$

и

$$E_k(\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) = (f_k(\mathbf{x}_k, \mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M) - \mathbf{y}_k)^2, \tag{6.28}$$

где $f_k(\cdot)$ – функция аппроксимации в каждой точке, а элементы матриц $\mathbf{A}_j$ определяются в процессе оптимизации.

Теперь алгоритм градиентного спуска (6.26) принимает вид

$$\mathbf{w}_{j+1}(\delta) = \mathbf{w}_j - \delta \nabla f_k(\mathbf{w}_j), \tag{6.29}$$

где $\mathbf{w}_j$ – вектор всех весов сети, образованный из элементов $\mathbf{A}_j$ ($j = 1, 2, \cdots, M$) на j-й итерации, а градиент вычисляется только в k-й точке и для функции $f_k(\cdot)$. Таким образом, вместо вычисления градиента во всех n точках мы вычисляем его лишь в одной случайно выбранной точке. На следующей итерации случайно выбирается другая точка, в ней вычисляется градиент и производится обновление решения. Для сходимости алгоритма может потребоваться несколько проходов по набору данных, но вычисления на каждом шаге просты по сравнению с дорогостоящим вычислением якобиана, необходимого для нахождения градиента. Если брать не одну, а несколько точек, то получится следующий алгоритм пакетного градиентного спуска:

$$\mathbf{w}_{j+1}(\delta) = \mathbf{w}_j - \delta \nabla f_K(\mathbf{w}_j), \tag{6.30}$$

где $K \in [k_1, k_2, \cdots, k_p]$ – p случайно выбранных точек k_j, используемых для приближенного вычисления градиента.

Показанный ниже код – модификация кода градиентного спуска в разделе 4.2. Здесь мы производим подвыборку данных для аппроксимации градиента. Конкретно показан пакетный градиентный спуск с постоянной скоростью обучения $\delta = 2$. Для аппроксимации градиента функции на каждом шаге используется десять точек.

Листинг 6.3 ❖ Алгоритм стохастического градиентного спуска

```
h=0.1; x=-6:h:6; y=-6:h:6; n=length(x);
[X,Y]=meshgrid(x,y); clear x, clear y

F1=1.5-1.6*exp(-0.05*(3*(X+3).^2+(Y+3).^2));
F=F1 + (0.5-exp(-0.1*(3*(X-3).^2+(Y-3).^2)));
[dFx,dFy]=gradient(F,h,h);

x0=[4 0 -5]; y0=[0 -5 2]; col=['ro','bo','mo'];
for jj=1:3
    q=randperm(n); i1=sort(q(1:10));
    q2=randperm(n); i2=sort(q2(1:10));
    x(1)=x0(jj); y(1)=y0(jj);
    f(1)=interp2(X(i1,i2),Y(i1,i2),F(i1,i2),x(1),y(1));
    dfx=interp2(X(i1,i2),Y(i1,i2),dFx(i1,i2),x(1),y(1));
    dfy=interp2(X(i1,i2),Y(i1,i2),dFy(i1,i2),x(1),y(1));

    tau=2;
    for j=1:50
        x(j+1)=x(j)-tau*dfx; % обновить x, y, f
        y(j+1)=y(j)-tau*dfy;
        q=randperm(n); ind1=sort(q(1:10));
        q2=randperm(n); ind2=sort(q2(1:10));
        f(j+1)=interp2(X(i1,i2),Y(i1,i2),F(i1,i2),x(j+1),y(j+1))
        dfx=interp2(X(i1,i2),Y(i1,i2),dFx(i1,i2),x(j+1),y(j+1));
        dfy=interp2(X(i1,i2),Y(i1,i2),dFy(i1,i2),x(j+1),y(j+1));
        if abs(f(j+1)-f(j))<10^(-6) % проверить сходимость
            break
        end
```

```
    end
    if jj==1; x1=x; y1=y; f1=f; end
    if jj==2; x2=x; y2=y; f2=f; end
    if jj==3; x3=x; y3=y; f3=f; end
    clear x, clear y, clear f
end
```

На рис. 6.11 показано, как сходится СГС при трех начальных условиях. Как и градиентный спуск, алгоритм может застрять в локальном минимуме. Но теперь градиент аппроксимируется всего по 100 точкам, а не по всем 10^4 точкам, т. е. объем вычислений уменьшается на три порядка. Важно, что алгоритм СГС масштабируется, что позволяет существенно сэкономить на вычислениях даже при росте размерности данных. Поэтому СГС стал неотъемлемой частью обучения НС. Заметим, что скорость обучения, размер пакета и способ выборки данных сильно влияют на скорость сходимости.

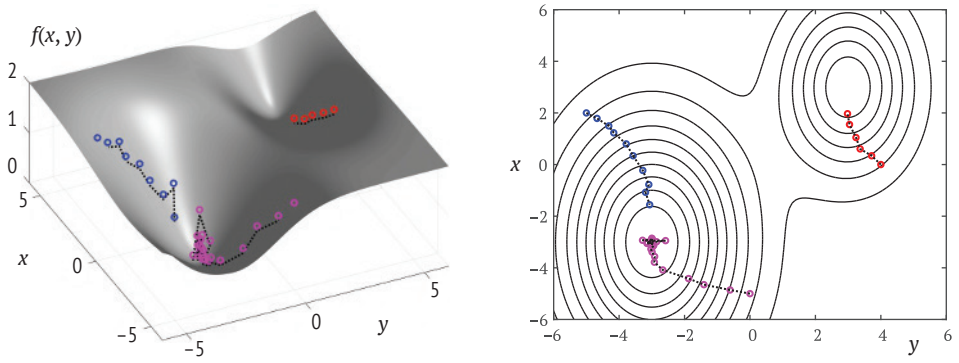

Рис. 6.11 ❖ Применение стохастического градиентного спуска в функции на рис. 4.3 (b). Можно сравнить сходимость с полным алгоритмом градиентного спуска (рис. 4.6). На каждом шаге пакетного СГС для аппроксимации градиента выбирается 100, а не все 10^4 точек. Алгоритм прогонялся с тремя начальными условиями $(x_0, y_0) = \{(4, 0), (0,-5), (-5, 2)\}$. В первом случае (красные точки) алгоритм застрял в локальном минимуме, а в двух остальных (синие и сиреневые точки) нашел глобальный минимум. Для обновления решения использовалась интерполяция градиентных функций на рис. 4.5

6.5. Глубокие сверточные нейронные сети

Познакомившись с основами архитектуры НС и поняв, как формулируется задача оптимизации (обратное распространение) и как эффективно реализовать градиентный спуск (СГС), мы теперь готовы построить глубокую сверточную нейронную сеть (ГСНС) – один из основных элементов, встречающихся в методах *глубокого обучения*. В самом деле, когда сегодня специалисты говорят о практическом использовании НС, они почти всегда имеют в виду ГСНС. Но как бы сильно мы ни желали заполучить теоретически обоснованный подход к построению ГСНС, создание высокопроизводительной

и качественной сети по-прежнему во многом остается искусством и зависит от интуиции эксперта. К тому же ГСНС особенно чувствительны к переобучению, поэтому нужно уделять повышенное внимание перекрестной проверке. В сравнительно недавно вышедшей книге по глубокому обучению Goodfellow et al. [216][1] подробно описано текущее состояние дел в ГСНС, в т. ч. многие эвристические приемы обучения эффективных сетей.

Пакет MATLAB для построения НС включает огромное количество средств и настраиваемых параметров. Эта гибкость одновременно привлекает и отталкивает. В начале данной главы мы уже говорили, что при проектировании сети возникает много вопросов. Сколько должно быть слоев? Каковы размеры каждого слоя? Как должен выглядеть выходной слой? Как лучше определить отображение между слоями: *линейное* или *нелинейное*?

Каноническая структура ГСНС показана на рис. 6.12. Мы видим несколько сверточных и пулинговых слоев. Также видно, что каждый слой может быть связан с несколькими последующими слоями, или пространствами *признаков*, различающимися выбором функций активации и (или) параметрами сети. Все эти слои в конечном итоге объединяются в выходном слое. Число связей, которые нужно обновить в ходе обратного распространения и СГС, может быть очень велико, поэтому даже для обучения сети скромного размера на умеренном объеме данных могут понадобиться значительные вычислительные ресурсы. Типичная ГСНС состоит из 7–10 слоев. Недавно были предприняты усилия для оценки преимущества по-настоящему глубокой сети, состоящей примерно из 100 слоев, но о достоинствах такой архитектуры пока еще рано судить. Далее мы обсудим некоторые из наиболее важных аспектов ГСНС, в т. ч. сверточные слои, пулинговые слои, полносвязные слои и прореживание.

Сверточные слои

Сверточные слои аналогичны оконным преобразованиям Фурье (преобразованиям Габора) или вейвлетам из главы 2 в том смысле, что для конструирования признаков используется небольшая выборка из полного пространства входов высокой размерности. На рис. 6.12 сверточные окна (темно-серые прямоугольники) скользят по всему слою (светло-серые прямоугольники). Каждое сверточное окно преобразует данные в новый блок с помощью заданной функции активации, как показано на рис. 6.12 (а). Таким образом, пространства признаков строятся из небольших участков данных (патчей). Сверточные слои особенно полезны для обработки изображений, потому что способны извлекать такие признаки, как границы. Известно, что вейвлеты тоже эффективно извлекают такие признаки, и, как показано в работах Маллата с сотрудниками [358, 12], между вейвлетами и ГСНС существуют глубокие математические связи. Заметим, что входной слой на рис. 6.12 можно использовать для построения многих разных слоев, просто изменяя функцию активации $f(\cdot)$ и размер сверточного окна.

---

[1] *Гудфеллоу Я., Бенджио И., Курвиль Ю. А.* Глубокое обучение. М.: ДМК-Пресс, 2018.

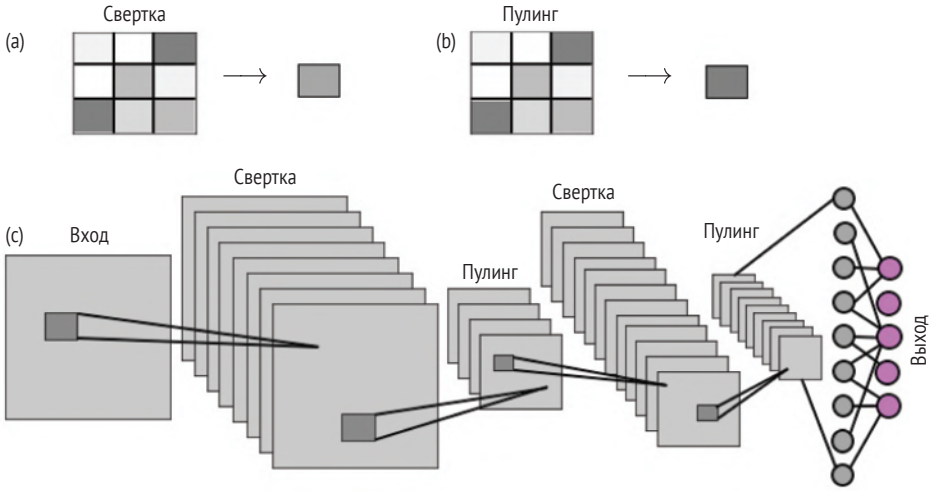

Рис. 6.12 ❖ Каноническая архитектура ГСНС со сверточными и пулинговыми слоями. Темно-серыми прямоугольниками обозначены сверточные окна, по которым формируются следующие слои. Заметим, что каждый слой может подвергаться нескольким преобразованиям для создания разных пространств признаков. В конечном итоге сеть интегрирует всю информацию в выходном слое

Пулинговые слои

Часто в ГСНС между сверточными слоями вставляют пулинговый слой. Его задача – уменьшить пространственный размер представления, а значит, число параметров сети и объем вычислений. Эта стратегия (i) помогает контролировать переобучение и (ii) позволяет избежать переполнения памяти. Пулинговые слои независимо применяются к каждому срезу входных данных и изменяют их размер. Если вычисляется максимальное значение всех блоков в сверточном окне, то говорят о *max-пулинге*. При обработке изображений чаще всего применяется max-пулинг с фильтрами 2×2, что позволяет уменьшить предыдущий слой вдвое по ширине и высоте, т. е. отбросить 75 % активаций. Каждая операция пулинга в этом случае вычисляет максимум из 4 чисел (по области 2×2). Количество слоев на данном уровне остается неизменным. На рис. 6.12 (b) приведен пример операции max-пулинга, когда сверточное окно размера 3×3 преобразуется в одно число, равное максимуму из 9 чисел.

Полносвязные слои

Иногда в ГСНС вставляются полносвязные слои, чтобы связать между собой различные области. В пулинговых и сверточных слоях имеются только локальные связи, тогда как полносвязный слой восстанавливает *глобальную* связность. Слои такого типа дают возможность потенциально важному пространству признаков повысить общее качество сети.

Прореживание

Переобучение – серьезная проблема в ГСНС. Именно из-за переобучения ГСНС часто не могут продемонстрировать хорошую обобщаемость (см. главу 4, посвященную регрессии). Кроме того, большие ГСНС работают медленно, поэтому бороться с переобучением путем комбинирования предсказаний многих больших нейронных сетей в онлайновой конфигурации неэффективно. Проблему позволяет решить прореживание. Идея заключается в том, чтобы случайным образом выбрасывать блоки (вместе с их связями) из ГСНС на этапе обучения, т. е. в процессе обновления весов алгоритмами СГС и обратного распространения. Это предотвращает слишком сильную совместную адаптацию блоков. Во время обучения прореживание образцов формирует комбинаторное число разных «истонченных» сетей. Эта идея похожа на ансамблевые методы построения случайных лесов. На этапе тестирования нетрудно аппроксимировать эффект усреднения предсказаний всех этих прореженных сетей, просто воспользовавшись одной непрореженной сетью с меньшими весами. Это значительно уменьшает переобучение и, как было показано, существенно превосходит другие методы регуляризации [499].

Для обучения ГСНС было придумано много других приемов, но описанные выше относятся к числу наиболее популярных. Самое успешное применение эти методы находят в задачах компьютерного зрения, где ГСНС демонстрируют непревзойденное качество по сравнению с другими методами машинного обучения. Важно, что именно набор ImageNET позволил слоям ГСНС приблизиться к качеству распознавания, не уступающему человеку.

Для иллюстрации обучения и выполнения ГСНС воспользуемся данными, включенными в MATLAB, а именно набором обучающих и тестовых данных, содержащим рукописные буквы A, B, C. Следующая программа загружает этот набор и выводит репрезентативную выборку символов (рис. 6.13).

Листинг 6.4 ❖ Загрузка изображений букв

```
load lettersTrainSet
perm = randperm(1500,20);
for j = 1:20
    subplot(4,5,j);
    imshow(XTrain(:,:,:,perm(j)));
end
```

Обучающий набор XTrain содержит 1500 полутоновых изображений букв A, B, C размера 28×28, организованных в виде 4-мерного массива. Все буквы встречаются в наборе одинаковое число раз. Переменная TTrain содержит категориальный массив меток, т. е. истинных букв, изображенных на картинке. Следующая программа строит и обучает ГСНС.

Листинг 6.5 ❖ Обучение ГСНС

```
layers = [imageInputLayer([28 28 1]);
          convolution2dLayer(5,16);
          reluLayer();
```

```
          maxPooling2dLayer(2,'Stride',2);
          fullyConnectedLayer(3);
          softmaxLayer();
          classificationLayer()];
options = trainingOptions('sgdm');
rng('default') % For reproducibility
net = trainNetwork(XTrain,TTrain,layers,options);
```

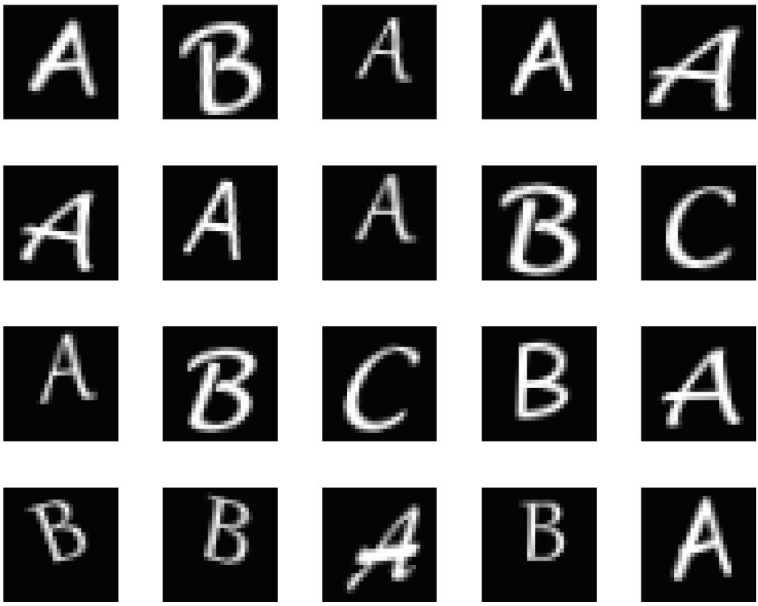

Рис. 6.13 ❖ Репрезентативная выборка изображений букв A, B, C. Всего имеется 1500 изображений размера 28×28 (XTrain) и соответствующие им метки (TTrain)

Обратите внимание, как просто соединяются слои сети. Отметим также, что задан слой с функцией активации ReLu и методом обучения sgdm (стохастический градиентный спуск). Команда **trainNetwork** получает все параметры и спецификации слоев и строит оптимальный классификатор. Обученную сеть можно затем применить к тестовому набору данных.

Листинг 6.6 ❖ Тестирование качества ГСНС

```
load lettersTestSet;
YTest = classify(net,XTest);
accuracy = sum(YTest == TTest)/numel(TTest)
```

Качество классификации получившейся сети составляет примерно 93 %. Из структуры кода видно, что изменить архитектуру и параметры сети очень просто. И наверное, было бы нетрудно построить сеть, которая превзойдет по качеству эту ГСНС. Как уже отмечалось, для создания высококачественных сетей необходимы искусство и интуиция эксперта.

6.6. Нейронные сети для динамических систем

Нейронные сети предлагают поразительно гибкую архитектуру для решения самых разных математических задач. Напомним слова С. Маллата: «Обучение с учителем – это многомерная задача интерполяции» [358]. Если имеется достаточно данных, то НС дают возможность решать различные задачи классификации и предсказания. До сих пор мы демонстрировали их потенциал только на примере компьютерного зрения. Однако НС можно использовать и для предсказания будущих состояний динамических систем (см. главу 7).

Чтобы проиллюстрировать полезность НС в приложениях динамических систем, рассмотрим дифференциальные уравнения Лоренца [345]:

$$\dot{x} = \sigma(y - x); \tag{6.31a}$$

$$\dot{y} = x(\rho - z) - y; \tag{6.31b}$$

$$\dot{z} = xy - \beta z, \tag{6.31c}$$

где состояние системы с параметрами $\sigma = 10$, $\rho = 28$ и $\beta = 8/3$ описывается вектором $\mathbf{x} = [x \ y \ z]^T$. Эта система будет подробно рассмотрена в следующей главе. А пока выполним моделирование данной нелинейной системы и воспользуемся ей для демонстрации того, как НС можно обучить давать характеристику динамической системе. Точнее, наша цель – показать, как обучить НС правилу обновления для перехода из состояния $\mathbf{x}_k$ в состояние $\mathbf{x}_{k+1}$, где $\mathbf{x}_k$ – состояние системы в момент t_k. Для получения точного решения требуется нелинейная передаточная функция, потому что сами уравнения Лоренца нелинейны.

Обучающие данные для НС берутся из моделирования уравнений Лоренца с высокой точностью. Программа в листинге 6.7 генерирует сто разных начальных условий для построения ста траекторий. Шаг дискретизации фиксирован и равен $\Delta t = 0.01$. Заметим, что шаг дискретизации отличается от шагов в методе Рунге-Кутта четвертого порядка [316], где шаги выбираются адаптивно, чтобы удовлетворить жесткие требования к точности.

Листинг 6.7 ❖ Создать обучающие данные для лоренцевых траекторий

```
% Моделирование уравнений Лоренца
dt=0.01; T=8; t=0:dt:T;
b=8/3; sig=10; r=28;

Lorenz = @(t,x)([ sig * (x(2) - x(1)) ; ...
                  r * x(1)-x(1) * x(3) - x(2) ; ...
                  x(1) * x(2) - b*x(3) ]);
ode_options = odeset('RelTol',1e-10, 'AbsTol',1e-11);

input=[]; output=[];
for j=1:100 % обучающие траектории
    x0=30*(rand(3,1)-0.5);
    [t,y] = ode45(Lorenz,t,x0);
```

```
    input=[input; y(1:end-1,:)];
    output=[output; y(2:end,:)];
    plot3(y(:,1),y(:,2),y(:,3)), hold on
    plot3(x0(1),x0(2),x0(3),'ro')
end
```

Для моделирования уравнений Лоренца нужны две матрицы: **input** и **output**. Первая описывает состояние системы $\mathbf{x}_k$, а вторая – состояние $\mathbf{x}_{k+1}$ в момент времени, сдвинутый на $\Delta t = 0.01$.

НС должна обучиться нелинейному отображению $\mathbf{x}_k$ в $\mathbf{x}_{k+1}$. На рис. 6.14 показаны траектории, на которых обучается НС. Обратите внимание на разнообразие начальных условий и аттрактор системы.

Теперь обучим на этих траекториях НС, которая будет предсказывать решение в момент, отстоящий от текущего на $\Delta t = 0.01$, при любых начальных условиях. Мы построим трехслойную сеть с десятью блоками в каждом слое и разными функциями активации. Наш выбор функций активации, количества блоков в слое и числа слоев произволен. Совсем нетрудно сделать сеть глубже и шире и поэкспериментировать с другими функциями активации. Но даже при нашем выборе НС работает замечательно и не нуждается в дополнительной настройке. Для ее построения нужно всего несколько строчек кода.

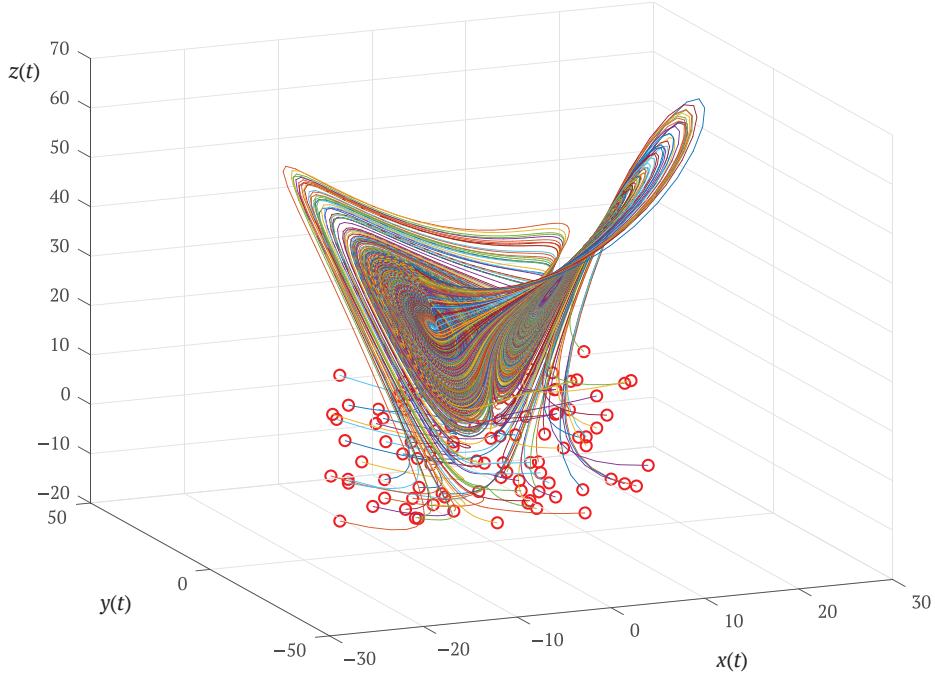

Рис. 6.14 ❖ Эволюция динамических уравнений Лоренца для ста случайно выбранных начальных условий (красные точки). С параметрами σ = 10, ρ = 28, β = 8/3 все траектории коллапсируют к аттрактору. Эти траектории, сгенерированные по разным начальным данным, используются для обучения нейронной сети, предсказывающей переход из $\mathbf{x}_k$ в $\mathbf{x}_{k+1}$

Листинг 6.8 ❖ Построение нейронной сети для уравнений Лоренца

```
net = feedforwardnet([10 10 10]);
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'radbas';
net.layers{3}.transferFcn = 'purelin';
net = train(net,input.',output.');
```

Программа порождает функцию **net**, которой можно предъявить новый набор данных для вычисления предсказаний. Точнее, функция **net** дает нелинейное отображение $\mathbf{x}_k$ в $\mathbf{x}_{k+1}$. На рис. 6.15 показана структура сети и качество обучения после 1000 эпох. Показаны также результаты перекрестной проверки. НС монотонно сходится к сети с ошибкой порядка 10^{-5}.

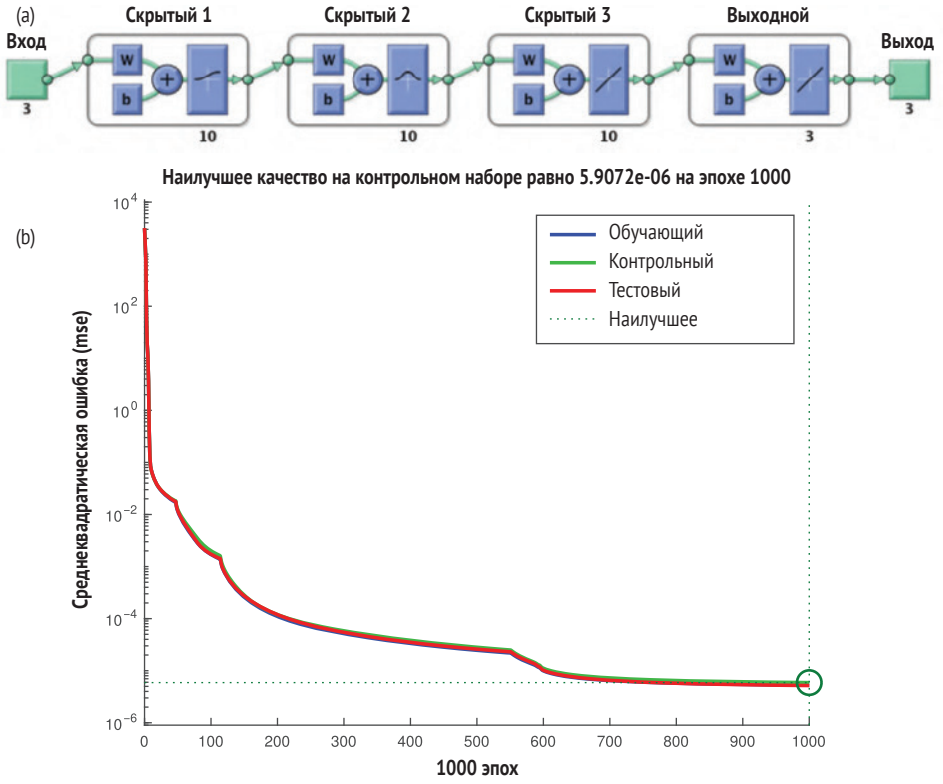

Рис. 6.15 ❖ (a) Архитектура сети, обучаемой на траекториях, показанных на рис. 6.14. Сеть состоит из трех слоев с десятью блоками в каждом слое и различными функциями активации; (b) качество алгоритма оптимизации НС. После 1000 эпох обучения порядок ошибки равен 10^{-5}. В процессе обучения НС подвергалась перекрестной проверке

После того как НС обучена на данных о траекториях, нелинейную модель отображения $\mathbf{x}_k$ в $\mathbf{x}_{k+1}$ можно использовать для предсказания будущего состояния системы по начальному условию. В программе ниже обученная функция **net** принимает начальное условие и вычисляет, каким будет решение

через время Δt. Результат можно снова подать на вход функции **net** и оценить решение через время $2\Delta t$. Таким образом можно предсказать состояние системы в любой момент в будущем. Ниже мы прогнозируем решение уравнений Лоренца на восемь временных шагов. Его можно сравнить с результатом точного моделирования методом Рунге-Кутта 4-го порядка. Показанная ниже итерационная схема аппроксимирует динамику с помощью НС.

Листинг 6.9 ❖ Предсказывающая нейронная сеть

```
ynn(1,:)=x0;
for jj=2:length(t)
    y0=net(x0);
    ynn(jj,:)=y0.'; x0=y0;
end

plot3(ynn(:,1),ynn(:,2),ynn(:,3),':','Linewidth',[2])
```

На рис. 6.16 показаны две случайно выбранные траектории (сплошные кривые) и предсказание тех же траекторий нейронной сетью (пунктирные кривые). Предсказание НС на удивление хорошо аппроксимирует результаты высокоточного моделирования. Это доказывает, что использованные для обучения данные способны породить высококачественную нелинейную модель перехода от $\mathbf{x}_k$ к $\mathbf{x}_{k+1}$. Качество аппроксимации более наглядно видно на рис. 6.17, где изменение во времени отдельных компонент $\mathbf{x}$ сравнивается с предсказаниями НС. Дополнительные сведения см. в главе 7.

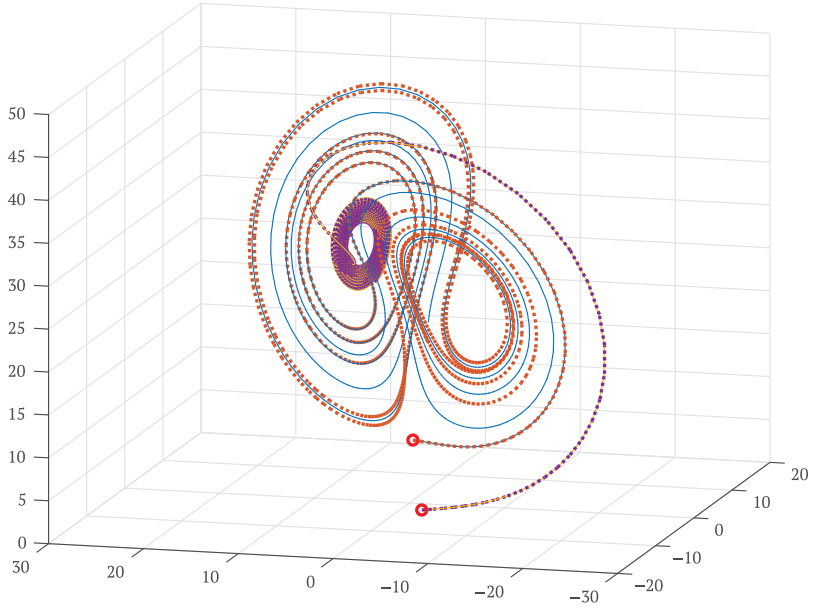

Рис. 6.16 ❖ Сравнение истинной эволюции системы Лоренца (сплошная кривая) с предсказанием нейронной сети (пунктирная кривая) для двух случайно выбранных начальных условий (красные точки). Предсказание НС всюду близко к динамической траектории модели. Более детальное сравнение приведено на рис. 6.17

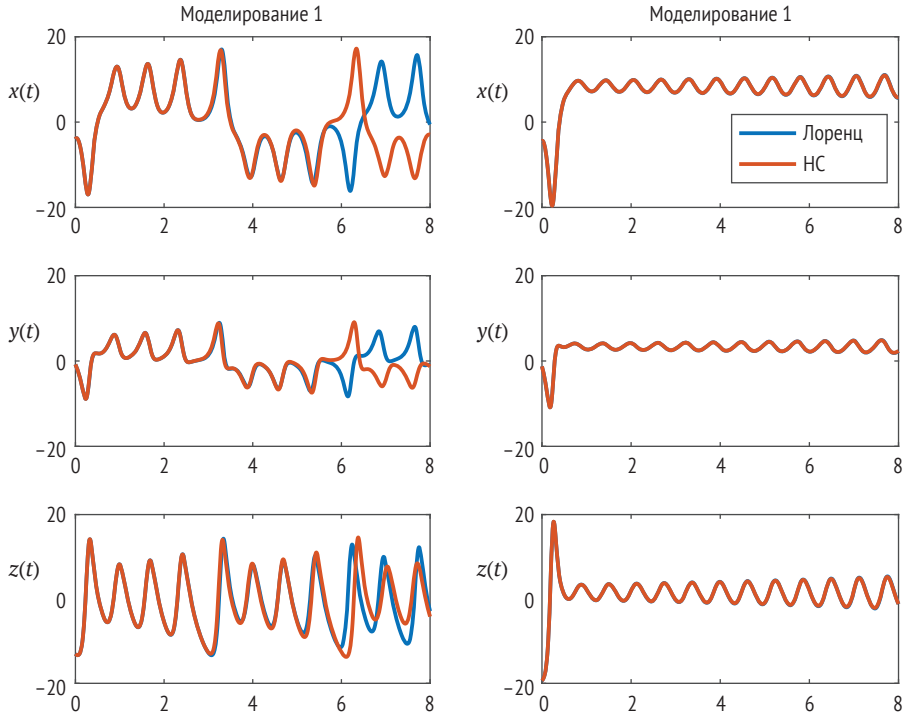

Рис. 6.17 ❖ Сравнение эволюции решений уравнений Лоренца для двух случайно выбранных начальных условий (показано также на рис. 6.16). В левом столбце показано, что истинное решение дифференциальных уравнений Лоренца и моделирование с помощью НС дают одинаковые результаты до $t \approx 5.5$, после чего расходятся. Но для других начальных условий предсказание НС (правый столбец) совпадает с истинной траекторией на протяжении всего временного окна

Итак, НС можно обучить динамике. Точнее, НС, похоже, обучилась алгоритму, приблизительно эквивалентному методу Рунге-Кутта четвертого порядка. На самом деле НС применялись для моделирования динамических систем [215] и других физических процессов [381] на протяжении десятилетий. Но настоящий прорыв был совершен недавно, когда глубокие НС стали использоваться для обучения погружений Купмана, что привело к появлению нескольких отличных статей [550, 368, 513, 564, 412, 332]. Например, в архитектуре VAMPnet [550, 368] используется автокодировщик с временной задержкой и специальная вариационная оценка для нахождения координат Купмана во впечатляющем примере укладки белка. Иными словами, вариационные автокодировщики могут на основе данных строить низкоранговые модели, являющиеся эффективными и компактными представлениями оператора Купмана [349]. Результирующая сеть по построению экономная и интерпретируемая, так что мы получаем одновременно гибкость нейронных сетей и физическую интерпретацию теории Купмана. Во всех недавних исследованиях показано, что при решении трудных задач представления в виде глубоких НС более гибкие и точные, чем прочие передовые методы.

6.7. Разнообразие нейронных сетей

Из многочисленных архитектур НС мы пока рассмотрели только несколько основных. Ни в этой главе, ни в книге в целом мы не пытаемся дать исчерпывающий обзор текущего состояния дел в области нейронных сетей. Мы лишь хотим проиллюстрировать некоторые ключевые идеи и архитектуры, благодаря которым НС заняли первое место в современной науке о данных. Более полный обзор смотрите в книге [216]. Но прежде чем закончить эту главу, мы хотели бы кратко упомянуть архитектуры НС, практически используемые для решения различных задач. На составление этого перечня нас вдохновил *зоопарк нейронных сетей* Федора ван Веена из института Азимова (http://www.asimovinstitute.org).

В зоопарке нейронных сетей перечислены некоторые архитектуры НС. Одни сети используются на практике часто, другие – в нишевых приложениях. Но как бы то ни было, зоопарк демонстрирует поразительное разнообразие сетей, достигнутое в результате усилий многих исследователей, превративших их в основной инструмент науки о данных. На рис. 6.18 перечислены канонические структуры, обсуждаемые ниже. В нижней части рисунка перечислены типы блоков сети, включая входные, выходные и скрытые. Кроме того, блоки скрытого слоя сети могут обладать эффектом памяти, ядерными структурами и (или) способностью осуществлять свертку либо пулинг. Для каждой архитектуры приводится краткое описание и ссылка на оригинальную работу, в которой она была предложена.

Перцептрон

Первая математическая модель НС, предложенная Фукусима в 1980 году [193], называлась *неокогнитрон*. В ней был один слой с одним выходным блоком, который назывался перцептроном. Модель принимала категориальное решение на основе знака выхода. На рис. 6.2 показано, как сеть с такой архитектурой классифицирует собак и кошек. Перцептрон – это алгоритм обучения бинарных классификаторов с учителем.

Сети прямого распространения (FF)

В сети прямого распространения входной слой связывается с выходным таким образом, что связи между блоками не образуют циклов. Пример такой архитектуры показан на рис. 6.1, где информация течет только слева направо. Это рабочая лошадка обучения с учителем, позволяющая обучать веса, при которых данные классифицируются наилучшим образом. Сеть прямого распространения использовалась на рис. 6.5 и 6.15 для обучения классификатора собак и кошек и предсказания траекторий аттрактора Лоренца. Важным подклассом сетей прямого распространения являются *глубокие сети прямого распространения* (deep feed forward – DFF), состоящие из большого числа скрытых

слоев, обычно 7–10. Второй важный класс FF-сетей – *радиально-базисные сети*, в которых в качестве функций активации используются радиально-базисные функции [87]. Как и любая FF-сеть, радиально-базисная сеть имеет много применений, в т. ч. для аппроксимации функций, предсказания временных рядов, классификации и управления.

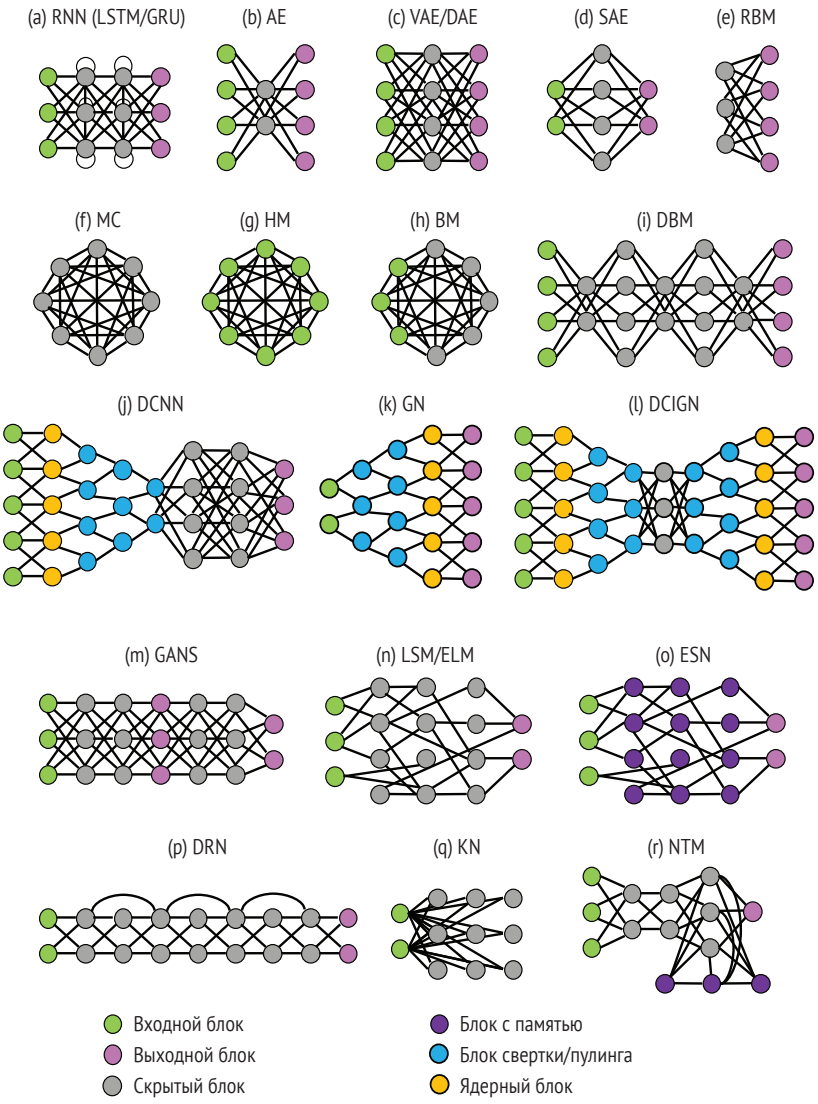

Рис. 6.18 ❖ Архитектуры нейронных сетей, часто встречающиеся в литературе. НС состоит из входных, выходных и скрытых блоков. Кроме того, блоки могут иметь память, выполнять свертку и (или) пулинг и ядерное преобразование. Все сети и их акронимы объясняются в тексте

Рекуррентная нейронная сеть (RNN)

На рис. 6.18 (a) RNN-сети характеризуются тем, что связи между блоками образуют ориентированный граф, вытянутый в одном направлении. Это позволяет моделировать поведение временной последовательности [172]. В отличие от сетей прямого распространения, RNN-сеть может использовать свое внутреннее состояние (память) для обработки последовательности событий. В канонической архитектуре на рис. 6.18 (a) каждый блок соединен петлей обратной связи с самим собой. Такая структура, отсутствующая в сетях прямого распространения, открывает возможность для многочисленных инноваций. Конкретно речь идет о временных задержках и контурах обратной связи. Так, управляемые состояния, называемые вентильными или вентильной памятью, служат основой для двух важнейших архитектур: сетей с *долгой краткосрочной памятью* (long-short term memory – LSTM) [248] и *вентильных рекуррентных блоков* (gated recurrent units – GRU) [132]. Особенно важны LSTM, совершившие революции в распознавании речи; они поставили целый ряд рекордов качества и превосходят все традиционные модели в различных приложениях обработки речи. GRU – вариант LSTM-сетей, демонстрирующий лучшее качество на небольших наборах данных.

Автокодировщик (AE)

Цель автокодировщика, представленного на рис. 6.18 (b), – обучиться представлению (кодировке) набора данных, обычно для понижения размерности. Входные и выходные блоки автокодировщика связываются так, что тот реализует нелинейное преобразование в новое представление и обратно, т. е. воздействует на данные приблизительно так же, как тождественное отображение. Поэтому AE можно рассматривать как обобщение методов линейного понижения размерности, например PCA. Потенциально AE может породить нелинейное PCA-представление данных или нелинейное многообразие, в которое следует погрузить данные [71]. Поскольку чаще всего данные расположены в нелинейных подпространствах, автокодировщики являются важным классом НС и имеют много разновидностей. Чаще всего используются три модификации стандартного AE. Вариационный автокодировщик (VAE) [290] (показан на рис. 6.18 (c)) – популярный подход к обучению сложным распределениям без учителя. Если выполняются сильные предположения о распределении латентных переменных, то стандартными алгоритмами градиентного спуска его можно обучить давать хорошие оценки данным – без учителя. *Шумоподавляющий автокодировщик* (denoising auto encoder – DAE) [541] (рис. 6.18 (c)) принимает частично искаженные обучающие входные данные и пытается восстановить исходный сигнал. Шум специально добавляется к входу, чтобы сеть обучилась нелинейному погружению. Наконец, разреженный автокодировщик (sparse auto encoder – SAE) [432] (рис. 6.18 (d)) намеренно увеличивает степень разреженности скрытых блоков на этапе

обучения, притом что скрытых блоков больше, чем входов, чтобы сеть могла обучиться распознаванию полезных структур в данных. Обычно разреженность реализуется пороговым отсечением всех блоков, кроме тех немногих, что дают наибольшую активацию.

Марковская цепь (MC)

Марковская цепь – это стохастическая модель, описывающая последовательность возможных событий, в которой вероятность каждого события зависит только от состояния на момент предыдущего события. Формально она не является нейронной сетью, но обладает многими свойствами рекуррентной НС. Марковские цепи изучаются в стандартных курсах теории вероятностей и математической статистики. На рис. 6.18 (f) показана базовая архитектура, в которой каждый блок соединен с другими, а модель определяет вероятности переходов.

Сеть Хопфилда (HN)

Сеть Хопфилда – это вариант RNN, введенный в обиход Джоном Хопфилдом в 1982 году в рамках изучения человеческой памяти [254]. На рис. 6.18 (g) показана базовая архитектура полносвязной сети, каждый блок которой может играть роль входа. Сеть является обучаемой системой *ассоциативной* памяти, адресуемой по содержимому, с бинарными пороговыми блоками. Входные данные продвигаются по сети, причем гарантируется сходимость к локальному минимуму. Иногда решение сходится к ложному паттерну (неправильному локальному минимуму), а не к хранимому (ожидаемому локальному минимуму).

Машина Больцмана (BM)

Машина Больцмана, которую иногда называют стохастической сетью Хопфилда со скрытыми блоками, – это стохастический порождающий аналог сети Хопфилда. Это была одна из первых нейронных сетей, способных обучаться представлениям и (при наличии достаточного времени) решать трудные комбинаторные задачи [246]. На рис. 6.18 (h) показана структура BM. Заметим, что, в отличие от марковских цепей (не имеющих входных блоков) или сетей Хопфилда (в которых все блоки входные), BM является гибридом, в котором имеются как входные, так и скрытые блоки. Машины Больцмана интуитивно понятны, потому что их поведение напоминает динамику простых физических процессов. Своим названием они обязаны распределению Больцмана в статистической механике, которое используется в их функции выборки.

Ограниченная машина Больцмана (RBM)

Названная Полом Смоленски [493] в 1986 году *Harmonium*, RBM предназначалась для понижения размерности, классификации, коллаборативной фильтрации, обучения признакам и тематического моделирования. Такую сеть можно обучить как с учителем, так и без учителя. Дж. Хинтон поспособствовал популяризации RBM, разработав быстрый алгоритм вычисления [397]. RBM – подкласс BM с одним ограничением: блоки HC должны образовывать двудольный граф (см. рис. 6.18 (e)). Таким образом, связь может существовать только между парами блоков из разных долей (обычно они называются «видимыми» и «скрытыми»), но не между блоками, принадлежащими одной доле. RBM можно использовать в сетях глубокого обучения и в сетях глубокого доверия, расположив их в виде стопки и дополнительно настроив получившуюся глубокую сеть с помощью градиентного спуска и обратного распространения.

Сеть глубокого доверия (DBN)

DBN – порождающая графическая модель, состоящая из нескольких слоев скрытых переменных со связями между слоями, но не между блоками одного слоя [52]. На рис. 6.18 (i) показана архитектура DBN. Обучение DBN можно производить послойно, начиная со слоя AE или RBM. Таким образом, каждый слой должен только обучиться кодировать предшествующую сеть – по сути дела, это жадный алгоритм обучения для нахождения локально оптимальных решений. Поэтому DBN можно рассматривать как композицию простых сетей, обучаемых без учителя, например RBM и AE, в которой скрытый слой каждой подсети играет роль видимого слоя для следующей подсети.

Глубокая сверточная нейронная сеть (DCNN)

DCNN – основной инструмент компьютерного зрения, мы уже рассматривали их в этой главе. В абстрактном виде они представлены на рис. 6.18 (j), а более предметно – на рис. 6.12. Невозможно переоценить их влияние на компьютерное зрение. Первоначально они были разработаны для распознавания документов [325].

Антисверточная сеть (DN)

Антисверточные сети (deconvolutional networks), показанные на рис. 6.18 (k), – это, по сути дела, противоположность DCNN [567]. Структура DN позволяет без учителя строить иерархические представления изображений. Эти пред-

ставления можно использовать как для решения низкоуровневых задач, например очистки от шума, так и для выделения признаков для распознавания объектов. На каждом уровне иерархии информация с предыдущего уровня группируется с целью формирования более сложных признаков, обнаруживаемых при увеличении масштаба изображения. Как и DCNN, антисверточные сети хороши в задачах компьютерного зрения.

Глубокая сверточная сеть обратной графики (DCIGN)

DCIGN (Deep Convolutional Inverse Graphics Network) – разновидность вариационного автокодировщика (VAE), в которой для кодирования и декодирования используются DCNN [313]. Как и в случае AE/VAE/SAE, выходной слой должен соответствовать входному, как показано на рис. 6.18 (l). DCIGN сочетают мощь DCNN с VAE в архитектуре, подходящей для задач компьютерного зрения и обработки изображений.

Порождающая состязательная сеть (GAN)

Придуманная сравнительно недавно, архитектура порождающей состязательной сети (Generative Adversarial Network), показанная на рис. 6.18 (m), позволяет одновременно обучать две сети [217]. Эти сети часто представляют собой комбинацию DCNN с сетями прямого распространения. В процессе обучения одна сеть порождает данные, а вторая пытается их оценивать. Вообще, порождающая сеть обучается отображать латентное пространство в некоторое представляющее интерес распределение данных, тогда как дискриминантная сеть различает примеры, выбранные из истинного распределения, и кандидаты, созданные генератором. Цель обучения порождающей сети – увеличить частоту ошибок дискриминантной сети (т. е. «обмануть» дискриминантную сеть, синтезируя примеры, которые выглядят в точности как настоящие). Благодаря архитектуре GAN удалось получить интересные результаты, связанные с порождением синтетических данных, в т. ч. изображений и фильмов.

Машина неустойчивых состояний (LSM)

LSM (Liquid State Machine), показанная на рис. 6.18 (n), – частный случай импульсной нейронной сети [352]. LSM состоит из большого количества блоков, каждый из которых получает меняющийся со временем входной сигнал от внешних источников (входов), а также от других блоков. Блоки связаны между собой случайным образом. Благодаря рекуррентной природе связей зависящий от времени входной сигнал преобразуется в пространственно-временной паттерн активаций блоков сети. Паттерны активации считыва-

ются линейными дискриминантными блоками. В основе этой архитектуры лежит импульсная деятельность нейронов в мозге. Сеть помогает понять, как импульсные нейроны могут участвовать в обработке и дифференциации информации.

Машина экстремального обучения (ELM)

Обладающая такой же базовой архитектурой, как LSM (рис. 6.18 (n)), ELM (Extreme Learning Machine) является сетью прямого распространения для классификации, регрессии, кластеризации, разреженной аппроксимации, сжатия и обучения признакам. Она состоит из одного или нескольких скрытых слоев, причем параметры скрытых блоков (не только веса их связей с входными блоками) не нужно настраивать. Этим скрытым блокам можно назначить случайные веса, которые никогда не обновляются, или веса можно унаследовать от предков без изменения. В большинстве случаев выходные веса скрытых блоков обучаются за один шаг, что по существу сводится к линейной модели [108].

Сеть с эхо-состояниями (ESN)

ESN (Echo State Network) – это рекуррентная нейронная сеть со скрытым слоем с разреженными связями (обычно количество связей составляет 1 % от числа блоков). Скрытые нейроны обладают памятью, связи и их веса фиксированы и инициализируются случайным образом (см. рис. 6.18 (o)). Таким образом, как и в случае LSM и ELM, они не образуют четко упорядоченной слоистой структуры. Веса выходных нейронов можно обучить, так чтобы сеть генерировала конкретные временные паттерны [263].

Глубокая остаточная сеть (DRN)

DRN (Deep Residual Network) ворвались в мир глубокого обучения, после того исследовательское подразделение компании Microsoft опубликовало статью «Deep Residual Learning for Image Recognition» [237]. Такие сети завоевали первые места во всех пяти основных направлениях конкурсов ImageNet и COCO 2015, в том числе классификация изображения, обнаружение объектов и семантическая сегментация. Робастность сетей ResNet была подтверждена в различных задачах распознавания в машинном зрении и таких областях, как распознавание речи и обработка естественного языка. DRN – это очень глубокие сети прямого распространения, в которых связи существуют не только между соседними слоями, но и между слоями, отстоящими друг от друга на 2–5 слоев. Поэтому входной сигнал передается сразу на отдаленный этап. Глубина сети может достигать 150 слоев, на рис. 6.18 (p) это удалось изобразить лишь условно.

Сеть Кохонена (KN)

Сети Кохонена называются также самоорганизующимися картами признаков [298]. Они вполне конкурентоспособны в задачах классификации данных без учителя. Входные данные подаются на вход KN, как показано на рис. 6.18 (q), после чего сеть оценивает, какие нейроны похожи на вход. Самоорганизующиеся карты отличаются от НС тем, что обучение в них соревновательное, а не основанное на исправлении ошибок (например, с помощью градиентного спуска и обратного распространения), и тем, как используется функция соседства для сохранения топологических свойств пространства входов. Поэтому KN полезны для визуализации многомерных данных в пространстве низкой размерности.

Нейронная машина Тьюринга (NTM)

NTM реализует контроллер на основе НС, взаимодействующий с внешней памятью (см. рис. 6.18 (r)) посредством механизмов внимания [219]. Взаимодействия с памятью дифференцируемы, поэтому их можно оптимизировать методом градиентного спуска. NTM с LSTM-контроллером может выводить простые алгоритмы, например копирование, сортировку и ассоциативный поиск по предъявленным примерам входа и выхода.

Рекомендуемая литература

Учебники

(1) **Deep learning**, by I. Goodfellow, Y. Bengio and A. Courville, 2016 [216].
(2) **Neural networks for pattern recognition**, by C. M. Bishop, 1995 [63].

Статьи и обзоры

(1) **Deep learning**, by Y. LeCun, Y. Bengio and G. Hinton, *Nature*, 2015 [324].
(2) **Understanding deep convolutional networks**, by S. Mallat, *Phil. Trans. R. Soc. A*, 2016 [358].
(3) **Deep learning: mathematics and neuroscience**, by T. Poggio, *Views & Reviews, McGovern Center for Brains, Minds and Machines*, 2016 [430].
(4) **Imagenet classification with deep convolutional neural**, by A. Krizhevsky, I. Sutskever and G. Hinton, *Advances in neural information processing systems*, 2012 [310].

Часть III

ДИНАМИЧЕСКИЕ СИСТЕМЫ И УПРАВЛЕНИЕ

Глава 7

Динамические системы, управляемые данными

Динамические системы предлагают математическую основу для описания окружающего нас мира путем моделирования разнообразных взаимодействий между величинами, совместно изменяющимися во времени. Формально в сферу интереса динамических систем входят анализ, прогнозирование и понимание поведения систем дифференциальных уравнений или итеративных отображений, которые описывают эволюцию состояния системы. Эта формулировка достаточно общая и включает широчайший круг явлений, наблюдаемых в классической механике, электрических схемах, турбулентных потоках жидкости и газа, климатологии, финансах, экологии, социальных системах, нейронауках, эпидемиологии, да и вообще почти в любой системе, изменяющейся во времени.

Изучение динамических систем в современном понимании началось с работы Пуанкаре о хаотическом движении планет. Она уходит корнями в классическую механику и может рассматриваться как кульминация сотен лет математического моделирования, начатого еще Ньютоном и Лейбницем. Полная история динамических систем слишком богата, чтобы уместить ее на нескольких страницах. Эта дисциплина привлекала внимание величайших умов и применялась к бесчисленному множеству областей и трудных задач. Динамические системы – один из самых полных и разветвленных разделов математики, имеющий связи с такими разными областями, как линейная алгебра, дифференциальные уравнения, топология, численный анализ и геометрия. Динамические системы играют центральную роль в моделировании и анализе систем чуть ли не во всех областях техники, физики и наук о жизни.

В настоящее время динамические системы переживают ренессанс в связи с тем, что аналитические выкладки и теоретические обоснования уступили место подходам, основанным на анализе данных. Смыкание больших данных и машинного обучения стало движущей силой смены парадигмы в анализе и осмыслении динамических систем в науке и технике. Данных сколько угодно, а физические законы или описывающие уравнения ускользают от понимания, и так обстоит дело в климатологии, финансах, эпидемиологии и нейронауках. Даже в таких классических областях знаний, как оптика и турбулентные течения, где уравнения известны, исследователи все чаще обращают взоры на анализ, управляемый данными. Многие важнейшие

задачи, например прогнозирование климатических изменений, осмысление процесса познания на основе записей активности нейронов, прогнозирование и купирование распространения заболеваний, а также контроль турбулентности в интересах энергоэффективного производства и транспортировки – первые кандидаты на достижение прогресса в выявлении динамики с помощью анализа данных.

Кроме того, классические геометрический и статистический взгляды на динамические системы дополняются третьим – *теоретико-операторным*, основанным на эволюции измерений системы. Эта так называемая теория *оператора Купмана* базируется на увеличившейся доступности данных измерения сложных систем. Теория Купмана открывает путь к идентификации внутренних систем координат, позволяющих представить нелинейную динамику с помощью линейной модели. Получение линейных представлений существенно нелинейных систем потенциально способно совершить революцию в нашей способности прогнозировать поведение таких систем и управлять ими.

В этой главе излагается современный взгляд на динамические системы, описываются текущие цели и открытые проблемы. Управляемые данными динамические системы – быстро развивающаяся область, поэтому мы будем описывать как давно устоявшиеся, так и вновь появляющиеся методы, которые применяются в актуальных разработках. В частности, нас будут интересовать ключевые проблемы выявления динамики по данным и нахождения представлений, благодаря которым нелинейные системы становятся пригодны для линейного анализа.

7.1. Обзор, мотивация и проблемы

Прежде чем переходить к недавним достижениям в области управляемых данными динамических систем, познакомимся с нотацией и кратко перечислим основные идеи и открытые проблемы.

Динамические системы

В этой главе мы будем рассматривать динамические системы вида

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t; \boldsymbol{\beta}), \tag{7.1}$$

где $\mathbf{x}$ – состояние системы, а $\mathbf{f}$ – векторное поле, которое может зависеть от состояния $\mathbf{x}$, времени t и параметров $\boldsymbol{\beta}$.

Например, рассмотрим уравнения Лоренца [345]

$$\dot{x} = \sigma(y - x), \tag{7.2a}$$

$$\dot{y} = x(\rho - z) - y, \tag{7.2b}$$

$$\dot{z} = xy - \beta z \tag{7.2c}$$

с параметрами $\sigma = 10$, $\rho = 28$, $\beta = 8/3$. Траектория системы Лоренца показана на рис. 7.1. В этом случае вектор состояния $\mathbf{x} = [x \;\; y \;\; z]^T$, а вектор параметров $\boldsymbol{\beta} = [\sigma \;\; \rho \;\; \beta]^T$.

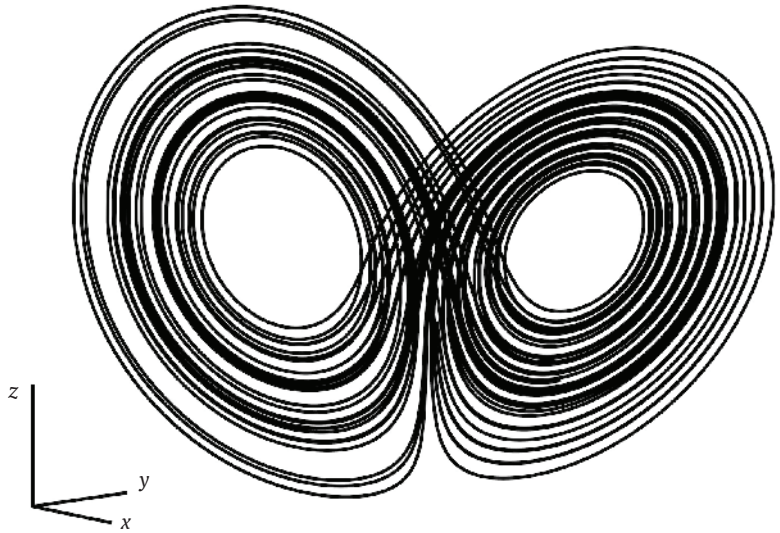

Рис. 7.1 ❖ Хаотическая траектория системы Лоренца (7.2)

Система Лоренца – одна из самых простых и хорошо изученных динамических систем, демонстрирующая хаотическое поведение, т. е. сильную зависимость от начальных условий. Две траектории с близкими начальными условиями быстро расходятся, и по прошествии некоторого времени можно сделать только статистические утверждения.

Динамические системы, подобные системе Лоренца, легко моделируются. Сначала с помощью функции **lorenz** определим векторное поле $\mathbf{f}(\mathbf{x}, t; \boldsymbol{\beta})$:

```
function dx = lorenz(t,x,Beta)
dx = [
Beta(1)*(x(2)-x(1));
x(1)*(Beta(2)-x(3))-x(2);
x(1)*x(2)-Beta(3)*x(3);
];
```

Затем определим параметры системы $\boldsymbol{\beta}$, начальное условие $\mathbf{x}_0$ и промежуток времени:

```
Beta = [10; 28; 8/3]; % параметры системы Лоренца (хаотической)

x0=[0; 1; 20];        % начальное условие
dt = 0.001;
tspan=dt:dt:50;
options = odeset('RelTol',1e-12,'AbsTol',1e-12*ones(1,3));
```

Наконец, решим уравнения с помощью команды **ode45**, которая реализует метод интегрирования Рунге-Кутта четвертого порядка с адаптивным временным шагом:

```
[t,x]=ode45(@(t,x) lorenz(t,x,Beta),tspan,x0,options);
plot3(x(:,1),x(:,2),x(:,3));
```

Часто нам будет встречаться более простая автономная система, не зависящая от времени и не имеющая параметров:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t)). \tag{7.3}$$

В общем случае $\mathbf{x}(t) \in \mathbf{M}$ – n-мерное состояние, расположенное на гладком многообразии $\mathbf{M}$, а $\mathbf{f}$ – элемент касательного к $\mathbf{M}$ пучка $\mathbf{TM}$, т. е. $\mathbf{f}(\mathbf{x}(t)) \in \mathbf{T}_{\mathbf{x}(t)}\mathbf{M}$. Но чаще мы будем рассматривать более простой случай, когда $\mathbf{x}$ – вектор, $\mathbf{M} = \mathbb{R}^n$, а $\mathbf{f}$ – липшицева функция, что гарантирует существование и единственность решения уравнения (7.3). Более общую постановку см. в [1].

Системы с дискретным временем

Мы также будем рассматривать динамическую систему с дискретным временем

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k). \tag{7.4}$$

Такие системы, называемые также *отображениями*, являются более общими, чем непрерывные системы (7.3), они включают еще разрывные и гибридные системы.

Например, рассмотрим логистическое отображение

$$x_{k+1} = \beta x_k(1 - x_k). \tag{7.5}$$

С ростом параметра β притягивающее множество становится все сложнее, как показано на рис. 7.2. Имеет место последовательность бифуркаций с удваивающимся периодом, и в конце концов притягивающее множество становится фрактальным.

Динамика с дискретным временем может быть индуцирована динамикой с непрерывным временем – $\mathbf{x}_k$ получается дискретизацией траектории (7.3) по времени, т. е. $\mathbf{x}_k = \mathbf{x}(k\Delta t)$. Теперь функция распространения с дискретным временем $\mathbf{F}_{\Delta t}$ параметризована временным шагом Δt. Для произвольного момента времени t *потоковое отображение* $\mathbf{F}_t$ определяется формулой

$$\mathbf{F}_t(\mathbf{x}(t_0)) = \mathbf{x}(t_0) + \int_{t_0}^{t_0+t} \mathbf{f}(\mathbf{x}(t))d\tau. \tag{7.6}$$

Системы с дикретным временем зачастую возникают естественно при рассмотрении экспериментальных данных и цифрового управления.

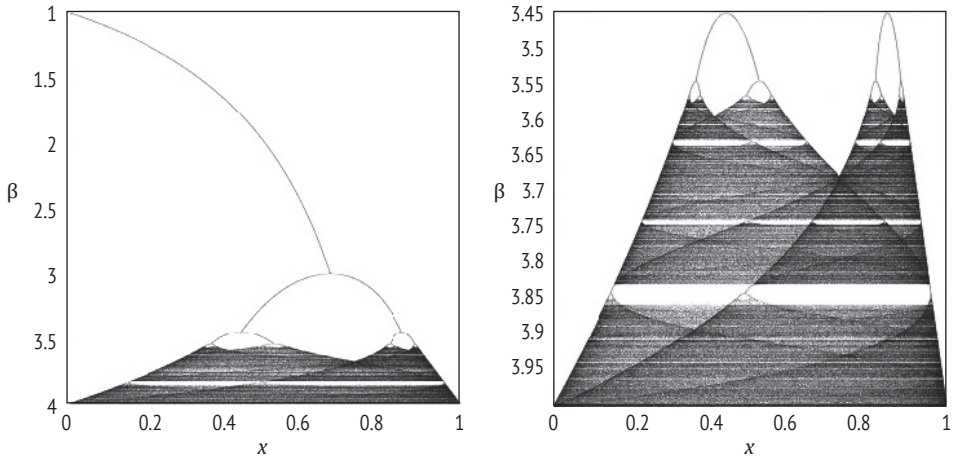

Рис. 7.2 ❖ Притягивающие множества логистического отображения для разных параметров β

Линейная динамика и спектральное разложение

Если есть возможность, желательно иметь дело с линейной динамикой вида

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x}. \tag{7.7}$$

Линейные динамические системы допускают решение в замкнутой форме, и для них разработано много методов анализа, прогнозирования, численного моделирования, оценивания и управления. Решение системы (7.7) имеет вид

$$\mathbf{x}(t_0 + t) = e^{\mathbf{A}t}\mathbf{x}(t_0). \tag{7.8}$$

Динамика полностью описывается собственными значениями и собственными векторами матрицы **A**, т. е. *спектральным разложением* **A**:

$$\mathbf{AT} = \mathbf{T\Lambda}. \tag{7.9}$$

Если **A** имеет n различных собственных значений, то $\mathbf{\Lambda}$ – диагональная матрица, содержащая собственные значения λ_j, а столбцами матрицы **T** являются линейно независимые собственные векторы $\boldsymbol{\xi}_j$, соответствующие собственным значениям λ_j. В таком случае можно написать $\mathbf{A} = \mathbf{T\Lambda T}^{-1}$, и решение (7.8) принимает вид:

$$\mathbf{x}(t_0 + t) = \mathbf{T}e^{\mathbf{\Lambda}t}\mathbf{T}^{-1}\mathbf{x}(t_0). \tag{7.10}$$

В более общем случае, когда собственные значения могут повторяться, матрица $\mathbf{\Lambda}$ состоит из жордановых клеток [427]. Подробный вывод этих соотношений для системы управления см. в разделе 8.2. Заметим, что система с непрерывным временем индуцирует систему с дискретным временем, в которой $\mathbf{F}_t$ определяется решением $\exp(\mathbf{A}t)$ в формуле (7.8). В таком случае собственные значения системы с дискретным временем равны $e^{\lambda t}$.

Матрица $\mathbf{T}^{-1}$ определяет преобразование $\mathbf{z} = \mathbf{T}^{-1}\mathbf{x}$ в систему координат, образованную собственными векторами, в которой динамика становится разделенной:

$$\frac{d}{dt}\mathbf{z} = \mathbf{\Lambda z}. \tag{7.11}$$

Иными словами, каждая координата z_j зависит только от себя, так что динамика описывается простыми уравнениями:

$$\frac{d}{dt}z_j = \lambda_j z_j. \tag{7.12}$$

Таким образом, крайне желательно работать с линейными системами, поскольку подобную систему можно легко преобразовать в систему координат, где динамика разделенная. Для нелинейных систем общего вида не существует ни решения в замкнутой форме, ни простого линейного преобразования координат.

Цели и проблемы современной теории динамических систем

Поскольку динамические системы используются для моделирования физических явлений, можно сформулировать ряд высокоприоритетных целей, относящихся к их анализу.

1. **Предсказание будущего состояния.** Во многих случаях, например в метеорологии и климатологии, мы хотим знать будущее состояние системы. Предсказания на длительный срок по-прежнему остаются проблемой.
2. **Структура и оптимизация.** Требуется подобрать параметры системы для повышения качества работы или устойчивости, например при определении места для размещения рулей ракеты.
3. **Оценка и управление.** Часто имеется возможность активно управлять динамической системой с помощью обратной связи, т. е. использовать результаты измерений системы для подачи сигналов на приводы и тем самым изменять ее поведение. В таком случае требуется оценить полное состояние системы по ограниченному набору измерений.
4. **Возможность интерпретации и понимания физического смысла.** Быть может, самая главная цель динамической системы – интерпретация ее поведения путем анализа траекторий и решений уравнений, описывающих движение.

Реальные системы обычно нелинейны и демонстрируют многомасштабное поведение в пространстве и времени. Следует также помнить о неточности уравнений движения, задания параметров и результатов измерений. Некоторые системы особенно чувствительны к таким неточностям, поэтому приходится использовать вероятностные подходы. Все чаще встречаются

задачи, для которых базовые уравнения движения вообще отсутствуют и вывести их теоретически не представляется возможным.

В этой главе будут рассмотрены недавно разработанные методы идентификации и анализа динамических систем на основе анализа данных. В основном мы будем заниматься двумя главными проблемами динамических систем.

1. **Нелинейность.** Нелинейность остается главным вызовом при анализе и управлении динамическими системами, поскольку приводит к сложной глобальной динамике. Выше мы видели, что линейную систему можно полностью охарактеризовать в терминах спектрального разложения (т. е. собственных значений и собственных векторов) матрицы **A**, что позволяет разработать общие процедуры предсказания, оценивания и управления. Для нелинейных систем такого всеобъемлющего подхода не существует, и его разработка – одна из самых важных математических задач XXI века.
   Наиболее перспективный подход к нелинейным динамическим системам основан на рассмотрении геометрии подпространств, допускающих локальную линеаризацию в окрестности неподвижных точек и периодических орбит, глобальных гетероклинических и гомоклинических орбит, соединяющих эти структуры, и более общих аттракторов [252]. Эта геометрическая теория, восходящая к Пуанкаре, изменила наш взгляд на моделирование сложных систем, а ее успехи многим обязаны таким теоретическим результатам, как теорема Хартмана–Гробмана, которая устанавливает, когда и где возможно аппроксимировать нелинейную систему системой с линейной динамикой. Таким образом, часто мы можем применить все богатство методов линейного анализа в небольшой окрестности неподвижной точки или периодической орбиты. Но хотя геометрический взгляд позволяет количественно оценивать локально линейные модели, глобальный анализ все еще остается в основном качественным и численным, что налагает ограничения на теорию нелинейного предсказания, оценивания и управления вдали от неподвижных точек и периодических орбит.
2. **Неизвестная динамика.** Но, пожалуй, еще более серьезной проблемой является отсутствие известных уравнений, описывающих поведение многих представляющих интерес систем. Исследователи все чаще сталкиваются со сложными системами в нейронауках, эпидемиологии и экологии. Но в этих областях неизвестны *физические законы*, которые теоретически описывали бы принципы, позволяющие вывести уравнения движения. Даже в системах, для которых уравнения известны, например турбулентные течения, укладка белка и горение, пока не удается найти паттерны, которые позволили бы выявить внутренне присущие координаты и основные направления, вдоль которых эволюционирует преобладающее поведение.
   Традиционный подход к анализу физических систем был таким: описать приближение в идеальном случае, а затем вывести простые дифференциальные уравнения на основе второго закона Ньютона. Часто уравнения удавалось существенно упростить, воспользовавшись сим-

метрией и хитроумно выбранными системами координат, что доказывают примеры лагранжевой и гамильтоновой динамики [2, 369]. Но с ростом сложности систем произошла смена парадигмы, и этот классический подход сменился выявлением определяющих уравнений с помощью анализа данных.

Все модели являются приближенными, и чем больше сложность, тем больше подозрений вызывают аппроксимации. Решение о правильности модели становится более субъективным, растет потребность в методах автоматизированного построения модели, которая проливала бы свет на истинные физические механизмы. Часто существуют латентные переменные, которые имеют прямое отношение к динамике, но не были измерены. Выявление этих скрытых эффектов – главная задача управляемых данными методов.

Идентификация неизвестной динамики по данным и обучение внутренним координатам, в которых нелинейные системы становятся линейными, – вот две самые животрепещущие задачи современной теории динамических систем. Их решение обещает перевернуть наши представления о сложных системах и, вероятно, принесет выгоды чуть ли не всем отраслям науки и техники.

На протяжении данной главы мы детально рассмотрим эти проблемы и опишем ряд перспективных методов их решения. В частности, есть два основных подхода к управляемым данными динамическим системам.

1. **Теоретико-операторные представления.** Для решения проблемы нелинейности все чаще применяются теоретико-операторные подходы к динамическим системам. Мы покажем, что нелинейную динамическую систему можно представить в терминах бесконечномерных, но линейных операторов, например оператора Купмана, описанного в разделе 7.4, который экстраполирует функции измерения, и оператора Перрона–Фробениуса, экстраполирующего плотности вероятности и ансамбли траекторий в фазовом пространстве динамической системы.
2. **Регрессия и машинное обучение на данных.** По мере того как объем имеющихся данных возрастает, а исследуемые системы не поддаются теоретическому описанию, регрессия и машинное обучение становятся важнейшими инструментами изучения динамики. Это основа многих описываемых в данной главе методов, в т. ч. разложения по динамическим модам (DMD) в разделе 7.2, разреженной идентификации нелинейной динамики (SINDy) в разделе 7.3, управляемых методов Купмана в разделе 7.5, а также использования генетического программирования для идентификации динамики [68, 477].

Важно отметить, что многие методы и подходы, описанные в этой главе, взаимосвязаны, а продолжающиеся усилия по выявлению новых связей – предмет активных исследований. Упомянем также третью серьезную проблему – высокую размерность многих изучаемых в настоящее время динамических систем, в т. ч. встречающихся в популяционной динамике, моделировании мозга и высокоточных численных решениях дифференциальных уравнений в частных производных. Проблема высокой размерности

подробно обсуждается в последующих главах, посвященных моделям пониженного порядка.

7.2. Разложение по динамическим модам (DMD)

Разложение по динамическим модам (dynamic mode decomposition – DMD) было разработано в статьях Schmid [474, 472] в контексте гидрогазодинамики для идентификации пространственно-временных когерентных структур в данных высокой размерности. DMD основано на собственном ортогональном разложении (proper orthogonal decomposition – POD), в котором используется вычислительно эффективное сингулярное разложение (singular value decomposition – SVD), поэтому оно хорошо масштабируется на системы высокой размерности. В отличие от SVD/POD, которое приводит к иерархии мод, основанной исключительно на пространственной корреляции и энергетическом спектре, при полном игнорировании временной информации, DMD порождает разложение, в котором каждая мода состоит из пространственно-коррелированных структур, обладающих одинаковым линейным поведением во времени (например, колебания на заданной частоте с возрастающей или убывающей амплитудой). Таким образом, DMD не только достигает понижения размерности в смысле уменьшения набора мод, но и дает модель эволюции этих мод во времени.

Вскоре после разработки оригинального алгоритма DMD [474, 472] Роули (Rowley), Мезич (Mezic) и их сотрудники установили важную связь между DMD и теорией Купмана [456] (см. раздел 7.4). DMD можно переформулировать как алгоритм нахождения наилучшей аппроксимации линейной динамической системой, которая экстраполирует измерения высокой размерности в будущее [535]. Таким образом, DMD аппроксимирует оператор Купмана, ограниченный на множество прямых измерений состояния многомерной системы. Эта связь между вычислительно простым линейным алгоритмом DMD и нелинейными динамическими системами вызвала значительный интерес к таким методам [317].

Прошло немного времени, и DMD стал основным алгоритмом управляемой данными характеристики систем высокой размерности. DMD одинаково хорошо подходит для экспериментальных и численных данных, поскольку основан не на знании описывающих уравнений, а только на данных измерений. Алгоритм DMD можно также рассматривать как сочетание удачных сторон SVD (см. главу 1) в части понижения пространственной размерности и БПФ (см. главу 2) в части идентификации временных частот [129, 317]. Каждая мода DMD ассоциируется с некоторым *собственным значением* $\lambda = a + ib$, где b – частота колебаний, а a – скорость роста или затухания.

Существует много вариантов DMD, которые связаны с существующими методами идентификации систем и выделения мод. DMD стал особенно популярен в последние годы в значительной степени благодаря простой численной реализации и тесным связям с нелинейными динамическими систе-

мами через спектральную теорию Купмана. Наконец, DMD – исключительно гибкая платформа как с математической, так и с численной точки зрения; она способствует инновациям в области сжатых измерений, теории управления и многомасштабных методов. Мы обсудим эти связи и расширения в конце данного раздела.

Алгоритм DMD

Было предложено несколько алгоритмов DMD, но здесь мы опишем *точное* DMD, представленное в работе Tu et al. [535]. Если ранние формулировки требовали равномерной по времени выборки из динамики, то этот подход работает и для нерегулярных выборок данных, а также для объединения данных, полученных в разных экспериментах или в процессе численного моделирования. Кроме того, в формулировке Ту с соавторами содержится точное математическое определение DMD, что открывает возможности для получения строгих теоретических результатов. Наконец, точное DMD основано на эффективном и численно хорошо обусловленном сингулярном разложении, как в оригинальной формулировке Шмида [472].

DMD принципиально управляется данными, поэтому первый шаг – собрать пары моментальных снимков состояния системы, изменяющейся во времени. Эти пары снимков можно обозначить $\{(\mathbf{x}(t_k), \mathbf{x}(t'_k)\}_{k=1}^{m}$, где $t'_k = t_k + \Delta t$ и временной шаг Δt достаточно мал для разрешения самых высоких частот в динамике системы. Как и раньше, снимок может быть состоянием системы, например полем скоростей трехмерного течения жидкости, измеренных в нескольких точках, и представлен в виде вектора-столбца высокой размерности. Такие снимки собираются в две матрицы $\mathbf{X}$ и $\mathbf{X}'$:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t_1) & \mathbf{x}(t_2) & \cdots & \mathbf{x}(t_m) \\ | & | & & | \end{bmatrix}; \tag{7.13a}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t'_1) & \mathbf{x}(t'_2) & \cdots & \mathbf{x}(t'_m) \\ | & | & & | \end{bmatrix}. \tag{7.13b}$$

В оригинальной формулировке Schmid [472] и Rowley et al. [456] предполагалась равномерная дискретизация по времени, т. е. $t_k = k\Delta t$ и $t'_k = t_k + \Delta t = t_{k+1}$. Если такое предположение делается, то будем обозначать $\mathbf{x}_k = \mathbf{x}(k\Delta t)$.

Алгоритм DMD ищет спектральное разложение (т. е. собственные значения и собственные векторы) наилучшего аппроксимирующего линейного оператора $\mathbf{A}$, связывающего матрицы двух соседних по времени снимков:

$$\mathbf{X}' \approx \mathbf{A}\mathbf{X}. \tag{7.14}$$

Этот наилучший аппроксимирующий оператор $\mathbf{A}$ определяет линейную динамическую систему, которая наиболее точно экстраполирует измерения

на будущие моменты. Если предположить, что выборка по времени равномерна, то это условие принимает вид

$$\mathbf{x}_{k+1} \approx \mathbf{A}\mathbf{x}_k. \tag{7.15}$$

Математически наилучший аппроксимирующий оператор $\mathbf{A}$ определяется так:

$$\mathbf{A} = \underset{\mathbf{A}}{\operatorname{argmin}} ||\mathbf{X} - \mathbf{A}\mathbf{X}||_F = \mathbf{X}'\mathbf{X}^{\dagger}, \tag{7.16}$$

где $||\cdot||_F$ – норма Фробениуса, а † обозначает операцию псевдообращения. Оптимизированный алгоритм DMD обобщает метод оптимизации точного DMD с целью выполнения регрессии к экспоненциальной временной динамике, тем самым предлагая улучшенный способ вычисления DMD-мод и их собственных значений [20].

Уже сейчас стоит отметить, что матрица $\mathbf{A}$ в (7.15) очень напоминает оператор Купмана в (7.53), если выбрать прямые линейные измерения состояния, т. е. положить $\mathbf{g}(\mathbf{x}) = \mathbf{x}$. Эта связь была первоначально подмечена Роули, Мезичем и их соавторами в работе [456] и тут же вызвала бурный интерес к DMD и теории Купмана. Более подробно мы рассмотрим эти связи ниже.

Для вектора состояния $\mathbf{x} \in \mathbb{R}^n$ матрица $\mathbf{A}$ содержит n^2 элементов, поэтому даже представить этот оператор, не говоря уже о его спектральном разложении, может оказаться невозможно. Поэтому алгоритм DMD и не пытается, а вместо этого применяет понижение размерности для вычисления доминирующих собственных значений и векторов $\mathbf{A}$ без явных вычислений, в которых участвует $\mathbf{A}$. В частности, псевдообратная матрица $\mathbf{X}^{\dagger}$ в (7.16) вычисляется с помощью сингулярного разложения матрицы $\mathbf{X}$. Поскольку в этой матрице столбцов обычно гораздо меньше, чем строк, т. е. $m \ll n$, существует не более m ненулевых сингулярных значений и соответствующих им сингулярных векторов, так что ранг матрицы $\mathbf{A}$ не превышает m. Вместо того чтобы вычислять $\mathbf{A}$ непосредственно, мы вычисляем проекцию $\mathbf{A}$ на эти сингулярные векторы, получая тем самым гораздо меньшую матрицу $\tilde{\mathbf{A}}$ размера не более $m \times m$. Главным вкладом Шмида [472] стала процедура аппроксимации многомерных DMD-мод (собственных векторов $\mathbf{A}$) упрощенной матрицы $\tilde{\mathbf{A}}$ и матрицы данных $\mathbf{X}$, не прибегая к вычислениям над полной $\mathbf{A}$. Позже в работе Tu et al. [535] было доказано, что при выполнении некоторых условий эти приближенные моды на самом деле являются точными собственными векторами полной матрицы $\mathbf{A}$. Таким образом, точный алгоритм DMD, описанный в работе Tu et al. [535], включает следующие шаги.

Шаг 1. Вычислить сингулярное разложение $\mathbf{X}$ (см. главу 1):

$$\mathbf{X} \approx \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*, \tag{7.17}$$

где $\tilde{\mathbf{U}} \in \mathbb{C}^{n \times r}$, $\tilde{\mathbf{\Sigma}} \in \mathbb{C}^{r \times r}$, $\tilde{\mathbf{V}} \in \mathbb{C}^{m \times r}$, а $r \leq m$ – точный или приближенный ранг матрицы $\mathbf{X}$. На практике выбор приближенного ранга r – один из самых важных

и субъективных шагов DMD, да и процедуры понижения размерности вообще. Мы выступаем за алгоритм жесткого порога Гавиша и Донохо [200], который определяет r по зашумленным данным (см. раздел 1.7). Столбцы матрицы $\tilde{\mathbf{U}}$ называются также POD-модами и удовлетворяют условию $\tilde{\mathbf{U}}^*\tilde{\mathbf{U}} = \mathbf{I}$. Аналогично столбцы $\tilde{\mathbf{V}}$ ортонормированы и удовлетворяют условию $\tilde{\mathbf{V}}^*\tilde{\mathbf{V}} = \mathbf{I}$.

Шаг 2. Согласно (7.16), полную матрицу $\mathbf{A}$ можно получить, вычислив псевдообращение $\mathbf{X}$:

$$\mathbf{A} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}^*. \tag{7.18}$$

Однако нас интересуют только первые r собственных значений и собственных векторов $\mathbf{A}$, поэтому мы можем спроецировать $\mathbf{A}$ на POD-моды, находящиеся в $\mathbf{U}$:

$$\tilde{\mathbf{A}} = \tilde{\mathbf{U}}^*\mathbf{A}\tilde{\mathbf{U}} = \tilde{\mathbf{U}}^*\mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}. \tag{7.19}$$

Ключевое наблюдение состоит в том, что упрощенная матрица $\tilde{\mathbf{A}}$ имеет такие же ненулевые собственные значения, как полная матрица $\mathbf{A}$. Поэтому нам нужно только вычислить упрощенную $\tilde{\mathbf{A}}$, а с полной матрицей $\mathbf{A}$ работать никогда не придется. Матрица пониженного порядка $\tilde{\mathbf{A}}$ определяет линейную модель динамики вектора POD-коэффициентов $\tilde{\mathbf{x}}$:

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k. \tag{7.20}$$

Заметим, что матрица $\tilde{\mathbf{U}}$ определяет отображение, позволяющее восстановить полное состояние $\mathbf{x}$ по упрощенному $\tilde{\mathbf{x}}$: $\mathbf{x} = \tilde{\mathbf{U}}\tilde{\mathbf{x}}$.

Шаг 3. Вычисляется спектральное разложение $\tilde{\mathbf{A}}$:

$$\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}. \tag{7.21}$$

Элементами диагональной матрицы $\mathbf{\Lambda}$ являются собственные значения DMD, которым соответствуют собственные значения полной матрицы $\mathbf{A}$. Столбцами $\mathbf{W}$ являются собственные векторы $\tilde{\mathbf{A}}$, они определяют преобразование координат, диагонализирующее матрицу. Эти столбцы можно интерпретировать как линейные комбинации амплитуд POD-мод, которые ведут себя линейно по единому временному паттерну, определяемому λ.

Шаг 4. Многомерные DMD-моды $\mathbf{\Phi}$ восстанавливаются с помощью собственных векторов $\mathbf{W}$ упрощенной системы и матрицы сдвинутого во времени снимка $\mathbf{X}'$:

$$\mathbf{\Phi} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\mathbf{W}. \tag{7.22}$$

Примечательно, что, как показано в работе Tu et al. [535], эти DMD-моды являются собственными векторами полной матрицы $\mathbf{A}$, соответствующими собственным значениям, представленным в $\mathbf{\Lambda}$:

$$\begin{aligned}\mathbf{A}\boldsymbol{\Phi} &= (\mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\underbrace{\tilde{\mathbf{U}}^{*})(\mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}}_{\tilde{\mathbf{A}}}\mathbf{W}) \\ &= \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{A}}\mathbf{W} \\ &= \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\mathbf{W}\boldsymbol{\Lambda} \\ &= \boldsymbol{\Phi}\boldsymbol{\Lambda}.\end{aligned}$$

В оригинальной статье Шмида [472] DMD-моды вычисляются с помощью соотношения $\boldsymbol{\Phi} = \tilde{\mathbf{U}}\mathbf{W}$ и называются *спроецированными модами*; однако не гарантируется, что эти моды являются точными собственными векторами $\mathbf{A}$. Поскольку по определению $\mathbf{A} = \mathbf{X}'\mathbf{X}^{\dagger}$, собственные векторы $\mathbf{A}$ должны принадлежать пространству столбцов $\mathbf{X}'$, как в точном определении DMD, а не пространству столбцов $\mathbf{X}$, как в оригинальном алгоритме DMD. На практике пространства столбцов $\mathbf{X}$ и $\mathbf{X}'$ почти идентичны для динамических систем с низкоранговой структурой, поэтому спроецированные и точные DMD-моды часто очень близки.

Для нахождения DMD-моды, соответствующей нулевому собственному значению $\lambda = 0$, можно воспользоваться точной формулировкой, если только $\boldsymbol{\varphi} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\mathbf{w} \neq 0$. Но если это выражение равно нулю, то следует использовать спроецированную моду $\boldsymbol{\varphi} = \tilde{\mathbf{U}}\mathbf{w}$.

Историческая справка

В оригинальной формулировке матрицы моментальных снимков $\mathbf{X}$ и $\mathbf{X}'$ строились как последовательность снимков с одинаковыми временными интервалами между ними:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix}; \tag{7.23a}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_{m+1} \\ | & | & & | \end{bmatrix}. \tag{7.23b}$$

Таким образом, матрицу $\mathbf{X}$ можно записать в терминах итераций матрицы $\mathbf{A}$:

$$\mathbf{X} \approx \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{A}\mathbf{x}_1 & \cdots & \mathbf{A}^{m-1}\mathbf{x}_1 \\ | & | & & | \end{bmatrix}. \tag{7.24}$$

Следовательно, столбцы матрицы $\mathbf{X}$ принадлежат подпространству Крылова, порожденному матрицей $\mathbf{A}$ и начальным условием $\mathbf{x}_1$. Кроме того, матрицу $\mathbf{X}'$ можно связать с $\mathbf{X}$ оператором *сдвига*:

$$\mathbf{X}' = \mathbf{X}\mathbf{S}, \tag{7.25}$$

где **S** имеет вид

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & a_1 \\ 1 & 0 & 0 & \cdots & 0 & a_2 \\ 0 & 1 & 0 & \cdots & 0 & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & a_m \end{bmatrix}. \tag{7.26}$$

Таким образом, первые $m - 1$ столбцов $\mathbf{X}'$ получаются просто сдвигом соответствующих столбцов **X**, а последний столбец является наилучшей комбинацией m столбцов **X**, которая доставляет минимум невязке. В таком виде алгоритм DMD напоминает алгоритм Арнольди итеративного нахождения доминирующих собственных значений и собственных векторов матрицы **A**. Матрица **S** имеет те же собственные значения, что полная матрица **A**, поэтому разложение **S** можно использовать для нахождения динамических мод и собственных значений. Однако вычисления с **S** численно не так устойчивы, как описанный выше точный алгоритм.

Спектральное разложение и DMD-продолжение

Одно из самых важных свойств DMD – способность представить состояние системы в терминах управляемого данными разложения:

$$\mathbf{x}_k = \sum_{j=1}^{r} \boldsymbol{\varphi}_j \lambda_j^{k-1} b_j = \boldsymbol{\Phi}\boldsymbol{\Lambda}^{k-1}\mathbf{b}, \tag{7.27}$$

где $\boldsymbol{\varphi}_j$ – DMD-моды (собственные векторы матрицы **A**), λ_j – собственные значения DMD (собственные значения матрицы **A**), а b_j – амплитуда моды. Вектор **b** амплитуд мод в общем случае вычисляется как

$$\mathbf{b} = \boldsymbol{\Phi}^{\dagger}\mathbf{x}_1. \tag{7.28}$$

Рассматривались и менее субъективные подходы к выбору доминирующих и разреженных мод [129, 270]. Однако вычисление амплитуд мод в общем случае обходится очень дорого, даже если использовать прямолинейное определение (7.28). Вместо этого часто можно вычислить амплитуды с помощью спроецированных данных POD:

$$\mathbf{x}_1 = \boldsymbol{\Phi}\mathbf{b} \tag{7.29a}$$

$$\Rightarrow \quad \tilde{\mathbf{U}}\tilde{\mathbf{x}}_1 = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\mathbf{W}\mathbf{b} \tag{7.29b}$$

$$\Rightarrow \quad \tilde{\mathbf{x}}_1 = \tilde{\mathbf{U}}^{*}\mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\mathbf{W}\mathbf{b} \tag{7.29b}$$

$$\Rightarrow \quad \tilde{\mathbf{x}}_1 = \tilde{\mathbf{A}}\mathbf{W}\mathbf{b} \tag{7.29d}$$

$$\Rightarrow \quad \tilde{\mathbf{x}}_1 = \mathbf{W}\boldsymbol{\Lambda}\mathbf{b} \tag{7.29e}$$

$$\Rightarrow \quad \mathbf{b} = (\mathbf{W}\boldsymbol{\Lambda})^{-1}\tilde{\mathbf{x}}_1. \tag{7.29f}$$

Обе матрицы **W** и **Λ** имеют размер $r \times r$, в отличие от большой матрицы **Φ** размера $n \times r$.

Показанное выше спектральное разложение можно записать также для системы с непрерывным временем, если ввести непрерывные собственные значения $\omega = \log(\lambda)/\Delta t$:

$$\mathbf{x}(t) = \sum_{j=1}^{r} \boldsymbol{\varphi}_j e^{\omega_j t} b_j = \boldsymbol{\Phi} \exp(\boldsymbol{\Omega} t)\mathbf{b}, \tag{7.30}$$

где $\boldsymbol{\Omega}$ – диагональная матрица, содержащая непрерывные собственные значения ω_j.

Пример и код

Ниже приведен базовый код алгоритма DMD.

```
function [Phi, Lambda, b] = DMD(X,Xprime,r)

[U,Sigma,V] = svd(X,'econ');   % шаг 1
Ur = U(:,1:r);
Sigmar = Sigma(1:r,1:r);
Vr = V(:,1:r);

Atilde = Ur'*Xprime*Vr/Sigmar; % шаг 2
[W,Lambda] = eig(Atilde);      % шаг 3

Phi = Xprime*(Vr/Sigmar)*W;    % шаг 4
alpha1 = Sigmar*Vr(1,:)';
b = (W*Lambda)\alpha1;
```

Применение этого кода продемонстрировано на рис. 7.3 на примере потока жидкости, обтекающего цилиндр круглого сечения, в предположении, что число Рейнольдса потока равно 100. Двумерные уравнения Навье–Стокса решаются методом проекции погруженной границы (immersed boundary projection method – IBPM)[1], основанным на быстром многодоменном методе Тайры–Колониуса [511, 135]. Данные, необходимые для этого примера, можно скачать, не выполняя код, с сайта dmdbook.com.

Имея данные, легко вычислить разложение по динамическим модам:

```
% VORTALL содержит векторные поля потока, представленные в виде
% векторов-столбцов
X = VORTALL;
[Phi, Lambda, b] = DMD(X(:,1:end-1),X(:,2:end),21);
```

Расширения, приложения и ограничения

Одно из главных преимуществ разложения по динамическим модам – простое выражение в терминах линейной регрессии. DMD не требует знания описывающих уравнений. Поэтому алгоритм DMD очень быстро был расширен

[1] Код IBPM опубликован по адресу https://github.com/cwrowley/ibpm.

путем включения ряда методологических новшеств и применяется далеко за пределами гидродинамики [317], в интересах которой был придуман. Мы представим несколько самых известных расширений алгоритма и перечислим многообещающие прикладные области. Также мы расскажем о текущих ограничениях теории DMD, которые должны быть сняты в ходе будущих исследований.

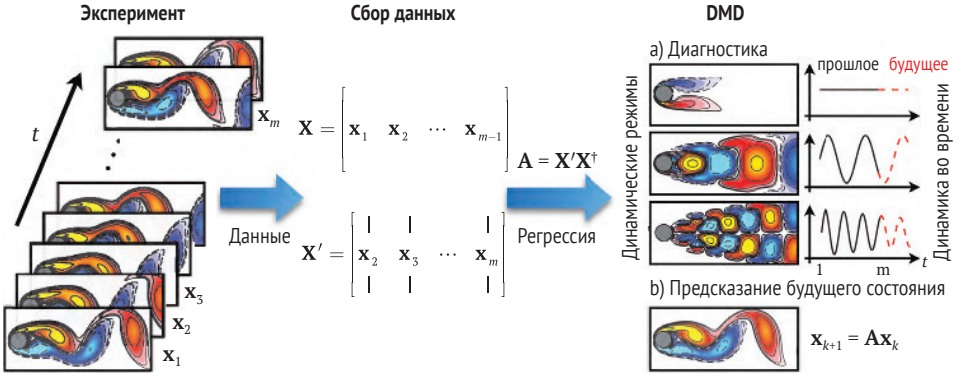

Рис. 7.3 ❖ Иллюстрация алгоритма DMD на примере потока жидкости, обтекающего цилиндр круглого сечения, число Рейнольдса равно 100. *Взято из работы [317]*

Методологические расширения

- **Сжатие и рандомизированная линейная алгебра.** DMD изначально разрабатывался для обработки многомерных наборов данных в гидродинамике, в частности поля скорости или завихренности жидкости. Такие данные могут иметь миллионы степеней свободы. Однако из того факта, что DMD часто выявляет структуру низкой размерности в многомерных данных, следует, что могут существовать более эффективные стратегии измерений и вычислений, опирающиеся на принцип *разреженности* (см. главу 3). Было предложено несколько независимых и весьма успешных расширений и модификаций DMD, в которых используется низкоранговая структура и разреженность.

  В 2014 году в работе Jovanovic et al. [270] была применена поощряющая разреженность оптимизация для идентификации наименьшего числа DMD-мод, необходимого для описания набора данных, по существу это свелось к нахождению нескольких доминирующих амплитуд DMD-мод в векторе **b**. Альтернативный подход – сравнение всех подмножеств множества DMD-мод – вычислительно неразрешимая задача.
  Еще одно направление исследований основывается на том факте, что DMD-моды обычно допускают разреженное представление в базисе Фурье или вейвлетном базисе. Кроме того, временная динамика каждой моды – это чистые гармоники, т. е. как раз квинтэссенция разреженности в базисе Фурье. Эта разреженность позволяет сформулировать

несколько эффективных стратегий сжатых измерений, при которых сокращается число измерений во времени [536] и в пространстве [96, 225, 174]. Это открывает интересные возможности для описания систем с высокой разрешающей способностью при наличии измерений с недостаточным разрешением.

В последнее время для ускорения вычислений DMD при наличии данных о полном состоянии стала использоваться рандомизированная линейная алгебра, тесно связанная со сжатыми измерениями. Вместо того чтобы делать подвыборку измерений и использовать сжатые измерения для выявления высокоразмерных структур, рандомизированные методы начинают работу с полными данными, а затем случайным образом проецируют данные на подпространство меньшей размерности, в котором вычисления можно выполнить более эффективно. В работе Bistrian and Navon [66] удалось ускорить DMD благодаря использованию рандомизированного сингулярного разложения, а в работе Erichson et al. [175] показано, как выполнить все тяжелые вычисления DMD в подпространстве проекции.

Наконец, использовались также библиотеки DMD-мод для идентификации динамических режимов [308] на основе разреженного представления для классификации [560] (см. раздел 3.6), которое раньше применялось для идентификации динамических режимов с помощью библиотек POD-мод [80, 98].

- **Входы и управление.** Главная сильная сторона DMD – способность описывать сложные динамические системы высокой размерности в терминах небольшого числа доминирующих мод, которые представляют пространственно-временные когерентные структуры. Благодаря понижению размерности с n (часто миллионы и даже миллиарды) до r (десятки или сотни) удается делать предсказания и оценки быстрее. А раз задержка предсказания мала, то устройства управления оказываются производительнее и надежнее. Поэтому уже давно велись поиски компактных и эффективных представлений сложных систем, в частности течений жидкости, что в конце концов привело к новой дисциплине – моделирование пониженного порядка. Однако оригинальный алгоритм DMD предназначался для описания естественно эволюционирующих систем, без учета влияния управляющих воздействий.

  Вскоре после изобретения оригинального алгоритма DMD Проктор с соавторами [434] модифицировали его, чтобы провести различие между естественной динамикой и последствиями управления. Это сводится к решению обобщенного уравнения эволюции

$$\mathbf{x}_{k+1} \approx \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \tag{7.31}$$

  что приводит еще к одной задаче линейной регрессии (см. раздел 10.1).

  Изначально побудительным мотивом для разработки DMD с управлением (DMDc) было использование алгоритма DMD для описания эпидемиологических систем (например, распространения малярии по континенту), когда невозможно исключить эффекты вмешательства,

например вакцинацию и надкроватные сетки, чтобы описать естественную динамику [433].

Спустя некоторое время после появления оригинального алгоритма DMDc он был объединен с DMD со сжатыми измерениями, в результате чего появилась новая схема сжимающей идентификации систем (compressive system identification) [30]. В этой схеме стало возможно собрать измерения управляемой системы с грубым шагом дискретизации и на их основе построить точную и эффективную модель низкого порядка с применением DMD и алгоритма реализации собственной системы (eigensystem realization algorithm – ERA, см. раздел 9.3) [272].

DMDc-модели, основанные на линейных и нелинейных измерениях системы, недавно стали использоваться для управления с прогнозирующими моделями (model predictive control – MPC), что позволило улучшить управление нелинейными системами (Korda and Mezić [302]). Системы управления с прогнозирующими DMDc-моделями впоследствии сравнивались с MPC, основанным на полностью нелинейных моделях, в работе Kaiser et al. [277], и показали себя на удивление хорошо, даже когда рассматривались существенно нелинейные системы.

- **Нелинейные измерения.** Интерес к DMD в значительной мере продиктован сильными связями с нелинейной динамикой посредством оператора Купмана [456]. Действительно, DMD позволяет точно охарактеризовать периодическое и квазипериодическое поведение даже в нелинейных системах, если только собрано достаточное количество данных. Однако в базовом алгоритме DMD используются линейные измерения системы, которые недостаточно полны для характеризации существенно нелинейных явлений, например переходных режимов, нерегулярных эффектов или широкополосных помех. В работе Williams et al. [556] DMD-измерения были дополнены нелинейными измерениями системы, что обогатило базис, используемый для представления оператора Купмана. Цель так называемого расширенного алгоритма DMD (eDMD) – построить линейную модель $\mathbf{A}_\mathbf{Y}$, экстраполирущую нелинейные измерения $\mathbf{y} = \mathbf{g}(\mathbf{x})$:

$$\mathbf{y}_{k+1} \approx \mathbf{A}_\mathbf{Y}\mathbf{y}_k. \tag{7.32}$$

Для систем высокой размерности это дополненное состояние $\mathbf{y}$ может оказаться слишком большим для вычислений, что вынуждает использовать ядерные методы для аппроксимации эволюционного оператора $\mathbf{A}_\mathbf{Y}$ [557]. Этот ядерный алгоритм DMD впоследствии был расширен путем включения методов обучения словаря [332].

Недавно было показано, что eDMD эквивалентен вариационному подходу из области конформационной динамики (VAC) [405, 407, 408], который впервые был предложен Ноэ (Noé) и Нюске (Nüske) в 2013 году для моделирования молекулярной динамики в широком спектре масштабов времени. Дополнительные связи между eDMD и VAC и между DMD и методом независимых компонент с задержкой (time lagged independent component analysis – TICA) исследованы в недавнем обзоре

[293]. Основной вклад VAC – вариационная оценка, позволяющая объективно оценить модели Купмана с помощью перекрестной проверки. Вслед за расширенным DMD было показано, что существуют довольно ограничительные условия, при которых можно получить модель линейной регрессии, включающую исходное состояние системы [92]. Для нелинейных систем с несколькими неподвижными точками, периодическими орбитами и другими притягивающими структурами не существует конечномерной линейной системы, которая включала бы состояние **x**, топологически сопряженное нелинейной системе. Вместо этого важно идентифицировать инвариантные относительно оператора Купмана подпространства, натянутые на собственные функции оператора Купмана; в общем случае невозможно непосредственно выразить состояние **x** через линейную оболочку этих собственных векторов, хотя иногда возможно идентифицировать **x** с помощью единственного обратного элемента. Практический алгоритм идентификации собственных функций предложен в работе Kaiser et al. [276].

- **Очистка от шума.** Алгоритм DMD целиком управляется данными, поэтому он равным образом применим к экспериментальным и численным данным. Если DMD используется для характеризации экспериментальных данных, то следует учитывать эффекты шума в датчиках и стохастические возмущения. Оригинальный алгоритм DMD очень чувствителен к шуму, и было показано, что он вносит в распределение собственных значений значительную систематическую ошибку [164, 28, 147, 241]. Хотя увеличение выборки уменьшает дисперсию распределения собственных значений, полностью устранить смещение она не в силах [241].

  Есть несколько подходов, позволяющих скорректировать влияние шума в датчиках и возмущений. В работе Hemati et al. [241] для учета влияния потенциально зашумленных измерений и возмущений на состояние используется регрессия наименьших полных квадратов вместо обычной регрессии наименьших квадратов, как в оригинальной версии. В работе Dawson et al. [147] DMD вычисляется по прошлым и будущим данным, а затем результирующий оператор усредняется, чтобы устранить систематическую ошибку. В этой работе также имеется великолепное обсуждение источников шума и сравнение различных алгоритмов очистки от шума.

  Позже в работе Askham and Kutz [20] был предложен оптимизированный алгоритм DMD, в котором используется метод переменной проекции для нелинейных наименьших квадратов и вычисляется DMD с неравномерной по времени выборкой, что существенно снижает ошибку, обусловленную шумом. Алгоритм DMD на подпространстве, описанный в работе Takeishi et al. [514], также компенсирует измерительный шум; в нем вычисляется ортогональная проекция будущих снимков на пространство предыдущих снимков, а затем строится линейная модель. Разработаны также расширения, сочетающие DMD с байесовскими подходами [512].
- **Многомасштабность.** DMD часто применяется к сложным многомерным динамическим системам, например в области турбулент-

ных течений жидкости или эпидемиологии, для которых характерна многомасштабная динамика как в пространстве, так и во времени. Во многих многомасштабных системах имеют место переходные или нерегулярные явления, примером может служить феномен Эль Ниньо в глобальных климатических данных. Такая переходная динамика плохо улавливается алгоритмом DMD, который ищет пространственно-временные моды, которые были бы глобально когерентны во всем временном ряде данных. Чтобы справиться с этой трудностью, был предложен многомасштабный алгоритм DMD (mrDMD) [318], который, по сути дела, разлагает динамику по разным временным шкалам, изолируя переходные и нерегулярные паттерны. Недавно было показано (Manohar et al. [367]), что многомасштабные DMD-моды дают преимущества при разреженном размещении датчиков.

- **Запаздывающие измерения.** Хотя DMD разрабатывался для данных высокой размерности в предположении, что имеется доступ к полному состоянию системы, часто желательно охарактеризовать пространственно-временные когерентные структуры при наличии неполных измерений. В качестве крайнего проявления рассмотрим единственное измерение, которое колеблется по синусоидальному закону, $x(t) = \sin(\omega t)$. Хотя может показаться, что это идеальный кандидат для DMD, алгоритм неправильно находит вещественное собственное значение, т. к. ранг данных недостаточен для выделения комплексно-сопряженной пары собственных значений $\pm i\omega$. Этот парадокс впервые был исследован в работе Tu et al. [535], где было найдено решение – собрать запаздывающие измерения в бóльшую матрицу, чтобы увеличить ранг данных и извлечь информацию о фазе. Измерение запаздывающих координат нашло также применение для выделения когерентных паттернов в записях нейронной активности [90]. Связи между DMD и теорией Купмана [91, 18, 144] будут подробнее обсуждаться в разделе 7.5.
- **Потоковая и параллельная обработка.** Из-за большого объема вычислений в алгоритме DMD для данных высокой размерности были предприняты усилия по ускорению DMD путем потоковой обработки и распараллеливания. DMD часто используется в потоковых приложениях, когда скользящее окно снимков обрабатывается непрерывно; это приводит к избыточным вычислениям при поступлении новых данных. Существует несколько алгоритмов потокового DMD, основанных на инкрементном SVD [242], потоковом методе моментальных снимков SVD [424] и обновлениях ранга 1 матрицы DMD [569]. Кроме того, алгоритм DMD легко распараллеливается, поскольку основан на SVD. Имеется несколько распараллеленных версий кода, основанных на QR [466] и SVD [175, 177, 176].

Приложения

- **Гидрогазодинамика.** DMD было первоначально предложено в контексте задач гидрогазодинамики [472] и с тех применялось к самым разным геометриям потока (струйное течение, кавитационное об-

текание, вихревой след, течение в канале, пограничные слои и т. д.), для изучения микширования в акустике, процессов горения и других явлений. В оригинальных статьях Шмида [474, 472] рассматривались кавитационное обтекание и струйное течение. В оригинальной работе Rowley et al. [456] исследовалась струя в поперечном потоке. Неудивительно, что впоследствии DMD широко использовалось для изучения как кавитационного обтекания [472, 350, 481, 43, 42], так и струйных течений [473, 49, 483, 475].

DMD также применялось к изучению течений в спутной струе, в т. ч. захвата частоты [534], вихря за щитком Герни [415], вихря при обтекании цилиндра [28] и динамического срыва потока [166]. Пограничные слои также активно изучались с помощью DMD [411, 465, 383]. В акустике DMD использовалось для изучения акустики ближнего и дальнего полей, которая приводит к неустойчивости, наблюдаемой в сдвиговых течениях [495]. При изучении горения DMD позволило понять механизм когерентного выделения тепла в турбулентном закрученном факеле [387] и пранализировать камеру сгорания ракетного двигателя [258]. DMD использовалось также для анализа аномальных механизмов роста в термоакустических взаимодействиях в трубке Рийке. Имеются сравнения DMD и POD при изучении реагирующих потоков [459]. DMD применялось и для анализа более экзотических течений, в т. ч. для моделирования высокоскоростного поезда [392]. Изучалось также взаимодействие со скачком уплотнения в турбулентном пограничном слое (shock turbulent boundary layer interaction – STBLI), и DMD использовалось для идентификации пульсирующей области срыва потока, сопровождающей движение ударной волны [222]. Есть пример применения DMD к изучению самовозбуждающихся колебаний в детонационных волнах [373]. Из других задач упомянем идентификацию шпилькообразных вихрей [516], декомпозицию течения за расположенным на поверхности кубом [393], решение уравнений мелкой воды [65], изучение наноструйных течений за прямой призмой [463] и измерение скорости роста неустойчивостей в кольцеобразных жидких пленках [163].

- **Эпидемиология.** Недавно DMD было применено к исследованию эпидемиологических систем в работе Proctor and Eckhoff [435]. Это приложение допускает особенно хорошую интерпретацию, потому что модальные частоты часто соответствуют годовым или сезонным колебаниям. Кроме того, фазы DMD-мод проливают свет на то, как фронт заболевания распространяется в пространстве, что потенциально может направить усилия по борьбе с эпидемией в нужное направление. Применение DMD к изучению заболеваний стало также побудительной причиной для разработки DMD с управлением [434], поскольку нереально прекратить вакцинацию для изучения естественной динамики.
- **Нейронауки.** Благодаря развитию оборудования, отодвигающему границы все дальше, сложные сигналы, получаемые при записи нейронной активности, становятся все более точными, а их размерность повышается. DMD потенциально способно преобразить анализ таких

записей, в пользу чего свидетельствует недавнее исследование, в котором были идентифицированы динамически релевантные признаки в данных электрокохлеографии (ECOG) спящих пациентов [90]. Впоследствии появилось несколько работ по применению DMD к записям нейронной активности и предложения по реализации соответствующего оборудования [3, 85, 520].

- **Обработка видео.** Разделение объектов переднего и заднего плана на видео – типичная задача в приложениях для видеонаблюдения. Разделение в режиме реального времени – проблема, которая еще больше осложняется из-за постоянно увеличивающегося разрешения видео. DMD предлагает гибкую платформу для разделения, поскольку задний план можно аппроксимировать DMD-модой с нулевым собственным значением [223, 174, 424].
- **Другие приложения.** Круг задач, к которым применяется DMD, постоянно растет и включает уже робототехнику [56], финансы [363] и физику плазмы [517]. Ожидается, что в будущем эта тенденция будет только усиливаться.

Проблемы

- **Распространяющиеся волны.** DMD основано на сингулярном разложении матрицы $\mathbf{X} = \mathbf{U\Sigma V}^*$, столбцами которой являются пространственные измерения, изменяющиеся с течением времени. В этом случае SVD служит для разделения пространственно-временных переменных на пространственные моды, определяемые столбцами **U**, и временную динамику, определяемую столбцами **V**. Поэтому, как и POD, DMD сталкивается с трудностями в задачах, где имеются распространяющиеся волны, для которых разделение переменных заведомо невозможно.
- **Переходные явления.** Во многих интересных системах имеются переходные режимы и нерегулярные явления. Было предложено несколько методов для идентификации таких событий, в т. ч. многомасштабное DMD и использование запаздывающих координат. Но все равно необходимо формализовать выбор релевантных временных масштабов и размер окна для вычисления DMD.
- **Непрерывный спектр.** С предыдущей проблемой связана еще одна: многие системы характеризуются широкополосным частотным составом, а не немногими дискретными частотами. Такой частотный состав называется непрерывным спектром, т. е. наблюдаются все частоты в непрерывном диапазоне. Например, спектр простого маятника непрерывный, потому что у системы имеется собственная частота малых колебаний, и эта частота непрерывно изменяется и замедляется по мере добавления энергии маятнику. В качестве других примеров можно привести нелинейную оптику и широкополосную турбулентность. Эти системы представляют серьезную трудность для DMD, поскольку количество мод в них велико, пусть даже динамика, скорее всего, порождается нелинейными взаимодействиями немногих доминирующих мод. В последнее время было предложено несколько подходов к системам

с непрерывным спектром. Показано, что применение DMD к вектору запаздывающих измерений системы, так называемый *HAVOK*-анализ, описываемый в разделе 7.5, аппроксимирует динамику таких хаотических систем, как уравнения Лоренца, спектр которых непрерывен. Кроме того, в работе Lusch et al. [349] показано, что можно спроектировать архитектуру глубокого обучения со вспомогательной сетью для параметризации непрерывной частоты.

- **Существенная нелинейность и выбор измерений.** Несмотря на то что в применении DMD к нелинейным системам достигнут значительный прогресс [557], выбор нелинейных измерений по-прежнему не является точной наукой. Идентификация подпространств измерений, замкнутых относительно оператора Купмана, – открытая проблема [92]. Недавний прогресс в области глубокого обучения потенциально может привести к нахождению представления очень сложных собственных функций на основе данных [550, 368, 513, 564, 412, 349].

7.3. Разреженная идентификация нелинейной динамики (SINDy)

Построение моделей динамических систем по данным – главная проблема математической физики, богатая история которой восходит к временам Кеплера и Ньютона и открытию законов движения планет. Исторически этот процесс опирался на сочетание высококачественных измерений и интуиции ученого. При наличии огромного объема данных и постоянно растущих вычислительных ресурсов стала возможной новая будоражащая ум научная парадигма – *автоматизированное* открытие описывающих уравнений и динамических систем.

Обычно вид потенциальной модели либо ограничен априорными знаниями об описывающих уравнениях, как в случае проекционного метода Галеркина [402, 455, 471, 404, 119, 549, 32, 118] (см. главу 12), либо определяется путем тестирования нескольких эвристических моделей с последующей оптимизацией параметров для подгонки к данным. Альтернативно можно получить линейные модели, дающие наилучшую аппроксимацию, с помощью методов DMD или ERA. Одновременная идентификация нелинейной структуры и параметров модели на основе данных – значительно более трудная задача, поскольку количество возможных структур модели растет комбинаторно.

Алгоритм разреженной идентификации нелинейной динамики (sparse identification of nonlinear dynamics – SINDy) [95] избегает практически нереализуемого комбинаторного перебора всевозможных структур модели, пользуясь тем фактом, что для многих динамических систем

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}) \tag{7.33}$$

динамика $\mathbf{f}$ содержит всего несколько ненулевых членов в пространстве возможных функций в правой части; например, в уравнениях Лоренца (7.2) во взаимодействии принимают участие только линейные и квадратичные члены.

Таким образом, мы ищем аппроксимацию $\mathbf{f}$ обобщенной линейной моделью

$$\mathbf{f}(\mathbf{x}) \approx \sum_{k=1}^{p} \theta_k(\mathbf{x})\xi_k = \boldsymbol{\Theta}(\mathbf{x})\boldsymbol{\xi}, \tag{7.34}$$

с наименьшим возможным числом ненулевых элементов $\boldsymbol{\xi}$. Тогда можно найти релевантные активные члены динамики с помощью разреженной регрессии [518, 573, 236, 264], которая штрафует за количество членов и хорошо масштабируется на крупномасштабные задачи.

Сначала мы собираем временной ряд для системы (7.33) и организуем данные в виде матрицы:

$$\mathbf{X} = [\mathbf{x}(t_1)\ \ \mathbf{x}(t_2)\ \ \cdots\ \ \mathbf{x}(t_m)]^T. \tag{7.35}$$

Сформируем аналогичную матрицу производных:

$$\dot{\mathbf{X}} = [\dot{\mathbf{x}}(t_1)\ \ \dot{\mathbf{x}}(t_2)\ \ \cdots\ \ \dot{\mathbf{x}}(t_m)]^T. \tag{7.36}$$

На практике ее можно вычислить непосредственно по данным в $\mathbf{X}$; для зашумленных данных регуляризация полной вариации обычно дает численно устойчивые производные [125]. Можно также сформулировать алгоритм SINDy для систем с дискретным временем $\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k)$, как в алгоритме DMD, и вообще избежать вычисления производных.

По данным в $\mathbf{X}$ можно построить библиотеку нелинейных функций-кандидатов $\boldsymbol{\Theta}(\mathbf{X})$:

$$\boldsymbol{\Theta}(\mathbf{X}) = [\mathbf{1}\ \ \mathbf{X}\ \ \mathbf{X}^2\ \ \cdots\ \ \mathbf{X}^d\ \ \cdots\ \ \sin(\mathbf{X})\ \ \cdots]. \tag{7.37}$$

Здесь $\mathbf{X}^d$ – матрица, столбцами которой являются векторы, образованные всеми возможными временными рядами полиномов степени d от состояния $\mathbf{x}$. Вообще говоря, состав этой библиотеки функций-кандидатов ограничен только вашим воображением.

Динамическую систему (7.33) теперь можно представить в терминах матриц (7.36) и (7.37) следующим образом:

$$\dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}. \tag{7.38}$$

Каждый столбец $\boldsymbol{\xi}_k$ матрицы $\boldsymbol{\Xi}$ – это вектор коэффициентов, определяющий ненулевые члены в k-й строке (7.33). Экономная модель даст хорошую аппроксимацию (7.38) с наименьшим возможным числом членов в $\boldsymbol{\Xi}$. Такую модель можно найти с помощью выпуклой разреженной регрессии с регуляризацией по норме ℓ_1:

$$\boldsymbol{\xi}_k = \operatorname{argmin}_{\boldsymbol{\xi}'_k} ||\dot{\mathbf{X}}_k - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\xi}'_k||_2 + \lambda||\boldsymbol{\xi}'_k||_1. \tag{7.39}$$

Здесь $\dot{\mathbf{X}}_k$ – k-й столбец $\dot{\mathbf{X}}$, а λ – параметр, поощряющий разреженность. Разреженная регрессия, например алгоритм LASSO [518] или последовательный метод наименьших квадратов с порогом (sequential thresholded least-squares – STLS), используемый в SINDy [95], улучшает численную устойчивость этой идентификации для зашумленных переопределенных задач – в отличие от предшествующих методов [548], в которых применялись сжатые измерения [150, 109, 112, 111, 113, 39, 529]. Мы рекомендуем использовать для выбора ненулевых членов метод STLS (листинг 7.1).

Листинг 7.1 ❖ Последовательный метод наименьших квадратов с порогом

```
function Xi = sparsifyDynamics(Theta,dXdt,lambda,n)
% Вычислить разреженную регрессию: последовательный метод наименьших квадратов
Xi = Theta\dXdt; % начальное приближение: наименьшие квадраты

% Lambda управляет разреженностью
for k=1:10
    smallinds = (abs(Xi)<lambda); % найти малые коэффициенты
    Xi(smallinds)=0;              % и порог
    for ind = 1:n                 % n - размерность состояния
        biginds = ~smallinds(:,ind);
        % Регрессия для нахождения разреженноого Xi
        Xi(biginds,ind) = Theta(:,biginds)\dXdt(:,ind);
    end
end
```

Из разреженных векторов $\boldsymbol{\xi}_k$ можно синтезировать динамическую систему:

$$\dot{x}_k = \boldsymbol{\Theta}(\mathbf{x})\boldsymbol{\xi}_k. \tag{7.40}$$

Заметим, что x_k – k-й элемент $\mathbf{x}$, а $\boldsymbol{\Theta}(\mathbf{x})$ – вектор-строка символических функций от $\mathbf{x}$, в отличие от матрицы данных $\boldsymbol{\Theta}(\mathbf{X})$. На рис. 7.4 показано, как можно использовать алгоритм SINDy для выведения уравнений Лоренца из данных. Программа в листинге 7.2 генерирует данные и выполняет SINDy-регрессию для системы Лоренца.

Листинг 7.2 ❖ SINDy-регрессия для идентификации системы Лоренца по данным

```
%% Сгенерировать данные
Beta = [10; 28; 8/3]; % параметры Лоренца (хаотические)
n = 3;
x0=[-8; 8; 27]; % начальное условие
tspan=[.01:.01:50];
options = odeset('RelTol',1e-12,'AbsTol',1e-12*ones(1,n));
[t,x]=ode45(@(t,x) lorenz(t,x,Beta),tspan,x0,options);

%% Вычислить производную
for i=1:length(x)
    dx(i,:) = lorenz(0,x(i,:),Beta);
end

%% Построить библиотеку и вычислить разреженную регрессию
```

```
Theta = poolData(x,n,3); % вплоть до полиномов третьей степени
lambda = 0.025;          % lambda управляет разреженностью
Xi = sparsifyDynamics(Theta,dx,lambda,n)
```

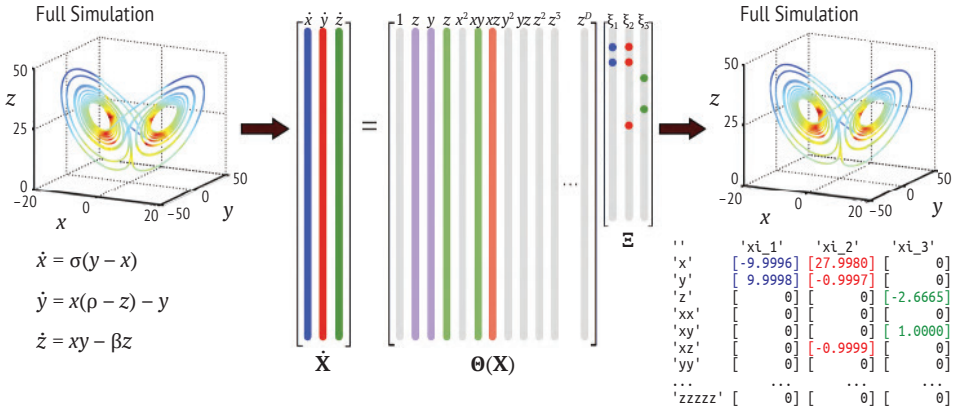

Рис. 7.4 ❖ Схема алгоритма разреженной идентификации нелинейной динамики (SINDy) [95]. Экономные модели выбираются из библиотеки нелинейных кандидатов с помощью разреженной регрессии. Эту библиотеку $\boldsymbol{\Theta}(\mathbf{X})$ можно построить на основе одних лишь данных измерений. *Взято из работы Brunton et al. [95]*

В этой программе также используется функция **poolData** для генерирования библиотеки $\boldsymbol{\Theta}$. В данном случае применяются полиномы не выше третьей степени. Код функции имеется в сети.

На выходе алгоритма SINDy получается разреженная матрица коэффициентов $\boldsymbol{\Xi}$:

```
''       'xdot'       'ydot'      'zdot'
 '1'   [       0]   [      0]   [      0]
 'x'   [-10.0000]   [28.0000]   [      0]
 'y'   [ 10.0000]   [-1.0000]   [      0]
 'z'   [       0]   [      0]   [-2.6667]
 'xx'  [       0]   [      0]   [      0]
 'xy'  [       0]   [      0]   [ 1.0000]
 'xz'  [       0]   [-1.0000]   [      0]
 'yy'  [       0]   [      0]   [      0]
 'yz'  [       0]   [      0]   [      0]
 'zz'  [       0]   [      0]   [      0]
 'xxx' [       0]   [      0]   [      0]
 'xxy' [       0]   [      0]   [      0]
 'xxz' [       0]   [      0]   [      0]
 'xyy' [       0]   [      0]   [      0]
 'xyz' [       0]   [      0]   [      0]
 'xzz' [       0]   [      0]   [      0]
 'yyy' [       0]   [      0]   [      0]
 'yyz' [       0]   [      0]   [      0]
 'yzz' [       0]   [      0]   [      0]
 'zzz' [       0]   [      0]   [      0]
```

Результатом SINDy-регрессии является экономная модель, включающая только самые важные члены, необходимые для объяснения наблюдаемого поведения. Процедура разреженной регрессии, применяемая для идентификации экономной нелинейной модели, выпуклая. Альтернативный подход – регрессия по всем возможным разреженным нелинейным структурам – вычислительно нереализуемый полный перебор комбинаторно большого множества форм модели-кандидата. SINDy избегает такого комбинаторного поиска с помощью современных методов выпуклой оптимизации и машинного обучения. Интересно отметить, что для динамических систем с дискретным временем, если $\boldsymbol{\Theta}(\mathbf{X})$ включает только линейные члены и если мы уберем член, поощряющий разреженность, положив $\lambda = 0$, то этот алгоритм сведется к разложению по динамическим модам [472, 456, 535, 317]. Если используется регрессия методом наименьших квадратов, как в DMD, то даже незначительная ошибка измерения или численное округление приведет к тому, что все члены в библиотеке будут давать ненулевой вклад в динамику, что не имеет физического смысла. Основное достоинство архитектуры SINDy – способность идентифицировать экономные модели, содержащие только необходимые нелинейные члены, которые, следовательно, допускают интерпретацию и не подвержены переобучению.

Приложения, расширения и исторический контекст

Недавно алгоритм SINDy был применен для идентификации динамических систем высокой размерности, таких как течения жидкости, для чего были использованы POD-коэффициенты [95, 341, 342]. На рис. 7.5 показано применение SINDy к построению целиком на основе данных модели течения за цилиндром, обобщающей модель среднего поля из работы Noack et al. [402]. SINDy использовался также для идентификации моделей в нелинейной оптике [497] и физике плазмы [141].

Поскольку SINDy сформулирован в терминах линейной регрессии для нелинейной библиотеки, он прекрасно расширяется. Недавно схема SINDy была обобщена в работе Loiseau and Brunton [341] с целью включения в уравнения известных физических ограничений и симметрий, для чего была реализована условная оптимизация методом последовательных наименьших квадратов с порогом. В частности, были наложены ограничения сохранения энергии на квадратичные нелинейности в уравнениях Навье–Стокса для идентификации жидкостных систем [341], для которых известно, что эти ограничения повышают устойчивость [355, 32, 118]. В этой работе также показано, что библиотеки полиномов особенно полезны для построения моделей течения жидкости в терминах POD-коэффициентов, поскольку порождают хорошо интерпретируемые модели, связанные с классическим проекционным методом Галеркина [95, 341]. В работе Loiseau et al. [342] также продемонстрировано, что SINDy может идентифицировать динамические модели систем высокой размерности, в частности модели течения газа, по показаниям небольшого количества датчиков, например по результатам измерения подъемной силы и лобового сопротивления на цилиндре (рис. 7.5). Для систем с приводами имеется обобщение SINDy, позволяющее включать вхо-

ды и управление [100], такие модели показали высокую эффективность для управления с упреждением [277]. Алгоритм SINDy можно также обобщить для идентификации динамики с привлечением рациональных нелинейностей [361], интегральных членов [469] и на основе сильно искаженных и неполных данных [522]. Недавно SINDy был расширен путем включения информационного критерия для объективного выбора модели [362] и для идентификации моделей со скрытыми переменными с использованием запаздывающих координат [91]. Наконец, схема SINDy была обобщена на частные производные, что позволяет идентифицировать модели, описываемые дифференциальными уравнениями в частных производных [460, 468]. Некоторые из последних усовершенствований будут подробно рассмотрены ниже.

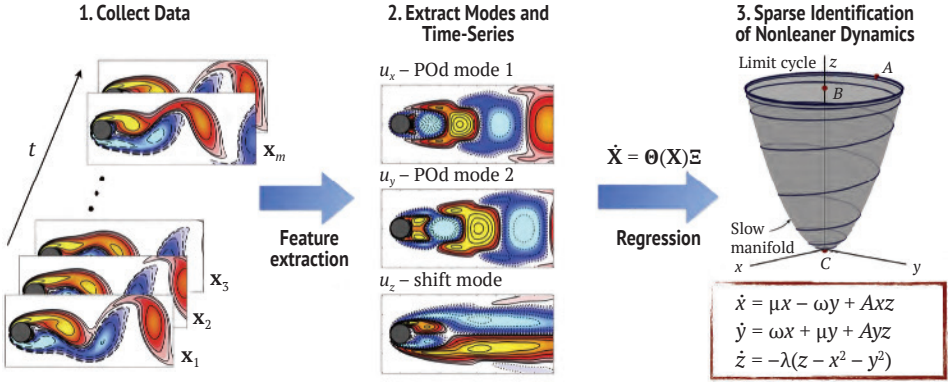

Рис. 7.5 ❖ Схема идентификации нелинейной модели по данным высокой размерности с применением алгоритма разреженной идентификации нелинейной динамики (SINDy) [95]. Это модульная процедура, поэтому для выделения признаков регрессии можно применять различные методы. В данном примере течения жидкости за цилиндром SINDy идентифицировал модель, описанную в работе Noack et al. [402]. *Взято из работы Brunton et al. [95]*

Вообще, использование методов, поощряющих разреженность, при изучении динамических систем – совсем новое направление [548, 467, 414, 353, 98, 433, 31, 29, 89, 364, 366]. Из других методов идентификации динамических систем упомянем нахождение уравнений по временным рядам [140], моделирование без уравнений [288], эмпирическое динамическое моделирование [503, 563], моделирование эмерджентного поведения [452], нелинейную авторегрессионную модель с экзогенными входами (NARMAX) [208, 571, 59, 484] и автоматизированный вывод динамики [478, 142, 143]. В самом общем смысле эти техники можно классифицировать как идентификацию систем, когда заимствованные из статистики и машинного обучения методы используются для идентификации динамических систем по данным. Почти во всех методах идентификации систем встречается та или иная форма регрессии для отображения данных на динамику, а различаются они прежде всего тем, в какой мере эта регрессия ограничена. Например, разложение по динамическим модам порождает линейные модели наилучшей аппроксимации. Не-

давно появившиеся методы нелинейной регрессии порождают нелинейные динамические модели, сохраняющие физические ограничения, например учитывающие закон сохранения энергии. Важный прорыв в автоматизированной идентификации нелинейных систем был совершен в работах Bongard and Lipson [68] и Schmidt and Lipson [477], где для идентификации структуры нелинейной динамики применялось генетическое программирование. Эти методы очень гибкие и налагают мало ограничений на форму динамики. Кроме того, SINDy тесно связан с методом NARMAX [59], который идентифицирует структуру моделей по временным рядам с помощью процедуры ортогональных наименьших квадратов.

Нахождение дифференциальных уравнений в частных производных

Важным расширением схемы моделирования SINDy стало обобщение библиотеки путем включения частных производных, что позволило идентифицировать уравнения в частных производных [460, 468]. Было продемонстрировано, что получившийся алгоритм функциональной идентификации нелинейной динамики с уравнениями в частных производных (partial differential equation functional identification of nonlinear dynamics – PDE-FIND), получая одни лишь зашумленные данные, успешно находит канонические УрЧП классической физики, в том числе уравнения Навье–Стокса, Курамото–Сивашинского, Шрёдингера, реакционной диффузии, Бюргерса, Кортевега–де Фриза и уравнения диффузии для броуновского движения [460].

PDE-FIND похож на SINDy тем, что основан на разреженной регрессии по библиотеке, построенной на основе данных измерений. Разреженная регрессия и метод идентификации показаны на рис. 7.6. Ниже PDE-FIND излагается для УрЧП с одной переменной, хотя теория легко обобщается на многомерные УрЧП. Пространственно-временные данные представлены единственным вектором-столбцом $\mathbf{\Upsilon} \in \mathbb{C}^{mn}$, который содержит данные, собранные в m моментов времени в n точках пространства. Дополнительные данные, например известный потенциал в уравнении Шрёдингера или модуль комплексных величин, представлены вектором-столбцом $\mathbf{Q} \in \mathbb{C}^{mn}$. Далее строится библиотека $\mathbf{\Theta}(\mathbf{\Upsilon}, \mathbf{Q}) \in \mathbb{C}^{mn \times D}$, состоящая из D линейных и нелинейных членов-кандидатов и частных производных. Производные вычисляются либо методом конечных разностей, если данные чистые, либо с помощью полиномиальной интерполяции, если добавлен шум. Затем линейные и нелинейные члены-кандидаты и частные производные объединяются в матрицу $\mathbf{\Theta}(\mathbf{\Upsilon}, \mathbf{Q})$, принимающую вид:

$$\mathbf{\Theta}(\mathbf{\Upsilon}, \mathbf{Q}) = [\mathbf{1} \;\; \mathbf{\Upsilon} \;\; \mathbf{\Upsilon}^2 \;\; \cdots \;\; \mathbf{Q} \;\; \cdots \;\; \mathbf{\Upsilon}_x \;\; \mathbf{\Upsilon}\mathbf{\Upsilon}_x \;\; \cdots]. \tag{7.41}$$

Каждый столбец $\mathbf{\Theta}$ содержит значения одной функции-кандидата во всех точках пространственно-временной сетки размера $m \times n$, где собраны данные. Производная по времени $\mathbf{\Upsilon}_t$ также вычисляется и сохраняется в векторе-столбце. На рис. 7.6 показан процесс сбора и обработки данных. Например, в каком-то столбце $\mathbf{\Theta}(\mathbf{\Upsilon}, \mathbf{Q})$ могут храниться значения qu_x^2.

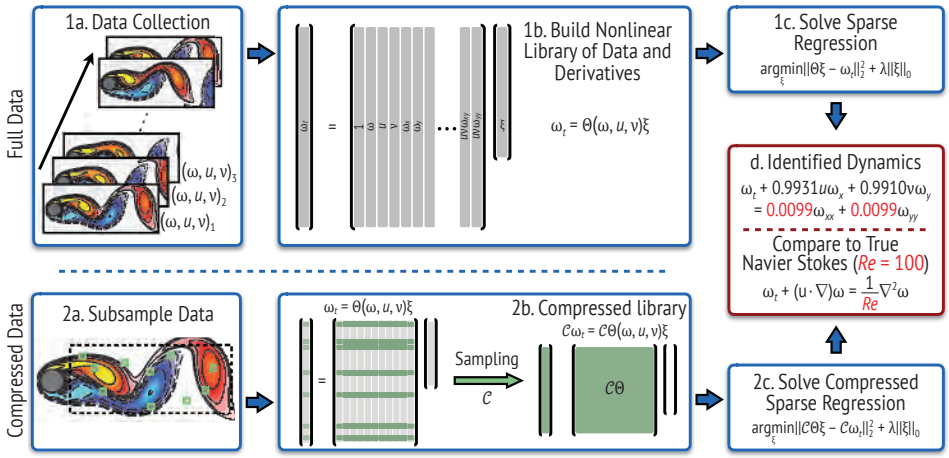

Рис. 7.6 ❖ Шаги алгоритма функциональной идентификации нелинейной динамики с уравнениями в частных производных (PDE-FIND), примененного для вывода уравнений Навье–Стокса из данных (взято из работы Rudy et al. [460]): 1a) собираются данные – снимки решения УрЧП; 1b) численно берутся производные, и данные собираются в большую матрицу $\boldsymbol{\Theta}$, включающую члены-кандидаты для УрЧП; 1c) используется разреженная регрессия для идентификации ненулевых членов в УрЧП; 2a) для больших наборов данных можно использовать разреженную выборку, чтобы уменьшить размер задачи; 2b) подвыборка из набора данных эквивалентна выбору подмножества строк из линейной системы (7.42); 2c) строится такая же задача разреженной регрессии, но с меньшим числом строк; 2d) ненулевые элементы ξ синтезируются в УрЧП

Видоизменение УрЧП можно выразить в этой библиотеке следующим образом:

$$\boldsymbol{\Upsilon}_t = \boldsymbol{\Theta}(\boldsymbol{\Upsilon}, \mathbf{Q})\boldsymbol{\xi}. \tag{7.42}$$

Каждый элемент вектора $\boldsymbol{\xi}$ – это коэффициент при некотором члене УрЧП, и для канонических УрЧП вектор $\boldsymbol{\xi}$ *разреженный*, т. е. в нем лишь несколько ненулевых элементов.

Если пространство столбцов в библиотеке $\boldsymbol{\Theta}$ настолько богато, что искомая динамика принадлежит их линейной оболочке, то УрЧП должно быть представлено в виде (7.42) с некоторым разреженным вектором коэффициентов $\boldsymbol{\xi}$. Чтобы идентифицировать немногие активные члены динамики, применяется поощряющая разреженность регрессия, как, например, в методе SINDy. Важно, что задача регрессии (7.42) может быть плохо обусловлена.

Алгоритм 1. STRidge($\boldsymbol{\Theta}$, $\boldsymbol{\Upsilon}_t$, λ, *tol*, iters)

$\hat{\boldsymbol{\xi}} = argmin_{\xi}||\boldsymbol{\Theta}\boldsymbol{\xi} - \boldsymbol{\Upsilon}_t||_2^2 + \lambda||\boldsymbol{\xi}||_2^2$ % гребневая регрессия
bigcoeffs $= \{j : |\hat{\xi}_j| \geq tol\}$ % выбрать большие коэффициенты
$\hat{\boldsymbol{\xi}}$ [~ bigcoeffs] $= 0$ % применить жесткий порог
$\hat{\boldsymbol{\xi}}$ [bigcoeffs] = STRidge($\boldsymbol{\Theta}$[:, bigcoeffs], $\boldsymbol{\Upsilon}_t$, *tol*, iters $-$ 1)
% рекурсивный вызов с меньшим числом коэффициентов
return $\hat{\boldsymbol{\xi}}$

Ошибка вычисления производных возрастает из-за численных погрешностей при обращении $\boldsymbol{\Theta}$. Таким образом, регрессия методом наименьших квадратов радикально изменяет качественную природу найденной динамики.

В общем случае мы ищем самый разреженный вектор $\boldsymbol{\xi}$, который удовлетворяет (7.42) с малой невязкой. Вместо практически нереализуемого комбинаторного поиска среди всех возможных разреженных векторов обычно применяется общий прием, позволяющий свести задачу к регрессии наименьших квадратов с регуляризацией по норме ℓ_1 [518], однако это плохо работает, когда данные сильно коррелированы. Поэтому мы используем гребневую регрессию с жестким порогом, которую назвали «гребневой регрессией с последовательным пороговым отсечением» (sequential threshold ridge regression – STRidge), в алгоритме 1, заимствованном из работы Rudy et al. [460]. При заданном допустимом отклонении и пороге λ она дает разреженную аппроксимацию $\boldsymbol{\xi}$. Отклонение в алгоритме 1 последовательно улучшается для нахождения наилучшего предиктора, при этом используется критерий

$$\hat{\boldsymbol{\xi}} = \text{argmin}_{\boldsymbol{\xi}} ||\boldsymbol{\Theta}(\boldsymbol{\Upsilon}, \mathbf{Q})\boldsymbol{\xi} - \boldsymbol{\Upsilon}_t||_2^2 + \varepsilon\kappa(\boldsymbol{\Theta}(\boldsymbol{\Upsilon}, \mathbf{Q}))||\boldsymbol{\xi}||_0, \tag{7.43}$$

где $\kappa(\boldsymbol{\Theta})$ – число обусловленности матрицы $\boldsymbol{\Theta}$, что обеспечивает более сильную регуляризацию для некорректных задач. Штраф за $||\boldsymbol{\xi}||_0$ предотвращает переобучение благодаря выбору из оптимальной позиции на Парето-фронте.

Как и в алгоритме SINDy, важно подать на вход достаточно полные обучающие данные, позволяющие проводить различие между несколькими моделями. Так, на рис. 7.7 показано применение алгоритма PDE-FIND для идентификации уравнения Кортевега–де Фриза (КдФ). Если анализируется только одна бегущая волна, то метод неправильно идентифицирует уравнение стандартной линейной адвекции, поскольку это простейшее уравнение, описывающее уединенную бегущую волну. Однако в результате анализа двух бегущих волн разной амплитуды правильно идентифицируется уравнение КдФ, поскольку оно описывает разные скорости волн, зависящие от амплитуды.

Алгоритм PDE-FIND можно также использовать для идентификации УрЧП по измерениям лагранжиана вдоль траектории отдельных частиц. Например, на рис. 7.8 показана идентификация уравнения диффузии, описывающего броуновское движение частицы на основе одного длинного временного ряда измерений положения частицы. В этом примере временной ряд разбивается на несколько коротких отрезков, и эволюция распределения положений в них используется для идентификации дифференциального уравнения.

Обобщение SINDy на рациональные нелинейности

Во многих динамических системах, например в метаболических и регуляторных сетях в биологии, динамика содержит нелинейности, описываемые рациональными функциями. Часто такие рациональные нелинейности возникают из-за разделения временных масштабов. Хотя оригинальный алгоритм SINDy допускает большую гибкость с точки зрения библиотеки нелинейно-

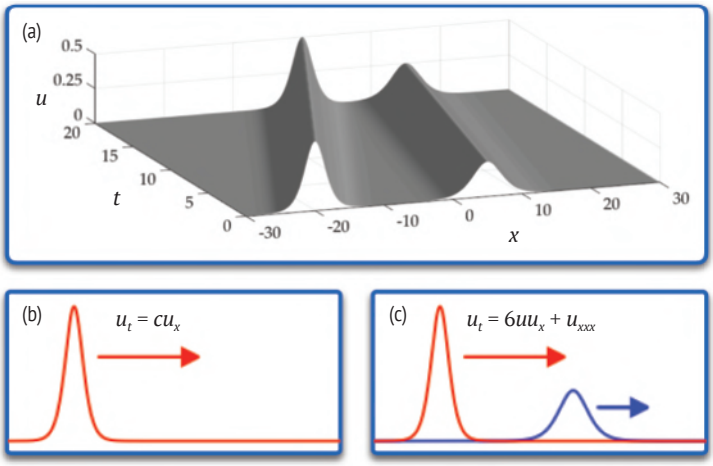

Рис. 7.7 ❖ Вывод нелинейности посредством наблюдения решений с несколькими амплитудами (*взято из работы Rudy et al. [460]*): (a) пример двухсолитонного решения уравнения КдФ; (b) применение нашего метода к решению с одним солитоном определяет, что оно описывается уравнением стандартной адвекции; (c) сравнение двух совершенно различных решений выявляет нелинейность

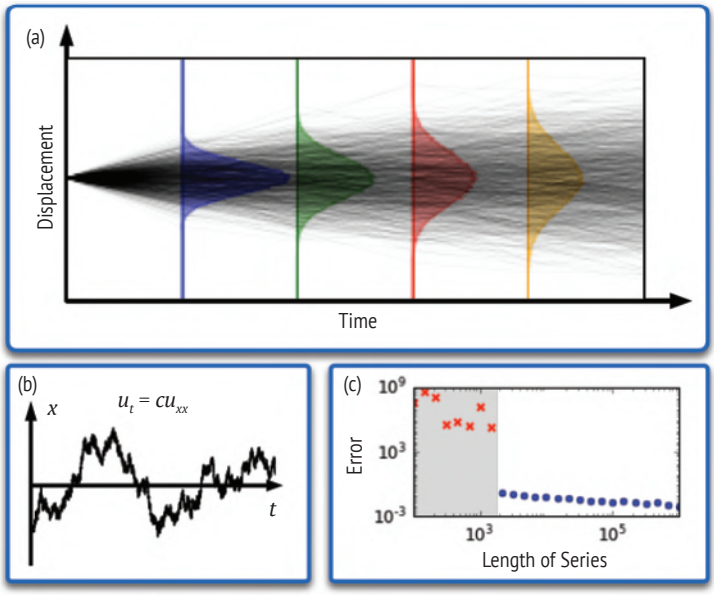

Рис. 7.8 ❖ Вывод уравнения диффузии из данных о броуновском движении одной частицы (*взято из работы Rudy et al. [460]*): (a) временной ряд разбивается на несколько коротких участков случайного блуждания, которые используются для построения гистограмм смещения; (**b**) траектория броуновского движения, описываемая уравнением диффузии; (**c**) зависимость параметрической ошибки ($||\boldsymbol{\xi}^* - \hat{\boldsymbol{\xi}}||_1$) от длины известного временного ряда. Синие точки соответствуют правильной идентификации структуры модели диффузии, $u_t = cu_{xx}$

стей, идентифицировать с его помощью рациональные функции нелегко, потому что, вообще говоря, рациональная функция непредставима линейной комбинацией нескольких базисных функций. Поэтому необходимо переформулировать динамику в виде *неявного* обыкновенного дифференциального уравнения и соответственно модифицировать процедуру оптимизации, как описано в работе Mangan et al. [361].

Мы рассматриваем динамические системы с рациональными нелинейностями:

$$\dot{x}_k = \frac{f_N(\mathbf{x})}{f_D(\mathbf{x})}, \tag{7.44}$$

где x_k – k-я переменная, а $f_N(\mathbf{x})$ и $f_D(\mathbf{x})$ – полиномы в числителе и знаменателе, зависящие от переменной состояния $\mathbf{x}$. Для каждого индекса k умножим обе части равенства на знаменатель f_D:

$$f_N(\mathbf{x}) - f_D(\mathbf{x})\dot{x}_k = 0. \tag{7.45}$$

Неявная форма (7.45) наводит на мысль о том, чтобы обобщить библиотеку $\boldsymbol{\Theta}$ (7.37), добавив состояние $\mathbf{x}$ и производную $\dot{x}_k$:

$$\boldsymbol{\Theta}(\mathbf{X}, \dot{x}_k(\mathbf{t})) = [\boldsymbol{\Theta}_N(\mathbf{X}) \quad \mathrm{diag}(\dot{x}_k(\mathbf{t}))\boldsymbol{\Theta}_D(\mathbf{X})]. \tag{7.46}$$

Первый член, $\boldsymbol{\Theta}_N(\mathbf{X})$, является библиотекой одночленов от $\mathbf{x}$ в числителе, как в (7.37). Второй член, $\mathrm{diag}(\dot{x}_k(\mathbf{t}))\boldsymbol{\Theta}_D(\mathbf{X})$, получается поэлементным умножением каждого столбца библиотеки полиномов в знаменателе $\boldsymbol{\Theta}_D(\mathbf{X})$ на вектор $\dot{x}_k(\mathbf{t})$. Для скалярной переменной x_k это дало бы:

$$\mathrm{diag}(\dot{x}_k(\mathbf{t}))\boldsymbol{\Theta}(\mathbf{X}) = \left[\dot{x}_k(\mathbf{t}) \quad (\dot{x}_k x_k)(\mathbf{t}) \quad (\dot{x}_k x_k^2)(\mathbf{t}) \quad \ldots\right]. \tag{7.47}$$

В большинстве случаев степени полиномов в числителе и знаменателе будут одинаковы, т. е. $\boldsymbol{\Theta}_N(\mathbf{X}) = \boldsymbol{\Theta}_D(\mathbf{X})$. Таким образом, дополненная библиотека (7.46) всего в два раза больше исходной библиотеки (7.37).

Теперь мы можем записать динамику в (7.45) в терминах дополненной библиотеки (7.46):

$$\boldsymbol{\Theta}(\mathbf{X}, \dot{x}_k(\mathbf{t}))\boldsymbol{\xi}_k = \mathbf{0}. \tag{7.48}$$

В разреженном векторе коэффициентов $\boldsymbol{\xi}_k$ активным членам динамики будут соответствовать ненулевые элементы. Однако использовать ту же процедуру разреженной регрессии, что в SINDy, невозможно, потому что самый разреженный вектор $\boldsymbol{\xi}_k$, удовлетворяющий (7.48), – это тривиальный нулевой вектор.

Вместо этого самый разреженный ненулевой вектор $\boldsymbol{\xi}_k$, удовлетворяющий (7.48), ищется как самый разреженный вектор в ядре $\boldsymbol{\Theta}$. В общем случае эта задача невыпуклая, хотя недавно в работе Qu et al. [440] были разработаны алгоритмы нахождения самого разреженного вектора в подпространстве, основанные на методе переменных направлений. В отличие от оригинального алгоритма SINDy, эта процедура весьма чувствительна к шуму, поскольку

ядро численно аппроксимируется как линейная оболочка сингулярных векторов, соответствующих малым сингулярным значениям. Если внести шум в матрицу данных $\mathbf{X}$, а значит, и в $\boldsymbol{\Theta}$, то уровень шума в сингулярном разложении повысится, что увеличит ранг численно найденного ядра.

Общая постановка для неявных ОДУ

Описанную выше процедуру можно обобщить на более широкий класс неявных обыкновенных дифференциальных уравнений в дополнение к содержащим рациональные нелинейности. Библиотека $\boldsymbol{\Theta}(\mathbf{X}, \dot{x}_k(\mathbf{t}))$ содержит подмножество столбцов библиотеки $\boldsymbol{\Theta}([\mathbf{X}\ \ \dot{\mathbf{X}}])$, которое получается путем построения нелинейных функций от состояния $\mathbf{x}$ и производной $\dot{\mathbf{x}}$. Идентификация самого разреженного вектора в ядре $\boldsymbol{\Theta}([\mathbf{X}\ \ \dot{\mathbf{X}}])$ повышает гибкость идентификации нелинейных уравнений со смешанными членами, содержащими любые комбинации различных степеней производных и состояний. Например, систему, описываемую уравнением

$$\dot{x}^2x^2 - \dot{x}x - x^2 = 0, \tag{7.49}$$

можно представить разреженным вектором в ядре $\boldsymbol{\Theta}([\mathbf{X}\ \ \dot{\mathbf{X}}])$. Эту постановку можно обобщить, включив в библиотеку $\boldsymbol{\Theta}$ производные более высокого порядка, например с целью идентификации неявных дифференциальных уравнений второго порядка:

$$\boldsymbol{\Theta}([\mathbf{X}\ \ \dot{\mathbf{X}}\ \ \ddot{\mathbf{X}}]). \tag{7.50}$$

Такая общность позволяет идентифицировать многие интересные системы, в т. ч. и системы с рациональными нелинейностями.

Применение информационного критерия для выбора модели

При выполнении разреженной регрессии в алгоритме SINDy параметр λ, управляющий разреженностью, является свободной переменной. На практике различные значения λ приводят к моделям с разным уровнем разреженности – от тривиальной модели $\dot{x} = 0$ при очень большом λ до простого решения методом наименьших квадратов при $\lambda = 0$. Таким образом, за счет изменения λ можно замести Парето-фронт, стараясь отыскать баланс между ошибкой и сложностью, как показано на рис. 7.9. Чтобы выбрать самую экономную модель разумной сложности с малой ошибкой, мы можем применить информационный критерий, как описано в работе Mangan et al. [362]. В частности, если взять информационный критерий Акаике (AIC) [6, 7], который штрафует за количество членов в модели, то самая экономная модель будет минимизировать AIC. Эта процедура применялась к нескольким задачам разреженной идентификации, и во всех случаях была найдена правильная модель [362].

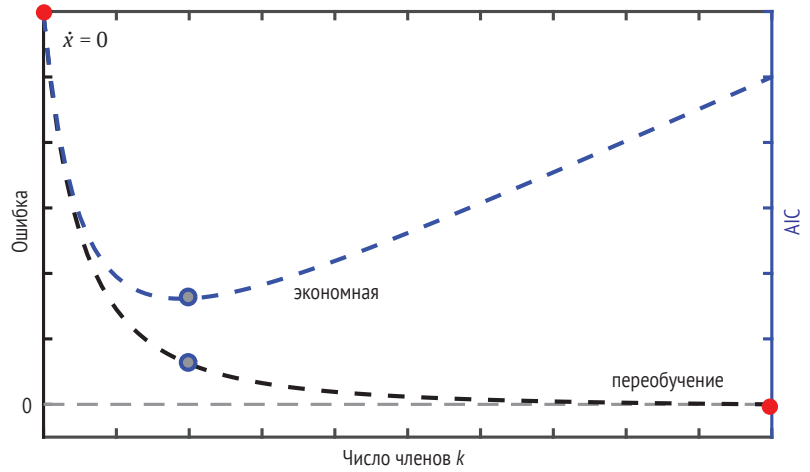

Рис. 7.9 ❖ Выбор модели с применением SINDy и информационного критерия, как в работе Mangan et al. [362]. Выбирается наиболее экономная модель на Парето-фронте, которая минимизирует оценку AIC (синяя точка), предотвращая переобучение

7.4. Теория оператора Купмана

Теория Купмана в последнее время стала альтернативным взглядом на динамические системы с точки зрения эволюции измерений $g(\mathbf{x})$. В 1931 году Бернард О. Купман продемонстрировал, что нелинейную динамическую систему можно представить в терминах бесконечномерного оператора в гильбертовом пространстве функций измерения состояния системы. Этот *оператор Купмана* линеен, а его спектральное разложение полностью характеризует поведение нелинейной системы по аналогии с (7.7). Однако он еще и бесконечномерный, поскольку для описания пространства всех возможных функций измерения g, зависящих от состояния, необходимо бесконечное число степеней свободы. Поэтому возникают новые проблемы. Получение конечномерной матричной аппроксимации оператора Купмана – предмет напряженных усилий многих исследователей, т. к. сулит нахождение глобально линейного представления нелинейных динамических систем. Выражение нелинейной динамики в линейном виде привлекательно, потому что для линейных систем существует множество методов оптимального оценивания и управления (см. главу 8). Нахождение конечномерной аппроксимации оператора Купмана на практике вызывает трудности, поскольку требуется найти подпространство, натянутое на подмножество собственных функций оператора Купмана.

Математическая теория оператора Купмана

Оператор Купмана экстраполирует функции измерения на будущие состояния динамической системы. Мы рассматриваем вещественные функции из-

мерения $g : \mathbf{M} \to \mathbb{R}$, являющиеся элементами бесконечномерного гильбертова пространства. Функции g часто называют *наблюдаемыми*, хотя тут может возникнуть путаница с никак не связанным понятием *наблюдаемости* из теории управления. Обычно в качестве гильбертова пространства берется множество квадратично интегрируемых по Лебегу функций на $\mathbf{M}$, но возможны и другие пространства с мерой.

Оператор Купмана $\mathcal{K}_t$ – это бесконечномерный линейный оператор, воздействующий на функции измерения g:

$$\mathcal{K}_t g = g \circ \mathbf{F}_t, \tag{7.51}$$

где $\circ$ – оператор композиции. Для системы с дискретным временем с временным шагом Δt это равенство принимает вид

$$\mathcal{K}_{\Delta t} g(\mathbf{x}_k) = g(\mathbf{F}_{\Delta t}(\mathbf{x}_k)) = g(\mathbf{x}_{k+1}). \tag{7.52}$$

Иными словами, оператор Купмана определяет бесконечномерную линейную динамическую систему, которая экстраполирует наблюдение состояния $g_k = g(\mathbf{x}_k)$ на следующий временной шаг:

$$g(\mathbf{x}_{k+1}) = \mathcal{K}_{\Delta t} g(\mathbf{x}_k). \tag{7.53}$$

Заметим, что это справедливо для *любой* наблюдаемой функции g и любого состояния $\mathbf{x}_k$.

Оператор Купмана линеен, это свойство наследуется от линейности операции сложения в пространствах функций:

$$\mathcal{K}_t(\alpha_1 g_1(\mathbf{x}) + \alpha_2 g_2(\mathbf{x}) = \alpha_1 g_1(\mathbf{F}_t(\mathbf{x})) + \alpha_2 g_2(\mathbf{F}_t(\mathbf{x})) \tag{7.54a}$$

$$= \alpha_1 \mathcal{K}_t g_1(\mathbf{x}) + \alpha_2 \mathcal{K}_t g_2(\mathbf{x}). \tag{7.54b}$$

Для достаточно гладких динамических систем можно определить также аналог динамической системы Купмана (7.53) с непрерывным временем:

$$\frac{d}{dt} g = \mathcal{K} g. \tag{7.55}$$

Оператор $\mathcal{K}$ – инфинитезимальная образующая однопараметрического семейства преобразований $\mathcal{K}_t$ [1]. Оно определяется действием оператора на наблюдаемую функцию g:

$$\mathcal{K} g = \lim_{t \to 0} \frac{\mathcal{K}_{tg} - g}{t} = \lim_{t \to 0} \frac{g \circ \mathbf{F}_t - g}{t}. \tag{7.56}$$

Линейные динамические системы в (7.55) и (7.53) аналогичны динамическим системам (7.3) и (7.4) соответственно. Важно отметить, что исходное состояние $\mathbf{x}$ может быть наблюдаемой функцией, а инфинитезимальный оператор $\mathcal{K}_t$ экстраполирует эту функцию. Однако простое представление $g = \mathbf{x}$ в выбранном базисе гильбертова пространства может оказаться сколь угодно сложным после итераций динамики. Иначе говоря, найти представление $\mathcal{K}_\mathbf{x}$ может быть совсем не тривиально.

Собственные функции оператора Купмана и внутренние координаты

Оператор Купмана линеен, что подкупает, но он бесконечномерный, что ставит проблемы для представления и вычислений. Вместо улавливания эволюции всех функций измерения в гильбертовом пространстве анализ Купмана в прикладных науках пытается идентифицировать ключевые функции измерения, которые линейно изменяются по мере развития динамики. Собственные функции оператора Купмана дают как раз такой набор специальных измерений, линейно зависящих от времени. На самом деле основой интереса к теории Купмана является ее способность упростить динамику посредством спектрального разложения оператора.

Собственная функция $\varphi(\mathbf{x})$ дискретного оператора Купмана, соответствующая собственному значению λ, удовлетворяет соотношению

$$\varphi(\mathbf{x}_{k+1}) = \mathcal{K}_{\Delta t}\,\varphi(\mathbf{x}_k) = \lambda\varphi(\mathbf{x}_k). \tag{7.57}$$

В случае непрерывного времени собственная функция $\varphi(\mathbf{x})$ оператора Купмана удовлетворяет соотношению

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi(\mathbf{x}) = \mathcal{K}\varphi(\mathbf{x}) = \lambda\varphi(\mathbf{x}). \tag{7.58}$$

Получение собственных функций Купмана на основе данных или аналитических выражений – главная проблема современных динамических систем. Нахождение этих собственных функций открывает путь к глобально линейному представлению существенно нелинейных систем.

Применение правила дифференцирования сложной функции к собственной функции Купмана $\varphi(\mathbf{x})$ дает

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi(\mathbf{x}) = \nabla\varphi(\mathbf{x})\cdot\dot{\mathbf{x}} = \nabla\varphi(\mathbf{x})\cdot\mathbf{f}(\mathbf{x}). \tag{7.59}$$

В сочетании с (7.58) это приводит к дифференциальному уравнению в частных производных для собственной функции $\varphi(\mathbf{x})$:

$$\nabla\varphi(\mathbf{x})\cdot\mathbf{f}(\mathbf{x}) = \lambda\varphi(\mathbf{x}). \tag{7.60}$$

С помощью этого нелинейного УрЧП можно аппроксимировать собственные функции либо посредством разложения в ряд Лорана, либо на основе данных с применением регрессии. Ниже мы рассмотрим оба подхода. В этой постановке предполагается, что динамика – непрерывная и дифференцируемая функция. Динамика с дискретным временем (7.4) более общая, хотя во многих примерах динамика с непрерывным временем имеет более простое представление. Например, простая система Лоренца имеет простое представление с непрерывным временем, но обновления с дискретным временем практически невозможно представить даже на не слишком длинных промежутках.

Главный вывод из формул (7.57) и (7.58) заключается в том, что нелинейная динамика становится полностью линейной в координатах, определяемых

собственной функцией $\varphi(\mathbf{x})$. Простой пример – любая сохраняющаяся величина динамической системы является собственной функцией оператора Купмана, соответствующей собственному значению $\lambda = 0$. Это составляет содержание обобщения Купмана знаменитой теоремы Нётер [406], из которого следует, что любая симметрия в описывающих уравнениях приводит к новой собственной функции Купмана с собственным значением $\lambda = 0$. Так, гамильтониан является собственной функцией Купмана для консервативной системы. Кроме того, постоянная функция $\varphi = 1$ всегда является тривиальной собственной функцией, соответствующей $\lambda = 0$, в любой динамической системе.

Решетки собственных значений. Интересно, что множество собственных функций Купмана можно использовать для порождения новых собственных функций. В случае дискретного времени произведение двух собственных функций $\varphi_1(\mathbf{x})$ и $\varphi_2(\mathbf{x})$ также является собственной функцией

$$\mathcal{K}_t(\varphi_1(\mathbf{x})\varphi_2(\mathbf{x})) = \varphi_1(\mathbf{F}_t(\mathbf{x}))\varphi_2(\mathbf{F}_t(\mathbf{x})) \tag{7.61a}$$

$$= \lambda_1\lambda_2\varphi_1(\mathbf{x})\varphi_2(\mathbf{x}), \tag{7.61b}$$

соответствующей собственному значению $\lambda_1\lambda_2$, равному произведению собственных значений функций $\varphi_1(\mathbf{x})$ и $\varphi_2(\mathbf{x})$.

В случае непрерывного времени это соотношение принимает вид

$$\mathcal{K}(\varphi_1\varphi_2) = \frac{d}{dt}(\varphi_1\varphi_2) \tag{7.62a}$$

$$= \dot{\varphi}_1\varphi_2 + \varphi_1\dot{\varphi}_2 \tag{7.62b}$$

$$= \lambda_1\varphi_1\varphi_2 + \lambda_2\varphi_1\varphi_2 \tag{7.62c}$$

$$= (\lambda_1 + \lambda_2)\varphi_1\varphi_2. \tag{7.62d}$$

Это означает, что множество собственных функций Купмана является коммутативным моноидом относительно операции поточечного умножения; моноид отличается от группы тем, что обратного элемента может не быть. Таким образом, в зависимости от динамической системы может существовать конечное множество *образующих* собственных функций, которые порождают все множество собственных функций. Соответствующие собственные значения образуют решетку относительно произведения $\lambda_1\lambda_2$ или суммы $\lambda_1 + \lambda_2$ в зависимости от того, рассматривается динамическая система с дискретным или непрерывным временем. Например, для линейной системы $\dot{x} = \lambda x$ функция $\varphi(x) = x$ является собственной функцией с собственным значением λ. Более того, $\varphi^\alpha = x^\alpha$ также является собственной функцией, соответствующей собственному значению $\alpha\lambda$ для любого α.

Между решетками с непрерывным и дискретным временем есть простая связь. Если у оператора с непрерывным временем есть собственное значение λ, то у соответствующего оператора с непрерывным временем есть собственное значение $e^{\lambda t}$. Таким образом, выражения для собственных значений (7.61b) и (7.62d) связаны следующим образом:

$$e^{\lambda_1 t}e^{\lambda_2 t}\varphi_1(\mathbf{x})\varphi_2(\mathbf{x}) = e^{(\lambda_1+\lambda_2)t}\varphi_1(\mathbf{x})\varphi_2(\mathbf{x}). \tag{7.63}$$

В качестве еще одного примера связи между собственными значениями в случае непрерывного и дискретного времени можно рассмотреть определение с непрерывным временем (7.56) применительно к собственной функции:

$$\lim_{t\to 0}\frac{K_t\varphi(\mathbf{x})-\varphi(\mathbf{x})}{t}=\lim_{t\to 0}\frac{e^{\lambda t}\varphi(\mathbf{x})-\varphi(\mathbf{x})}{t}=\lambda\varphi(\mathbf{x}). \tag{7.64}$$

Разложение по модам Купмана и конечные представления

До сих пор мы рассматривали скалярные измерения системы и обнаружили специальные *собственные* измерения, которые линейно изменяются с течением времени. Однако часто мы производим несколько измерений системы. В предельном случае может измеряться все состояние многомерной пространственной системы, как, например, в случае развивающегося потока жидкости. Затем эти измерения можно собрать в вектор $\mathbf{g}$:

$$\mathbf{g}(\mathbf{x})=\begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_p(\mathbf{x}) \end{bmatrix}. \tag{7.65}$$

Каждое отдельное измерение можно разложить по собственным функциям $\varphi_j(\mathbf{x})$, образующим базис гильбертова пространства:

$$g_i(\mathbf{x})=\sum_{j=1}^{\infty}v_{ij}\,\varphi_j(\mathbf{x}). \tag{7.66}$$

И вектор наблюдаемых функций $\mathbf{g}$ можно разложить аналогичным образом:

$$\mathbf{g}(\mathbf{x})=\begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_p(\mathbf{x}) \end{bmatrix}=\sum_{j=1}^{\infty}\varphi_j(\mathbf{x})\mathbf{v}_j, \tag{7.67}$$

где $\mathbf{v}_j$ – j-я *мода Купмана*, ассоциированная с собственной функцией φ_j.

Для консервативных динамических систем, например описываемых гамильтоновой динамикой, оператор Купмана является унитарным. Таким образом, для консервативных систем собственные функции Купмана ортонормированы, и моды Купмана $\mathbf{v}_j$ можно вычислить непосредственно по проекции:

$$\mathbf{v}_j=\begin{bmatrix} \langle\varphi_j, g_1\rangle \\ \langle\varphi_j, g_2\rangle \\ \vdots \\ \langle\varphi_j, g_p\rangle \end{bmatrix}, \tag{7.68}$$

где $\langle \cdot, \cdot \rangle$ – стандартное скалярное произведение функций в гильбертовом пространстве. В случае прямых пространственных измерений системы $\mathbf{g}(\mathbf{x}) = \mathbf{x}$ у этих мод имеется физическая интерпретация – они являются когерентными *пространственными* модами, которые ведут себя линейно с одинаковой временной динамикой (например, колебания с линейно возрастающей или убывающей амплитудой).

Имея разложение (7.67), можно представить динамику измерений $\mathbf{g}$ следующим образом:

$$\mathbf{g}(\mathbf{x}_k) = \mathcal{K}_{\Delta t}^k \mathbf{g}(\mathbf{x}_0) = \mathcal{K}_{\Delta t}^k \sum_{j=0}^{\infty} \varphi_j(\mathbf{x}_0)\mathbf{v}_j \tag{7.69a}$$

$$= \sum_{j=0}^{\infty} \mathcal{K}_{\Delta t}^k \varphi_j(\mathbf{x}_0)\mathbf{v}_j \tag{7.69b}$$

$$= \sum_{j=0}^{\infty} \lambda_j^k \varphi_j(\mathbf{x}_0)\mathbf{v}_j. \tag{7.69c}$$

Эта последовательность троек $\{(\lambda_j, \varphi_j, \mathbf{v}_j)\}_{j=0}^{\infty}$, называемая *разложением по модам Купмана*, была введена в 2005 году в работе Мезича [376]. Впоследствии разложение по модам Купмана было связано с управляемой данными регрессией с помощью разложения по динамическим модам [456], которое обсуждалось в разделе 7.2.

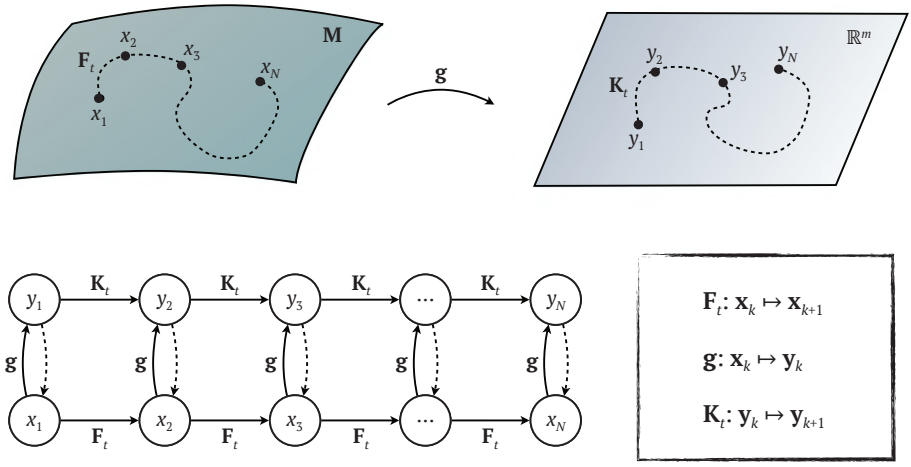

Рис. 7.10 ❖ Схематическая иллюстрация оператора Купмана для нелинейных динамических систем. Пунктирные линии $\mathbf{y}_k \to \mathbf{x}_k$ показывают, что мы хотели иметь возможность восстановить первоначальное состояние

Инвариантные собственные пространства и конечномерные модели

Вместо того чтобы улавливать эволюцию всех функций измерения в гильбертовом пространстве, в прикладном анализе Купмана аппроксимируется

эволюция на инвариантном подпространстве, натянутом на конечное множество функций измерения.

Линейная оболочка множества функций $\{g_1, g_2, \cdots, g_p\}$ называется *инвариантным относительно оператора Купмана подпространством*, если все функции g, принадлежащие этому подпространству,

$$g = \alpha_1 g_1 + \alpha_2 g_2 + \cdots + \alpha_p g_p, \tag{7.70}$$

остаются в нем после применения к ним оператора Купмана $\mathcal{K}$:

$$\mathcal{K}g = \beta_1 g_1 + \beta_2 g_2 + \cdots + \beta_p g_p. \tag{7.71}$$

Можно получить представление оператора Купмана в виде конечномерной матрицы, ограничив его на инвариантное подпространство, натянутое на конечное число функций $\{g_j\}_{j=0}^p$. Матричное представление $\mathbf{K}$ воздействует на векторное пространство $\mathbb{R}^p$ с координатами, определяемыми значениями $g_j(\mathbf{x})$. Это отображение индуцирует конечномерную линейную систему, как в (7.53) и (7.55).

Линейная оболочка любого конечного множества собственных функций оператора Купмана определяет инвариантное подпространство. Нахождение такого множества собственных функций является поэтому главной задачей, поскольку они дают оси координат, вдоль которых динамика развивается линейно. На практике более вероятно найти *приближенно* инвариантное подпространство, определяемое множеством функций $\{g_j\}_{j=0}^p$, каждая из которых аппроксимирована конечной суммой собственных функций: $g_j \approx \sum_{k=0}^p \alpha_k \varphi_k$.

Примеры погружений Купмана

Нелинейная система с одной неподвижной точкой и медленным многообразием

Рассмотрим пример системы с одной неподвижной точкой:

$$\dot{x}_1 = \mu x_1, \tag{7.72a}$$
$$\dot{x}_2 = \lambda(x_2 - x_1^2). \tag{7.72b}$$

При $\lambda < \mu < 0$ система обладает медленным притягивающим многообразием $x_2 = x_1^2$. Можно дополнить состояние $\mathbf{x}$ нелинейным измерением $g = x_1^2$ и тем самым определить трехмерное инвариантное подпространство Купмана. В этих координатах динамика становится линейной:

$$\frac{d}{dt}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \mu & 0 & 0 \\ 0 & \lambda & -\lambda \\ 0 & 0 & 2\mu \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \quad \text{где} \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}. \tag{7.73a}$$

Полное трехмерное наблюдаемое векторное пространство Купмана показано на рис. 7.11. Траектории, начинающиеся на инвариантном многообразии $y_3 = y_1^2$, которое закрашено голубым цветом, остаются на этом многообразии. Существует *медленное* подпространство, натянутое на собственные векторы, соответствующие медленным собственным значениям μ и 2μ; это подпространство изображено зеленым цветом. Наконец, существует асимптотически притягивающее многообразие исходной системы, $y_2 = y_1^2$, закрашенное розовым цветом. Голубая и розовая параболические поверхности пересекаются по параболе, наклоненной под углом 45° к плоскости y_2–y_3. Зеленая поверхность стремится к этой плоскости пересечения, когда отношение быстрой динамики к медленной стремится к бесконечности. В полном трехмерном наблюдаемом пространстве Купмана динамика порождает один устойчивый узел, так что траектории быстро притягиваются к зеленому подпространству, а затем медленно приближаются к неподвижной точке.

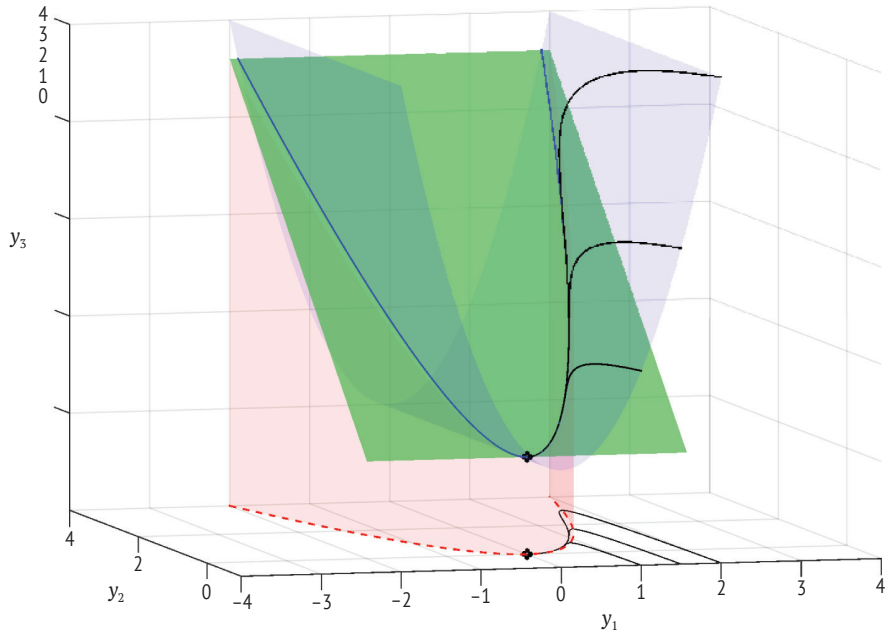

Рис. 7.11 ❖ Визуализация трехмерной линейной системы Купмана (7.73а) вместе с проекцией динамики на плоскость y_1–y_2. Притягивающее медленное многообразие показано розовым цветом, ограничение $y_3 = y_1^2$ – голубым, а медленное неустойчивое подпространство (7.73а) – зеленым. Черные траектории линейной системы Купмана в координатах **y** проецируются на траектории полной нелинейной системы в координатах **x** на плоскости y_1–y_2. Здесь $\mu = -0.05$ и $\lambda = 1$. *Взято из работы Brunton et al. [92]*

Внутренние координаты, определяемые собственными функциями оператора Купмана. Левые собственные векторы оператора Купмана дают собственные функции Купмана (т. е. собственные наблюдаемые величины). Собственные

функции Купмана для системы (7.73а), соответствующие собственным значениям μ и λ, имеют вид:

$$\varphi_\mu = x_1,\ \varphi_\lambda = x_2 - bx_1^2, \text{ где } b = \lambda/(\lambda - 2\mu). \tag{7.74}$$

Постоянная b в φ_λ отражает тот факт, что для конечного отношения λ/μ динамика только повторяет асимптотически притягивающее медленное многообразие, но на самом деле следует вдоль соседних с ним параболических траекторий. Это наглядно видно из поверхностей на рис. 7.11 для разных значений отношения λ/μ.

Таким образом, систему внутренних координат можно определить на основе наблюдаемых функций, определенных левыми собственными векторами оператора Купмана на инвариантном подпространстве. В явном виде

$$\varphi_\alpha(\mathbf{x}) = \boldsymbol{\xi}_\alpha \mathbf{y}(\mathbf{x}), \text{ где } \boldsymbol{\xi}_\alpha \mathbf{K} = \alpha \boldsymbol{\xi}_\alpha. \tag{7.75}$$

Эти собственные наблюдаемые функции определяют наблюдаемые подпространства, остающиеся инвариантными относительно оператора Купмана даже после преобразования координат. Поэтому их можно рассматривать как внутренние координаты [556] на инвариантном подпространстве.

Пример неразрешимого представления

Рассмотрим логистическое отображение

$$x_{k+1} = \beta x_k (1 - x_k). \tag{7.76}$$

Пусть наблюдаемое подпространство включает функции x и x^2:

$$\mathbf{y}_k = \begin{bmatrix} x \\ x^2 \end{bmatrix}_k \triangleq \begin{bmatrix} x_k \\ x_k^2 \end{bmatrix}. \tag{7.77}$$

В записи оператора Купмана с первой строкой матричного уравнения все просто:

$$\mathbf{y}_{k+1} = \begin{bmatrix} x \\ x^2 \end{bmatrix}_{k+1} = \begin{bmatrix} \beta & -\beta \\ ? & ? \end{bmatrix} \begin{bmatrix} x \\ x^2 \end{bmatrix}, \tag{7.78}$$

но со второй все не так очевидно. Чтобы найти это выражение, распишем x_{k+1}^2:

$$x_{k+1}^2 = (\beta x_k (1 - x_k))^2 = \beta^2 (x_k^2 - 2x_k^3 + x_k^4). \tag{7.79}$$

Таким образом, для экстраполяции x^2 необходимы члены третьей и четвертой степеней. А для экстраполяции этих членов нужны соответственно члены вплоть до шестой и восьмой степеней. И так далее до бесконечности:

$$\begin{bmatrix} x \\ x^2 \\ x^3 \\ x^4 \\ x^5 \\ \vdots \end{bmatrix}_{k+1} = \begin{bmatrix} x & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 & x^8 & x^9 & x^{10} & \\ \beta & -\beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & \beta^2 & -2\beta^2 & r^2 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & \beta^3 & -3\beta^3 & 3\beta^3 & \beta^3 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \beta^4 & -4\beta^4 & 6\beta^4 & -4\beta^4 & \beta^4 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \beta^5 & -5\beta^5 & 10\beta^5 & -10\beta^5 & 5\beta^5 & -\beta^5 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} x \\ x^2 \\ x^3 \\ x^4 \\ x^5 \\ \vdots \end{bmatrix}_k .$$

Интересно отметить связь между строками этого уравнения и строками треугольника Паскаля: n-я строка последнего умножается на β^n, а самая первая строка опущена:

$$[x^0]_{k+1} = [0][x^0]_k. \tag{7.80}$$

Это представление оператора Купмана в полиномиальном базисе вызывает проблемы. Мало того что оно не замкнуто, так еще и определитель любой конечной угловой подматрицы очень велик при $\beta > 1$. Это иллюстрирует трудность, связанную с наивным представлением бесконечномерного оператора Купмана для простой хаотической системы. Усечение системы или аппроксимация методом наименьших квадратов дополненного вектора наблюдаемых величин (т. е. разложение нелинейных измерений по динамическим модам, см. раздел 7.5) дает непригодные результаты – усеченная система согласуется с истинной динамикой только на небольшом количестве итераций, т. к. сложность представления быстро увеличивается:

$$\begin{matrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \\ x^5 \\ x^6 \\ x^7 \\ x^8 \\ \vdots \end{matrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \overset{\mathcal{K}}{\Rightarrow} \begin{bmatrix} 0 \\ \beta \\ -\beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \overset{\mathcal{K}}{\Rightarrow} \begin{bmatrix} 0 \\ \beta^2 \\ -\beta^2 - \beta^3 \\ 2\beta^3 \\ -\beta^3 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \overset{\mathcal{K}}{\Rightarrow} \begin{bmatrix} 0 \\ \beta^3 \\ -\beta^3 - \beta^4 - \beta^5 \\ 2\beta^4 + 2\beta^5 + 2\beta^6 \\ -\beta^4 - \beta^5 - 6\beta^6 - \beta^7 \\ 6\beta^6 + 4\beta^7 \\ -2\beta^6 - 6\beta^7 \\ 4\beta^7 \\ -\beta^7 \\ \vdots \end{bmatrix}. \tag{7.81}$$

Аналитическое разложение собственных функций в ряд

Имея динамическую систему (7.1), мы можем разложить УрЧП (7.60), применив стандартные методы, например рекурсивное нахождение членов разло-

жения в ряд Тейлора или Лорана. Ниже рассматривается несколько простых примеров.

Линейная динамика

Рассмотрим простую линейную динамическую систему

$$\frac{d}{dt}x = x. \tag{7.82}$$

Если разложение $\varphi(x)$ в ряд Тейлора имеет вид

$$\varphi(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \cdots,$$

то градиент и производная по направлению равны

$$\nabla\varphi = c_1 + 2c_2 x + 3c_3 x^2 + 4c_4 x^3 + \cdots,$$
$$\nabla\varphi \cdot f = c_1 x + 2c_2 x^2 + 3c_3 x^3 + 4c_4 x^4 + \cdots.$$

Находя коэффициенты из УрЧП (7.60) для собственной функции Купмана, мы видим, что должно быть $c_0 = 0$. Для любого целого положительного λ в (7.60) только один коэффициент может быть отличен от нуля. Точнее, для $\lambda = k \in \mathrm{Z}^+$ функция $\varphi(x) = cx^k$ является собственной для любой постоянной c. Например, если $\lambda = 1$, то $\varphi(x) = x$.

Квадратичная динамика

Рассмотрим нелинейную динамическую систему

$$J(t) = \int_0^t \mathbf{x}(\tau)^* \mathbf{Q}\mathbf{x}(\tau) + \mathbf{u}(\tau)^* \mathbf{R}\mathbf{u}(\tau) d\tau. \tag{7.83}$$

Не существует ряда Тейлора, удовлетворяющего (7.60), кроме тривиального решения $\varphi = 0$ для $\lambda = 0$. Вместо этого рассмотрим разложение в ряд Лорана:

$$\varphi(x) = \cdots + c_{-3}x^{-3} + c_{-2}x^{-2} + c_{-1}x^{-1} + c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \cdots.$$

Градиент и производная по направлению равны

$$\nabla\varphi = \cdots -3c_{-3}x^{-4} - 2c_{-2}x^{-3} - c_{-1}x^{-2} + c_1 + 2c_2 x + 3c_3 x^2 + 4c_4 x^3 + \cdots,$$
$$\nabla\varphi \cdot f = \cdots -3c_{-3}x^{-2} - 2c_{-2}x^{-1} - c_{-1} + c_1 x^2 + 2c_2 x^3 + 3c_3 x^4 + 4c_4 x^5 + \cdots.$$

При поиске ряда Лорана, удовлетворяющего уравнению (7.60), мы обнаруживаем, что все коэффициенты с положительными индексами равны нулю, т. е. $c_k = 0$ для всех $k \geq 1$. Но для коэффициентов с неположительными индексами $k \leq -1$ имеет место рекуррентное соотношение $\lambda c_{k+1} = kc_k$. Таким образом, разложение в ряд Лорана имеет вид

$$\varphi(x) = c_0\left(1 - \lambda x^{-1} + \frac{\lambda^2}{2}x^{-2} - \frac{\lambda^3}{3}x^{-3} + \cdots\right) = c_0 e^{-\lambda/x}.$$

Это справедливо для всех $\lambda \in \mathbb{C}$. Существуют и другие собственные функции Купмана, которые можно найти с помощью ряда Лорана.

Полиномиальная динамика

Для более общей нелинейной динамической системы

$$\frac{d}{dt} = ax^n \tag{7.84}$$

функция $\varphi\varphi(x) = e^{\frac{\lambda}{(1-n)a}x^{1-n}}$ является собственной для всех $\lambda \in \mathbb{C}$.

Как уже отмечалось, можно также строить новые собственные функции, возводя эти примитивные собственные функции в степень; соответствующие собственные значения образуют *решетку* на комплексной плоскости.

История и недавние достижения

Оригинальный анализ Купмана, представленный в 1931 году, был призван описать эволюцию измерений гамильтоновых систем [300]. В 1932 году эта теория была обобщена Купманом и фон Нейманом на системы с непрерывным спектром [301]. В случае гамильтоновых потоков оператор Купмана $\mathcal{K}_t$ унитарный и порождает однопараметрическое семейство унитарных преобразований гильбертова пространства. Читателю уже должны быть знакомы унитарные операторы, поскольку и дискретное преобразование Фурье (ДПФ), и сингулярное разложение (SVD) являются унитарными преобразованиями координат. Из унитарности следует, что скалярное произведение любых двух наблюдаемых функций не изменяется под действием оператора Купмана, что интуитивно связано со свойством сохранения фазового объема гамильтоновых систем. В оригинальной статье [300] Купман вывел связи между спектром собственных значений оператора Купмана и сохраняемыми величинами, интегрируемостью и эргодичностью. Интересно, что статья Купмана 1931 года легла в основу знаменитых доказательств эргодической теоремы, данных Биркгофом и фон Нейманом [62, 399, 61, 389].

Анализ Купмана недавно снова оказался в центре внимания в связи с пионерской работой Мезича с сотрудниками [379, 376, 102, 104, 103, 377, 322]. Оператор Купмана известен также как оператор композиции, формально это pullback-оператор на пространстве скалярных наблюдаемых функций, он является двойственным, или левосопряженным к оператору Перрона–Фробениуса, или оператору переноса – pushforward-оператору в пространстве функций плотности вероятности. Если для представления оператора Купмана выбран полиномиальный базис, то возникает тесная связь с линеаризацией Карлемана [121, 122, 123], которая была детально изучена в теории нелинейного управления [500, 305, 38, 509]. Анализ Купмана связан также с теорией резольвентного оператора в гидродинамике [487].

Недавно было показано, что теоретико-операторный подход дополняет традиционные геометрический и вероятностный подходы. Например, мно-

жества уровня собственных функций Купмана образуют инвариантные разбиения пространства состояний динамической системы [103]; в частности, собственные функции оператора Купмана можно использовать для анализа эргодического разбиения [380, 102]. Также недавно было установлено, что анализ Купмана обобщает теорему Хартмана–Гробмана на всю область притяжения точки устойчивого или неустойчивого равновесия или периодической орбиты [322].

На момент написания книги представление собственных функций Купмана для динамических систем общего вида оставалось центральной нерешенной проблемой. Немало усилий направлено на разработку управляемых данными методов идентификации собственных функций Купмана и их использование для управления. Эту тему мы обсудим в последующих разделах и главах. Недавно появились работы, в которых делается попытка применить всю мощь глубокого обучения для идентификации и представления собственных функций на основе данных [550, 368, 513, 564, 412, 349].

7.5. Управляемый данными анализ Купмана

Нахождение линейного представления существенно нелинейных систем может совершить революцию в нашей способности предсказывать поведение таких систем и управлять ими. Линеаризация динамики вблизи неподвижных точек или периодических орбит уже давно используется для *локального* представления динамики [252]. Оператор Купмана подкупает тем, что предлагает *глобально* линейное представление, в том числе и вдали от неподвижных точек и периодических орбит. Однако все предыдущие попытки получить конечномерные аппроксимации оператора Купмана имели ограниченный успех. Цель разложения по динамическим модам [472, 456, 317] – найти наилучшую линейную аппроксимацию оператора Купмана, экстраполирующую пространственные измерения на следующий момент времени, хотя для многих нелинейных систем одних лишь линейных измерений недостаточно. Дополнение DMD нелинейными измерениями может обогатить модель, но нет гарантии, что результирующая модель будет замкнута относительно оператора Купмана [92]. Ниже мы опишем несколько подходов к идентификации погружений и собственных функций Купмана на основе данных. К ним относятся расширенное разложение по динамическим модам [556], расширения, основанные на методе SINDy [276], и использование запаздывающих координат [91].

Расширенный DMD

Алгоритм расширенного DMD [556] по существу совпадает с обычным DMD [535] – с тем отличием, что вместо выполнения регрессии прямых измерений состояния регрессия применяется к дополненному вектору, содержащему нелинейные измерения. Как уже было сказано, eDMD эквивалентен вариацион-

ному подходу конформационной динамики [405, 407, 408], разработанному в 2013 году Ноэ и Нюске.

Далее мы немного изменим нотацию, приспособив ее к рассматриваемым методам. В eDMD строится дополненное состояние:

$$\mathbf{y} = \boldsymbol{\Theta}^T(\mathbf{x}) = \begin{bmatrix} \theta_1(\mathbf{x}) \\ \theta_2(\mathbf{x}) \\ \vdots \\ \theta_p(\mathbf{x}) \end{bmatrix}. \tag{7.85}$$

$\boldsymbol{\Theta}$ может содержать исходное состояние $\mathbf{x}$, а также нелинейные измерения, поэтому зачастую $p \gg n$. Далее, как и в DMD, строятся две матрицы:

$$\mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_m \\ | & | & & | \end{bmatrix}, \qquad \mathbf{Y}' = \begin{bmatrix} | & | & & | \\ \mathbf{y}_2 & \mathbf{y}_3 & \cdots & \mathbf{y}_{m+1} \\ | & | & & | \end{bmatrix}. \tag{7.86a}$$

Наконец, ищется наилучший линейный оператор $\mathbf{A}_\mathbf{Y}$, отображающий $\mathbf{Y}$ в $\mathbf{Y}'$:

$$\mathbf{A}_\mathbf{Y} = \underset{\mathbf{A}_\mathbf{Y}}{\operatorname{argmin}} ||\mathbf{Y}' - \mathbf{A}_\mathbf{Y}\mathbf{Y}|| = \mathbf{Y}'\mathbf{Y}^\dagger. \tag{7.87}$$

Эту регрессию можно записать в терминах матриц данных $\boldsymbol{\Theta}(\mathbf{X})$ и $\boldsymbol{\Theta}(\mathbf{X}')$:

$$\mathbf{A}_\mathbf{Y} = \underset{\mathbf{A}_\mathbf{Y}}{\operatorname{argmin}} ||\boldsymbol{\Theta}^T(\mathbf{X}') - \mathbf{A}_\mathbf{Y}\boldsymbol{\Theta}^T(\mathbf{X})|| = \boldsymbol{\Theta}^T(\mathbf{X}')\left(\boldsymbol{\Theta}^T(\mathbf{X})\right)^\dagger. \tag{7.88}$$

Поскольку дополненный вектор $\mathbf{y}$ может быть значительно больше вектора состояния $\mathbf{x}$, для вычисления этой регрессии часто применяются ядерные методы [557]. В принципе, обогащенная библиотека $\boldsymbol{\Theta}$ предлагает более широкий базис для аппроксимации оператора Купмана. Недавно было показано, что в пределе, когда снимки бесконечны, расширенный оператор DMD сходится к проекции оператора Купмана на подпространство, натянутое на $\boldsymbol{\Theta}$ [303]. Однако если линейная оболочка $\boldsymbol{\Theta}$ не является инвариантным подпространством Купмана, то спроецированный оператор может не иметь никакого сходства с оригинальным оператором Купмана, поскольку все собственные значения и собственные векторы могут различаться. На самом деле было доказано, что расширенный оператор DMD будет иметь паразитные собственные значения и векторы, если он не представлен в терминах инвариантного подпространства Купмана [92]. Поэтому так важно использовать методы контроля и перекрестной проверки, чтобы модель eDMD не оказалась переобученной; мы обсудим этот вопрос ниже. Например, было показано, что eDMD не может содержать оригинальное состояние $\mathbf{x}$ в качестве измерения и представлять систему, имеющую несколько неподвижных точек, периодических орбит или других аттракторов, потому что такие системы не могут быть топологически сопряженными ни с какой конечномерной линейной системой [92].

Аппроксимация собственных функций Купмана на основе данных

В системах с дискретным временем собственная функция Купмана $\varphi(\mathbf{x})$, вычисленная в нескольких точках $\mathbf{X}$, удовлетворяет соотношению

$$\begin{bmatrix} \lambda_\varphi(\mathbf{x}_1) \\ \lambda_\varphi(\mathbf{x}_2) \\ \vdots \\ \lambda_\varphi(\mathbf{x}_m) \end{bmatrix} = \begin{bmatrix} \varphi(\mathbf{x}_1) \\ \varphi(\mathbf{x}_2) \\ \vdots \\ \varphi(\mathbf{x}_{m+1}) \end{bmatrix}. \tag{7.89}$$

Эту собственную функцию можно аппроксимировать в виде разложения по множеству функций-кандидатов

$$\boldsymbol{\Theta}(\mathbf{x}) = [\theta_1(\mathbf{x}) \quad \theta_2(\mathbf{x}) \quad \cdots \quad \theta_p(\mathbf{x})]. \tag{7.90}$$

Собственную функцию Купмана можно аппроксимировать в этом базисе как

$$\varphi(\mathbf{x}) \approx \sum_{k=1}^{p} \theta_k(\mathbf{x})\xi_k = \boldsymbol{\Phi}(\mathbf{x})\boldsymbol{\xi}. \tag{7.91}$$

Запись (7.89) в терминах этого разложения дает матричную систему уравнений:

$$(\lambda\boldsymbol{\Theta}(\mathbf{X}) - \boldsymbol{\Theta}(\mathbf{X}'))\boldsymbol{\xi} = \mathbf{0}. \tag{7.92}$$

Если мы хотим найти аппроксимацию (7.92) методом *наименьших квадратов*, то задача сводится к расширенному DMD [557, 556]:

$$\lambda\boldsymbol{\xi} = \boldsymbol{\Theta}(\mathbf{X})^\dagger\boldsymbol{\Theta}(\mathbf{X}')\boldsymbol{\xi}. \tag{7.93}$$

Заметим, что (7.93) получается транспонированием (7.88), поэтому левые собственные векторы становятся правыми. Таким образом, собственные векторы $\boldsymbol{\xi}$ матрицы $\boldsymbol{\Theta}^\dagger\boldsymbol{\Theta}'$ дают коэффициенты собственной функции $\varphi(\mathbf{x})$, представленной в базисе $\boldsymbol{\Theta}(\mathbf{x})$. Затем абсолютно необходимо удостовериться, что предсказанные собственные функции ведут себя линейно на траекториях, для чего нужно сравнить их с предсказанной динамикой $\varphi_{k+1} = \lambda\varphi_k$, т. к. описанная выше регрессия порождает паразитные собственные значения и векторы, если только линейной оболочкой элементов базиса θ_j не является инвариантное подпространство Купмана [92].

Разреженная идентификация собственных функций

Для идентификации собственных функций Купмана, соответствующих конкретному собственному значению λ, мы можем воспользоваться SINDy-регрессией [95], когда из библиотеки $\boldsymbol{\Theta}(\mathbf{x})$ выбирается лишь небольшое число

активных членов, чтобы предотвратить переобучение. Если заданы матрицы данных $\mathbf{X}$ и $\dot{\mathbf{X}}$, то можно построить библиотеку базисных функций $\boldsymbol{\Theta}(\mathbf{X})$ и библиотеку производных по направлению, представляющие возможные члены $\nabla\varphi(\mathbf{x})\cdot\mathbf{f}(\mathbf{x})$ в уравнении (7.60):

$$\boldsymbol{\Gamma}(\mathbf{x},\dot{\mathbf{x}}) = [\nabla\theta_1(\mathbf{x})\cdot\dot{\mathbf{x}} \quad \nabla\theta_2(\mathbf{x})\cdot\dot{\mathbf{x}} \quad \cdots \quad \nabla\theta_p(\mathbf{x})\cdot\dot{\mathbf{x}}]. \tag{7.94}$$

Затем можно построить $\boldsymbol{\Gamma}$ по данным:

$$\boldsymbol{\Gamma}(\mathbf{X},\dot{\mathbf{X}}) = \begin{bmatrix} \nabla\theta_1(\mathbf{x}_1)\cdot\dot{\mathbf{x}}_1 & \nabla\theta_2(\mathbf{x}_1)\cdot\dot{\mathbf{x}}_1 & \cdots & \nabla\theta_p(\mathbf{x}_1)\cdot\dot{\mathbf{x}}_1 \\ \nabla\theta_1(\mathbf{x}_2)\cdot\dot{\mathbf{x}}_2 & \nabla\theta_2(\mathbf{x}_2)\cdot\dot{\mathbf{x}}_2 & \cdots & \nabla\theta_p(\mathbf{x}_2)\cdot\dot{\mathbf{x}}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \nabla\theta_1(\mathbf{x}_m)\cdot\dot{\mathbf{x}}_m & \nabla\theta_2(\mathbf{x}_m)\cdot\dot{\mathbf{x}}_m & \cdots & \nabla\theta_p(\mathbf{x}_m)\cdot\dot{\mathbf{x}}_m \end{bmatrix}.$$

Для заданного собственного значения λ в УрЧП Купмана (7.60) можно подставить данные:

$$(\lambda\boldsymbol{\Theta}(\mathbf{X}) - \boldsymbol{\Gamma}(\mathbf{X},\dot{\mathbf{X}}))\boldsymbol{\xi} = \mathbf{0}. \tag{7.95}$$

Формулировка (7.95) неявная и означает, что $\boldsymbol{\xi}$ принадлежит ядру матрицы $\lambda\boldsymbol{\Theta}(\mathbf{X}) - \boldsymbol{\Gamma}(\mathbf{X},\dot{\mathbf{X}})$. Правое ядро (7.95) для данного λ натянуто на правые сингулярные векторы $\lambda\boldsymbol{\Theta}(\mathbf{X}) - \boldsymbol{\Gamma}(\mathbf{X},\dot{\mathbf{X}}) = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*$ (т. е. столбцы $\mathbf{V}$), соответствующие нулевым сингулярным значениям. Идентифицировать небольшое число активных членов в собственной функции удастся, если отыскать самый разреженный вектор в ядре матрицы [440], как в неявном алгоритме SINDy [361], описанном в разделе 7.3. В этой постановке собственные значения λ априори неизвестны, их необходимо найти в процессе обучения с приближенной собственной функцией. Собственные функции и собственные значения Купмана можно также определить, решив задачу $\mathbf{A}_\mathbf{Y}\boldsymbol{\xi}_\alpha = \lambda_\alpha\boldsymbol{\xi}_\alpha$, где $\mathbf{A}_\mathbf{Y} = \boldsymbol{\Theta}^\dagger\boldsymbol{\Gamma}$ получена с помощью регрессии наименьших квадратов, как в версии eDMD с непрерывным временем. Многие собственные функции паразитные, но те, что соответствуют слабо демпфированным собственным значениям, можно аппроксимировать хорошо.

С практической точки зрения, данные в $\mathbf{X}$ необязательно выбирать из полных траекторий, для их получения можно использовать более изощренные стратегии, например выборку латинского гиперкуба или выборку из распределения в фазовом пространстве. Кроме того, гильбертово пространство с воспроизводящим ядром можно использовать для описания $\varphi(\mathbf{x})$ *локально* в некоторых участках пространства состояний.

Пример: система Дуффинга (Kaiser et al [276])

Мы продемонстрируем разреженную идентификацию собственных функций Купмана на примере незатухающего осциллятора Дуффинга:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1 - x_1^3 \end{bmatrix},$$

где x_1 – положение, а x_2 – скорость частицы, движущейся в двухъямном потенциале с точками равновесия (0, 0) и (±1, 0). Это консервативная система, ее гамильтониан $\mathcal{H} = 1/2x_2^2 - 1/2x_1^2 + 1/4x_1^4$. Гамильтониан и вообще любая сохраняемая величина – собственная функция Купмана с нулевым собственным значением.

Для собственного значения $\lambda = 0$ (7.95) принимает вид $-\mathbf{\Gamma}(\mathbf{X}, \dot{\mathbf{X}})\boldsymbol{\xi} = \mathbf{0}$, а значит, мы ищем разреженный вектор $\boldsymbol{\xi}$ в ядре матрицы $-\mathbf{\Gamma}(\mathbf{X}, \dot{\mathbf{X}})$. Библиотека функций-кандидатов строится по данным и состоит из полиномов степени не выше 4:

$$\mathbf{\Theta}(\mathbf{X}) = \begin{bmatrix} | & | & | & | & | \\ x_1(t) & x_2(t) & x_1^2(t) & x_1(t)x_2(t) & \cdots & x_2^4(t) \\ | & | & | & | & | \end{bmatrix}$$

и

$$\mathbf{\Gamma}(\mathbf{X}, \dot{\mathbf{X}}) = \begin{bmatrix} | & | & | & | & | \\ \dot{x}_1(t) & \dot{x}_2(t) & 2x_1(t)\dot{x}_1(t) & x_2(t)\dot{x}_1(t) + x_1(t) + \dot{x}_2(t) & \cdots & 4x_2(t)^3\dot{x}_2(t) \\ | & | & | & | & | \end{bmatrix}.$$

Можно найти разреженный вектор коэффициентов $\boldsymbol{\xi}$, так что несколько его ненулевых элементов будут определять активные члены собственной функции Купмана. Идентифицированная собственная функция Купмана с собственным значением $\lambda = 0$ имеет вид:

$$\varphi(\mathbf{x}) = -2/3x_1^2 + 2/3x_2^2 + 1/3x_1^4. \tag{7.96}$$

Эта собственная функция совпадает с гамильтонианом с точностью до постоянного множителя.

Управляемый данными анализ Купмана и запаздывающие координаты

Вместо того чтобы прямо экстраполировать моментальные линейные или нелинейные измерения состояния системы, как в DMD, можно получить внутренние координаты измерения для анализа Купмана, основанные на запаздывающих измерениях системы [506, 91, 18, 144]. Этот подход можно считать управляемым данными, он опирается на большой объем информации из прошлых измерений для прогнозирования будущего. В отличие от линейных или слабо нелинейных систем, когда траектории могут притягиваться к неподвижным точкам или периодическим орбитам, для этого анализа особенно хорошо подходит хаотическая динамика; эволюционирующие траектории плотно заполняют аттрактор, поэтому чем больше данных, тем больше информации. Особенно важно использование запаздывающих координат в системах с эффектом долговременной памяти, для которых, как было недавно показано, анализ Купмана проявил себя очень успешно [508]. Интересно, что связь между

оператором Купмана и погружением Такенса исследовали еще в 2004 году [379], когда был определен стохастический оператор Купмана и доказана статистическая теорема Такенса.

Схематически запаздывающие измерения изображены на рис. 7.12 на примере системы Лоренца для временного ряда измерений первой переменной $x(t)$. Условия теоремы Такенса о погружении удовлетворяются [515], поэтому существует диффеоморфизм между погруженным с задержкой аттрактором и аттрактором в исходных координатах. Затем мы получаем собственные запаздывающие координаты из одного временного ряда измерений $x(t)$, выполнив сингулярное разложение матрицы Ганкеля $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} x(t_1) & x(t_2) & \cdots & x(t_{m_c}) \\ x(t_2) & x(t_3) & \cdots & x(t_{m_c+1}) \\ \vdots & \vdots & \ddots & \vdots \\ x(t_{m_0}) & x(t_{m_o+1}) & \cdots & x(t_m) \end{bmatrix} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*. \tag{7.97}$$

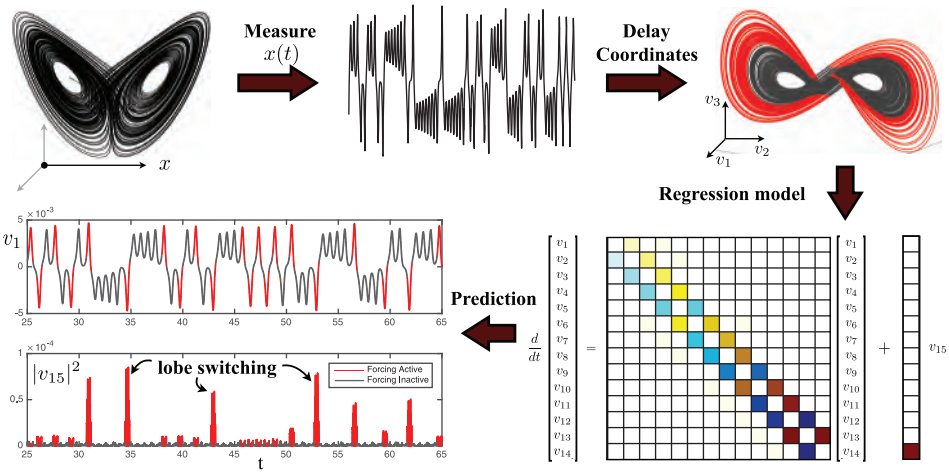

Рис. 7.12 ❖ Разложение хаоса в линейную систему с вынуждением. Временной ряд $x(t)$ упаковывается в матрицу Ганкеля $\mathbf{H}$. Сингулярное разложение $\mathbf{H}$ дает иерархию *собственных* временных рядов, которая порождает погруженный с запаздыванием аттрактор. По запаздывающим координатам $\mathbf{v}$ строится наилучшая модель линейной регрессии; линейная аппроксимация для первых $r - 1$ переменных очень хороша, но для последней координаты v_r линейная модель не отражает действительности. На самом деле v_r является входом, который оказывает вынуждающее воздействие на первые $r - 1$ переменных. Редкие события вынуждения соответствуют переключению лепестков в хаотической динамике. Эта архитектура, называемая альтернативным представлением Ганкеля анализа Купмана (Hankel alternative view of Koopman – HAVOK), описана в работе [91]. *Рисунок взят из работы Brunton et al. [91]*

Столбцы матриц $\mathbf{U}$ и $\mathbf{V}$ упорядочены иерархически по их способности моделировать столбцы и строки $\mathbf{H}$ соответственно. Часто $\mathbf{H}$ допускает низ-

кранговую аппроксимацию первыми r столбцами $\mathbf{U}$ и $\mathbf{V}$. Отметим, что матрица Ганкеля (7.97) – основа алгоритма реализации собственной системы [272] в теории идентификации линейных систем (см. раздел 9.3) и алгоритма сингулярного спектрального анализа (singular spectrum analysis – SSA) [88] в анализе климатических временных рядов.

Низкоранговая аппроксимация матрицы (7.97) – это *управляемая данными* система измерений, приближенно инвариантная относительно оператора Купмана для состояний, принадлежащих аттрактору. По определению, динамика отображает аттрактор в себя, делая его *инвариантным* к потоку. Иными словами, столбцы $\mathbf{U}$ образуют инвариантное подпространство Купмана. Мы можем переписать (7.97) с помощью оператора Купмана $\mathcal{K} \triangleq \mathcal{K}_{\Delta t}$:

$$\mathbf{H} = \begin{bmatrix} x(t_1) & \mathcal{K}x(t_1) & \cdots & \mathcal{K}^{m_c-1}x(t_1) \\ \mathcal{K}x(t_1) & \mathcal{K}^2x(t_1) & \cdots & \mathcal{K}^{m_c}x(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{K}^{m_o-1}x(t_1) & \mathcal{K}^{m_o}x(t_1) & \cdots & \mathcal{K}^{m-1}x(t_1) \end{bmatrix}. \tag{7.98}$$

Столбцы (7.97) хорошо аппроксимируются первыми r столбцами $\mathbf{U}$. Первые r столбцов $\mathbf{V}$ дают временной ряд абсолютных величин каждого столбца $\mathbf{U\Sigma}$ в данных. Нанеся на график первые три столбца $\mathbf{V}$, мы получим погруженный аттрактор для системы Лоренца (см. рис. 7.12).

Связь между собственными запаздывающими координатами из (7.97) и оператором Купмана наводит на мысль о модели линейной регрессии по переменным в $\mathbf{V}$. Даже при наличии системы измерений, приближенно инвариантной относительно оператора Купмана, остается проблема идентификации линейной модели для хаотической системы. Линейная модель, сколь угодно детальная, не может отразить несколько неподвижных точек или непредсказуемое поведение, характерное для хаоса с положительным показателем Ляпунова [92]. Вместо построения замкнутой линейной модели для первых r переменных в $\mathbf{V}$ мы строим линейную модель по первым $r-1$ переменным, а последнюю переменную v_r интерпретируем как вынуждающий член:

$$\frac{d}{dt}\mathbf{v}(t) = \mathbf{A}\mathbf{v}(t) + \mathbf{B}v_r(t), \tag{7.99}$$

где $\mathbf{v} = [v_1 \;\; v_2 \;\; \cdots \;\; v_{r-1}]^T$ – вектор, состоящий из первых $r-1$ собственных запаздывающих координат. В другой работе было исследовано разделение динамики на детерминированную линейную и стохастическую хаотическую [376].

Во всех примерах, рассмотренных в работе [91], линейная модель, построенная по первым $r-1$ членам, точна, но никакая линейная модель не может представить v_r. Вместо этого v_r является вынуждающим входом для линейной динамики в (7.99), которая аппроксимирует нелинейную динамику. Статистика $v_r(t)$ негауссова, а длинные хвосты соответствуют редким событиям вынуждения, которые отвечают за переключение лепестков в системе Лоренца; это связано с распределениями редких событий вынуждения, которые

наблюдались и были смоделированы в других работах [355, 461, 356]. Линейная система с вынуждением (7.99) была идентифицирована в результате применения алгоритма SINDy [95] к запаздывающим координатам системы Лоренца. Дальнейшее применение анализа Купмана к запаздывающим координатам является многообещающим в контексте управления с обратной связью, где, возможно, позволит управлять поведением хаотической системы, рассматривая v_r как возмущение.

Кроме того, использование запаздывающих координат в качестве внутренних измерений в анализе Купмана наводит на мысль, что теорию Купмана можно использовать для совершенствования технологий пространственно распределенных датчиков. Например, в случае пространственной матрицы, содержащей порядка 100 тензометрических датчиков на крыльях летающих насекомых, можно использовать фазовые запаздывающие координаты для вычисления близкого к оптимальному размещения, которое позволит измерять и регулировать конвективные структуры (например, срыв потока из-за порыва ветра, формирование вихрей на передней кромке крыла и турбулентную конвекцию и т. д.).

Код HAVOK для системы Лоренца

Ниже приведен код построения модели HAVOK для системы Лоренца с теми же данными, что в листинге 7.2. Мы взяли $\Delta t = 0.01$, $m_o = 10$, $r = 10$, хотя при $\Delta t = 0.001$, $m_o = 100$, $r = 15$ результаты могли бы получиться более точными.

Листинг 7.3 ❖ Код HAVOK для системы Лоренца с теми же данными, что в разделе 7.1

```
%% Собственные запаздывающие координаты
stackmax = 10; % число строк матрицы
r=10;          % ранг модели HAVOK
H = zeros(stackmax,size(x,1)-stackmax);
for k=1:stackmax
    H(k,:) = x(k:end-stackmax-1+k,1);
end
[U,S,V] = svd(H,'econ'); % собственные запаздывающие координаты

%% Вычисление производных (центральные разности 4-го порядка)
dV = zeros(length(V)-5,r);
for i=3:length(V)-3
    for k=1:r
        dV(i-2,k) = (1/(12*dt))*(-V(i+2,k)+8*V(i+1,k)-8*V(i-1,k)+V(i-2,k));
    end
end
% Отбросить два первых и два последних столбца
V = V(3:end-3,1:r);

%% Построить модель HAVOK-регрессии по запаздывающим координатам
Xi = V\dV;
A = Xi(1:r-1,1:r-1)';
B = Xi(end,1:r-1)';
```

Нейронные сети для погружений Купмана

Несмотря на все надежды, связанные с погружениями Купмана, получение практически реализуемых представлений остается серьезной проблемой. Напомним, что даже для сравнительно простых динамических систем собственные функции оператора Купмана могут быть сколь угодно сложными. Глубокое обучение, хорошо приспособленное для представления произвольных функций, стало многообещающим подходом к идентификации и представлению собственных функций Купмана [550, 368, 513, 564, 412, 332, 349], поскольку позволяет на основе данных отыскать погружение существенно нелинейных систем во внутренние линейные координаты. В частности, подход Купмана естественно укладывается в структуру глубокого автокодировщика, обсуждавшуюся в главе 6, когда выявляется небольшое число ключевых латентных переменных $\mathbf{y} = \boldsymbol{\varphi}(\mathbf{x})$, параметризующих динамику. В сети Купмана налагается дополнительное ограничение – динамика должна линейно зависеть от латентных переменных, вследствие чего функции $\boldsymbol{\varphi}(\mathbf{x})$ оказываются собственными функциями Купмана, как показано на рис. 7.13. Ограничение линейности динамики реализуется путем задания функции потерь $||\boldsymbol{\varphi}(\mathbf{x}_{k+1}) - \mathbf{K}\boldsymbol{\varphi}(\mathbf{x}_k)||$, где $\mathbf{K}$ – некоторая матрица. В общем случае линейность поддерживается на нескольких временных шагах, так что в процессе повторного применения $\mathbf{K}$ к латентным переменным запоминается траектория. Кроме того, важно, чтобы можно было выполнить обратное отображение в физические переменные $\mathbf{x}$, именно поэтому предпочтителен автокодировщик [349]. Для стохастических динамических систем, например в молекулярной динамике, когда обратное отображение латентных переменных в физическую конфигурацию имеет вероятностный характер, использовались также вариационные автокодировщики [550, 368].

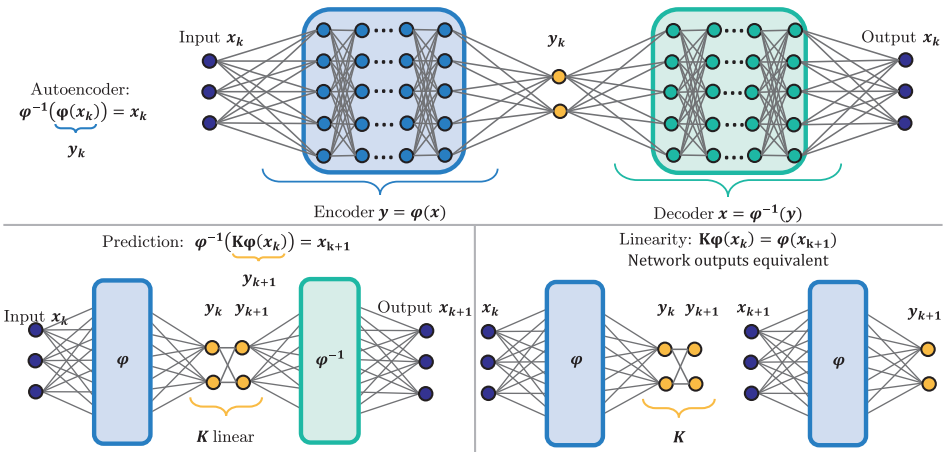

Рис. 7.13 ❖ Архитектура глубокой нейронной сети для идентификации собственных функций Купмана $\boldsymbol{\varphi}(\mathbf{x})$. Сеть основана на глубоком автокодировщике (a), который идентифицирует внутренние координаты $\mathbf{y} = \boldsymbol{\varphi}(\mathbf{x})$. Включены дополнительные функции потерь, чтобы переменные автокодировщика описывали линейную динамику (b, c). *Взято из работы Lusch et al. [349] с разрешения авторов*

Для простых систем с дискретным спектром можно получить компактное представление в терминах небольшого числа переменных автокодировщика. Однако для динамических систем с непрерывным спектром не удается найти низкоразмерные представления с применением многих известных нейронных сетей или представлений Купмана. Динамические системы с непрерывным спектром встречаются повсеместно – от простого маятника до нелинейной оптики и широкополосной турбулентности. Например, у классического маятника, описываемого уравнением

$$\ddot{x} = -\sin(\omega x), \tag{7.100}$$

частота непрерывно изменяется от ω до 0 по мере увеличения амплитуды колебаний маятника. Таким образом, непрерывный спектр мешает получить простое описание в терминах собственных функций Купмана [378]. Действительно, при удалении от линейного режима для аппроксимации частотного сдвига необходимо бесконечно много членов ряда Фурье.

В недавней работе Lusch et al. [349] для параметризации непрерывно изменяющихся собственных значений используется вспомогательная сеть, в результате чего получается экономная сеть, допускающая интерпретацию. Эта параметризованная сеть схематически изображена на рис. 7.14, а ее применение к простому маятнику показано на рис. 7.15. В отличие от других структур сети, в которых требуется большой автокодировочный слой для кодирования сдвига непрерывной частоты асимптотическим разложением по гармоникам собственной частоты, эта параметризованная сеть способна идентифицировать единственную пару комлексно сопряженных собственных функций, соответствующих изменяющейся паре мнимых собственных значений. Если не учитывать эту явную зависимость от частоты, то необходима сеть высокой размерности для объяснения изменяющейся частоты и собственных значений.

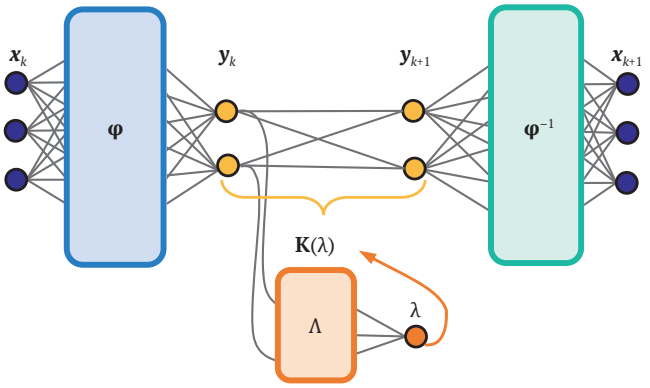

Рис. 7.14 ❖ Модифицированная архитектура нейронной сети для параметризации системы с непрерывным спектром. Благодаря непрерывности собственных значений λ возможно агрессивное понижение размерности в автокодировщике, что позволяет избежать высших гармоник собственной частоты, порождаемых нелинейностью. *Взято из работы Lusch et al. [349] с разрешения авторов*

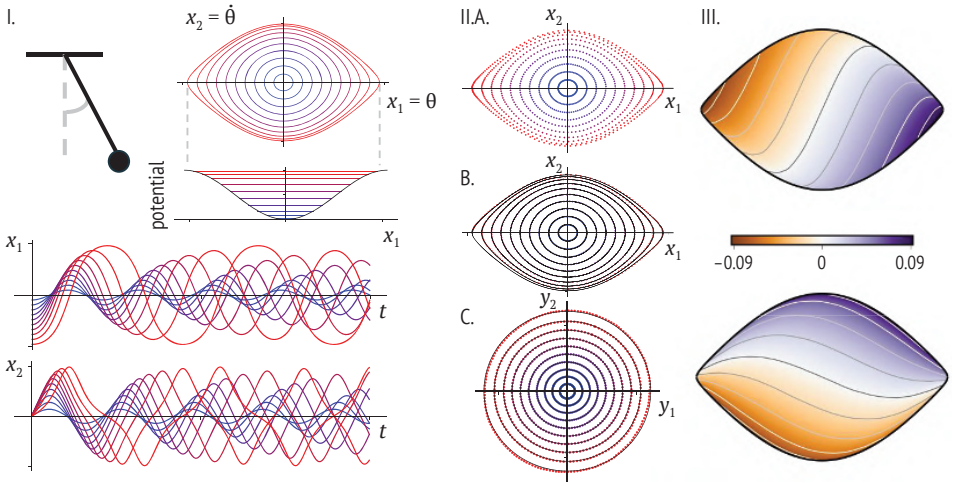

Рис. 7.15 ❖ Погружение нелинейного маятника с помощью параметризованной сети на рис. 7.14. По мере увеличения амплитуды колебаний частота непрерывно изменяется (I). В координатах собственных функций Купмана (III) динамика становится линейной и представлена идеальными окружностями (IIIC).
Взято из работы Lusch et al. [349] с разрешения авторов

Ожидается, что представления динамических систем нейронными сетями и, в частности, погружения Купмана будут и дальше оставаться в фокусе внимания специалистов по управляемой данными динамике. Сочетание способности глубокого обучения находить представления с элегантностью и простотой погружений Купмана потенциально может совершить переворот в анализе сложных систем и управлении ими.

Рекомендуемая литература

Учебники

(1) **Nonlinear oscillations, dynamical systems, and bifurcations of vector fields**, by P. Holmes and J. Guckenheimer, 1983 [252].
(2) **Dynamic mode decomposition: Data-driven modeling of complex systems**, by J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, 2016 [317].
(3) **Differential equations and dynamical systems**, by L. Perko, 2013 [427].

Статьи и обзоры

(1) **Distilling free-form natural laws from experimental data**, by M. Schmidt and H. Lipson, *Science*, 2009 [477].
(2) **Discovering governing equations from data by sparse identification of nonlinear dynamical systems**, by S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Proceedings of the National Academy of Sciences*, 2016 [95].

(3) On dynamic mode decomposition: theory and applications, by J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, J. N. Kutz, *Journal of Computational Dynamics*, 2014 [535].

(4) Hamiltonian systems and transformation in Hilbert Space, by B. O. Koopman, *Proceedings of the National Academy of Sciences*, 1931 [300].

(5) Spectral properties of dynamical systems, model reduction and decompositions, by I. Mezi´c, *Nonlinear Dynamics*, 2005 [376].

(6) Data-driven model reduction and transfer operator approximation, by S. Klus, F. Nuske, P. Koltai, H. Wu, I. Kevrekidis, C. Schutte, and F. Noe, *Journal of Nonlinear Dynamics*, 2018 [293].

(7) Hidden physics models: Machine learning of nonlinear partial differential equations, by M. Raissi and G. E. Karniadakis, *Journal of Computational Physics*, 2018 [445].

Глава 8

Теория линейного управления

В центре внимания этой книги находится характеризация сложных систем с помощью понижения размерности, разреженной выборки и моделирования динамических систем. Но во многих системах главная задача – способность активно управлять поведением, стремясь достичь поставленной инженерами цели. Изучение и практическое манипулирование динамическими системами называется теорией управления, это одна из наиболее успешных дисциплин на стыке прикладной математики и практической инженерии. Теория управления неотделима от науки о данных, поскольку опирается на показания датчиков (данные), получаемые от системы во имя достижения заданной цели. На самом деле теория управления имеет дело с постоянно изменяющимися данными, поскольку успешное воздействие изменяет динамику системы, а следовательно, и характеристики измерений. Теория управления заставляет столкнуться с реальностью, так как проверяет упрощающие предположения и приближенные модели.

Теория управления способствовала формированию современного технологического и индустриального ландшафта. Примеров не счесть: круиз-контроль в автомобилях, позиционирование в строительной технике, электродистанционное управление летательным аппаратом, автоматизация в промышленности, маршрутизация пакетов в интернете, коммерческие системы отопления, вентиляции и охлаждения, стабилизация ракет, ПИД-регуляторы температуры и давления в современных кофе-машинах – и это лишь малая толика приложений. В будущем теория управления будет все шире применяться к многомерным, существенно нелинейным и многомасштабным задачам, например в теории турбулентности, нейронауках, финансах, эпидемиологии, автономных роботах и беспилотных автомобилях. В этих будущих приложениях управляемое данными моделирование и управление будут иметь решающее значение; это тема глав 7 и 10.

В данной главе мы рассмотрим основные концепции замкнутых систем управления с обратной связью. Наша цель – на примерах практических проблем выработать интуитивное понимание того, когда и как использовать управление с обратной связью. В основном мы будем разрабатывать теорию линейных систем, для которых существует немало мощных методов [165, 492]. Мы продемонстрируем применение теории на простых, интуи-

тивно понятных примерах, например разработка системы круиз-контроля для автомобиля и стабилизация обратного маятника, смонтированного на движущейся тележке.

Типы управления

Существует много способов манипулирования поведением динамических систем, эти подходы к управлению схематически изображены на рис. 8.1. Для пассивного управления не нужно подводить энергию, и если этого достаточно, то такой подход желателен по причине простоты, надежности и низкой стоимости. Пример – знаки «стоп» на дорожных перекрестках для регулирования движения транспорта. Для активного управления подводить энергию нужно, такие устройства управления (регуляторы) делятся на два больших класса в зависимости от того, нужны датчики для передачи информации регулятору или нет. В первом случае мы имеем управление с разомкнутым контуром, которое зависит от заранее запрограммированной управляющей последовательности; в примере регулирования дорожного движения можно запрограммировать смену сигналов в зависимости от времени суток. Во втором случае активная система управления использует датчики для формирования закона управления. В системах упреждающего регулирования по возмущению измеряются экзогенные возмущения системы, и показания передаются регулятору системы с разомкнутым контуром; примером упреждающего регулирования может служить упреждающее изменение направления транспортных потоков вблизи стадиона перед выходом большого количества людей. Наконец, имеется управление с замкнутым контуром обратной

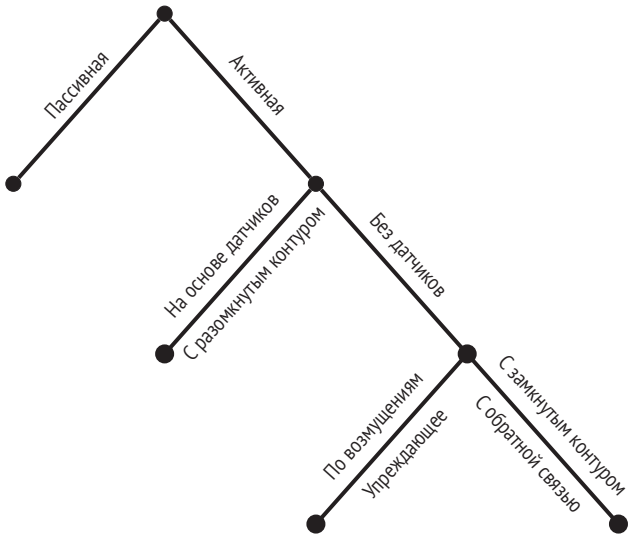

Рис. 8.1 ❖ Различные типы управления. В этой главе речь идет в основном об управлении с замкнутым контуром обратной связи

связи, которое и будет основной темой этой главы. В системах управления с замкнутым контуром используются датчики для прямого измерения системы, их показания служат для формирования управляющего воздействия, зависящего от того, насколько близка система к достижению поставленной цели. В состав многих современных транспортных систем входят «умные» светофоры с управляющими схемами, на которые поступает информация с индуктивных датчиков плотности транспортного потока, встроенных в дорожное покрытие.

8.1. Управление с замкнутым контуром обратной связи

В этой главе мы уделим основное внимание управлению с замкнутым контуром обратной связи. Так обычно строятся системы, для которых характерны неопределенность, неустойчивость и (или) внешние помехи. На рис. 8.2 показана общая схема управления с обратной связью – показания датчиков **y** подаются обратно на вход регулятора, который принимает решение об исполнительной команде **u** для управления динамикой и обеспечения надежного функционирования, несмотря на присутствующую в модели неопределенность и экзогенные возмущения. Во всех примерах, обсуждаемых в этой главе, вектор экзогенных возмущений можно разложить на компоненты $\mathbf{w} = [\mathbf{w}_d^T \quad \mathbf{w}_n^T \quad \mathbf{w}_r^T]^T$, где $\mathbf{w}_d$ – возмущения состояния системы, $\mathbf{w}_n$ – шум измерения, а $\mathbf{w}_r$ – опорная траектория, к которой должна стремиться система с замкнутым контуром.

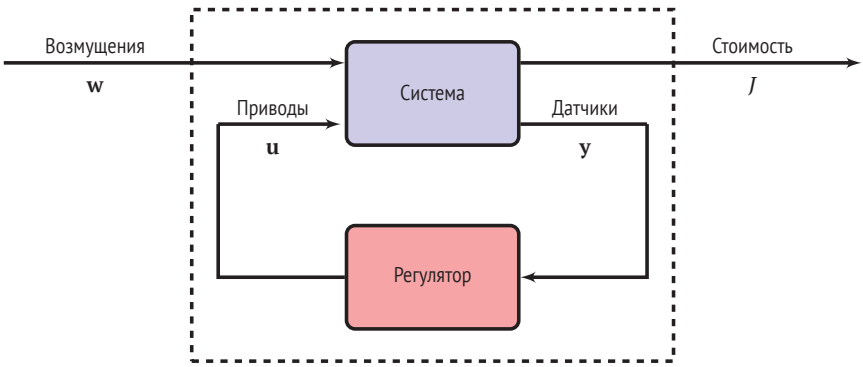

Рис. 8.2 ❖ Стандартная схема управления с обратной связью. Измерения системы $\mathbf{y}(t)$ подаются обратно регулятору, который определяет, какую исполнительную команду $\mathbf{u}(t)$ подать для управления системой. Проектируется закон управления, который изменяет динамику системы и обеспечивает правильное функционирование, несмотря на экзогенные возмущения и шум **w**. Качество управления оценивается стоимостью J. Экзогенные входные данные **w** могут также включать заданную траекторию $\mathbf{w}_r$, к которой должна стремиться система

Математически система и измерения обычно описываются динамической системой:

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}_d), \tag{8.1a}$$

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}, \mathbf{w}_n). \tag{8.1b}$$

Цель заключается в построении закона управления

$$\mathbf{u} = \mathbf{k}(\mathbf{y}, \mathbf{w}_r), \tag{8.2}$$

который доставляет минимум функции стоимости

$$J \triangleq J(\mathbf{x}, \mathbf{u}, \mathbf{w}_r). \tag{8.3}$$

Таким образом, современное управление опирается на методы оптимизации [74]. В общем случае регулятор (8.2) является динамической системой, а не статической функцией своих входов. Например, фильтр Калмана, описываемый в разделе 8.5, динамически оценивает полное состояние $\mathbf{x}$ по измерениям $\mathbf{u}$ и $\mathbf{y}$. В этом случае закон управления принимает вид $\mathbf{u} = \mathbf{k}(\mathbf{y}, \hat{\mathbf{x}}, \mathbf{w}_r)$, где $\hat{\mathbf{x}}$ – оценка полного состояния.

Чтобы оправдать дополнительную стоимость и сложность управления с обратной связью от датчиков, полезно сравнить его с управлением с разомкнутым контуром. Для задач с заданной траекторией регулятор проектируется так, чтобы направлять выход системы в сторону желаемого выходного значения $\mathbf{w}_r$ и тем самым минимизировать ошибку $\varepsilon = \mathbf{y} - \mathbf{w}_r$. В системе управления с разомкнутым контуром, показанной на рис. 8.3, модель системы направлена на выдачу исполнительной команды $\mathbf{u}$, порождающей желаемый выход. Однако эта заранее заданная стратегия не может скорректировать внешние возмущения и принципиально неспособна к изменению динамики. Поэтому с помощью управления с разомкнутым контуром невозможно стабилизировать неустойчивую систему, такую, например, как обратный маятник, поскольку модель системы должна быть точно известна, а сама система полностью изолирована от возмущений. Кроме того, любая неопределенность модели сразу же выливается в ошибку траектории системы с разомкнутым контуром.

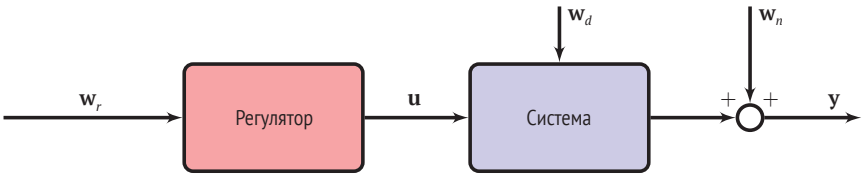

Рис. 8.3 ❖ Схема управления с разомкнутым контуром. Зная желаемый выходной сигнал $\mathbf{w}_r$, закон управления с разомкнутым контуром строит протокол управления системой $\mathbf{u}$ на основе некоторой модели. Внешние возмущения ($\mathbf{w}_d$) и шум в датчиках ($\mathbf{w}_n$), а также немоделируемая динамика системы и неопределенность не учитываются, что приводит к снижению качества работы

С другой стороны, в системе управления с замкнутым контуром обратной связи, показанной на рис. 8.4, измерения системы, поступающие от датчиков, используются для информирования регулятора о немоделируемой динамике и возмущениях, которые были бы фатальны для системы с разомкнутым контуром. Более того, при наличии обратной связи часто имеется возможность изменить и стабилизировать динамику замкнутой системы, что для системы с разомкнутым контуром невозможно. Таким образом, механизм управления с замкнутым контуром обратной связи часто способен поддерживать на должном уровне работу систем с неустойчивой динамикой, неопределенной моделью и внешними возмущениями.

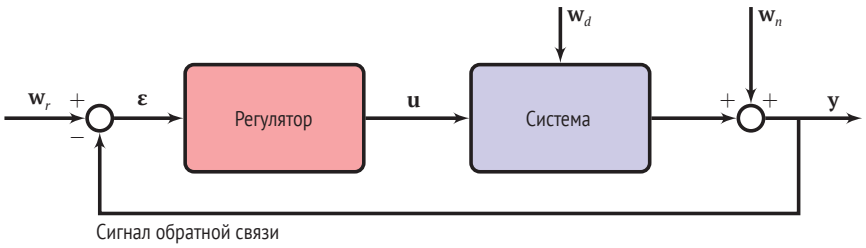

Рис. 8.4 ❖ Схема управления с замкнутым контуром. Сигнал датчика **y** подается обратно на вход и вычитается из опорного сигнала $\mathbf{w}_r$, что сообщает информацию о реакции системы на управляющее воздействие и внешние возмущения. Регулятор использует результирующую ошибку **ε**, чтобы определить правильную исполнительную команду **u** для получения желаемой реакции. Наличие обратной связи часто позволяет стабилизировать неустойчивую динамику путем компенсации возмущений $\mathbf{w}_d$ и ослабления шума $\mathbf{w}_n$

Примеры преимуществ управления с обратной связью

Подводя итоги, можно сказать, что управление с замкнутым контуром обратной связи имеет следующие преимущества по сравнению с управлением с разомкнутым контуром:

- есть возможность стабилизировать неустойчивую систему;
- можно компенсировать внешние возмущения;
- можно скорректировать немоделируемую динамику и смоделировать неопределенность.

Проиллюстрируем эти положения на двух простых примерах.

Обратный маятник. Рассмотрим уравнения неустойчивого обратного маятника, которые будут выведены в разделе 8.2. Линеаризованные уравнения имеют вид:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ g/L & d \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \tag{8.4}$$

где $x_1 = \theta$, $x_2 = \dot{\theta}$, u – крутящий момент, приложенный к рычагу маятника, g – ускорение силы тяжести, L – длина маятника, а d – гашение колебаний. Мы можем записать эту систему в стандартной форме:

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}u.$$

Если выбрать постоянные так, что собственная частота $\omega_n = \sqrt{g/L} = 1$ и $d = 0$, то система будет иметь собственные значения $\lambda = \pm 1$, соответствующие неустойчивой седловой неподвижной точке.

Никакая стратегия управления с разомкнутым контуром не может изменить динамику системы, определяемую собственными значениями $\mathbf{A}$. Однако если добавить обратную связь $u = -\mathbf{K}\mathbf{x}$, то получится система с замкнутым контуром

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}u = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}.$$

Если выбрать $\mathbf{K} = [4\ \ 4]$, что соответствует закону управления $u = -4x_1 - 4x_2 = -4\theta - 4\dot{\theta}$, то система с разомкнутым контуром $(\mathbf{A} - \mathbf{B}\mathbf{K})$ будет иметь устойчивые собственные значения $\lambda = -1$ и $\lambda = -3$.

Определение того, когда возможно изменить собственные значения системы с замкнутым контуром и с помощью какого закона управления $\mathbf{K}$ этого достичь, будет предметом следующих разделов.

Круиз-контроль. Чтобы оценить способность управления с замкнутым контуром компенсировать немоделируемую динамику и возмущения, рассмотрим простую модель автомобильного круиз-контроля. Обозначим u скорость подачи топлива в двигатель, а y – скорость автомобиля. Пренебрегая переходными эффектами, мы можем составить грубую модель[1]:

$$y = u. \tag{8.5}$$

Таким образом, если удвоить подачу топлива, то и скорость автомобиля удвоится.

На основе этой модели мы можем спроектировать систему круиз-контроля с разомкнутым контуром, которая будет поддерживать желаемую скорость w_r, просто подавая топливо со скоростью $u = w_r$. Однако при этом не учитывается, что модель автомобиля неправильная (в действительности $y = 2u$), и игнорируются внешние возмущения, например холмистая местность (когда $y = u + \sin(t)$).

С другой стороны, закон управления с замкнутым контуром, основанный на измерениях скорости, может компенсировать немоделируемую динамику и возмущения. Рассмотрим закон управления $u = K(w_r - y)$; это означает, что

[1] Более реалистичная модель должна учитывать динамику разгона, т. е. $\dot{x} = -x + u$ и $y = x$.

подача топлива увеличивается, если измеренная скорость ниже заданной, и уменьшается, если выше. Тогда если на самом деле динамика описывается формулой $y = 2u$, а не $y = u$, то система с разомкнутым контуром будет давать 50%-ное стационарное отклонение от заданного режима, а работу системы с замкнутым контуром можно значительно улучшить при больших K:

$$y = 2K(w_r - y) \;\Rightarrow\; (1 + 2K)y = 2Kw_r \;\Rightarrow\; y = \frac{2K}{1 + 2K} w_r. \tag{8.6}$$

При $K = 50$ система с замкнутым контуром в стационарном режиме отклоняется от заданной скорости всего на 1 %. Аналогично аддитивное возмущение w_d будет ослабляться с коэффициентом $1/(2K + 1)$.

Для конкретики рассмотрим случай, когда задана скорость 60 км/ч. Модель имеет вид $y = u$, тогда как истинная система удовлетворяет уравнению $y = 0.5u$. Кроме того, имеет место возмущение – подъемы и спуски, которые уменьшают и увеличивают скорость на 10 км/ч с частотой 0.5 Гц. На рис. 8.5 и в листинге 8.1 сравниваются регулятор с разомкнутым контуром и пропорциональный регулятор с замкнутым контуром при $K = 50$. Хотя регулятор с замкнутым контуром работает гораздо лучше, впоследствии мы увидим, что с большим коэффициентом пропорциональности может быть связана потеря робастности. Качество улучшится, если добавить интегральный член.

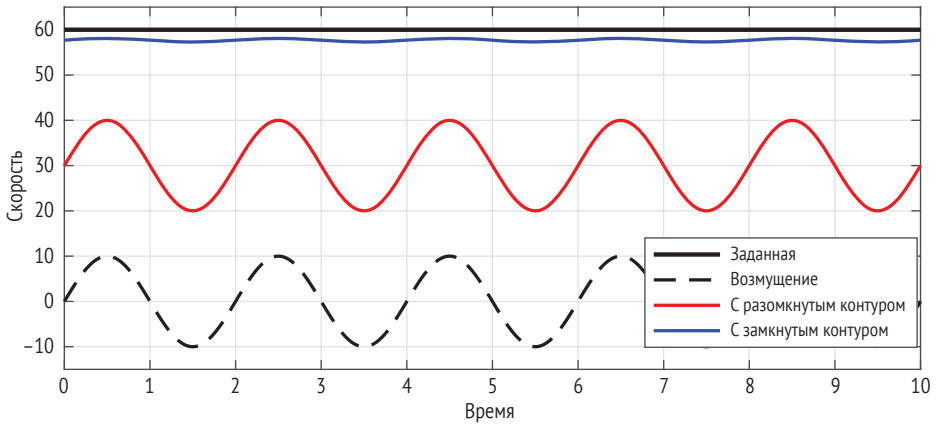

Рис. 8.5 ❖ Круиз-контроль с разомкнутым и с замкнутым контурами

Листинг 8.1 ❖ Сравнение круиз-контроля с разомкнутым и с замкнутым контурами

```
clear all, close all, clc

t = 0:.01:10;           % время

wr = 60*ones(size(t));  % заданная скорость
d = 10*sin(pi*t);       % возмущение

aModel = 1;             % y = aModel*u
```

```
aTrue = .5;              % y = aTrue*u

uOL = wr/aModel;         % u с разомкнутым контуром на основе модели
yOL = aTrue*uOL + d;     % отклик с разомкнутым контуром

K = 50;                  % коэффициент усиления, u = K(wr - y);
yCL = aTrue*K/(1+aTrue*K)*wr + d/(1+aTrue*K);
```

8.2. Линейные стационарные системы

Наиболее полно теория управления разработана для линейных систем [492, 165, 22]. В общем случае линейная система получается линеаризацией нелинейной системы в окрестности неподвижной точки или периодической орбиты. Однако из-за неустойчивости траектория может быстро отдалиться от неподвижной точки. К счастью, эффективный стабилизирующий регулятор сможет удержать состояние системы в малой окрестности неподвижной точки, где линейная аппроксимация имеет место. Например, в случае обратного маятника управление с обратной связью может удерживать маятник в вертикальном положении, где динамика почти линейна.

Линеаризация нелинейной динамики

Пусть дана нелинейная система

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{8.7a}$$

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}). \tag{8.7b}$$

Тогда мы можем линеаризовать динамику в окрестности неподвижной точки $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$, в которой $\mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) = \mathbf{0}$. Для небольших $\Delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$ и $\Delta\mathbf{u} = \mathbf{u} - \bar{\mathbf{u}}$ динамику $\mathbf{f}$ можно разложить в ряд Тейлора в окрестности точки $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$:

$$\mathbf{f}(\bar{\mathbf{x}} + \Delta\mathbf{x}, \bar{\mathbf{u}} + \Delta\mathbf{u}) = \mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \underbrace{\left.\frac{\mathbf{df}}{\mathbf{dx}}\right|_{(\bar{\mathbf{x}}, \bar{\mathbf{u}})}}_{\mathbf{A}} \cdot \Delta\mathbf{x} + \underbrace{\left.\frac{\mathbf{df}}{\mathbf{du}}\right|_{(\bar{\mathbf{x}}, \bar{\mathbf{u}})}}_{\mathbf{B}} \cdot \Delta\mathbf{u} + \cdots. \tag{8.8}$$

Аналогично можно разложить в ряд Тейлора выходной сигнал $\mathbf{g}$:

$$\mathbf{g}(\bar{\mathbf{x}} + \Delta\mathbf{x}, \bar{\mathbf{u}} + \Delta\mathbf{u}) = \mathbf{g}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \underbrace{\left.\frac{\mathbf{dg}}{\mathbf{dx}}\right|_{(\bar{\mathbf{x}}, \bar{\mathbf{u}})}}_{\mathbf{C}} \cdot \Delta\mathbf{x} + \underbrace{\left.\frac{\mathbf{dg}}{\mathbf{du}}\right|_{(\bar{\mathbf{x}}, \bar{\mathbf{u}})}}_{\mathbf{D}} \cdot \Delta\mathbf{u} + \cdots. \tag{8.9}$$

Для небольших отклонений от неподвижной точки члены высшего порядка пренебрежимо малы. Если опустить Δ и сдвинуть оси, так что $\bar{\mathbf{x}}$, $\bar{\mathbf{u}}$ и $\bar{\mathbf{y}}$ окажутся в начале координат, то линеаризованную динамику можно записать в виде:

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{8.10a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}. \tag{8.10b}$$

Заметим, что мы пренебрегли возмущением и шумом, $\mathbf{w}_d$ и $\mathbf{w}_n$; мы вернем их в рассмотрение, когда будем обсуждать фильтр Калмана в разделе 8.5.

Неуправляемая линейная система

В отсутствие управления (когда $\mathbf{u} = \mathbf{0}$) и при наличии измерений полного состояния (когда $\mathbf{y} = \mathbf{x}$) динамическая система (8.10) принимает вид:

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x}. \tag{8.11}$$

Ее решением $\mathbf{x}(t)$ является

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0), \tag{8.12}$$

где матричная экспонента определена следующим образом:

$$e^{\mathbf{A}t} = \mathbf{I} + \mathbf{A}t + \frac{\mathbf{A}^2t^2}{2} + \frac{\mathbf{A}^3t^3}{3} + \cdots. \tag{8.13}$$

Решение (8.12) полностью определено собственными значениями и собственными векторами матрицы $\mathbf{A}$. Рассмотрим спектральное разложение $\mathbf{A}$:

$$\mathbf{A}\mathbf{T} = \mathbf{T}\mathbf{\Lambda}. \tag{8.14}$$

В простейшем случае $\mathbf{\Lambda}$ – диагональная матрица, содержащая различные собственные значения, а столбцами матрицы $\mathbf{T}$ являются соответствующие им линейно независимые собственные векторы $\mathbf{A}$. Если собственные значения повторяются, то $\mathbf{\Lambda}$ можно записать в жордановой форме, в которой над диагональю имеются ненулевые значения для собственных значений кратности ≥ 2; соответствующие столбцы $\mathbf{T}$ называются обобщенными собственными векторами.

В любом случае матричную экспоненту $e^{\mathbf{\Lambda}t}$ вычислить проще, чем $e^{\mathbf{A}t}$. Если $\mathbf{\Lambda}$ – диагональная матрица, то матричная экспонента имеет вид:

$$e^{\mathbf{\Lambda}t} = \begin{bmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{\lambda_n t} \end{bmatrix}. \tag{8.15}$$

А для матриц $\mathbf{\Lambda}$ с нетривиальными жордановыми клетками существуют простые обобщения, связанные с нильпотентными матрицами (подробнее см. в работе Perko [427]). Реорганизуя члены в (8.14), находим, что степени $\mathbf{A}$ можно без труда выразить через собственные векторы и собственные значения:

$$\mathbf{A} = \mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1} \tag{8.16a}$$

$$\mathbf{A}^2 = (\mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1})(\mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1}) = \mathbf{T}\mathbf{\Lambda}^2\mathbf{T}^{-1} \tag{8.16b}$$

...

$$\mathbf{A}^k = (\mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1})(\mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1}) \cdots (\mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1}) = \mathbf{T}\mathbf{\Lambda}^k\mathbf{T}^{-1}. \tag{8.16c}$$

Наконец, подставляя эти выражения в (8.13), получаем:

$$e^{\mathbf{A}t} = e^{\mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1}} = \mathbf{T}\mathbf{T}^{-1} + \mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1}t + \frac{\mathbf{T}\mathbf{\Lambda}^2\mathbf{T}^{-1}t^2}{2} + \frac{\mathbf{T}\mathbf{\Lambda}^3\mathbf{T}^{-1}t^3}{3} + \cdots \tag{8.17a}$$

$$= \mathbf{T}\left[\mathbf{I} + \mathbf{\Lambda}t + \frac{\mathbf{\Lambda}^2t^2}{2} + \frac{\mathbf{\Lambda}^3t^3}{3} + \cdots\right]\mathbf{T}^{-1} \tag{8.17b}$$

$$= \mathbf{T}e^{\mathbf{\Lambda}t}\mathbf{T}^{-1}. \tag{8.17c}$$

Таким образом, мы видим, что матричную экспоненту можно эффективно вычислить, зная спектральное разложение **A**. Более того, матрица собственных векторов **T** определяет замену координат, которая существенно упрощает динамику:

$$\mathbf{x} = \mathbf{T}\mathbf{z} \;\Rightarrow\; \dot{\mathbf{z}} = \mathbf{T}^{-1}\dot{\mathbf{x}} = \mathbf{T}^{-1}\mathbf{A}\mathbf{x} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\mathbf{z} \;\Rightarrow\; \dot{\mathbf{z}} = \mathbf{\Lambda}\mathbf{z}. \tag{8.18}$$

Иными словами, при переходе к системе координат собственных векторов динамика становится диагональной. Объединяя (8.12) с (8.17c), можно записать решение $\mathbf{x}(t)$ в виде

$$\underbrace{\mathbf{T}\underbrace{e^{\mathbf{\Lambda}t}\underbrace{\mathbf{T}^{-1}\mathbf{x}(0)}_{\mathbf{z}(0)}}_{\mathbf{z}(t)}}_{\mathbf{x}(t)}. \tag{8.19}$$

На первом шаге $\mathbf{T}^{-1}$ отображает начальное условие, записанное в физических координатах, $\mathbf{x}(0)$, в систему координат собственных векторов, $\mathbf{z}(0)$. На следующем шаге эти начальные условия экстраполируются с помощью диагонального преобразования $e^{\mathbf{\Lambda}}$, которое выглядит значительно проще в координатах собственных векторов **z**. Наконец, умножение на **T** переводит $\mathbf{z}(t)$ обратно в физические координаты $\mathbf{x}(t)$.

Помимо возможности вычислить матричную экспоненту, а значит, и решение $\mathbf{x}(t)$, спектральное разложение **A** даже более полезно для понимания динамики и устойчивости системы. Из (8.19) мы видим, что единственная зависящая от времени часть решения – это $e^{\mathbf{\Lambda}t}$. В общем случае собственные значения $\lambda = a + ib$ могут быть комплексными числами, поэтому решения имеют вид $e^{\lambda t} = e^{at}(\cos(bt) + i\sin(bt))$. Следовательно, если у всех собственных значений λ_k вещественная часть отрицательна ($Re(\lambda) = a < 0$), то система устойчива, и все решения стремятся к $\mathbf{x} = \mathbf{0}$ при $t \to \infty$. Но если вещественная часть хотя бы одного собственного значения положительна, то система неустойчива и отклоняется от неподвижной точки вдоль соответствующего собственного вектора. С большой вероятностью у любого случайно выбранного начального значения имеется компонента

в таком неустойчивом направлении, а кроме того, возмущения, скорее всего, воздействуют на все собственные векторы системы.

Управляемая линейная система

При наличии управления и нулевом начальном условии $\mathbf{x}(0) = \mathbf{0}$ решение уравнения (8.10а) имеет вид:

$$\mathbf{x}(t) = \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau \triangleq e^{\mathbf{A}t}\mathbf{B} * \mathbf{u}(t). \tag{8.20}$$

Управляющий сигнал $\mathbf{u}(t)$ сворачивается с ядром $e^{\mathbf{A}t}\mathbf{B}$. Если выходной сигнал $\mathbf{y} = \mathbf{C}\mathbf{x}$, то имеем $\mathbf{y}(t) = \mathbf{C}e^{\mathbf{A}t}\mathbf{B} * \mathbf{u}(t)$. Эта свертка показана на рис. 8.6 для системы с одним входом и одним выходом (single-input, single-output – SISO) в терминах импульсной характеристики $g(t) = \mathbf{C}e^{\mathbf{A}t}\mathbf{B} = \int_0^t \mathbf{C}e^{\mathbf{A}(t-\tau)}\mathbf{B}\delta(\tau)d\tau$, когда $u(t) = \delta(t)$ – дельта-функция Дирака.

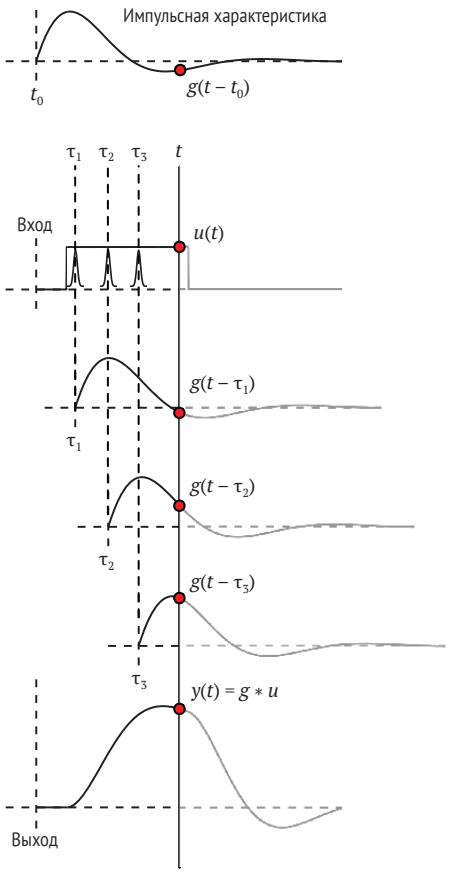

Рис. 8.6 ❖ Свертка для системы с одним входом и одним выходом

Системы с дискретным временем

Во многих практических приложениях системы дискретизируются по времени. Поэтому системы с цифровым управлением обычно описываются в терминах динамических систем с дискретным временем:

$$\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k, \tag{8.21a}$$

$$\mathbf{y}_k = \mathbf{C}_d\mathbf{x}_k + \mathbf{D}_d\mathbf{u}_k, \tag{8.21b}$$

где $\mathbf{x}_k = \mathbf{x}(k\Delta t)$. Матрицы системы (8.21) можно получить из системы с непрерывным временем (8.10) в виде

$$\mathbf{A}_d = e^{\mathbf{A}\Delta t}; \tag{8.22a}$$

$$\mathbf{B}_d = \int_0^{\Delta t} e^{\mathbf{A}\tau}\mathbf{B}d\tau; \tag{8.22b}$$

$$\mathbf{C}_d = \mathbf{C}; \tag{8.22c}$$

$$\mathbf{D}_d = \mathbf{D}. \tag{8.22d}$$

Устойчивость системы с дискретным временем (8.21) по-прежнему определяется собственными значениями матрицы $\mathbf{A}_d$, но теперь система устойчива в том и только в том случае, когда все дискретные собственные значения находятся внутри единичной окружности на комплексной плоскости. Таким образом, $\exp(\mathbf{A}\Delta t)$ определяет конформное отображение комплексной плоскости, описывающее переход от непрерывного времени к дискретному, при котором собственные значения в левой полуплоскости отображаются в собственные значения внутри единичной окружности.

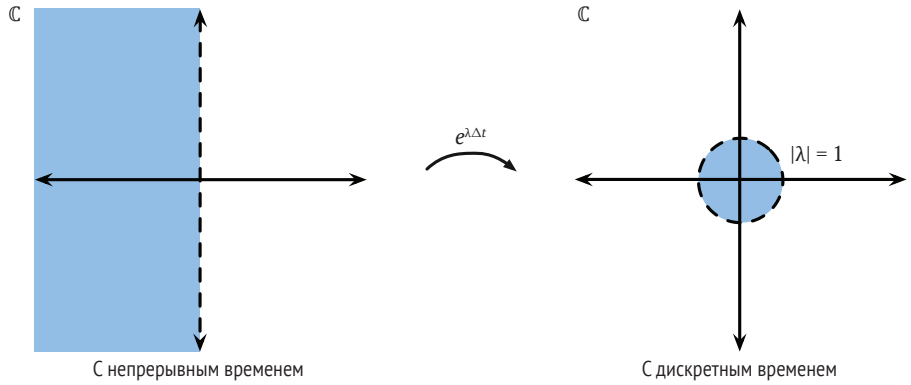

Рис. 8.7 ❖ Матричная экспонента определяет конформное отображение комплексной плоскости, переводящее собственные значения в левой полуплоскости в собственные значения внутри единичной окружности

Пример: обратный маятник

Рассмотрим обратный маятник на рис. 8.8, к нижней точке которого приложен крутящий момент u. Уравнение движения выводится из уравнений Эйлера–Лагранжа[1]:

$$\ddot{\theta} = -\frac{g}{L}\sin(\theta) + u. \tag{8.23}$$

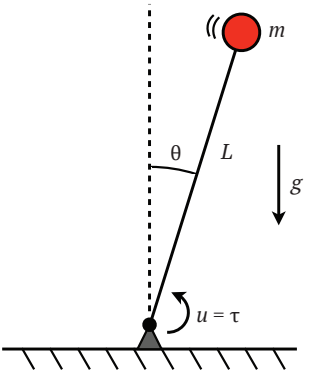

Рис. 8.8 ❖ Обратный маятник

Введя в рассмотрение состояние $\mathbf{x}$, описываемое угловым положением и скоростью, мы сможем записать это дифференциальное уравнение второго порядка в виде системы уравнений первого порядка:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \quad \Rightarrow \quad \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\dfrac{g}{L}\sin(x_1) + u \end{bmatrix}. \tag{8.24}$$

Вычислив якобиан $\mathbf{f}(\mathbf{x}, \mathbf{u})$, получаем

$$\frac{\mathbf{df}}{\mathbf{dx}} = \begin{bmatrix} 0 & 1 \\ -\dfrac{g}{L}\cos(x_1) & 0 \end{bmatrix}, \quad \frac{\mathbf{df}}{\mathbf{du}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{8.25}$$

Линеаризация в верхней ($x_1 = \pi$, $x_2 = 0$) и нижней ($x_1 = 0$, $x_2 = 0$) точках равновесия дает

---

[1] Лагранжиан равен $\mathcal{L} = \frac{m}{2}L^2\dot{\theta}^2 - mgL\cos(\theta)$, а уравнение Эйлера–Лагранжа имеет вид $\frac{d}{dt}\partial\mathcal{L}/\partial\dot{\theta} - \partial\mathcal{L}/\partial\theta = \tau$, где τ – заданный крутящий момент.

$$\underbrace{\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \dfrac{g}{L} & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u;}_{\text{Верхнее положение маятника } \lambda=\pm\sqrt{g/L}} \qquad \underbrace{\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\dfrac{g}{L} & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u.}_{\text{Нижнее положение маятника } \lambda=\pm i\sqrt{g/L}} \tag{8.25}$$

Итак, мы видим, что нижнее положение – устойчивый центр с собственными значениями $\lambda = \pm i\sqrt{g/L}$, соответствующий колебаниям с собственной частотой $\sqrt{g/L}$. Верхнее же положение – неустойчивая седловая точка с собственными значениями $\lambda = \pm\sqrt{g/L}$.

8.3. Управляемость и наблюдаемость

В теории линейного управления возникает естественный вопрос: до какой степени обратная связь $\mathbf{u} = -\mathbf{K}\mathbf{x}$ может влиять на поведение системы (8.10a)? В разделе 8.1 мы уже видели, что можно модифицировать собственные значения неустойчивой системы обратного маятника посредством замкнутого контура обратной связи и получить в результате новую матрицу системы $(\mathbf{A} - \mathbf{B}\mathbf{K})$ с устойчивыми собственными значениями.

В этом разделе мы приведем конкретные условия, показывающие, когда и как можно повлиять на динамику системы с помощью управления с обратной связью. Мы рассмотрим также двойственный вопрос: когда можно оценить полное состояние $\mathbf{x}$ по измерениям $\mathbf{y}$.

Управляемость

Возможность повлиять на собственные значения системы с разомкнутым контуром посредством выбора $\mathbf{K}$ зависит от *управляемости* системы (8.10a). Управляемость линейной системы полностью определяется пространством столбцов матрицы *управляемости* $\mathcal{C}$:

$$\mathcal{C} = [\mathbf{B} \;\; \mathbf{A}\mathbf{B} \;\; \mathbf{A}^2\mathbf{B} \;\; \cdots \;\; \mathbf{A}^{n-1}\mathbf{B}]. \tag{8.26}$$

Если в матрице $\mathcal{C}$ имеется n линейно независимых столбцов, линейной оболочкой которых, следовательно, является все пространство $\mathbb{R}^n$, то система (8.10a) управляема. Линейная оболочка столбцов матрицы управляемости $\mathcal{C}$ образует подпространство Крылова, которое определяет, какими направлениями вектора состояния в $\mathbb{R}^n$ можно манипулировать с помощью управления. Таким образом, управляемость влечет за собой не только возможность произвольного выбора собственных значений, но и тот факт, что любое состояние $\boldsymbol{\xi} \in \mathbb{R}^n$ достижимо за конечное время посредством некоторого управляющего воздействия $\mathbf{u}(t)$.

Следующие три условия эквивалентны:

1. *Управляемость*. Линейная оболочка $\mathcal{C}$ совпадает с $\mathbb{R}^n$. Матрицу $\mathcal{C}$ можно построить командой

   ```
   >> ctrb(A,B)
   ```

   а проверить, равен ли ее ранг n, – командой

   ```
   >> rank(ctrb(A,B))
   ```

2. *Произвольный выбор собственных значений*. С помощью подходящего выбора обратной связи $\mathbf{u} = -\mathbf{K}\mathbf{x}$ можно по собственному усмотрению задавать собственные значения системы с замкнутым контуром:

   $$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}. \tag{8.27}$$

   Если задано множество желаемых собственных значений, то матрицу усиления $\mathbf{K}$ можно построить командой

   ```
   >> K = place(A,B,neweigs);
   ```

   Проектирование $\mathbf{K}$ с целью достижения наилучшего качества обсуждается в разделе 8.4.

3. *Достижимость* $\mathbb{R}^n$. Систему можно привести в любое состояние $\mathbf{x}(t) = \boldsymbol{\xi} \in \mathbb{R}^n$ за конечное время с помощью подходящего управляющего воздействия $\mathbf{u}(t)$.

Заметим, что достижимость относится и к системам с разомкнутым контуром. В частности, если направление $\boldsymbol{\xi}$ не входит в линейную оболочку $\mathcal{C}$, то посредством управления невозможно понудить систему к движению в данном направлении ни в системе с замкнутым контуром, ни в системе с разомкнутым контуром.

Примеры. Понятие управляемости проще освоить, рассмотрев несколько простых примеров. Сначала рассмотрим систему

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \quad \Rightarrow \quad \mathcal{C} = \begin{bmatrix} 0 & 0 \\ 1 & 2 \end{bmatrix}. \tag{8.28}$$

Эта система неуправляемая, потому что матрица управляемости $\mathcal{C}$ состоит из двух линейно зависимых векторов, так что ее линейная оболочка не совпадает $\mathbb{R}^2$. Даже до проверки ранга матрицы управляемости видно, что система не может быть управляемой, потому что состояния x_1 и x_2 полностью развязаны и исполнительная команда u влияет только на второе состояние.

Если модифицировать этот пример, включив два управляющих сигнала, то система станет управляемой:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \Rightarrow \quad \mathcal{C} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}. \tag{8.29}$$

Эта *полноприводная* система, очевидно, управляема, потому что состояниями x_1 и x_2 можно независимо управлять с помощью сигналов u_1 и u_2. Управляемость подтверждается тем, что линейная оболочка столбцов $\mathcal{C}$ действительно совпадает с $\mathbb{R}^2$.

Наиболее интересные случаи не столь очевидны, как эти два примера. Рассмотрим систему

$$\frac{d}{dt}\begin{bmatrix}x_1\\x_2\end{bmatrix}=\begin{bmatrix}1&1\\0&2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}u \quad \Rightarrow \quad \mathcal{C}=\begin{bmatrix}0&1\\1&2\end{bmatrix}. \tag{8.30}$$

Эта система с двумя состояниями управляема с помощью одной исполнительной команды, потому что состояния x_1 и x_2 теперь динамически связаны. Аналогично система

$$\frac{d}{dt}\begin{bmatrix}x_1\\x_2\end{bmatrix}=\begin{bmatrix}1&0\\0&2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}+\begin{bmatrix}1\\1\end{bmatrix}u \quad \Rightarrow \quad \mathcal{C}=\begin{bmatrix}1&1\\1&2\end{bmatrix} \tag{8.31}$$

управляема, несмотря на то что динамика x_1 и x_2 развязана, т. к. исполнительная команда $\mathbf{B}=[1\ \ 1]^T$ может одновременно воздействовать на оба состояния, и временные шкалы у них разные.

В разделе 8.3 мы увидим, что управляемость тесно связана с совпадением направлений столбцов **B** и собственных векторов **A**.

Наблюдаемость

С точки зрения математики, наблюдаемость системы (8.10) – почти то же самое, что управляемость, хотя физический смысл у этих понятий разный. Система называется *наблюдаемой*, если любое состояние $\boldsymbol{\xi} \in \mathbb{R}^n$ можно оценить на основе истории измерений $\mathbf{y}(t)$.

Наблюдаемость системы полностью определяется пространством строк матрицы *наблюдаемости* $\mathcal{O}$:

$$\mathcal{O}=\begin{bmatrix}\mathbf{C}\\\mathbf{CA}\\\mathbf{CA}^2\\\vdots\\\mathbf{CA}^{n-1}\end{bmatrix}. \tag{8.32}$$

В частности, если линейная оболочка строк матрицы $\mathcal{O}$ совпадает с $\mathbb{R}^n$, то любое полноразмерное состояние $\mathbf{x} \in \mathbb{R}^n$ можно оценить по истории $\mathbf{y}(t)$. Матрица $\mathcal{O}$ строится командой

```
>> obsv(A,C);
```

Стремление получить оценку всего состояния легко объяснимо. Мы уже видели, что если обратная связь воздействует на все состояние, $\mathbf{u}=-\mathbf{Kx}$, то

можно модифицировать поведение управляемой системы. Но если измерения полного состояния $\mathbf{x}$ недоступны, то необходимо *оценить* $\mathbf{x}$ по измерениям. Это возможно, когда система наблюдаемая. В разделе 8.5 мы увидим, что можно спроектировать динамическую систему-наблюдатель, которая будет оценивать полное состояние по зашумленным измерениям. Как и в случае управляемой системы, если система наблюдаема, то можно подобрать собственные значения динамической системы-наблюдателя с желаемыми характеристиками, например с целью быстрого оценивания или эффективного ослабления шума.

Интересно, что критерий наблюдаемости математически является двойственным критерию управляемости. В самом деле, матрица наблюдаемости – результат транспонирования матрицы управляемости для пары ($\mathbf{A}^T$, $\mathbf{C}^T$):

```
>> O = ctrb(A',C')'; % 'obsv' двойственна 'crtb'
```

Критерий управляемости PBH

Существует много критериев управляемости системы. Один из самых полезных и информативных – критерий Попова–Белевича–Хаутуса (PBH). Этот критерий утверждает, что пара ($\mathbf{A}$, $\mathbf{B}$) управляема тогда и только тогда, когда столбцовый ранг матрицы $[(\mathbf{A} - \lambda\mathbf{I}) \quad \mathbf{B}]$ равен n для всех $\lambda \in \mathbb{C}$. Этот тест особенно интересен тем, что устанавливает связь между управляемостью[1] и соотношением между столбцами $\mathbf{B}$ и собственным пространством $\mathbf{A}$.

Отметим для начала, что критерий PBH нужно проверять только для λ, являющихся собственными значениями $\mathbf{A}$, поскольку ранг $\mathbf{A} - \lambda\mathbf{I}$ равен n при всех λ, кроме собственных значений $\mathbf{A}$. На самом деле для нахождения собственных значений $\mathbf{A}$ используется характеристическое уравнение $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$, поскольку это именно те значения, для которых матрица $\mathbf{A} - \lambda\mathbf{I}$ имеет неполный ранг, иначе говоря, является вырожденной.

Далее, из того что $(\mathbf{A} - \lambda\mathbf{I})$ имеет неполный ранг только для собственных значений λ, следует также, что ядро, или нуль-пространство $\mathbf{A} - \lambda\mathbf{I}$, натянуто на собственные векторы, соответствующие данному собственному значению. Таким образом, чтобы $[(\mathbf{A} - \lambda\mathbf{I}) \quad \mathbf{B}]$ имела ранг n, столбцы $\mathbf{B}$ должны иметь ненулевую компоненту в направлении каждого из собственных векторов $\mathbf{A}$, дабы дополнить ядро $\mathbf{A} - \lambda\mathbf{I}$.

Если $\mathbf{A}$ имеет n различных собственных значений, то система будет управляемой одной исполнительной командой, потому что в ядре матрицы $\mathbf{A} - \lambda\mathbf{I}$ будет не более одного направления собственного вектора. В частности, мы можем взять в качестве $\mathbf{B}$ сумму всех n линейно независимых собственных векторов, которая гарантированно будет иметь ненулевую компоненту

[1] Существует эквивалентный PBH критерий наблюдаемости, согласно которому матрица $\begin{bmatrix}(\mathbf{A} - \lambda\mathbf{I}) \\ \mathbf{C}\end{bmatrix}$ должна иметь ранг n для всех $\lambda \in \mathbb{C}$.

в каждом направлении. Интересно также отметить, что если **B** – случайный вектор (>>**B=randn(n,1);**), то (**A**, **B**) с высокой вероятностью будет управляема, т. к. крайне маловероятно, что вклад хотя бы одного собственного вектора в **B** будет нулевым.

Если существуют вырожденные собственные значения кратности ≥ 2, т. е. размерность ядра $\mathbf{A} - \lambda\mathbf{I}$ больше 1, то исполнительная команда должна иметь столько же степеней свободы. Иными словами, несколько управляющих сигналов (столбцов **B**) строго необходимо, только когда у системы вырожденные собственные значения. Однако если система совсем уж аномальна, то на практике полезно иметь несколько управляющих сигналов для повышения авторитетности управления. Для таких аномальных систем характерно наличие значительного транзиентного роста вследствие ослабляющей интерференции между почти параллельными собственными векторами, зачастую с близкими собственными значениями.

Теорема Кэли–Гамильтона и достижимость

Чтобы пролить свет на связь между управляемостью пары (**A**, **B**) и достижимостью любого вектора $\boldsymbol{\xi} \in \mathbb{R}^n$ с помощью управляющего сигнала $\mathbf{u}(t)$, мы воспользуемся теоремой Кэли–Гамильтона. Эта жемчужина линейной алгебры дает элегантный способ представить решения уравнения $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ в терминах конечной суммы степеней **A**, а не бесконечной суммы в матричной экспоненте (8.13).

Теорема Кэли–Гамильтона утверждает, что любая матрица **A** удовлетворяет своему характеристическому уравнению $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = \lambda^n + a_{n-1}\lambda^{n-1} + \cdots + a_2\lambda^2 + a_1\lambda + a_0 = 0 \tag{8.33a}$$

$$\Rightarrow \mathbf{A}^n + a_{n-1}\mathbf{A}^{n-1} + \cdots + a_2\mathbf{A}^2 + a_1\mathbf{A} + a_0\mathbf{I} = \mathbf{0}. \tag{8.33b}$$

Несмотря на простоту формулировки, эта теорема имеет важные следствия. В частности, она дает возможность выразить $\mathbf{A}^n$ в виде линейной комбинации меньших степеней **A**:

$$\mathbf{A}^n = -a_0\mathbf{I} - a_1\mathbf{A} - a_2\mathbf{A}^2 - \cdots - a_{n-1}\mathbf{A}^{n-1}. \tag{8.34}$$

Как легко видеть, отсюда также следует, что любую более высокую степень $\mathbf{A}^{k\geq n}$ можно выразить в виде линейной комбинации матриц $\{\mathbf{I}, \mathbf{A}, \cdots, \mathbf{A}^{n-1}\}$:

$$\mathbf{A}^{k\geq n} = \sum_{j=0}^{n-1} \alpha_j \mathbf{A}^j. \tag{8.35}$$

Следовательно, бесконечную сумму в экспоненте $e^{\mathbf{A}t}$ можно выразить как

$$e^{\mathbf{A}t} = \mathbf{I} + \mathbf{A}t + \frac{\mathbf{A}^2t^2}{2} + \cdots \tag{8.36a}$$

$$= \beta_0(t)\mathbf{I} + \beta_1(t)\mathbf{A} + \beta_2(t)\mathbf{A}^2 + \cdots + \beta_{n-1}(t)\mathbf{A}^{n-1}. \tag{8.36b}$$

Теперь у нас есть все, чтобы понять, как управляемость связана с достижимостью произвольного вектора $\boldsymbol{\xi} \in \mathbb{R}^n$. Из (8.20) видно, что состояние $\boldsymbol{\xi}$ достижимо, если существует такое $\mathbf{u}(t)$, что:

$$\beta = \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau. \tag{8.37}$$

Выражая экспоненту в правой части в виде (8.36b), получаем:

$$\begin{aligned}
\boldsymbol{\xi} &= \int_0^t [\beta_0(t-\tau)\mathbf{I}\mathbf{B}\mathbf{u}(\tau) + \beta_1(t-\tau)\mathbf{A}\mathbf{B}\mathbf{u}(\tau) + \cdots \\
&\quad \cdots + \beta_{n-1}(t-\tau)\mathbf{A}^{n-1}\mathbf{B}\mathbf{u}(\tau)]d\tau \\
&= \mathbf{B}\int_0^t \beta_0(t-\tau)\mathbf{u}(\tau)d\tau + \mathbf{A}\mathbf{B}\int_0^t \beta_1(t-\tau)\mathbf{u}(\tau)d\tau + \cdots \\
&\quad \cdots + \mathbf{A}^{n-1}\mathbf{B}\int_0^t \beta_{n-1}(t-\tau)\mathbf{u}(\tau)d\tau \\
&= \begin{bmatrix}\mathbf{B} & \mathbf{A}\mathbf{B} & \cdots & \mathbf{A}^{n-1}\mathbf{B}\end{bmatrix}\begin{bmatrix}\int_0^t \beta_0(t-\tau)\mathbf{u}(\tau)d\tau \\ \int_0^t \beta_1(t-\tau)\mathbf{u}(\tau)d\tau \\ \vdots \\ \int_0^t \beta_{n-1}(t-\tau)\mathbf{u}(\tau)d\tau\end{bmatrix}.
\end{aligned}$$

Заметим, что матрица в левой части – это матрица управляемости $\mathcal{C}$, и мы видим, что для того чтобы все пространство $\mathbb{R}^n$ было достижимо, необходимо, чтобы $\mathbb{R}^n$ было натянуто на пространство столбцов $\mathcal{C}$. Несколько труднее доказать, что если ранг $\mathcal{C}$ равен n, то можно спроектировать $\mathbf{u}(t)$, так чтобы было достижимо произвольное состояние $\boldsymbol{\xi} \in \mathbb{R}^n$; доказательство опирается на тот факт, что n функций $\{\beta_j(t)\}_{j=0}^{n-1}$ линейно независимы. Важно также, что управляющий сигнал $\mathbf{u}(t)$, переводящий систему в заданное состояние $\boldsymbol{\xi}$, не *единственный*, существует много путей перехода.

Грамианы и степень управляемости и наблюдаемости

Предыдущие критерии управляемости и наблюдаемости были двухальтернативными: ранг $\boldsymbol{\mathcal{C}}$ (соответственно $\boldsymbol{\mathcal{O}}$) либо равен n, либо нет. Однако существует понятие *степени* управляемости и наблюдаемости, поскольку управление или оценка для одних состояний может быть проще, чем для других.

Чтобы узнать, какие состояния более или менее управляемы, следует проанализировать спектральное разложение грамиана управляемости:

$$\mathbf{W}_c(t) = \int_0^t e^{\mathbf{A}\tau}\mathbf{B}\mathbf{B}^* e^{\mathbf{A}^*\tau}d\tau. \tag{8.38}$$

Аналогично грамиан наблюдаемости определяется как

$$\mathbf{W}_o(t) = \int_0^t e^{\mathbf{A}^*\tau}\mathbf{C}^*\mathbf{C}e^{\mathbf{A}\tau}d\tau. \tag{8.39}$$

Часто эти грамианы вычисляются на бесконечном промежутке времени, и если не оговорено противное, будем обозначать $\mathbf{W}_c = \lim_{t\to\infty}\mathbf{W}_c(t)$ и $\mathbf{W}_o = \lim_{t\to\infty}\mathbf{W}_o(t)$.

Управляемость состояния $\mathbf{x}$ измеряется величиной $\mathbf{x}^*\mathbf{W}_c\mathbf{x}$, которая тем больше, чем более управляемым является состояние. Если значение $\mathbf{x}^*\mathbf{W}_c\mathbf{x}$ велико, то систему можно далеко продвинуть в направлении $\mathbf{x}$ с помощью единичного управляющего воздействия. Наблюдаемость состояния измеряется величиной $\mathbf{x}^*\mathbf{W}_o\mathbf{x}$. Оба грамиана – симметричные и положительно полуопределенные матрицы с неотрицательными собственными значениями. Поэтому собственные значения и собственные векторы можно упорядочить, так что собственные векторы, соответствующие большим собственным значениям, проще для управления или наблюдения. Таким образом, грамианы индуцируют новое скалярное произведение в пространстве состояний, выраженное в терминах их управляемости или наблюдаемости.

Грамианы можно наглядно представить эллипсоидами в пространстве состояний, главные оси которых направлены вдоль собственных векторов, упорядоченных в смысле управляемости или наблюдаемости. Пример такой визуализации приведен на рис. 9.2 в главе 9. На самом деле грамианы можно использовать для проектирования моделей пониженного порядка для систем высокой размерности. С помощью уравновешивающего преобразования идентифицируется подпространство, соответствующее модам, допускающим лучшее совместное управление или наблюдение. Эти моды определяют хороший базис проекции для определения модели, отражающей доминирующую динамику входа-выхода. Эту форму сбалансированного упрощения модели мы рассмотрим подробнее в разделе 9.2.

Грамианы также полезны для определения оптимального по энергозатратам управления $\mathbf{u}(t)$, необходимого для перевода системы в состояние $\mathbf{x}(t_f)$ в момент t_f из состояния $\mathbf{x}(0) = \mathbf{0}$:

$$\mathbf{u}(t) = \mathbf{B}^*\left(e^{\mathbf{A}(t_f-t)}\right)^*\mathbf{W}_c(t_f)^{-1}\mathbf{x}(t_f). \tag{8.40}$$

Полная энергия, расходуемая этим законом управления, равна

$$\int_0^{t_f}||\mathbf{u}(\tau)||^2 d\tau = \mathbf{x}^*\mathbf{W}_c(t_f)^{-1}\mathbf{x}. \tag{8.41}$$

Теперь видно, что если матрица управляемости почти сингулярна, то существуют направления, требующие экстремальной энергии управляющего сигнала. Обратно, если все собственные значения $\mathbf{W}_c$ велики, то система легко управляется.

В общем случае непрактично вычислять грамианы непосредственно по формулам (8.38) и (8.39). Вместо этого грамиан управляемости ищется как решение следующего уравнения Ляпунова:

$$\mathbf{A}\mathbf{W}_c + \mathbf{W}_c\mathbf{A}^* + \mathbf{B}\mathbf{B}^* = \mathbf{0}, \tag{8.42}$$

а грамиан наблюдаемости – как решение уравнения

$$\mathbf{A}^*\mathbf{W}_o + \mathbf{W}_o\mathbf{A} + \mathbf{C}^*\mathbf{C} = \mathbf{0}. \tag{8.43}$$

Нахождение грамианов в виде решений уравнения Ляпунова для систем высокой размерности, как правило, обходится очень дорого [213, 231, 496, 489, 55]. Поэтому грамианы часто аппроксимируют эмпирически, используя моментальные снимки прямой и сопряженной систем; мы обсудим это в разделе 9.2.

Стабилизируемость и распознаваемость

На практике в системах высокой размерности было бы слишком оптимистично ожидать управляемости и наблюдаемости всего состояния. Например, в многомерной гидросистеме вряд ли стоит надеяться, что удастся управлять всеми мельчайшими вихрями, вместо этого часто бывает достаточно авторитетного управления большими когерентными структурами, содержащими значительную часть энергии.

Под стабилизируемостью понимается способность управлять всеми направлениями неустойчивых собственных векторов $\mathbf{A}$, так чтобы они оставались в линейной оболочке $\mathcal{C}$. На практике это определение можно ослабить, включив слабо затухающие моды, соответствующие собственным значениям с небольшой отрицательной вещественной частью. Аналогично, если все неустойчивые собственные векторы $\mathbf{A}$ принадлежат линейной оболочке $\mathcal{O}^*$, то система называется распознаваемой.

В описании модели могут также присутствовать состояния, не нужные для управления. В качестве примера рассмотрим систему управления коммерческого пассажирского самолета. Состояние системы может включать положения пассажирских кресел, хотя они, очевидно, не управляются пилотом, да и не должны управляться.

8.4. Оптимальное управление полным состоянием: линейно-квадратичный регулятор (ЛКР)

В предыдущих разделах мы видели, что если пара ($\mathbf{A}$, $\mathbf{B}$) управляема, то можно произвольно манипулировать собственными значениями системы ($\mathbf{A} - \mathbf{B}\mathbf{K}$) с замкнутым контуром посредством выбора закона управления полным состоянием с обратной связью $\mathbf{u} = -\mathbf{K}\mathbf{x}$. Здесь неявно предполагается, что доступны измерения всего состояния (т. е. $\mathbf{C} = \mathbf{I}$ и $\mathbf{D} = \mathbf{0}$, так что $\mathbf{y} = \mathbf{x}$). Хотя измерения всего состояния доступны не всегда, особенно в системах высокой размерности, в следующем разделе мы покажем, что если система наблюдаемая, то можно дать оценку всего состояния на основе показаний датчиков.

Если дана управляемая система, для которой либо имеются измерения полного состояния, либо она наблюдаемая и имеется оценка полного состояния, то существует много способов стабилизировать закон управления $\mathbf{u} = -\mathbf{K}\mathbf{x}$. Можно сделать собственные значения системы с замкнутым контуром ($\mathbf{A} - \mathbf{B}\mathbf{K}$) сколь угодно устойчивыми, разместив их настолько далеко в левой комплексной полуплоскости, насколько это необходимо. Однако если собственные значения излишне устойчивы, то затраты на управление могут оказаться непомерно велики, а уровень управляющих сигналов может превысить допустимую величину. Кроме того, система управления с чересчур устойчивыми собственными значениями может чрезмерно сильно реагировать на шум и возмущения точно так же, как начинающий водитель слишком активно откликается на вибрации на руле, что вызывает дрожания в системе с замкнутым контуром. В противоречие с интуицией чрезмерная стабилизация уменьшает робастность и может привести к неустойчивости при наличии малых временных задержек или немоделируемой динамики. Робастность мы будем обсуждать в разделе 8.8.

Выбор наилучшей матрицы усиления $\mathbf{K}$ для стабилизации системы без чрезмерных затрат на управление – важная цель *оптимального* управления. Необходимо найти баланс между устойчивостью системы с замкнутым контуром и агрессивностью управления. Очень важно принимать во внимание затраты, чтобы: 1) предотвратить избыточную реакцию регулятора на высокочастотный шум и возмущения, 2) амплитуда управляющего сигнала не превышала максимально допустимой и 3) управление не оказалось непомерно дорогим. Конкретно, функция стоимости

$$J(t) = \int_0^t \mathbf{x}(\tau)^* \mathbf{Q}\mathbf{x}(\tau) + \mathbf{u}(\tau)^* \mathbf{R}\mathbf{u}(\tau) d\tau \tag{8.44}$$

предлагает компромисс между эффективным регулированием состояния и стоимостью управления. Матрицы $\mathbf{Q}$ и $\mathbf{R}$ определяют соответственно веса отклонения состояния от нуля и стоимости приведения в действие исполнительных механизмов. Матрица $\mathbf{Q}$ положительно полуопределенная, а $\mathbf{R}$ положительно определенная; часто эти матрицы диагональные, а элементы на диагонали подбираются в соответствии с относительной важностью целей управления.

Наличие такой функции стоимости делает выбор закона управления корректно поставленной задачей оптимизации, для решения которой существует достаточно теоретических и численных методов [74]. Закон управления под названием линейно-квадратичный регулятор (ЛКР) $\mathbf{u} = -\mathbf{K}_r\mathbf{x}$ призван минимизировать величину $J = \lim_{t\to\infty} J(t)$. ЛКР назван так, потому что это линейный закон управления, предназначенный для линейной системы, но при этом минимизирует квадратичную функцию стоимости, так чтобы состояние системы стремилось к нулю: $\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{0}$. Поскольку функция стоимости (8.44) квадратичная, оптимальную матрицу усиления $\mathbf{K}_r$ можно найти аналитически:

$$\mathbf{K}_r = \mathbf{R}^{-1}\mathbf{B}^*\mathbf{X}, \tag{8.45}$$

где $\mathbf{X}$ – решение алгебраического уравнения Риккати:

$$\mathbf{A}^*\mathbf{X} + \mathbf{X}\mathbf{A} - \mathbf{X}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^*\mathbf{X} + \mathbf{Q} = \mathbf{0}. \tag{8.46}$$

Алгоритм решения уравнения Риккати относительно $\mathbf{X}$ и, стало быть, нахождения $\mathbf{K}_r$ численно устойчив и реализован на многих языках программирования [323, 55]. В MATLAB для вычисления $\mathbf{K}_r$ нужно выполнить команду

```
>> Kr = lqr(A,B,Q,R);
```

Однако вычислительная сложность этого алгоритма составляет $O(n^3)$, где n – размерность состояния, что делает его непригодным для больших систем или онлайновых вычислений при медленно изменяющемся состоянии системы или для линейного варьирующегося по параметру (linear parameter varying – LPV) управления. Поэтому возникает необходимость в разработке моделей пониженного порядка, которые улавливают то же доминирующее поведение, но имеют гораздо меньше состояний. Этой темой мы займемся в главе 9.

Схема ЛКР-регулятора показана на рис. 8.9. Из всех возможных законов управления $\mathbf{u} = \mathbf{K}(\mathbf{x})$, в т. ч. с нелинейными регуляторами, ЛКР-регулятор $\mathbf{u} = -\mathbf{K}_r\mathbf{x}$ оптимален, как будет показано в разделе 8.4. Однако может оказаться, что линеаризованная система не допускает линейного управления, тогда как полная нелинейная система (8.7) управляема с помощью нелинейного закона управления $\mathbf{u} = \mathbf{K}(\mathbf{x})$.

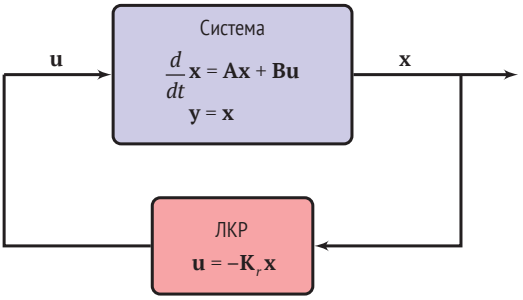

Рис. 8.9 ❖ Схема линейно-квадратичного регулятора (ЛКР) для оптимального управления с обратной связью при наличии полного состояния. Оптимальный регулятор линейной системы, для которой имеются измерения полного состояния $\mathbf{y} = \mathbf{x}$, описывается пропорциональным регулированием $\mathbf{u} = -\mathbf{K}_r\mathbf{x}$, где $\mathbf{K}_r$ – постоянная матрица усиления, полученная в результате решения алгебраического уравнения Риккати

Вывод уравнения Риккати оптимального управления

Сделаем теоретическое отступление и выведем уравнение Риккати (8.46) для задачи оптимального регулирования с полным состоянием. Это станет примером того, как задачи выпуклой оптимизации решаются методами вариационного исчисления, а также послужит образцом для нахождения

оптимального управления в случае *нелинейных* систем. В силу сходства задачи оптимального управления с вариационными принципами классической механики, а конкретно с лагранжианом и гамильтонианом, мы будем применять схожую терминологию и обозначения.

Прежде всего прибавим терминальную стоимость к нашей функции стоимости ЛКР (8.44) и для упрощения вычисления включим множитель 1/2:

$$J = \int_0^{t_f} \underbrace{\frac{1}{2}(\mathbf{x}^*\mathbf{Q}\mathbf{x} + \mathbf{u}^*\mathbf{R}\mathbf{u})}_{\text{Лагранжиан, } \mathcal{L}} d\tau + \underbrace{\frac{1}{2}(\mathbf{x}(t_f)^*\mathbf{Q}_f\mathbf{x}(t_f)}_{\text{Терминальная стоимость}}. \tag{8.47}$$

Наша цель – минимизировать квадратичную функцию стоимости J при наличии динамического ограничения:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}. \tag{8.48}$$

Мы можем решить эту задачу с помощью вариационного исчисления, введя в рассмотрение следующую расширенную функцию стоимости:

$$J_{\text{aug}} = \int_0^{t_f} \left[\frac{1}{2}(\mathbf{x}^*\mathbf{Q}\mathbf{x} + \mathbf{u}^*\mathbf{R}\mathbf{u}) + \boldsymbol{\lambda}^*(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} - \dot{\mathbf{x}})\right] d\tau + \frac{1}{2}(\mathbf{x}(t_f)^*\mathbf{Q}_f\mathbf{x}(t_f). \tag{8.49}$$

Переменная $\boldsymbol{\lambda}$ – это множитель Лагранжа, называемый *сопряженным состоянием*, он необходим для гарантии соблюдения динамических ограничений. $\boldsymbol{\lambda}$ может принимать любое значение, и будет иметь место равенство $J_{\text{aug}} = J$.

Возьмем полную вариацию J_{aug}:

$$\delta J_{\text{aug}} = \int_0^{t_f} \left[\frac{\partial L}{\partial \mathbf{x}}\delta\mathbf{x} + \frac{\partial L}{\partial \mathbf{u}}\delta\mathbf{u} + \boldsymbol{\lambda}^*\mathbf{A}\delta\mathbf{x} + \boldsymbol{\lambda}^*\mathbf{B}\delta\mathbf{u} - \boldsymbol{\lambda}^*\delta\dot{\mathbf{x}})\right] d\tau + \mathbf{Q}_f\mathbf{x}(t_f)\delta\mathbf{x}(t_f). \tag{8.50}$$

Частные производные[1] лагранжиана равны $\partial L/\partial\mathbf{x} = \mathbf{x}^*\mathbf{Q}$ и $\partial L/\partial\mathbf{u} = \mathbf{u}^*\mathbf{R}$. Последний член подынтегрального выражения можно модифицировать, воспользовавшись интегрированием по частям:

$$-\int_0^{t_f} \boldsymbol{\lambda}^*\delta\dot{\mathbf{x}}d\tau = -\boldsymbol{\lambda}^*(t_f)\delta\mathbf{x}(t_f) + \boldsymbol{\lambda}^*(0)\delta\mathbf{x}(0) + \int_0^{t_f} \dot{\boldsymbol{\lambda}}^*\delta\mathbf{x}d\tau.$$

Член $\boldsymbol{\lambda}^*(0)\delta\mathbf{x}(0)$ равен нулю, иначе система управления была бы непричинной (т. е. будущие управляющие действия могли бы изменить начальное состояние системы).

Наконец, полная вариация расширенной функции стоимости (8.50) упрощается следующим образом:

---

[1] Производная матричного выражения $\mathbf{A}\mathbf{x}$ по $\mathbf{x}$ равна $\mathbf{A}$, а производная $\mathbf{x}^*\mathbf{A}$ по $\mathbf{x}$ равна $\mathbf{A}^*$.

$$\delta J_{\text{aug}} = \int_0^{t_f} (\mathbf{x}^*\mathbf{Q} + \boldsymbol{\lambda}^*\mathbf{A} + \dot{\boldsymbol{\lambda}}^*)\delta\mathbf{x}d\tau + \int_0^{t_f} (\mathbf{u}^*\mathbf{R} + \boldsymbol{\lambda}^*\mathbf{B})\delta\mathbf{u}d\tau + \left(\mathbf{x}(t_f)^*\mathbf{Q}_f - \boldsymbol{\lambda}^*(t_f)\right)\delta\mathbf{x}(t_f). \tag{8.51}$$

Для оптимального управления, минимизирующего J, все члены вариации в формуле (8.51) должны быть равны нулю. Таким образом, мы получаем три уравнения:

$$\mathbf{X}^*\mathbf{Q} + \boldsymbol{\lambda}^*\mathbf{A} + \dot{\boldsymbol{\lambda}}^* = \mathbf{0}, \tag{8.52a}$$

$$\mathbf{u}^*\mathbf{R} + \boldsymbol{\lambda}^*\mathbf{B} = \mathbf{0}, \tag{8.52b}$$

$$\mathbf{x}(t_f)^*\mathbf{Q}_f - \boldsymbol{\lambda}^*(t_f) = \mathbf{0}. \tag{8.52c}$$

Заметим, что ограничение (8.52c) представляет начальное условие в момент t_fдля уравнения с обратным течением времени относительно $\boldsymbol{\lambda}$. Таким образом, уравнение (8.48) с начальным условием $\mathbf{x}(0) = \mathbf{x}_0$ и уравнение (8.52) с условием в конечный момент времени $\boldsymbol{\lambda}(t_f) = \mathbf{Q}_f\mathbf{x}(t_f)$ образуют двухточечную краевую задачу. Ее можно проинтегрировать численно и найти решение, описывающее оптимальное управление, даже для нелинейной системы.

Поскольку динамика линейная, мы можем *постулировать* форму $\boldsymbol{\lambda} = \mathbf{P}\mathbf{x}$ и подставить это в уравнения (8.52). Первое уравнение принимает вид:

$$(\dot{\mathbf{P}}\mathbf{x} + \mathbf{P}\dot{\mathbf{x}})^* + \mathbf{x}^*\mathbf{Q} + \boldsymbol{\lambda}^*\mathbf{A} = \mathbf{0}.$$

Транспонируя и подставляя (8.48) вместо $\dot{\mathbf{x}}$, получаем:

$$\dot{\mathbf{P}}\mathbf{x} + \mathbf{P}(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}) + \mathbf{Q}\mathbf{x} + \mathbf{A}^*\mathbf{P}\mathbf{x} = \mathbf{0}.$$

Из (8.52b) имеем:

$$\mathbf{u} = -\mathbf{R}^{-1}\mathbf{B}^*\boldsymbol{\lambda} = -\mathbf{R}^{-1}\mathbf{B}^*\mathbf{P}\mathbf{x}.$$

Наконец, объединяя оба равенства, получаем:

$$\dot{\mathbf{P}}\mathbf{x} + \mathbf{P}\mathbf{A}\mathbf{x} + \mathbf{A}^*\mathbf{P}\mathbf{x} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^*\mathbf{P}\mathbf{x} + \mathbf{Q}\mathbf{x} = \mathbf{0}. \tag{8.53}$$

Это равенство должно иметь место для любого $\mathbf{x}$, поэтому его можно записать в виде матричного уравнения. Если опустить терминальную стоимость и устремить время к бесконечности, то член $\dot{\mathbf{P}}$ исчезнет, и мы приходим к алгебраическому уравнению Риккати:

$$\mathbf{P}\mathbf{A} + \mathbf{A}\mathbf{P}^* - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^*\mathbf{P} + \mathbf{Q} = \mathbf{0}.$$

Хотя процедура довольно запутанная, каждый шаг в отдельности сравнительно прост. Кстати говоря, динамику в уравнении (8.48) можно заменить нелинейной динамикой $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, и аналогичную нелинейную двухточечную краевую задачу сформулировать, заменив $\mathbf{A}$ на $\partial\mathbf{f}/\partial\mathbf{x}$, а $\mathbf{B}$ – на $\partial\mathbf{f}/\partial\mathbf{u}$. Это очень общая процедура, которую можно применять для численного получения траекторий нелинейного оптимального управления.

Гамильтонова формулировка. По аналогии с лагранжевой формулировкой, описанной выше, можно решить задачу оптимизации, введя в рассмотрение следующий гамильтониан:

$$\mathcal{H} = \underbrace{\frac{1}{2}(\mathbf{x}^*\mathbf{Q}\mathbf{x} + \mathbf{u}^*\mathbf{R}\mathbf{u})}_{\mathcal{L}} + \boldsymbol{\lambda}^*(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}). \tag{8.54}$$

Тогда уравнение Гамильтона принимает вид:

$$\dot{\mathbf{x}} = \left(\frac{\partial \mathcal{H}}{\partial \boldsymbol{\lambda}}\right)^* = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \qquad \mathbf{x}(0) = \mathbf{x}_0,$$

$$-\dot{\boldsymbol{\lambda}} = \left(\frac{\partial \mathcal{H}}{\partial \mathbf{x}}\right)^* = \mathbf{Q}\mathbf{x} + \mathbf{A}^*\boldsymbol{\lambda} \qquad \boldsymbol{\lambda}(t_f) = \mathbf{Q}_f\mathbf{x}(t_f).$$

Это также двухточечная краевая задача относительно $\mathbf{x}$ и $\boldsymbol{\lambda}$. Подставляя то же выражение $\boldsymbol{\lambda} = \mathbf{P}\mathbf{x}$, мы снова придем к уравнению Риккати.

8.5. Оптимальное оценивание полного состояния: фильтр Калмана

Оптимальный ЛКР-регулятор из раздела 8.4 зависит от наличия измерения полного состояния системы. Однако получить измерения полного состояния может оказаться невозможно по экономическим или техническим причинам, особенно для систем высокой размерности. Кроме того, объем вычислений, необходимых для сбора и обработки измерений полного состояния, может быть настолько велик, что временные задержки при вычислении робастного поведения неприемлемы.

Вместо того чтобы измерять полное состояние $\mathbf{x}$, иногда можно оценить состояние по ограниченным зашумленным измерениям $\mathbf{y}$. На самом деле оценка полного состояния теоретически возможна, если пара ($\mathbf{A}$, $\mathbf{C}$) наблюдаемая, хотя эффективность оценки зависит от степени наблюдаемости, количественно выражаемой грамианом наблюдаемости. Фильтр Калмана [279, 551, 221] чаще всего используется для оценки полного состояния, поскольку он находит оптимальный баланс между шумом измерений, возмущениями и неопределенностью модели. В следующем разделе мы покажем, что оценку полного состояния, полученную с помощью фильтра Калмана, можно использовать совместно с оптимальным законом управления ЛКР с обратной связью, предполагающим доступность полного состояния.

При выводе оптимальной оценки полного состояния необходимо восстановить возмущения состояния $\mathbf{w}_d$ и шум в датчиках $\mathbf{w}_n$:

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{w}_d; \tag{8.56a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} + \mathbf{w}_n. \tag{8.56b}$$

Фильтр Калмана предполагает, что возмущения и шум – гауссовы случайные процессы с нулевым средним и известными ковариациями:

$$\mathbb{E}(\mathbf{w}_d(t)\mathbf{w}_d(\tau)^*) = \mathbf{V}_d\delta(t-\tau), \tag{8.57a}$$
$$\mathbb{E}(\mathbf{w}_n(t)\mathbf{w}_n(\tau)^*) = \mathbf{V}_n\delta(t-\tau). \tag{8.57b}$$

Здесь $\mathbb{E}$ – математическое ожидание, а $\delta(\cdot)$ – дельта-функция Дирака. Положительно полуопределенные матрицы $\mathbf{V}_d$ и $\mathbf{V}_n$ содержат ковариации возмущения и шума соответственно. Существуют обобщения фильтра Калмана для коррелированных, смещенных и неизвестных членов шума и возмущений [498, 372].

Оценку $\hat{\mathbf{x}}$ полного состояния $\mathbf{x}$ по измерениям входа $\mathbf{u}$ и выхода $\mathbf{y}$ можно получить из следующей динамической системы-оценивателя:

$$\frac{d}{dt}\hat{\mathbf{x}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{K}_f(\mathbf{y} - \hat{\mathbf{y}}); \tag{8.58a}$$
$$\hat{\mathbf{y}} = \mathbf{C}\hat{\mathbf{x}} + \mathbf{D}\mathbf{u}. \tag{8.58b}$$

Матрицы $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ мы получаем из модели системы, а матрица усиления фильтра $\mathbf{K}_f$ определяется с помощью такой же процедуры, как для ЛКР:

$$\mathbf{K}_f = \mathbf{Y}\mathbf{C}^*\mathbf{V}_n, \tag{8.59}$$

где $\mathbf{Y}$ – решение еще одного алгебраического уравнения Риккати:

$$\mathbf{Y}\mathbf{A}^* + \mathbf{A}\mathbf{Y} - \mathbf{Y}\mathbf{C}^*\mathbf{V}_n^{-1}\mathbf{C}\mathbf{Y} + \mathbf{V}_d = \mathbf{0}. \tag{8.60}$$

Это решение обычно называют фильтром Калмана, оно является оптимальной оценкой полного состояния относительно следующей функции стоимости:

$$J = \lim_{t\to\infty}\mathbb{E}((\mathbf{x}(t) - \hat{\mathbf{x}}(t))^*(\mathbf{x}(t) - \hat{\mathbf{x}}(t))). \tag{8.61}$$

Эта функция стоимости неявно включает эффекты возмущений и шума, что необходимо для определения оптимального баланса между агрессивным оцениванием и ослаблением шума. Фильтр Калмана называют линейно-квадратичной оценкой (ЛКО). Стоимость (8.61) вычисляется путем усреднения по ансамблю, состоящему из большого числа реализаций.

В MATLAB матрицу усиления фильтра $\mathbf{K}_f$ возвращает команда

```
>> Kf = lqe(A,Vd,C,Vd,Vn); % матрица усиления фильтра Калмана
```

Оптимальное управление и оптимальное оценивание – двойственные математические задачи, так же, как управляемость и наблюдаемость, поэтому фильтр Калмана можно рассчитать с помощью ЛКР:

```
>> Kf = (lqr(A',C',Vd,Vn))'; % ЛКР и ЛКО - двойственные задачи
```

Схема фильтра Калмана изображена на рис. 8.10.

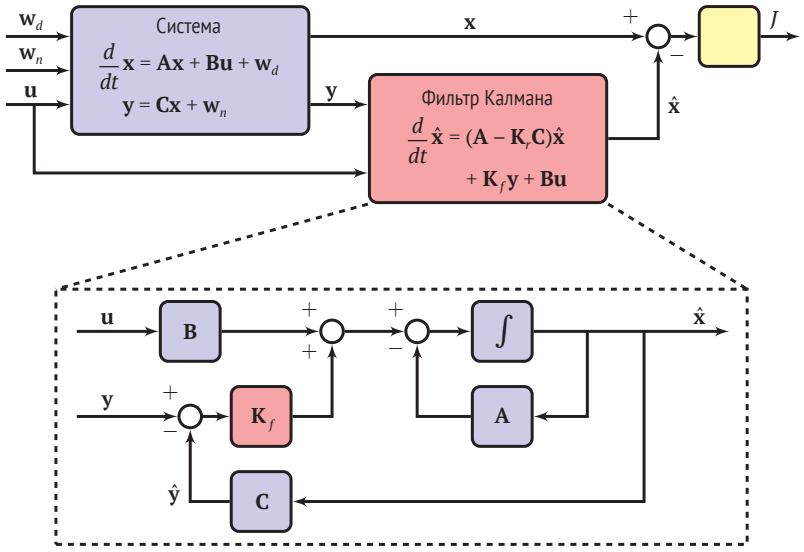

Рис. 8.10 ❖ Схема фильтра Калмана для оценивания полного состояния по зашумленным измерениям $\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{w}_n$ с возмущениями $\mathbf{w}_d$. На диаграмме отсутствует член $\mathbf{D}$, но его можно включить

Подстановка оценки выхода $\hat{\mathbf{y}}$ из (8.58b) в (8.58a) дает:

$$\frac{d}{dt}\hat{\mathbf{x}} = (\mathbf{A} - \mathbf{K}_f\mathbf{C})\hat{\mathbf{x}} + \mathbf{K}_f\mathbf{y} + (\mathbf{B} - \mathbf{K}_f\mathbf{D})\mathbf{u} \tag{8.62a}$$

$$= (\mathbf{A} - \mathbf{K}_f\mathbf{C})\hat{\mathbf{x}} + \begin{bmatrix}\mathbf{K}_f, & (\mathbf{B} - \mathbf{K}_f\mathbf{D})\end{bmatrix}\begin{bmatrix}\mathbf{y}\\ \mathbf{u}\end{bmatrix}. \tag{8.62b}$$

Динамическая система-оцениватель выражается в терминах оценки $\hat{\mathbf{x}}$ с входами $\mathbf{y}$ и $\mathbf{u}$. Если система наблюдаемая, то с помощью подходящего выбора $\mathbf{K}_f$ можно произвольно разместить собственные значения $\mathbf{A} - \mathbf{K}_f\mathbf{C}$. Если собственные значения оценивателя устойчивы, то оценка состояния $\hat{\mathbf{x}}$ асимптотически сходится к полному состоянию $\mathbf{x}$, при условии что модель верно отражает истинную динамику системы. Чтобы убедиться в этом, рассмотрим динамику ошибки оценивания $\boldsymbol{\varepsilon} = \mathbf{x} - \hat{\mathbf{x}}$:

$$\begin{aligned}\frac{d}{dt}\boldsymbol{\mu} &= \frac{d}{dt}\mathbf{x} - \frac{d}{dt}\hat{\mathbf{x}}\\ &= [\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{w}_d] - [(\mathbf{A} - \mathbf{K}_f\mathbf{C})\hat{\mathbf{x}} + \mathbf{K}_f\mathbf{y} + (\mathbf{B} - \mathbf{K}_f\mathbf{D})\mathbf{u}]\\ &= \mathbf{A}\boldsymbol{\mu} + \mathbf{w}_d + \mathbf{K}_f\mathbf{C}\hat{\mathbf{x}} - \mathbf{K}_f\mathbf{y} + \mathbf{K}_f\mathbf{D}\mathbf{u}\\ &= \mathbf{A}\boldsymbol{\mu} + \mathbf{w}_d + \mathbf{K}_f\mathbf{C}\hat{\mathbf{x}} - \mathbf{K}_f\underbrace{[\mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} + \mathbf{w}_n]}_{\mathbf{y}} + \mathbf{K}_f\mathbf{D}\mathbf{u}\\ &= (\mathbf{A} - \mathbf{K}_f\mathbf{C})\boldsymbol{\mu} + \mathbf{w}_d - \mathbf{K}_f\mathbf{w}_n.\end{aligned}$$

Поэтому оценка $\hat{\mathbf{x}}$ сходится к истинному полному состоянию, если собственные значения $\mathbf{A} - \mathbf{K}_f\mathbf{C}$ устойчивы. Как и в случае ЛКР, существует комп-

ромисс между чрезмерной стабилизацией этих собственных значений и усилением шума в датчиках. Это напоминает поведение неопытного водителя, который слишком крепко держит руль и остро реагирует на каждую неровность на дороге.

Существует много вариантов фильтра Калмана для нелинейных систем [274, 275, 538], в т. ч. расширенный и сигма-точечный. Ансамблевый фильтр Калмана [14] – это обобщение, которое хорошо работает с системами высокой размерности, например для ассимиляции геофизических данных [449]. Во всех этих методах по-прежнему предполагаются гауссовы процессы шума, тогда как фильтр частиц предлагает более общее, хотя и более сложное в вычислительном плане решение, пригодное для любого распределения шума [226, 451]. Сигма-точечный фильтр Калмана является компромиссом между эффективностью фильтра Калмана и точностью фильтра частиц.

8.6. Оптимальное управление с использованием датчиков: линейно-квадратичное гауссово управление (ЛКГ)

Оценка полного состояния с помощью фильтра Калмана обычно используется в сочетании с законом управления с обратной связью на основе ЛКР, что выливается в оптимальное управление с обратной связью с использованием датчиков. Примечательно, что матрицу усиления ЛКР $\mathbf{K}_r$ и матрицу усиления фильтра Калмана $\mathbf{K}_f$ можно проектировать по отдельности, а получающееся управление с обратной связью с использованием датчиков останется оптимальным, и после объединения собственные значения системы с замкнутым контуром сохранятся.

Результатом сочетания ЛКР с обратной связью с фильтром Калмана, оценивающим полное состояние, является линейно-квадратичный гауссов (ЛКГ) регулятор. ЛКГ-регулятор – это динамическая система с входом $\mathbf{y}$, выходом $\mathbf{u}$ и внутренним состоянием $\hat{\mathbf{x}}$:

$$\frac{d}{dt}\hat{\mathbf{x}} = (\mathbf{A} - \mathbf{K}_f\mathbf{C} - \mathbf{B}\mathbf{K}_r)\hat{\mathbf{x}} + \mathbf{K}_f\mathbf{y}; \tag{8.63a}$$

$$\mathbf{u} = -\mathbf{K}_r\hat{\mathbf{x}}. \tag{8.63b}$$

ЛКГ-регулятор оптимален относительно следующей усредненной по ансамблю версии функции стоимости (8.44):

$$J(t) = \left\langle \int_0^t \left[\mathbf{x}(\tau)^* \mathbf{Q}\mathbf{x}(\tau) + \mathbf{u}(\tau)^* \mathbf{R}\mathbf{u}(\tau)\right] d\tau \right\rangle. \tag{8.64}$$

Регулятор $\mathbf{u} = -\mathbf{K}_r\hat{\mathbf{x}}$ выражен в терминах оценки состояния, так что эту функцию стоимости необходимо усреднить по большому числу реализаций

возмущения и шума. Применение ЛКР к $\hat{\mathbf{x}}$ приводит к следующей динамике состояния:

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{K}_r\hat{\mathbf{x}} + \mathbf{w}_d \tag{8.65a}$$

$$= \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{K}_r\mathbf{x} + \mathbf{B}\mathbf{K}_r(\mathbf{x} - \hat{\mathbf{x}}) + \mathbf{w}_d \tag{8.65b}$$

$$= \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{K}_r\mathbf{x} + \mathbf{B}\mathbf{K}_r\boldsymbol{\varepsilon} + \mathbf{w}_d. \tag{8.65c}$$

Здесь, как и раньше, $\boldsymbol{\varepsilon} = \mathbf{x} - \hat{\mathbf{x}}$. Наконец, систему с замкнутым контуром можно записать в виде:

$$\frac{d}{dt}\begin{bmatrix}\mathbf{x}\\ \boldsymbol{\mu}\end{bmatrix} = \begin{bmatrix}\mathbf{A} - \mathbf{B}\mathbf{K}_r & \mathbf{B}\mathbf{K}_r\\ \mathbf{0} & \mathbf{A} - \mathbf{K}_f\mathbf{C}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \boldsymbol{\mu}\end{bmatrix} + \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{I} & -\mathbf{K}_f\end{bmatrix}\begin{bmatrix}\mathbf{w}_d\\ \mathbf{w}_n\end{bmatrix}. \tag{8.66}$$

Таким образом, собственные значения ЛКГ-регулируемой системы с замкнутым контуром определяются собственными значениями матриц $\mathbf{A} - \mathbf{B}\mathbf{K}_r$ и $\mathbf{A} - \mathbf{K}_f\mathbf{C}$, которые зависят соответственно от оптимальных матриц усиления ЛКГ и фильтра Калмана.

В схеме ЛКГ, показанной на рис. 8.11, предполагается, что модель системы правильна, а величины возмущений и шума измерений известны и описываются гауссовыми процессами. В реальных системах любое из этих предположений может оказаться неверным, а даже небольшие временные задержки и неопределенность модели могут свести на нет робастность ЛКГ и привести к неустойчивости [155]. Тот факт, что ЛКГ-регуляторы неустойчивы относительно неопределенности модели, является причиной разработки робастного управления, описанного в разделе 8.8. Например, ЛКГ-регулятор можно сделать робастным с помощью процедуры восстановления передаточной функции контура. Однако, несмотря на проблемы робастности, ЛКГ-управление чрезвычайно эффективно во многих системах и занимает достойное место в ряду наиболее употребительных парадигм управления.

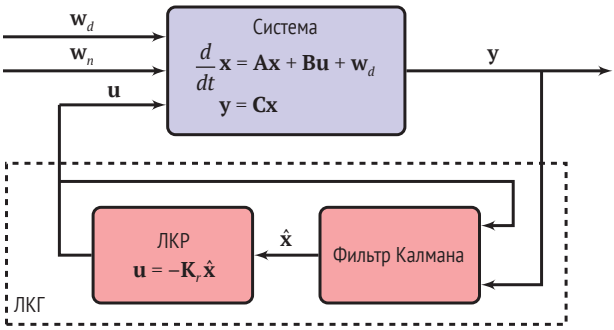

Рис. 8.11 ❖ Схема линейно-квадратичного гауссова (ЛКГ) регулятора для оптимального управления с обратной связью, основанного на зашумленных измерениях $\mathbf{y}$. Матрицы усиления оптимального ЛКГ и фильтра Калмана ($\mathbf{K}_r$ и $\mathbf{K}_f$) можно проектировать независимо на основе двух разных алгебраических уравнений Риккати. После объединения результирующая обратная связь с использованием датчиков остается оптимальной

В отличие от классических подходов к управлению, например пропорционально-интегрально-дифференциального (ПИД) регулирования и проектирования более быстрого управления во внутреннем контуре и более медленного во внешнем, что предполагает разделение временных шкал, ЛКГ-регулятор способен работать в системах с несколькими входами и несколькими выходами (MIMO) с перекрывающимися временными шкалами и многокритериальными целевыми функциями, не увеличивая при этом сложность алгоритмов и реализации.

8.7. Практический пример: обратный маятник на тележке

Чтобы собрать воедино рассмотренные концепции оптимального управления, мы реализуем стабилизирующий регулятор для обратного маятника, установленного на тележке (рис. 8.12). Полная нелинейная динамика описывается уравнениями:

$$\dot{x} = v; \tag{8.67a}$$

$$\dot{v} = \frac{-m^2L^2g\cos(\theta)\sin(\theta) + mL^2(mL\omega^2\sin(\theta) - \delta v) + mL^2u}{mL^2(M + m(1 - \cos(\theta)^2))}; \tag{8.67b}$$

$$\dot{\theta} = \omega; \tag{8.67c}$$

$$\mathbf{y} = \underbrace{(\mathbf{I} + \mathbf{GK})^{-1}\mathbf{GK}}_{\mathbf{T}}\mathbf{w}_r - \underbrace{(\mathbf{I} + \mathbf{GK})^{-1}\mathbf{GK}}_{\mathbf{T}}\mathbf{w}_n + \underbrace{(\mathbf{I} + \mathbf{GK})^{-1}}_{\mathbf{S}}\mathbf{G}_d\mathbf{w}_d, \tag{8.67d}$$

где x – положение тележки, v – скорость, θ – угол наклона маятника, ω – угловая скорость, m – масса маятника, M – масса тележки, L – длина маятника, g – ускорение силы тяжести, δ – фрикционное гашение колебаний на тележке, u – управляющее воздействие, приложенное к тележке.

Для моделирования этой полной нелинейной системы (8.67) можно воспользоваться следующей функцией **pendcart**, написанной на MATLAB:

Листинг 8.2 ❖ Функция для моделирования обратного маятника на тележке

```
function dx = pendcart(x,m,M,L,g,d,u)

Sx = sin(x(3));
Cx = cos(x(3));
D = m*L*L*(M+m*(1-Cx^2));

dx(1,1) = x(2);
dx(2,1) = (1/D)*(-m^2*L^2*g*Cx*Sx + m*L^2*(m*L*x(4)^2*Sx - d*x
    (2))) + m*L*L*(1/D)*u;
dx(3,1) = x(4);
dx(4,1) = (1/D)*((m+M)*m*g*L*Sx - m*L*Cx*(m*L*x(4)^2*Sx - d*x(2)
    )) - m*L*Cx*(1/D)*u;
```

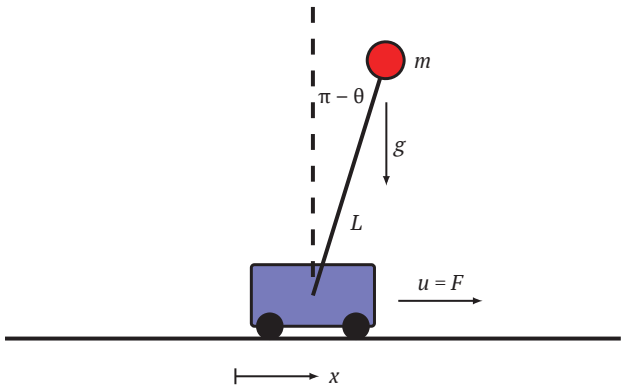

Рис. 8.12 ❖ Схема обратного маятника на тележке. Прилагаемая в результате управления сила ускоряет или замедляет тележку. В этом примере предполагаются следующие значения параметров: масса маятника (m = 1), масса тележки (M = 5), длина маятника (L = 2), ускорение силы тяжести (g = −10) и гашение колебаний на тележке (δ = 1)

Существуют две неподвижные точки, соответствующие нижнему ($\theta = 0$) и верхнему ($\theta = \pi$) положениям маятника; в обоих случаях $v = \omega = 0$, а положение тележки x является свободной переменной, поскольку уравнения явно не зависят от x. Уравнения (8.67) можно линеаризовать в окрестности верхней и нижней точек, что дает следующую линеаризованную динамику:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{\delta}{M} & b\frac{mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -b\frac{\delta}{ML} & -b\frac{(m+M)g}{ML} & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ b\frac{1}{ML} \end{bmatrix} u, \quad \text{при} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x \\ v \\ \theta \\ \omega \end{bmatrix}, \tag{8.68}$$

где $b = 1$ – для верхней неподвижной точки и $b = -1$ для нижней. Матрицы **A** и **B** можно ввести в MATLAB, воспользовавшись значениями констант, приведенными на рис. 8.12.

Листинг 8.3 ❖ Построение матриц системы обратного маятника на тележке

```
clear all, close all, clc

m = 1; M = 5; L = 2; g = -10; d = 1;

b = 1; % верхняя точка маятника (b = 1)

A = [0 1 0 0;
     0 -d/M b*m*g/M 0;
     0 0 0 1;
     0 -b*d/(M*L) -b*(m+M)*g/(M*L) 0];
B = [0; 1/M; 0; b*1/(M*L)];
```

Мы также можем вычислить собственные значения **A** и убедиться, что система с разомкнутым контуром неустойчива:

```
>> lambda = eig(A)
lambda =
         0
   -2.4311
   -0.2336
    2.4648
```

Ниже мы проверим систему на управляемость и наблюдаемость, разработаем регулятор с обратной связью при наличии полного состояния (ЛКР), дадим оценку полного состояния (фильтр Калмана) и построим ЛКГ-регулятор с обратной связью на основе датчиков.

Управление маятником на тележке с обратной связью

В этом разделе мы спроектируем ЛКР-регулятор для стабилизации обратного маятника ($\theta = \pi$) в предположении, что имеются измерения полного состояния $\mathbf{y} = \mathbf{x}$. Прежде чем что-либо проектировать, нужно убедиться, что система допускает линейное управление с данными матрицами **A** и **B**:

```
>> rank(ctrb(A,B))
ans =
      4
```

Таким образом, пара (**A**, **B**) управляема, поскольку матрица управляемости имеет полный ранг. Следовательно, мы можем определить матрицы **Q** и **R** для функции стоимости и вычислить матрицу усиления **K** ЛКР-регулятора:

Листинг 8.4 ❖ Проектирование ЛКР-регулятора для стабилизации обратного маятника на тележке

```
%% Спроектировать ЛКР-регулятор
Q = eye(4); % матрица 4×4
R = .0001;

K = lqr(A,B,Q,R);
```

Далее мы можем смоделировать характеристику полной нелинейной системы с замкнутым контуром. Пусть в начальный момент система немного отклонилась от точки равновесия: $\mathbf{x}_0 = [-1 \;\; 0 \;\; \pi + .1 \;\; 0]^T$, и мы хотим довести тележку от $x = -1$ до $x = 1$.

Листинг 8.5 ❖ Моделирование обратного маятника на тележке

```
%% Моделировать систему с замкнутым контуром
tspan = 0:.001:10;
```

```
x0 = [-1; 0; pi+.1; 0];  % начальное условие
wr = [1; 0; pi; 0];      % конечное положение
u=@(x)-K*(x - wr);       % закон управления
[t,x] = ode45(@(t,x)pendcart(x,m,M,L,g,d,u(x)),tspan,x0);
```

В этой программе управляющее воздействие имеет вид:

$$u = -\mathbf{K}(\mathbf{x} - \mathbf{w}_r), \tag{8.69}$$

где $\mathbf{w}_r = [1 \;\; 0 \;\; \pi \;\; 0]^T$ – конечное положение. Характеристика системы показана на рис. 8.13.

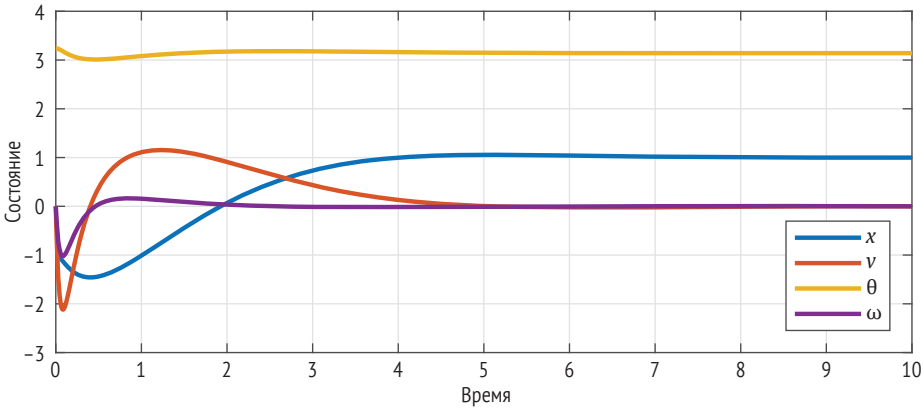

Рис. 8.13 ❖ Характеристика системы обратного маятника на тележке, стабилизированной ЛКР-регулятором

В описанной выше процедуре задание динамики системы и моделирование ее характеристики значительно сложнее, чем собственно проектирование регулятора, которое сводится к вызову всего одной функции в MATLAB. Полезно также сравнить характеристику ЛКР с характеристикой при неоптимальном размещении собственных значений. В частности, на рис. 8.14 показана характеристика системы и функция стоимости для 100 случайно сгенерированных наборов устойчивых собственных значений в интервале $[-3.5, -0.5]$. У ЛКР-регулятора наименьшая общая стоимость, поскольку он и строился с целью минимизации J. Код, рисующий графики системы маятник–тележка, имеется в сети.

Неминимально-фазовые системы. Из характеристики видно, что для перемещения из $x = -1$ в $x = 1$ система сначала движется не в том направлении. Это поведение означает, что система *неминимально-фазовая*, что, в свою очередь, создает проблемы для робастного управления, как мы вскоре увидим. В теории и практике управления существует много примеров неминимально-фазовых систем. Например, для параллельной парковки автомобиля нужно сначала отвести его центр масс от бордюра, а только потом притереть машину. Другой пример – увеличение высоты самолета, когда рули высоты сначала

смещают центр масс вниз, чтобы увеличить угол атаки на основных крыльях, и только потом подъемная сила увеличивает высоту. Добавление холодного топлива в турбину также сначала понижает температуру, но в конечном итоге она увеличивается.

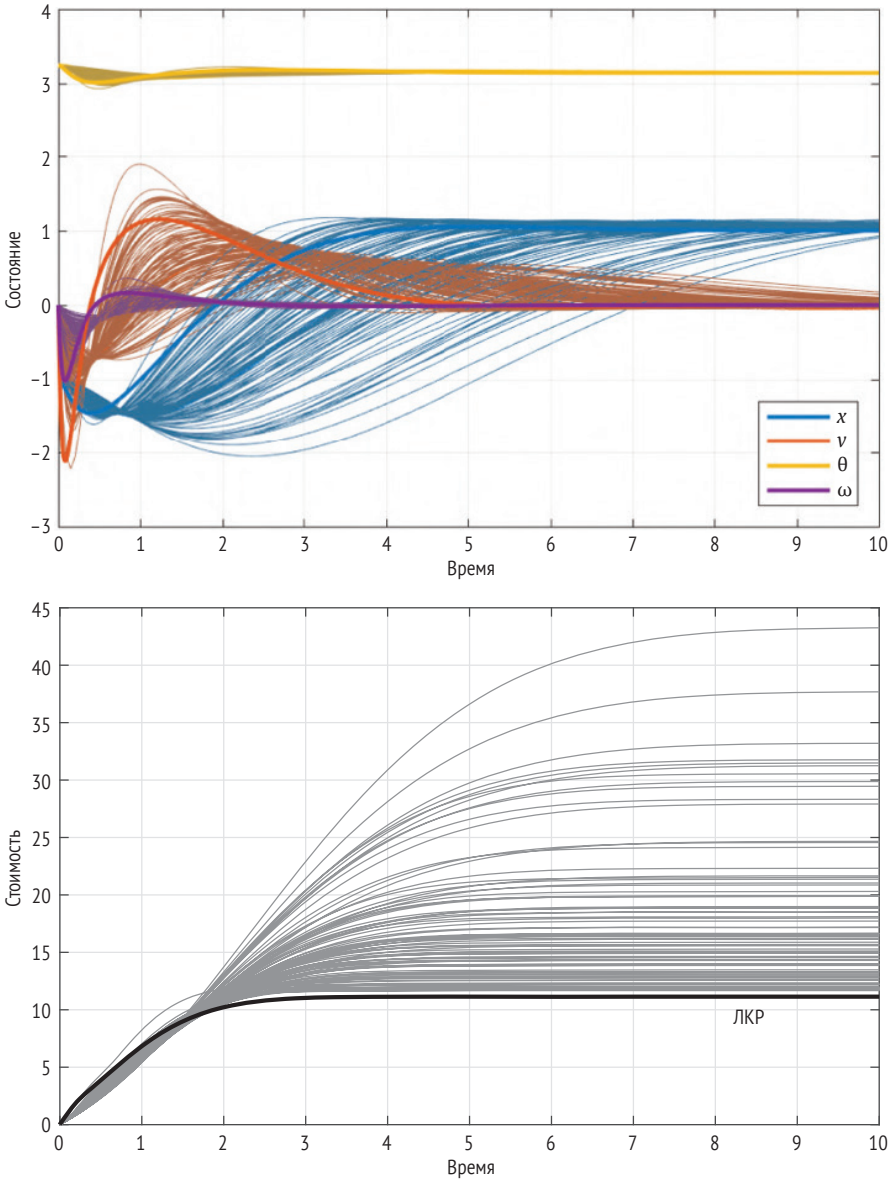

Рис. 8.14 ❖ Сравнение характеристики и функции стоимости ЛКР-регулятора с другими размещениями собственных значений. Жирные линии соответствуют ЛКР-решению

Оценка полного состояния системы маятник–тележка

Обратимся теперь к задаче оценивания полного состояния по ограниченным зашумленным измерениям **y**. Для этого мы разработаем фильтр Калмана для условия, когда маятник находится в нижней точке ($\theta = 0$), поскольку без обратной связи система, находящаяся в состоянии «маятник в верхней точке», быстро покинет неподвижную точку, в окрестности которой модель линейна. Сочетание фильтра Калмана с ЛКР в следующем примере позволит нам управлять неустойчивой конфигурацией обратного маятника. Программно перейти в состояние «маятник в нижней точке» просто:

```
b = -1; % маятник внизу (b = -1)
```

Прежде чем проектировать фильтр Калмана, мы должны выбрать датчик и проверить, является ли система наблюдаемой. Если мы будем измерять положение тележки, $y = x_1$,

```
C = [1 0 0 0]; % измерить положение тележки x
```

то матрица наблюдаемости будет иметь полный ранг:

```
>> rank(obsv(A,C))
ans =
     4
```

Поскольку положение тележки x_1 явно не участвует в динамике, система не является полностью наблюдаемой ни для какого измерения, не включающего x_1. Поэтому невозможно оценить положение тележки, имея только измерение угла маятника. Однако если положение тележки несущественно для функции стоимости (т. е. если мы хотим только стабилизировать маятник, не обращая внимания на положение тележки), то допустимы и другие датчики.

Теперь спроектируем фильтр Калмана, задав ковариации возмущения и шума:

```
%% Задать величины возмущения и шума
Vd = eye(4);  % ковариация возмущения
Vn = 1;       % ковариация шума

% Построить фильтр Калмана
[Kf,P,E] = lqe(A,eye(4),C,Vd,Vn); % фильтр Калмана
% Можно также спроектировать фильтр с помощью кода ЛКР
Kf = (lqr(A',C',Vd,Vn))';
```

Получается такая матрица усиления фильтра Калмана:

```
Kf =
    1.9222
    1.3474
   -0.6182
   -1.8016
```

Наконец, чтобы выполнить моделирование системы и фильтра Калмана, нужно включить в исходную систему возмущение и шум:

```
%% Включить в систему дополнительные входные данные
B_aug = [B eye(4) 0*B];  % [u I*wd 0*wn]
D_aug = [0 0 0 0 0 1];   % матрица D пропускает шум

sysC = ss(A,B_aug,C,D_aug); % система с одним измерением

% "истинная" система с полным состоянием на выходе, возмущением и шумом
sysTruth = ss(A,B_aug,eye(4),zeros(4,size(B_aug,2)));

sysKF = ss(A-Kf*C,[B Kf],eye(4),0*[B Kf]); % фильтр Калмана
```

Теперь смоделируем систему с одним измерением, включающую аддитивные возмущения и шум, и воспользуемся этой моделью для подачи данных на вход фильтра Калмана. В моменты $t = 1$ и $t = 15$ придадим системе большие положительный и отрицательный импульсы соответственно.

```
%% Оценить линеаризованную систему, когда маятник находится в нижней точке
dt = .01;
t = dt:dt:50;

uDIST = sqrt(Vd)*randn(4,size(t,2)); % случайное возмущение
uNOISE = sqrt(Vn)*randn(size(t));    % случайный шум
u = 0*t;
u(1/dt) = 20/dt;   % положительный импульс
u(15/dt) = -20/dt; % отрицательный импульс

u_aug = [u; uDIST; uNOISE]; % вход с возмущением и шумом

[y,t] = lsim(sysC,u_aug,t);         % зашумленные измерения
[xtrue,t] = lsim(sysTruth,u_aug,t); % истинное состояние
[xhat,t] = lsim(sysKF,[u; y'],t);   % оценка состояния
```

На рис. 8.15 показан зашумленный измеренный сигнал, используемый в фильтре Калмана, а на рис. 8.16 – полное незашумленное состояние с возмущениями и оценка, найденная фильтром Калмана.

Для тренировки интуиции рекомендуем читателю исследовать результаты фильтра Калмана в случае, когда модель неточно представляет истинную динамику. В сочетании с управлением при полной информации о состоянии, рассматриваемым в следующем разделе, добавление небольших временных задержек и внесение изменений в модель системы могут стать причиной недостоверности.

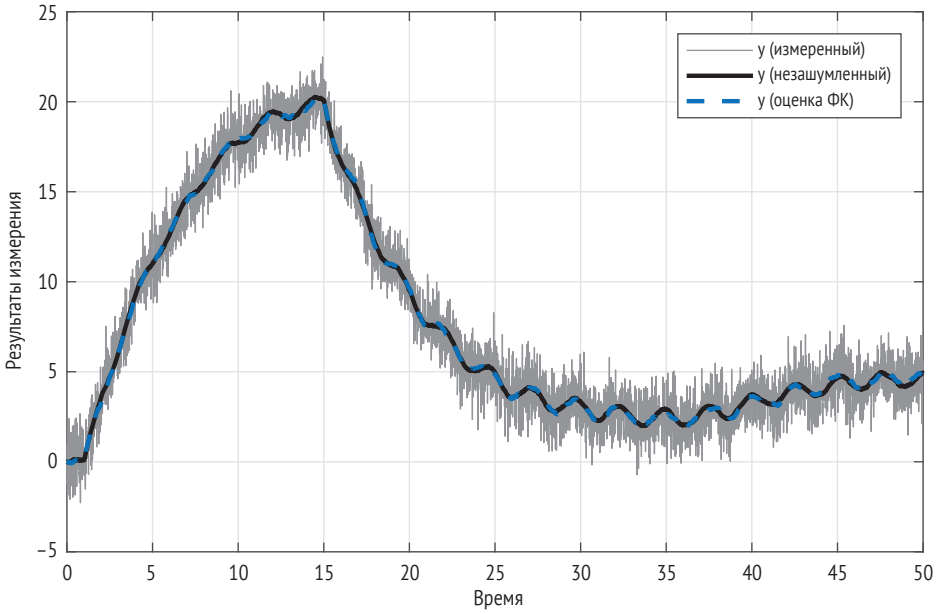

Рис. 8.15 ❖ Зашумленные измерения, используемые в фильтре Калмана, а также исходный незашумленный сигнал и оценка, полученная фильтром Калмана

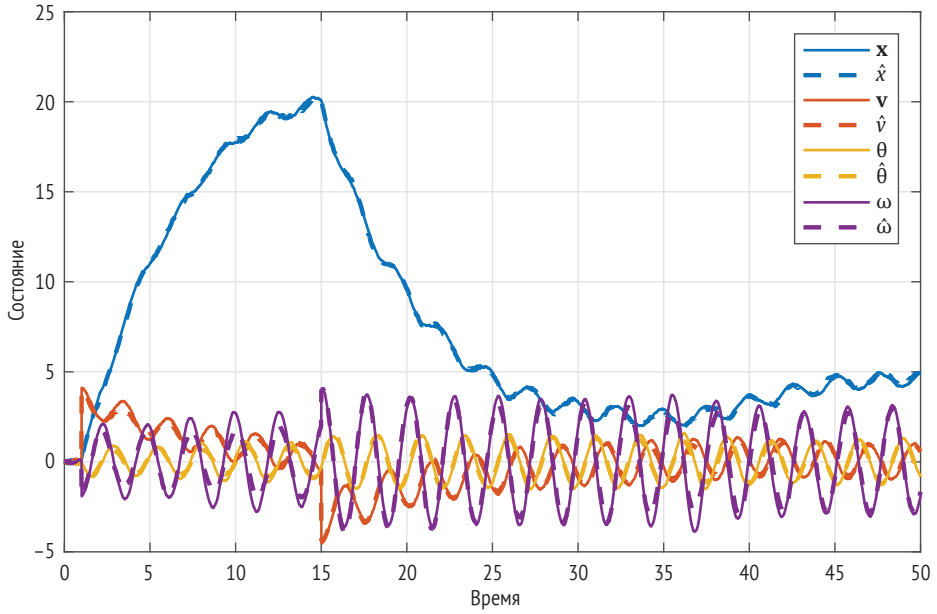

Рис. 8.16 ❖ Истинное и оцененное фильтром Калмана состояния маятника на тележке

Управление с обратной связью системой маятник–тележка с использованием датчиков

Для применения ЛКГ-регулятора к обратному маятнику на тележке мы выполним моделирование полной нелинейной системы в среде Simulink, как показано на рис. 8.17. Нелинейная динамика инкапсулирована в блоке **cartpend_sim**, а вход состоит из исполнительной команды u и возмущения $\mathbf{w}_d$. Мы запоминаем все состояние для анализа работы, хотя фильтру Калмана передаются только зашумленные измерения $y = \mathbf{C}\mathbf{x} + w_n$ и исполнительная команда u. Затем оценка полного состояния передается блоку ЛКР, который формирует желаемую исполнительную команду. В этом примере используются следующие матрицы весов ЛКР и ЛКО:

```
Q = eye(4);       % стоимость состояния
R = .000001;      % стоимость управляющего воздействия
Vd = .04*eye(4); % ковариация возмущений
Vn = .0002;       % ковариация шума
```

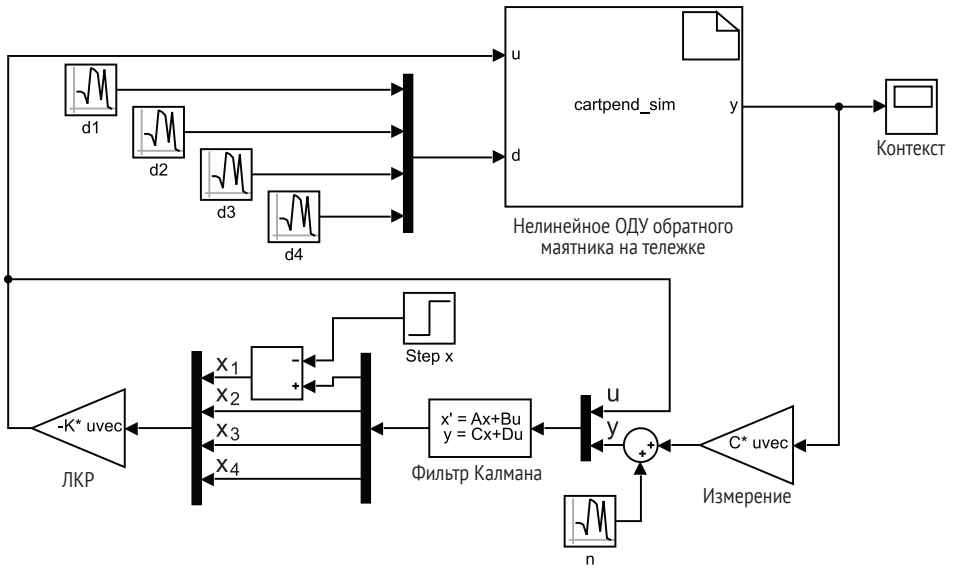

Рис. 8.17 ❖ Модель MATLAB Simulink для ЛКГ-управления с обратной связью с использованием датчиков

Система начинает работу в состоянии вертикального равновесия – точке $\mathbf{x}_0 = [\,0 \;\; 0 \;\; 3.14 \;\; 0\,]^T$, и мы подаем сигнал для перемещения тележки из $x = 0$ в $x = 1$ в момент $t = 10$. Результирующая характеристика показана на рис. 8.18. Несмотря на зашумленные измерения (рис. 8.19) и возмущения (рис. 8.20), регулятор смог отследить положение тележки и стабилизировать обратный маятник.

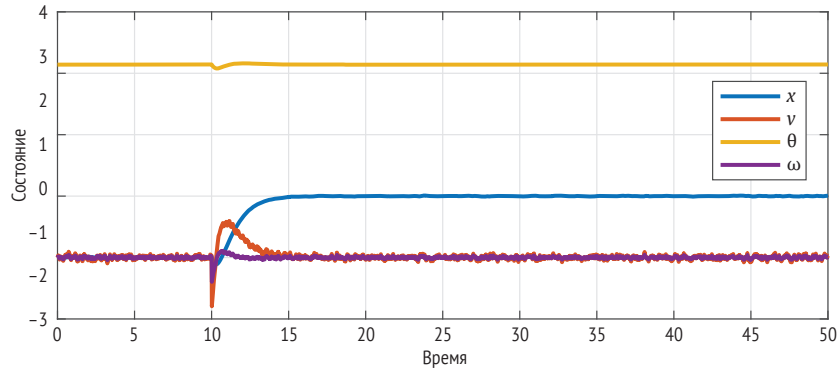

Рис. 8.18 ❖ Выходная характеристика при использовании ЛКГ-регулятора с обратной связью

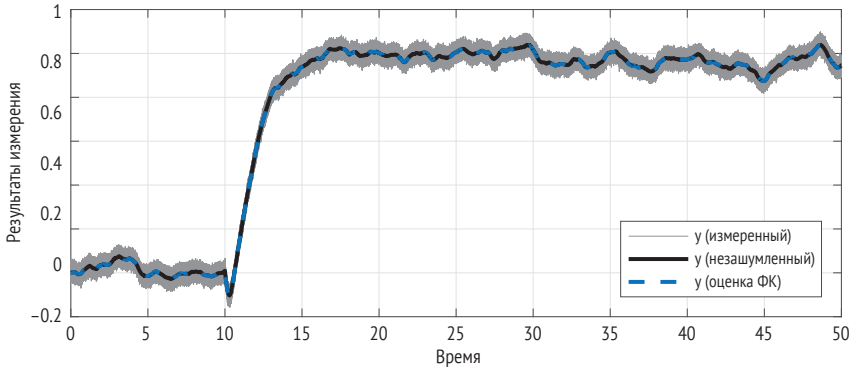

Рис. 8.19 ❖ Зашумленные измерения, использованные в фильтре Калмана, а также исходный незашумленный сигнал и оценка, полученная фильтром Калмана

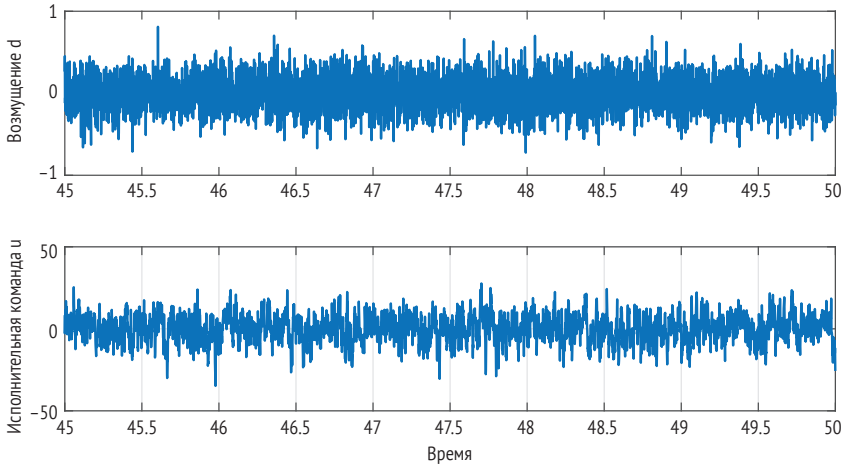

Рис. 8.20 ❖ Зашумленные измерения, использованные в фильтре Калмана, а также исходный незашумленный сигнал и оценка, полученная фильтром Калмана

8.8. Робастное управление и методы анализа в частотной области

До сих пор мы описывали системы управления в терминах систем обыкновенных дифференциальных уравнений в пространстве состояний. Этот подход ведет к анализу устойчивости и проектированию путем размещения собственных значений системы с замкнутым контуром. Однако в основополагающей работе Джона Дойла 1978 года [155][1] было показано, что ЛКГ-регуляторы могут иметь сколь угодно малый запас устойчивости, что делает их *уязвимыми* к неопределенности, временным задержкам и другим несовершенствам модели.

К счастью, спустя немного времени после знаменитой статьи Дойла была разработана строгая математическая теория проектирования робастных регуляторов. В действительности эта новая теория робастного управления обобщает схему оптимального управления, применявшуюся для разработки ЛКР/ЛКГ, путем включения другой функции стоимости, которая штрафует за приближение к *наихудшему сценарию*.

Чтобы понять, как проектируются робастные регуляторы, полезно взглянуть на *передаточные функции* различных сигналов в частотной области. В частности, мы рассмотрим функцию чувствительности, дополнительную функцию чувствительности и передаточную функцию контура. Это позволит нам развить количественный и визуальный подходы к оценке робастности, а также разработать интуитивно понятные и компактные представления систем управления.

Робастное управление возникает естественно при рассмотрении неопределенных моделей, полученных по зашумленным или неполным данным. Кроме того, иногда возможно обращаться с нелинейностью системы как с некоторой формой структурной неопределенности модели. Наконец, мы обсудим известные факторы, ограничивающие робастность, в т. ч. временные задержки и неминимально-фазовое поведение.

Методы в частотной области

Чтобы понять, как устроен компромисс между робастностью и качеством в системе управления, полезно проектировать и анализировать регуляторы с помощью частотных методов.

Для перехода между временной областью (пространством состояний) и частотной областью служит преобразование Лапласа:

$$\mathcal{L}\{f(t)\} = f(s) = \int_{0-}^{\infty} f(t)e^{-st}dt. \tag{8.70}$$

Здесь s – комплексная переменная Лапласа. Преобразование Лапласа можно рассматривать как одностороннее обобщенное преобразование Фурье,

[1] Название статьи: Guaranteed margins for LQG regulators. Автореферат: отсутствует.

применимое к функциям, которые не стремятся к нулю при $t \to \infty$. Преобразование Лапласа особенно полезно тем, что преобразует дифференциальные уравнения в алгебраические, а интегральная свертка во временной области становится простым произведением в частотной. Чтобы понять, что происходит с производными при преобразовании Лапласа, воспользуемся интегрированием по частям:

$$\begin{aligned}\mathcal{L}\left\{\frac{d}{dt}f(t)\right\} &= \int_{0^-}^{\infty} \underbrace{\frac{d}{dt}f(t)}_{dv}\underbrace{e^{-st}}_{u}dt \\ &= \left[f(t)e^{-st}\right]_{t=0^-}^{t=\infty} - \int_{0^-}^{\infty} f(t)(-se^{-st})dt \\ &= f(0^-) + s\mathcal{L}\{f(t)\}.\end{aligned}$$

Таким образом, для нулевых начальных условий $\mathcal{L}\{df/dt\} = sf(s)$.

Применив преобразование Лапласа к системе управления (8.10), получим

$$s\mathbf{x}(s) = \mathbf{A}\mathbf{x}(s) + \mathbf{B}\mathbf{u}(s), \tag{8.71a}$$

$$\mathbf{y}(s) = \mathbf{C}\mathbf{x}(s) + \mathbf{D}\mathbf{u}(s). \tag{8.71b}$$

Первое уравнение можно решить относительно $\mathbf{x}(s)$:

$$(s\mathbf{I} - \mathbf{A})\mathbf{x}(s) = \mathbf{B}\mathbf{u}(s) \quad \Rightarrow \quad \mathbf{x}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{u}(s). \tag{8.72}$$

Подставляя это выражение во второе уравнение, приходим к отображению входов $\mathbf{u}$ на выходы $\mathbf{y}$:

$$\mathbf{y}(s) = [\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{C} + \mathbf{D}]\mathbf{u}(s). \tag{8.73}$$

На основе этого отображения определим *передаточную функцию*:

$$\mathbf{G}(s) = \frac{\mathbf{y}(s)}{\mathbf{u}(s)} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}. \tag{8.74}$$

Для линейных систем существует три эквивалентных представления: 1) во временной области, в терминах импульсной характеристики; 2) в частотной области, в терминах передаточной функции, и 3) в пространстве состояний, в терминах системы дифференциальных уравнений. Схематически эти представления показаны на рис. 8.21. Как мы увидим, анализ систем управления в частотной области имеет много преимуществ.

Частотная характеристика

Передаточная функция (8.74) особенно полезна тем, что с ней связана частотная характеристика – графическое представление системы управления в терминах допускающих измерение данных. Для иллюстрации этого факта рассмотрим систему с одним входом и одним выходом (SISO). Линейные системы с нулевыми начальными условиями обладают тем свойством, что

синусоидальный входной сигнал порождает синусоидальный выходной сигнал той же частоты, но, возможно, с другой амплитудой A и фазой φ:

$$u(t) = \sin(\omega t) \quad \Rightarrow \quad y(t) = A\sin(\omega t + \varphi). \tag{8.75}$$

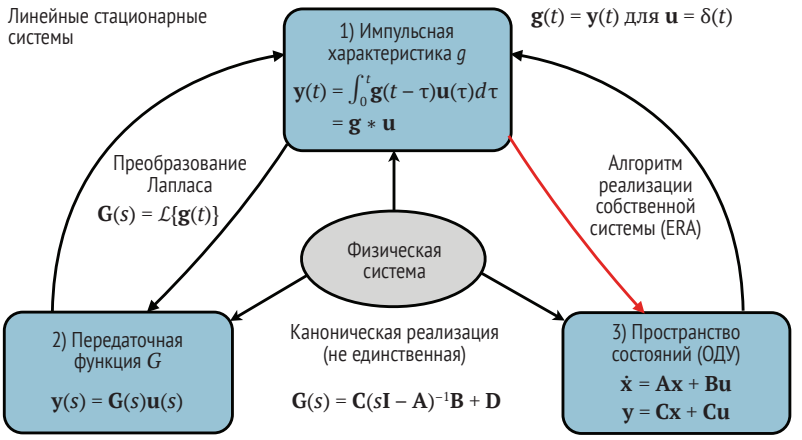

Рис. 8.21 ❖ Три эквивалентных представления линейных стационарных систем

Это верно для длительных промежутков времени, после того как стихнут переходные процессы. Амплитуда A и фаза φ выходной синусоиды зависят от входной частоты ω. Функции $A(\omega)$ и $\varphi(\omega)$ можно определить, поставив ряд экспериментов с входным сигналом различной частоты ω. Или же эту информацию можно получить от комплексной передаточной функции $G(s)$:

$$A(\omega) = |G(i\omega)|, \qquad \varphi(\omega) = \angle G(i\omega). \tag{8.76}$$

Таким образом, амплитуду и фазу для входного сигнала $\sin(\omega t)$ можно найти, вычислив передаточную функцию при $s = i\omega$ (т. е. на мнимой оси комплексной плоскости). Затем эти величины можно нанести на график и получить *частотную характеристику*, или *диаграмму Боде*.

В качестве конкретного примера рассмотрим систему пружина–масса–демпфер, показанную на рис. 8.22. Уравнения движения имеют вид:

$$m\ddot{x} = -\delta\dot{x} - kx + u. \tag{8.77}$$

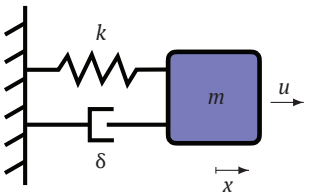

Рис. 8.22 ❖ Система пружина–масса–демпфер

Положив $m = 1, \delta = 1, k = 2$ и применив преобразование Лапласа, получим:

$$G(s) = 1/(s^2 + s + 2). \tag{8.78}$$

Здесь предполагается, что выход y – результат измерения положения массы x. Заметим, что знаменатель передаточной функции $G(s)$ является характеристическим уравнением (8.77), записанным в пространстве состояний. Таким образом, полюсы комплексной функции $G(s)$ – это собственные значения системы в пространстве состояний.

Теперь можно создать эту систему в MATLAB и построить график частотной характеристики, показанный на рис. 8.23. Заметим, что частотная характеристика допускает простую интерпретацию и имеет интуитивно понятный физический смысл. Например, нулевой наклон амплитуды при низких частотах означает, что медленное приложение силы приводит непосредственно к перемещению массы, тогда как спад амплитуды при высоких частотах означает, что быстро приложенная сила гасится и влияет на перемещение массы незначительно. Кроме того, сразу видна резонансная частота, соответствующая пику амплитуды. На этой частоте происходит усиление приложенной силы.

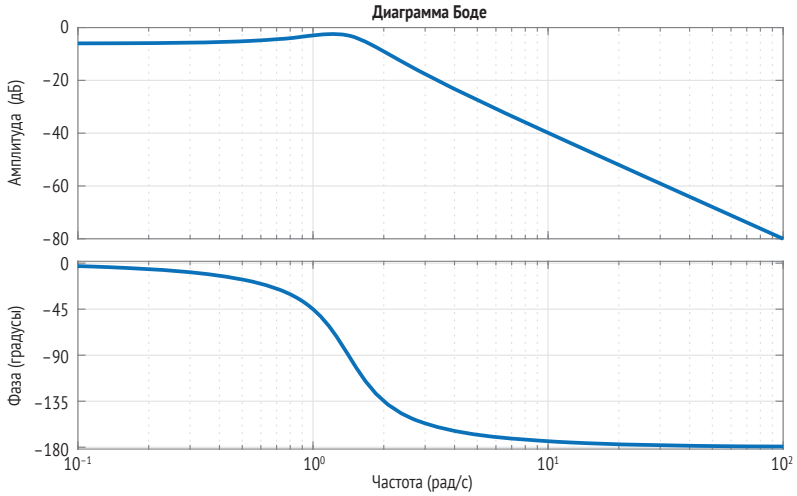

Рис. 8.23 ❖ Частотная характеристика системы пружина–масса–демпфер. График амплитуды построен в логарифмическом масштабе, величина амплитуды выражена в децибелах (дБ). Частота также нанесена в логарифмическом масштабе

Листинг 8.6 ❖ Создание передаточной функции и построение графика частотной характеристики (диаграммы Боде)

```
s = tf('s');          % переменная Лапласа
G = 1/(s^2 + s + 2); % передаточная функция

bode(G);              % частотная характеристика
```

Если дана реализация в пространстве состояний

```
>> A = [0 1; -2 -1];
>> B = [0; 1];
>> C = [1 0];
>> D = 0;
```

то легко получить представление в частотной области:

```
>> [num,den] = ss2tf(A,B,C,D); % переход от пр-ва состояний к перед. функции
>> G = tf(num,den)             % создать передаточную функцию
G =
       1
  -----------
  s^2 + s + 2
```

Аналогично можно получить систему в пространстве состояний, зная передаточную функцию, хотя такое представление не единственно:

```
>> [A,B,C,D] = tf2ss(G.num{1},G.den{1})
A =
   -1.0000 -2.0000
    1.0000 0
B =
    1
    0
C =
    0 1
D =
    0
```

Заметим, что в этом представлении порядок переменных изменен на противоположный: $\mathbf{x} = [v \quad x]^T$, но характеристика входа-выхода по-прежнему правильна.

Частотная область полезна также потому, что импульсные или ступенчатые входные сигналы особенно просто представляются с помощью преобразования Лапласа. Это просто и в MATLAB. Импульсную характеристику (рис. 8.24) возвращает команда

```
>> impulse(G); % импульсная характеристика
```

а ступенчатую характеристику (рис. 8.25) – команда

```
>> step(G); % ступенчатая характеристика
```

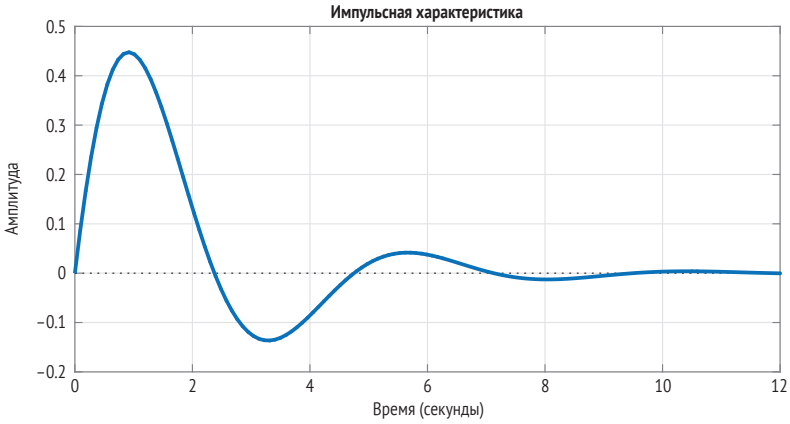

Рис. 8.24 ❖ Импульсная характеристика системы пружина–масса–демпфер

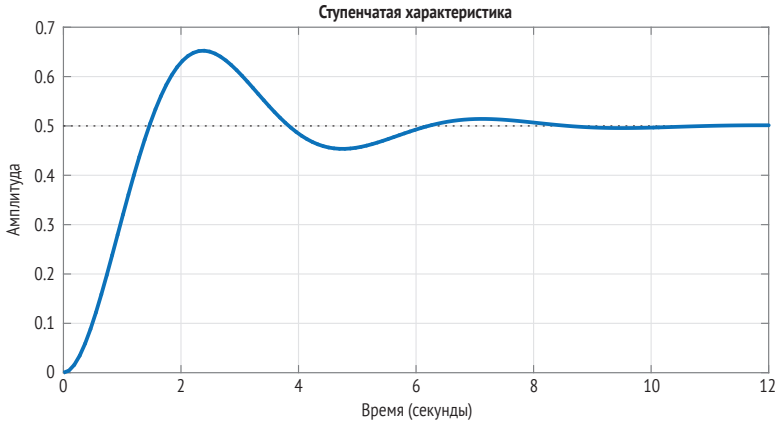

Рис. 8.25 ❖ Ступенчатая характеристика системы пружина–масса–демпфер

Качество управления и передаточная функция контура: чувствительность и дополнительная чувствительность

Рассмотрим немного модифицированный вариант рис. 8.4, когда для возмущения имеется модель $\mathbf{P}_d$. Новую схему, показанную на рис. 8.26, мы используем для вывода важных передаточных функций, имеющих отношение к оценке робастности.

$$\mathbf{y} = \mathbf{GK}(\mathbf{w}_r - \mathbf{y} - \mathbf{w}_n) + \mathbf{G}_d\mathbf{w}_d \tag{8.79a}$$

$$\Rightarrow \quad (\mathbf{I} + \mathbf{GK})\mathbf{y} = \mathbf{GK}\mathbf{w}_r - \mathbf{GK}\mathbf{w}_n + \mathbf{G}_d\mathbf{w}_d. \tag{8.79b}$$

$$\Rightarrow \quad \mathbf{y} = \underbrace{(\mathbf{I} + \mathbf{GK})^{-1}\mathbf{GK}}_{\mathbf{T}}\mathbf{w}_r - \underbrace{(\mathbf{I} + \mathbf{GK})^{-1}\mathbf{GK}}_{\mathbf{T}}\mathbf{w}_n + \underbrace{(\mathbf{I} + \mathbf{GK})^{-1}}_{\mathbf{S}}\mathbf{G}_d\mathbf{w}_d. \tag{8.79c}$$

Здесь **S** – функция чувствительности, **T** – дополнительная функция чувствительности. Мы можем также определить передаточную функцию контура $\mathbf{L} = \mathbf{GK}$, т. е. передаточную функцию разомкнутого контура в отсутствие обратной связи. **S** и **T** можно выразить проще через **L**:

$$\mathbf{S} = (\mathbf{I} + \mathbf{L})^{-1}; \tag{8.80a}$$

$$\mathbf{T} = (\mathbf{I} + \mathbf{L})^{-1}\mathbf{L}. \tag{8.80b}$$

В сумме чувствительность и дополнительная чувствительность дают тождественную функцию: $\mathbf{S} + \mathbf{T} = \mathbf{I}$.

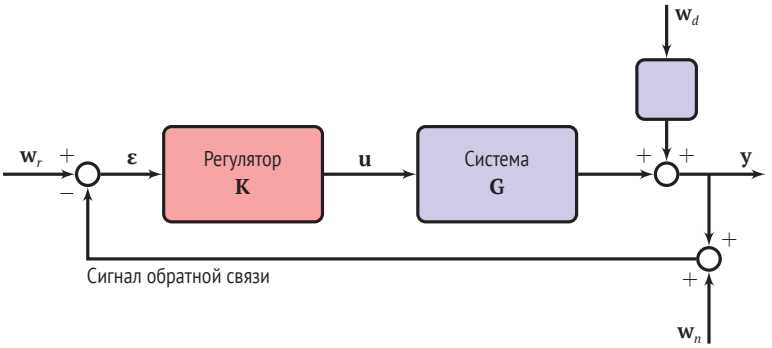

Рис. 8.26 ❖ Диаграмма управления с обратной связью с опорным сигналом, шумом и возмущением. Мы будем рассматривать различные передаточные функции, отображающие экзогенные входные сигналы в незашумленную ошибку ε, и таким образом выведем передаточную функцию контура, а также функцию чувствительности и дополнительную функцию чувствительности

На практике для проектирования более удобна передаточная функция, отображающая экзогенные входные сигналы в незашумленную ошибку ε:

$$\boldsymbol{\varepsilon} = \mathbf{w}_r - \mathbf{y} = \mathbf{S}\mathbf{w}_r + \mathbf{T}\mathbf{w}_n - \mathbf{S}\mathbf{G}_d\mathbf{w}_d. \tag{8.81}$$

Таким образом, мы видим, что функция чувствительности и дополнительная функция чувствительности отображают опорный сигнал, возмущение и шум в ошибку слежения. Поскольку нам нужна малая ошибка слежения, мы можем специфицировать желательные свойства **S** и **T** и в идеале сумеем реализовать эти спецификации, спроектировав передаточную функцию контура **L**. На практике мы выберем регулятор **K** с учетом знаний о модели **G**, так чтобы передаточная функция обладала нужными нам свойствами в частотной области. Например, небольшое усиление при высоких частотах ослабляет шум в датчиках, поскольку при этом **T** будет мало. Аналогично

большое усиление при низких частотах дает хорошее качество слежения, поскольку $\mathbf{S}$ будет мало при низких частотах. Однако $\mathbf{S}$ и $\mathbf{T}$ не могут быть малы всюду одновременно, потому что $\mathbf{S} + \mathbf{T} = \mathbf{I}$, как следует из определений (8.80), следовательно, эти проектные цели конкурируют между собой.

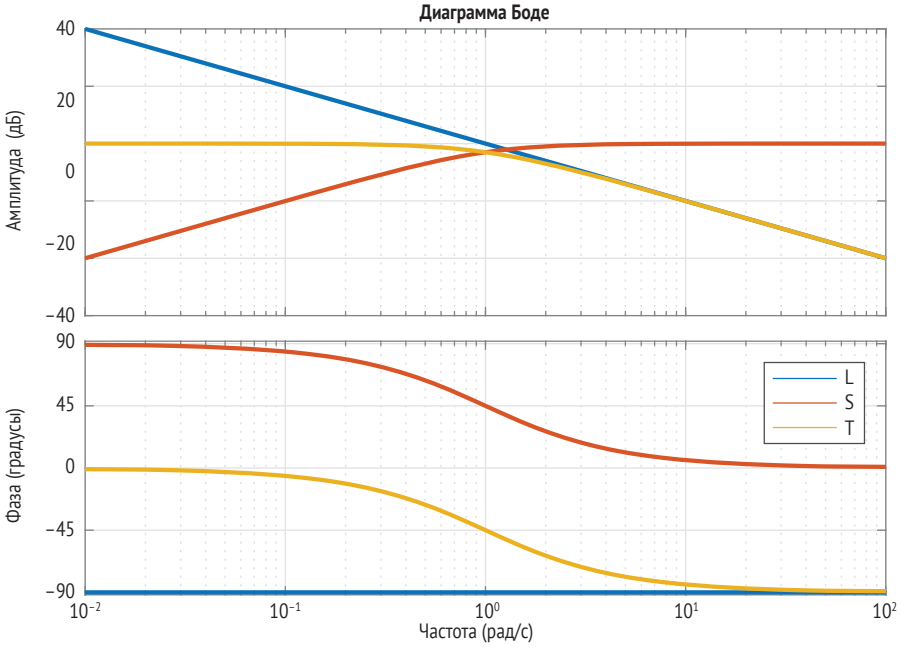

Рис. 8.27 ❖ Передаточная функция контура, а также функция чувствительности и дополнительная функция чувствительности

Для обеспечения высокого качества и робастности нам нужно, чтобы максимум $\mathbf{S}$, $M_{\mathbf{S}} = ||\mathbf{S}||_{\infty}$, был как можно меньше. Из (8.81) понятно, что в отсутствие шума управление с обратной связью повышает качество (т. е. уменьшает ошибку) для всех частот, при которых $|\mathbf{S}| < 1$; следовательно, управление эффективно, когда $\mathbf{T} \approx \mathbf{1}$. Как объясняется в работе [492] (стр. 37), все реальные системы имеют диапазон частот, для которых $|\mathbf{S}| > 1$, а в таком случае качество ухудшается. Минимизация пика $M_{\mathbf{S}}$ сглаживает ухудшение, наблюдаемое в результате обратной связи на этих частотах, а значит, повышает качество. Кроме того, минимальное расстояние от передаточной функции контура $\mathbf{L}$ до точки -1 на комплексной плоскости равно $M_{\mathbf{S}}^{-1}$. Согласно критерию устойчивости Найквиста, чем больше это расстояние, тем больше запас устойчивости системы с замкнутым контуром, что увеличивает ее робастность. Существует две основные причины для минимизации $M_{\mathbf{S}}$.

Полосой пропускания регулятора ω_B называется частота, ниже которой управление с обратной связью эффективно. Это субъективное определение. Часто под ω_B понимают частоту, при которой $|\mathbf{S}(j\omega)|$ впервые пересекает уровень -3 дБ. В идеале хотелось бы, чтобы полоса пропускания регулятора была как можно больше, но чтобы при этом не усиливался шум в датчиках,

как правило, высокочастотный. Однако существуют фундаментальные ограничения на полосу пропускания, действующие для систем с временными задержками или нулями в правой полуплоскости [492].

Обращение динамики

Если модель имеет вид (8.10) или (8.73), то иногда возможно спроектировать закон управления с открытым контуром для реализации желаемой спецификации, не прибегая к управлению с обратной связью или с упреждением, основанному на применении датчиков. Например, если в системе на рис. 8.3 требуется идеальное отслеживание опорного входного сигнала, то при некоторых условиях можно спроектировать регулятор, обратив динамику системы $\mathbf{G}$: $\mathbf{K}(s) = \mathbf{G}^{-1}(s)$. В этом случае передаточная функция, отображающая опорный сигнал $\mathbf{w}_r$ в выход $\mathbf{s}$, удовлетворяет равенству $\mathbf{G}\mathbf{G}^{-1} = \mathbf{1}$, так что выход идеально соответствует опорному сигналу. Однако в реальных системах идеальное управление недостижимо и такую стратегию следует применять с осторожностью, поскольку в общем случае она зависит от ряда существенных предположений о системе $\mathbf{G}$. Прежде всего для эффективного управления, основанного на обращении, необходимы очень точные знания о $\mathbf{G}$, а возмущения должны быть детально описаны и предсказуемы; почти не остается места для ошибок или неопределенностей в описании модели, т. к. нет никаких показаний датчиков, позволяющих определить, соответствует ли качество ожиданиям, и не существует механизмов корректирующей обратной связи, которые могли бы модифицировать стратегию управляющего воздействия для компенсации отклонений.

Кроме того, для управления с открытым контуром с использованием обращения система $\mathbf{G}$ должна быть устойчива. Невозможно фундаментально изменить динамику линейной системы посредством управления с открытым контуром, поэтому неустойчивую систему не удастся стабилизировать без обратной связи. Попытка стабилизировать неустойчивую систему путем обращения динамики почти всегда приводит к печальным последствиям. Например, рассмотрим следующую неустойчивую систему с полюсом в точке $s = 5$ и нулем в точке $s = -10$: $G(s) = (s + 10)/(s - 5)$. Обращение динамики привело бы к регулятору $K = (s - 5)/(s + 10)$; однако если в модели есть хотя бы малейшая неопределенность, т. е. полюс в действительности находится в точке $5 - \varepsilon$, то система с открытым контуром будет иметь вид:

$$G_{\text{true}}(s)K(s) = \frac{s-5}{s-5+\mu}.$$

Помимо устойчивости, система $\mathbf{G}$ должна обладать дополнительными свойствами: отсутствие временных задержек, отсутствие нулей в правой полуплоскости, и число полюсов должно быть равно числу нулей. Если $\mathbf{G}$ имеет нули в правой полуплоскости, то обращенный регулятор $\mathbf{K}$ будет неустойчивым, потому что будет иметь полюсы в правой полуплоскости. Такие системы называются *неминимально-фазовыми*, и существуют обобщения об-

ращения динамики, позволяющие получить для них ограниченные обращения [149]. Аналогично система с временными задержками необратима, а если **G** имеет больше полюсов, чем нулей, то результирующий регулятор нереализуем, поскольку может иметь сколь угодно большие управляющие сигналы **b**. Существуют также обобщения, позволяющие получить *регуляризированное* обращение модели, когда применяются схемы оптимизации со штрафующими членами, призванными ограничить управляющий сигнал **b**. Эти регуляризированные регуляторы с разомкнутым контуром зачастую значительно эффективнее, а их робастность выше.

В общем и целом из этих ограничений на **G** вытекает, что управление с разомкнутым циклом, основанное на модели, следует применять, только когда система корректно определена, точно описывается моделью, когда имеется характеристика возмущений, а оборудование для реализации обратной связи обошлось бы неоправданно дорого. В противном случае цели по качеству работы не должны быть завышенными. Обращение модели с разомкнутым контуром часто используется на производстве и в робототехнике, где системы допускают точное описание и их функционирование ограничено стандартным режимом эксплуатации.

Робастное управление

Как уже было сказано, ЛКГ-регуляторы известны тем, что могут иметь сколь угодно малый запас устойчивости. Это серьезная проблема в таких областях, как управление турбулентностью, нейромеханические системы и эпидемиология, где динамика искажена неопределенностью и временными задержками.

На рис. 8.2 показана самая общая схема управления с замкнутым контуром обратной связи, охватывающая как оптимальную, так и робастную стратегию управления. В современной обобщенной теории управления цель состоит в минимизации передаточной функции, отображающей экзогенные входы **w** (опорный сигнал, возмущения, шум и т. д.) в многокритериальную функцию стоимости **J** (точность, стоимость управляющего воздействия, качество во временной области и т. д.). Оптимальное управление (например, ЛКР, ЛКО, ЛКГ) является оптимальным по норме $\mathcal{H}_2$, ограниченной 2-норме в пространстве Харди, состоящем из устойчивых и строго собственных передаточных функций (это означает, что усиление спадает при высоких частотах). Робастное управление аналогично оптимальному, но по норме $\mathcal{H}_\infty$, описывающей устойчивые собственные передаточные функции (усиление не должно стремиться к бесконечности при высоких частотах). Эта норма определяется следующим образом:

$$||\mathbf{G}||_\infty \triangleq \max_\omega \sigma_1(\mathbf{G}(i\omega)). \tag{8.82}$$

Здесь σ_1 обозначает максимальное сингулярное значение. Поскольку норма $||\cdot||_\infty$ – это максимум значений передаточной функции по всем частотам, ее часто называют *нормой в худшем случае*; следовательно, минимизация по этой

норме обеспечивает робастность даже при самых худших экзогенных входах. $\mathcal{H}_\infty$-робастные регуляторы используются, когда робастность важна. Существует много связей между $\mathcal{H}_2$-управлением и $\mathcal{H}_\infty$-управлением, поскольку схема в обоих случаях одинакова, просто оптимизация производится по разным нормам. Мы отсылаем читателя к отличным учебникам, в которых подробно рассматривается эта теория [492, 165].

Если обозначить $\mathbf{G}_{\mathbf{w}\to\mathbf{J}}$ передаточную функцию, отображающую **w** в **J**, то целью $\mathcal{H}_\infty$-управления является построение регулятора, минимизирующего $\|\mathbf{G}_{\mathbf{w}\to\mathbf{J}}\|_\infty$. Как правило, это трудно, и в общем случае не существует аналитического решения, описывающего оптимальный регулятор в замкнутой форме. Но есть достаточно эффективные итеративные методы нахождения такого регулятора, что $\|\mathbf{G}_{\mathbf{w}\to\mathbf{J}}\|_\infty < \gamma$, они описаны в работе [156]. Правда, для применения этих методов должны быть выполнены многочисленные условия. Кроме того, имеются вычислительно эффективные алгоритмы, реализованные на языках MATLAB и Python, которые не требуют значительного вмешательства со стороны пользователя.

Выбор функции стоимости **J**, которая позволит исполнить техническое задание, также является важной частью проектирования робастного управления. Следует учесть такие вещи, как компенсация возмущений, ослабление шума, полоса пропускания регулятора и стоимость управляющих воздействий. Это делается путем выбора весов во взвешенной сумме передаточных функций **S**, **T** и **KS**. В задаче *смешанной чувствительности* для поиска подходящего баланса этих факторов в различных диапазонах частот подбираются веса этих передаточных функций. Например, мы можем в качестве веса **S** использовать фильтр нижних частот, а в качестве веса **KS** – фильтр верхних частот, так что при низких частотах поощряется компенсация возмущений, а при высоких не поощряется управляющее воздействие. Общая функция стоимости может включать три весовых фильтра $\mathbf{F}_k$, на которые умножаются **S**, **T** и **KS**:

$$\left\| \begin{bmatrix} \mathbf{F}_1\mathbf{S} \\ \mathbf{F}_2\mathbf{T} \\ \mathbf{F}_3\mathbf{KS} \end{bmatrix} \right\|_\infty .$$

Еще один возможный подход к проектированию робастного управления называется формированием контура $\mathcal{H}_\infty$. Во многих задачах эта процедура может оказаться проще, чем синтез смешанной чувствительности. Метод формирования контура состоит из двух основных шагов. Сначала задается передаточная функция системы с разомкнутым контуром, основанная на заданных параметрах качества и с применением классических методов проектирования регулятора. Затем сформированный контур делается робастным относительно широкого класса неопределенностей модели. Фактически процедура формирования контура $\mathcal{H}_\infty$ позволяет спроектировать идеальный регулятор, имеющий заданные технические параметры, например время нарастания, полоса пропускания, время успокоения и т. д. Как правило, форма контура должна обеспечивать большое усиление при низких частотах, чтобы гарантировать точное следование опорному сигналу и компенсацию медленных возмущений, низкое усиление при высоких частотах, чтобы ослабить

шум в датчиках, и частоту среза, гарантирующую желаемую полосу пропускания. Затем передаточная функция контура делается робастной, чтобы повысить запас устойчивости по амплитуде и фазе.

Парадигма оптимального $\mathcal{H}_2$-управления (например, ЛКР, ЛКО и ЛКГ) была чрезвычайно популярна в силу простоты математической постановки и легкости настраивания пользователем. Однако постепенно приходит понимание преимуществ $\mathcal{H}_\infty$-управления. Ко всему прочему, имеются многочисленные программные решения, которые упрощают реализацию. В MATLAB смешанная чувствительность реализуется командой **mixsyn**, входящей в пакет робастного управления. Для формирования контура служит команда **loopsyn** из того же пакета.

Фундаментальные ограничения на качество робастного управления

Как уже было сказано, для повышения робастности мы хотим минимизировать пики **S** и **T**. Какая-то пиковость неизбежна, и у системы имеются характеристики, существенно ограничивающие достижимые качество и робастность. Прежде всего временные задержки и нули в правой полуплоскости системы с разомкнутым контуром ограничивают эффективную полосу пропускания и увеличивают достижимую нижнюю границу пиков **S** и **T**. Это одновременно ухудшает качество управления и снижает робастность.

Аналогично система подвержена ограничениям качества и робастности, если число полюсов превосходит число нулей больше, чем на 2. Эти фундаментальные ограничения количественно выражаются интегралами *водяного ложа* (waterbed integral), названными так, потому что если продавить водяное ложе в одном месте, то оно вспучится в другом. Поэтому попытка уменьшить пик **S** приведет к увеличению других пиков.

Понять, почему вызывают затруднения временные задержки, довольно просто: временная задержка τ привносит дополнительное отставание по фазе $\tau\omega$ на частоте ω, а это ограничивает способность регулятора быстро реагировать (т. е. полосу пропускания). Таким образом, обычно полоса пропускания для регулятора с приемлемым запасом устойчивости по фазе $\omega_B < 1/\tau$.

В соответствии с обсуждением в [492] эти фундаментальные ограничения можно понять, связав их с ограничениями управления с разомкнутым контуром, основанного на обращении модели. Если рассмотреть обратную связь с большим коэффициентом усиления $\mathbf{u} = \mathbf{K}(\mathbf{w}_r - \mathbf{y})$ для системы, описываемой рис. 8.26 и уравнением (8.81), но без возмущений и шума, то будем иметь:

$$\mathbf{u} = \mathbf{K}\boldsymbol{\varepsilon} = \mathbf{K}\mathbf{S}\mathbf{w}_r. \tag{8.83}$$

Мы можем записать это в терминах дополнительной функции чувствительности **T**, заметив, что поскольку $\mathbf{T} = \mathbf{I} - \mathbf{S}$, имеет место равенство $\mathbf{T} = \mathbf{L}(\mathbf{I} + \mathbf{L})^{-1} = \mathbf{G}\mathbf{K}\mathbf{S}$:

$$\mathbf{u} = \mathbf{G}^{-1}\mathbf{T}\mathbf{w}_r. \tag{8.84}$$

Таким образом, на тех частотах, где **T** приближенно равна тождественной функции **I** и управление эффективно, управляющее воздействие фактиче-

ски является обращением **G**. Даже в случае обратной связи с применением датчиков идеальное управление недостижимо. Например, если **G** имеет нули в правой полуплоскости, то управляющий сигнал становится неограниченным, если усиление **K** слишком агрессивно. Точно так же ограничения возникают, когда имеют место временные задержки или число полюсов **G** больше числа нулей, как в случае обращения системы с разомкнутым контуром, основанной на модели.

В качестве последней иллюстрации ограничения, связанного с нулями в правой полуплоскости, рассмотрим случай пропорционального управления $u = Ky$ в системе с одним входом и одним выходом, когда $G(s) = N(s)/D(s)$. Здесь корни числителя $N(s)$ являются нулями, а корни знаменателя $D(s)$ – полюсами. Передаточная функция системы с замкнутым контуром, отображающая опорный сигнал w_r в датчики s, имеет вид:

$$\frac{y(s)}{w_r(s)} = \frac{GK}{1+GK} = \frac{NK/D}{1+NK/D} + \frac{NK}{D+NK}. \tag{8.85}$$

Для небольшого коэффициента усиления K член NK в знаменателе мал, и полюсы системы с замкнутым контуром располагаются рядом с полюсами G, т. е. корнями D. При увеличении K член NK в знаменателе начинает преобладать, и полюсы системы с замкнутым контуром притягиваются к корням N, которые являются нулями системы с разомкнутым контуром G. Таким образом, если у системы с разомкнутым контуром G существуют нули в правой полуплоскости, то пропорциональное управление с высоким коэффициентом усиления делает систему неустойчивой. Эти эффекты часто наблюдаются на корневом годографе из классической теории управления. Итак, мы видим, что наличие нулей в правой полуплоскости является прямым ограничением на запас регулятора по амплитуде.

Рекомендуемая литература

Учебники

(1) **Feedback Systems: An Introduction for Scientists and Engineers**, by K. J. Aström and R. M. Murray, 2010 [22].
(2) **Feedback Control Theory**, by J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, 2013 [157].
(3) **Multivariable Feedback Control: Analysis and Design**, by S. Skogestad and I. Postlethwaite, 2005 [492].
(4) **A Course in Robust Control Theory: A Convex Approach**, by G. E. Dullerud and F. Paganini, 2000 [165].
(5) **Optimal Control and Estimation**, by R. F. Stengel, 2012 [501].

Статьи и обзоры

(1) **Guaranteed margins for LQG regulators**, by J. C. Doyle, *IEEE Transactions on Automatic Control*, 1978 [155].

Глава 9

Сбалансированные модели, пригодные для управления

Многие представляющие интерес системы имеют очень высокую размерность, из-за чего их трудно охарактеризовать. Высокая размерность также ограничивает робастность регулятора, поскольку велики временные задержки, связанные с вычислениями. Например, дискретизированные уравнения гидродинамики могут иметь миллионы и миллиарды степеней свободы, так что их моделирование обходится очень дорого. Поэтому были предприняты большие усилия для получения моделей пониженного порядка, которые тем не менее улавливают наиболее существенные механизмы и пригодны для управления с обратной связью.

В отличие от моделей пониженного порядка, основанных на собственном ортогональном разложении (см. главы 11 и 12), при котором моды упорядочиваются по содержанию энергии в данных, здесь мы обсудим класс *сбалансированных* моделей пониженного порядка, в которых для упорядочения мод используется другое скалярное произведение, основанное на энергии входа-выхода. Таким образом, отбираются только моды, которые одновременно хорошо управляемы и хорошо наблюдаемы, что делает сбалансированные модели идеальными для приложений автоматического управления. В этой главе мы также опишем близкие процедуры упрощения модели и идентификации системы: в первом случае пользователь начинает с модели высокой верности, а во втором просто имеет доступ к данным измерений.

9.1. Упрощение модели и идентификация системы

Во многих нелинейных системах все же возможно использовать методы линейного управления. Например, в гидродинамике множество случаев успешного применения управления потоком на основе линейной модели [27, 180,

94], скажем затягивание ламинарно-турбулентного перехода в пространственно развивающемся пограничном слое с целью уменьшить приповерхностное трение в турбулентной пристеночной области и стабилизировать течение за открытой каверной. Однако многие линейные подходы к управлению плохо масштабируются на большие пространства состояний, и иногда их реализация в системах управления в реальном масштабе времени, когда время ограничено, обходится слишком дорого. Поэтому необходимо разрабатывать низкоразмерные аппроксимации системы для управления с обратной связью в реальном времени.

Существует два общих подхода к построению моделей пониженного порядка (reduced-order model – ROM). Во-первых, можно начать с системы высокой размерности, скажем дискретизированных уравнений Навье–Стокса, и спроецировать динамику на идентифицированное низкоразмерное подпространство, например применив собственное ортогональное разложение (POD, глава 11) [57, 251] и проекционный метод Галеркина [441, 53]. Есть много вариантов этой процедуры, в т. ч. дискретные эмпирические методы интерполяции (discrete empirical interpolation method – DEIM, раздел 12.5) [127, 419], неполное POD (раздел 12.1) [179], сбалансированное собственное ортогональное разложение (balanced proper orthogonal decomposition – BPOD, раздел 9.2) [554, 458] и другие. Во-вторых, можно собрать данные, полученные в результате моделирования или эксперимента, и найти низкоранговую модель с помощью методов, основанных на анализе данных. Этот подход обычно называется идентификацией системы и часто является предпочтительным при проектировании систем управления в силу относительной простоты реализации. В качестве примеров упомянем разложение по динамическим модам (DMD, раздел 7.2) [472, 456, 535, 317], алгоритм реализации собственной системы (ERA, раздел 9.3) [272, 351], идентификацию наблюдателей с помощью фильтра Калмана (observer-Kalman filter identification – OKID, раздел 9.3) [273, 428, 271], NARMAX [59] и разреженную идентификацию нелинейной динамики (SINDy, раздел 7.3) [95].

После того как линейная модель идентифицирована – с помощью упрощения модели или идентификации системы, – ее можно использовать для проектирования управления на основе модели. Однако на практике может возникнуть ряд проблем, поскольку линейное управление на основе модели может оказаться непригодным для широкого класса систем. Во-первых, моделируемая система может быть существенно нелинейной, в таком случае линейная аппроксимация сумеет уловить только небольшую часть динамических эффектов. Во-вторых, система может испытывать стохастические воздействия, так что линейная модель будет усреднять релевантные флуктуации. Наконец, в результате применения управления к полной системе динамика аттрактора может измениться, так что линеаризованная модель окажется некорректной. Исключением является стабилизация неподвижных точек, когда управление с обратной связью компенсирует нелинейные возмущения и удерживает систему в окрестности неподвижной точки, где линеаризованная модель верна. Также существуют такие методы идентификации системы и упрощения модели, которые нелинейны, учитывают стохастичность и изменяются вместе с аттрактором. Однако эти методы обычно

сложны, и их применение может оказаться ограниченным доступным оборудованием.

9.2. Сбалансированное упрощение модели

Высокая размерность и короткие временные масштабы, характерные для сложных систем, могут стать причиной неприменимости описанных в главе 8 стратегий управления на основе модели к системам реального времени. Кроме того, синтез оптимальных $\mathcal{H}_2$- и $\mathcal{H}_\infty$-регуляторов может оказаться вычислительно нереализуемым, поскольку требуется либо решать многомерное уравнение Риккати, либо выполнять дорогостоящую итеративную оптимизацию. Но, как уже было продемонстрировано в этой книге, даже если размерность объемлющего пространства велика, может существовать несколько доминирующих когерентных структур, характеризующих систему. Модели пониженного порядка предлагают эффективные низкоразмерные представления этих наиболее релевантных механизмов. Затем модели меньшего порядка можно использовать для проектирования эффективных регуляторов, которые применимы в режиме реального времени даже к системам высокой размерности. Альтернатива – разработка регуляторов на основе полноразмерной модели с последующим применением методов упрощения модели непосредственно к полному регулятору [209, 194, 410, 128].

Упрощение модели – это, по существу, сокращение объема данных с учетом того факта, что данные порождаются динамическим процессом. Если динамическая система с входом и выходом линейна и стационарна, то нет недостатка в механизмах упрощения модели, а границы качества управления можно оценить количественно. Рассмотренные здесь методы основаны на сингулярном разложении (SVD, глава 1) [212, 106, 211] и теории минимальной реализации Хо и Калмана [247, 388]. Общая идея – определить иерархическое разложение состояния системы по модам, которое можно усечь, остановившись на некотором порядке модели, и сохранить когерентные структуры, наиболее важные для управления.

Цель упрощения модели

Рассмотрим систему высокой размерности, схематически изображенную на рис. 9.1:

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{9.1a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{9.1b}$$

возникающую, например, в результате пространственной дискретизации УрЧП. Основная цель упрощения модели – найти такое преобразование координат $\mathbf{x} = \mathbf{\Psi}\tilde{\mathbf{x}}$, при котором система $(\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}, \tilde{\mathbf{D}})$ имела бы похожие характеристики входа-выхода:

$$\frac{d}{dt}\tilde{\mathbf{x}} = \tilde{\mathbf{A}}\tilde{\mathbf{x}} + \tilde{\mathbf{B}}\mathbf{u}, \tag{9.2a}$$

$$\mathbf{y} = \tilde{\mathbf{C}}\tilde{\mathbf{x}} + \tilde{\mathbf{D}}\mathbf{u}, \tag{9.2b}$$

в терминах состояния $\tilde{\mathbf{x}} \in \mathbb{R}^r$ пониженной размерности $r \ll n$. Заметим, что $\mathbf{u}$ и $\mathbf{y}$ в уравнениях (9.1) и (9.2) одинаковы, хотя состояния системы различны. В этом разделе мы займемся построением оператора проецирования $\mathbf{\Psi}$.

Рис. 9.1 ❖ Система с входом и выходом. Модель пониженного порядка, ориентированная на управление, улавливает характер передаточной функции, отображающей **u** в **y**

В качестве мотивирующего примера рассмотрим следующую упрощенную модель:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 10^{-10} \end{bmatrix} u; \tag{9.3a}$$

$$y = \begin{bmatrix} 1 & 10^{-10} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{9.4a}$$

В этом случае состояние x_2 слабо управляемое и слабо наблюдаемое. Если просто выбрать $\tilde{x} = x_1$, то получится модель пониженного порядка, которая точно отражает динамику входа-выхода. Хотя в этом крайнем случае выбор $\tilde{x} = x_1$ кажется интуитивно очевидным, многие методы упрощения модели ошибочно предпочли бы состояние $\tilde{x} = x_2$, поскольку оно слабее демпфировано. В данном разделе мы узнаем, как правильно и эффективно находить матрицу преобразования $\mathbf{\Psi}$, которая наилучшим образом улавливает динамику входа-выхода.

Собственное ортогональное разложение [57, 251] из главы 11 дает матрицу преобразования $\mathbf{\Psi}$, столбцами которой являются моды, упорядоченные по содержанию энергии[1]. POD широко использовалось для построения моделей пониженного порядка сложных систем, в т. ч. для управления, и гарантируется, что оно дает оптимальный низкоранговый базис, улавливающий максимальную энергию или дисперсию в наборе данных. Однако может случиться, что самые энергетичные моды почти неуправляемы или ненаблюдае-

[1] Если обучающие данные – поля скоростей, например из дискретизированной гидродинамической системы высокой размерности, то сингулярные значения буквально равны кинетической энергии соответствующей моды. О том, что POD-моды упорядочены по содержанию *энергии*, принято говорить и в других приложениях, хотя технически правильнее было использовать термин *дисперсия*.

мы, а потому неинтересны с точки зрения управления. С другой стороны, направления, соответствующие большинству управляемых и наблюдаемых состояний, могут иметь очень низкую энергию; например, энергия акустических мод, как правило, очень низкая, но они опосредуют доминирующую динамику входа-выхода во многих гидродинамических системах. Можно провести аналогию с рулем судна: на него приходится малая часть общей энергии, но с точки зрения динамики он – важная часть системы управления.

Вместо того чтобы упорядочивать моды по энергии, можно определить иерархию мод, ориентируясь на управляемость и наблюдаемость, и тем самым уловить самую важную информацию о связи между входами и выходами. Эти моды порождают *сбалансированную* модель, в которой управляемости и наблюдаемости состояния приписываются одинаковые веса посредством преобразования координат, делающего грамианы управляемости и наблюдаемости равными и диагональными. Такие модели оказались чрезвычайно успешны, хотя их вычисление традиционными методами для систем высокой размерности запредельно дорого. В этом разделе мы опишем процедуру балансировки, а также современные методы эффективного вычисления сбалансированных моделей. Набор вычислительно эффективных алгоритмов упрощения модели и идентификации системы можно найти в работе [50].

Сбалансированная модель пониженного порядка должна отображать входы в выходы настолько точно, насколько это возможно для модели порядка r. Поэтому важно ввести *операторную норму*, которая будет количественно оценивать близость воздействия (9.1) и (9.2) на заданный набор входов. Как правило, выбирается норма $\|\cdot\|_\infty$ разности передаточных функций $\mathbf{G}(s)$ и $\mathbf{G}_r(s)$ полной системы (9.1) и упрощенной системы (9.2) соответственно. Эта норма определяется следующим образом:

$$\|\mathbf{G}\|_\infty \triangleq \max_\omega \sigma_1 (\mathbf{G}(i\omega)). \tag{9.4}$$

Введение в передаточные функции см. в разделе 8.8. Итак, мы ищем такую модель (9.2) пониженного порядка $r \ll n$, что операторная норма $\|\mathbf{G} - \mathbf{G}_r\|_\infty$ мала.

Замена переменных в системах управления

Задачу построения сбалансированной модели пониженного порядка можно сформулировать как нахождение преобразования координат

$$\mathbf{x} = \mathbf{T}\mathbf{z}, \tag{9.5}$$

которое иерархически упорядочивает состояния $\mathbf{z}$ по их способности улавливать связь между входом и выходом системы. Мы начнем с рассмотрения обратимого преобразования $\mathbf{T} \in \mathbb{R}^{n\times n}$, а затем опишем метод вычисления только первых r столбцов, которые и составят преобразование $\mathbf{\Psi}$ в формуле (9.2). Таким образом, мы сможем сохранить лишь первые r самых управляемых и наблюдаемых состояний, отбросив остальные. Это напоминает замену переменных для перехода к базису собственных векторов в (8.18), только

теперь мы ставим на первое место управляемость и наблюдаемость, а не характеристики динамики.

Подстановка $\mathbf{Tz}$ в (9.1) дает:

$$\frac{d}{dt}\mathbf{Tz} = \mathbf{ATz} + \mathbf{Bu}, \tag{9.6a}$$

$$\mathbf{y} = \mathbf{CTz} + \mathbf{Du}. \tag{9.6b}$$

Умножая (9.6a) на $\mathbf{T}^{-1}$, получаем:

$$\frac{d}{dt}\mathbf{z} = \mathbf{T}^{-1}\mathbf{ATz} + \mathbf{T}^{-1}\mathbf{Bu}, \tag{9.7a}$$

$$\mathbf{y} = \mathbf{CTz} + \mathbf{Du}. \tag{9.7b}$$

Это дает нам следующие преобразованные уравнения:

$$\frac{d}{dt}\mathbf{z} = \hat{\mathbf{A}}\mathbf{z} + \hat{\mathbf{B}}\mathbf{u}, \tag{9.8a}$$

$$\mathbf{y} = \hat{\mathbf{C}}\mathbf{z} + \mathbf{Du}, \tag{9.8b}$$

где $\hat{\mathbf{A}} = \mathbf{T}^{-1}\mathbf{AT}$, $\hat{\mathbf{B}} = \mathbf{T}^{-1}\mathbf{B}$, $\hat{\mathbf{C}} = \mathbf{CT}$. Заметим, что если столбцы $\mathbf{T}$ ортонормированы, то замена координат принимает вид:

$$\frac{d}{dt}\mathbf{z} = \mathbf{T}^{*}\mathbf{ATz} + \mathbf{T}^{*}\mathbf{Bu}, \tag{9.9a}$$

$$\mathbf{y} = \mathbf{CTz} + \mathbf{Du}. \tag{9.9b}$$

Грамианы и преобразования координат

Грамианы управляемости и наблюдаемости порождают скалярные произведения в пространстве состояний, описывающие, в какой мере данное состояние является соответственно управляемым или наблюдаемым. Поэтому грамианы зависят от конкретного выбора системы координат и изменяются при замене координат. В системе координат $\mathbf{z}$, описываемой формулой (9.5), грамиан управляемости принимает вид:

$$\hat{\mathbf{W}}_c = \int_0^\infty e^{\hat{\mathbf{A}}\tau}\hat{\mathbf{B}}\hat{\mathbf{B}}^{*}e^{\hat{\mathbf{A}}^{*}\tau}d\tau \tag{9.10a}$$

$$= \int_0^\infty e^{\mathbf{T}^{-1}\mathbf{AT}\tau}\mathbf{T}^{-1}\mathbf{BB}^{*}\mathbf{T}^{-*}e^{\mathbf{T}^{*}\mathbf{A}^{*}\mathbf{T}^{-*}\tau}d\tau \tag{9.10b}$$

$$= \int_0^\infty \mathbf{T}^{-1}e^{\mathbf{A}\tau}\mathbf{TT}^{-1}\mathbf{BB}^{*}\mathbf{T}^{-*}\mathbf{T}^{*}e^{\mathbf{A}^{*}\tau}\mathbf{T}^{-*}d\tau \tag{9.10c}$$

$$= \mathbf{T}^{-1}\left(\int_0^\infty e^{\mathbf{A}\tau}\mathbf{BB}^{*}e^{\mathbf{A}\tau}d\tau\right)\mathbf{T}^{-*} \tag{9.10d}$$

$$= \mathbf{T}^{-1}\mathbf{W}_c\mathbf{T}^{-*}. \tag{9.10e}$$

Заметим, что здесь мы ввели обозначение $\mathbf{T}^{-*} := (\mathbf{T}^{-1})^{*} = (\mathbf{T}^{*})^{-1}$. Грамиан наблюдаемости преобразуется аналогично:

$$\hat{\mathbf{W}}_o = \mathbf{T}^* \mathbf{W}_o \mathbf{T}. \tag{9.11}$$

Доказательство оставляем в качестве упражнения для читателя. Оба грамиана преобразуются как тензоры (т. е. в терминах матрицы преобразования $\mathbf{T}$ и транспонированной к ней, а не $\mathbf{T}$ и обратной к ней), это согласуется с тем, что они индуцируют скалярное произведение в пространстве состояний.

Простое масштабирование

Этот пример, заимствованный из работы Мура 1981 года [388], демонстрирует возможность балансировки системы путем замены координат. Рассмотрим систему

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -10 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 10^{-3} \\ 10^{3} \end{bmatrix} u; \tag{9.12a}$$

$$y = \begin{bmatrix} 10^{3} & 10^{-3} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{9.12b}$$

В этом примере первое состояние x_1 слабо управляемо, а второе слабо наблюдаемо. Однако после замены координат $z_1 = 10^3 x_1$ и $z_2 = 10^{-3} x_2$ система становится сбалансированной:

$$\frac{d}{dt}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -10 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u; \tag{9.13a}$$

$$y = \begin{bmatrix} 1 & 1 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}. \tag{9.13b}$$

Здесь замена координат сводится просто к масштабированию состояния $\mathbf{x}$. Например, может быть, что первое состояние измерялось в миллиметрах, а второе в километрах. Выразив оба состояния в метрах, мы сбалансировали динамику, т. е. грамианы управляемости и наблюдаемости стали равными и диагональными.

Балансирующие преобразования

Теперь мы готовы вывести балансирующее преобразование координат $\mathbf{T}$, которое делает грамианы управляемости и наблюдаемости равными и диагональными:

$$\hat{\mathbf{W}}_c = \hat{\mathbf{W}}_o = \mathbf{\Sigma}. \tag{9.14}$$

Сначала рассмотрим произведение грамианов (9.10) и (9.11):

$$\hat{\mathbf{W}}_c \hat{\mathbf{W}}_o = \mathbf{T}^{-1} \mathbf{W}_c \mathbf{W}_o \mathbf{T}. \tag{9.15}$$

Подстановка $\hat{\mathbf{W}}_c = \hat{\mathbf{W}}_o = \boldsymbol{\Sigma}$ дает

$$\mathbf{T}^{-1}\mathbf{W}_c\mathbf{W}_o\mathbf{T} = \boldsymbol{\Sigma}^2 \quad \Rightarrow \quad \mathbf{W}_c\mathbf{W}_o\mathbf{T} = \mathbf{T}\boldsymbol{\Sigma}^2. \tag{9.16}$$

Последнее выражение в (9.16) – уравнение спектрального разложения $\mathbf{W}_c\mathbf{W}_o$ – произведения грамианов в исходных координатах. Таким образом, балансирующее преобразование $\mathbf{T}$ связано со спектральным разложением $\mathbf{W}_c\mathbf{W}_o$. Выражение (9.16) имеет место при любом масштабировании собственных векторов, и мы должны выбрать такое масштабирование, которое точно балансирует грамианы. Иными словами, существует много таких преобразований $\mathbf{T}$, при которых $\hat{\mathbf{W}}_c\hat{\mathbf{W}}_o = \boldsymbol{\Sigma}^2$, но по отдельности грамианы не равны друг другу (например, диагональные грамианы $\hat{\mathbf{W}}_c = \boldsymbol{\Sigma}_c$ и $\hat{\mathbf{W}}_o = \boldsymbol{\Sigma}_o$ удовлетворяют (9.16), если $\boldsymbol{\Sigma}_c\boldsymbol{\Sigma}_o = \boldsymbol{\Sigma}^2$).

Для упрощения обозначений введем в рассмотрение матрицу $\mathbf{S} = \mathbf{T}^{-1}$.

Масштабирование собственных векторов в качестве балансирующего преобразования

Чтобы найти правильный масштаб собственных векторов, при котором $\hat{\mathbf{W}}_c = \hat{\mathbf{W}}_o = \boldsymbol{\Sigma}$, сначала рассмотрим более простой случай балансировки первого диагонального элемента $\boldsymbol{\Sigma}$. Обозначим $\boldsymbol{\xi}_u$ немасштабированный первый столбец $\mathbf{T}$, а $\boldsymbol{\eta}_u$ – немасштабированную первую строку $\mathbf{S} = \mathbf{T}^{-1}$. Тогда

$$\boldsymbol{\eta}_u\mathbf{W}_c\boldsymbol{\eta}_u^* = \sigma_c, \tag{9.17a}$$

$$\boldsymbol{\xi}_u^*\mathbf{W}_o\boldsymbol{\xi}_u = \sigma_o. \tag{9.17b}$$

Таким образом, первый элемент диагонализированного грамиана управляемости равен σ_c, а первый элемент диагонализированного грамиана наблюдаемости равен σ_o. Если умножить собственный вектор $\boldsymbol{\xi}_u$ на σ_s, то обратный собственный вектор $\boldsymbol{\eta}_u$ умножится на σ_s^{-1}. Преобразование, определяемое новыми масштабированными собственными векторами $\boldsymbol{\xi}_s = \sigma_s\boldsymbol{\xi}_u$ и $\boldsymbol{\eta}_s = \sigma_s^{-1}\boldsymbol{\eta}_u$, дает:

$$\boldsymbol{\eta}_s\mathbf{W}_c\boldsymbol{\eta}_s^* = \sigma_s^{-2}\sigma_c, \tag{9.18a}$$

$$\boldsymbol{\xi}_s^*\mathbf{W}_o\boldsymbol{\xi}_s = \sigma_s^2\sigma_o. \tag{9.18b}$$

Таким образом, чтобы два грамиана были равны, необходимо, чтобы

$$\sigma_s^{-2}\sigma_c = \sigma_s^2\sigma_o \quad \Rightarrow \quad \sigma_s = \left(\frac{\sigma_c}{\sigma_o}\right)^{1/4}. \tag{9.19}$$

Чтобы сбалансировать все диагональные элементы грамианов управляемости и наблюдаемости, сначала рассмотрим преобразование с немасштабированным собственным вектором $\mathbf{T}_u$ (9.16); индекс *u* означает «немасштабированный» (unscaled). В качестве примера воспользуемся стандартным масштабированием, применяемым в большинстве программ: столбцы $\mathbf{T}_u$ должны иметь единичную норму. Тогда оба грамиана становятся диагональными, но необязательно равными:

$$\mathbf{T}_u^{-1}\mathbf{W}_c\mathbf{T}_u^{-*} = \mathbf{\Sigma}_c, \tag{9.20a}$$

$$\mathbf{T}_u^{*}\mathbf{W}_o\mathbf{T}_u = \mathbf{\Sigma}_o. \tag{9.20b}$$

Тогда масштабирование, точно балансирующее эти грамианы, определяется матрицей $\mathbf{\Sigma}_s = \mathbf{\Sigma}_c^{1/4}\mathbf{\Sigma}_o^{-1/4}$. Таким образом, балансирующее преобразование имеет вид:

$$\mathbf{T} = \mathbf{T}_u\mathbf{\Sigma}_s. \tag{9.21}$$

Можно прямыми выкладками убедиться, что это преобразование действительно балансирует грамианы:

$$(\mathbf{T}_u\mathbf{\Sigma}_s)^{-1}\mathbf{W}_c(\mathbf{T}_u\mathbf{\Sigma}_s)^{-*} = \mathbf{\Sigma}_s^{-1}\mathbf{T}_u^{-1}\mathbf{W}_c\mathbf{T}_u^{-*}\mathbf{\Sigma}_s^{-1} = \mathbf{\Sigma}_s^{-1}\mathbf{\Sigma}_c\mathbf{\Sigma}_s^{-1} = \mathbf{\Sigma}_c^{1/2}\mathbf{\Sigma}_o^{1/2}; \tag{9.22a}$$

$$(\mathbf{T}_u\mathbf{\Sigma}_s)^{*}\mathbf{W}_o(\mathbf{T}_u\mathbf{\Sigma}_s) = \mathbf{\Sigma}_s\mathbf{T}_u^{*}\mathbf{W}_o\mathbf{T}_u\mathbf{\Sigma}_s = \mathbf{\Sigma}_s\mathbf{\Sigma}_o\mathbf{\Sigma}_s = \mathbf{\Sigma}_c^{1/2}\mathbf{\Sigma}_o^{1/2}. \tag{9.22b}$$

В этих вычислениях используется тот факт, что диагональные матрицы коммутируют, т. е. $\mathbf{\Sigma}_c\mathbf{\Sigma}_o = \mathbf{\Sigma}_o\mathbf{\Sigma}_c$ и т. д.

Пример балансирующего преобразования и грамианов

Прежде чем переходить к практическим проблемам, связанным с точным и эффективным вычислением балансирующего преобразования, полезно рассмотреть пример.

В MATLAB для вычисления сбалансированной системы и балансирующего преобразования достаточно одной команды:

```
[sysb,g,Ti,T] = balreal(sys); % сбалансировать систему
```

Здесь **T** – преобразование, **Ti** – обратное преобразование, **sysb** – сбалансированная система, а **g** – вектор, содержащий диагональные элементы сбалансированных грамианов.

Следующий пример иллюстрирует сбалансированную реализацию двумерной системы. Сначала мы генерируем систему и вычисляем ее сбалансированную реализацию, а также грамианы обеих систем. Затем грамианы несбалансированной и сбалансированной систем визуализируются, как показано на рис. 9.2.

Листинг 9.1 ❖ Получение сбалансированной реализации

```
A = [-.75 1; -.3 -.75];
B = [2; 1];
C = [1 2];
D = 0;

sys = ss(A,B,C,D);

Wc = gram(sys,'c'); % грамиан управляемости
Wo = gram(sys,'o'); % грамиан наблюдаемости

[sysb,g,Ti,T] = balreal(sys); % сбалансировать систему
```

```
BWc = gram(sysb,'c') % сбалансированные грамианы
BWo = gram(sysb,'o')
```

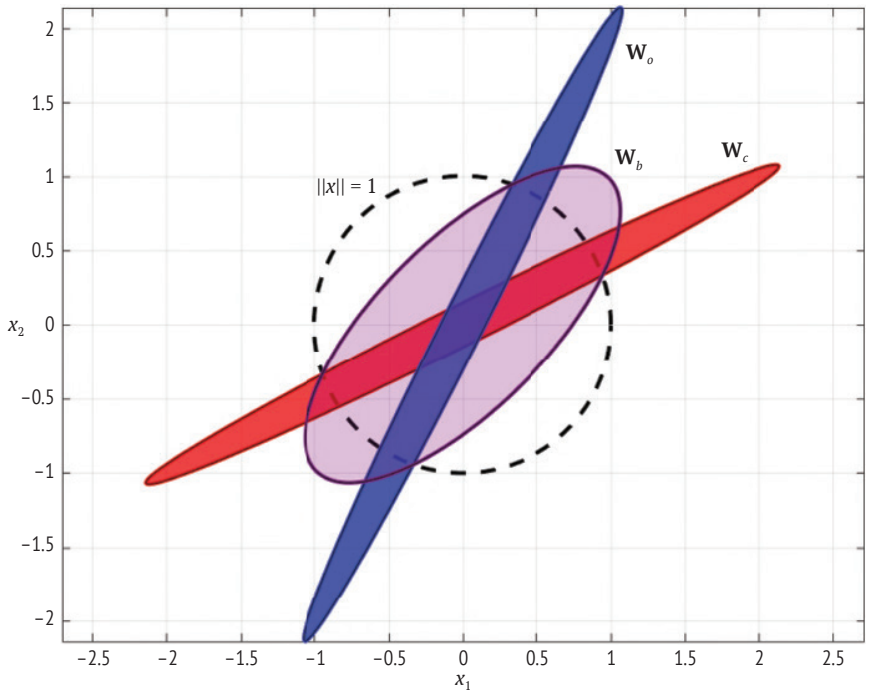

Рис. 9.2 ❖ Иллюстрация балансирующего преобразования грамианов. Достижимое множество при единичном входе показано красным цветом, оно является образом множества ||**x**|| = 1 при отображении $\mathbf{W}_c^{1/2}\mathbf{x}$. Соответствующее наблюдаемое множество показано синим цветом. После применения балансирующего преобразования **T** грамианы становятся равны, они показаны сиреневым цветом

Результирующие сбалансированные грамианы равные, диагональные и упорядочены по убыванию управляемых и наблюдаемых мод:

```
>>BWc =
    1.9439  -0.0000
   -0.0000   0.3207
>>BWo =
    1.9439   0.0000
    0.0000   0.3207
```

Чтобы визуализировать грамианы (рис. 9.2), вспомним, что расстояние, которое система может пройти в направлении **x** при единичном входе, равно $\mathbf{x}^*\mathbf{W}_c\mathbf{x}$. Следовательно, грамиан управляемости можно визуализировать, нарисовав множество $\mathbf{W}_c^{1/2}\mathbf{x}$ для **x**, принадлежащего сфере ||**x**|| = 1. Грамиан наблюдаемости визуализируется аналогично.

Из этого примера видно, что управляемые и наблюдаемые направления вполне могут не совпадать. Однако с помощью замены координат можно

найти новое направление, которое будет и управляемым, и наблюдаемым. Затем можно представить систему в этом одномерном подпространстве, сохранив значительную часть энергии входа-выхода. Если бы красный и синий грамианы были в точности перпендикулярны, т. е. наиболее управляемое направление было бы наименее наблюдаемым, и наоборот, то сбалансированный грамиан изображался бы окружностью. В этом случае не существует предпочтительного направления в пространстве состояний, и оба одинаково важны для связи входа и выхода.

Вместо того чтобы использовать команду **balreal**, можно вручную построить балансирующее преобразование по спектральному разложению $\mathbf{W}_c\mathbf{W}_o$, как описано выше. Соответствующий код имеется в сети.

Сбалансирование усечение

Мы показали, что можно определить такую замену координат, что грамианы управляемости и наблюдаемости станут равными и диагональными. Более того, новые координаты можно ранжировать в терминах совместной управляемости и наблюдаемости. Также можно отбросить часть координат, оставив только наиболее управляемые и наблюдаемые направления, и таким образом получить модель пониженного порядка, улавливающую основные черты динамики входа-выхода.

Зная новые координаты $\mathbf{z} = \mathbf{T}^{-1}\mathbf{x} \in \mathbb{R}^n$, можно определить состояние пониженного порядка $\tilde{\mathbf{x}} \in \mathbb{R}_r$

$$\mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_r \\ z_{r+1} \\ \vdots \\ z_n \end{bmatrix} \Bigg\} \tilde{\mathbf{x}} \tag{9.23}$$

в терминах первых r самых управляемых и наблюдаемых направлений. Разделим балансирующее преобразование $\mathbf{T}$ и обратное ему $\mathbf{S} = \mathbf{T}^{-1}$ на первые r сохраняемых мод и последние $n - r$ отбрасываемых:

$$\mathbf{T} = \begin{bmatrix} \boldsymbol{\Psi} & \mathbf{T}_t \end{bmatrix}, \qquad \mathbf{S} = \begin{bmatrix} \boldsymbol{\Phi}^* \\ \mathbf{S}_t \end{bmatrix}. \tag{9.24}$$

Тогда преобразованную динамику (9.7) можно записать в виде:

$$\frac{d}{dt}\begin{bmatrix} \tilde{\mathbf{x}} \\ \mathbf{z}_t \end{bmatrix} = \left[\begin{array}{c|c} \boldsymbol{\Phi}^*\mathbf{A}\boldsymbol{\Psi} & \boldsymbol{\Phi}^*\mathbf{A}\mathbf{T}_t \\ \hline \mathbf{S}_t\mathbf{A}\boldsymbol{\Psi} & \mathbf{S}_t\mathbf{A}\mathbf{T}_t \end{array}\right]\begin{bmatrix} \tilde{\mathbf{x}} \\ \mathbf{z}_t \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Phi}^*\mathbf{B} \\ \mathbf{S}_t\mathbf{B} \end{bmatrix}\mathbf{u}; \tag{9.25a}$$

$$\mathbf{y} = \left[\begin{array}{c|c} \mathbf{C\Psi} & \mathbf{CT}_t \end{array}\right] \left[\begin{array}{c} \tilde{\mathbf{x}} \\ \hline \mathbf{z}_t \end{array}\right] + \mathbf{Du}. \tag{9.25b}$$

В процессе сбалансированного усечения состояние $\mathbf{z}_t$ попросту отбрасывается (т. е. полагается равным нулю), и остаются только уравнения для $\tilde{\mathbf{x}}$:

$$\frac{d}{dt}\tilde{\mathbf{x}} = \mathbf{\Phi}^{*}\mathbf{A\Psi}\tilde{\mathbf{x}} + \mathbf{\Phi}^{*}\mathbf{Bu}, \tag{9.26a}$$

$$\mathbf{y} = \mathbf{C\Psi}\tilde{\mathbf{x}} + \mathbf{Du}. \tag{9.26b}$$

Только первые r столбцов $\mathbf{T}$ и $\mathbf{S}^{*} = \mathbf{T}^{-*}$ необходимы для построения $\mathbf{\Psi}$ и $\mathbf{\Phi}$, поэтому вычислять полное балансирующее преобразование $\mathbf{T}$ необязательно. Заметим, что матрица $\mathbf{\Phi}$ отлична от матрицы DMD-мод из раздела 7.2. Вычисление $\mathbf{\Psi}$ и $\mathbf{\Phi}$ без вычисления $\mathbf{T}$ обсуждается в следующих разделах. Основное преимущество сбалансированного усечения – существование верхней и нижней границ ошибки при усечении заданного порядка.

$$\textbf{Верхняя граница: } \|\mathbf{G} - \mathbf{G}_r\|_{\infty} \leq 2\sum_{j=r+1}^{n} \sigma_j; \tag{9.27a}$$

$$\textbf{нижняя граница: } \|\mathbf{G} - \mathbf{G}_r\|_{\infty} > \sigma_{r+1}, \tag{9.27b}$$

где σ_j – j-й диагональный элемент сбалансированных грамианов. Диагональные элементы $\mathbf{\Sigma}$ называются также *сингулярными значениями Ганкеля*.

Вычисление сбалансированных реализаций

В предыдущем разделе мы показали, что можно найти преобразование координат, которое балансирует управляемость и наблюдаемость грамианов. Однако вычисление этого преобразования нетривиально, поэтому была проделана значительная работа по нахождению точных и эффективных методов. Она была начата в статье Мура 1981 года [388] и продолжена в работах Lall, Marsden, and Glavaški 2002 года [321], Willcox and Peraire 2002 года [554] и Rowley 2005 года [458]. Очень хорошее и полное изложение методов сбалансированной реализации и упрощения модели см. в Antoulas [17].

На практике вычисление грамианов $\mathbf{W}_c$ и $\mathbf{W}_o$ и спектрального разложения произведения $\mathbf{W}_c\mathbf{W}_o$ (9.16) для систем высокой размерности может оказаться слишком дорогим. Вместо этого балансирующее преобразование можно аппроксимировать на основе импульсной характеристики, воспользовавшись сингулярным разложением для эффективного выделения наиболее значимых подпространств.

Сначала мы покажем, что грамианы можно аппроксимировать с помощью матрицы снимка, полученной на основе импульсной характеристики, построенной в результате экспериментов или имитационного моделирования. Затем будет показано, как с помощью этих данных найти балансирующее преобразование.

Эмпирические грамианы

На практике вычисление грамианов из уравнения Ляпунова слишком дорого, его вычислительная сложность составляет $O(n^3)$. Но грамианы можно аппроксимировать, зная измерения полного состояния системы с дискретным временем и сопряженной ей:

$$\textit{прямая}: \mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k, \tag{9.28a}$$

$$\textit{сопряженная}: \mathbf{x}k{+}1 = \mathbf{A}_d^*\mathbf{x}_k + \mathbf{C}_d^*\mathbf{y}_k. \tag{9.28b}$$

(9.28a) – уравнение обновления динамической системы с дискретным временем (8.21), а (9.28b) – сопряженное уравнение. $\mathbf{A}_d$, $\mathbf{B}_d$ и $\mathbf{C}_d$ – матрицы системы с дискретным временем (8.22). Заметим, что сопряженное уравнение, вообще говоря, нефизическое, его необходимо моделировать; таким образом, описываемые ниже методы применяются к аналитическим уравнениям и данным моделирования, а не к экспериментальным данным. Другая формулировка, не зависящая от сопряженных данных и потому обобщаемая на экспериментальные данные, будет приведена в разделе 9.3.

Вычисление импульсной характеристики прямой и сопряженной систем дает следующие матрицы снимков с дискретным временем:

$$\mathcal{C}_d = \begin{bmatrix}\mathbf{B}_d & \mathbf{A}_d\mathbf{B}_d & \cdots & \mathbf{A}_d^{m_c-1}\mathbf{B}_d\end{bmatrix}, \qquad \mathcal{O}_d = \begin{bmatrix}\mathbf{C}_d \\ \mathbf{C}_d\mathbf{A}_d \\ \vdots \\ \mathbf{C}_d\mathbf{A}_d^{m_o-1}\end{bmatrix}. \tag{9.29}$$

Заметим, что если $m_c = n$, то $\mathcal{C}_d$ – матрица управляемости с дискретным временем, а если $m_o = n$, то $\mathcal{O}_d$ – матрица наблюдаемости с дискретным временем; однако обычно мы рассматриваем случай, когда $m_c, m_o \ll n$. Эти матрицы можно также получить посредством дискретизации прямой и сопряженной систем с непрерывным временем с регулярным интервалом Δt.

Теперь можно вычислить эмпирические грамианы и аппроксимировать истинные грамианы, не решая уравнения Ляпунова (8.42) и (8.43):

$$\mathbf{W}_c \approx \mathbf{W}_c^e = \mathcal{C}_d\mathcal{C}_d^*, \tag{9.30a}$$

$$\mathbf{W}_o \approx \mathbf{W}_o^e = \mathcal{O}_d^*\mathcal{O}_d. \tag{9.30b}$$

Эмпирические грамианы по существу представляют собой аппроксимацию интеграла в грамианах с непрерывным временем римановыми суммами. Эта аппроксимация становится точной, когда временной шаг системы с непрерывным временем стремится к нулю, а продолжительность импульсного отклика стремится к бесконечности. На практике снимки импульсной характеристики следует собирать, пока слабодемпфированные переходные процессы не утихнут. Метод эмпирических грамианов весьма эффективен и широко применяется [388, 320, 321, 554, 458]. Отметим, что требуется p импульсных характеристик сопряженной системы, где p – число выходов. Если число выходов велико (например, это измерения полного состояния),

то метод становится вычислительно нереализуемым, так что приходится обращаться к проекции выхода, описанной в следующем разделе.

Сбалансированное POD (BPOD)

Вместо того чтобы вычислять спектральное разложение матрицы $\mathbf{W}_c\mathbf{W}_o$ размера $n\times n$, можно вычислить балансирующее преобразование с помощью сингулярного разложения произведения матриц снимков

$$\mathcal{O}_d\mathcal{C}_d, \tag{9.31}$$

что напоминает метод снимков из раздела 1.3 [490]. Этот подход был реализован в работе Rowley [458].

Сначала определим обобщенную матрицу Ганкеля, равную произведению сопряженной ($\mathcal{O}_d$) и прямой ($\mathcal{C}_d$) матриц снимков (9.29) для системы с дискретным временем:

$$\mathbf{H} = \mathcal{O}_d\mathcal{C}_d = \begin{bmatrix} \mathbf{C}_d \\ \mathbf{C}_d\mathbf{A}_d \\ \vdots \\ \mathbf{C}_d\mathbf{A}_d^{m_o-1} \end{bmatrix} \begin{bmatrix} \mathbf{B}_d & \mathbf{A}_d\mathbf{B}_d & \cdots & \mathbf{A}_d^{m_c-1}\mathbf{B}_d \end{bmatrix} \tag{9.32a}$$

$$= \begin{bmatrix} \mathbf{C}_d\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c-1}\mathbf{B}_d \\ \mathbf{C}_d\mathbf{A}_d\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^2\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c}\mathbf{B}_d \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_d\mathbf{A}_d^{m_o-1}\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^{m_o}\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c+m_o-2}\mathbf{B}_d \end{bmatrix}. \tag{9.32b}$$

Затем выполним сингулярное разложение $\mathbf{H}$:

$$\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* = \begin{bmatrix} \tilde{\mathbf{U}} & \mathbf{U}_t \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{\Sigma}} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_t \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{V}}^* \\ \mathbf{V}_t^* \end{bmatrix} \approx \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*. \tag{9.33}$$

Для получения желаемой модели порядка $r \ll n$ мы оставляем только первые r столбцов $\mathbf{U}$ и $\mathbf{V}$, а также левый верхний блок $\mathbf{\Sigma}$ размера $r\times r$; остальной вклад $\mathbf{U}_t\mathbf{\Sigma}_t\mathbf{V}_t^*$ можно отбросить. Это дает биортогональный набор мод:

$$\textit{прямые моды}: \mathbf{\Psi} = \mathcal{C}_d\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1/2}, \tag{9.34a}$$

$$\textit{сопряженные моды}: \mathbf{\Phi} = \mathcal{O}_d^*\tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}^{-1/2}. \tag{9.34b}$$

Прямые моды $\mathbf{\Psi} \in \mathbb{R}^{n\times r}$ и сопряженные моды $\mathbf{\Phi} \in \mathbb{R}^{n\times r}$ биортогональны, $\mathbf{\Phi}^*\mathbf{\Psi} = \mathbf{I}^{r\times r}$, и Роули [458] показал, что они определяют замену координат, балансирующую усеченные эмпирические грамианы. Таким образом, $\mathbf{\Psi}$ аппроксимирует первые r столбцов полного балансирующего преобразования $\mathbf{T}$ размера $n\times n$, а $\mathbf{\Phi}^*$ – первые r строк обратного балансирующего преобразования $\mathbf{S} = \mathbf{T}^{-1}$ размера $n\times n$.

Теперь можно спроецировать исходную систему на эти моды, что даст сбалансированную модель пониженного порядка r:

$$\tilde{\mathbf{A}} = \mathbf{\Phi}^{*}\mathbf{A}_d\mathbf{\Psi}, \tag{9.35a}$$

$$\tilde{\mathbf{B}} = \mathbf{\Phi}^{*}\mathbf{B}_d, \tag{9.35b}$$

$$\tilde{\mathbf{C}} = \mathbf{C}_d\mathbf{\Psi}. \tag{9.35c}$$

Динамику упрощенной системы (9.35a) можно вычислить, не имея прямого доступа к $\mathbf{A}_d$. Иногда $\mathbf{A}_d$ может быть очень большой и громоздкой, поэтому можно лишь вычислить действие данной матрицы на входной вектор. Например, во многих современных гидродинамических программах матрица $\mathbf{A}_d$ фактически не представлена, но поскольку она разреженная, можно эффективно реализовать процедуры ее умножения на вектор.

Важно отметить, что в модели пониженного порядка (9.35) время дискретное, так как она основана на эмпирических матрицах снимков системы с дискретным временем. Однако не сложно получить соответствующую систему с непрерывным временем:

```
>>sysD = ss(Atilde,Btilde,Ctilde,D,dt); % дискретное время
>>sysC = d2c(sysD);                     % непрерывное время
```

В этом примере $\mathbf{D}$ одинакова для дискретного и непрерывного времени, а также для моделей полного и пониженного порядков.

Отметим, что к модели BPOD верхняя граница сбалансированного усечения (см. (9.27)) может быть неприменима из-за ошибок при вычислении эмпирических грамианов.

Проекция выхода

Часто при моделировании систем высокой размерности мы предполагаем наличие измерений полного состояния, так что $p = n$ очень велико. Чтобы избежать $p = n$ моделирований сопряженной системы, можно решить сопряженное уравнение для проекции выхода [458]:

$$\mathbf{x}_{k+1} = \mathbf{A}_d^{*}\mathbf{x}_k + \mathbf{C}_d^{*}\tilde{\mathbf{U}}\mathbf{y}, \tag{9.36}$$

где $\tilde{\mathbf{U}}$ – матрица, содержащая первые r сингулярных векторов $\mathcal{C}_d$. Таким образом, мы сначала находим POD-подпространство $\tilde{\mathbf{U}}$ низкой размерности по прямой импульсной характеристике, а затем выполняем моделирование сопряженной импульсной характеристики, производя измерения только этих немногих *коэффициентов POD*. Вообще, если размерность $\mathbf{y}$ высока, но это не измерение полного состояния, то можно использовать POD-подпространство, обученное на измерениях, если известны первые r сингулярных векторов $\tilde{\mathbf{U}}$ матрицы $\mathbf{C}_d\mathcal{C}_d$. Затем можно моделировать сопряженные импульсные характеристики в этих выходных направлениях POD.

Сбор данных и формирование матрицы

Степени m_c и m_o в равенстве (9.32) означают, что данные следует собирать, пока ранг матриц $\mathcal{C}_d$ и $\mathcal{O}^{*}$ не станет полным, после чего можно считать, что произведена выборка из управляемого и наблюдаемого подпространств. Если

сбор данных завершается до затихания переходных процессов, то истинные грамианы аппроксимированы лишь приближенно. Вместо этого можно собирать данные, пока ранг матрицы Ганкеля не станет полным, сбалансировать получившуюся модель и затем усечь ее. Этот более эффективный подход развит в работах [533] и [346].

Матрицы снимков (9.29) строятся по результатам моделирования импульсной характеристики прямой системы (9.28a) и сопряженной системы (9.36). Эти временные ряды затем чередуются с целью формирования матриц снимков.

Историческое замечание

Метод сбалансированного POD, описанный в предыдущем подразделе, разработан в оригинальной работе Мура 1981 года [388], в которой было представлено управляемое данными обобщение теории минимальной реализации Хо и Калмана [247]. До этого минимальные реализации определялись в терминах идеализированных управляемого и наблюдаемого подпространств, а тонкости, связанные со степенью управляемости и наблюдаемости, не принимались во внимание.

В работе Мура сформулирован ряд важнейших идей, перебросивших мостик между теорией и реальностью. Во-первых, он установил связь между методом главных компонент (PCA) и грамианами, показав, что информацию о степенях управляемости и наблюдаемости можно извлечь из данных с помощью SVD. Во-вторых, Мур показал, что существует балансирующее преобразование, которое делает грамианы равными, диагональными и упорядоченными по сбалансированной управляемости (наблюдаемости); он даже разработал алгоритм вычисления этого преобразования. В результате была подготовлена почва для теоретически обоснованного упрощения модели, благодаря которому состояния можно усекать на основе совместной управляемости и наблюдаемости. Далее Мур ввел понятие эмпирического грамиана, хотя и не пользовался этой терминологией. Он также понял, что вычисление $\mathbf{W}_c$ и $\mathbf{W}_o$ напрямую менее точно, чем вычисление SVD эмпирических матриц снимков по прямой и сопряженной системам, и сумел избежать прямого вычисления спектрального разложения $\mathbf{W}_c\mathbf{W}_o$, воспользовавшись этими SVD-преобразованиями. В 2002 году Lall, Marsden и Glavaški [321] обобщили эту теорию на нелинейные системы.

Недостатком подхода Мура было то, что он вычислял все балансирующие преобразования размера $n \times n$, что не годится для систем очень высокой размерности. В 2002 году Willcox и Peraire [554] обобщили его метод на системы высокой размерности, предложив вариант, основанный на разложениях ранга r матриц $\mathbf{W}_c$ и $\mathbf{W}_o$, полученных из матриц снимков прямой и сопряженной систем. Затем можно вычислить спектральное разложение $\mathbf{W}_c\mathbf{W}_o$, пользуясь эффективными программами вычисления собственных значений, даже не выписывая полные матрицы размера $n \times n$. Однако и у этого подхода есть недостаток – требуется столько моделирований сопряженной импульсной характеристики, сколько имеется уравнений, описывающих выход, а когда измеряется полное состояние, таких уравнений может быть очень много.

В 2005 году Роули [458] решил эту проблему с помощью проекции выхода, которую мы обсуждали выше. В этом случае количество моделирований сопряженной импульсной характеристики ограничено числом значимых POD-мод в данных. Он также показал, что можно использовать спектральное разложение произведения $\mathcal{O}_d\mathcal{C}_d$. Произведение $\mathcal{O}_d\mathcal{C}_d$ часто оказывается меньше по размеру, и такие вычисления могут быть более точными.

Интересно отметить, что почти эквивалентная формулировка была предложена двадцатью годами ранее в области идентификации систем. Так называемый алгоритм реализации собственной системы (eigensystem realization algorithm – ERA), разработанный в 1985 году в работе Juang and Pappa [272], позволяет получить эквивалентные сбалансированным модели, не прибегая к сопряженным данным, что делает его полезным для идентификации системы с помощью экспериментов. Эта связь между ERA и BPOD была установлена в 2011 году в работе Ma et al. [351].

Пример сбалансированного упрощения модели

В этом примере мы продемонстрируем вычисление сбалансированного усечения и сбалансированных POD-моделей в системе со случайным пространством $n = 100$ состояний с $q = 2$ входами и $p = 2$ выходами. Сначала сгенерируем систему в MATLAB:

```
q = 2;   % число входов
p = 2;   % число выходов
n = 100; % размерность состояния
sysFull = drss(n,p,q); % случайная система с дискретным временем
```

Затем вычислим сингулярные значения Ганкеля, изображенные на рис. 9.3. Мы видим, что на $r = 10$ мод приходится свыше 90 % энергии входа-выхода.

```
hsvs = hsvd(sysFull); % сингулярные значения Ганкеля
```

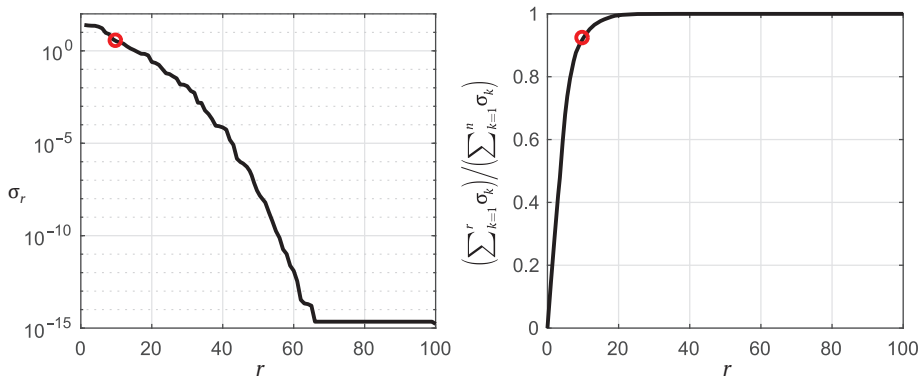

Рис. 9.3 ❖ Сингулярные значения Ганкеля (слева) и суммарная энергия (справа) для системы со случайным пространством состояний с $n = 100$, $p = q = 2$. На первые $r = 10$ сингулярных значений Ганкеля приходится 92.9 % энергии

Теперь построим точное сбалансированное усечение модели порядка $r = 10$:

```
%% Точное сбалансированное усечение
sysBT = balred(sysFull,r); % сбалансированное усечение
```

Для сравнения на рис. 9.4 показаны система полного порядка, сбалансированная усеченная модель и сбалансированная POD-модель. BPOD-модель вычислена программой в листинге 9.2. Видно, что сбалансированная модель точно улавливает доминирующую динамику входа-выхода, хотя оставлено только 10 % мод.

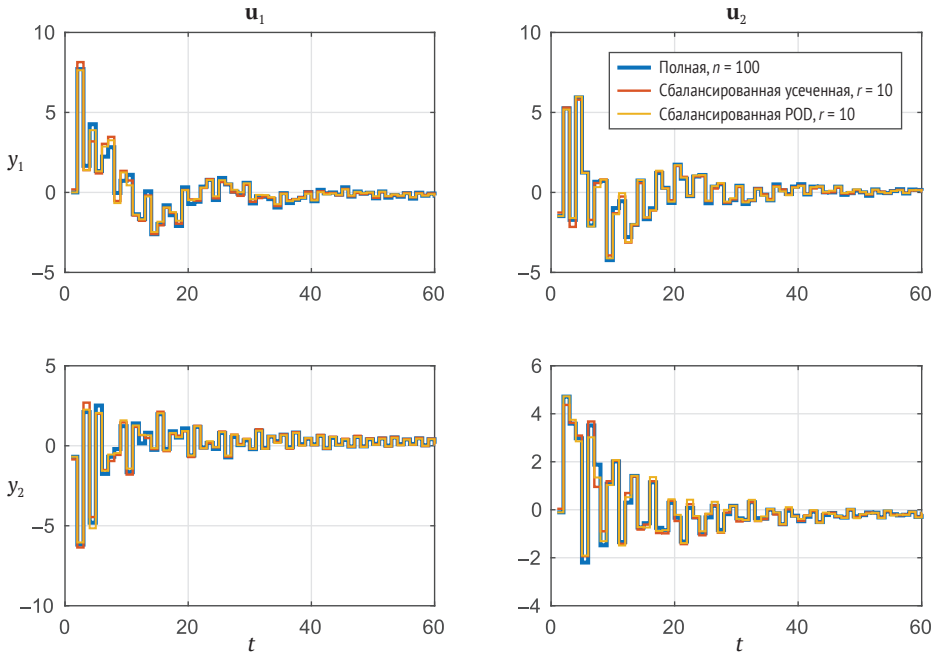

Рис. 9.4 ❖ Импульсная характеристика полной модели с $n = 100$, $p = q = 2$, а также сбалансированная усеченная модель и сбалансированная POD-модель с $r = 10$

Листинг 9.2 ❖ Сбалансированное собственное ортогональное разложение (BPOD)

```
sysBPOD = BPOD(sysFull,sysAdj,r)
[yFull,t,xFull] = impulse(sysFull,0:1:(r*5)+1);
sysAdj = ss(sysFull.A',sysFull.C',sysFull.B',sysFull.D',-1);
[yAdj,t,xAdj] = impulse(sysAdj,0:1:(r*5)+1);
% Не самый быстрый, но достаточный для иллюстрации способ вычисления.
% xAdj и xFull имеют размер m×n×2
HankelOC = []; % вычислить матрицу Ганкеля H = OC
for i=2:size(xAdj,1) % начать с 2, чтобы избежать вычисления матрицы D
Hrow = [];
```

```
    for j=2:size(xFull,1)
        Ystar = permute(squeeze(xAdj(i,:,:)),[2 1]);
        MarkovParameter = Ystar*squeeze(xFull(j,:,:));
        Hrow = [Hrow MarkovParameter];
    end
    HankelOC = [HankelOC; Hrow];
end
[U,Sig,V] = svd(HankelOC);
Xdata = [];
Ydata = [];
for i=2:size(xFull,1) % начать с 2, чтобы избежать вычисления матрицы D
    Xdata = [Xdata squeeze(xFull(i,:,:))];
    Ydata = [Ydata squeeze(xAdj(i,:,:))];
end
Phi = Xdata*V*Sig^(-1/2);
Psi = Ydata*U*Sig^(-1/2);
Ar = Psi(:,1:r)'*sysFull.a*Phi(:,1:r);
Br = Psi(:,1:r)'*sysFull.b;
Cr = sysFull.c*Phi(:,1:r);
Dr = sysFull.d;
sysBPOD = ss(Ar,Br,Cr,Dr,-1);
```

9.3. Идентификация системы

В отличие от упрощения системы, когда модель системы (**A**, **B**, **C**, **D**) известна, идентификация системы целиком основана на анализе данных. Идентификацию системы можно представлять себе как форму машинного обучения, при которой отображение входа в выход обучается на основе данных и строится представление, обобщающееся на данные, которые отсутствуют в обучающем наборе. Методам идентификации систем посвящена обширная литература [271, 338], и многие из числа наиболее популярных основаны на той или иной форме динамической регрессии, т. е. аппроксимации данных моделями, как, например, в методе DMD из раздела 7.2. В этом разделе мы рассмотрим алгоритм реализации собственной системы (ERA) и идентификацию наблюдателей с помощью фильтра Калмана (OKID). Выбор именно этих алгоритмов обусловлен их связями со сбалансированным упрощением модели [388, 458, 351, 535] и успешным применением к системам высокой размерности, например управлению вибрацией в авиационно-космических конструкциях и управлению потоком с обратной связью [27, 26, 261]. Алгоритмы ERA и OKID применимы также к системам с несколькими входами и несколькими выходами (MIMO). Из других методов упомянем модели авторегрессионного скользящего среднего (autoregressive moving average – ARMA) и авторегрессионного скользящего среднего с экзогенным входом (autoregressive moving average with exogenous inputs – ARMAX) [552, 72], модель нелинейного авторегрессионного скользящего среднего с экзогенным входом (NARMAX) [59] и метод SINDy из раздела 7.3.

Алгоритм реализации собственной системы

Алгоритм реализации собственной системы порождает низкоразмерные модели с линейной связью между входом и выходом по экспериментальным измерениям импульсной характеристики с помощью датчиков. В основе лежит теория «минимальной реализации» Хо и Калмана [247]. Современная теория была разработана для идентификации структурных моделей различных космических аппаратов [272], а в работе Ma et al. [351] показано, что ERA-модели эквивалентны BPOD-моделям[1]. Однако ERA основан исключительно на измерениях импульсной характеристики и не нуждается в априорном знании модели.

Мы рассматриваем систему с дискретным временем, описанную в разделе 8.2:

$$\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k, \tag{9.37a}$$

$$\mathbf{y}_k = \mathbf{C}_d\mathbf{x}_k + \mathbf{D}_d\mathbf{u}_k. \tag{9.37b}$$

Управляющее воздействие $\mathbf{u}$ в форме дельта-функции с дискретным временем

$$\mathbf{u}_k^\delta \triangleq \mathbf{u}^\delta(k\Delta t) = \begin{cases} \mathbf{I}, & k = 0 \\ \mathbf{0}, & k = 1,2,3,\cdots \end{cases} \tag{9.38}$$

порождает дискретный импульсный отклик, измеряемый датчиками $\mathbf{y}$:

$$\mathbf{y}_k^\delta \triangleq \mathbf{y}^\delta(k\Delta t) = \begin{cases} \mathbf{D}_d, & k = 0 \\ \mathbf{C}_d\mathbf{A}_d^{k-1}\mathbf{B}_d, & k = 1,2,3,\cdots \end{cases}. \tag{9.39}$$

В ходе эксперимента или моделирования обычно измеряется q импульсных характеристик, по одной для каждого из q входных каналов. Выходные характеристики собираются для каждого импульсного входного сигнала, а выходной вектор, полученный в ответ на j-й импульсный вход на k-м временном шаге, является j-м столбцом матрицы $\mathbf{y}_k^\delta$. Таким образом, каждая $\mathbf{y}_k^\delta$ – матрица $\mathbf{CA}^{k-1}\mathbf{B}$ размера $p \times q$. Отметим, что матрицы системы ($\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$) в действительности не существуют, потому что метод, описываемый в следующем разделе, целиком управляется данными.

Матрица Ганкеля $\mathbf{H}$ (9.32) образована путем объединения сдвинутых временных рядов, содержащих измерения импульсной характеристики, как в методе HAVOK из раздела 7.5:

$$\mathbf{H} = \begin{bmatrix} \mathbf{y}_1^\delta & \mathbf{y}_2^\delta & \cdots & \mathbf{y}_{m_c}^\delta \\ \mathbf{y}_2^\delta & \mathbf{y}_3^\delta & \cdots & \mathbf{y}_{m_c+1}^\delta \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_{m_0}^\delta & \mathbf{y}_{m_0+1}^\delta & \cdots & \mathbf{y}_{m_c+m_0-1}^\delta \end{bmatrix} \tag{9.40a}$$

[1] И BPOD-, и ERA-модели балансируют эмпирические грамианы и осуществляют приближенное сбалансированное усечение [388] систем высокой размерности при наличии достаточного количества данных.

$$= \begin{bmatrix} \mathbf{C}_d\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c-1}\mathbf{B}_d \\ \mathbf{C}_d\mathbf{A}_d\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^{2}\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c}\mathbf{B}_d \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_d\mathbf{A}_d^{m_o-1}\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^{m_o}\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c+m_o-2}\mathbf{B}_d \end{bmatrix}. \tag{9.40b}$$

Матрицу $\mathbf{H}$ можно построить только лишь по измерениям $\mathbf{y}^{\delta}$, не строя $\mathcal{O}_d$ и $\mathcal{C}_d$ по отдельности. Таким образом, доступ к сопряженным уравнениям нам не нужен.

Применение SVD к матрице Ганкеля дает доминирующие временные паттерны во временных рядах:

$$\mathbf{H} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{*} = \begin{bmatrix} \tilde{\mathbf{U}} & \mathbf{U}_t \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\Sigma}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_t \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{V}}^{*} \\ \mathbf{V}_t^{*} \end{bmatrix} \approx \tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{V}}^{*}. \tag{9.41}$$

Малые сингулярные значения в $\boldsymbol{\Sigma}_t$ отбрасываются, а в $\tilde{\boldsymbol{\Sigma}}$ сохраняются только первые r сингулярных значений. Столбцы $\tilde{\mathbf{U}}$ и $\tilde{\mathbf{V}}$ определяют *собственные* запаздывающие координаты.

До сих пор алгоритм ERA был очень похож на процедуру BPOD из раздела 9.2. Однако нам не требуется прямой доступ к $\mathcal{O}_d$, $\mathcal{C}_d$ и матрицам системы ($\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$) для построения прямого или сопряженного балансирующего преобразования. Вместо этого мы используем экспериментальные измерения импульсной характеристики, и этого достаточно также для построения второй, сдвинутой, матрицы Ганкеля $\mathbf{H}'$:

$$\mathbf{H}' = \begin{bmatrix} \mathbf{y}_2 & \mathbf{y}_3^{\delta} & \cdots & \mathbf{y}_{m_c+1}^{\delta} \\ \mathbf{y}_3^{\delta} & \mathbf{y}_4^{\delta} & \cdots & \mathbf{y}_{m_c+2}^{\delta} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_{m_0+1}^{\delta} & \mathbf{y}_{m_0+2}^{\delta} & \cdots & \mathbf{y}_{m_c+m_0}^{\delta} \end{bmatrix} \tag{9.42a}$$

$$= \begin{bmatrix} \mathbf{C}_d\mathbf{A}_d\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^{2}\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c}\mathbf{B}_d \\ \mathbf{C}_d\mathbf{A}_d^{2}\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^{3}\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c+1}\mathbf{B}_d \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_d\mathbf{A}_d^{m_o}\mathbf{B}_d & \mathbf{C}_d\mathbf{A}_d^{m_o+1}\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{A}_d^{m_c+m_o-1}\mathbf{B}_d \end{bmatrix} = \mathcal{O}_d\mathbf{A}\mathcal{C}_d. \tag{9.42b}$$

Зная матрицы $\mathbf{H}$ и $\mathbf{H}'$, мы можем следующим образом построить модель пониженного порядка:

$$\tilde{\mathbf{A}} = \tilde{\boldsymbol{\Sigma}}^{-1/2}\tilde{\mathbf{U}}^{*}\mathbf{H}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1/2}; \tag{9.43a}$$

$$\tilde{\mathbf{B}} = \tilde{\boldsymbol{\Sigma}}^{1/2}\tilde{\mathbf{V}}^{*}\begin{bmatrix} \mathbf{I}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}; \tag{9.43b}$$

$$\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{I}_q & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\tilde{\mathbf{U}}\tilde{\boldsymbol{\Sigma}}^{1/2}. \tag{9.43c}$$

Здесь $\mathbf{I}_p$ – единичная матрица размера $p \times p$, которая выделяет первые p столбцов, и $\mathbf{I}_q$ – единичная матрица размера $q \times q$, которая выделяет первые q строк. Таким образом, мы выражаем динамику входа-выхода в терминах упрощенной системы с низкоразмерным состоянием $\tilde{\mathbf{x}} \in \mathbb{R}^r$:

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k + \tilde{\mathbf{B}}\mathbf{u}, \tag{9.44a}$$

$$\mathbf{y} = \tilde{\mathbf{C}}\tilde{\mathbf{x}}_k. \tag{9.44b}$$

$\mathbf{H}$ и $\mathbf{H}'$ построены по данным импульсной характеристики, полученным в результате моделирования или экспериментов, при этом не пришлось хранить прямой и обратный снимки, как в других методах сбалансированного упрощения модели. Однако если доступны снимки полного состояния, как, например, при регистрации полей скорости в процессе моделирования или экспериментальной анемометрии по изображениям частиц, то можно построить прямые моды. Снимки полного состояния образуют матрицу $\mathcal{C}_d$, а моды строятся следующим образом:

$$\mathbf{\Psi} = \mathcal{C}_d \tilde{\mathbf{V}} \tilde{\mathbf{\Sigma}}^{-1/2}. \tag{9.45}$$

Затем эти моды можно использовать для аппроксимации полного состояния системы высокой размерности по низкоразмерной модели (9.44):

$$\mathbf{x} \approx \mathbf{\Psi}\tilde{\mathbf{x}}. \tag{9.46}$$

Если при построении матрицы Ганкеля $\mathbf{H}$ собрано достаточно данных, то метод ERA балансирует эмпирические грамианы управляемости и наблюдаемости $\mathcal{O}_d\mathcal{O}_d^*$ и $\mathcal{C}_d^*\mathcal{C}_d$. Но если данных собрано мало, так что слабо демпфированные переходные процессы не успели утихнуть, то ERA сбалансирует систему лишь приближенно. Но можно вместо этого собрать ровно столько данных, чтобы матрица Ганкеля $\mathbf{H}$ имела полный численный ранг (т. е. оставшиеся сингулярные значения меньше порога погрешности), и вычислить ERA-модель. Получающаяся ERA-модель обычно имеет относительно низкий порядок, определяемый численным рангом подпространств управляемости и наблюдаемости. Затем к этой меньшей модели можно применить точное сбалансированное усечение, как предлагается в работах [533] и [346].

Программа вычисления ERA-модели приведена в листинге 9.3.

Листинг 9.3 ❖ Алгоритм реализации собственной системы

```
function [Ar,Br,Cr,Dr,HSVs] = ERA(YY,m,n,nin,nout,r)
 for i=1:nout
     for j=1:nin
         Dr(i,j) = YY(i,j,1);
         Y(i,j,:) = YY(i,j,2:end);
     end
 end
% Yss = Y(1,1,end);
% Y = Y-Yss;
% Y(i,j,k)::
% i относится к i-му входу
```

```
% j относится к j-му входу
% k относится к k-му временному шагу
% nin,nout – число входов и выходов
% m,n – размеры матрицы Ганкеля
% r – порядок упрощенной модели

assert(length(Y(:,1,1))==nout);
assert(length(Y(1,:,1))==nin);
assert(length(Y(1,1,:))>=m+n);

for i=1:m
    for j=1:n
        for Q=1:nout
            for P=1:nin
                H(nout*i-nout+Q,nin*j-nin+P) = Y(Q,P,i+j-1);
                H2(nout*i-nout+Q,nin*j-nin+P) = Y(Q,P,i+j);
            end
        end
    end
end

[U,S,V] = svd(H,'econ');
Sigma = S(1:r,1:r);
Ur = U(:,1:r);
Vr = V(:,1:r);
Ar = Sigma^(-.5)*Ur'*H2*Vr*Sigma^(-.5);
Br = Sigma^(-.5)*Ur'*H(:,1:nin);
Cr = H(1:nout,:)*Vr*Sigma^(-.5);
HSVs = diag(S);
```

Идентификация наблюдателей с помощью фильтра Калмана

Алгоритм OKID был разработан с целью дополнить ERA в случае слабо демпфированных экспериментальных систем с шумом [273]. На практике постановка изолированных экспериментов по измерению импульсной характеристики сопряжена с трудностями, а из-за шума измерений результаты могут быть искажены. Кроме того, если временные масштабы сильно разнятся, то для использования ERA необходимо собрать огромный объем данных. В этом разделе ставится общая задача аппроксимации импульсной характеристики по произвольным входным и выходным данным. Как правило, применяется следующая общая процедура идентификации моделей пониженного порядка:

1. Собрать выходные данные, соответствующие псевдослучайным входам.
2. Эта информация обрабатывается алгоритмом OKID, который на выходе выдает линейную импульсную характеристику, очищенную от шума.
3. Импульсная характеристика обрабатывается алгоритмом ERA, который строит систему с пространством состояний пониженного порядка.

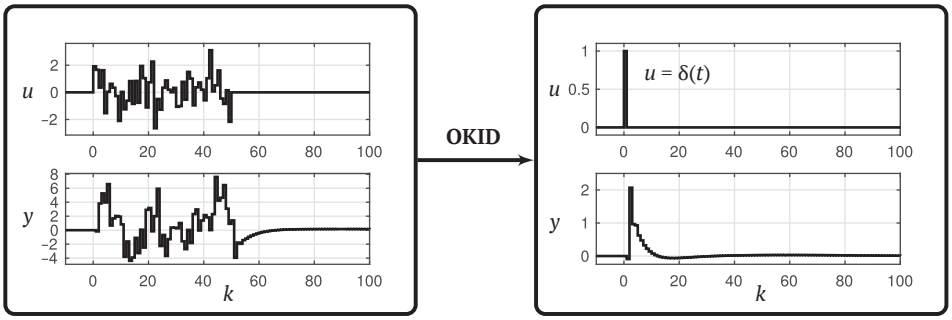

Рис. 9.5 ❖ Схема процедуры OKID.
На выходе OKID получается импульсная характеристика,
которую можно использовать для идентификации системы методом ERA

Отклик $\mathbf{y}_k$ на входной сигнал общего вида $\mathbf{u}_k$ при нулевом начальном условии $\mathbf{x}_0 = \mathbf{0}$ вычисляется по формулам:

$$\mathbf{y}_0 = \mathbf{D}_d\mathbf{u}_0; \tag{9.47a}$$

$$\mathbf{y}_1 = \mathbf{C}_d\mathbf{B}_d\mathbf{u}_0 + \mathbf{D}_d\mathbf{u}_1; \tag{9.47b}$$

$$\mathbf{y}_2 = \mathbf{C}_d\mathbf{A}_d\mathbf{B}_d\mathbf{u}_0 + \mathbf{C}_d\mathbf{B}_d\mathbf{u}_1 + \mathbf{D}_d\mathbf{u}_2; \tag{9.47c}$$

...

$$\mathbf{y}_k = \mathbf{C}_d\mathbf{A}_d^{k-1}\mathbf{B}_d\mathbf{u}_0 + \mathbf{C}_d\mathbf{A}_d^{k-2}\mathbf{B}_d\mathbf{u}_1 + \cdots + \mathbf{C}_d\mathbf{B}_d\mathbf{u}_{k-1} + \mathbf{D}_d\mathbf{u}_k. \tag{9.47d}$$

Заметим, что в выражении для $\mathbf{y}_0$ нет члена $\mathbf{C}$ вследствие нулевого начального условия $\mathbf{x}_0 = \mathbf{0}$. Эту последовательность измерений $\mathbf{y}_k$ можно еще упростить и выразить в терминах измерений импульсной характеристики $\mathbf{y}_k^\delta$:

$$\underbrace{\begin{bmatrix}\mathbf{y}_0 & \mathbf{y}_1 & \cdots & \mathbf{y}_m\end{bmatrix}}_{\mathcal{S}} = \underbrace{\begin{bmatrix}\mathbf{y}_0^\delta & \mathbf{y}_1^\delta & \cdots & \mathbf{y}_m^\delta\end{bmatrix}}_{\mathcal{S}^\delta} \underbrace{\begin{bmatrix}\mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_m \\ \mathbf{0} & \mathbf{u}_0 & \cdots & \mathbf{u}_{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{u}_0\end{bmatrix}}_{\mathcal{B}}. \tag{9.48}$$

Часто возможно обратить матрицу входов $\mathcal{B}$ и найти марковские параметры $\mathcal{S}^\delta$. Однако может быть и так, что $\mathcal{B}$ необратима или обращение плохо обусловлено. Кроме того, для слабо демпфированных систем матрица $\mathcal{B}$ велика, поэтому обращение оказывается вычислительно дорогостоящим. Наконец, если просто обратить $\mathcal{B}$ ради нахождения марковских параметров, то шум отфильтровывается неоптимально.

Метод OKID решает все эти проблемы. Вместо исходной системы с дискретным временем мы вводим в рассмотрение систему с оптимальным наблюдателем:

$$\hat{\mathbf{x}}_{k+1} = \mathbf{A}_d\hat{\mathbf{x}}_k + \mathbf{K}_f(\mathbf{y}_k - \hat{\mathbf{y}}_k) + \mathbf{B}_d\mathbf{u}_k; \tag{9.49a}$$

$$\hat{\mathbf{y}}_k = \mathbf{C}_d\hat{\mathbf{x}}_k + \mathbf{D}_d\mathbf{u}_k, \tag{9.49b}$$

которую можно переписать в виде:

$$\hat{\mathbf{x}}_{k+1} = \underbrace{(\mathbf{A}_d - \mathbf{K}_f\mathbf{C}_d)}_{\bar{\mathbf{A}}_d}\hat{\mathbf{x}}_k + \underbrace{\begin{bmatrix}\mathbf{B}_d - \mathbf{K}_f\mathbf{D}_d, & \mathbf{K}_f\end{bmatrix}}_{\bar{\mathbf{B}}_d}\begin{bmatrix}\mathbf{u}_k\\ \mathbf{y}_k\end{bmatrix}. \tag{9.50}$$

Напомним, что если система наблюдаемая, то полюсы матрицы $\mathbf{A}_d - \mathbf{K}_f\mathbf{C}_d$ можно разместить там, где нам нужно. Однако существуют *оптимальные* точки размещения полюсов, зависящие от степени зашумленности измерений, амплитуды шума и неопределенности модели. Эти точки находятся с помощью *фильтра Калмана* (см. раздел 8.5). Теперь мы можем найти *марковские параметры наблюдателя* $\bar{\boldsymbol{\mathcal{S}}}^\delta$ для системы (9.50) в терминах результатов измерения входов и выходов, следуя описанному ниже алгоритму, взятому из работы [273]:

1. Задать количество подлежащих идентификации марковских параметров наблюдателя l.
2. Построить матрицы данных $\bar{\boldsymbol{\mathcal{S}}}^\delta$ в виде:

$$\boldsymbol{\mathcal{S}} = [\mathbf{y}_0 \quad \mathbf{y}_1 \quad \cdots \quad \mathbf{y}_l \quad \cdots \quad \mathbf{y}_m]; \tag{9.51}$$

$$\boldsymbol{\mathcal{V}} = \begin{bmatrix}\mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_l & \cdots & \mathbf{u}_m\\ \mathbf{0} & \mathbf{v}_0 & \cdots & \mathbf{v}_{l-1} & \cdots & \mathbf{v}_{m-1}\\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots\\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{v}_0 & \cdots & \mathbf{v}_{m-l}\end{bmatrix}, \tag{9.52}$$

   где $\mathbf{v}_i = [\mathbf{u}_i^T \quad \mathbf{y}_i^T]_T$.
   Матрица $\mathcal{V}$ отличается от $\mathcal{B}$ добавленными входами $\mathbf{y}_i$. Таким образом, мы работаем с системой, дополненной фильтром Калмана. И теперь мы ищем марковские параметры наблюдателя для *дополненной* системы $\bar{\boldsymbol{\mathcal{S}}}^\delta$ из уравнения $\mathcal{S} = \bar{\boldsymbol{\mathcal{S}}}^\delta\mathcal{V}$. Мы сможем найти эти марковские параметры наблюдателя по данным, а затем получить импульсную характеристику (марковские параметры) исходной системы.
3. Идентифицировать матрицу $\bar{\boldsymbol{\mathcal{S}}}^\delta$ марковских параметров наблюдателя, решив уравнение $\mathcal{S} = \bar{\boldsymbol{\mathcal{S}}}^\delta\mathcal{V}$ относительно $\bar{\boldsymbol{\mathcal{S}}}^\delta$, для чего нужно найти правую псевдообратную матрицу для $\mathcal{V}$ (т. е. произвести сингулярное разложение).
4. Восстановить марковские параметры системы $\mathcal{S}^\delta$ по марковским параметрам наблюдателя $\bar{\boldsymbol{\mathcal{S}}}^\delta$:
   (a) упорядочить марковские параметры наблюдателя $\bar{\boldsymbol{\mathcal{S}}}^\delta$:

$$\bar{\boldsymbol{\mathcal{S}}}^\delta{}_0 = \mathbf{D}; \tag{9.53}$$

$$\bar{\boldsymbol{\mathcal{S}}}^\delta_k = [(\bar{\boldsymbol{\mathcal{S}}}^\delta)^{(1)}_k \quad (\bar{\boldsymbol{\mathcal{S}}}^\delta)^{(2)}_k] \quad \text{при } k \geq 1, \tag{9.54}$$

   где $(\bar{\boldsymbol{\mathcal{S}}}^\delta)^{(1)}_k \in \mathbb{R}^{q\times p}$, $(\bar{\boldsymbol{\mathcal{S}}}^\delta)^{(2)}_k \in \mathbb{R}^{q\times q}$ и $\mathbf{y}^\delta_0 = \bar{\boldsymbol{\mathcal{S}}}^\delta_0 = \mathbf{D}$;
   (b) реконструировать марковские параметры системы:

$$\mathbf{y}^\delta_k = (\bar{\boldsymbol{\mathcal{S}}}^\delta)^{(1)}_k + \sum_{i=1}^{k}(\bar{\boldsymbol{\mathcal{S}}}^\delta)^{(2)}_i\mathbf{y}^\delta_{k-i} \quad \text{при } k \geq 1. \tag{9.55}$$

Таким образом, метод OKID идентифицирует марковские параметры системы, дополненной асимптотически устойчивым фильтром Калмана. Марковские параметры системы получаются из марковских параметров наблюдателя по формуле (9.55). Эти марковские параметры системы аппроксимируют импульсную характеристику системы и могут прямо подаваться на вход алгоритма ERA. Программа вычисления OKID приведена в листинге 9.4.

Комбинация ERA/OKID широко применялась к самым разным задачам идентификации систем, в т. ч. для идентификации моделей аэроупругих конструкций и гидродинамических систем. Существуют многочисленные расширения методов ERA и OKID. Например, имеются обобщения для систем линейного управления с изменением параметров (linear parameter varying – LPV) и систем, линеаризованных в окрестности предельного цикла.

Листинг 9.4 ❖ Идентификация наблюдателя с помощью фильтра Калмана (OKID)

```
function H = OKID(y,u,r)
% Входы: y (дискретизированный выход), u (дискретизированный вход), r (порядок)
% Выход: H (марковские параметры)

% Шаг 0, проверить формы y,u
p = size(y,1); % p - число выходов
m = size(y,2); % m - число выборок выходов
q = size(u,1); % q - число входов

% Шаг 1, выбрать длину импульса l (в 5 раз больше порядка r)
l = r*5;

% Шаг 2, сформировать y, V, найти марковские параметры наблюдателя Ybar
V = zeros(q + (q+p)*l,m);
for i=1:m
    V(1:q,i) = u(1:q,i);
end
for i=2:l+1
    for j=1:m+1-i
        vtemp = [u(:,j);y(:,j)];
        V(q+(i-2)*(q+p)+1:q+(i-1)*(q+p),i+j-1) = vtemp;
    end
end
Ybar = y*pinv(V,1.e-3);

% Шаг 3, изолировать марковские параметры системы H
D = Ybar(:,1:q); % сквозной член (D) является первым членом
for i=1:l
    Ybar1(1:p,1:q,i) = Ybar(:,q+1+(q+p)*(i-1):q+(q+p)*(i-1)+q);
    Ybar2(1:p,1:q,i) = Ybar(:,q+1+(q+p)*(i-1)+q:q+(q+p)*i);
end
Y(:,:,1) = Ybar1(:,:,1) + Ybar2(:,:,1)*D;
for k=2:l
    Y(:,:,k) = Ybar1(:,:,k) + Ybar2(:,:,k)*D;
    for i=1:k-1
        Y(:,:,k) = Y(:,:,k) + Ybar2(:,:,i)*Y(:,:,k-i);
    end
end
```

```
H(:,:,1) = D;
for k=2:l+1
    H(:,:,k) = Y(:,:,k-1);
end
```

Комбинация ERA и OKID

Ниже мы продемонстрируем совместное применение ERA и OKID к той же модельной системе, что в разделе 9.2. Поскольку ERA дает те же сбалансированные модели, что и BPOD, характеристики упрощенных систем должны быть одинаковыми.

Сначала вычислим импульсную характеристику полной системы и подадим ее на вход ERA:

```
%% Получить импульсную характеристику полной системы
[yFull,t] = impulse(sysFull,0:1:(r*5)+1);
YY = permute(yFull,[2 3 1]); % переупорядочить, так чтобы массив имел
                             % форму p×q×m (по умолчанию m×p×q)

%% Вычислить ERA по импульсной характеристике
mco = floor((length(yFull)-1)/2); % m_c = m_o = (m - 1)/2
[Ar,Br,Cr,Dr,HSVs] = ERA(YY,mco,mco,numInputs,numOutputs,r);
sysERA = ss(Ar,Br,Cr,Dr,-1);
```

Далее, если импульсная характеристика недоступна, то можно возбудить систему случайным входным сигналом и воспользоваться OKID для получения импульсной характеристики. Затем эта импульсная характеристика подается ERA для идентификации модели.

```
%% Моделирование случайного входного сигнала для OKID
uRandom = randn(numInputs,200); % случайный входной сигнал
yRandom = lsim(sysFull,uRandom,1:200)'; % выход

%% Вычислить OKID, затем ERA
H = OKID(yRandom,uRandom,r);
mco = floor((length(H)-1)/2); % m_c = m_o
[Ar,Br,Cr,Dr,HSVs] = ERA(H,mco,mco,numInputs,numOutputs,r);
sysERAOKID = ss(Ar,Br,Cr,Dr,-1);
```

На рис. 9.6 показаны вход и выход, использованные OKID для аппроксимации импульсной характеристики. Импульсные характеристики результирующих систем вычисляются следующим образом:

```
[y1,t1] = impulse(sysFull,0:1:200);
[y2,t2] = impulse(sysERA,0:1:100);
[y3,t3] = impulse(sysERAOKID,0:1:100);
```

Наконец, на рис. 9.7 показаны импульсные характеристики системы. Низкоразмерные модели, полученные применением ERA и ERA/OKID, хорошо согласуются с полной моделью и по качеству сопоставимы с описанными выше BPOD-моделями. Поскольку ERA и BPOD математически эквивалент-

ны, это совпадение не должно вызывать удивления. Однако способность ERA/OKID выделять низкоразмерную модель из случайных входных данных на рис. 9.6 весьма примечательна. Кроме того, в отличие от BPOD, эти методы без труда могут применяться к экспериментальным измерениям, потому что не требуют нефизических сопряженных уравнений.

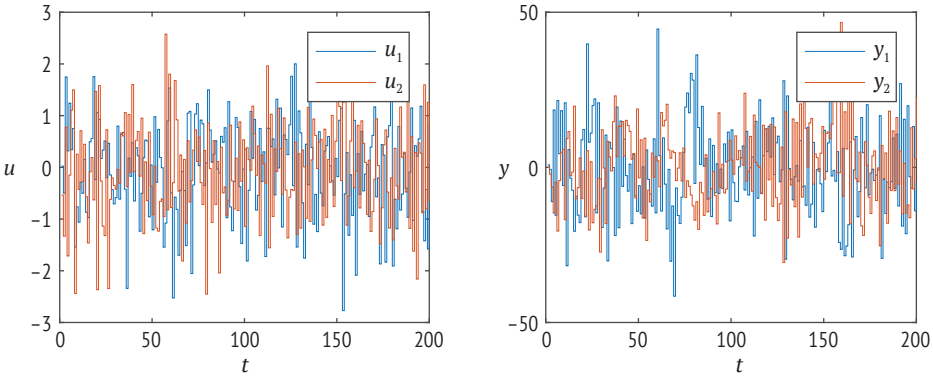

Рис. 9.6 ❖ Входные и выходные данные, используемые OKID

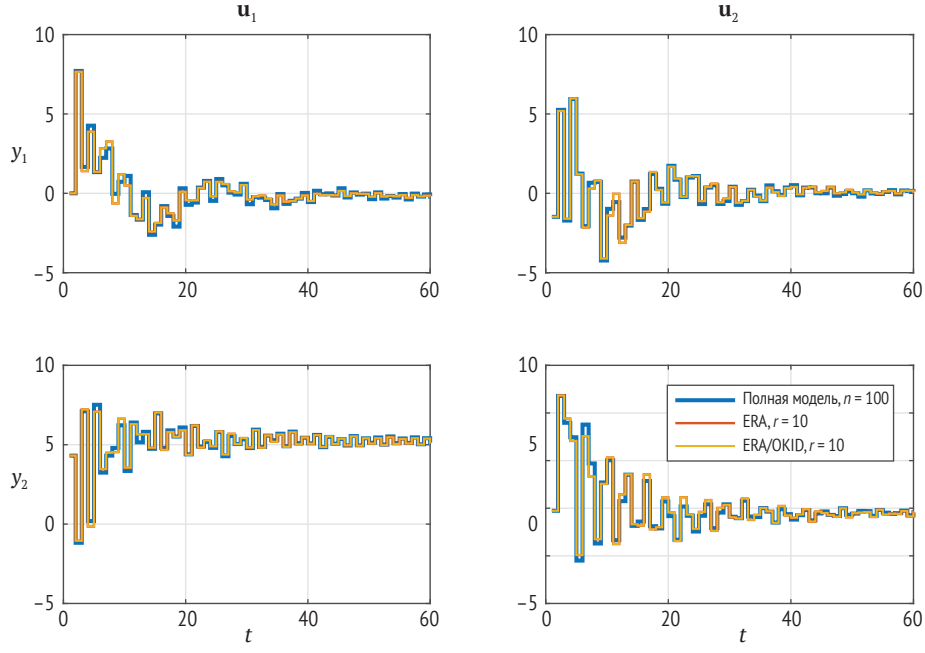

Рис. 9.7 ❖ Импульсная характеристика полной модели с $n = 100$, $p = q = 2$, а также ERA- и ERA/OKID-моделей с $r = 10$

Рекомендуемая литература

Статьи и обзоры

(1) **Principal component analysis in linear systems: Controllability, observability, and model reduction**, by B. C. Moore, *IEEE Transactions on Automatic Control*, 1981 [388].
(2) **Identification of linear parameter varying models**, by B. Bamieh and L. Giarré, *International Journal of Robust and Nonlinear Control*, 2002 [34].
(3) **Balanced model reduction via the proper orthogonal decomposition**, by K.Willcox and J. Peraire, *AIAA Journal*, 2002 [554].
(4) **Model reduction for fluids using balanced proper orthogonal decomposition**, by C. W. Rowley, *International Journal of Bifurcations and Chaos*, 2005 [458].
(5) **An eigensystem realization algorithm for modal parameter identification and model reduction**, by J. N. Juang and R. S. Pappa, *Journal of Guidance, Control,and Dynamics*, 1985 [272].

Глава 10

Управление на основе данных

Как описано в главе 8, проектирование управления часто начинается с построения модели управляемой системы. Есть и исключения из этого правила: безмодельные стратегии адаптивного управления и многие примеры ПИД-управления. Для механических систем не слишком большой размерности можно выписать модель (например, на основе ньютоновского, лагранжева или гамильтонова формализма) и линеаризовать динамику в окрестности неподвижной точки либо периодической орбиты. Однако для современных интересных систем, встречающихся в нейронауках, изучении турбулентности, эпидемиологии, климатологии и финансах, обычно не существует простых моделей, пригодных для проектирования управления. В главе 9 были описаны методы получения ориентированных на управление моделей пониженного порядка для систем высокой размерности на основе данных, но эти подходы ограничены линейными системами. Реальные же системы обычно нелинейны, и цель управления с помощью линейных методов недостижима. Нелинейное управление все же можно поставить как задачу оптимизации с многомерной функцией стоимости с невыпуклой поверхностью, имеющей несколько локальных минимумов. На помощь приходит машинное обучение, представляющее собой постоянно растущий набор методов, которые можно охарактеризовать как нелинейную оптимизацию в многомерном пространстве на основе данных. В этой главе мы опишем перспективные методы, в которых машинное обучение применяется для характеризации и управления существенно нелинейными, многомерными и многомасштабными системами с использованием все более доступных высококачественных данных измерений.

В самом общем смысле методы машинного обучения можно использовать, чтобы 1) охарактеризовать систему для последующего использования совместно с управлением на основе модели или 2) напрямую охарактеризовать закон управления, эффективно взаимодействующий с системой. Это схематически показано на рис. 10.1, где управляемые данными методы можно применить как к *системе*, так и к *регулятору*. Существуют также дополнительные методы для идентификации хороших датчиков и приводов, мы обсуждали их выше в разделе 3.8. В разделе 10.1 мы рассмотрим применение машинного

обучения для идентификации нелинейных моделей входа-выхода для управления, основанных на методах из главы 7. В разделе 10.2 будут рассмотрены методы машинного обучения для прямой идентификации регуляторов по входным и выходным данным. Это быстро развивающаяся область, в которой имеется много мощных методов, например обучение с подкреплением, итеративное обучение управлению и генетические алгоритмы. Мы дадим общий обзор этих методов, затем рассмотрим пример применения генетического алгоритма. Однако важно подчеркнуть: это очень глубокая и широкая область знаний, а любому методу можно было бы посвятить целую книгу. Наконец, в разделе 10.3 мы опишем адаптивную стратегию управления посредством поиска экстремума, которая оптимизирует управляющий сигнал, исходя из того, как система реагирует на возмущения.

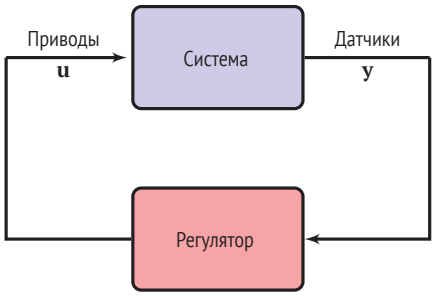

Рис. 10.1 ❖ В стандартной схеме управления, описанной в главе 8, машинное обучение может использоваться для 1) разработки модели системы или 2) для обучения регулятора

10.1. Идентификация нелинейной системы для управления

Управляемое данными моделирование и управление сложными системами переживает революцию, вызванную пришествием больших данных, появлением новых алгоритмов машинного обучения и оптимизации и доступностью современного вычислительного оборудования. Несмотря на все более широкое распространение не описываемых уравнениями и адаптивных методов управления, никуда не делись многочисленные мощные стратегии управления, основанные на модели, например линейное оптимальное управление (см. главу 8) и управление на основе прогнозирующих моделей (model predictive control – MPC) [195, 107]. Все чаще такие стратегии управления на основе моделей дополняются управляемыми данными методами, которые характеризуют динамику входа-выхода интересующей системы, опираясь только на результаты измерения, без привлечения теоретической модели. Эта так называемая *идентификация системы* имеет долгую и богатую историю в теории управления, восходящую к работам Калмана, написанным

несколько десятков лет назад. Однако с распространением управляемых данными методов, в частности описанных в главе 7, возродился интерес к идентификации нелинейных систем.

Цель идентификации системы – найти модель низкого порядка, описывающую динамику входа-выхода, т. е. отображение управляющего воздействия $\mathbf{u}$ на измерения $\mathbf{y}$. Если мы можем измерить полное состояние $\mathbf{x}$ системы, то задача сводится к идентификации динамики $\mathbf{f}$, удовлетворяющей уравнению:

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u}). \tag{10.1}$$

Эту задачу можно сформулировать для дискретного времени, потому что данные обычно собираются в дискретные моменты времени, а законы управления часто реализуются цифровым оборудованием. В этом случае динамика имеет вид:

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k). \tag{10.2}$$

Если динамика приближенно линейна, то мы можем идентифицировать линейную систему

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \tag{10.3}$$

такой подход принят в разложении по динамическим модам и реализуется описанным ниже алгоритмом управления DMDc.

Иногда также имеет смысл идентифицировать набор измерений $\mathbf{y} = \mathbf{g}(\mathbf{x})$, для которого свободная нелинейная динамика представляется линейной:

$$\mathbf{y}_{k+1} = \mathbf{A}_\mathbf{Y}\mathbf{y}_k. \tag{10.4}$$

Такой подход принят в описанном ниже методе управления Купмана. В этом случае для нелинейного оценивания и управления применяется стандартная, описанная во всех учебниках линейная теория управления во внутренних координатах $\mathbf{y}$ [302, 276].

Наконец, нелинейную динамику (10.1) или (10.2) можно идентифицировать непосредственно, воспользовавшись алгоритмом SINDY с управлением. Получающиеся модели можно использовать совместно с управлением на основе прогнозирующих моделей для управления полностью нелинейными системами [277].

DMD с управлением

В работе Proctor et al. [434] алгоритм DMD обобщен и учитывает управляющее воздействие; новый алгоритм называется DMD с управлением (DMDc). Было замечено, что наивное применение DMD к данным, собранным в системе с управляющим воздействием, часто дает неправильную динамику, потому что эффекты, обусловленные внутренней динамикой, путают с эффектами управления. Первоначально DMDc разрабатывался для решения задачи опи-

сания и управления распространением заболевания, когда не имеет смысла прекращать попытки противодействия (например, вакцинацию) только для того, чтобы охарактеризовать динамику в отсутствие управления [435]. Вместо этого если измеряется управляющий сигнал, то можно поставить новую задачу DMD-регрессии, чтобы разделить эффекты внутренней динамики и управления. Впоследствии этот подход был обобщен на случай малых подвыборок или сжатых измерений в работе Bai et al. [30].

Метод DMDc пытается идентифицировать наилучшие линейные операторы $\mathbf{A}$ и $\mathbf{B}$, которые приближенно удовлетворяют следующим уравнениям динамики на данных измерений:

$$\mathbf{x}_{k+1} \approx \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k. \tag{10.5}$$

В дополнение к матрице снимков $\mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_m]$ и сдвинутой по времени матрице снимков $\mathbf{X}' = [\mathbf{x}_2 \quad \mathbf{x}_3 \quad \cdots \quad \mathbf{x}_{m+1}]$ из (7.23) строится еще матрица истории управляющих воздействий:

$$\Upsilon = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \\ | & | & & | \end{bmatrix}. \tag{10.6}$$

Динамику (10.5) можно записать в терминах матриц данных:

$$\mathbf{X}' \approx \mathbf{A}\mathbf{X} + \mathbf{B}\Upsilon. \tag{10.7}$$

Как и в алгоритме DMD (см. раздел 7.2), для получения первых собственных значений и собственных векторов наилучшего линейного оператора $\mathbf{A}$ используется понижение размерности и регрессия. Если матрица управляющих воздействий $\mathbf{B}$ известна, то нетрудно скорректировать воздействие и идентифицировать спектральное разложение $\mathbf{A}$, заменив $\mathbf{X}'$ на $\mathbf{X}' - \mathbf{B}\Upsilon$ в алгоритме DMD:

$$(\mathbf{X}' - \mathbf{B}\Upsilon) \approx \mathbf{A}\mathbf{X}. \tag{10.8}$$

Если же $\mathbf{B}$ неизвестна, то $\mathbf{A}$ и $\mathbf{B}$ нужно идентифицировать одновременно. В таком случае динамика (10.7) записывается в виде:

$$\mathbf{X}' \approx \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \Upsilon \end{bmatrix} = \mathbf{G}\Omega, \tag{10.9}$$

а матрица $\mathbf{G} = [\mathbf{A} \quad \mathbf{B}]$ получается с помощью регрессии методом наименьших квадратов:

$$\mathbf{G} \approx \mathbf{X}'\Omega^{\dagger}. \tag{10.10}$$

Матрица данных $\Omega = [\mathbf{X}^* \quad \Upsilon^*]^*$ обычно имеет высокую размерность, но ее можно аппроксимировать, применив сингулярное разложение:

$$\Omega = \tilde{\mathbf{U}}\tilde{\Sigma}\tilde{\mathbf{V}}^*. \tag{10.11}$$

Матрицу $\tilde{\mathbf{U}}$ нужно разбить на две матрицы, $\tilde{\mathbf{U}} = [\tilde{\mathbf{U}}_1^* \quad \tilde{\mathbf{U}}_2^*]^*$, являющиеся базисами $\mathbf{X}$ и $\mathbf{Y}$. В отличие от алгоритма DMD, $\tilde{\mathbf{U}}$ дает редуцированный базис *пространства входов*, а $\hat{\mathbf{U}}$ из разложения

$$\mathbf{X}' = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^* \tag{10.12}$$

определяет редуцированный базис *пространства выходов*. Тогда можно аппроксимировать $\mathbf{G} = [\mathbf{A} \quad \mathbf{B}]$, спроецировав на этот базис:

$$\tilde{\mathbf{G}} = \hat{\mathbf{U}}^*\mathbf{G}\begin{bmatrix}\hat{\mathbf{U}} \\ \mathbf{I}\end{bmatrix}. \tag{10.13}$$

Результирующие спроецированные матрицы $\tilde{\mathbf{A}}$ и $\tilde{\mathbf{B}}$, составляющие $\tilde{\mathbf{G}}$, равны:

$$\tilde{\mathbf{A}} = \hat{\mathbf{U}}^*\mathbf{A}\hat{\mathbf{U}} = \hat{\mathbf{U}}^*\mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}_1^*\hat{\mathbf{U}}, \tag{10.14a}$$

$$\tilde{\mathbf{B}} = \hat{\mathbf{U}}^*\mathbf{B} = \hat{\mathbf{U}}^*\mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}_2^*. \tag{10.14b}$$

Важнее то, что можно восстановить собственные векторы DMD $\mathbf{\Phi}$ по спектральному разложению $\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}$:

$$\mathbf{\Phi} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\mathbf{U}_1^*\hat{\mathbf{U}}\mathbf{W}. \tag{10.15}$$

Неоднозначность идентификации систем с замкнутым контуром

Для систем, которые активно управляются с помощью обратной связи $\mathbf{u} = \mathbf{K}\mathbf{x}$,

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \tag{10.16a}$$

$$= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{K}\mathbf{x}_k \tag{10.16b}$$

$$= (\mathbf{A} + \mathbf{B}\mathbf{K})\mathbf{x}_k, \tag{10.16c}$$

невозможно разделить динамику $\mathbf{A}$ и управляющее воздействие $\mathbf{B}\mathbf{K}$. В этом случае необходимо добавить возмущения к управляющему сигналу $\mathbf{u}$, чтобы получить дополнительную информацию. В качестве возмущений может выступать белый шум или нерегулярные импульсы, приводящие систему в действие; этот сигнал позволит отделить динамику от сигнала обратной связи.

Нелинейное управление с помощью оператора Купмана

Для нелинейных систем иногда полезно найти с помощью анализа данных такое преобразование координат, при котором динамика оказывается линейной. Это преобразование связано с *внутренними* координатами, определяемыми собственными функциями оператора Купмана (см. раздел 7.4).

Поэтому анализ Купмана применялся для нелинейного оценивания [504, 505] и управления [302, 276, 423].

Можно спроектировать оцениватели и регуляторы непосредственно по моделям DMD или eDMD, и в работе Korda et al. [302] использовалось управление на основе прогнозирующих моделей (MPC) для управления нелинейными системами с eDMD-моделями. MPC демонстрирует на удивление хорошее качество для DMD-моделей, как показано в работе Kaiser et al. [277]. Кроме того, в работе Peitz et al. [423] продемонстрировано применение MPC с целью управления переключением небольшого множества управляющих сигналов для отслеживания опорного значения подъемной силы в нестационарном потоке жидкости; для каждого постоянного значения управляющего сигнала строилась отдельная eDMD-модель. В работах Surana [504] и Surana and Banaszuk [505] также приведены отличные нелинейные оцениватели на основе оператора Купмана. Однако, как было отмечено выше, eDMD-модели могут содержать много паразитных собственных значений и векторов из-за проблем с замкнутостью, связанных с нахождением инвариантного относительно оператора Купмана подпространства. Вместо этого может быть более выгодно идентифицировать несколько релевантных собственных функций Купмана и осуществлять управление прямо в этих координатах [276].

В разделе 7.5 мы описали несколько стратегий аппроксимации собственных функций Купмана $\varphi(\mathbf{x})$, для которых динамика становится линейной:

$$\frac{d}{dt}\varphi(\mathbf{x}) = \lambda\varphi(\mathbf{x}). \tag{10.17}$$

В работе Kaiser et al. [276] уравнение собственных функций Купмана было обобщено на аффинные по управлению нелинейные системы:

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}) + \mathbf{B}\mathbf{u}. \tag{10.18}$$

Для таких систем можно применить правило дифференцирования сложной функции к $\frac{d}{dt}\varphi(\mathbf{x})$:

$$\frac{d}{dt}\varphi(\mathbf{x}) = \nabla\varphi(\mathbf{x}) \cdot (\mathbf{f}(\mathbf{x}) + \mathbf{B}\mathbf{u}) \tag{10.19a}$$

$$= \lambda\varphi(\mathbf{x}) + \nabla\varphi(\mathbf{x}) \cdot \mathbf{B}\mathbf{u}. \tag{10.19b}$$

Заметим, что даже при наличии управляющего воздействия динамика собственных функций Купмана остается линейной, а эффект управления по-прежнему аддитивный. Однако теперь мода $\nabla\varphi(\mathbf{x}) \cdot \mathbf{B}$ может зависеть от состояния. На самом деле управляющее воздействие *обязательно* будет зависеть от состояния, если только производная собственной функции по направлению $\mathbf{B}$ не постоянна. По счастью, существует много хорошо разработанных обобщений стандартной теории линейного управления на основе уравнения Риккати (например, ЛКР, фильтры Калмана и т. д.) для систем, в которых уравнение Риккати *зависит от состояния*.

SINDy с управлением

Хотя было бы очень заманчиво найти внутренние координаты, в которых нелинейная динамика выглядит как линейная, сделать это весь трудно даже для относительно простых систем. Вместо этого имеет смысл непосредственно идентифицировать нелинейную динамическую систему с управляющим сигналом в (10.1) или (10.2) для использования совместно со стандартным управлением на основе модели. Применение метода разреженной идентификации нелинейной динамики (SINDy) (см. раздел 7.3) дает вычислительно эффективные модели, которые можно использовать в режиме реального времени вместе с управлением на основе прогнозирующих моделей [277]. И кстати, эти модели можно идентифицировать, имея сравнительно немного обучающих данных по сравнению с нейронными сетями и другими передовыми методами машинного обучения, поэтому их даже можно охарактеризовать в онлайновом режиме в ответ на резкие изменения динамики системы.

Алгоритм SINDy легко обобщается для учета эффектов управления [100, 277]. В дополнение к сбору измерений снимков состояния $\mathbf{x}$ в матрицу $\mathbf{X}$ управляющие сигналы $\mathbf{u}$ собираются в матрицу Υ из формулы (10.6), как в DMDc. Затем строится дополненная библиотека функций-кандидатов в правой части $\Theta([\mathbf{X} \quad \Upsilon])$:

$$\Theta([\mathbf{X} \quad \Upsilon]) = [\mathbf{1} \quad \mathbf{X} \quad \Upsilon \quad \mathbf{X}^2 \quad \mathbf{X} \otimes \Upsilon \quad \Upsilon^2 \quad \cdots]. \tag{10.20}$$

Здесь $\mathbf{X} \otimes \Upsilon$ обозначает произведение Кронекера матриц $\mathbf{X}$ и Υ, вычисленное по данным.

В алгоритме SINDy с управлением (SINDYc) такая же разреженная регрессия используется для определения наименьшего числа ненулевых членов библиотеки, необходимых для описания наблюдаемой динамики. Как и в DMDc, если система активно управляется с помощью обратной связи $\mathbf{u} = \mathbf{K}(\mathbf{x})$, то невозможно отделить внутреннюю динамику от управляющего воздействия, если только не добавить возмущающий сигнал для получения дополнительной информации.

Пример управления на основе прогнозирующих моделей (MPC)

В этом примере мы воспользуемся алгоритмом SINDYc для идентификации модели управляемых уравнений Лоренца на основе данных, а затем применим к ней управление на основе прогнозирующих моделей (MPC). MPC [107, 195, 438, 391, 447, 439, 196, 326, 173] стало одним из краеугольных камней современного управления процессами и встречается в промышленности повсеместно. MPC применяется для управления существенно нелинейными системами с ограничениями, временными задержками, неминимально-фазовой динамикой и неустойчивостью. В большинстве применений MPC в промышленности используются эмпирические модели, основанные на идентификации линейных систем (см. главу 8), нейронные сети (см. главу 6),

ряды Вольтерра [86, 73] и авторегрессионные модели [6] (например, ARX, ARMA, NARX и NARMAX). Недавно в сочетании с MPC стали использовать глубокое обучение и обучение с подкреплением [330, 570] – результаты впечатляют. Однако для глубокого обучения требуется очень большой объем данных, а результаты не допускают интерпретации. Другое направление исследований – идентифицировать модели для MPC на основе ограниченного объема данных с целью характеризации систем в ответ на резкие изменения.

При управлении на основе прогнозирующих моделей следующее управляющее воздействие определяется путем решения задачи оптимального управления с удаляющимся горизонтом. В частности, управляющий сигнал $\mathbf{u}$ в системе с разомкнутым контуром оптимизируется при удаляющемся временном горизонте $tc = m_c \Delta t$ с целью минимизации стоимости J на некотором горизонте прогнозирования $t_p = m_p \Delta t$. Горизонт управления обычно меньше или равен горизонту прогнозирования, и управляющий сигнал остается постоянным между t_c и t_p. Затем оптимальное управление применяется на одном временном шаге, после чего процедура повторяется, и на каждом временном шаге управление с удаляющимся горизонтом оптимизируется заново. Это приводит к такому закону управления:

$$\mathbf{K}(\mathbf{x}_j) = \mathbf{u}_{j+1}(\mathbf{x}_j), \tag{10.21}$$

где $\mathbf{u}_{j+1}$ – первый временной шаг оптимизированного управляющего воздействия, начинающегося в состоянии $\mathbf{x}_j$. Схематически это показано на рис. 10.2. Так можно оптимизировать высокоспециализированные функции стоимости для систем с нелинейной динамикой, с ограничениями на управляющие воздействия и состояние. Однако вычислительные затраты на повторную оптимизацию на каждом шаге весьма значительны, что налагает ограничения на сложность модели и методы оптимизации. К счастью, быстрый прогресс в вычислительной технике и математических методах уже позволяет применять MPC для нелинейного управления в режиме реального времени.

Применение MPC к управлению уравнениями Лоренца с помощью SINDYc

В следующем примере показано, как с помощью алгоритма SINDYc идентифицировать модель для использования в MPC. В основе своей код такой же, как для SINDY, но при построении библиотеки $\boldsymbol{\Theta}$ управляющий сигнал включен в качестве переменной.

Протестируем идентификацию модели с помощью SINDYc на примере управляемых уравнений Лоренца:

$$\dot{x} = \sigma(y - x) + g(u), \tag{10.22a}$$

$$\dot{y} = x(\rho - z) - y, \tag{10.22b}$$

$$\dot{z} = xy - \beta z. \tag{10.22c}$$

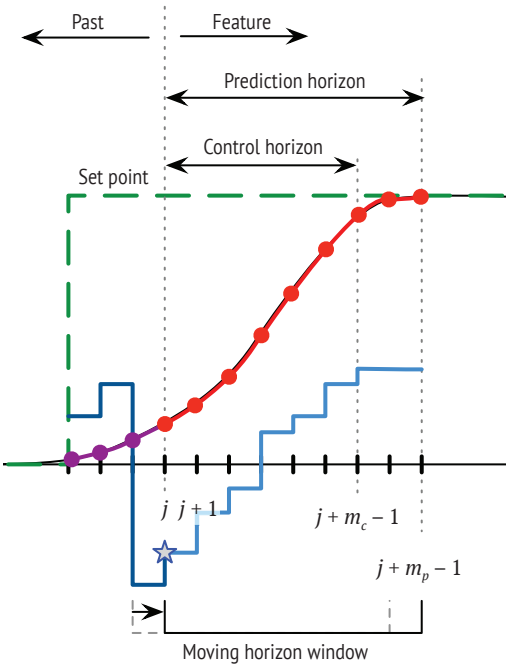

Рис. 10.2 ❖ Схема управления с прогнозирующими моделями, в которой управляющее воздействие **u** итеративно оптимизируется с удаляющимся горизонтом. *Взято из работы Kaiser et al. [277] с разрешения авторов*

В этом примере мы обучаем модель на данных о поведении управляемой системы в течение 20 единиц времени и проверяем ее на данных еще за 20 единиц времени, переключившись на периодический управляющий сигнал $u(t) = 50\sin(10t)$. Алгоритм SINDY не улавливает эффект управления, тогда как SINDYc правильно идентифицирует модель с управлением и предсказывает поведение в ответ на новое воздействие, которое не предъявлялось в составе обучающих данных (см. рис. 10.3).

Наконец, модели системы Лоренца, построенные с помощью SINDYc и с помощью нейронной сети, используются для проектирования регуляторов с прогнозирующей моделью, как показано на рис. 10.4. Оба метода правильно идентифицируют модели, улавливающие динамику, хотя SINDYc требует меньше данных, идентифицирует модель быстрее и более робастен к шуму, чем модель на основе нейронной сети. Дополнительная эффективность и робастность обеспечивается благодаря оптимизации с поощрением разреженности, т. е. регуляризации задачи идентификации модели.

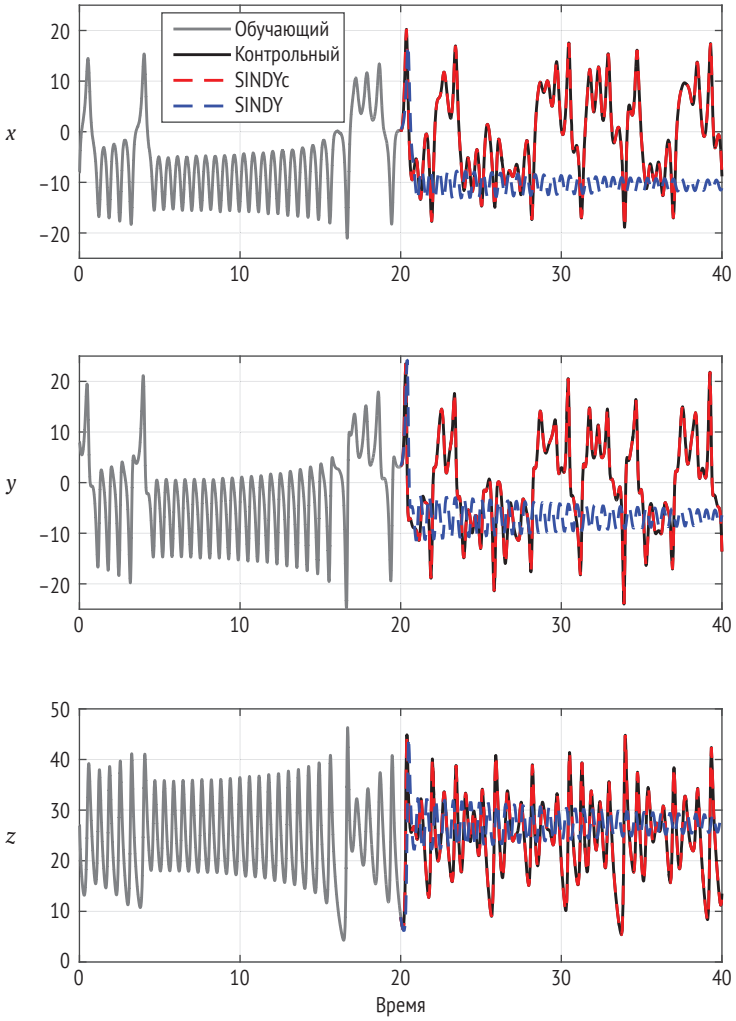

Рис. 10.3 ❖ Предсказания алгоритмов SINDY и SINDYc для управляемой системы Лоренца (10.22). В состав обучающего набора входят данные измерений системы Лоренца с обратной связью. В течение периода обучения входной сигнал равен $u(t) = 26 - x(t) + d(t)$, где d – нормально распределенное возмущение. Затем u переключается на периодический сигнал $u(t) = 50\sin(10t)$. *Взято из работы [100] с разрешения авторов*

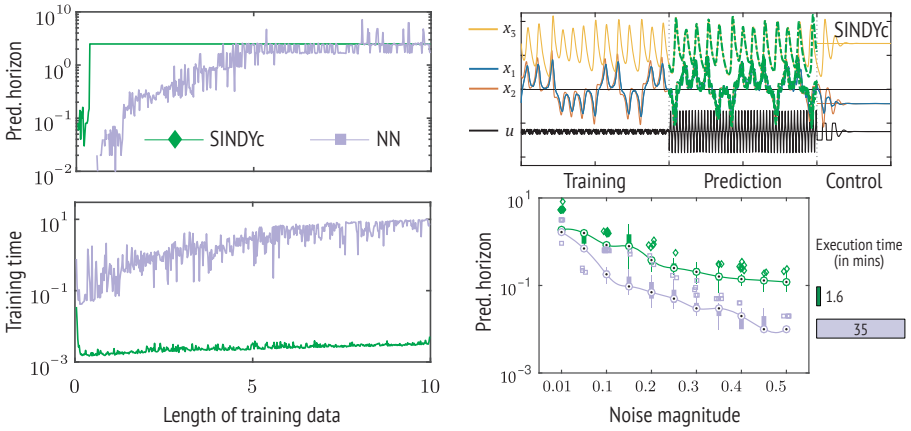

Рис. 10.4 ❖ Управление системой Лоренца с прогнозирующими моделями, построенной с помощью нейронной сети и алгоритма SINDy.
Взято из работы Kaiser et al. [277] с разрешения авторов

10.2. Управление с машинным обучением

Машинное обучение – быстро развивающаяся область, благодаря которой мы получили возможность описывать сложные системы на основе данных наблюдений, а не теоретических моделей [382, 161, 64, 396]. До недавнего времени эти методы разрабатывались в основном для статических данных, хотя растет интерес к использованию машинного обучения для характеризации динамических систем. Применять машинное обучение к законам управления (с целью найти эффективное отображение данных датчиков на управляющие сигналы) стали еще позже [184]. Поскольку машинное обучение охватывает широкий круг методов оптимизации систем высокой размерности, в т. ч. нелинейных, естественно применить его к управлению сложными нелинейными системами. К числу специальных методов машинного обучения для управления относятся адаптивные нейронные сети, генетические алгоритмы, генетическое программирование и обучение с подкреплением. Общая архитектура управления с машинным обучением показана на рис. 10.5. Многие из этих алгоритмов машинного обучения основаны на биологических принципах, в т. ч. нейронные сети, обучение с подкреплением и эволюционные алгоритмы.

Важно отметить, что безмодельные методы управления можно применять к экспериментальным данным и к данным численного моделирования почти без изменений. Во всех безмодельных методах имеется некоторая макроскопическая целевая функция, обычно основанная на показаниях датчиков (прошлых и настоящих). Приведем несколько примеров сложных реальных целевых функций в различных дисциплинах.

Гидрогазодинамика. В аэродинамических приложениях цель часто состоит в том, чтобы найти некоторую комбинацию уменьшения лобового сопротивления, увеличения подъемной силы и снижения шума, тогда как

в фармацевтике и химической промышленности требуется оптимизировать процесс смешения.

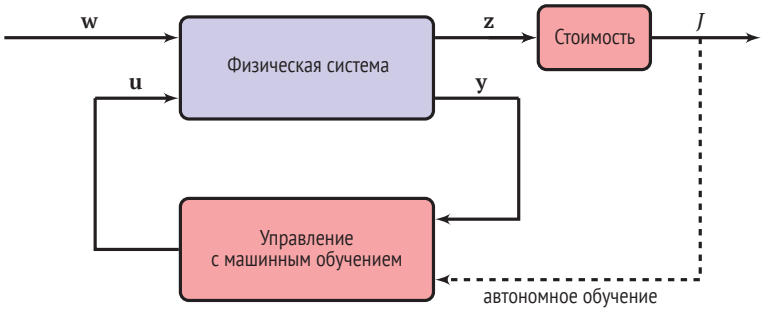

Рис. 10.5 ❖ Схема управления с машинным обучением для сложной системы с обратной связью, основанной на зашумленных показаниях датчиков. Цель управления – минимизировать корректно определенную функцию стоимости J на пространстве возможных законов управления. В цикле автономного обучения экспериментальные данные используются для обучения регулятора. Генетическое программирование – особенно гибкий алгоритм для поиска эффективных законов управления. Вектор **z** содержит всю информацию, которая может войти в функцию стоимости

Финансы. Цель – максимизировать прибыль при заданном уровне риска, не нарушая законов.

Эпидемиология. Цель – эффективно подавить распространение заболевания с ограничениями на процессы измерения (например, анализ крови, клинические исследования и т. д.) и управляющие воздействия (например, вакцинация, надкроватные сетки и т. д.).

Производство. Цель – увеличение производительности труда – должна достигаться при соблюдении ряда ограничений, в т. ч. законов о труде и технике безопасности, а также с заботой об окружающей среде, что зачастую сопряжено со значительной неопределенностью.

Автономные устройства и робототехника. Цель разработки беспилотных автомобилей и автономных роботов – выполнить задачу, безопасно взаимодействуя с окружающей средой, включая в том числе кооперацию с агентами-людьми.

В этих примерах цель состоит в минимизации или максимизации заданной величины при определенных ограничениях. Ограничения могут быть жесткими, как в случае подавления эпидемии при фиксированном бюджете, или могут включать сложный многокритериальный компромисс. Часто ограниченная оптимизация приводит к решениям, находящимся на границе ограничения, что объясняет, почему многие компании работают на грани законности. Во всех случаях оптимизацию необходимо проводить с учетом динамики конкретной системы: течение жидкостей и газов описывается уравнениями Навье–Стокса, финансы определяются поведением людей и экономикой, а распространение заболевания – результат сложного взаимодействия биологии, человеческого поведения и географии.

Эти реальные проблемы управления исключительно трудны по целому ряду причин. Они имеют очень высокую размерность и существенно нелинейны, часто речь идет о миллионах и миллиардах переменных, вступающих друг с другом в неизвестные нелинейные взаимодействия. Кроме того, апробирование различных сценариев идентификации систем может стоить очень дорого или вообще оказаться невозможным; например, имеются серьезные этические проблемы, связанные с испытаниями различных стратегий вакцинации, когда на кону стоит человеческая жизнь.

Все чаще к решению трудных задач оптимизации привлекают машинное обучение, пользуясь доступностью огромных и постоянно увеличивающих объемов данных. Многие из недавних успехов достигнуты при анализе статических данных (например, классификация изображений, распознавание речи и т. д.) и решении маркетинговых проблем (например, онлайновая торговля и размещение рекламных объявлений). Однако предпринимаются усилия, направленные на применение машинного обучения к анализу и управлению сложными динамическими системами. Потенциально это может совершить революцию в нашей способности взаимодействовать с такими системами и управлять ими. В следующих разделах описано несколько эффективных методов обучения, которые широко применяются для управления сложными системами, когда модель отсутствует. Заметим, что относительная важность описанных методов не пропорциональна месту, отведенному для них в книге.

Обучение с подкреплением

Обучение с подкреплением (ОП) – важная дисциплина на стыке машинного обучения и управления [507], в настоящее время она широко используется такими компаниями, как Google, для построения искусственного интеллекта в общем смысле, автономных роботов и беспилотных автомобилей. В обучении с подкреплением политика управления уточняется со временем, так что с накоплением опыта ее качество повышается. Самая распространенная схема ОП – марковский процесс принятия решений, когда динамика системы и политика управления описываются в терминах вероятностей, так что в систему с самого начала встроена стохастичность. Таким образом, вероятностная политика управления поощряет нахождение баланса между оптимизацией и исследованием. Обучение с подкреплением тесно связано с оптимальным управлением, хотя постановка может быть и более общей.

Обучение с подкреплением можно рассматривать как обучение с частичным привлечением учителя, поскольку не всегда становится сразу известно, было управляющее воздействие эффективным или нет. В ОП политика управления реализуется *агентом*, который может получить лишь частичную информацию об эффективности своей стратегии. Например, в играх типа крестиков и ноликов или шахмат не ясно, привел конкретный промежуточный ход к победе или к поражению. Игрок получает бинарную обратную связь в конце игры в виде информации о выигрыше или проигрыше. Основная проблема, решаемая ОП, – разработка *функции ценности* (или функции качества) Q, которая описывает ценность нахождения в конкретном состоя-

нии и принятия того или иного решения об управляющем воздействии. Со временем агент обучается и уточняет функцию Q, улучшая свою способность принимать хорошие решения. Так, в шахматах опытный игрок, оценивая позицию на доске, начинает интуитивно выбирать хорошую стратегию, которая весьма сложным образом зависит от пространства состояний очень высокой размерности (пространства всех возможных позиций). Q-обучение – это безмодельная стратегия обучения с подкреплением, при которой функция ценности обучается на опыте. В последнее время с помощью глубокого обучения удалось кардинально улучшить процесс Q-обучения в ситуациях, когда данные легко доступны [336, 385, 386, 384]. Например, алгоритм Google DeepMind сумел освоить многие классические видеоигры Atari, а недавно победил лучших в мире игроков в го. Мы оставляем углубленное обсуждение обучения с подкреплением другим книгам, но подчеркнем его важность в быстро развивающейся области управления с машинным обучением.

Управление с итеративным обучением

Управление с итеративным обучением (ILC) [5, 67, 83, 130, 343, 390] – широко применяемый метод обучения, в ходе которого уточняются и оптимизируются повторяющиеся задачи управления, например перемещение роботизированной руки на производственной линии, когда рука совершает одно и то же движение тысячи раз. В отличие от методов управления с обратной связью (глава 8), когда управляющий сигнал корректируется в реальном времени на основе результатов измерений, ILC уточняет всю последовательность управления с разомкнутым контуром после каждой итерации выполнения задания. Процесс уточнения может быть совсем простым – например, пропорциональная коррекция на основе измеренной ошибки, а может включать более сложное правило обновления. Для управления с итеративным обучением не нужно знать уравнения системы, а для линейных систем оно дает определенные гарантии качества. Поэтому ILC является основным методом управления повторяющимися производственными процессами в хорошо контролируемой среде, например управления траекторией роботизированной руки или головкой принтера в аддитивных технологиях.

Генетические алгоритмы

Генетический алгоритм (ГА) – один из самых ранних и самых простых алгоритмов оптимизации параметров. Он основан на биологическом принципе оптимизации путем естественного отбора и приспособляемости [250, 146, 210]. ГА часто применяется для настройки и адаптации параметров регулятора. В ГА имеется популяция, состоящая из многих реализаций системы с различными значениями параметров, которые конкурируют между собой с целью минимизации заданной функции стоимости, а успешные значения параметров переходят в следующие поколения благодаря применению набора *генетических* правил. Параметры системы обычно представлены в виде

двоичных последовательностей, как показано на рис. 10.6 для системы с ПИД-управлением с тремя параметрами: коэффициентами усиления K_P, K_I и K_D. Затем инициализируется несколько реализаций с разными значениями параметров – *индивидуумов* – и оценивается их качество на заданной корректно поставленной задаче. Успешные индивидуумы с низкой стоимостью имеют больше шансов быть отобранными для перехода в следующее поколение в результате перечисленных ниже генетических операций.

- **Элитизм (факультативно):** заданное число самых приспособленных индивидуумов с наилучшими показателями переходит в следующее поколение без модификации.
- **Репликация:** выбирается индивидуум для перехода в следующее поколение.
- **Скрещивание:** выбираются два индивидуума, они обмениваются частями кода и переходят в следующее поколение; скрещивание призвано исследовать и усилить уже найденные успешные стратегии.
- **Мутация:** у выбранного индивидуума часть кода заменяется новыми значениями; мутация поощряет разнообразие и призвана расширить исследование пространства параметров.

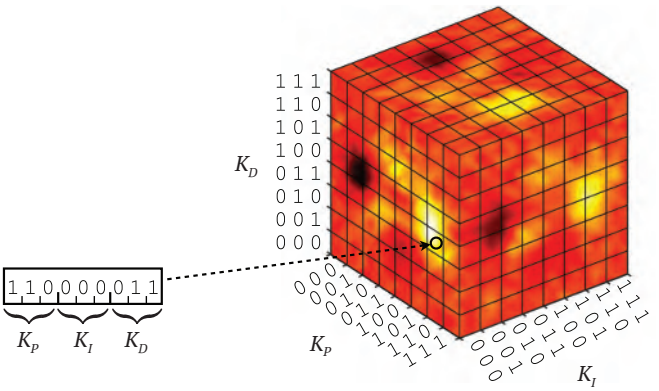

Рис. 10.6 ❖ Куб параметров ПИД-управления. Генетический алгоритм представляет заданное значение параметра в виде *генетической последовательности*, являющейся конкатенацией различных параметров. В этом примере параметры представлены в двоичном виде и масштабированы так, что нижняя граница равна **000**, а верхняя – **111**. Цвет отражает стоимость, ассоциированную с каждым значением параметра

Индивидуумы, подвергаемые репликации, скрещиванию и мутации, выбираются случайно, а вероятность выбора тем больше, чем лучше приспособленность индивидуума. Генетические операции для ПИД-управления показаны на рис. 10.7. Поколения эволюционируют, пока приспособленность лучших индивидуумов не сойдется или не будет выполнено какое-то другое условие остановки.

Генетические алгоритмы обычно применяются для нахождения глобально оптимальных значений параметров, поскольку они позволяют исследовать

и использовать локальные минимумы функции стоимости. ГА представляет собой нечто среднее между полным перебором и выпуклой оптимизацией и является альтернативой затратному методу Монте-Карло, который плохо масштабируется на пространства параметров высокой размерности. Однако нет никакой гарантии, что генетический алгоритм сойдется к глобально оптимальному решению. Существует также ряд гиперпараметров, влияющих на качество алгоритма, в т. ч. размер популяции, число поколений и относительные частоты выбора генетических операций.

Генетические алгоритмы нашли широкое применение в оптимизации и управлении нелинейными системами [184]. Например, ГА использовался для настройки параметров управления с разомкнутым контуром [394], в приложениях, связанных со струйным перемешиванием [304], процессами горения [101], управлением вихревым следом [431, 192] и снижением лобового сопротивления [201]. Также ГА применялся для настройки $\mathcal{H}_\infty$-регулятора в эксперименте с камерой сгорания [233].

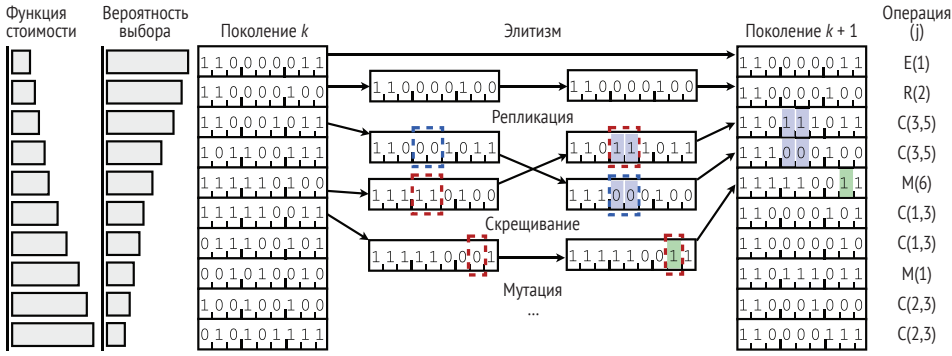

Рис. 10.7 ❖ Эволюция в генетическом алгоритме. Индивидуумы в k-м поколении оцениваются и ранжируются в порядке возрастания функции стоимости, обратно пропорциональной вероятности отбора для применения генетических операций. Затем отбираются индивидуумы в соответствии с этой взвешенной вероятностью перехода в поколение $k + 1$, и к ним применяются четыре операции: элитизм, репликация, скрещивание и мутация. В результате формируется $(k+1)$-е поколение, и эта последовательность действий повторяется, пока статистика популяции не сойдется или не будет выполнено другое условие остановки

Генетическое программирование

Генетическое программирование (ГП) [307, 306] – обобщение генетических алгоритмов с целью одновременной оптимизации структуры и параметров отображения входа в выход. Недавно генетическое программирование применялось также для получения законов управления, которые отображают выходы датчика во входы приводов, как показано на рис. 10.8. Представление дерева функций в ГП очень гибкое, оно позволяет кодировать сложные

функции от сигнала датчика **y** с помощью рекурсивной структуры дерева. Каждая ветвь представляет сигнал, а узлы дерева – математические операции. Датчики и константы находятся в листьях дерева, а результирующий управляющий сигнал *u* – в корне. Генетические операции скрещивания, мутации и репликации схематически изображены на рис. 10.9. Эта схема легко обобщается на запаздывающие координаты и временные фильтры, как обсуждается в работе Duriez et al. [167].

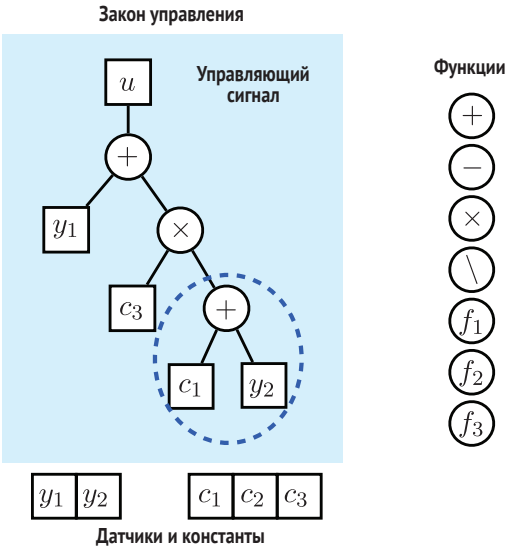

Рис. 10.8 ❖ Дерево функций для представления закона управления **u** в управлении с генетическим программированием

Недавно генетическое программирование продемонстрировало впечатляющие результаты в экспериментах по управлению турбулентностью, поставленных группой Бернда Ноака [403, 417, 199, 168, 169, 416]. Это открывает новую парадигму управления существенно нелинейными системами, которая позволяет идентифицировать структуру нелинейных законов управления. Генетическое программирование особенно хорошо подходит для экспериментов, где можно быстро оценить данный закон управления, поскольку это дает возможность проверить сотни или тысячи индивидуумов за короткое время. Недавние демонстрации генетического программирования для управления турбулентностью позволили выявить несколько макроскопических видов поведения, в частности снижение лобового сопротивления и улучшение смешения, в нескольких конфигурациях потока, а именно слой смешения [417, 416, 168, 169], обратный уступ [199, 169] и отрыв турбулентного пограничного слоя [169].

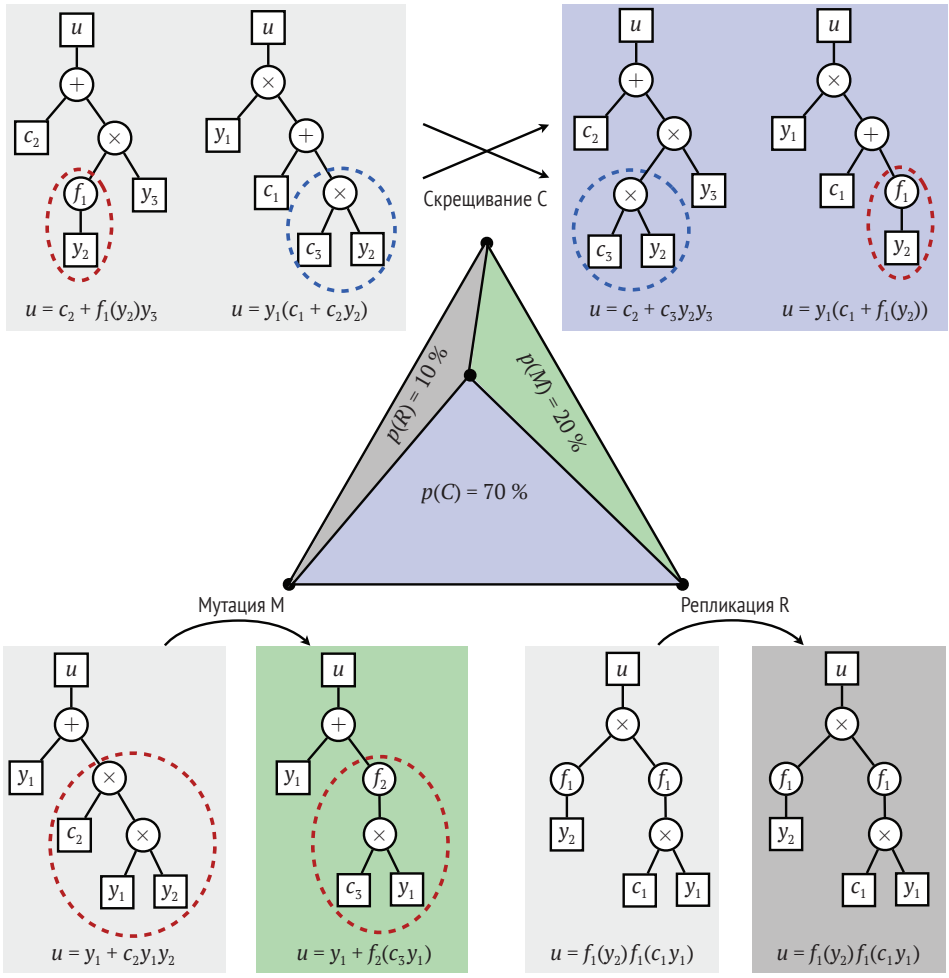

Рис. 10.9 ❖ Генетические операции, используемые в генетическом программировании для модификации деревьев функций при переходе от одного поколения к другому. Относительные частоты выбора репликации, скрещивания и мутации равны $p(R) = 0.1$, $p(C) = 0.7$ и $p(M) = 0.2$ соответственно

Пример: применение генетического алгоритма для настройки ПИД-регулятора

В этом примере мы воспользуемся генетическим алгоритмом для настройки пропорционально-интегрально-дифференциального (ПИД) регулятора. Однако следует отметить, что это лишь простая демонстрация эволюционного алгоритма, а на практике применение такой тяжелой артиллерии для настройки ПИД-регулятора не рекомендуется, потому что есть гораздо более простые методы.

ПИД-управление – одна из самых простых и широко распространенных архитектур управления в промышленных системах. В частности, она применяется для управления на основе датчиков положения и скорости, для настройки различных автомобильных подсистем и для управления температурой и давлением в современных кофе-машинах – и это лишь несколько из огромного количества применений. Как явствует из названия, ПИД-управление аддитивно комбинирует три члена для формирования управляющего сигнала в зависимости от сигнала ошибки, а также его интеграла и производной по времени. Схема ПИД-управления показана на рис. 10.10.

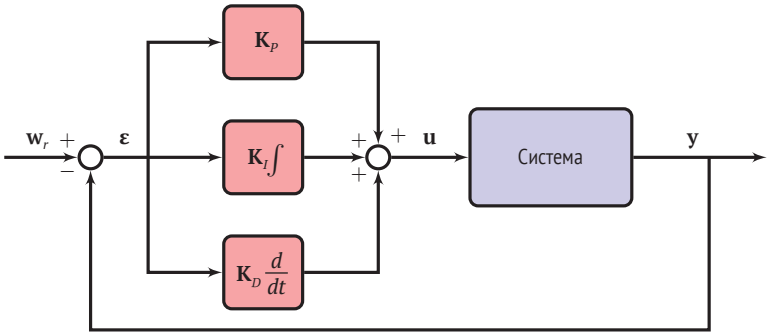

Рис. 10.10 ❖ Схема пропорционально-интегрально-дифференциального управления (ПИД). ПИД по-прежнему повсеместно применяется для управления производственными процессами

В примере круиз-контроля из раздела 8.1 мы видели, что ошибку следования опорной траектории можно уменьшить, увеличив коэффициент пропорционального усиления K_P в законе управления $u = -K_P(w_r - y)$. Однако в некоторых системах увеличение коэффициента усиления может в конечном итоге вызвать неустойчивость и не позволяет полностью устранить ошибку следования в установившемся режиме. Добавление члена интегрального управления $K_I\int_0^t(w_r - y)$ помогает устранить ошибку следования опорной траектории в установившемся режиме, компенсируя в то же время последствия пропорционального члена.

Существуют формальные правила выбора коэффициентов усиления ПИД для различных технических требований, например быстрого отклика и минимального перерегулирования и «звона». В этом примере мы исследуем применение генетического алгоритма для нахождения эффективных коэффициентов усиления ПИД с целью минимизации функции стоимости. Используется функция стоимости ЛКР

$$J = \int_0^T Q(w_r - y)^2 + Ru^2 d\tau$$

с $Q = 1$ и $R = 0.001$ для ступенчатого отклика $w_r = 1$. Подлежащая управлению система описывается передаточной функцией

$$G(s) = \frac{1}{s^4 + s}.$$

Первый шаг – написать функцию, которая оценивает данный ПИД-регулятор (листинг 10.1). Все три коэффициента усиления хранятся в переменной **parms**.

Листинг 10.1 ❖ Вычислить функцию стоимости для ПИД-регулятора

```
function J = pidtest(G,dt,parms)

s = tf('s');
K = parms(1) + parms(2)/s + parms(3)*s/(1+.001*s);
Loop = series(K,G);
ClosedLoop = feedback(Loop,1);
t = 0:dt:20;
[y,t] = step(ClosedLoop,t);

CTRLtf = K/(1+K*G);
u = lsim(K,1-y,t);
```

Затем сравнительно несложно воспользоваться генетическим алгоритмом для оптимизации коэффициентов усиления ПИД-регулятора (листинг 10.2). В этом примере мы прогоняем ГА на протяжении 10 поколений с размером популяции 25 в каждом поколении.

Листинг 10.2 ❖ Применение генетического алгоритма для настройки ПИД-регулятора

```
dt = 0.001;
PopSize = 25;
MaxGenerations = 10;
s = tf('s');
G = 1/(s*(s*s+s+1));

options = optimoptions(@ga,'PopulationSize',PopSize,'
    MaxGenerations',MaxGenerations,'OutputFcn',@myfun);
[x,fval] = ga(@(K)pidtest(G,dt,K),3,-eye(3),zeros(3,1)
    ,[],[],[],[],[],options);
```

Результаты в промежуточных поколениях сохраняются функцией в листинге 10.3.

Листинг 10.3 ❖ Функция для сохранения поколений

```
function [state,opts,optchanged]=myfun(opts,state,flag)
persistent history
persistent cost
optchanged = false;

switch flag
 case 'init'
      history(:,:,1) = state.Population;
```

```
        cost(:,1) = state.Score;
 case {'iter','interrupt'}
        ss = size(history,3);
        history(:,:,ss+1) = state.Population;
        cost(:,ss+1) = state.Score;
 case 'done'
        ss = size(history,3);
        history(:,:,ss+1) = state.Population;
        cost(:,ss+1) = state.Score;
        save history.mat history cost
end
```

Изменение функции стоимости со сменой поколений показано на рис. 10.11. При переходе от поколения к поколению эта функция монотонно убывает. Отдельные коэффициенты усиления показаны на рис. 10.12 – чем точка краснее, тем раньше поколение, к которому она относится, а чем синее, тем позже. Постепенно коэффициенты сходятся к кластеру, окружающему оптимальное решение (черная точка).

На рис. 10.13 показан отклик на действия ПИД-регуляторов из первого поколения. Видно, что многим вариантам регулятора не удалось стабилизировать систему, что выражается в больших отклонениях y. С другой стороны, на рис. 10.14 показан отклик на действия ПИД-регуляторов из последнего поколения. В общем и целом они более эффективно порождают устойчивую ступенчатую характеристику.

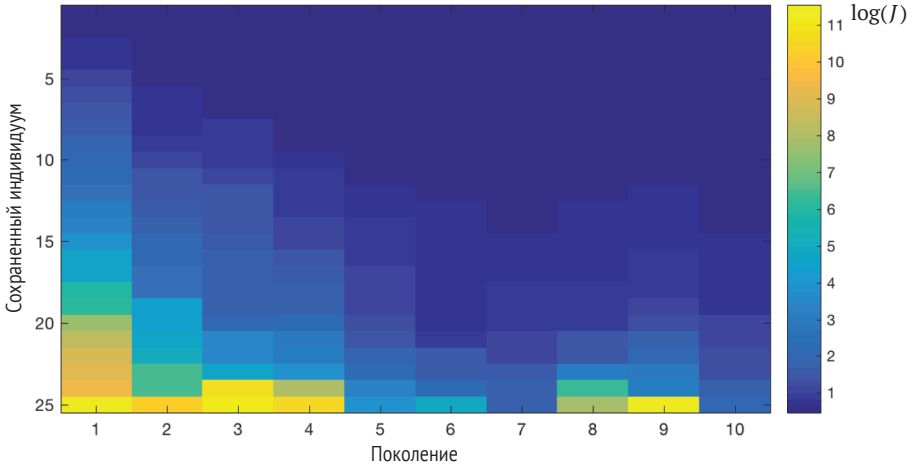

Рис. 10.11 ❖ Зависимость функции стоимости от номера поколения по мере того, как ГА оптимизирует коэффициент усиления ПИД

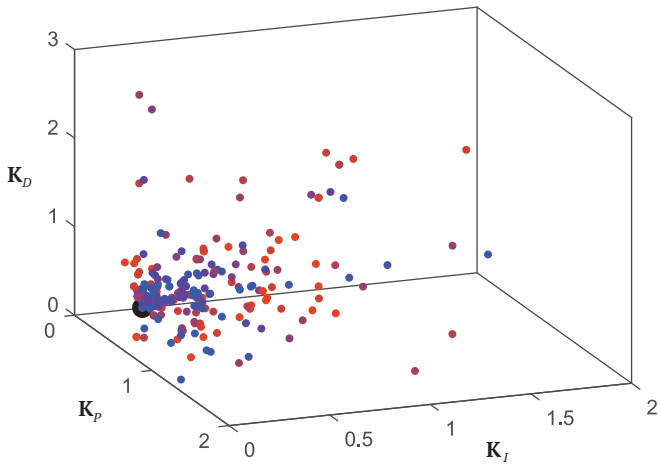

Рис. 10.12 ❖ Коэффициенты усиления ПИД, полученные с помощью генетического алгоритма. Красные точки соответствуют ранним поколениям, синие – более поздним. Черная точка – лучший индивидуум, найденный ГА

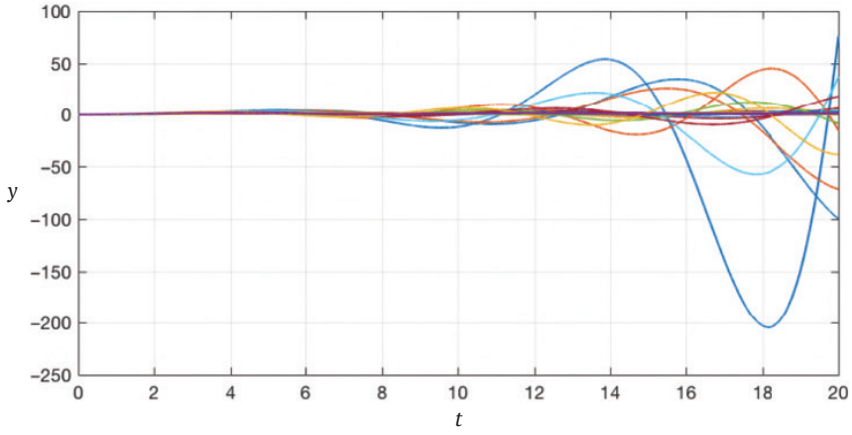

Рис. 10.13 ❖ Характеристики ПИД-регуляторов из первого поколения генетического алгоритма

Лучшие регуляторы в каждом поколении показаны на рис. 10.15. Регуляторы из ранних поколений показаны красным цветом, а из более поздних – синим. Мы видим, что чем больше номер поколения, тем лучше регулятор справляется с минимизацией выходных колебаний и снижением времени нарастания сигнала.

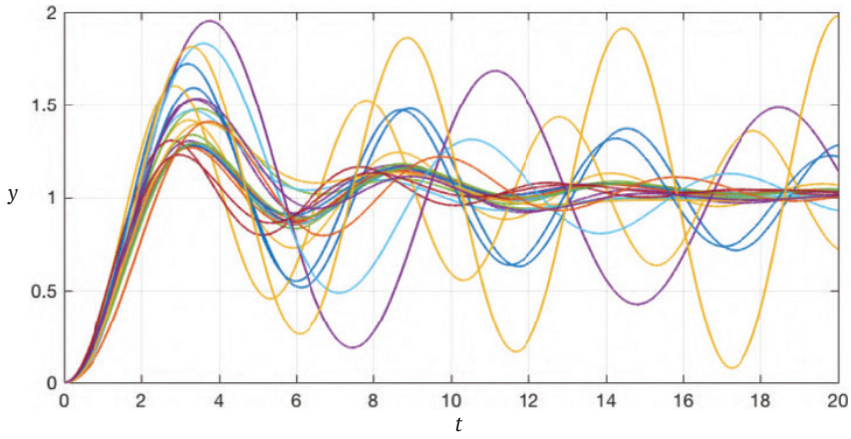

Рис. 10.14 ❖ Характеристики ПИД-регуляторов из последнего поколения генетического алгоритма

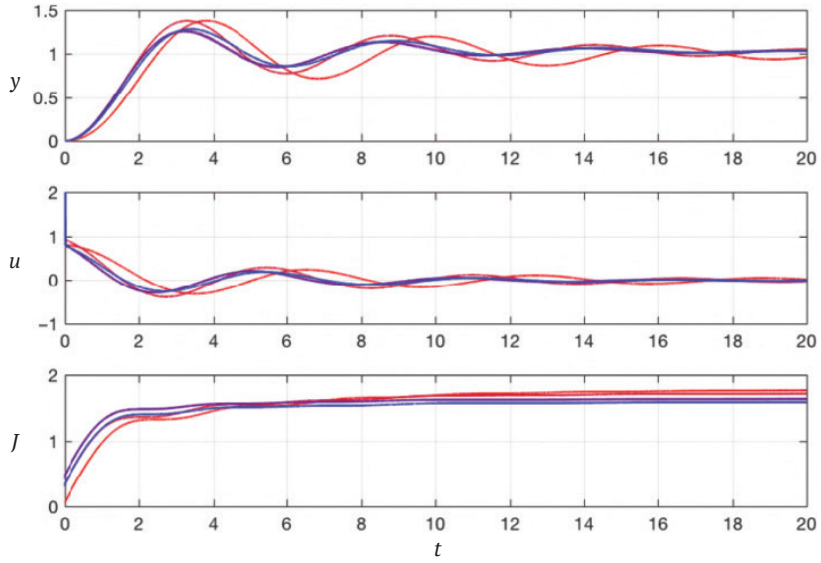

Рис. 10.15 ❖ Лучшие ПИД-регуляторы в каждом поколении. Красные траектории соответствуют ранним поколениям, синие – более поздним

10.3. Адаптивное управление с поиском экстремума

При всем многообразии мощных методов проектирования управления на основе модели у них имеется ряд недостатков. Во-первых, во многих системах модель может быть неизвестна или непригодна для управления (на-

пример, она существенно нелинейна или представлена в нетрадиционной форме). Далее, даже после того как аттрактор идентифицирован и динамика охарактеризована, управление может сделать модель недействительной, поскольку изменяет аттрактор, что приводит к новой, ранее не охарактеризованной динамике. Очевидное исключение – стабилизация в окрестности неподвижной точки или периодической орбиты, в этом случае эффективное управление оставляет систему в области, где линеаризованная модель верна. Наконец, система может претерпевать медленную эволюцию, в ходе которой ее динамика изменяется, а измерить и смоделировать такие эффекты может быть трудно.

Решение этих проблем – сфера *адаптивного* управления, которое допускает гибкое изменение закона управления в зависимости от изменяющей динамики системы. Управление с поиском экстремума (extremum-seeking control – ESC) [312, 19] – особенно заманчивая форма адаптивного управления сложными системами, потому что оно не зависит от наличия модели и гарантирует сходимость и устойчивость при четко определенных условиях. Поиск экстремума может применяться к нахождению локальных максимумов целевой функции в присутствии возмущений, изменяющихся параметров системы и нелинейностей. Адаптивное управление можно применить для своевременного реагирования или для медленной настройки параметров работающего регулятора.

Управление с поиском экстремума можно рассматривать как развитие метода «возмущение и наблюдение», когда синусоидальное возмущение аддитивно подмешивается к управляющему сигналу и используется для оценки градиента целевой функции J, подлежащей максимизации или минимизации. Целевая функция обычно вычисляется на основе показаний датчиков, хотя в конечном итоге зависит от внутренней динамики и выбора входного сигнала. При поиске экстремума переменная управления $\mathbf{u}$ может относиться как к управляющему сигналу, так и к набору параметров, описывающих поведение регулятора, таких, например, как частота периодического воздействия или коэффициенты усиления ПИД-регулятора.

Архитектура управления с поиском экстремума показана на рис. 10.16. Изображено ESC для скалярного входа u, хотя метод легко обобщается на векторные входы $\mathbf{u}$. На рис. 10.17 показана выпуклая целевая функция $J(u)$ для статической динамики объекта (т. е. $y = u$). Регулятор с поиском экстремума использует входное возмущение для оценки градиента целевой функции J и направления среднего управляющего сигнала в сторону оптимизирующего значения.

При управлении с поиском экстремума нужно рассматривать три временных масштаба:

1) медленный – внешние помехи и изменение параметров;
2) средний – частота возмущения ω;
3) быстрый – динамика системы.

Во многих системах изменение внутренней динамики описывается быстрым временным масштабом. Например, турбулентные флуктуации могут приходить в равновесие быстро по сравнению с временным масштабом управляющего воздействия. Так же обстоит дело в оптических системах,

например в оптоволоконном лазере [93], где речь идет о динамике света, распространяющегося в волокне.

При управлении с поиском экстремума прибавляется синусоидальное возмущение, чтобы оценить входной сигнал, который доставляет максимум целевой функции $\hat{u}$:

$$u = \hat{u} + a\sin(\omega t). \tag{10.23}$$

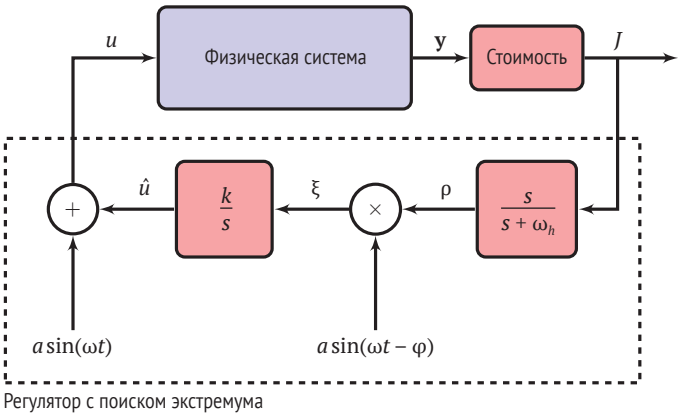

Рис. 10.16 ❖ Схема регулятора с поиском экстремума. Синусоидальное возмущение прибавляется к наилучшей гипотезе о входе $\hat{u}$ и пропускается через управляемый объект, что приводит к синусоидальному возмущению на выходе, которое можно наблюдать с помощью сигнала датчика **y** и функции стоимости *J*. Благодаря фильтру верхних частот возмущение на выходе имеет нулевое среднее. Это возмущение затем умножается (демодулируется) на то же самое входное возмущение, что дает сигнал ξ. Наконец, этот демодулированный сигнал интегрируется в наилучшую гипотезу $\hat{u}$ для оптимизирующего входа *u*

Входное возмущение отражается на динамике системы и выходном сигнале, в результате чего целевая функция *J* синусоидально колеблется вокруг некоторого среднего значения, как показано на рис. 10.17. Выход *J* пропускается через фильтр верхних частот, чтобы вычесть среднее (постоянную составляющую), в итоге порождается осциллирующий сигнал ρ. Простой фильтр верхних частот можно представить в частотной области функцией

$$\frac{s}{s + \omega_h}, \tag{10.24}$$

где s – переменная Лапласа, а ω_h – частота фильтра. Фильтр верхних частот устроен так, чтобы пропускать частоту возмущения ω. Прошедший через фильтр выходной сигнал умножается на входную синусоиду, возможно, со сдвигом фазы φ, что дает *демодулированный сигнал* ξ:

$$\xi = a\sin(\omega t - \varphi)\rho. \tag{10.25}$$

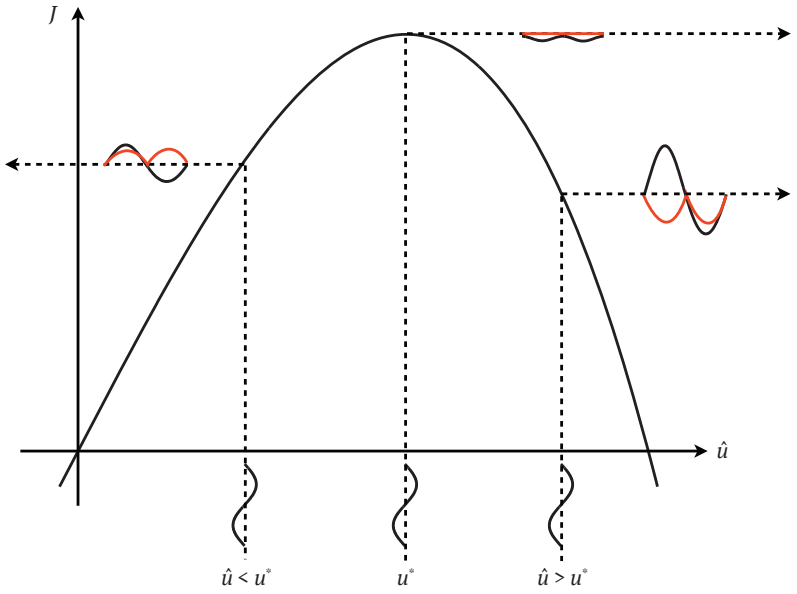

Рис. 10.17 ❖ Схема управления с поиском экстремума для статической целевой функции $J(u)$. Выходное возмущение (красным цветом) синфазно, когда входной сигнал находится слева от пикового значения (т. е. $u < u^*$), и несинфазно, когда входной сигнал справа от пика (т. е. $u > u^*$). Таким образом, интегрирование произведения входной и выходной синусоид сдвигает $\hat{u}$ в сторону u^*

Сигнал ξ в основном положителен, если вход u находится слева от оптимального значения u^*, и в основном отрицателен, если u находится справа от u^*. Эти кривые показаны красным цветом на рис. 10.17. Таким образом, демодулированный сигнал ξ интегрируется в $\hat{u}$, наилучшую оценку оптимизирующего значения

$$\frac{d}{dt}\hat{u} = k\xi, \tag{10.26}$$

так что оценка $\hat{u}$ направлена в сторону оптимального входа u^*. Здесь k – интегральный коэффициент усиления, который определяет, насколько агрессивно управляющее воздействие вдоль направления градиента J.

Грубо говоря, демодулированный сигнал ξ измеряет градиенты целевой функции, так что алгоритм движется в сторону оптимума тем быстрее, чем больше градиент. Это легко видеть для постоянной динамики объекта, когда J является просто функцией от входа $J(u) = J(\hat{u} + a\sin(\omega t))$. Разложение $J(u)$ в ряд по амплитуде возмущения a, которая предполагается малой, дает:

$$J(u) = J(\hat{u} + a\sin(\omega t)), \tag{10.27a}$$

$$J(\hat{u}) + \left.\frac{\partial J}{\partial u}\right|_{u=\hat{u}} \cdot a\sin(\omega t) + \mathcal{O}(a^2). \tag{10.27b}$$

Старший член прошедшего через фильтр верхних частот сигнала равен $\rho \approx \partial J/\partial u|_{u=\hat{u}} \cdot a\sin(\omega t)$. Усредняя $\xi = a\sin(\omega t - \varphi)\rho$ по одному периоду, получаем:

$$\langle \psi_k, \psi_j \rangle = \int_{-L}^{L} \psi_k \psi_j^* dx \tag{10.28a}$$

$$= \frac{\omega}{2\pi} \int_0^{2\pi/\omega} \left. \frac{\partial J}{\partial u} \right|_{u=\hat{u}} a^2 \sin(\omega t - \varphi)\sin(\omega t) dt \tag{10.28b}$$

$$= \frac{a^2}{2} \left. \frac{\partial J}{\partial u} \right|_{u=\hat{u}} \cos(\varphi). \tag{10.28c}$$

Таким образом, в случае тривиальной динамики объекта средний сигнал ξ_{avg} пропорционален градиенту целевой функции J по входу u.

В общем случае управление с поиском экстремума можно применять к системам с нелинейной динамикой, связывающей вход u с выходами $\mathbf{y}$, временной масштаб которой быстрее, чем частота возмущения ω. Стало быть, J может быть нестационарной, что усложняет приведенный выше упрощенный анализ с усреднением. Общий случай управления с поиском экстремума для нелинейных систем проанализирован в работе Krstić and Wang [312], где выведены сильные гарантии устойчивости на основе разделения временных масштабов и анализа сингулярных возмущений. Базовый алгоритм также можно модифицировать, добавив фазу φ синусоидального возмущения входа (10.25). В работе [312] имеется дополнительный фильтр нижних частот $\omega_l/(s + \omega_l)$ перед интегратором, который выделяет постоянную составляющую из демодулированного сигнала ξ. Существует также обобщение поиска экстремума, которое называется поиском наклона, когда ищется конкретный наклон [19], а не стандартный нулевой наклон, соответствующий минимуму или максимуму. Поиск наклона предпочтителен, если экстремума не существует, как в случае, когда выходная характеристика стремится к асимптоте. Поиск экстремума часто применяется для выбора частоты, а поиск наклона – для выбора амплитуды при настройке периодического возбуждения разомкнутого контура.

Важно отметить, что при управлении с поиском экстремума ищутся только локальные максимумы целевой функции, и нет никакой гарантии, что будет найден глобальный максимум. Поэтому важно выбирать хорошее начальное приближение. В ряде исследований управление с поиском экстремума используется в сочетании с другими методами глобальной оптимизации, например генетическими алгоритмами или разреженным представлением для классификации [191, 99].

Простой пример управления с поиском экстремума

Рассмотрим простой пример поиска максимума статической квадратичной функции стоимости

$$J(u) = 25 - (5 - u)^2. \tag{10.29}$$

Эта функция имеет единственный глобальный максимум в точке $u^* = 5$. Начав с $u = 0$, мы применяем управление с поиском экстремума с частотой возмущения $\omega = 10$ Гц и амплитудой $a = 0.2$. На рис. 10.18 показана характеристика регулятора и быстрое приближение к оптимальному значению $u^* = 5$. В программе в листинге 10.4 поиск экстремума реализуется с помощью простого фильтра верхних частот Баттерворта.

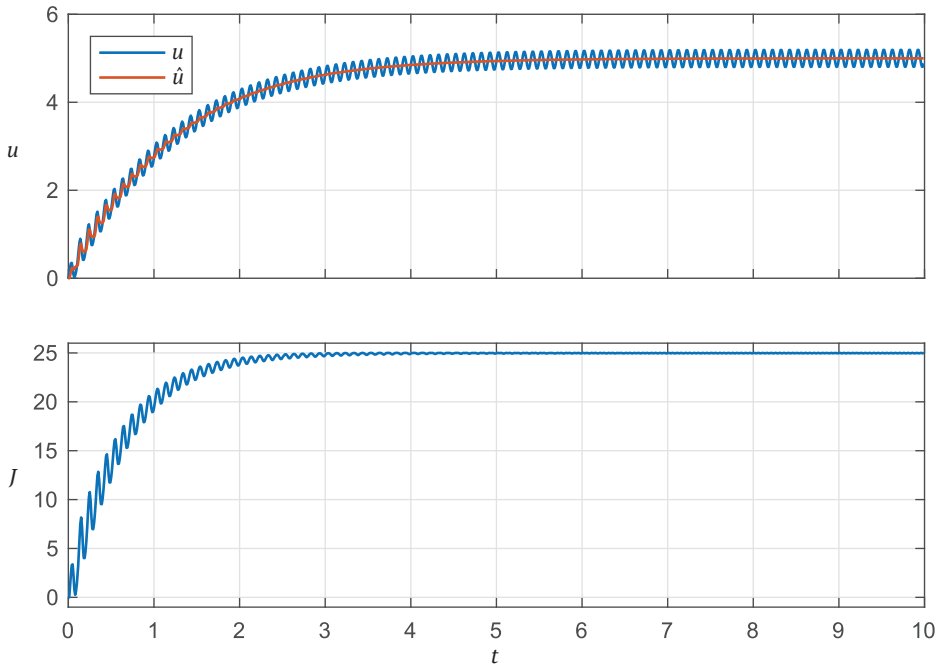

Рис. 10.18 ❖ Характеристика управления с поиском экстремума для функции стоимости (10.29)

Заметим, что когда градиент функции стоимости велик (т. е. в точках, близких к $u = 0$), колебания J больше и характеристика быстро растет. По мере приближения u к точке оптимума $u^* = 5$, несмотря на то что входное возмущение по-прежнему имеет амплитуду a, возмущение на выходе близко к нулю (порядка a^2), потому что квадратичная функция стоимости является почти плоской в окрестности пика. Таким образом, мы достигли быстрого сближения с опорной траекторией вдали от оптимума и малых отклонений рядом с пиком.

Листинг 10.4 ❖ Код управления с поиском экстремума

```
J = @(u,t)(25-(5-(u)).^2);
y0 = J(0,0); % u = 0

% Параметры управления с поиском экстремума
freq = 10*2*pi; % выборочная частота
```

```
dt = 1/freq;
T = 10; % полное время моделирования (в секундах)
A = .2; % амплитуда
omega = 10*2*pi; % 10 Hz
phase = 0;
K = 5; % коэффициент интегрирующего преобразования

% Фильтр верхних частот (фильтр Баттерворта)
butterorder=1;
butterfreq=2; % в Гц для 'high'
[b,a] = butter(butterorder,butterfreq*dt*2,'high')
ys = zeros(1,butterorder+1)+y0;
HPF=zeros(1,butterorder+1);

uhat=u;
for i=1:T/dt
    t = (i-1)*dt;
    yvals(i)=J(u,t);

    for k=1:butterorder
        ys(k) = ys(k+1);
        HPF(k) = HPF(k+1);
    end
    ys(butterorder+1) = yvals(i);
    HPFnew = 0;
    for k=1:butterorder+1
        HPFnew = HPFnew + b(k)*ys(butterorder+2-k);
    end
    for k=2:butterorder+1
        HPFnew = HPFnew - a(k)*HPF(butterorder+2-k);
    end
    HPF(butterorder+1) = HPFnew;

    xi = HPFnew*sin(omega*t + phase);
    uhat = uhat + xi*K*dt;
    u = uhat + A*sin(omega*t + phase);
    uhats(i) = uhat;
    uvals(i) = u;
end
```

Чтобы убедиться в способности управления с поиском экстремума справляться с изменяющимися параметрами системы, рассмотрим зависящую от времени функцию стоимости

$$J(u) = 25 - (5 - u - \sin(t))^2. \tag{10.30}$$

Изменение параметров, которые колеблются с частотой $1/2\pi$ Гц, можно считать медленным по сравнению с частотой возмущения 10 Гц. Характеристика управления для этой медленно изменяющейся системы показана на рис. 10.19. Как видим, удается поддерживать хорошее качество благодаря тому, что управляющее воздействие колеблется, приближенно повторяя оптимальный сигнал u^*, который периодически изменяется в диапазоне от

4 до 6. Выходная функция J остается близка к оптимальному значению 25, хотя изменяющийся параметр неизвестен.

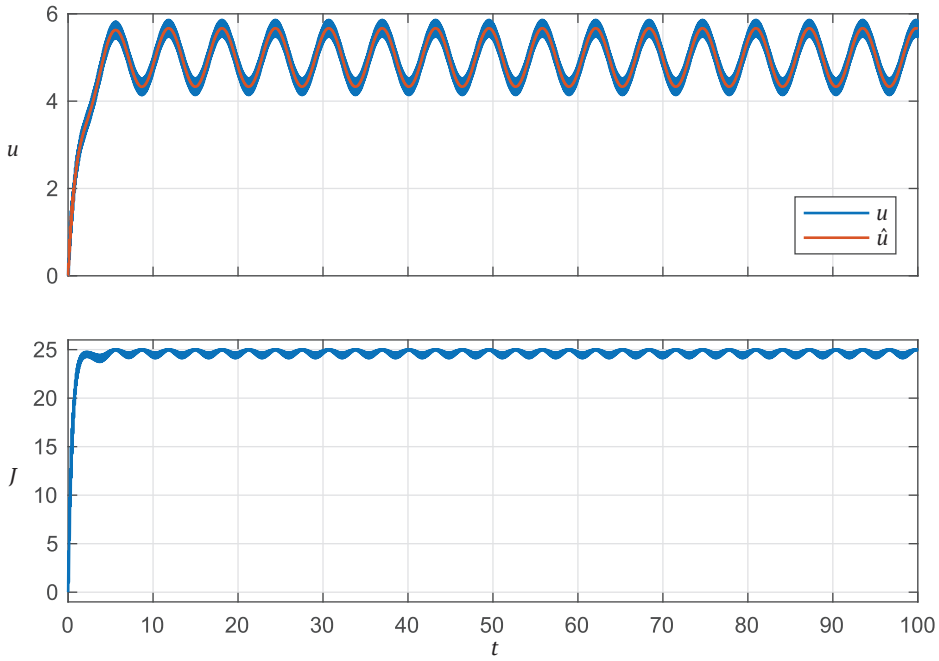

Рис. 10.19 ❖ Характеристика регулятора с поиском экстремума, имеющего медленно изменяющуюся функцию стоимости $J(u, t)$

Пример управления с поиском экстремума в сложной ситуации

Рассмотрим пример, на который навела трудная эталонная задача, приведенная в разделе 1.3 книги [19]. У этой системы зависящая от времени целевая функция $J(t)$, а динамика имеет нуль в правой полуплоскости, что затрудняет управление ей.

В одной из формулировок управления с поиском экстремума [133, 19] приводятся дополнительные рекомендации по проектированию регулятора в случае, когда объект можно разделить на три блока, определяющих входную динамику, нестационарную целевую функцию без внутренней динамики и выходную динамику, как показано на рис. 10.20. В этом случае существуют процедуры проектирования фильтра верхних частот и интегратора.

В данном примере целевая функция имеет вид

$$J(\theta) = 0.05\delta(t - 10) + (\theta - \theta^*(t))^2,$$

где δ – дельта-функция Дирака, а оптимальное значение $\theta^*(t)$ равно

$$\theta^* = 0.01 + 0.001t.$$

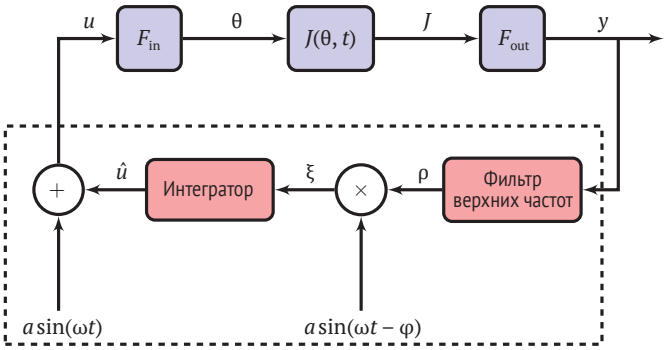

Рис. 10.20 ❖ Схема одной из архитектур управления с поиском экстремума, в которой используются различные методики проектирования [133, 19]

Оптимальная целевая функция имеет вид $J^* = 0.05\delta(t - 10)$. Входная и выходная динамики взяты из примера в книге [19]:

$$F_{\text{in}}(s) = \frac{s-1}{(s+2)(s+1)}, \qquad F_{\text{out}}(s) = \frac{1}{s+1}.$$

Применяя процедуру, описанную в [19], находим фильтр верхних частот $s/(s+5)$ и интегратор $50(s-4)/(s-0.01)$. Дополнительно используется возмущение с частотой $\omega = 5$ и амплитудой $a = 0.05$, а демодулирующее возмущение сдвинуто по фазе на угол $\varphi = 0.7955$; эта фаза получена вычислением входной функции F_{in} в точке $i\omega$. Характеристика этого регулятора показана на рис. 10.21, а его реализация в Simulink – на рис. 10.22. Регулятор хорошо отслеживает оптимизирующий входной сигнал, несмотря на аддитивный шум в датчиках.

Приложения управления с поиском экстремума

Благодаря отсутствию априорных предположений и простоте реализации управление с поиском экстремума широко применяется во многих сложных системах. Хотя ESC, вообще говоря, применимо для своевременного управления динамическими системами, оно также активно используется в качестве алгоритма онлайновой оптимизации, способного адаптироваться к медленным изменениям и возмущениям. Из многочисленных приложений ESC упомянем лишь некоторые.

Поиск экстремума широко применялся в алгоритмах слежения за точкой максимальной мощности в фотоэлектрических устройствах [331, 178, 75, 97] и преобразователях ветровой энергии [395]. В случае фотоэлектрических устройств в качестве возмущающего сигнала используется пульсация тока или напряжения в преобразователях энергии, создаваемая широтно-импульсной модуляцией, а в случае ветровых преобразователей возмущением служит турбулентность. Турбулентные флуктуации в атмосфере также использовались в качестве возмущающего сигнала для оптимизации управления летательным аппаратом [309]; в этом случае нельзя подавать возмущающий сигнал на

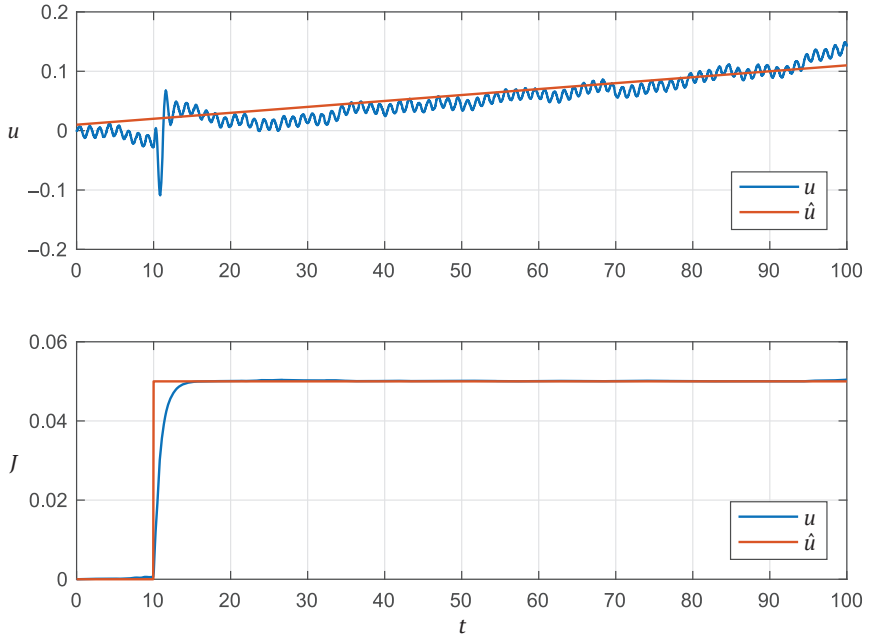

Рис. 10.21 ❖ Характеристика регулятора с поиском экстремума для трудной тестовой системы с нулем в правой полуплоскости, взятой из книги [19]

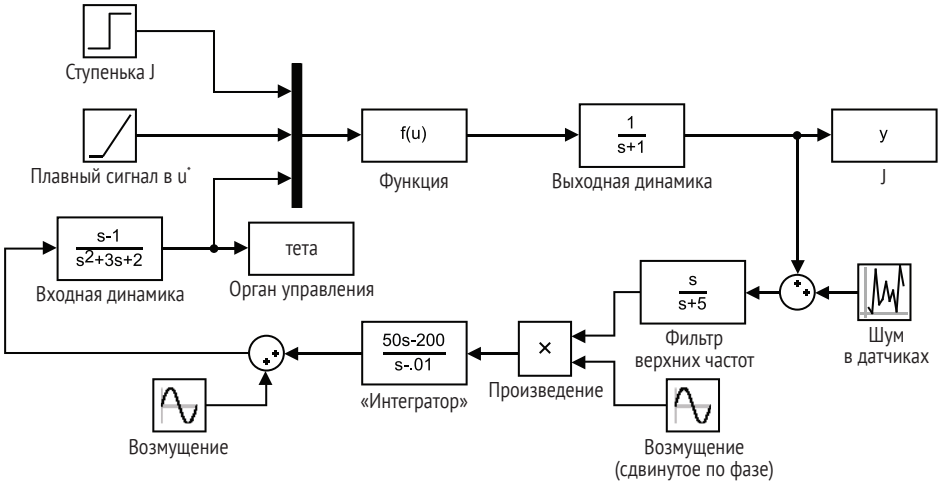

Рис. 10.22 ❖ Созданная в Simulink модель регулятора с поиском экстремума, показанного на рис. 10.21

плоскости управления ЛА, требуется естественное возмущение. ESC применялось в оптике и электронике для формирования лазерного импульса [450], настройки волоконного лазера с высоким усилением [93, 99] и для управления лучом в переналаживаемых антенных системах из голографических метама-

териалов [265]. Из других применений отметим оптимизацию полета в строю [60], биореакторы [546], настройку ПИД- [289] и ПИ- [311] регуляторов, системы активного торможения [568] и управление токамаками [413].

Поиск экстремума также широко применялся для управления турбулентным течением. Несмотря на способность ESC управлять динамикой своевременно, часто оно используется для оптимизации с медленной обратной связью, чтобы настроить параметры работающего регулятора с разомкнутым контуром. У такой медленной обратной связи много преимуществ, в т. ч. поддержание требуемого качества при наличии медленных изменений окружающих условий. Поиск экстремума применялся для управления осевым компрессором [547], для уменьшения лобового сопротивления плохо обтекаемого тела в эксперименте [45, 46] с использованием вращающегося цилиндра в верхней части хвостовой кромки задней поверхности и для управления отрывом в конфигурации крыла с отклоненной механизацией [47] с датчиками давления и пульсирующими воздушно-реактивными двигателями на передней кромке однощелевого закрылка. Имеются также впечатляющие применения управления с поиском экстремума в промышленном масштабе, например для управления термоакустическими режимами в камере сгорания газотурбинной установки мощностью 4 МВт в широком спектре частот [37, 35]. ESC применялось и для управления разделением в плоском диффузоре с отрывом полностью турбулентного потока [36], а также для управления шумом реактивного двигателя [375].

Имеются многочисленные расширения метода поиска экстремума, призванные улучшить качество управления. Например, в работе [202] расширенные фильтры Калмана использовались в качестве фильтров для управления термоакустической неустойчивостью в экспериментах с камерой сгорания; в результате удалось уменьшить флуктуации давления почти на 40 дБ. Также фильтры Калмана использовались в сочетании с ESC, чтобы уменьшить отрыв потока и увеличить степень повышения давления в осевом воздухонагнетателе высокого давления с помощью нагнетаемого пульсирующего потока воздуха [553]. Включение фильтра Калмана улучшило полосу пропускания регулятора в 10 раз по сравнению с традиционным ESC.

Рекомендуемая литература

Учебники

(1) **Reinforcement learning: An introduction**, by R. S. Sutton and A. G. Barto, 1998 [507].
(2) **Real-time optimization by extremum-seeking control**, by K. B. Ariyur and M. Krstić, 2003 [19].
(3) **Machine learning control: Taming nonlinear dynamics and turbulence**, by T. Duriez, S. L. Brunton, and B. R. Noack, 2016 [167].
(4) **Model predictive control**, by E. F. Camancho, C. B. Alba, 2013 [107].

Статьи и обзоры

(1) Stability of extremum seeking feedback for general nonlinear dynamic systems, by M. Krstić and H. H. Wang, *Automatica*, 2000 [312].
(2) Dynamic mode decomposition with control, by J. L. Proctor, S. L. Brunton, and J. N. Kutz, *SIAM Journal on Applied Dynamical Systems*, 2016 [434].
(3) Model predictive control: theory and practice – a survey, by C. E. Garcia, D. M. Prett, and M. Morari, *Automatica*, 1989 [195].
(4) Closed-loop turbulence control: Progress and challenges, by S. L. Brunton and B. R. Noack, *Applied Mechanics Reviews*, 2015 [94].

Часть IV

МОДЕЛИ ПОНИЖЕННОГО ПОРЯДКА

Глава 11

Модели пониженного порядка (ROM)

Собственное ортогональное разложение (POD) – это алгоритм SVD, примененный к дифференциальным уравнениям в частных производных (УрЧП). В этом качестве оно является одним из самых важных методов понижения размерности, пригодных для изучения сложных пространственно-временных систем. Такие системы обычно описываются нелинейными дифференциальными уравнениями в частных производных. Своим успехом POD обязано универсальному наблюдению, согласно которому в большинстве сложных систем значимое поведение сосредоточено в низкоразмерных паттернах динамической активности. Идея метода POD – воспользоваться этим фактом и найти динамические системы низкого ранга, способные правильно смоделировать полную эволюцию истинной системы в пространстве и времени. Точнее, в *моделях пониженного порядка* (reduced order model – ROM) моды POD используются для проецирования динамики УрЧП на низкоранговое подпространство, в котором проще обсчитать модель, описываемую УрЧП. Важно, что порождаемые низкоранговые модели позволяют значительно ускорить вычисления и, возможно, даже применить чрезвычайно затратное моделирование УрЧП методом Монте-Карло, оптимизацию параметризованных УрЧП-систем и (или) управление такими системами в режиме реального времени. POD широко используется в исследованиях по гидрогазодинамике [251], а также находит разнообразные применения в приложениях в строительной механике и анализе колебательных процессов [287, 23, 232, 329], в оптических и микроэлектромеханических технологиях [333, 488], в изучении атмосферы (где POD называется эмпирическими ортогональными функциями (EOF)) [116, 117], в конструировании ветряных генераторов [494], в акустике [181] и в нейронауках [33, 519, 284]. Успех метода связан с его способностью находить пространственно-временные разложения данных, допускающие физическую интерпретацию [316, 57, 181, 286, 126, 333].

11.1. POD для дифференциальных уравнений в частных производных

Во многих системах, встречающихся в инженерных, физических и биологических науках, известны пространственно-временные соотношения, описывающие паттерны динамической активности. Даже простая пространственно-временная связь может приводить к очень сложной, но тем не менее когерентной динамике, что и является побудительным мотивом для аналитических и численных исследований. Задача моделирования – вывести эти пространственно-временные соотношения либо из основных законов, либо с помощью хорошо аргументированных заключений о существующих связях. В итоге получается дифференциальное уравнение в частных производных, описывающее поведение сложной системы. Как правило, решить такие УрЧП аналитически невозможно. Поэтому в нашем распоряжении остаются две стратегии: численное решение и (или) асимптотическое упрощение. В первом случае сложная система дискретизируется в пространстве и во времени, в результате чего порождается система уравнений очень высокой размерности, которую можно решить с требуемой точностью, но чем выше точность, тем больше должна быть размерность дискретизированной системы. При таком подходе высокая размерность искусственна и является просто следствием лежащей в основе численной схемы решения. С другой стороны, цель асимптотического упрощения – заменить сложную систему более простой системой уравнений, предпочтительно *линейной*, поскольку она удобнее для анализа. До 1960-х годов и взлета вычислительной техники такое асимптотическое упрощение было основой прикладной математики в таких областях, как гидрогазодинамика. Именно на асимптотике были сосредоточены первые усилия в части понижения размерности. В этой книге асимптотические методы не рассматриваются, зато рассматриваются численные методы, приведшие к созданию моделей пониженного порядка.

Чтобы придать нашему изучению сложных систем математическую строгость, будем рассматривать общую систему нелинейных УрЧП с одной пространственной переменной вида

$$\mathbf{u}_t = \mathbf{N}(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, \cdots, x, t; \boldsymbol{\beta}), \tag{11.1}$$

где нижние индексы обозначают частные производные, функция $\mathbf{N}(\cdot)$ описывает общее нелинейное изменение, а $\boldsymbol{\beta}$ – бифуркационный параметр, о котором мы поговорим позже. С уравнением (11.1) ассоциированы также начальные и граничные условия на отрезке $x \in [-L, L]$. На протяжении длительного времени были разработаны различные аналитические методы изучения уравнений вида (11.1). Как правило, цель таких методов – свести УрЧП (11.1) к набору обыкновенных дифференциальных уравнений (ОДУ). Стандартные методы решения УрЧП – *разделение переменных* и *метод подобия* – предназначены именно для этой цели. После того как уравнение приведено к форме ОДУ, к нему можно применить более широкий спектр аналитических методов, а в случае нелинейного поведения – также *качест-*

венную теорию [252]. И это снова приводит нас к роли *асимптотики* в характеризации поведения.

Хотя мы и сказали, что существует несколько стратегий решения, в общем случае уравнение (11.1) не допускает решения в замкнутой форме. Достаточно даже самой простой нелинейности или пространственной зависимости коэффициентов – и все известные аналитические методы становятся бесполезными. Впрочем, нет недостатка в численных способах решения (11.1), применимых к физическим, техническим и биологическим наукам. Эти методы дают приближенное численное решение системы (11.1), имеющей высокую размерность. Рассмотрим, например, стандартную пространственную дискретизацию системы (11.1), когда переменная x вычисляется в $n \gg 1$ точках

$$\mathbf{u}(x_k, t),\ k = 1, 2, \cdots, n, \tag{11.2}$$

с интервалом $\Delta x = x_{k+1} - x_k = 2L/n$. С помощью обычных формул конечных разностей пространственные производные можно вычислить по соседним точкам, например:

$$\mathbf{u}_x = \frac{\mathbf{u}(x_{k+1},t) - \mathbf{u}(x_{k-1},t)}{2\Delta x}; \tag{11.3a}$$

$$\mathbf{u}_{xx} = \frac{\mathbf{u}(x_{k+1},t) - 2\mathbf{u}(x_k,t) + \mathbf{u}(x_{k-1},t)}{\Delta x^2}. \tag{11.3b}$$

В результате такой пространственной дискретизации УрЧП (11.1) преобразуется в систему n ОДУ

$$\frac{d\mathbf{u}_k}{dt} = \mathbf{N}(\mathbf{u}(x_{k+1},t), \mathbf{u}(x_k,t), \mathbf{u}(x_{k-1},t), \cdots, x_k, t, \boldsymbol{\beta}), \quad k = 1, 2, \cdots, n. \tag{11.4}$$

Этот процесс дискретизации приводит к более удобной системе уравнений ценой повышения размерности (11.1). Отметим, что чем строже требования к точности, тем выше размерность n системы (11.4), потому что $\Delta x = 2L/n$. Таким образом, размерность базовой вычислительной схемы искусственно определяется точностью конечно-разностной схемы.

Пространственная дискретизация (11.1) иллюстрирует, как возникают системы высокой размерности. Искусственное порождение таких систем присутствует в вычислительных схемах повсеместно, что сильно осложняет научные расчеты. В качестве дополнительной иллюстрации этого феномена рассмотрим еще одну вычислительную схему для решения системы (11.1). Речь идет о самом распространенном методе аналитического решения УрЧП: разделении переменных. В этом методе предполагается, что решение системы устроено так, что пространство и время независимы:

$$\mathbf{u}(x, t) = \mathbf{a}(t)\psi(x), \tag{11.5}$$

где в функции $\mathbf{a}(t)$ сосредоточена вся зависимость от времени, а в функции $\psi(x)$ – вся зависимость от пространства. Аналитическое разделение пере-

менных работает, только если уравнение (11.1) линейно, а коэффициенты постоянны. В этом, очень специальном случае можно вывести два дифференциальных уравнения, которые порознь характеризуют пространственные и временные зависимости сложной системы. Эти дифференциальные уравнения связаны постоянным параметром, присутствующим в обоих.

Для общего уравнения (11.1) разделение переменных можно использовать для создания вычислительного алгоритма, способного находить точные решения. Поскольку пространственные решения заранее неизвестны, обычно предполагают существование набора базисных мод, используемых для построения $\psi(x)$. На самом деле такое предположение о базисных модах лежит в основе идеи метода *разложения по собственным функциям*. Это дает решение с разделенными переменными вида

$$\mathbf{u}(x,t) = \sum_{k=1}^{n} \mathbf{a}_k(t)\psi_k(x), \tag{11.6}$$

где $\psi_k(x)$ образуют набор $n \gg 1$ базисных мод. Как и раньше, такое разложение искусственно приводит к системе уравнений высокой размерности, потому что требуется n мод. Решение с разделенными переменными аппроксимирует истинное решение при условии, что n достаточно велико. Увеличение числа мод n эквивалентно увеличению числа шагов пространственной дискретизации в конечно-разностной схеме.

Благодаря свойствам ортогональности базисных функций $\psi_k(x)$ мы можем воспользоваться формулой (11.6). Для иллюстрации рассмотрим скалярный вариант (11.1) и соответствующее скалярное решение с разделенными переменными $u(x,t) = \sum_{k=1}^{n} a_k(t)\psi_k(x)$. Подстановка этого решения в описывающие уравнения дает

$$\sum \psi_k \frac{da_k}{dt} = \mathbf{N}\left(\sum a_k\psi_k, \sum a_k(\psi_k)_x, \sum a_k(\psi_k)_{xx}, \cdots, x, t, \boldsymbol{\beta}\right), \tag{11.7}$$

где суммирование производится по $k = 1, 2, \cdots, n$. Из ортогональности базисных функций вытекает, что

$$\langle \psi_k, \psi_j \rangle = \delta_{kj} = \begin{cases} 0 & j \neq k \\ 1 & j = k \end{cases}, \tag{11.8}$$

где δ_{kj} – символ Кронекера, а $\langle \psi_k, \psi_j \rangle$ – скалярное произведение, определенное формулой

$$\langle \psi_k, \psi_j \rangle = \int_{-L}^{L} \psi_k \psi_j^* dx, \tag{11.9}$$

в которой * обозначает комплексное сопряжение.

После того как базис определен, уравнения для $a_k(t)$ можно вывести, умножив (11.7) на $\psi_j(x)$ и проинтегрировав по $x \in [-L, L]$. Затем из ортогональности следуют уравнения по времени, или проекция Галеркина, для каждой моды:

$$\frac{da_k}{dt} = \left\langle \mathbf{N}\left(\sum a_j\psi_j, \sum a_j(\psi_j)_x, \sum a_j(\psi_j)_{xx}, \cdots, x, t, \boldsymbol{\beta}\right), \psi_k \right\rangle \tag{11.10}$$

при $k = 1, 2, \cdots, n$.

Заданная форма $\mathbf{N}(\cdot)$ определяет взаимодействие между различными модами. В действительности отличительной особенностью нелинейности является порождение смеси мод в (11.10).

Численные схемы, основанные на проекционном методе Галеркина (11.10), часто применяются для моделирования полной системы (11.1). Сходимости к истинному решению можно добиться с помощью разумного выбора базисных мод ψ_k и общего числа мод n. Интересно, что идея разделения переменных, первоначально относившаяся к *линейным* УрЧП, работает также для *нелинейных* УрЧП и УрЧП с *непостоянными коэффициентами*, при условии что выбрано достаточно базисных функций, чтобы полностью учесть нелинейное смешение мод в (11.10). Хороший выбор элементов базиса позволяет обойтись меньшим числом мод n для достижения желаемой точности. Метод POD предназначен специально для управляемого данными выбора базисных мод, соответствующих конкретной динамике, геометрии и параметрам.

Разложение по модам Фурье

Самый удобный базис для проекционного метода Галеркина – моды Фурье. Точнее, быстрое преобразование Фурье (БПФ) и его варианты доминируют в научных расчетах в инженерных, физических и биологических науках. Тому есть две основные причины: (i) моды Фурье интуитивно хорошо понятны, потому что напрямую связаны с длинами пространственных волн и частотами, и, что еще более важно, (ii) алгоритм вычисления правой части (11.10) имеет сложность $O(n \log n)$. Именно вторая причина сделала БПФ одним из десяти самых широко распространенных алгоритмов минувшего столетия и краеугольным камнем научных расчетов.

Базис Фурье состоит из функций

$$\psi_k(x) = \frac{1}{L}\exp\left(i\frac{2\pi kx}{L}\right), \quad x \in [0, L] \text{ и } k = -n/2, \cdots, -1, 0, 1, \cdots, n/2 - 1. \tag{11.11}$$

Следует отметить, что в большинстве программных пакетов, включая MATLAB, алгоритм БПФ предполагает пространственный интервал $x \in [0, 2\pi]$. Поэтому перед применением БПФ необходимо выполнить масштабирование, перейдя от длины L к длине 2π.

Очевидно, что моды Фурье (11.11) – комплексные периодические функции на отрезке $x \in [0, L]$. Однако разложение применимо к гораздо более широкому классу функций, необязательно периодических. Например, рассмотрим локализованную гауссову функцию

$$u(x,t) = \exp(-\sigma x^2), \tag{11.12}$$

преобразование Фурье которой тоже является гауссовой функцией. Для представления такой функции модами Фурье часто требуется много мод, потому что функция непериодическая. На рис. 11.1 показано разложение гауссовой функции по модам Фурье для трех значений σ. Заметим, что для представления даже этой простой функции требуется большое число мод, особенно когда гауссова ширина уменьшается. Таким образом, решение задач с помощью БПФ зачастую выливается в представления очень высокой размерности ($n \gg 1$), необходимые для аппроксимации локализованного пространственного поведения общего вида. Наша конечная цель – избавиться от такого искусственного порождения задач слишком большой размерности.

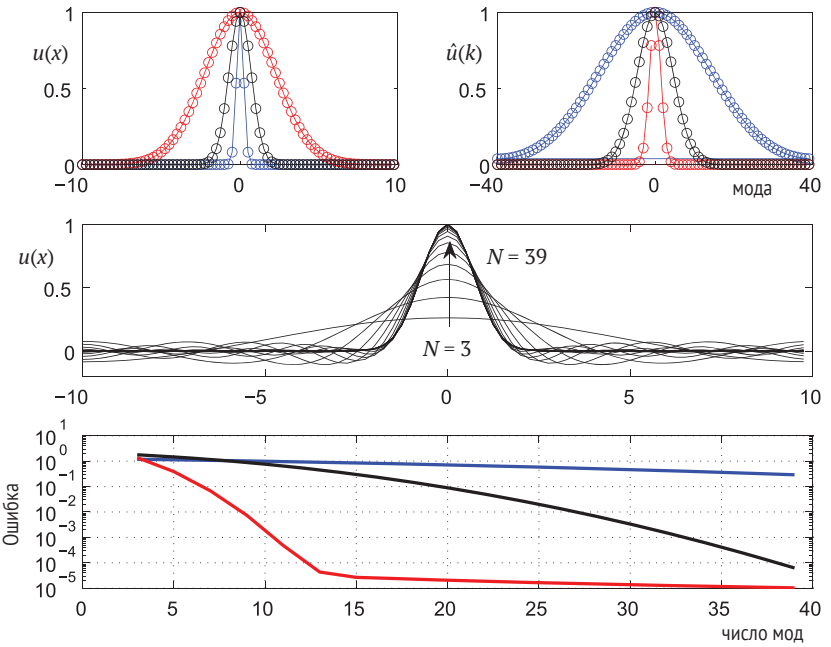

Рис. 11.1 ❖ Моды Фурье для представления локализованной гауссовой функции: (a) $n = 80$ мод Фурье используется для представления функции $u(x) = \exp(-\sigma x^2)$ на отрезке $x \in [-10, 10]$ для $\sigma = 0.1$ (красный), $\sigma = 1$ (черный) и $\sigma = 10$ (синий); (b) разложение по модам Фурье, показывающее, сколько мод необходимо для точного представления локализованной гауссовой функции; (c) сходимость решения с n модами к гауссовой функции ($\sigma = 1$); (d) L^2-отклонение от истинного решения для трех значений σ

Специальные функции и теория Штурма–Лиувилля

В XIX и начале XX века в математической физике были сформулированы многие законы теплопередачи, электромагнетизма, квантовой механики и других физических явлений. Многие важные явления описывались *ли-*

нейной динамикой, что давало возможность найти аналитические решения. А поскольку эти проблемы исследовались до наступления эры компьютеров, нелинейности обычно рассматривались как возмущения базового линейного уравнения. Поэтому часто изучали сложные системы вида

$$\mathbf{u}_t = \mathbf{L}\mathbf{u} + \varepsilon\mathbf{N}(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, \cdots, x, t, \boldsymbol{\beta}), \tag{11.13}$$

где $\mathbf{L}$ – линейный оператор, а $\varepsilon \ll 1$ – малый параметр для включения возмущений. В математической физике $\mathbf{L}$ часто является оператором Штурма–Лиувилля, собственные значения и собственные функции которого обладают многими полезными свойствами.

Для решения уравнений вида (11.13) нередко используются специальные моды, идеально подходящие для конкретной проблемы. Они являются собственными функциями соответствующего линейного оператора $\mathbf{L}$ в уравнении (11.13):

$$\mathbf{L}\psi_k = \lambda_k\psi_k, \tag{11.14}$$

где $\psi_k(x)$ – ортонормированные собственные функции оператора $\mathbf{L}$. При этом возможно решение в виде *разложения по собственным функциям* $\mathbf{u}(x, t) = \sum a_k(t)\psi_k(x)$. Это дает решение в таком виде:

$$\frac{da_k}{dt} = \langle \mathbf{L}\mathbf{u}, \psi_k \rangle + \varepsilon\langle \mathbf{N}, \psi_k \rangle. \tag{11.15}$$

Суть подобного разложения заключается в том, что собственные функции, возможно, идеально подходят для моделирования пространственных вариаций, специфических для рассматриваемой задачи. Поэтому они являются идеальными модами для (11.13). В этом и состоит отличие от разложения в ряд Фурье, поскольку синусоидальные моды могут быть совершенно не связаны с физикой явлений или геометрическими симметриями. Так, для представления гауссовой функции потенциально более эффективны многочлены Эрмита. На самом деле для эффективного представления решений в форме оператора Штурма–Лиувилля, тесно связанного с физикой и геометрией изучаемого явления, разработано много специальных функций, в т. ч. функции *Бесселя*, *Лагерра*, *Эрмита*, *Лежандра*. Использование таких функций можно считать способом *понижения размерности* посредством идеально подобранных базисных функций.

Понижение размерности

Приведенные выше примеры и методы решения УрЧП иллюстрируют общую проблему научных расчетов: появление систем высокой размерности n. Для многих сложных УрЧП с несколькими пространственными переменными не редкость, когда дискретизация или разложения по модам дают системы дифференциальных уравнений с миллионами или миллиардами степеней свободы. Столь большие системы очень трудно решать даже на самых со-

временных компьютерах, поэтому точность моделирования таких сложных систем, как течение жидкости с большим числом Рейнольдса, ограничено, а время расчетов велико.

Для упрощения вычислений критически важно выбирать набор оптимальных базисных мод, поскольку при этом количество дифференциальных уравнений может сильно сократиться. Во многих методах возникают линейные системы размерности n, для решения которых в общем случае требуется $O(n^3)$ операций. Поэтому важность уменьшения n невозможно переоценить. Мы уже видели, что еще в XIX и начале XX века для различных задач математической физики были разработаны специальные функции, это была аналитическая попытка найти идеальный набор мод для представления динамики сложной системы. Но для существенно нелинейных систем вида (11.1) даже такие специальные функции редко дают наилучший набор мод. В следующем разделе мы покажем, как сгенерировать моды ψ_k, специально приспособленные к динамике и геометрии уравнения (11.1). Основанное на алгоритме SVD, *собственное ортогональное разложение* (POD) порождает набор мод, *оптимальный* для представления данных измерений или имитационного моделирования, что потенциально позволяет значительно уменьшить число мод n, необходимых для моделирования поведения (11.1) с заданной точностью [57, 542, 543].

11.2. Элементы оптимального базиса: собственное ортогональное разложение

В предыдущем разделе мы видели, что выбор хорошего базиса для решения уравнения (11.1) с помощью разложения Галеркина (11.6) критически важен для эффективных научных расчетов. Многие алгоритмы решения УрЧП предполагают, что базисные моды выбраны заранее с учетом (i) скорости вычислений, (ii) точности и (или) (iii) ограничений на граничные условия. Все эти факторы убедительно обоснованы и составляют основу практически реализуемых вычислительных методов. Но в этой главе нас будет интересовать метод, который позволяет достичь максимальной вычислительной эффективности посредством *понижения размерности*. Как уже было сказано, многие алгоритмы порождают искусственно большие системы размерности n. Ниже мы представим управляемую данными стратегию, при которой оптимальные моды, называемые также POD-модами, выбираются на основе наблюдений – экспериментальных или полученных в ходе численного моделирования, – что позволяет найти минимальное число мод $r \ll n$, достаточное для характеризации динамики (11.1).

Существует два способа выделить оптимальные базисные моды для данной сложной системы. Первый – собрать данные экспериментально, второй – выполнить имитационное моделирование системы и производить выборку состояния в ходе ее эволюции в соответствии с динамикой. В обоих случаях берутся снимки динамики и идентифицируются оптимальные моды. Можно

возразить, что если для выделения мод производится моделирование системы, то мы не получаем никакого вычислительного выигрыша. Однако, как и в случае LU-разложения, для которого стоимость однократных начальных вычислений составляет $O(n^3)$, а стоимость последующих операций – $O(n^2)$, дорогостоящий процесс выделения мод выполняется только один раз. А затем найденные оптимальные моды можно эффективно использовать для вычислений.

Для построения оптимальных POD-мод из динамики (11.1) производится выборка с заранее заданным интервалом. Снимок $\mathbf{u}_k$ содержит выборку состояния сложной системы в момент времени t_k:

$$\mathbf{u}k := [\mathbf{u}(x_1, t_k) \quad \mathbf{u}(x_2, t_k) \quad \cdots \quad \mathbf{u}(x_n, t_k)]^T.$$

Затем в n дискретных точках вычисляются непрерывные функции и моды, что дает представление в виде вектора высокой размерности; мы будем обозначать такие представления полужирными буквами. В общем случае нас интересует анализ большого набора данных $\mathbf{X}$, полученного экспериментально или численно:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \\ | & | & & | \end{bmatrix}, \tag{11.16}$$

где столбцы $\mathbf{u}_k = \mathbf{u}(t_k) \in \mathbb{C}^n$ содержат результаты измерений. $\mathbf{X}$ состоит из *временных рядов* данных, каждый ряд содержит измерения в m моментов времени. Зачастую *размерность пространства состояний* n очень велика, для гидродинамических систем она может быть порядка миллиона или миллиарда. В типичном случае $n \gg m$, т. е. мы получаем *высокую и тощую* матрицу, в отличие от *низкой и толстой* в случае $n \ll m$.

Как уже отмечалось, сингулярное разложение (SVD) дает единственное разложение произвольной комплексной матрицы $\mathbf{X} \in \mathbb{C}^{n\times m}$:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \tag{11.17}$$

где $\mathbf{U} \in \mathbb{C}^{n\times n}$ и $\mathbf{V} \in \mathbb{C}^{m\times m}$ – *унитарные* матрицы, а $\mathbf{\Sigma} \in \mathbb{C}^{n\times m}$ – матрица с неотрицательными элементами на диагонали. Здесь * обозначает операцию комплексного сопряжения и транспонирования. Столбцы $\mathbf{U}$ называются *левыми сингулярными векторами* $\mathbf{X}$, а столбцы $\mathbf{V}$ – правыми сингулярными векторами. Элементы на диагонали $\mathbf{\Sigma}$ называются *сингулярными значениями*, они расположены в порядке убывания. SVD играет важнейшую роль в построении оптимального базиса для конкретной задачи. В частности, гарантируется, что матрица $\mathbf{U}$ дает наилучший набор мод для аппроксимации $\mathbf{X}$ по норме ℓ_2. Точнее, столбцы этой матрицы содержат ортогональные моды, необходимые для формирования идеального базиса. Матрица $\mathbf{V}$ дает историю мод во времени, а диагональная матрица $\mathbf{\Sigma}$ определяет относительные веса мод. Напомним, что моды упорядочены в порядке убывания, так что наиболее важные расположены в начале, а наименее важные – в конце.

Общее число сгенерированных мод обычно определяется числом снимков m, сделанных при построении $\mathbf{X}$ (как правило, $n \gg m$). Наша цель – определить минимальное число мод, необходимое для точного представления динамики (11.1) с помощью проекционного метода Галеркина (11.6). То есть нас интересует аппроксимация ранга r истинной динамики, где $r \ll m$, иначе говоря – низкоранговое SVD-разложение

$$\tilde{\mathbf{X}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^{*}, \tag{11.18}$$

где $||\mathbf{X} - \tilde{\mathbf{X}}|| < \varepsilon$ для заданного малого ε. Это разложение позволяет построить интересующие нас моды $\boldsymbol{\psi}_k$ по столбцам матрицы $\tilde{\mathbf{U}}$. В частности, оптимальные базисные моды имеют вид

$$\tilde{\mathbf{U}} = \mathbf{\Psi} = \begin{bmatrix} | & | & & | \\ \boldsymbol{\psi}_1 & \boldsymbol{\psi}_2 & \cdots & \boldsymbol{\psi}_m \\ | & | & & | \end{bmatrix} \tag{11.19}$$

и получаются в результате усечения, сохраняющего r доминирующих мод в (11.6). Оставленные r мод $\{\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \cdots, \boldsymbol{\psi}_r\}$ используются в качестве ортогонального базиса низкого ранга для представления динамики (11.1).

Описанный метод выделения низкорангового r-мерного подпространства на основе снимков для системы с динамикой (11.1) представляет собой пример управляемой данными архитектуры. В самом деле, никакие уравнения в этом методе не используются, т. е. описывающее уравнение (11.1) может быть неизвестно. В том случае, когда истинная динамика неизвестна, выделение низкорангового подпространства открывает возможность построить потенциальную модель в r-мерном подпространстве вместо модели в пространстве гораздо более высокой размерности $n \gg r$. Далее мы разовьем эти идеи. Но сейчас достаточно отметить, что для получения оптимального базиса знание об истинной природе сложной системы (11.1) необязательно.

Проекция Галеркина на POD-моды

Мы можем аппроксимировать состояние $\mathbf{u}$ УрЧП с помощью разложения Галеркина:

$$\mathbf{u}(t) \approx \mathbf{\Psi}\mathbf{a}(t), \tag{11.20}$$

где $\mathbf{a}(t) \in \mathbb{R}^r$ – зависящий от времени вектор коэффициентов и $r \ll n$. Подставив это разложение по модам в уравнение (11.13) и воспользовавшись ортогональностью (путем умножения на $\mathbf{\Psi}^T$), получаем уравнение эволюции пониженной размерности

$$\frac{d\mathbf{a}(t)}{dt} = \mathbf{\Psi}^T\mathbf{L}\mathbf{\Psi}\mathbf{a}(t) + \mathbf{\Psi}^T\mathbf{N}(\mathbf{\Psi}\mathbf{a}(t), \boldsymbol{\beta}). \tag{11.21}$$

Решив это уравнение гораздо меньшей размерности, мы аппроксимируем решение исходной нелинейной динамической системы. Очень важно вычис-

лять нелинейные члены эффективно, пользуясь архитектурами неполного POD или DEIM, описанными в главе 12. В противном случае придется, как и раньше, вычислять функции и скалярные произведения в пространстве исходной размерности n. Иногда, например в случае квадратичной нелинейности в уравнениях Навье–Стокса, нелинейные члены можно вычислить один раз в автономном режиме. Но при работе с параметрическими системами обычно требуется вычислять нелинейные члены многократно, поскольку POD-моды изменяются вместе с $\boldsymbol{\beta}$.

Пример: гармонический осциллятор

Для иллюстрации применения метода POD к выбору оптимального базиса рассмотрим классическую задачу математической физики: *квантовый гармонический осциллятор*. Идеальные базисные функции (многочлены Эрмита) для этой задачи известны, но мы хотели бы вывести эти специальные функции с помощью одного лишь анализа данных. Вопрос: как это сделать, имея лишь моментальные снимки динамики? Стандартный гармонический осциллятор возникает при изучении системы пружина–масса. Предполагается, что возвращающая сила пружина F подчиняется линейному закону Гука:

$$F(t) = -kx, \tag{11.22}$$

где k – постоянный коэффициент, а $x(t)$ – отклонение пружины от положения равновесия. Этой силе соответствует потенциальная энергия пружины $V = kx^2/2$ (без ограничения общности можно предположить, что $k = 1$), что приводит к уравнению Шрёдингера с параболическим потенциалом:

$$iu_t + \frac{1}{2}u_{xx} - \frac{x^2}{2}u = 0, \tag{11.23}$$

где второй член дифференциального уравнения в частных производных представляет кинетическую энергию квантовой частицы, а последний член – параболический потенциал, соответствующий линейной возвращающей силе.

Решение задачи о квантовом гармоническом осцилляторе легко выразить в терминах специальных функций. Предположив, что решение имеет вид

$$u(x, t) = a_k\psi_k(x)\exp[-i(k + 1/2)t], \tag{11.24}$$

где a_k определяется из начальных условий, получаем следующую граничную задачу для нахождения собственных мод системы:

$$\frac{d^2\psi_k}{dx^2} + (2k + 1 - x^2)\psi_k \tag{11.25}$$

с граничными условиями $\psi_k \to 0$ при $x \to \pm\infty$. Нормированные решения этого уравнения можно выразить через *многочлены Эрмита* $H_k(x)$:

$$\psi_k = \left(2^k k\sqrt{\pi}\right)^{-1/2} \exp(-x^2/2) H_k(x) \tag{11.26a}$$

$$= (-1)^k \left(2^k k\sqrt{\pi}\right)^{-1/2} \exp(-x^2/2) \frac{d^k}{dx^k} \exp(-x^2). \tag{11.26b}$$

Многочлены Эрмита обычно считаются оптимальными базисными функциями для гармонического осциллятора, потому что они естественно представляют истинную динамику, описываемую уравнением Шрёдингера с параболическим потенциалом. Действительно, решения сложной системы (11.23) можно представить в виде суммы

$$u(x,t) = \sum_{k=0}^{\infty} a_k \left(2^k k\sqrt{\pi}\right)^{-1/2} \exp(-x^2/2) H_k \exp[-i(k+1/2)t]. \tag{11.27}$$

Такой подход к решению сплошь и рядом встречается в математической физике, о чем свидетельствует большое количество специальных функций, часто в форме Штурма–Лиувилля, для различных геометрий и граничных условий. В качестве примеров упомянем функции Бесселя, многочлены Лагерра, многочлены Лежандра, функции параболического цилиндра, сферические гармоники и т. д.

Численное решение УрЧП (11.23), основанное на быстром преобразовании Фурье, легко реализуется [316]. Программа в листинге 11.1 находит полное численное решение с начальным условием $u(x, 0) = \exp(-0.2(x - x_0)^2)$ в виде гауссовой функции с центром в точке $x = x_0$. При таком начальном условии возбуждается ряд мод, описываемых многочленами Эрмита. В частности, начальная проекция на собственные моды вычисляется из условий ортогональности:

$$a_k = \langle u(x, 0), \psi_k \rangle. \tag{11.28}$$

Это скалярное произведение проецирует начальное условие на каждую из мод ψ_k.

Листинг 11.1 ❖ Код гармонического осциллятора

```
L=30; n=512; x2=linspace(-L/2,L/2,n+1); x=x2(1:n); % пространственная дискретизация
k=(2*pi/L)*[0:n/2-1 -n/2:-1].'; % волновые числа для БПФ
V=x.^2.';              % потенциал
t=0:0.2:20;            % точки во временной области
u=exp(-0.2*(x-1).^2); % начальные условия
ut=fft(u);             % БПФ начальных данных
[t,utsol]=ode45('pod_harm_rhs',t,ut,[],k,V); % проинтегрировать УрЧП
for j=1:length(t)
    usol(j,:)=ifft(utsol(j,:)); % обратное преобразование
end
```

Функция в правой части, **pod_harm_rhs.m**, упомянутая в этой программе, содержит уравнение (11.23), записанное тремя строчками на MATLAB.

Листинг 11.2 ❖ Правая часть гармонического осциллятора

```
function rhs=pod_harm_rhs(t,ut,dummy,k,V)
u=ifft(ut);
rhs=-(i/2)*(k.^2).*ut - 0.5*i*fft(V.*u);
```

Вместе обе программы порождают динамику квантового гармонического осциллятора. На рис. 11.2 показана эволюция системы с начальным условием $u(x, 0) = \exp(-0.2(x - x_0)^2)$ при $x_0 = 0$ (слева) и $x_0 = 1$ (справа). Из результатов моделирования видно, что всего собран 101 снимок (начальное условие и измерения в 100 моментов времени). Эти снимки можно организовать в виде (11.16) и выполнить сингулярное разложение. Полученные сингулярные значения позволяют сделать предположения об истинной размерности динамики. Сингулярные значения, соответствующие эволюции на верхних рисунках, показаны на рисунках внизу. В случае симметричного (относительно $x = 0$) начального условия доминирует пять мод. В случае же асимметричного начального условия для представления динамики с той же точностью понадобилось вдвое больше мод.

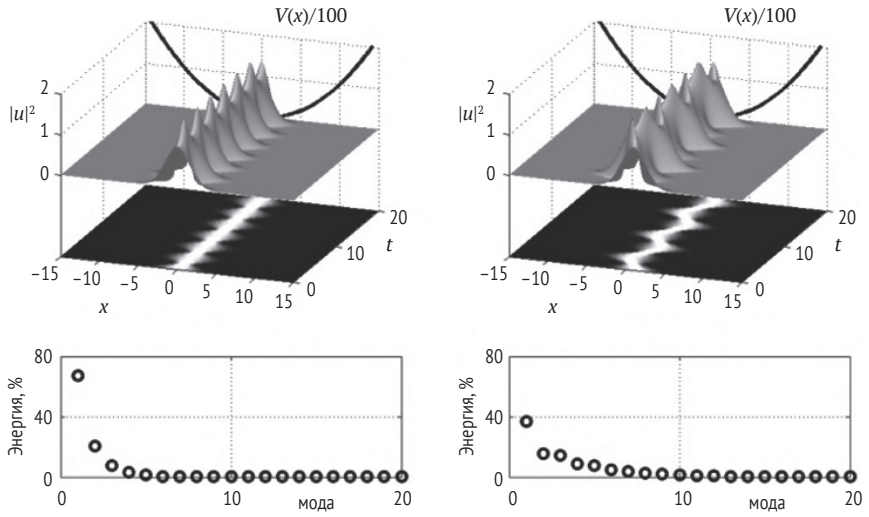

Рис. 11.2 ❖ Динамика квантового гармонического осциллятора (11.23) с начальным условием $u(x, 0) = \exp(-0.2(x - x_0)^2)$ при $x_0 = 0$ (слева) и $x_0 = 1$ (справа). При симметричных начальных данных выделяется пять доминирующих мод, а для начального условия $x_0 = 1$ активируется десять мод. В нижней части показаны сингулярные значения, соответствующие верхним рисункам, и процент энергии (по норме L^2) в каждой моде. Учитывая быстрое затухание сингулярных значений, очевидно, что динамика низкоранговая

Сингулярное разложение дает не только распределение энергии в первых модах, но и оптимальный базис в виде столбцов матрицы **U**. Распределение

сингулярных значений наглядно показывает, как выбрать низкоранговое подпространство, натянутое на r мод, и, значит, позволяет построить пространство пониженной размерности (11.19), подходящее для разложения POD–Галеркина.

Моды квантового гармонического осциллятора показаны на рис. 11.3. Точнее, показаны первые пять мод для (i) многочленов Эрмита, представляющих решение, (ii) мод, полученных с помощью SVD для симметричного ($x_0 = 0$) начального условия, (iii) мод, полученных с помощью SVD для асимметричного ($x_0 = 1$) начального условия. Многочлены Эрмита по построению соответствуют собственным значениям оператора Штурма–Лиувилля (11.25), упорядоченным по убыванию. Собственные моды для симметричного и асимметричного начальных условий выглядят по-разному.

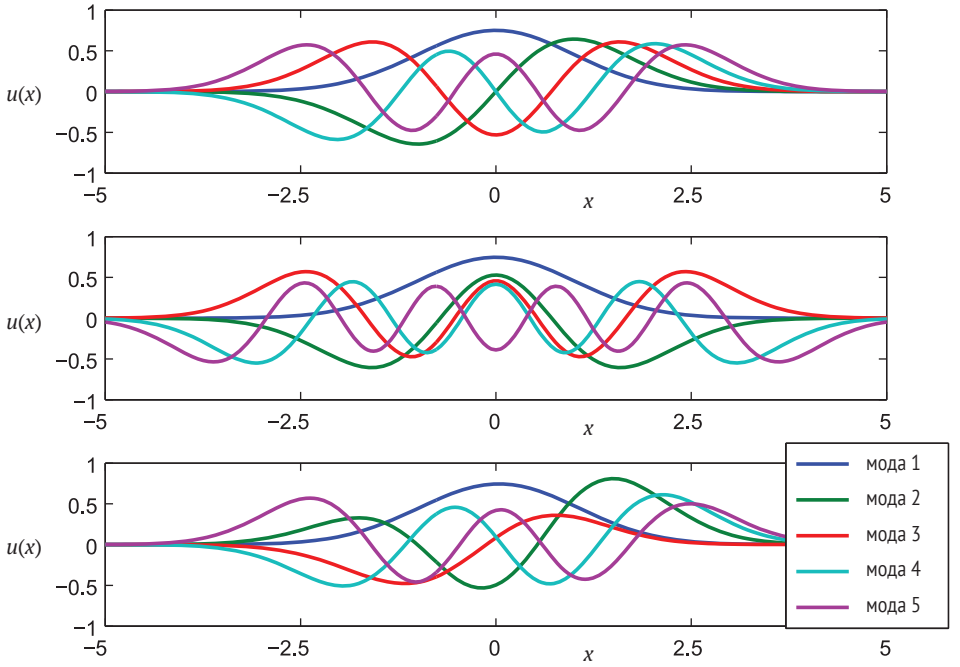

Рис. 11.3 ❖ Первые пять мод квантового гармонического осциллятора. На верхнем рисунке показаны первые пять многочленов Эрмита (11.26), соответствующих собственным значениям оператора Штурма–Лиувилля. На втором рисунке показаны доминирующие моды, вычисленные по сингулярному разложению динамики гармонического осциллятора с начальным условием $u(x, 0) = \exp(-0.2x^2)$, показанному в левой части рис. 11.2. Заметим, что все моды симметричны, поскольку никакой асимметричной динамики действительно не наблюдается. На нижнем рисунке, где показаны результаты моделирования гармонического осциллятора с начальным условием $u(x, 0) = \exp(-0.2(x - 1)^2)$, асимметрия, очевидно, присутствует. При этом также порождаются моды, очень похожие на многочлены Эрмита. Таким образом, метод, основанный исключительно на анализе снимков, смог воспроизвести почти идеальный базис для гармонического осциллятора

В симметричном (относительно $x = 0$) случае, когда $u(x, 0) = \exp(-0.2x^2)$, все первые пять мод симметричны, потому что метод, основанный на анализе снимков, не может породить асимметричные моды, т. к. они отсутствуют в динамике и, следовательно, не наблюдаемы. С другой стороны, даже при небольшом смещении, $u(x, 0) = \exp(-0.2(x - 1)^2)$, снимки эволюции порождают асимметричные моды, близкие к асимметричным модам разложения по многочленам Эрмита. Интересно, что в этом случае SVD упорядочивает моды по количеству содержащейся в них энергии. Таким образом, первая асимметричная мода (красная мода 3 на нижнем рисунке) эквивалентна второй моде точного решения в виде многочленов Эрмита (зеленая мода на верхнем рисунке). Ключевое наблюдение заключается в том, что метод, основанный на анализе снимков, оказался очень близок к отысканию многочленов Эрмита, которые, как известно, оптимальны для данной системы. Важно, что метод POD–Галеркина обобщается на более сложные физические и геометрические конфигурации, для которых решение заранее неизвестно.

11.3. POD и динамика солитонов

Для иллюстрации полной реализации метода POD–Галеркина мы рассмотрим систему с существенно нелинейной динамикой, а именно нелинейное уравнение Шрёдингера (НУШ)

$$iu_t + \frac{1}{2}u_{xx} + |u|^2 u = 0 \tag{11.29}$$

с граничными условиями $u \to 0$ при $x \to \pm\infty$. Если бы не нелинейный член, то это уравнение можно было бы решить в замкнутой форме. Однако из-за нелинейности члены разложения по собственным функциям (11.6) смешиваются, и найти простое аналитическое решение невозможно.

Для численного решения НУШ используется разложение по модам Фурье. Поэтому можно воспользоваться быстрым преобразованием Фурье. Если переписать (11.29) в частотной области, применив преобразование Фурье к обеим частям, то получим систему дифференциальных уравнений

$$\hat{u}_t = -\frac{i}{2}k^2\hat{u} + i\widehat{|u|^2 u}, \tag{11.30}$$

в которой из-за нелинейного смешения в кубическом члене имеет место смешение мод Фурье. Поэтому для вычисления динамики НУШ нам нужно решить систему дифференциальных уравнений.

В приведенной ниже программе решение этого УрЧП ищется в виде разложения (11.6) по собственным функциям НУШ (11.29). Первый шаг – определить интересующую нас пространственно-временную область решения и частоты Фурье, присутствующие в системе. Это делается в следующем коде.

Листинг 11.3 ❖ Решение нелинейного уравнения Шрёдингера

```
L=40; n=512; x2=linspace(-L/2,L/2,n+1); x=x2(1:n); % пространственная дискретизация
k=(2*pi/L)*[0:n/2-1 -n/2:-1].'; % волновые числа для БПФ
t=linspace(0,2*pi,21);          % точки во временной области
N=1;
u=N*sech(x);   % начальные условия
ut=fft(u);     % БПФ начальных данных
[t,utsol]=ode45('pod_sol_rhs',t,ut,[],k); % проинтегрировать УрЧП
for j=1:length(t)
usol(j,:)=ifft(utsol(j,:));               % обратное преобразование
end
```

Функция в правой части, **pod_sol_rhs.m**, упомянутая в этой программе, содержит уравнение (11.29), записанное тремя строчками на MATLAB.

Листинг 11.4 ❖ Правая часть НУШ

```
function rhs=pod_sol_rhs(t,ut,dummy,k)
u=ifft(ut);
rhs=-(i/2)*(k.^2).*ut + i*fft( (abs(u).^2).*u );
```

Остается рассмотреть конкретную пространственную конфигурацию для начального условия. В случае НУШ имеется множество специальных начальных условий, называемых солитонами, вида

$$u(x, 0) = N\text{sech}(x), \tag{11.31}$$

где N – целое число. Мы будем рассматривать динамику солитонов для $N = 1$ и $N = 2$. Прежде всего начальное условие проецируется на моды Фурье с помощью быстрого преобразования Фурье. Динамика солитонов с $N = 1$ и $N = 2$ показана на рис. 11.4. В процессе эволюции солитон с $N = 1$ претерпевает изменение фазы, тогда как его амплитуда остается постоянной. Напротив, солитон с $N = 2$ совершает периодические колебания. В обоих случаях для моделирования простого поведения требуется много мод Фурье, приблизительно 50 и 200.

В свете интересующего нас понижения размерности возникает простой вопрос: действительно ли динамика солитона описывается системой с 50 или 200 степенями свободы, как то диктует разложение по модам Фурье? Ответ – нет. Можно показать, что в подходящем базисе, определяемом POD-модами, найденными с помощью SVD, динамика сводится всего к 1 и 2 модам соответственно. Достаточно вычислить сингулярное разложение эволюций, показанных на рис. 11.4, и убедиться, что при $N = 1$ и $N = 2$ солитоны и в самом деле имеют низкую размерность.

На рис. 11.5 наглядно продемонстрирована низкоразмерная природа численных решений, для чего были вычислены сингулярные значения и моды, используемые в нашем разложении по собственным функциям. В обоих случаях динамика действительно низкоразмерная: солитон с $N = 1$ хорошо моделируется одной POD-модой, а солитон с $N = 2$ – двумя POD-модами. Таким

образом, при выполнении разложения по собственным функциям в качестве мод следует выбирать POD-моды, сгенерированные на основе самих данных имитационного моделирования. В следующем разделе мы выведем динамику взаимодействия мод для этих двух случаев – низкоразмерную и пригодную для анализа.

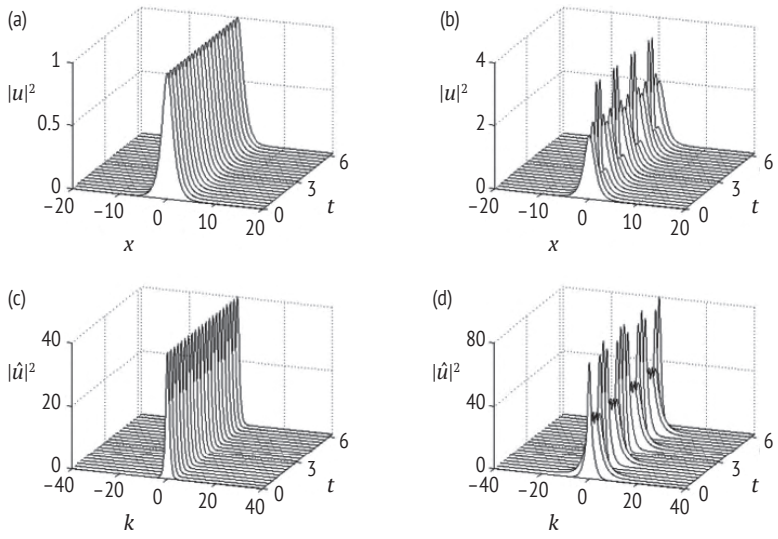

Рис. 11.4 ❖ Эволюция солитонов с (a) $N = 1$ и (b) $N = 2$. На рисунках (a) и (c) показана стационарная динамика ($N = 1$), а на рисунках (b) и (d) – периодическая. Для моделирования поведения понадобилось соответственно 50 и 200 мод Фурье

Упрощение солитона ($N = 1$)

Чтобы воспользоваться низкоранговой структурой, рассмотрим сначала динамику солитона с $N = 1$. На рис. 11.5 показано, что в динамике доминирует одна мода, полученная с помощью SVD. Это первый столбец матрицы **U**. Таким образом, динамика сводится к одной моде, т. е.

$$u(x, t) = a(t)\psi(x). \tag{11.32}$$

Подставляя это в НУШ (11.29), получаем:

$$ia_t\psi + \frac{1}{2}a\psi_{xx} + |a|^2 a|\psi|^2\psi = 0. \tag{11.33}$$

Вычисление скалярного произведения с ψ дает

$$ia_t + \frac{\alpha}{2}a + \beta|a|^2 a = 0, \tag{11.34}$$

где

$$\alpha = \frac{\langle \psi_{xx}, \psi \rangle}{\langle \psi, \psi \rangle}; \tag{11.35a}$$

$$\beta = \frac{\langle |\psi|^2 \psi, \psi \rangle}{\langle \psi, \psi \rangle}. \tag{11.35b}$$

Это низкоранговая аппроксимация, полученная методом POD–Галеркина.

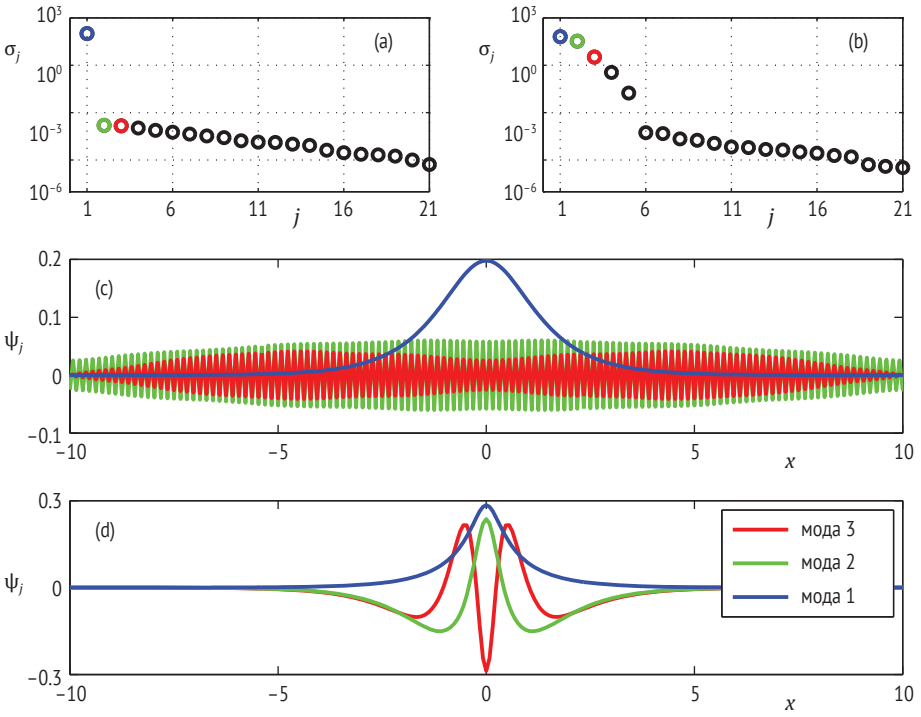

Рис. 11.5 ❖ Проекция эволюции солитонов с $N = 1$ и $N = 2$ на их POD-моды. На рисунках (a) и (b) показаны сингулярные значения σ_j в логарифмическом масштабе для двух эволюций рис. 11.4. Видно, что динамика солитонов с $N = 1$ и $N = 2$ по преимуществу низкоранговая, причем в случае $N = 1$ доминирует одна мода, а в случае $N = 2$ – две моды, содержащие приблизительно 95 % всего изменения в процессе эволюции. На рисунках (c) и (d) показаны первые три моды в обоих случаях

Дифференциальное уравнение (11.34) относительно $a(t)$ можно решить явно, получаем:

$$a(t) = a(0)\exp\left(i\frac{\alpha}{2}t + \beta|a(0)|^2 t\right), \tag{11.36}$$

где $a(0)$ – начальное условие для $a(t)$. Чтобы найти начальное условие, вспомним, что

$$u(x, 0) = \operatorname{sech}(x) = a(0)\psi(x). \tag{11.37}$$

Скалярное умножение на $\psi(x)$ дает

$$a(0) = \frac{\langle \operatorname{sech}(x), \psi \rangle}{\langle \psi, \psi \rangle}. \tag{11.38}$$

Таким образом, разложение с одной модой дает приближенное решение УрЧП:

$$u(x,t) = a(0)\exp\left(i\frac{\alpha}{2}t + \beta|a(0)|^2 t\right)\psi(x). \tag{11.39}$$

Это решение – не что иное, как низкоразмерная POD-аппроксимация УрЧП в наилучшем возможном базисе, каковым является SVD-базис.

Для солитона с $N = 1$ пространственный профиль остается постоянным, тогда как фаза претерпевает нелинейное вращение. POD-решение (11.39) можно вычислить явно и точно охарактеризовать это вращение фазы.

Упрощение солитона ($N = 2$)

Солитон с $N = 2$ несколько сложнее, но и интереснее. В этом случае в поведении системы явно доминируют две моды, содержащие 96 % всей энергии. Эти моды, ψ_1 и ψ_2, являются первыми двумя столбцами матрицы $\mathbf{U}$ и используются для аппроксимации динамики, наблюдаемой на рис. 11.4. Разложение с двумя модами имеет вид:

$$u(x, t) = a_1(t)\psi_1(x) + a_2(t)\psi_2(x). \tag{11.40}$$

Подставляя это выражение в уравнение (11.29), получаем:

$$i(a_{1_t}\psi_1 + a_{2_t}\psi_2) + \frac{1}{2}(a_1\psi_{1_{xx}} + a_2\psi_{2_{xx}}) + (a_1\psi_1 + a_2\psi_2)^2(a_1^*\psi_1^* + a_2^*\psi_2^*) = 0. \tag{11.41}$$

Раскрытие кубического члена дает:

$$\begin{aligned} &i(a_{1_t}\psi_1 + a_{2_t}\psi_2) + \frac{1}{2}(a_1\psi_{1_{xx}} + a_2\psi_{2_{xx}}) \\ &\quad + (|a_1|^2 a_1|\psi_1|^2\psi_1 + |a_2|^2 a_2|\psi_2|^2\psi_2) + 2|a_1|^2 a_2|\psi_1|^2\psi_2 + 2|a_2|^2 a_1|\psi_2|^2\psi_1 \\ &\quad + a_1^2 a_2^*\psi_1^2\psi_2^* + a_2^2 a_1^*\psi_2^2\psi_1^*). \end{aligned} \tag{11.42}$$

Осталось только вычислить скалярное произведение обеих частей этого уравнения на $\psi_1(x)$ и $\psi_2(x)$. Вспоминая, что эти моды ортогональны, получаем следующую систему двух уравнений с двумя неизвестными:

$$\begin{aligned} &ia_{1_t} + \alpha_{11}a_1 + \alpha_{12}a_2 + (\beta_{111}|a_1|^2 + 2\beta_{211}|a_2|^2)a_1 \\ &\quad + (\beta_{121}|a_1|^2 + 2\beta_{221}|a_2|^2)a_2 + \sigma_{121}a_1^2 a_2^* + \sigma_{211}a_2^2 a_1^* = 0 \end{aligned} \tag{11.43a}$$

$$ia_{2_t} + \alpha_{21}a_1 + \alpha_{22}a_2 + (\beta_{112}|a_1|^2 + 2\beta_{212}|a_2|^2)a_1 \\ + (\beta_{122}|a_1|^2 + 2\beta_{222}|a_2|^2)a_2 + \sigma_{122}a_1^2a_2^* + \sigma_{212}a_2^2a_1^* = 0, \tag{11.43b}$$

где

$$\alpha_{jk} = \langle \psi_{jxx}, \psi_k \rangle / 2, \tag{11.44a}$$

$$\beta_{jkl} = \langle |\psi_j|^2 \psi_k, \psi_l \rangle, \tag{11.44b}$$

$$\sigma_{jkl} = \langle \psi_j^2 \psi_k^*, \psi_l \rangle, \tag{11.44c}$$

а начальные значения обеих компонент равны

$$a_1(0) = \frac{\langle 2\mathrm{sech}(x), \psi_1 \rangle}{\langle \psi_1, \psi_1 \rangle}; \tag{11.45a}$$

$$a_2(0) = \frac{\langle 2\mathrm{sech}(x), \psi_2 \rangle}{\langle \psi_2, \psi_2 \rangle}. \tag{11.45b}$$

Таким образом, мы имеем полное описание динамики с двумя модами, полученными с помощью сингулярного разложения.

Двухмодовая динамика хорошо аппроксимирует решение. Однако имеет место дрейф фазы, для устранения которого необходима более высокая точность временного ряда полного УрЧП и более точное интегрирование скалярных произведений, описывающих коэффициенты. Действительно, для вычисления скалярных произведений мы использовали простейшую формулу трапеций, точность которой вызывает подозрения; этот вопрос мы обсудим в следующем разделе. Схемы более высокого порядка, безусловно, повысят точность. Кроме того, было бы полезно включить третью и даже более высокие моды. Так или иначе, мы продемонстрировали, как с помощью низкоразмерных структур можно аппроксимировать динамику УрЧП на практике.

11.4. POD в непрерывной формулировке

До сих пор мы строили POD-упрощение на основе снимков $\mathbf{X}$, содержащих дискретные измерения (см. формулу (11.16)). Результатом POD-упрощения является множество мод $\mathbf{\Psi}$, образующее низкоранговый базис, для которого достигает минимума следующая среднеквадратическая ошибка:

$$\underset{\mathbf{\Psi} \text{ такое, что } \mathrm{rank}(\mathbf{\Psi})=r}{\mathrm{argmin}} \; ||\mathbf{X} - \mathbf{\Psi}\mathbf{\Psi}^T\mathbf{X}||_F. \tag{11.46}$$

Напомним, что $\mathbf{X} \in \mathbb{C}^{n \times m}$ и $\mathbf{\Psi} \in \mathbb{C}^{n \times r}$, где r – ранг упрощенной системы.

Во многих случаях измерения производятся непрерывным процессом в заданной пространственной области, так что рассматриваемые данные строятся на основе траекторий

$$u(x, t), \quad t \in [0, T], x \in [-L, L]. \tag{11.47}$$

Для таких данных необходима *непрерывная* по времени формулировка POD-упрощения. В частности, для таких непрерывных траекторий нужно предложить эквивалент выражения (11.46). Заметим, что вместо зависящей от пространственной переменной функции $u(x, t)$ можно рассматривать вектор траекторий $\mathbf{u}(t) \in \mathbb{C}^n$. Подобная необходимость может возникнуть, когда УрЧП дискретизировано, так что бесконечномерная переменная x становится конечномерной. В работах Wolkwein [542, 543] приведен великолепный технический обзор метода POD и его непрерывной формулировки.

Чтобы определить непрерывную формулировку, введем скалярное произведение

$$\langle f(x), g(x)\rangle = \int_{-L}^{L} f(x)g^{*}(x)dx. \tag{11.48}$$

Для нахождения функции, которая лучше всего аппроксимирует полную траекторию во времени $u(x, t)$ в (11.47), необходимо решить следующую задачу минимизации:

$$\min_{\psi} \frac{1}{T}\int_0^T \left|\langle u(x,t), \psi(x)\rangle\right|^2 dt \quad \text{при условии} \quad ||\psi||^2 = 1, \tag{11.49}$$

где нормировка интеграла по времени на $1/T$ нужна для усреднения разности между истинными данными и их низкоранговой аппроксимацией с помощью функции ψ на отрезке $t \in [0, T]$. Выражение (11.49) эквивалентно максимизации скалярного произведения данных $u(x, t)$ на функцию $\psi(x)$, т. е. они максимально параллельные в пространстве функций. Поэтому задачу минимизации можно переформулировать в виде

$$\max_{\psi} \frac{1}{T}\int_0^T \left|\langle u(x,t), \psi(x)\rangle\right|^2 dt \quad \text{при условии} \quad ||\psi||^2 = 1. \tag{11.50}$$

Задача ограниченной оптимизации (11.50) может быть записана как функционал Лагранжа

$$\mathcal{L}(\psi, \lambda) = \frac{1}{T}\int_0^T \left|\langle u(x,t), \psi(x)\rangle\right|^2 dt + \lambda(1 - ||\psi||^2), \tag{11.51}$$

где λ – множитель Лагранжа, соответствующий ограничению $||\psi||^2 = 1$. Это выражение можно переписать в виде

$$\begin{aligned}\mathcal{L}(\psi, \lambda) = \frac{1}{T}\int_0^T \left(\int_{-L}^{L} u(\xi,t)\psi^{*}(\xi)d\xi \int_{-L}^{L} u^{*}(x,t)\psi(x)dx\right)dt \\ + \lambda(1 - ||\psi||^2) + \lambda\left(1 - \int_{-L}^{L} \psi(x)\psi^{*}(x)dx\right).\end{aligned} \tag{11.52}$$

Для решения задачи с множителем Лагранжа требуется, чтобы производная лагранжиана была равна нулю:

$$\frac{\partial \mathcal{L}}{\partial \psi^{*}} = 0. \tag{11.53}$$

Применяя это условие к (11.52) и меняя порядок интегрирования, получаем

$$\frac{\partial \mathcal{L}}{\partial \psi^*} = \int_{-L}^{L} d\xi \left[\frac{1}{T} \int_0^T \left(u(\xi,t) \int_{-L}^{L} u^*(x,t)\psi(x)dx \right) dt - \lambda\psi(x) \right] = 0. \tag{11.54}$$

Приравнивание подынтегрального выражения к нулю дает следующую задачу о собственных значениях:

$$\langle R(\xi, x), \psi \rangle = \lambda\psi, \tag{11.55}$$

где $R(\xi, x)$ – двухточечный корреляционный тензор непрерывных данных $u(x, t)$, усредненный по временному интервалу, на котором собирались данные:

$$R(\xi, x) = \frac{1}{T} \int_0^T u(\xi,t)u^*(x,t)dt. \tag{11.56}$$

Если пространственная переменная x дискретизирована вектором высокой размерности $\mathbf{u}(t) = [u(x_1, t) \quad u(x_2, t) \quad \cdots \quad u(x_n, t)]^T$, то $R(\xi, x)$ принимает вид:

$$\mathbf{R} = \frac{1}{T} \int_0^T \mathbf{u}(t)\mathbf{u}^*(t)dt. \tag{11.57}$$

На практике функция $\mathbf{R}$ вычисляется при помощи приближенного интегрирования по формуле прямоугольников. Это позволяет связать данный метод с методом на основе снимков, что мы и обсудим ниже.

Квадратурные правила для R: правило трапеций

Вычислить интеграл (11.57) можно численно с помощью квадратурных формул [316]. Простейшей квадратурной формулой является правило трапеций, согласно которому интеграл вычисляется суммированием площадей трапеций. На рис. 11.6 показан вариант формулы трапеций, когда суммируются площади прямоугольников. Таким образом, мы получаем аппроксимацию двухточечного корреляционного тензора:

$$\begin{aligned} \mathbf{R} &= \frac{1}{T} \int_0^T \mathbf{u}(t)\mathbf{u}^*(t)dt \\ &\approx \frac{\Delta t}{T} [\mathbf{u}^*(t_1)\mathbf{u}(t_1) + \mathbf{u}^*(t_2)\mathbf{u}(t_2) + \cdots + \mathbf{u}^*(t_m)\mathbf{u}(t_m)] \\ &= \frac{\Delta t}{T} [\mathbf{u}_1^*\mathbf{u}_1 + \mathbf{u}_2^*\mathbf{u}_2 + \cdots + \mathbf{u}_m^*\mathbf{u}_m], \end{aligned} \tag{11.58}$$

где предполагается, что $u(x, t)$ дискретизирована в виде вектора $\mathbf{u}_j = \mathbf{u}(t_j)$ и рассматривается m прямоугольных столбиков ширины Δt – такой, что (m) $\Delta t = T$. Определив матрицу

$$\mathbf{X} = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \quad \mathbf{u}_m], \tag{11.59}$$

мы сможем переписать двухточечный корреляционный тензор в виде:

$$\mathbf{R} \approx \frac{1}{m}\mathbf{X}^{*}\mathbf{X}, \tag{11.60}$$

а это в точности определение ковариационной матрицы (1.27), т. е. $\mathbf{C} \approx \mathbf{R}$. Заметим, что роль $1/T$ состоит в усреднении по различным траекториям, так что среднее вычитается, что приводит к определению, согласованному с ковариацией.

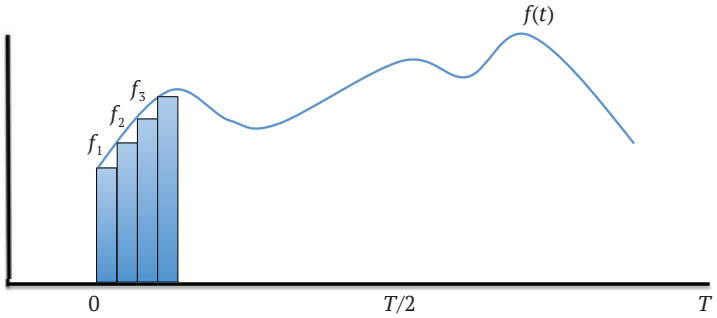

Рис. 11.6 ❖ Применение формулы прямоугольников для вычисления интеграла $\int_0^T f(t)dt$. Интеграл аппроксимируется суммой площадей прямоугольников высотой $f(t_j) = f_j$ и шириной δt

Квадратурные правила более высокого порядка

Методы численного интегрирования просто вычисляют площадь под заданной кривой. Эта идея вытекает из самого определения интеграла

$$\int_a^b f(t)dt = \lim_{\Delta t \to 0} \sum_{j=0}^{m-1} f(t_j)\Delta t, \tag{11.61}$$

где $b - a = (m - 1)\Delta t$. Площадь под кривой определяется как предел суммы площадей прямоугольников, когда их количество стремится к бесконечности. Этот процесс называется численной квадратурой. Любую сумму такого рода можно представить в виде:

$$Q[f] = \sum_{j=0}^{m-1} w_j f(t_j) = w_0 f(t_0) + w_1 f(t_1) + \cdots + w_{m-1} f(t_{m-1}), \tag{11.62}$$

где $a = t_0 < t_1 < t_2 < \cdots < t_{m-1} = b$. Таким образом, интеграл равен

$$\int_a^b f(t)dt = Q[f] + E[f], \tag{11.63}$$

где член $E[f]$ – ошибка аппроксимации интеграла квадратурной суммой (11.62). В типичном случае ошибка $E[f]$ обусловлена отбрасыванием. Для

интегрирования используется полиномиальная аппроксимация значений $f(t_j)$. Таким образом, мы предполагаем, что функцию $f(t)$ можно аппроксимировать полиномом

$$P_n(t) = a_n t_n + a_{n-1} t_{n-1} + \cdots + a_1 t + a_0, \tag{11.64}$$

и в этом случае ошибка аппроксимации пропорциональна $(n + 1)$-й производной $E[f] = Af^{(n+1)}(c)$, где A – постоянная. Процедура полиномиальной аппроксимации данных описывается *формулами Ньютона–Коутса*.

Следующие методы приближенного интегрирования вытекают из различных способов полиномиальной аппроксимации подынтегрального выражения. Предполагается, что

$$t_k = t_0 + \Delta t_k, \qquad f_k = f(t_k). \tag{11.65}$$

Тогда имеем следующие алгоритмы интегрирования:

формула трапеций $\int_{t_0}^{t_1} f(t)dt = \frac{\Delta t}{2}(f_0 + f_1) - \frac{\Delta t^3}{12} f''(c);$ (11.66a)

формула Симпсона $\int_{t_0}^{t_2} f(t)dt = \frac{\Delta t}{3}(f_0 + 4f_1 + f_2) - \frac{\Delta t^5}{90} f''''(c);$ (11.66b)

формула 3/8 Симпсона

$$\int_{t_0}^{t_3} f(t)dt = \frac{3\Delta t}{8}(f_0 + 3f_1 + 3f_2 + f_3) - \frac{3\Delta t^5}{80} f''''(c); \tag{11.66c}$$

формула Буля

$$\int_{t_0}^{t_4} f(t)dt = \frac{2\Delta t}{45}(7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4) - \frac{8\Delta t^7}{945} f^6(c). \tag{11.66d}$$

Эти алгоритмы дают разную точность. Точнее, они определяют схемы с точностью $O(\Delta t^2)$, $O(\Delta t^4)$, $O(\Delta t^4)$ и $O(\Delta t^6)$ соответственно. Точность зависит от степени аппроксимирующего полинома. Отметим, что в *формуле трапеций* интеграл аппроксимируется суммой площадей трапеций. В *формуле Симпсона* через три соседние точки проводится квадратичная кривая, и интеграл аппроксимируется суммой получившихся криволинейных трапеций. В *формуле 3/8 Симпсона* через четыре соседние точки проводится кубическая кривая, а в *формуле Буля* строится полином четвертой степени по пяти точкам.

Методы интегрирования (11.66) дают значения интегралов лишь на малой части области интегрирования. Например, по формуле трапеции мы получаем значения только для $t \in [t_0, t_1]$. Однако наша конечная цель – вычислить интеграл по всему отрезку $t \in [a, b]$. Если отрезок разбит на части точками $a = t_0 < t_1 < t_2 < \cdots < t_{m-1} = b$, то, применив правило трапеций к каждой части, получим полное значение интеграла:

$$\int_a^b f(t)dt \approx Q[f] = \sum_{j=1}^{m} \frac{\Delta t}{2}(f_j + f_{j+1}). \tag{11.67}$$

Раскрывая эту сумму, получаем

$$\sum_{j=1}^{m}\frac{\Delta t}{2}(f_j+f_{j+1})=\frac{\Delta t}{2}(f_0+f_1)+\frac{\Delta t}{2}(f_1+f_2)+\cdots+\frac{\Delta t}{2}(f_m+f_{m-1})$$
$$=\frac{\Delta t}{2}(f_0+2f_1+2f_2+\cdots+2f_m+f_{m-1})$$
$$=\frac{\Delta t}{2}\left(f_0+f_{m-1}+2\sum_{j=1}^{m}f_j\right). \tag{11.68}$$

В окончательном выражении значения в точках между f_0 и f_{m-1} не вычисляются дважды, каждое из них фигурирует только один раз, так что алгоритм работает вдвое быстрее, чем вычисления по первой формуле. Такие приемы оптимизации вычислений следует использовать при любой возможности.

POD-моды и квадратурные формулы

Любую из описанных выше формул можно использовать для аппроксимации двухточечного корреляционного тензора $\mathbf{R}(\xi, x)$. В методе снимков для построения матрицы $\mathbf{X}$ неявно используется формула трапеций. Напомним, что

$$\mathbf{X}=\begin{bmatrix}| & | & & |\\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m\\ | & | & & |\end{bmatrix}, \tag{11.69}$$

где столбцы $\mathbf{u}_k \in \mathbb{C}^n$ могут содержать данные, полученные экспериментально или в результате моделирования. Сингулярное разложение этой матрицы дает моды, используемые для создания низкорангового погружения данных $\mathbf{\Psi}$.

Можно было бы для построения низкорангового разложения воспользоваться квадратурной формулой более высокого порядка. Тогда вместо матрицы (11.69) мы получили бы

$$\mathbf{X}=\begin{bmatrix}| & | & | & | & | & & | & |\\ \mathbf{u}_1 & 4\mathbf{u}_2 & 2\mathbf{u}_3 & 4\mathbf{u}_4 & 2\mathbf{u}_5 & \cdots & 4\mathbf{u}_{m-1} & \mathbf{u}_m\\ | & | & | & | & | & & | & |\end{bmatrix}, \tag{11.70}$$

если взять формулу Симпсона. На практике чаще всего используется именно формула Симпсона, потому что вычисления по ней просты, а повышение точности по сравнению с правилом трапеций весьма заметно. Для построения этой матрицы нужно просто умножить матрицу данных справа на вектор $[1 \quad 4 \quad 2 \quad 4 \quad \cdots \quad 2 \quad 4 \quad 1]^T$. Затем производится SVD для вычисления низкорангового погружения $\mathbf{\Psi}$. Прежде чем аппроксимировать низкоранговое решение, нужно выполнить действие, обратное умножению на веса квадратуры. Насколько нам известно, существует очень мало исследований по оценке преимуществ других квадратурных формул. Впрочем, интересую-

щемуся читателю стоит ознакомиться со стратегией оптимальной выборки снимков, описанной в работе Kunisch and Volkwein [315].

11.5. POD с симметриями: повороты и сдвиги

Метод POD не лишен недостатков. Хорошо известно, что лежащий в его основе алгоритм SVD оптимально обрабатывает инвариантности, присутствующие в данных. Наиболее часто встречаются сдвиговые и поворотные инвариантности. Сдвиговая инвариантность наблюдается при простом распространении волн, она затрудняет вычисление корреляции, поскольку важные признаки не совмещены в последовательных снимках.

Далее мы рассмотрим эффекты сдвига (параллельного переноса) и поворота. В качестве иллюстраций приводятся физические задачи, представляющие практический интерес. Важно отметить, что если структура инвариантности не принимается во внимание, то POD-упрощение дает описание истинной динамики с искусственно завышенной размерностью. Это подвергает сомнению нашу способность использовать POD в качестве диагностического средства или платформы для построения моделей пониженного порядка.

Сдвиг: распространение волн

Для иллюстрации влияния сдвига на POD-анализ рассмотрим поступательное распространение простого гауссова импульса со скоростью c:

$$u(x, t) = \exp[-(x - ct + 15)^2]. \tag{11.71}$$

Это решение рассматривается на пространственном интервале $x \in [-20, 20]$ и временном интервале $t \in [0, 10]$. Показанная ниже программа порождает репрезентативное решение и его низкоранговое представление.

Листинг 11.5 ❖ Поступательно распространяющаяся волна для POD-анализа

```
n=200; L=20; x=linspace(-L,L,n); y=x; % пространство
m=41; T=10; t=linspace(0,T,m);        % время
c=3; % скорость волны

X=[];
for j=1:m
X(:,j)=exp(-(x+15-c*t(j)).^2).'; % снимки данных
end
[U,S,V]=svd(X); % SVD-разложение
```

На рис. 11.7 (а) показана простая рассматриваемая эволюция. Видно, что сдвиг импульса влияет на корреляцию в данной точке пространства. Наивное применение SVD не учитывает сдвиговой природы данных. В результате

порождаемые SVD сингулярные значения убывают медленно, как показано на рис. 11.7 (b) и (c). На самом деле в каждой из нескольких первых мод сосредоточено примерно по 8 % дисперсии.

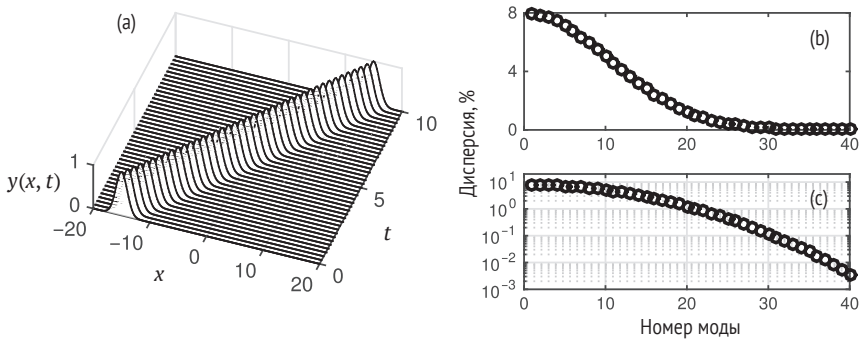

Рис. 11.7 ❖ (a) Поступательное распространение гауссова импульса со скоростью $c = 3$. Сингулярное разложение порождает медленно убывающие сингулярные значения, что показано на рисунке (b) в обычном и на рисунке (c) в логарифмическом масштабе

Медленное убывание сингулярных значений позволяет предположить, что построить низкоранговое погружение будет нелегко. К тому же возникают интересные вопросы об интерпретации POD-мод и их временной динамики. На рис. 11.8 показаны первые четыре пространственные (**U**) и временные моды (**V**), найденные с помощью SVD. Пространственные моды глобальны в том смысле, что охватывают всю область, в которой происходило распространение импульса. Интересно, что они выглядят как моды Фурье в этой области. Временные моды иллюстрируют аналогичный базис мод Фурье для данного конкретного примера поступательного распространения волны с постоянной скоростью.

Неудача POD в этом случае объясняется просто сдвиговой инвариантностью. Если *устранить*, или вынести за скобки [457], инвариантность, до того как предпринята попытка понизить размерность данных, то метод POD все же можно будет использовать для низкоранговой аппроксимации. Чтобы устранить инвариантность, ее нужно сначала идентифицировать и определить вспомогательную переменную. Поэтому перепишем динамику в виде:

$$u(x, t) \to u(x - c(t)), \tag{11.72}$$

где $c(t)$ соответствует сдвиговой инвариантности в системе, виновной в неприменимости метода POD. Параметр c можно найти разными способами. В работе Rowley and Marsden [457] предложен метод факторизации инвариантности на основе шаблонов. Можно также использовать простое вычисление центра масс, чтобы определить положение волны и переменную $c(t)$ [316].

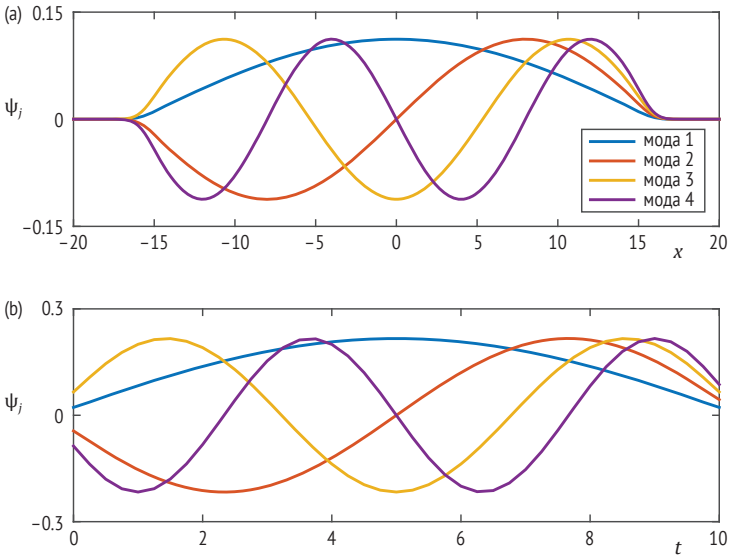

Рис. 11.8 ❖ Первые четыре пространственные (a) (первые четыре столбца матрицы **U**) и временные моды (b) (первые четыре столбца матрицы **V**). Волна, поступательно перемещающаяся с постоянной скоростью, порождает структуру, характерную для мод Фурье, в пространстве и во времени

Поворот: спиральные волны

Вторая часто встречающаяся инвариантность связана с поворотом. Как и сдвиг, поворот перемещает когерентные низкоранговые структуры таким образом, что корреляций между поведением системы в определенных точках пространства больше не наблюдается. Для иллюстрации эффектов поворотной инвариантности рассмотрим локализованную спиральную волну.

Спиральная волна с центром в начале координат определяется следующим образом:

$$u(x,y) = \tanh\left[\sqrt{x^2+y^2}\cos\left(A\angle(x+iy) - \sqrt{x^2+y^2}\right)\right], \tag{11.73}$$

где A – количество ветвей спирали, а $\angle$ обозначает фазовый угол величины $(x+iy)$. Чтобы локализовать спираль в пространственной области, умножим ее на гауссову функцию двух переменных с центром в начале координат, т. е. будем рассматривать функцию

$$f(x,y) = u(x,y)\exp[-0.01(x^2+y^2)]. \tag{11.74}$$

Эта функция вычисляется следующей программой.

Листинг 11.6 ❖ Спиральная волна для POD-анализа

```
n=100;
L=20; x=linspace(-L,L,n); y=x;
```

```
[X,Y]=meshgrid(x,y);

Xd=[];
for j=1:100
    u=tanh(sqrt(X.^2+Y.^2)).*cos(angle(X+i*Y)-(sqrt(X.^2+Y.^2))+j/10);
    f=exp(-0.01*(X.^2+Y.^2));
    uf=u.*f;
    Xd(:,j)=reshape(uf,n^2,1);
    pcolor(x,y,uf), shading interp, colormap(hot)
end
```

Отметим, что в этой программе на j-й итерации цикла порождается снимок с фазовым углом спиральной волны, равным $j/10$. Тем самым создается *вращательная* структура, которую мы хотим исследовать. Вращение можно ускорить или замедлить, соответственно уменьшив или увеличив значение в знаменателе.

Помимо функции $u(x, y)$, мы также рассматриваем тесно связанные с ней функции $|u(x, y)|$ и $u(x, y)^5$, показанные на рис. 11.9. Хотя все три функции, очевидно, имеют одну и ту же вращательную структуру, из-за изменения функциональной формы низкоранговые аппроксимации волн получаются совершенно различными.

Рис. 11.9 ❖ Спиральные волны (a) $u(x, y)$, (b) $|u(x, y)|$ и (c) $u(x, y)^5$ в области $x \in [-20, 20]$, $y \in [-20, 20]$. Спирали закручиваются по часовой стрелке с угловой скоростью ω

Мы начнем анализ с рассмотрения функции $u(x, y)$ на рис. 11.9 (a). SVD этой матрицы и ее низкоранговая структура вычисляются следующей программой.

Листинг 11.7 ❖ SVD-разложение спиральной волны

```
[U,S,V]=svd(Xd,0);

figure(2)
subplot(4,1,3)
plot(100*diag(S)/sum(diag(S)),'ko','Linewidth',[2])
subplot(4,1,4)
semilogy(100*diag(S)/sum(diag(S)),'ko','Linewidth',[2])
subplot(2,1,1)
plot(V(:,1:4),'Linewidth',[2])
figure(3)
for j=1:4
    subplot(4,4,j)
    mode=reshape(U(:,j),n,n);
    pcolor(X,Y,mode), shading interp,caxis([-0.03 0.03]), colormap(gray)
end
```

Программа создает два рисунка. На первом оценивается ранг наблюдаемой динамики и поведение первых четырех мод **V** во времени. На рис. 11.10 (b) и (c) показаны графики убывания собственных значений в обычном и логарифмическом масштабах. Примечательно, что в пределах точности вычислений в первых двух модах сосредоточена *вся* дисперсия данных. Это подтверждается временной динамикой первых четырех мод. Именно первые две моды на рис. 11.10 (a) имеют отчетливый колебательный характер, что связано с вращением мод 1 и 2 на рис. 11.11. Моды же 3 и 4 похожи на шум как во времени, так и в пространстве – это обусловлено ошибками округления.

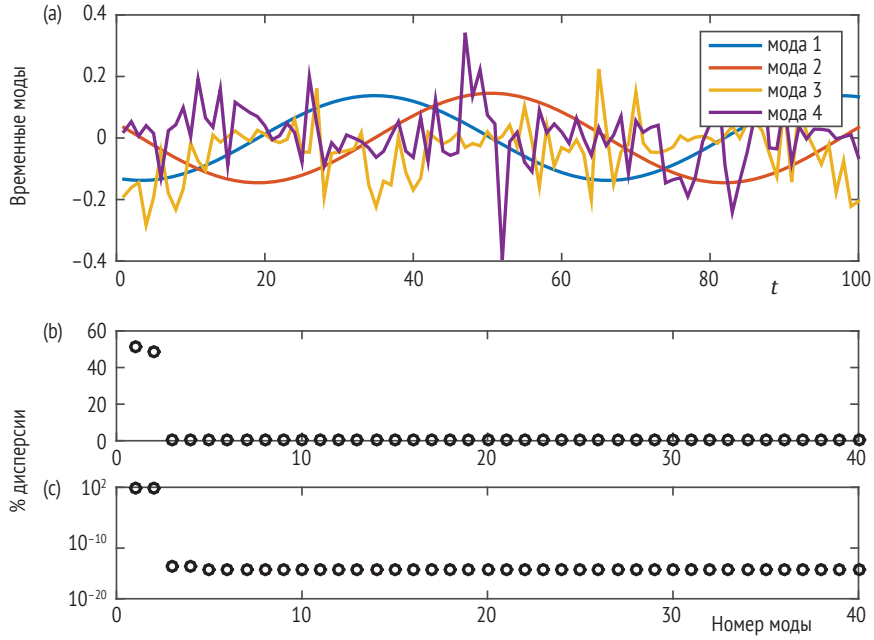

Рис. 11.10 ❖ (a) Первые четыре временные моды матрицы **V**. В пределах точности вычислений вся дисперсия сосредоточена в первых двух модах, что видно из графиков убывания сингулярных значений в обычном (b) и логарифмическом (c) масштабах. Примечательно, что POD выделяет ровно две моды (см. рис. 11.11) для представления вращающейся спиральной волны

Спиральная волна (11.74) допускает двухмодовое представление в пределах точности вычислений. Отчасти это объясняется синусоидальной природой решения при обходе его по окружности фиксированного радиуса. Достаточно перейти от $u(x, t)$ к $|u(x, t)|$ или $u(x, t)^5$, как выяснится, что и сами низкоранговые моды, и их временная динамика сильно отличаются. На рис. 11.12 (a) и (b) показано убывание сингулярных значений для этих двух функций и продемонстрированы существенные отличия от ранее рассмотренной эволюции мод. Также показана доминирующая временная динамика, вычисленная по матрице **V**. В случае функции $|u(x, t)|$ сингулярные значения убывают медленно и не приближаются к пределам точности вычислений.

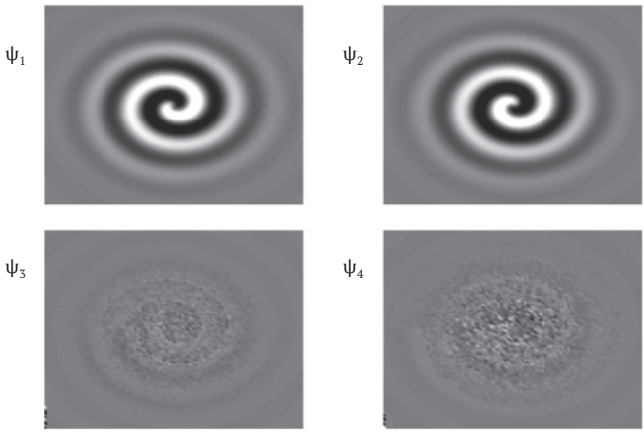

Рис. 11.11 ❖ Первые четыре POD-моды вращающейся спиральной волны $u(x, y)$. В пределах точности вычислений в первых двух модах сосредоточена почти вся дисперсия, тогда как третья и четвертая моды зашумлены из-за ошибок округления. Рассматривается область $x \in [-20, 20]$, $y \in [-20, 20]$

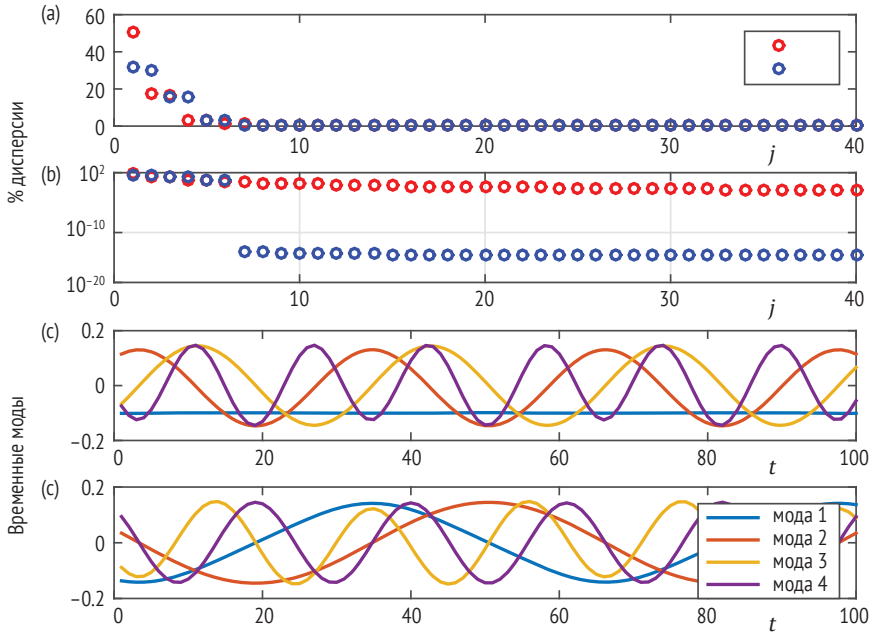

Рис. 11.12 ❖ Графики убывания сингулярных значений в обычном (a) и логарифмическом (b) масштабах показывают, что для функции $|u(x, t)|$ убывание медленное, а для $u(x, t)^5$ для аппроксимации в пределах точности вычислений необходим ранг $r = 6$. На рисунках (c) и (d) показаны первые четыре временные моды матрицы **V** для этих функций

Что касается пятой степени, то точная аппроксимация возможна при ранге $r = 6$. Это доказывает, что поворотная инвариантность усложняет процедуру понижения размерности методом POD. Ведь единственное различие между тремя решениями – конкретный вид функции, а вращаются все они с одной и той же скоростью.

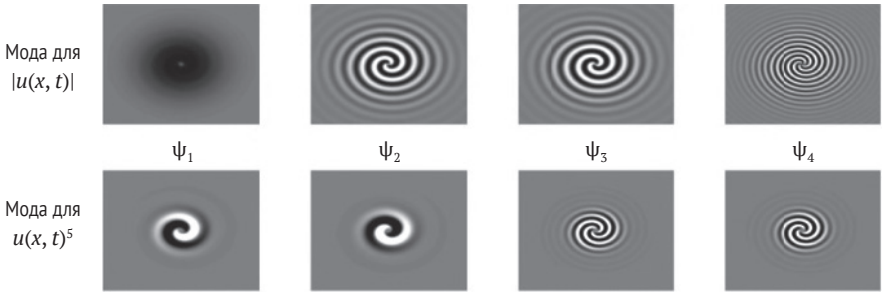

Рис. 11.13 ❖ Первые четыре POD-моды для вращающейся спиральной волны $|u(x, y)|$ (верхний ряд) и $u(x, t)^5$ (нижний ряд). В отличие от предыдущего примера, первые четыре моды не содержат всю дисперсию, так что для точной аппроксимации нужны дополнительные моды. Рассматривается область $x \in [-20, 20]$, $y \in [-20, 20]$

В заключение отметим, что инвариантности могут существенно ограничить применение метода POD. Главное, что они искусственно увеличивают размерность системы и мешают интерпретировать результаты. Экспертные знания о данной системе и ее потенциальных инвариантностях помогают выработать математическую стратегию их устранения, т. е. преобразования данных [316, 457]. Но и у такой стратегии есть ограничения, особенно если инвариантных структур две или больше. Например, если в данных присутствует две волны, движущиеся с разными скоростями, то предложенные методы устранения инвариантностей не смогут справиться с обеими волнами одновременно. В итоге обработка инвариантностей до сих пор остается открытой проблемой.

Рекомендуемая литература

Учебники

(1) **Certified reduced basis methods for parametrized partial differential equations**, by J. Hesthaven, G. Rozza and B. Stamm, 2015 [244].

(2) **Reduced basis methods for partial differential equations: An introduction**, by A. Quarteroni, A. Manzoni and N. Federico, 2015 [442].

(3) **Model reduction and approximation: Theory and algorithms**, by P. Benner, A. Cohen, M. Ohlberger and K. Willcox, 2017 [54].

(4) **Turbulence, coherent structures, dynamical systems and symmetry**, by P. Holmes, J. L. Lumley, G. Berkooz and C. W. Rowley, 2012 [251].

Статьи и обзоры

(1) **A survey of model reduction methods for parametric systems**, by P. Benner, S. Gugercin and K. Willcox, *SIAM Review*, 2015 [53].

(2) **Model reduction using proper orthogonal decomposition**, by S. Volkwein, *Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz*, 2011 [542].

(3) **The proper orthogonal decomposition in the analysis of turbulent flows**, by G. Berkooz, P. Holmes and J. L. Lumley, *Annual Review of FluidMechanics*, 1993 [57].

Глава 12

Интерполяция для ROM

В предыдущей главе была описана математическая подоплека ROM. Точнее, было продемонстрировано применение метода POD для проецирования динамики УрЧП на низкоранговое подпространство, в котором проще обсчитать модель системы, описываемую PDE. Однако проецирование на низкоранговое аппроксимирующее подпространство остается сложной проблемой из-за нелинейности. Интерполяция в сочетании с POD позволяет преодолеть эту трудность, предлагая вычислительно эффективный метод дискретной (разреженной) выборки и обсчета нелинейности. В этой главе мы разовьем идеи алгоритмов сжатой и разреженной выборки из главы 3, позволяющих с помощью выборки небольшого объема реконструировать низкоранговую динамику УрЧП. Благодаря этим методам вычислительная сложность ROM хорошо масштабируется на ранг аппроксимации даже для сложных нелинейностей. В основном данная глава посвящена методам разреженной интерполяции, позволяющим быстро строить модели пониженного порядка. На практике эти методы вызывают особый интерес сообщества ROM, поскольку именно они позволяют обсчитывать параметрические УрЧП, в которых необходимо часто обновлять ROM-модель.

12.1. Неполное POD

Успех понижения порядка нелинейной модели в значительной мере определяется двумя ключевыми инновациями – это (i) хорошо известный метод POD–Галеркина [251, 57, 542, 543], который теоретически обосновывает проецирование нелинейной динамики высокой размерности на подпространство низкой размерности, и (ii) разреженная выборка из пространства состояний для интерполяции нелинейных членов, необходимой для проекции на подпространство. Таким образом, разреженность уже вошла в обиход как критически важный математический подход к упрощению модели, чему способствовали метод неполного POD и его варианты [179, 555, 565, 120, 159]. На самом деле важность эффективности управления обсчетом нели-

нейностей уже давно признана специалистами по ROM, и для решения этой задачи предложены различные методы. Пожалуй, первым результатом по разреженной выборке с помощью POD-мод был разработанный Иверсоном (Everson) и Сировичем (Sirovich) метод, получивший название «неполное POD» [179]. В этой схеме разреженной выборки случайные измерения использовались для аппроксимации скалярных произведений. Теоретически обоснованный выбор точек интерполяции – с помощью инфраструктуры неполного POD [179, 555, 565, 120, 159] или оценки отсутствующих точек (missing point estimation – MPE) [400, 21] – очень скоро был включен в ROM для повышения их качества. Предложенный позже эмпирический метод интерполяции (empirical interpolation method – EIM) [41] и его самый успешный вариант – адаптированный для POD дискретный эмпирический метод интерполяции (discrete empirical interpolation method – DEIM) [127] – представляют собой жадный алгоритм, позволяющий почти оптимально реконструировать нелинейные члены исходной системы высокой размерности. В DEIM проекция сочетается с интерполяцией, а именно отобранные индексы интерполяции используются для задания основанной на интерполяции проекции на почти оптимальное по норме ℓ_2 подпространство, аппроксимирующее нелинейность.

Низкоранговая аппроксимация, найденная POD, позволяет реконструировать решение $\mathbf{u}(x, t)$ в формуле (12.9) по r измерениям в n-мерном пространстве. Из этого факта вытекают важные следствия относительно того, как подходить к измерениям динамической системы [179]. В частности, раз для реконструкции необходимо всего $r \ll n$ измерений, то мы можем определить разреженное представление $\tilde{\mathbf{u}} \in \mathbb{C}^r$

$$\tilde{\mathbf{u}} = \mathbf{P}\mathbf{u}, \tag{12.1}$$

где матрица измерений $\mathbf{P} \in \mathbb{R}^{r \times n}$ определяет места r измерений полного состояния $\mathbf{u} \in \mathbb{C}^n$. Например, матрица измерений может иметь вид

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & \cdots & & & & & & \cdots & 0 \\ 0 & \cdots & 0 & 1 & 0 & \cdots & & & \cdots & 0 \\ 0 & \cdots & & & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \vdots & 0 & & & \cdots & 0 & 0 & 1 & \cdots & \vdots \\ 0 & \cdots & & & \cdots & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{12.2}$$

где элементы, соответствующие точкам измерения, равны 1, а остальные элементы матрицы равны нулю. Матрица $\mathbf{P}$ определяет проекцию $\tilde{\mathbf{u}}$ на r-мерное пространство, которую можно использовать для аппроксимации решений УрЧП.

Наблюдение (12.1) лежит в основе метода *неполного POD* (*gappy POD*), введенного в работе Everson and Sirovich [179]. Идея в том, чтобы использовать небольшое число измерений, или неполные данные, для реконструкции полного состояния системы. Это позволяет справиться со сложностью вычисления нелинейных членов высшего порядка при POD-упрощении.

Разреженные измерения и реконструкция

Матрица измерений $\mathbf{P}$ дает возможность аппроксимировать вектор состояния $\mathbf{u}$ по r измерениям. Аппроксимация строится с использованием формулы (12.1) со стандартной POD-проекцией:

$$\tilde{\mathbf{u}} \approx \mathbf{P}\sum_{k=1}^{r}\tilde{a}_k\boldsymbol{\psi}_k, \tag{12.3}$$

где коэффициенты $\tilde{a}_k$ подобраны так, чтобы минимизировать ошибку аппроксимации $||\tilde{\mathbf{u}} - \mathbf{Pu}||$. Теперь проблема заключается в том, как найти значения $\tilde{a}_k$, коль скоро вычислить скалярные произведения (12.3) нельзя. Точнее, размерность вектора $\tilde{\mathbf{u}}$ равна r, размерность POD-мод равна n, а скалярно перемножить векторы разной размерности невозможно. Таким образом, в общем случае моды $\boldsymbol{\psi}_k(x)$ не ортогональны на r-мерном носителе $\tilde{\mathbf{u}}$. Будем обозначать носитель символом $s[\tilde{\mathbf{u}}]$. Если быть точным, то ортогональность следует рассматривать в полном пространстве, а не на носителе. Имеют место такие два утверждения:

$$M_{kj} = \langle\boldsymbol{\psi}_k, \boldsymbol{\psi}_j\rangle = \delta_{kj}, \tag{12.4a}$$

$$M_{kj} = \langle\boldsymbol{\psi}_k, \boldsymbol{\psi}_j\rangle_{s[\tilde{\mathbf{u}}]} \neq 0 \text{ для всех } k, j, \tag{12.4b}$$

где M_{kj} – элементы эрмитовой матрицы $\mathbf{M}$, а δ_{kj} – символ Кронекера. Тот факт, что POD-моды не ортогональны на носителе $s[\tilde{\mathbf{u}}]$, подталкивает нас к поиску альтернативных способов вычисления вектора $\tilde{\mathbf{a}}$.

Чтобы найти коэффициенты $\tilde{a}_k$, мы можем воспользоваться методом наименьших квадратов для минимизации ошибки $\tilde{\mathbf{A}}$

$$E = \int_{s[\tilde{\mathbf{u}}]}\left[\tilde{\mathbf{u}} - \sum_{k=1}^{r}\tilde{a}_k\boldsymbol{\psi}_k\right]^2 d\mathbf{x}, \tag{12.5}$$

где скалярное произведение вычисляется на носителе $s[\tilde{\mathbf{u}}]$, так что оба члена под интегралом имеют одинаковую размерность r. Для минимизации (12.5) нужно, чтобы невязка была ортогональна каждой моде $\boldsymbol{\psi}_k$, т. е.

$$\left\langle\tilde{\mathbf{u}} - \sum_{k=1}^{r}\tilde{a}_k\boldsymbol{\psi}_k, \boldsymbol{\psi}_j\right\rangle_{s[\tilde{\mathbf{u}}]} = 0, \quad j \neq k, \ j = 1, 2, \cdots, r. \tag{12.6}$$

На практике мы можем спроецировать вектор полного состояния $\mathbf{u}$ на пространство-носитель и определить вектор $\tilde{\mathbf{a}}$:

$$\mathbf{M}\tilde{\mathbf{a}} = \mathbf{f}, \tag{12.7}$$

где элементы $\mathbf{M}$ вычисляются по формуле (12.4b), а элементы вектора $\mathbf{f}$ равны

$$f_k = \langle\mathbf{u}, \boldsymbol{\psi}_k\rangle_{s[\tilde{\mathbf{u}}]}. \tag{12.8}$$

Заметим, что если пространство измерений достаточно плотное или носитель совпадает со всем пространством, то $\mathbf{M} = \mathbf{I}$, откуда следует, что соб-

ственные значения $\mathbf{M}$ близки к единице для плотных измерений. После того как вектор $\tilde{\mathbf{a}}$ определен, решение реконструируется следующим образом:

$$\mathbf{u}(x, t) \approx \mathbf{\Psi}\tilde{\mathbf{a}}. \tag{12.9}$$

Когда измерения становятся плотными, то мало того что матрица $\mathbf{M}$ стремится к единичной, так еще и $\tilde{\mathbf{a}} \to \mathbf{a}$. Эти наблюдения наводят на мысль оценить эффективность метода и (или) аппроксимации, рассмотрев число обусловленности матрицы $\mathbf{M}$ [524]:

$$\kappa(\mathbf{M}) = ||\mathbf{M}||\ ||\mathbf{M}^{-1}|| = \frac{\sigma_1}{\sigma_m}. \tag{12.10}$$

Здесь мы воспользовались нормой ℓ_2. Если $\kappa(\mathbf{M})$ мало, то говорят, что матрица хорошо обусловлена. Минимум $\kappa(\mathbf{M})$ достигается для единичной матрицы $\mathbf{M} = \mathbf{I}$. Таким образом, когда выборочное пространство становится плотным, число обусловленности также стремится к единице. Это можно использовать как метрику, определяющую качество разреженной выборки. Если число обусловленности велико, то реконструкция будет плохой, а если оно близко к единице, то можно рассчитывать на успех.

Моды гармонического осциллятора

Для демонстрации метода неполной выборки и эффективности реконструкции применим описанную процедуру к многочленам Эрмита, определяемым формулами (11.25) и (11.26). Программа в листинге 12.1 вычисляет первые десять мод по формуле (11.26). Для вычисления второй производной мы пользуемся тем фактом, что преобразование Фурье $\mathcal{F}$ может порождать спектрально точную аппроксимацию, т. е. $u_{xx} = \mathcal{F}^{-1}[(ik)^2\mathcal{F}u]$. Чтобы вычислить производные точнее, мы рассматриваем область определения $x \in [-10, 10]$, но затем продолжаем работать с меньшей областью $x \in [-4, 4]$, представляющей интерес. Напомним также, что при вычислении преобразования Фурье предполагается периодичность с периодом 2π. Этому условию можно удовлетворить, введя масштабный коэффициент в волновые векторы. Первые пять мод были показаны на рис. 11.3. Следующая программа строит первые 10 мод и выводит их на цветном рисунке, чтобы наглядно показать их различные особенности.

Листинг 12.1 ❖ Моды гармонического осциллятора

```
L=10; x3=-L:0.1:L; n=length(x3)-1; % область определения
x2=x3(1:n); k=(2*pi/(2*L))*[0:n/2-1 -n/2:-1]; % k-вектор
ye=exp(-(x2.^2)); ye2=exp((x2.^2)/2); % определить гауссианы
for j=0:9 % цикл по 10 модам
    yd=real(ifft(((i*k).^j).*fft(ye))); % вторая производная
    mode=((-1)^(j))*(((2^j)*factorial(j)*sqrt(pi))^(-0.5))*ye2.*yd;
    y(:,j+1)=(mode).'; % сохранить моды в столбцах
end
```

```
x=x2(n/2+1-40:n/2+1+40); % оставить только интервал -4 < x < 4
yharm=y(n/2+1-40:n/2+1+40,:);
pcolor(flipud((yharm(:,10:-1:1).')))
```

Построенные моды показаны на верхнем рис. 12.1. Каждая цветная клетка представляет дискретное значение моды в интервале $x \in [-4, 4]$ с шагом $\Delta x = 0.1$. Всего, стало быть, имеется 81 дискретное значение для каждой моды $\boldsymbol{\psi}_k$. Наша цель – реконструировать функцию вне базисных мод гармонического осциллятора. В частности, рассмотрим функцию

$$f(x) = \exp[-(x - 0.5)^2) + 3\exp(-2(x + 3/2)^2], \tag{12.11}$$

которую мы дискретизируем в той же области, что и базисные моды гармонического осциллятора. Программа в листинге 12.2 вычисляет эту функцию и далее строит ее проекцию на базисные функции $\boldsymbol{\psi}_n$. Исходная функция показана на верхнем рис. 12.2. Заметим, что сейчас наша цель – реконструировать функцию как по низкоранговой проекции на моды гармонического осциллятора, так и по неполным выборочным данным, попавшим в матрицы измерений $\mathbf{P}_j$. Программа строит тестовую функцию и производит реконструкцию по 10-модовому базису гармонического осциллятора. Затем она строит матрицу $\mathbf{M}$ измерений полного состояния и вычисляет ее число обусловленности.

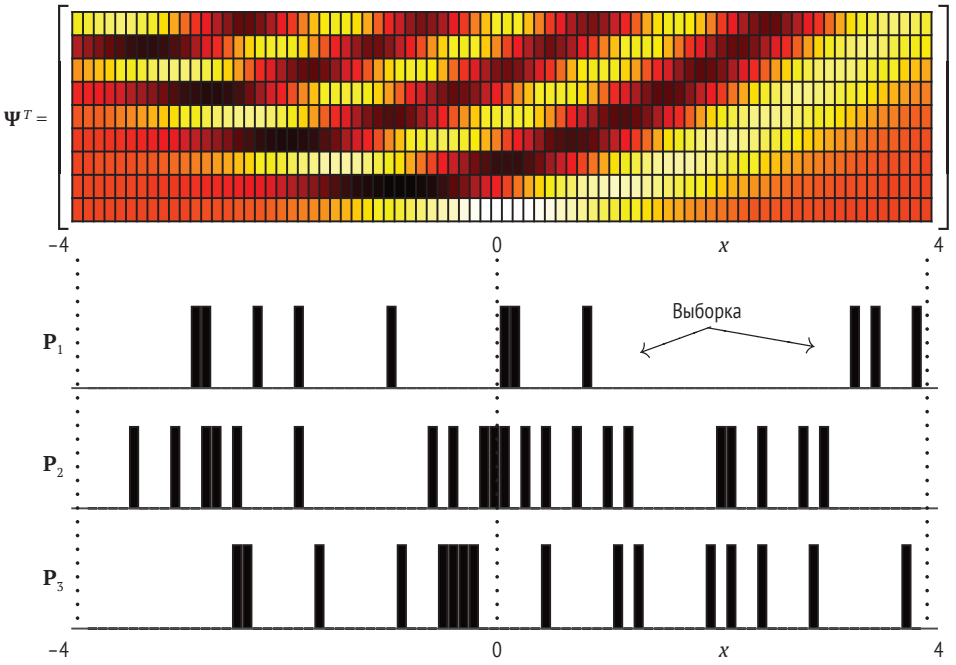

Рис. 12.1 ❖ На верхнем рисунке показаны первые 10 мод квантового гармонического осциллятора, рассмотренного в (11.25) и (11.26). Изображены три случайно сгенерированные матрицы измерений $\mathbf{P}_j$, $j = 1, 2, 3$. Интервал $x \in [-4, 4]$ дискретизирован с шагом $\Delta x = 0.1$. Измерение в любой из образовавшихся точек x_j производится с вероятностью 20 %

Листинг 12.2 ❖ Тестовая функция и ее реконструкция

```
f=(exp(-(x-0.5).^2)+3*exp(-2*(x+1.5).^2))';
for j=1:10 % полная реконструкция
    a(j,1)=trapz(x,f.*yharm(:,j));
end
f2=yharm*a;
subplot(2,1,1), plot(x,f2,'r')
Err(1)=norm(f2-f); % ошибка реконструкции

for j=1:10 % реконструкция матрицы M
    for jj=1:j
        Area=trapz(x,yharm(:,j).*yharm(:,jj));
        M(j,jj)=Area;
        M(jj,j)=Area;
    end
end
cond(M) % вычислить число обусловленности
```

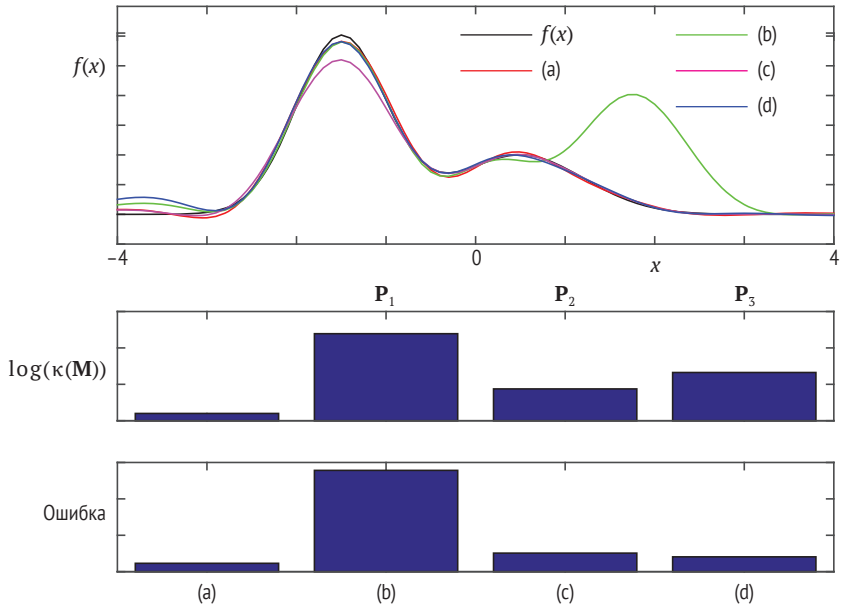

Рис. 12.2 ❖ На верхней панели показана исходная тестовая функция (черным цветом) $f(x) = \exp(-(x-0.5)^2) + 3\exp(-2(x+3/2)^2)$ и ее реконструкция по 10 модам, выбранным в полном пространстве (красным цветом) и на трех репрезентативных носителях $s[\tilde{\mathbf{u}}]$ на рис. 12.1, а именно (b) $\mathbf{P}_1$, (c) $\mathbf{P}_2$ и (d) $\mathbf{P}_3$. Заметим, что ошибка измерения зависит от рассматриваемой функции, тогда как число обусловленности от функции не зависит. Хотя обе метрики могут служить показателями качества, число обусловленности годится для любой функции, что можно считать преимуществом

Результаты этой низкоранговой реконструкции показаны на рис. 12.2. Реконструкция выполняется по полным измерениям, спроецированным на

10 главных мод гармонического осциллятора. В данном случае матрица скалярных произведений мод определяется по формуле (12.4a) и приближенно равна единичной матрице. То, что $\mathbf{M}$ совпадает с $\mathbf{I}$ приближенно, а не точно, объясняется тем, мы работаем в ограниченной области $x \in [-4, 4]$ с шагом дискретизации $\Delta x = 0.1$. Показана также реконструкция для трех разных вариантов разреженных измерений $\mathbf{P}_j$, представленных на рис. 12.1, вместе со среднеквадратической ошибкой и логарифмом числа обусловленности $\log[\kappa(\mathbf{M}_j)]$. На рис. 12.3 мы также наглядно представили все три матрицы $\mathbf{M}_j$. Число обусловленности каждой из них позволяет определить точность соответствующей реконструкции.

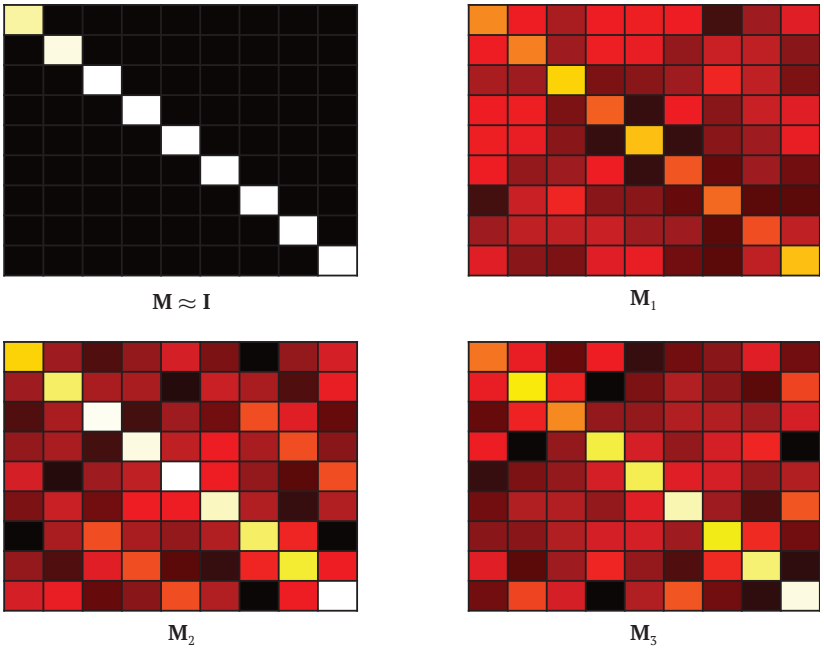

Рис. 12.3 ❖ Ухудшение ортогональности базисных мод на носителе $s[\ddot{\mathbf{u}}]$ иллюстрируется матрицей $\mathbf{M}$, определенной формулой (12.4). Слева вверху показана матрица, порожденная по всем измерениям: она почти совпадает с единичной, а различия объясняются ограничением на область $x \in [-4, 4]$. Матрицы $\mathbf{M}_j$, которые уже не выглядят диагональными, соответствуют матрицам разреженной выборки $\mathbf{P}_j$ на рис. 12.1. Таким образом, очевидно, что в пространстве-носителе измерений моды не ортогональны

Листинг 12.3 ❖ Неполная выборка измерений гармонического осциллятора

```
c=['g','m','b']; % три разные маски измерений
for jloop=1:3
    figure(1), subplot(6,1,3+jloop)
    s=(rand(n,1)>0.8); % выбрать 20 % измерений случайным образом
    bar(x,double(s)), axis([-4.2 4.2 0 1]), axis off

    figure(2) % построить M_j
```

```
    for j=1:10
        for jj=1:j
            Area=trapz(x,s.*(yharm(:,j).*yharm(:,jj)));
            M2(j,jj)=Area; M2(jj,j)=Area;
        end
    end
    subplot(2,2,jloop+1), pcolor(10:-1:1,1:10,(M2'));
    colormap(hot), caxis([-0.1 .3]), axis off
    con(jloop)=cond(M2)

    for j=1:10 % реконструкция по неполным данным
        ftild(j,1)=trapz(x,s.*(f.*yharm(:,j)));
    end

    atild=M2\ftild; % вычислить ошибку
    f2=yharm*atild;
    figure(4),subplot(2,1,1),plot(x,f2,c(jloop))
    Err(jloop+1)=norm(f2-f);
end
```

12.2. Ошибка и сходимость неполного POD

В предыдущем разделе было показано, что способность стратегии неполной выборки точно реконструировать заданную функцию сильно зависит от местоположения точек измерения (датчиков). Учитывая важность этого вопроса, в следующих разделах мы подробно обсудим несколько теоретических методов размещения ограниченного числа датчиков. Сейчас наша цель – изучить свойства сходимости и ошибку метода неполного POD как функцию от процентной доли выборки из полных данных. Точки измерения будут выбираться случайно.

Поскольку стратегия выборки случайная, приведенные ниже результаты по природе своей статистические, так что мы будем вычислять средние и дисперсии выборки. В роли базисных мод снова будут выступать многочлены Эрмита (11.25) и (11.26), которые сгенерированы программой в листинге 12.1 и показаны на рис. 12.1 сверху.

Случайная выборка и сходимость

Мы начнем изучение со случайной выборки объемом 10 %, 20 %, 30 %, 40 %, 50 % и 100 %. Последний случай соответствует идеальной полной выборке измерений. Как и следовало ожидать, чем больше объем выборки, тем меньше ошибка реконструкции. Чтобы продемонстрировать сходимость неполной выборки, мы рассмотрим две метрики: (i) ошибка реконструкции по норме ℓ_2 и (ii) число обусловленности матрицы $\mathbf{M}$ для данной матрицы измерений $\mathbf{P}_j$. Напомним, что число обусловленности – это способ измерить ошибку, не зная истинной функции, т. е. (12.11).

На рис. 12.4 показаны усредненные по 1000 испытаний логарифм ошибки средних квадратов $\log(E + 1)$ (единица прибавлена, чтобы избежать взятия логарифма от нуля) и логарифм числа обусловленности $\log(\kappa(\mathbf{M}))$ в виде функции от процентной доли случайных измерений. Также красными отрезками показана дисперсия σ (концы отрезков соответствуют значениям $\mu \pm \sigma$, где μ – среднее). Ошибка и число обусловленности тем лучше, чем больше объем выборки. Заметим, что ошибка не стремится к нулю, потому что используется разложение всего по 10 модам, что ограничивает точность реконструкции даже при наличии полных измерений.

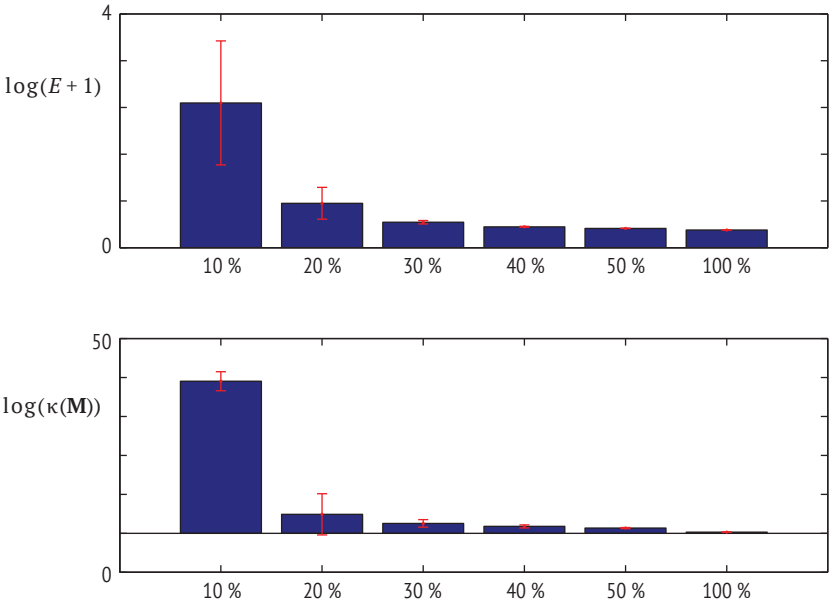

Рис. 12.4 ❖ Логарифм среднеквадратической ошибки $\log(E+1)$ (единица прибавлена, чтобы избежать взятия логарифма от нуля) и логарифм числа обусловленности $\log(\kappa(\mathbf{M}))$ в виде функции от процентной доли измерений. Для уровня 10 % ошибка и число обусловленности наибольшие, как и следовало ожидать. Однако дисперсия результатов, изображенная красными отрезками, весьма велика, а это значит, что при малом числе датчиков результат очень сильно зависит от того, где они размещены

Следующая далее программа, с помощью которой был построен рис. 12.4, производит усреднение по более чем 1000 случайных конфигураций датчиков при объеме выборки 10 %, 20 %, 30 %, 40 % и 50 %. Полная реконструкция (когда объем выборки равен 100 %) выполнена программой в листинге 12.2 и была использована при создании рис. 12.4. Как и следовало ожидать, тренды ошибки и числа обусловленности похожи, и это подтверждает гипотезу о том, что число обусловленности может служить мерой эффективности разреженных измерений без знания истинной функции (в данном случае (12.11)).

Листинг 12.4 ❖ Сходимость ошибки и числа обусловленности

```
for thresh=1:5;
    for jloop=1:1000 % 1000 случайных испытаний
        n2=randsample(n,8*thresh); % случайная выборка
        P=zeros(n,1); P(n2)=1;
        for j=1:10
            for jj=1:j % вычислить матрицу M
                Area=trapz(x,P.*(yharm(:,j).*yharm(:,jj)));
                M2(j,jj)=Area; M2(jj,j)=Area;
            end
        end
        for j=1:10 % реконструкция методом неполного POD
            ftild(j,1)=trapz(x,P.*(f.*yharm(:,j)));
        end
        atild=M2\ftild; % вычислить ошибку
        f2=yharm*atild; % вычислить реконструкцию
        Err(jloop)=norm(f2-f); % L2-ошибка
        con(jloop)=cond(M2);   % число обусловленности
    end
    % среднее и дисперсия
    E(thresh)=mean(log(Err+1)); V(thresh)=(var(log(Err+1)));
    Ec(thresh)=mean(log(con)); Vc(thresh)=(var(log(con)));
end
E=[E Efull]; V=[V 0];
Ec=[Ec log(Cfull)]; Vc=[Vc 0];
```

Неполные измерения и качество реконструкции

Мы можем продолжить статистический анализ метода реконструкции по неполным измерениям, более внимательно изучив 200 случайных испытаний с 20 % измерений. На рис. 12.5 показаны три основные характеристики случайных испытаний. На верхнем рисунке мы видим большую дисперсию в распределении числа обусловленности к($\mathbf{M}$) для 20%-ной выборки – при одном и том же числе датчиков оно может разниться на несколько порядков в зависимости от точек их размещения. На двух нижних рисунках показаны гистограммы распределения логарифма ошибки $\log(E+1)$ и логарифма числа обусловленности. Видно, что ошибка убывает экспоненциально, тогда как распределение числа обусловленности ближе к нормальному. В обоих случаях имеются отчетливые выбросы, показывающие, каких конфигураций датчиков следует избегать.

Чтобы визуализировать случайную выборку неполных измерений объемом 200, подвергнутую статистическому анализу на рис. 12.5, мы создали матрицу, строки которой содержат маски измерения $\mathbf{P}_j$ (см. рис. 12.6). Белые области представляют точки, в которых измерения не производились, черные – точки размещения датчиков. Именно эти измерения порождают огромную дисперсию ошибки и числа обусловленности.

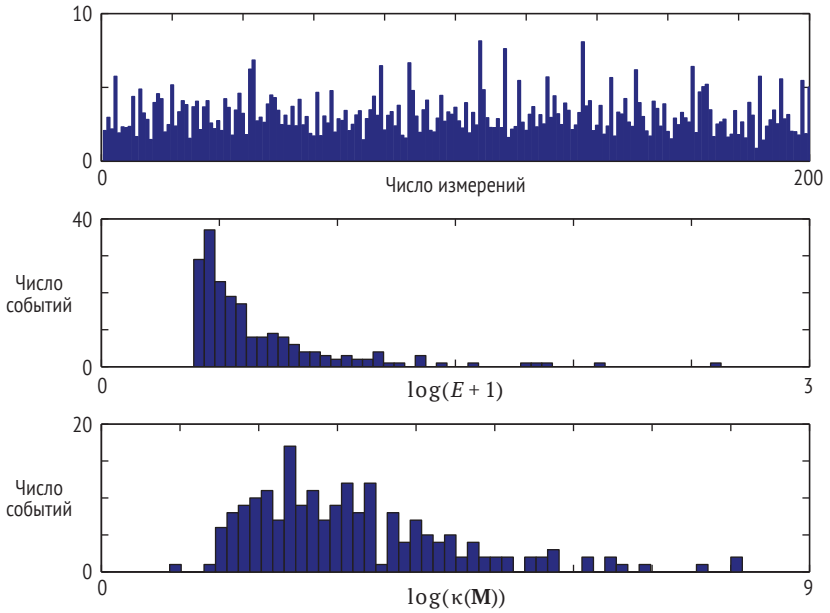

Рис. 12.5 ❖ Статистика по 20 % случайных измерений, рассмотренных на рис. 12.4. На рисунке (a) показано 200 случайных испытаний и число обусловленности log(к(**M**)) в каждом испытании. Для тех же испытаний на рисунке (b) показан логарифм среднеквадратической ошибки, log(E + 1), а на рисунке (c) – число обусловленности, log(к(**M**)). Видно, что разброс в случайных разреженных испытаниях очень велик. По 20 % измерений можно получать как великолепные, так и никуда не годные результаты в зависимости от того, в каких точках производятся измерения. Векторы измерений **P**, по которым собрана эта статистика, показаны на рис. 12.6

И в качестве завершающего шага анализа мы можем выбрать из 200 случайных измерений на рис. 12.6 десять лучших и десять худших векторов $\mathbf{P}_j$. На рис. 12.7 показаны результаты этого процесса. На двух верхних рисунках изображены лучшие и худшие конфигурации датчиков. Интересно, что в худших конфигурациях имеются длинные участки без датчиков вблизи центра области, где сосредоточена большая часть дисперсии мод. Напротив, в лучших конфигурациях датчики распределены более-менее равномерно, а длинных пустых участков мало. На нижнем рисунке показано, что для лучших измерений (слева) число обусловленности на два порядка меньше, чем для худших (справа).

12.3. Неполные измерения: минимизация числа обусловленности

В предыдущем разделе было показано, что от размещения датчиков сильно зависит точность реконструкции решения методом POD. Поэтому важно

иметь теоретически обоснованный подход к определению мест измерений. Далее мы опишем метод, первоначально предложенный в работе Willcox [555]. Он основан на минимизации числа обусловленности к(**M**). Мы уже убедились, что число обусловленности – хорошая метрика для оценки эффективности реконструкции, не зависящая от выбора конкретной функции. Предложенный алгоритм вычислительно затратный, но его можно выполнить на стадии обучения в автономном режиме. После того как положения датчиков определены, их можно использовать для онлайновой реконструкции. Алгоритм состоит из следующих шагов.

1. Для каждой возможной точки вычислить число обусловленности к(**M**) при размещении в ней датчика. Рассматриваются только точки, в которых датчиков еще нет.
2. Определить, какой точке соответствует минимальное число обусловленности к(**M**). В этой точке размещается k-й датчик.
3. Повторить шаги 1–2 для $(k + 1)$-го датчика.

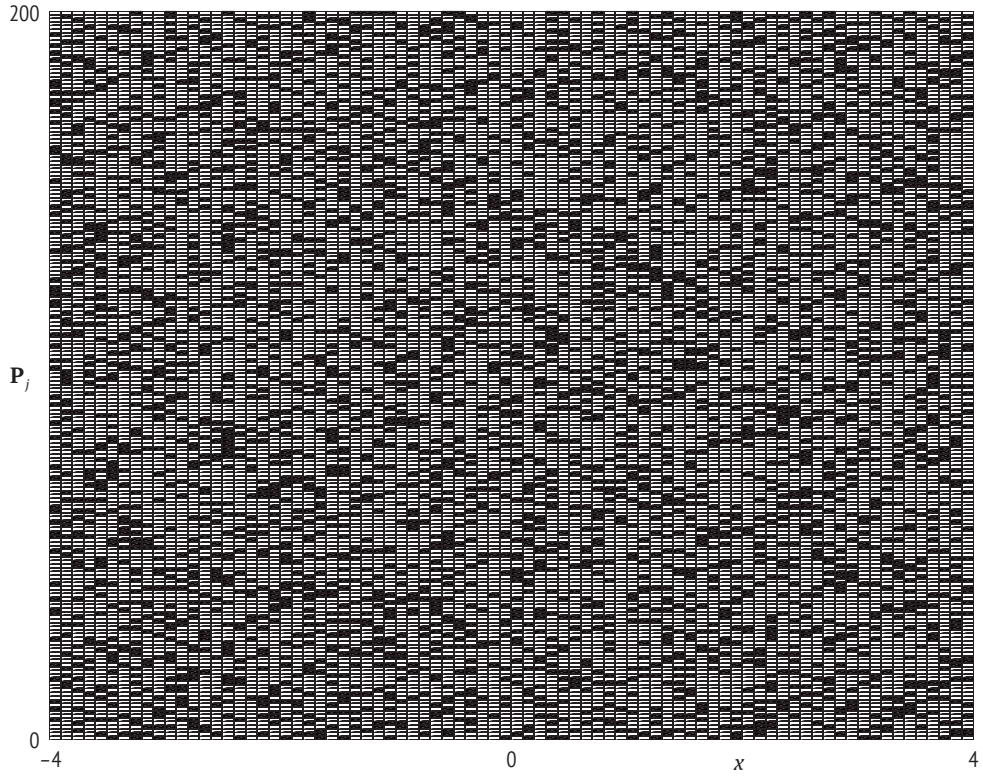

Рис. 12.6 ❖ 200 векторов $\mathbf{P}_j$, построенных по 20 % случайно выбранных измерений, на основе которых созданы графики рис. 12.5. Каждая строка содержит результаты одного испытания (от 1 до 200), а столбцы представляют точки измерения в области $x \in [-4, 4]$ с шагом $\Delta x = 0.1$

Этот алгоритм не оптимален и не дает никаких гарантий. Тем не менее на практике он работает вполне прилично, поскольку конфигурации датчиков с малым числом обусловленности обеспечивают хорошую реконструкцию по POD-модам.

Мы применим данный алгоритм к построению матрицы неполных измерений **P**. Как и раньше, в качестве базисных мод выбираются многочлены Эрмита, определяемые формулами (11.25) и (11.26). Алгоритм построения **P** описан выше. Отметим, что этот алгоритм размещает по одному датчику за раз, так что при 10-модовом разложении система остается недоопределенной, пока не будет размещено 10 датчиков. Поэтому для первых 9 датчиков получаются числа обусловленности порядка 10^{16}. Это также означает, что положения первых 10 датчиков могут быть найдены в результате неточных вычислений числа обусловленности.

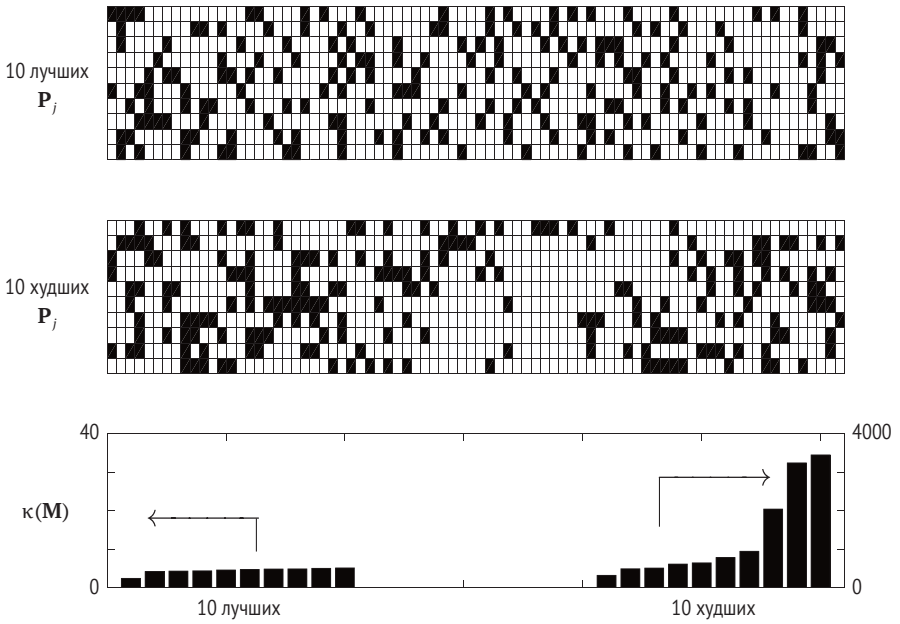

Рис. 12.7 ❖ 10 лучших и 10 худших векторов измерений $\mathbf{P}_j$, построенных по 20%-ной выборке (см. рис. 12.5 и 12.6). На верхнем рисунке видно, что лучшие векторы довольно равномерно распределены в области $x \in [-4, 4]$ с шагом $\Delta x = 0.1$. С другой стороны, для худших векторов (средний рисунок) наблюдаются протяженные пустые участки вблизи центра области, что приводит к большому числу обусловленности κ(**M**). На нижнем рисунке приведена столбчатая диаграмма лучших и худших значений числа обусловленности. Заметим, что на 20%-ной выборке числа обусловленности различаются на два порядка, что говорит о важности хорошего выбора мест размещения датчиков

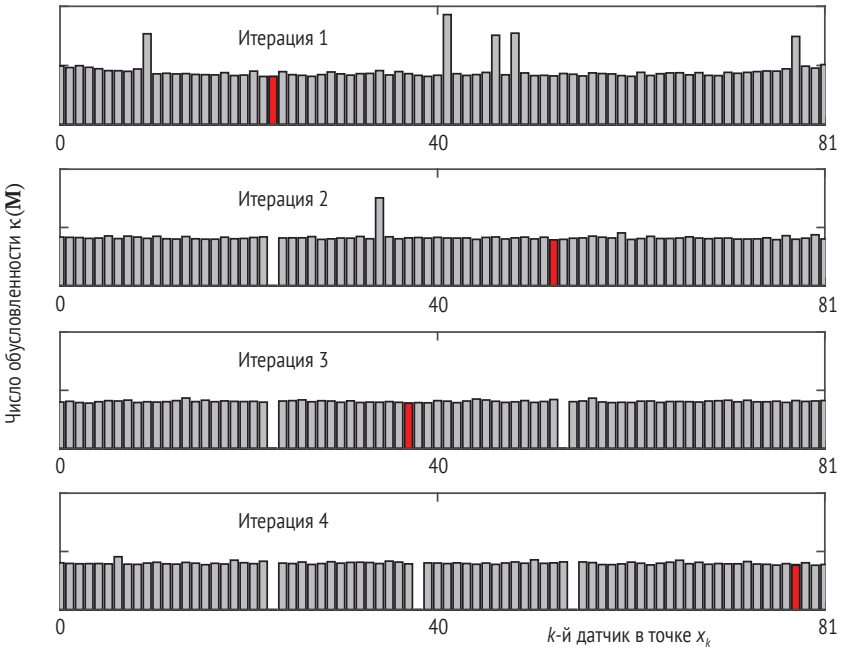

Рис. 12.8 ❖ Первые четыре итерации алгоритма размещения датчиков для неполных измерений из работы Willcox [555]. Алгоритм применяется к 10-модовому разложению по многочленам Эрмита (11.25) и (11.26), дискретизированным на интервале $x \in [-4, 4]$ с шагом $\Delta x = 0.1$. На верхнем рисунке показано число обусловленности к(**M**) для случая, когда по одному датчику размещено в каждой из 81 точки x_k. Первый датчик, минимизирующий число обусловленности (показан красным цветом), размещен в точке x_{23}. Из датчиков в оставшихся 80 точках минимальное число обусловленности приходится на датчик x_{52} (показан красным цветом). Повторяя этот процесс, находим третий и четвертый датчики в точках x_{37} и x_{77} соответственно. После того как место размещения датчика выбрано, оно не рассматривается на следующих итерациях – на рисунках таким местам соответствуют пустые позиции

Следующая программа основана на программе из листинга 12.1, которая используется для порождения 10-модового разложения по многочленам Эрмита. Она минимизирует число обусловленности и находит положения первых 20 датчиков. Таким образом, программа реализует систематический способ построения матрицы измерений **P**, которая обеспечивает хорошую реконструкцию по POD-модам при ограниченном числе измерений.

Листинг 12.5 ❖ Размещение датчиков: минимизация числа обусловленности

```
n2=20; % число датчиков
nall=1:n; ns=[]; %
for jsense=1:n2
    for jloop=1:(n-jsense)
        P=zeros(n,1); P(ns)=1;
        P(nall(jloop))=1;
        for j=1:10
            for jj=1:j % матрица M
                Area=trapz(x,P.*(yharm(:,j).*yharm(:,jj)));
                M2(j,jj)=Area; M2(jj,j)=Area;
            end
        end
        con(jloop)=cond(M2); % вычислить число обусловленности
    end % конец перебора точек
    [s1,n1]=min(con); % точка, в которой достигается минимум
    kond(jsense)=s1; clear con
    ns=[ns nall(n1)]; % добавить точку размещения датчика
    nall=setdiff(nall,ns); % новые индексы датчиков
    P=zeros(n,1); P(ns)=1;
    Psum(:,jsense)=P;
    for j=1:10
        for jj=1:j
            Area=trapz(x,P.*(yharm(:,j).*yharm(:,jj)));
            M2(j,jj)=Area; M2(jj,j)=Area;
        end
    end
    for j=1:10 % реконструкция по неполным измерениям
        ftild(j,1)=trapz(x,P.*(f.*yharm(:,j)));
    end
    atild=M2\ftild; % вычислить ошибку
    f1(:,jsense)=yharm*atild; % реконструкция на данной итерации
    E(jsense)=norm(f1(:,jsense)-f); % ошибка на данной итерации
end % конец цикла размещения датчиков
```

Помимо нахождения точек размещения первых 20 датчиков, эта программа реконструирует тестовую функцию (12.11) на каждой итерации цикла. Обратите внимание на команду **setdiff**, которая исключает точку размещения очередного датчика из рассмотрения на следующей итерации.

Чтобы оценить алгоритм разреженного размещения датчиков, мы запоминаем число обусловленности как функцию от числа итераций. Кроме того, на каждой итерации производится реконструкция тестовой функции (12.11) и вычисляется среднеквадратическая ошибка. На рис. 12.9 показано, как изменяются характеристики реконструкции при увеличении числа датчиков. По построению алгоритм минимизирует число обусловленности $\kappa(\mathbf{M})$ на каждой итерации, поэтому по мере добавления датчиков число обусловленности монотонно убывает (верхний рисунок). Обратите внимание, как резко уменьшается число обусловленности после выбора 10 датчиков, – это объясняется тем, что система перестает быть недоопределенной (с теоретически бесконечным числом обусловленности). Среднеквадратическая ошибка

реконструкции в целом ведет себя примерно так же, но убывание не монотонное, как в случае числа обусловленности. После 10 измерений СКО также резко уменьшается. В общем случае при разложении по r POD-модам результаты становятся разумными только после размещения первых r датчиков.

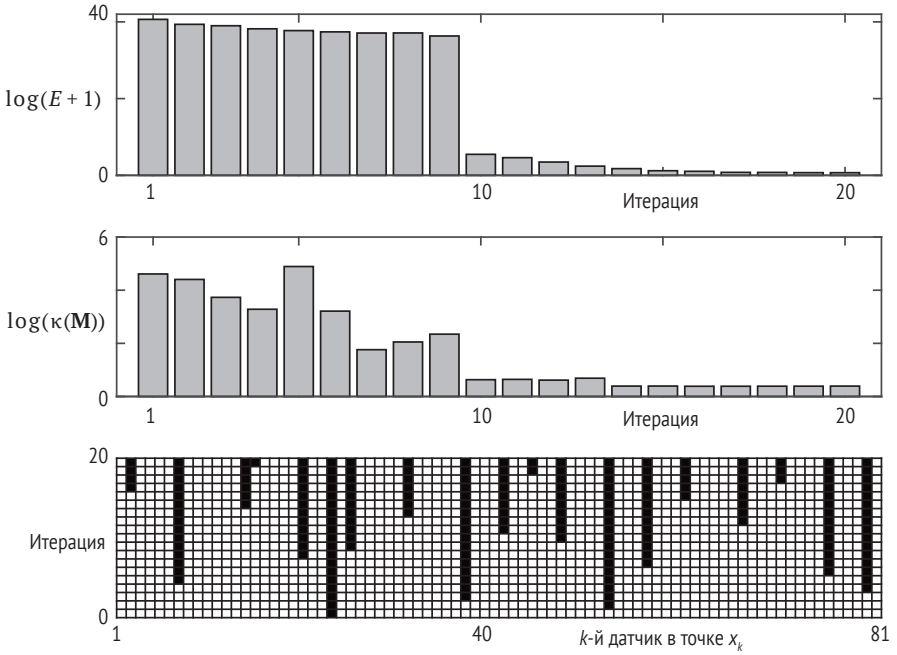

Рис. 12.9 ❖ Число обусловленности и среднеквадратическая ошибка (логарифмы) как функции от числа итераций в алгоритме разреженного размещения датчиков. Логарифм числа обусловленности log[к(**M**)] монотонно убывает, поскольку он минимизируется на каждом шаге. Логарифм среднеквадратической ошибки реконструкции тестовой функции (12.11) также демонстрирует тенденцию к убыванию при увеличении числа датчиков. После того как размещено 10 датчиков, ранг системы становится полным и число обусловленности снижается сразу на несколько порядков. На нижнем рисунке показано, какие датчики включаются (черные квадратики) на каждой из первых 20 итераций. Например, первый датчик размещается в позиции x_{23}

Теперь рассмотрим размещение датчиков как функцию от номера итерации (рис. 12.9 внизу). Мы показываем, какие датчики идентифицированы на каждой итерации. Первым размещен датчик x_{23}, за ним x_{52}, x_{37} и x_{77}. Процесс продолжается, пока не будут определены первые 20 датчиков. Характер размещения датчиков важен, поскольку мы видим, что выборка из области приблизительно равномерная. Ниже будут рассмотрены другие схемы.

И напоследок рассмотрим реконструкцию тестовой функции (12.11) по мере увеличения количества датчиков. Как и следовало ожидать, чем больше датчиков производят неполные измерения, тем лучше реконструкция, особенно если датчики размещаются систематически, в соответствии с ал-

горитмом из работы Wilcox [555]. На рис. 12.10 показано, как изменяется реконструированная функция при увеличении номера итерации, отложенного по оси *z*. На левом рисунке видно, что реконструированная функция поначалу ничем не напоминает исходную, но постепенно реконструкция сходится к тестовой функции. Для сравнения показана также истинная тестовая функция. Этот рисунок убеждает нас, что при систематическом размещении датчиков реконструкция действительно сходится к решению.

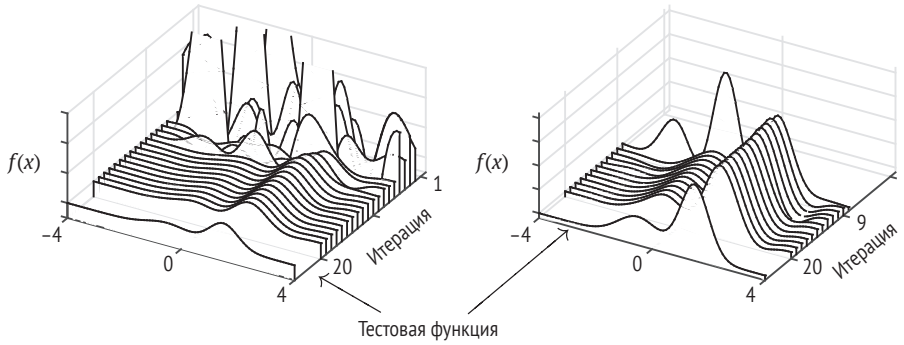

Рис. 12.10 ❖ Сходимость реконструкции к тестовой функции (12.11). На левом рисунке видно, что на начальных итерациях, когда число датчиков мало, ошибки реконструкции велики. Действительно, мы знаем, что на первых девяти итерациях число обусловленности и среднеквадратическая ошибка очень велики, потому что система не достигла полного ранга. На правом рисунке в увеличенном масштабе показаны результаты на итерациях с 9 по 20, когда сходимость наблюдается отчетливо. На обоих рисунках для сравнения приведена сама тестовая функция

Замены числа обусловленности

В конце этого раздела мы рассмотрим альтернативы числу обусловленности $\kappa(\mathbf{M})$. Вычисление числа обусловленности само по себе требует много ресурсов. Кроме того, если производится разложение по *r* POD-модам, то пока не будет размещено *r* датчиков, это вычисление численно неустойчиво. Однако мы понимаем, какая именно цель ставится перед алгоритмом минимизации числа обусловленности: приблизить матрицу измерений **M** к единичной. Это подсказывает идею альтернативного алгоритма, также разработанного Уиллкоксом [555].

1. Для каждой возможной точки вычислить разность между суммой диагональных и суммой внедиагональных элементов матрицы **M**. Обозначим эту величину $\kappa_2(\mathbf{M})$. Рассматриваются только точки, в которых датчиков еще нет.
2. Найти точку, в которой эта величина достигает максимума. В этой точке размещается *k*-й датчик.
3. Повторить шаги 1–2 для $(k + 1)$-го датчика.

Этот алгоритм является простой модификацией исходного алгоритма, минимизирующего число обусловленности. В листинге 12.5 нужно изме-

нить всего две строки. Туда, где вычисляется число обусловленности, следует включить строку:

```
nall=setdiff(nall,ns); % новые индексы датчиков
```

Кроме того, датчики размещаются там, где достигается максимум, поэтому применима следующая строка:

```
P=zeros(n,1); P(ns)=1;
```

Таким образом, изменив всего две строки, мы реализовали новую метрику, обходящуюся без вычисления числа обусловленности.

Чтобы оценить этот новый алгоритм разреженного размещения датчиков, мы построим график новой метрики как функции от числа итераций, а также график среднеквадратической ошибки реконструкции функции (12.11). В этом случае рассматривается 60 датчиков, потому что сходимость более медленная. На рис. 12.11 показаны результаты работы алгоритма при увеличении числа датчиков. По построению алгоритм максимизирует разность между суммой диагональных и суммой внедиагональных элементов на каждой итерации, поэтому при добавлении датчиков эта величина монотонно возрастает (рис. 12.11 слева вверху). Среднеквадратическая ошибка реконструкции убывает, но не монотонно. И еще – алгоритм сходится довольно медленно. По крайней мере в этом примере, метод работает хуже, чем основанный на числе обусловленности. Но в некоторых случаях его качество удается улучшить [555], а с вычислительной точки зрения он гораздо эффективнее.

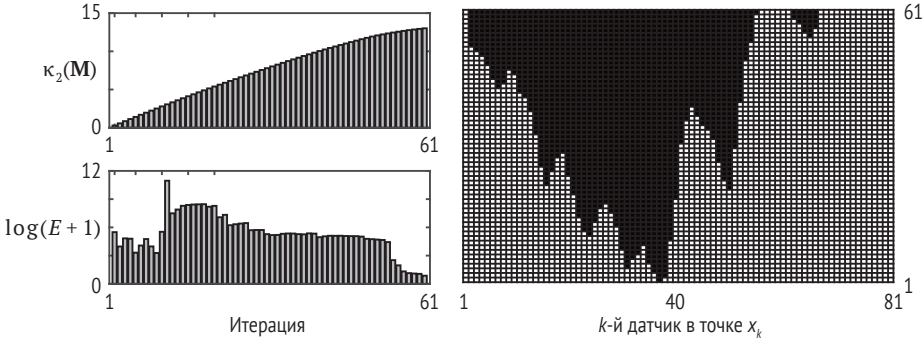

Рис. 12.11 ❖ Сумма диагональных элементов минус сумма внедиагональных (слева вверху) и среднеквадратическая ошибка (логарифм) как функции от числа итераций во втором алгоритме разреженного размещения датчиков. Новый вариант числа обусловленности монотонно возрастает, поскольку максимизируется на каждой итерации. Логарифм среднеквадратической ошибки реконструкции тестовой функции (12.11) имеет тенденцию к уменьшению при увеличении числа датчиков. На правом рисунке показано, какие датчики включаются (черные квадратики) на каждой из первых 60 итераций. Например, первый датчик размещается в позиции x_{37}

Как и раньше, в правой части рис. 12.11 показано, какие датчики включаются на каждой из первых 60 итераций. Первый датчик размещается в позиции x_{37}, за ним следуют датчики в позициях x_{38}, x_{36} и x_{31}. Характер включения датчиков совсем не такой, как в алгоритме минимизации числа обусловленности, – позиции датчиков не распределены равномерно по всей области.

12.4. Неполные измерения: максимальная дисперсия

В предыдущем разделе мы описали систематические способы определения положения датчиков для неполных измерений в алгоритме POD. Результаты оказались гораздо лучше, чем при случайном размещении датчиков. Минимизация числа обусловленности позволила существенно увеличить точность и снизить среднеквадратическую ошибку реконструкции. Но у этого метода два недостатка. Во-первых, сам алгоритм затратный, поскольку приходится вычислять число обусловленности для каждого возможного положения датчика, т. е. производить исчерпывающий поиск. Во-вторых, алгоритм плохо обусловлен до момента размещения r-го датчика, где r – число POD-мод. В этом случае число обусловленности теоретически равно бесконечности, а на практике имеет порядок 10^{17}.

В работе Karniadakis et al. [565] предложен альтернативный алгоритм для преодоления вычислительных трудностей. Вместо одного датчика на каждой итерации на первой же итерации размещается сразу r датчиков. Тогда построенная матрица больше не является плохо обусловленной.

В алгоритме Карниадакиса предложен также систематический способ выбора первых r датчиков. Для их размещения выбираются точки экстремума POD-мод, которые по построению призваны уловить направления максимальной дисперсии данных. Алгоритм состоит из следующих шагов.

1. Сначала разместить r датчиков.
2. Точки размещения этих датчиков совпадают с максимумами POD-мод $\boldsymbol{\psi}_k$.
3. Добавить датчики в точках следующих экстремумов POD-мод.

Приведенная ниже программа находит максимумы мод и строит матрицу неполных измерений $\mathbf{P}$ для этих точек.

Листинг 12.6 ❖ Разреженное размещение: максимизация дисперсии

```
ns=[];
for j=1:10 % перебрать моды
        [s1,n1]=max(yharm(:,j)); % выбрать максимум
        ns=[ns n1];
end
P=zeros(n,1); P(ns)=1;
```

Если измерений всего r, то качество этого алгоритма не слишком высокое, но, по крайней мере, число обусловленности вычисляется устойчиво. Чтобы улучшить качество, можно было бы использовать также минимумы мод $\boldsymbol{\psi}_k$, т. е. рассматривать максимальную и минимальную дисперсии. В случае

гармонического осциллятора у первой моды нет минимума, потому что она асимптотически убывает при $x \to \pm\infty$. Поэтому следующая программа выбирает 19 мест для размещения датчиков.

Листинг 12.7 ❖ Разреженное размещение: максимальная и минимальная дисперсии

```
ns=[];
for j=1:10 % перебрать моды
    [s1,n1]=max(yharm(:,j)); % выбрать максимум
    ns=[ns n1];
end
for j=2:10
    [s2,n2]=min(yharm(:,j)); % выбрать минимум
    ns=[ns n2];
end
P=zeros(n,1); P(ns)=1;
```

Заметим, что в данном случае количество датчиков почти вдвое больше, чем в предыдущем. Кроме того, поиск производится лишь в тех местах, где изменчивость наибольшая, – интуитивно понятно, что это самые подходящие точки для измерения.

В общем случае алгоритм Карниадакиса предполагает случайное размещение p датчиков в точках M экстремумов с последующей модификацией точек размещения с целью улучшить число обусловленности. Чтобы произвести выборку, необходимо найти все максимумы и минимумы POD-мод. Моды гармонического осциллятора вместе с их максимумами и минимумами показаны на рис. 12.12. В листинге 12.8 приведена программа, которая находит экстремумы мод для использования в алгоритме разреженного размещения.

Листинг 12.8 ❖ Разреженное размещение: нахождение экстремумов

```
nmax=[]; nmin=[];
Psum = zeros(n,10);
for j=1:10 % перебрать моды
    nmaxt=[]; nmint=[];
    for jj=2:n-1
        if yharm(jj,j)>yharm(jj-1,j) & yharm(jj,j)>yharm(jj+1,j)
            nmax=[nmax jj];
            nmaxt=[nmaxt jj];
        end
        if yharm(jj,j)<yharm(jj-1,j) & yharm(jj,j)<yharm(jj+1,j)
            nmin=[nmin jj];
            nmint=[nmint jj];
        end
    end
    nst=[nmaxt nmint]
    Psum(nst,j)=1;
end
ns=[nmax nmin];
ni=randsample(length(ns),20);
nsr=ns(ni);
P=zeros(n,1); P(nsr)=1;
```

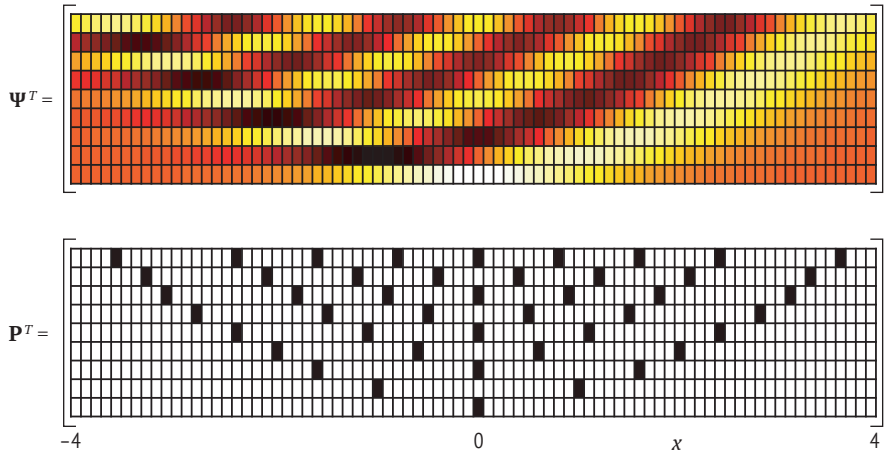

Рис. 12.12 ❖ На верхнем рисунке показана структура мод **Ψ** при низкоранговой аппроксимации с помощью POD-разложения по многочленам Эрмита. Выбран интервал дискретизации $x \in [-4, 4]$ с шагом $\Delta x = 0.1$. На цветовой карте показаны максимумы (белым цветом) и минимумы (черным цветом) мод. На нижнем рисунке показаны ячейки сетки, соответствующие экстремумам дисперсии. Эти экстремумы – кандидаты для размещения датчиков, т. е. для построения матрицы измерений **P**. Обычно в начале вычисления разреженного размещения берут случайную выборку из этих экстремумов

Заметим, что результирующий вектор **ns** содержит все 55 экстремумов. В этом вычислении предполагается, что данные довольно гладкие, поэтому для нахождения экстремумов достаточно просто рассмотреть соседние точки, т. е. найден максимум, если значения в двух соседних с ним точках больше, и минимум – если значения в двух соседних точках меньше.

Алгоритм максимальной дисперсии предлагает проверять различные конфигурации датчиков в точках экстремумов. В частности, если требуется 20 измерений, то нужно будет проверить различные комбинации 20 датчиков в 55 возможных точках. Такой комбинаторный поиск требует слишком много вычислений. Но если просто взять 100 случайных конфигураций и выбрать из них наилучшую, то результат получится близким к алгоритму минимизации числа обусловленности. В листинге 12.9 показана полная реализация этого алгоритма вместе с вычислением числа обусловленности и среднеквадратической ошибки для тестовой функции (12.11).

Листинг 12.9 ❖ Разреженное размещение: случайная выборка

```
ntot=length(ns);
for jtrials=1:100
    ni=randsample(ntot,20);
    nsr=ns(ni);

    P=zeros(n,1); P(nsr)=1;

    for j=1:10
        for jj=1:j
```

```
            Area=trapz(x,P.*(yharm(:,j).*yharm(:,jj)));
            M2(j,jj)=Area; M2(jj,j)=Area;
        end
    end

    for j=1:10 % реконструкция по неполным измерениям
        ftild(j,1)=trapz(x,P.*(f.*yharm(:,j)));
    end
    atild=M2\ftild; % вычислить ошибку
    f1=yharm*atild; % реконструкция на данной итерации
    E_tri(jtrials)=norm(f1-f); % ошибка на данной итерации
    con_tri(jtrials)=cond(M2);
end
subplot(2,1,1), bar(log(con_tri),'Facecolor',[0.7 0.7 0.7])
subplot(2,1,2), bar(log(E_tri+1),'Facecolor',[0.7 0.7 0.7])
```

На рис. 12.13 показаны число обусловленности и среднеквадратическая ошибка для 100 испытаний. Все конфигурации работают лучше, чем случайные измерения, но некоторые демонстрируют просто отличное качество.

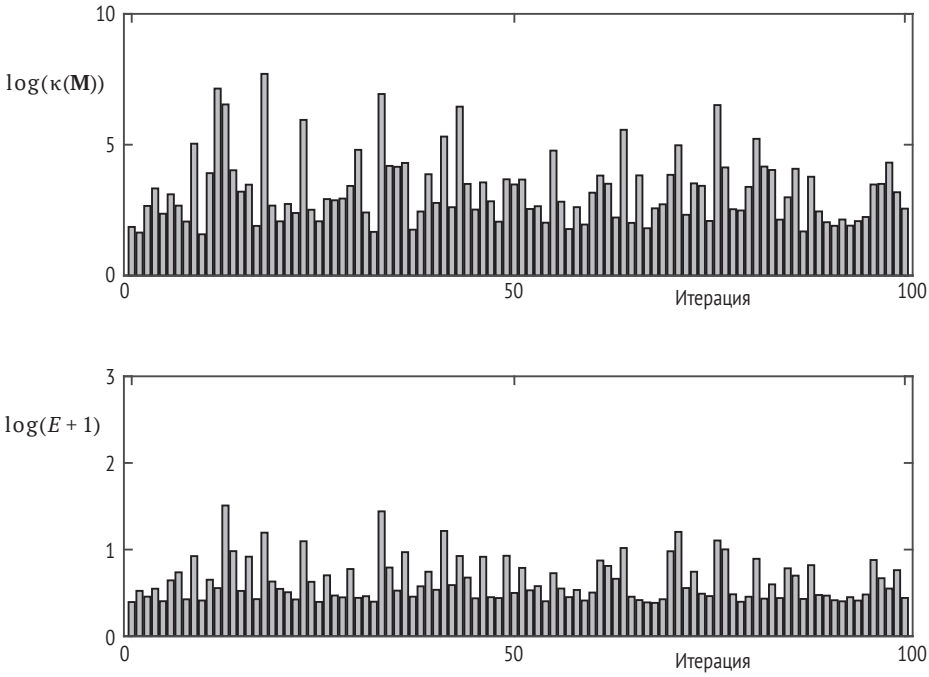

Рис. 12.13 ❖ Число обусловленности и среднеквадратическая ошибка реконструкции тестовой функции (12.11) в 100 случайных испытаниях, в ходе которых выбирается 20 положений датчиков из 55 возможных экстремумов, изображенных на рис. 12.12. Сто испытаний дают ряд конфигураций датчиков, при которых достигается примерно такое же качество, как в алгоритме минимизации числа обусловленности из предыдущего раздела. Однако вычислительные затраты на проведение таких испытаний могут оказаться гораздо ниже

На рис. 12.14 приведено прямое сравнение всех этих методов. А именно показаны результаты для размещения датчиков (a) в точках максимума POD-мод, (b) в точках максимума и минимума POD-мод и (c) в 20 точках, случайно выбранных из 55 точек экстремума POD-мод. Они сравниваются с (d) размещением 5 лучших из 20 датчиков в точках экстремума, выбранных в 100 случайных испытаниях, и (e) алгоритмом минимизации числа обусловленности (красный столбик). Алгоритм максимальной дисперсии работает примерно так же хорошо, как алгоритм минимизации числа обусловленности. Но при этом он быстрее и никогда не вычисляет числа обусловленности плохо обусловленных матриц. В работе Karniadakis et al. [565] предложены также модификации этой базовой реализации. В частности, предложено рассматривать по одному датчику за раз и пытаться разместить его во всех остальных доступных точках. Если число обусловленности уменьшается, то датчик перемещается в новую точку и рассматривается следующий датчик.

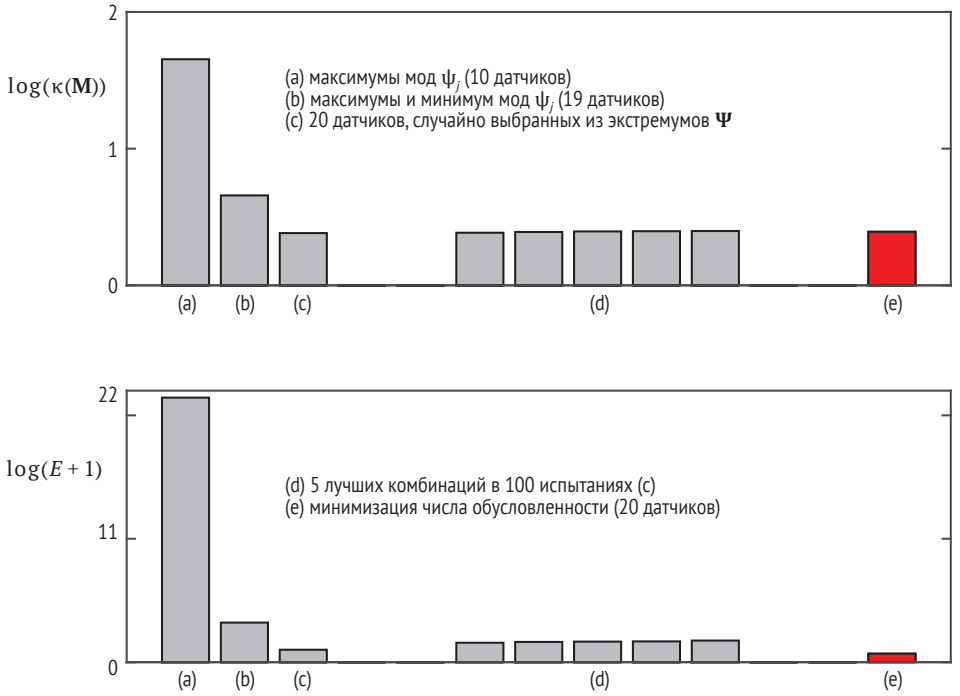

Рис. 12.14 ❖ Метрики качества размещения датчиков на основе экстремумов дисперсии POD-мод. Рассматривается среднеквадратическая ошибка реконструкции тестовой функции (12.11) и число обусловленности. Показаны результаты при использовании (a) точек максимума POD-мод, (b) точек максимума и минимума POD-мод и (c) случайной выборки 20 из 55 точек экстремума POD-мод. Результаты сравниваются с (d) 5 лучшими комбинациями 20 датчиков в 100 случайных испытаниях и (e) алгоритмом минимизации числа обусловленности (красный столбик). Размещение датчиков в случайно выбранных точках экстремума дает результат, близкий к минимизации числа обусловленности со значительно меньшими затратами

12.5. POD и дискретный эмпирический метод интерполяции (DEIM)

Цель рассмотренного выше метода POD – воспользоваться низкоразмерной динамикой, присутствующей во многих системах высокой размерности. POD часто используется для построения моделей пониженного порядка (ROM), которые приобретают все большую важность в научных приложениях и расчетах. ROM позволяют уменьшить вычислительную сложность и время решения сложных крупномасштабных систем [53, 442, 244, 17]. Точнее, ROM предлагают систематический подход к аппроксимации многомерных пространственно-временных систем [139], часто возникающих в процессе численной дискретизации, низкоразмерными подпространствами, порождающими почти такие же характеристики входа-выхода исходной нелинейной динамической системы. Однако, несмотря на значительное снижение размерности с помощью POD-базиса, вычисление нелинейных членов высшего порядка может быть столь же сложным, как в исходной задаче [41, 127]. Эмпирический метод интерполяции (EIM) и его упрощенный дискретный вариант (DEIM), применяемые для собственного ортогонального разложения (POD) [347, 251], преодолевают эту трудность, предлагая вычислительно эффективный способ дискретной (разреженной) выборки и вычисления нелинейности. Эти методы гарантируют, что вычислительная сложность ROM будет хорошо масштабироваться с повышением ранга аппроксимации даже для сложных нелинейностей.

EIM разработан с целью эффективного управления вычислением нелинейности в схемах понижения размерности, а DEIM специально адаптирован к POD и проекции Галеркина. В DEIM нелинейность аппроксимируется с помощью небольшой дискретной выборки точек, определяемой алгоритмически. Это гарантирует, что стоимость вычисления нелинейности масштабируется с увеличением ранга POD-базиса. В качестве примера рассмотрим упрощение POD–Галеркина с r модами. При наличии простой кубической нелинейности требуется возводить аппроксимацию POD–Галеркина в куб, а это выливается в r^3 операций для вычисления нелинейного члена. DEIM аппроксимирует кубическую нелинейность, выбирая $O(r)$ дискретных точек, и тем самым сохраняет сложность вычислений на уровне $O(r)$, чего мы и добиваемся. В подходе DEIM проекция сочетается с интерполяцией. Точнее, в DEIM используются выбранные точки интерполяции для определения основанной на интерполяции проекции на почти оптимальное по норме ℓ_2 подпространство, аппроксимирующей нелинейность. EIM/DEIM – не единственные методы, призванные уменьшить сложность вычисления нелинейных членов; см. также методы оценки отсутствующей точки (MPE) [400, 21] и неполного POD [555, 565, 120, 462]. Однако они оказались успешными во многих приложениях и моделях из разных областей [127]. Как бы то ни было, в MPE, неполном POD и EIM/DEIM используется небольшая выборка точек из пространственной решетки, чтобы избежать дорогостоящего вычисления скалярных произведений, связанного с нелинейными членами.

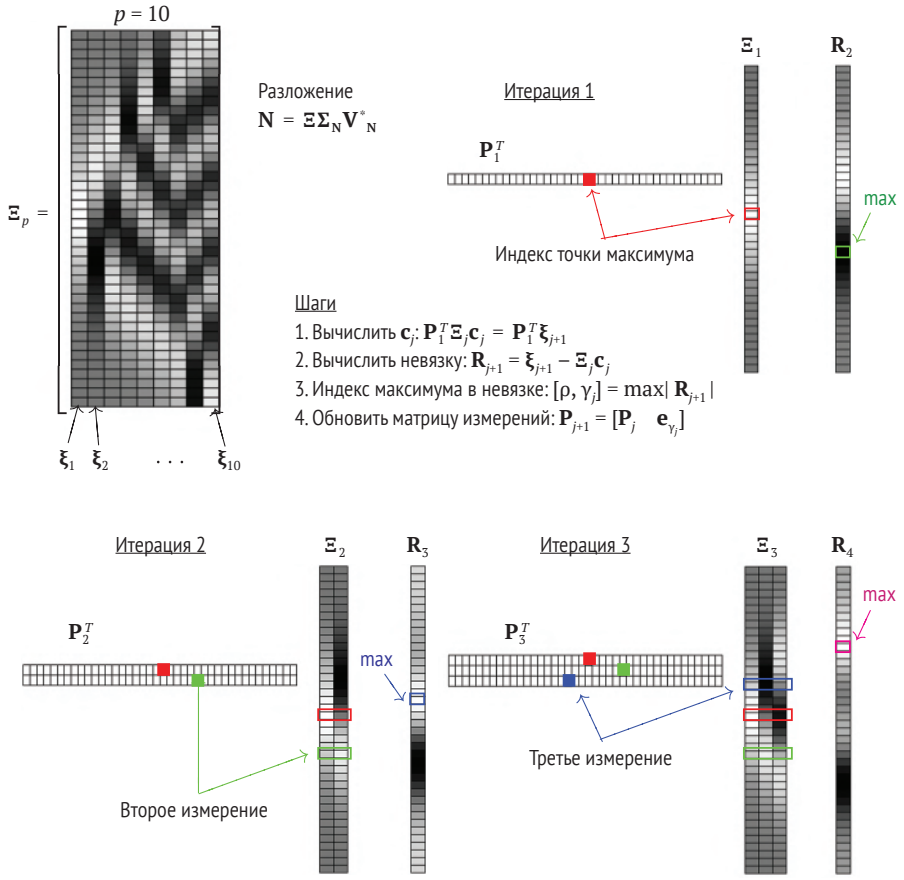

Рис. 12.15 ❖ Первые три итерации алгоритма DEIM. Только в целях иллюстрации предполагается, что матрица нелинейности $\mathbf{N} = \Xi\Sigma_N\mathbf{V}^*_N$ образована модами гармонического осциллятора, причем первые десять мод образуют Ξ_p. Точка начального измерения совпадает с максимумом первой моды $\boldsymbol{\xi}_1$. Далее выполняется трехшаговый процесс выбора последующих точек измерений, основанный на поиске максимума вектора невязок $\mathbf{R}_j$. Показаны первая (красная), вторая (зеленая) и третья (синяя) точки измерения и построение выборочной матрицы $\mathbf{P}$

POD и DEIM

Рассмотрим многомерную систему нелинейных дифференциальных уравнений, которая может возникнуть, например, при конечноразностной дискретизации дифференциального уравнения в частных производных. В дополнение к построению матрицы снимка, соответствующей решению УрЧП, с целью выделить POD-моды алгоритм DEIM также строит матрицу снимка для нелинейного члена УрЧП:

$$\mathbf{N} = \begin{bmatrix} | & | & & | \\ \mathbf{N}_1 & \mathbf{N}_2 & \cdots & \mathbf{N}_m \\ | & | & & | \end{bmatrix}, \tag{12.12}$$

где столбцы $\mathbf{N}_k \in \mathbb{C}^n$ – результаты вычисления нелинейности в момент t_k.

Для получения решений с высокой точностью n обычно очень велико, поэтому вычисление решения обходится чрезвычайно дорого или вообще невозможно. Метод POD–Галеркина дает теоретически обоснованную схему понижения размерности, которая аппроксимирует функцию $\mathbf{u}(t)$ функциями, образующими оптимальный базис ранга r, где $r \ll n$. В предыдущей главе было показано, что этот оптимальный базис вычисляется с помощью сингулярного разложения ряда снимков сложной системы в последовательные моменты времени.

Стандартная процедура POD [251] повсеместно применяется для построения моделей пониженного порядка. Однако остается место и для таких инноваций, как DEIM, неполное POD и MPE. Рассмотрим нелинейную компоненту низкоразмерной эволюции (11.21): $\mathbf{\Psi}^T\mathbf{N}(\mathbf{\Psi a}(t))$. Для простой нелинейности $N(u(x, t)) = u(x, t)^3$ рассмотрим ее влияние на пространственную дискретизацию POD-разложения по двум модам: $u(x, t) = a_1(t)\psi_1(x) + a_2(t)\psi_2(x)$. В алгоритме вычисления нелинейности необходимо вычислять выражение

$$u(x,t)^3 = a_1^3\psi_1^3 + 3a_1^2a_2\psi_1^2\psi_2 + 3a_1a_2^2\psi_1\psi_2^2 + a_2^3\psi_2^3. \tag{12.13}$$

Затем следовало бы вычислить динамику $a_1(t)$ и $a_2(t)$ путем проецирования на низкоразмерный базис, для чего нужно вычислить скалярное произведение этого нелинейного члена с ψ_1 и ψ_2. Таким образом, мало того что объем вычислений удваивается, так еще и необходимо вычислять скалярные произведения с n-мерными векторами. Метод DEIM и ему подобные обходят это сложное вычисление.

DEIM

Как отмечено в предыдущем разделе, недостатки метода POD–Галеркина связаны с необходимостью вычислять нелинейный член $\mathbf{N}(\mathbf{\Psi a}(t))$. Чтобы избежать этой трудности, DEIM аппроксимирует $\mathbf{N}(\mathbf{\Psi a}(t))$ посредством проецирования и интерполяции, а не вычисляет напрямую. Точнее, вычисляется низкоранговое представление нелинейности по сингулярному разложению

$$\mathbf{N} = \mathbf{\Xi\Sigma}_\mathbf{N}\mathbf{V}_\mathbf{N}^*, \tag{12.14}$$

где матрица $\mathbf{\Xi}$ содержит оптимальный базис для нелинейности. А именно мы рассматриваем базис ранга p

$$\mathbf{\Xi}_p = [\boldsymbol{\xi}_1 \quad \boldsymbol{\xi}_2 \quad \cdots \quad \boldsymbol{\xi}_p], \tag{12.15}$$

который аппроксимирует нелинейную функцию ($p \ll n$ и $p \sim r$). Аппроксимация нелинейности $\mathbf{N}$ имеет вид:

$$\mathbf{N} \approx \Xi_p \mathbf{c}(t), \tag{12.16}$$

где $\mathbf{c}(t)$ аналогично $\mathbf{a}(t)$ в (11.20). Поскольку это сильно переопределенная система, подходящий вектор $\mathbf{c}(t)$ можно найти, выбрав p строк системы. Алгоритм DEIM как раз и предназначен для поиска этих p строк.

Алгоритм DEIM начинается с рассмотрения векторов $\mathbf{e}_{\gamma_j} \in \mathbb{R}^n$, являющихся γ_j-ми столбцами n-мерной единичной матрицы. Затем строится матрица проекции $\mathbf{P} = [\mathbf{e}_{\gamma_1} \quad \mathbf{e}_{\gamma_2} \quad \cdots \quad \mathbf{e}_{\gamma_p}]$, которая выбирается так, чтобы $\mathbf{P}^T\Xi_p$ была несингулярной. После этого $\mathbf{c}(t)$ однозначно определяется из уравнения $\mathbf{P}^T\mathbf{N} = \mathbf{P}^T\Xi_p\mathbf{c}(t)$, и, следовательно,

$$\mathbf{N} \approx \Xi_p(\mathbf{P}^T\Xi_p)^{-1}\mathbf{P}^T\mathbf{N}. \tag{12.17}$$

Из этого результата вытекает важнейшее для упрощения модели следствие: для вычисления нелинейности член $\mathbf{P}^T\mathbf{N}$ нужно вычислить лишь в $p \ll n$ точках. DEIM далее предлагает систематический метод выбора базисных векторов $\boldsymbol{\xi}_j$ и индексов γ_j. Этот алгоритм, основанный на жадном поиске, детально описан в работе [127] и более кратко в табл. 12.1.

Таблица 12.1. Алгоритм DEIM нахождения аппроксимирующего базиса для нелинейности и точек его интерполяции. Сначала алгоритм строит моды базиса нелинейной системы и инициализирует первую точку измерения и матрицу $\mathbf{P}_1$, используя в этом качестве точку максимума $\boldsymbol{\xi}_1$. Затем алгоритм итеративно строит столбцы $\mathbf{P}_j$, находя максимум невязки $\mathbf{R}_j$

Алгоритм DEIM	
Построение базиса и инициализация	
• собрать данные, построить матрицу снимка • построить матрицу нелинейного снимка • сингулярное разложение $\mathbf{N}$ • построить аппроксимирующий базис ранга p • выбрать первый индекс (инициализация) • построить первую матрицу измерений	$\mathbf{X} = [\mathbf{u}(t_1) \;\; \mathbf{u}(t_2) \;\; \cdots \;\; \mathbf{u}(t_m)]$ $\mathbf{N} = [N(\mathbf{u}(t_1)) \;\; N(\mathbf{u}(t_2)) \;\; \cdots \;\; N(\mathbf{u}(t_m))]$ $\mathbf{N} = \Xi\Sigma_\mathbf{N}\mathbf{V}_\mathbf{N}^*$ $\Xi_p = [\boldsymbol{\xi}_1 \;\; \boldsymbol{\xi}_2 \;\; \cdots \;\; \boldsymbol{\xi}_p]$ $[\rho, \gamma_1] = \max\lvert\boldsymbol{\xi}_1\rvert$ $\mathbf{P}_1 = [\mathbf{e}_{\gamma_1}]$
Точки интерполяции и цикл (j = 1, 2, …, p)	
• вычислить $\mathbf{c}_j$ • вычислить невязку • найти индекс максимальной невязки • добавить столбец в матрицу измерений	$\mathbf{P}_j^T\Xi_j\mathbf{c}_j = \mathbf{P}_j^T\boldsymbol{\xi}_{j+1}$ $\mathbf{R}_{j+1} = \boldsymbol{\xi}_{j+1} - \Xi_j\mathbf{c}_j$ $[\rho, \gamma_j] = \max\lvert\mathbf{R}_{j+1}\rvert$ $\mathbf{P}_{j+1} = [\mathbf{P}_j \; \mathbf{e}_{\gamma_j}]$

Алгоритмы POD и DEIM обладают многими преимуществами, когда речь заходит об упрощении нелинейных моделей сложных систем. POD предлагает теоретически обоснованный способ построения r-мерного подпространства $\boldsymbol{\Psi}$, характеризующего динамику. DEIM дополняет POD, предлагая способ вычисления проблематичных нелинейных членов с использовани-

ем p-мерного подпространства Ξ_p, представляющего нелинейности. Таким образом, вычисление нелинейных членов в модели пониженной размерности можно производить в небольшой выборке точек.

12.6. Реализация алгоритма DEIM

Для демонстрации упрощения модели с помощью DEIM мы снова рассмотрим нелинейное уравнение Шрёдингера (НУШ) (11.29). Программа его численного решения приведена в листингах 11.3 и 11.4. Ее результатом является матрица **usol**, в строках которой находятся снимки в разные моменты времени, а в столбцах – точки пространственной дискретизации. Как и в первом разделе этой главы, мы начнем с того, что транспонируем эту матрицу, так чтобы моментальные снимки находились в столбцах, а не в строках. Следующая программа осуществляет транспонирование и последующее сингулярное разложение для нахождения POD-мод.

Листинг 12.10 ❖ Понижение размерности для НУШ

```
X=usol.';          % матрица данных X
[U,S,W]=svd(X,0); % сингулярное разложение
```

Сингулярное разложение нужно не только для стандартного нахождения POD-мод, но и для алгоритма DEIM, где оно применяется к нелинейному члену. В результате вычисляется низкоранговое представление члена $N(u) = |u|^2 u$ в виде $\mathbf{N} = \boldsymbol{\Xi}\boldsymbol{\Sigma}_\mathbf{N}\mathbf{V}_\mathbf{N}^*$.

Листинг 12.11 ❖ Понижение размерности нелинейности в НУШ

```
NL=i*(abs(X).^2).*X;
[XI,S_NL,W]=svd(NL,0);
```

После того как низкоранговые структуры вычислены, выбирается ранг системы r. Далее мы полагаем $r = p = 3$, так что матрицы стандартных POD-мод и мод нелинейности, $\boldsymbol{\Psi}$ и Ξ_p, имеют по три столбца. Следующая программа выбирает POD-моды для $\boldsymbol{\Psi}$ и проецирует начальное условие на подпространство POD.

Листинг 12.12 ❖ Выбор ранга и POD-мод

```
r=3;           % выбрать ранг упрощенной системы
Psi=U(:,1:r); % выбрать POD-моды
a=Psi'*u0;     % спроецировать начальное условие
```

Теперь построим матрицу интерполяции $\mathbf{P}$, выполнив алгоритм DEIM, описанный в предыдущем разделе. Алгоритм начинается с выбора первой точки интерполяции – точки максимума первой, доминирующей моды Ξ_p.

Листинг 12.13 ❖ Первая точка, выбираемая DEIM

```
[Xi_max,nmax]=max(abs(XI(:,1)));
XI_m=XI(:,1);
z=zeros(n,1);
P=z; P(nmax)=1;
```

Алгоритм строит $\mathbf{P}$ итеративно, добавляя каждый раз по одному столбцу. Следующий шаг алгоритма – вычислить точки интерполяции со второй по r-ю с помощью жадного поиска. А именно вектор $\mathbf{c}_j$ вычисляется по $\mathbf{P}_j^T\boldsymbol{\Xi}_j\mathbf{c}_j = \mathbf{P}_j^T\boldsymbol{\xi}_{j+1}$, где $\boldsymbol{\xi}_j$ – столбцы матрицы нелинейности $\boldsymbol{\Xi}_p$. Точка интерполяции определяется путем поиска максимума невязки $\mathbf{R}_{j+1} = \boldsymbol{\xi}_{j+1} - \boldsymbol{\Xi}_j\mathbf{c}_j$. На каждой итерации алгоритма создается очередной столбец матрицы разреженной интерполяции $\mathbf{P}$. Вектор целых чисел **nmax** определяет индексы точек интерполяции.

Листинг 12.14 ❖ Точки интерполяции со второй по r-ю

```
for j=2:r
    c=(P'*XI_m)\(P'*XI(:,j));
    res=XI(:,j)-XI_m*c;
    [Xi_max,nmax]=max(abs(res));
    XI_m=[XI_m,XI(:,j)];
    P=[P,z]; P(nmax,j)=1;
end
```

Имея матрицу интерполяции, мы готовы построить ROM. Сначала строится линейный член $\boldsymbol{\Psi}^T\mathbf{L}\boldsymbol{\Psi}$ (11.21), где линейным оператором в НУШ является лапласиан. Производные вычисляются с помощью преобразования Фурье.

Листинг 12.15 ❖ Проекция линейных членов

```
for j=1:r % линейные члены, содержащие производные
    Lxx(:,j)=ifft(-k.^2.*fft(Psi(:,j)));
end
L=(i/2)*(Psi')*Lxx; % спроецированный линейный член
```

Для проецирования нелинейности используется матрица интерполяции $\mathbf{P}$ и формула (12.17). Напомним, что нелинейный член (11.21) умножается на $\boldsymbol{\Psi}^T$. Также вычисляется интерполированная версия низкорангового подпространства, натянутого на $\boldsymbol{\Psi}$.

Листинг 12.16 ❖ Проекция нелинейных членов

```
P_NL=Psi'*( XI_m*inv(P'*XI_m) ); % проекция нелинейных членов
P_Psi=P'*Psi; % интерполяция Psi
```

Осталось только сделать шаг интегрирования по времени. Для этого используется схема Рунге–Кутта четвертого порядка.

Листинг 12.17 ❖ Шаг по времени обсчета ROM

```
[tt,a]=ode45('rom_deim_rhs',t,a,[],P_NL,P_Psi,L);
Xtilde=Psi*a'; % DEIM-аппроксимация
waterfall(x,t,abs(Xtilde')), shading interp, colormap gray
```

Правая часть интегратора по времени теперь стала низкоразмерной.

Листинг 12.18 ❖ Правая часть ROM

```
function rhs=rom_deim_rhs(tspan, a,dummy,P_NL,P_Psi,L)
N=P_Psi*a;
rhs=L*a + i*P_NL*((abs(N).^2).*N);
```

Сравнение динамики полной модели и упрощенной модели ранга $r = 3$ ROM с тремя точками интерполяции, найденными DEIM, показано на рис. 12.16. Там же показаны положения точек интерполяции относительно POD-мод. Все точки, кроме первой, не совпадают ни с минимумами, ни с максимумами POD-мод. Алгоритм размещает их, так чтобы максимизировать невязку.

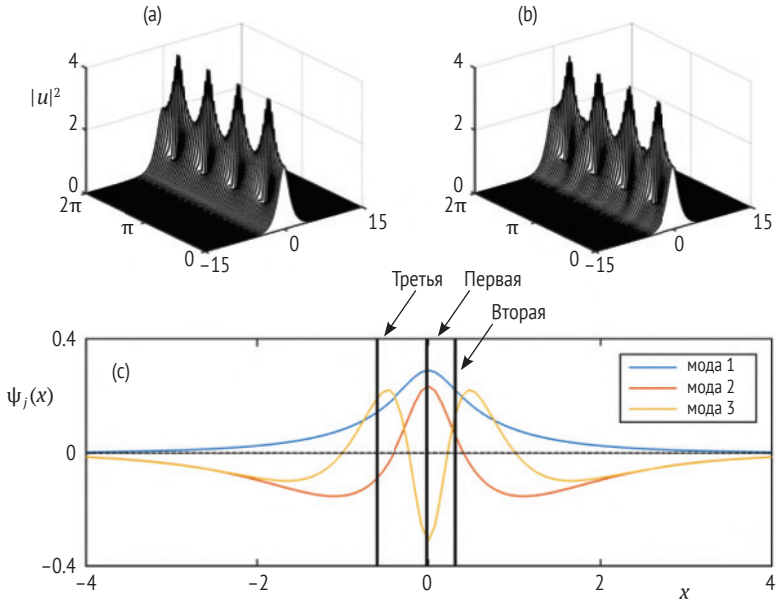

Рис. 12.16 ❖ Сравнение (a) полной смоделированной динамики и (b) упрощенной модели ранга $r = 3$ с тремя точками интерполяции, найденными DEIM. (c) Изображение всех трех POD-мод, использованных для моделирования, а также первой, второй и третьей точек интерполяции. Эти три точки позволяют точно реконструировать динамику эволюции полной системы, описываемой УрЧП

Алгоритм QDEIM

Жадный алгоритм DEIM эффективно находит точки интерполяции, но существуют и другие, не менее эффективные методы. В недавно предложенном алгоритме QDEIM [159] для этой цели применяется QR-разложение. Показано, что это робастная математическая архитектура размещения датчиков во многих приложениях [366]. Более общее обсуждение см. в разделе 3.8. В алгоритме QDEIM датчики размещаются в точках поворота QR-разложения. Следующий код может заменить алгоритм DEIM при построении матрицы интерполяции **P**.

Листинг 12.19 ❖ Нахождение точек интерполяции с помощью QR-разложения

```
[Q,R,pivot]=qr(NL.');
P=pivot(:,1:r);
```

Применение этой матрицы дает те же самые точки интерполяции, как показано на рис. 12.16. В общем случае существуют оценки, показывающие, что QDEIM может уменьшить ошибку по сравнению с обычным DEIM [159]. Благодаря простоте использования алгоритма QR-разложение может стать интересным методом разреженной интерполяции.

12.7. Машинное обучение ROM

На волне успеха методов машинного обучения различные базисы POD для параметрических систем были собраны в библиотеку POD-мод Ψ_L, содержащую все низкоранговые подпространства, наблюдаемые в динамических системах. При этом используется тот факт, что POD является теоретически обоснованным способом построения r-мерного подпространства Ψ_r, характеризующего динамику, тогда как разреженная выборка дополняет метод POD способом вычисления проблематичных нелинейных членов с помощью матрицы $\mathbf{P}$ проекции на p-мерное подпространство. Таким образом, для аппроксимации нелинейных членов в ROM можно выбрать небольшое число точек. На рис. 12.17 показана процедура создания библиотеки, при которой производится выборка из динамического режима для построения подходящего POD-базиса Ψ.

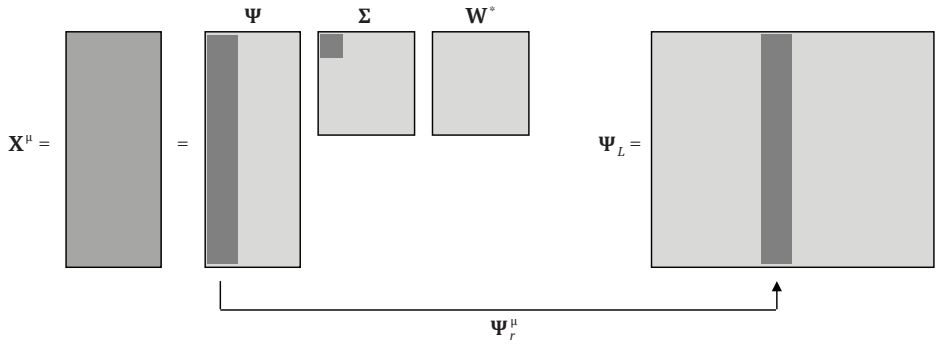

Рис. 12.17 ❖ Построение библиотеки по результатам численного моделирования уравнений (11.1). Моделируется параметрическая система с разными значениями бифуркационного параметра μ. Для каждого режима вычисляются низкоразмерные POD-моды Ψ_r методом сингулярного разложения. Различные упрощенные подпространства ранга r хранятся в матрице Ψ_L, представляющей библиотеку мод (*взято из работы Kutz et al. [319]*)

Описываемый здесь метод, опираясь на эти методы, строит низкоразмерные библиотеки, ассоциированные с полной нелинейной динамикой систе-

мы, а также с конкретными нелинейностями. Как будет показано ниже, точки интерполяции можно использовать совместно с разреженным представлением и сжатыми измерениями, чтобы (i) идентифицировать динамические режимы, (ii) реконструировать полное состояние системы и (iii) реализовать эффективное упрощение нелинейной модели и предсказание будущего состояния методом POD–Галеркина.

Идея построения библиотеки низкоразмерных *признаков* по данным давно укоренилась в сообществе специалистов по информатике. В области создания моделей пониженного порядка она недавно стала основной стратегией расчета параметрических систем. Опубликовано немало работ по библиотекам ROM-моделей [80, 98, 462, 10, 134, 422, 421, 420], которые можно выбирать и (или) интерполировать путем измерений и классификации. С другой стороны, в кластерных моделях пониженного порядка используется кластеризация методом k средних для построения модели марковских переходов между динамическими состояниями [278]. Эти недавние работы похожи на излагаемые здесь идеи. Но нас прежде всего интересует, как подходящим образом выбранную матрицу $\mathbf{P}$ можно использовать со всеми библиотеками для выбора POD-мод и реконструкции. Можно также построить два набора библиотек: один для полной динамики, другой для нелинейности, чтобы добиться вычислительной эффективности с помощью стратегии DEIM [462]. Но еще до разработки этих формальных методов, основанных на машинном обучении, пришло понимание, что области параметров можно разбить на подобласти и в каждой из них вычислить локальное POD-разложение и ROM. В работе Patera et al. [171] используется разбиение на основе двоичного дерева, а в работе Amsallem et al. [9] – диаграмма Вороного. Такие методы тесно связаны с работой Du and Gunzburger [160], в которой снимки данных разбивались на подмножества и вычислялось несколько упрощенных базисов. Затем эти несколько базисов объединялись в один, т. е. библиотека как таковая не создавалась. Обзор стратегий разбиения области параметров см. в работе [11].

Выбор POD-моды

Существует несколько методов выбора подходящих элементов библиотеки POD, включая алгоритм кластеризации k средних [10, 134, 422, 421, 420], но можно вместо этого воспользоваться для характеризации динамической системы разреженной выборкой и разреженным представлением для классификации (SRC), описанными в главе 3 [80, 98, 462]. Наша цель – использовать ограниченное число датчиков (точек интерполяции) для классификации динамического режима системы, пользуясь потенциальными элементами библиотеки POD, характеризуемыми параметром β. После успешной классификации можно предпринять стандартную ℓ_2-реконструкцию полного состояния с помощью выбранного подмножества POD-мод, а для предсказания будущих состояний воспользоваться методом POD–Галеркина.

В общем случае мы получим разреженный вектор измерений $\tilde{\mathbf{u}}$ (12.1). Вектор полного состояния $\mathbf{u}$ можно аппроксимировать с помощью библиотечных POD-мод ($\mathbf{u} = \Psi_L \mathbf{a}$), поэтому

$$\tilde{\mathbf{u}} = \mathbf{P}\Psi_L\mathbf{a}, \tag{12.18}$$

где Ψ_L – низкоранговая матрица, столбцами которой являются базисные векторы POD, конкатенированные по всем режимам β, а **c** – вектор коэффициентов, определяющий проекцию **u** на эти POD-моды. Если $\mathbf{P}\Psi_L$ обладает свойством ограниченной изометрии, а **u** достаточно разрежен в Ψ_L, то из сильно недоопределенной системы (12.18) можно найти самый разреженный вектор **a**. Математически это эквивалентно задаче ℓ_0-оптимизации, являющейся NP-трудной. Однако при некоторых условиях разреженное решение уравнения (12.18) все же можно получить (см. главу 3), произведя вместо этого минимизацию по норме l_1, так что

$$\mathbf{c} = \underset{\mathbf{a}'}{\operatorname{argmin}} ||\mathbf{a}'||_1 \text{ при условии } \tilde{\mathbf{u}} = \mathbf{P}\Psi_L\mathbf{a}. \tag{12.19}$$

Эту задачу уже можно решить стандартными методами выпуклой оптимизации. Таким образом, норма ℓ_1 является заменителем разреженности. Заметим, что мы используем разреженность только для классификации, но не для реконструкции. На рис. 12.18 показана стратегия разреженной выборки и типичное разреженное решение **a**.

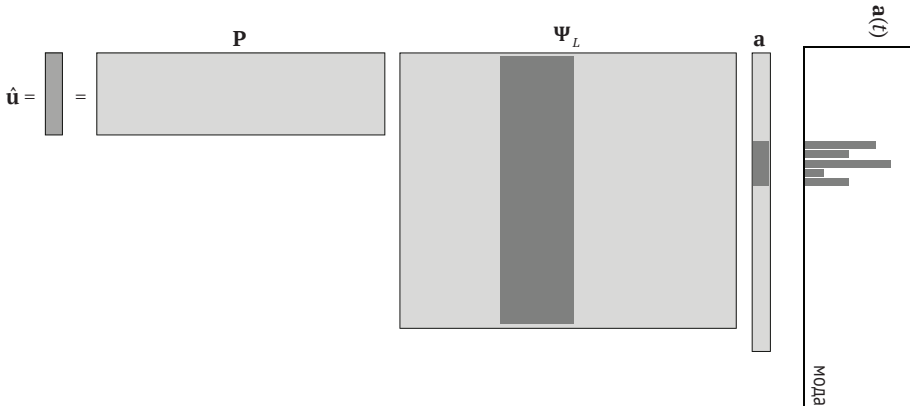

Рис. 12.18 ❖ Применение алгоритма разреженного представления для классификации (SRC) к выбору библиотечной моды; детали см. в разделе 3.6. В этой математической архитектуре производится разреженное измерение системы (11.1) и решается сильно недоопределенная система уравнений $\mathbf{P}\Psi_L\mathbf{a} = \tilde{\mathbf{u}}$ со штрафом по норме ℓ_1, так чтобы доставить минимум величине $||\mathbf{a}||_1$. На рисунке показан выбор μ-й POD-моды. На столбчатой диаграмме справа изображены ненулевые элементы вектора **a**, соответствующие библиотечным элементам Ψ_r. Заметим, что выборочная матрица **P**, порождающая разреженную выборку $\tilde{\mathbf{u}} = \mathbf{P}\mathbf{u}$, играет важнейшую роль для успеха классификации правильных библиотечных элементов Ψ_r и соответствующей реконструкции (*взято из работы Kutz et al. [319]*)

Пример: обтекание цилиндра

Для демонстрации алгоритма разреженной классификации и реконструкции рассмотрим каноническую задачу обтекания цилиндра. Она хорошо изучена и уже становилась предметом исследований по разреженным пространственным измерениям [80, 98, 462, 281, 374, 89, 540]. Известно, что для малых и умеренных чисел Рейнольдса размерность пространственной динамики низкая, и подходы на основе POD успешно применялись для ее расчета. Число Рейнольдса *Re* играет роль бифуркационного параметра β в уравнении (11.1), т. е. это параметрическая динамическая система.

Мы будем рассматривать данные, полученные в результате численного моделирования уравнений Навье–Стокса несжимаемой жидкости:

$$\frac{\partial u}{\partial t} + u \cdot \nabla u + \nabla p - \frac{1}{\mathrm{Re}} \nabla^2 u = 0; \tag{12.20a}$$

$$\nabla \cdot u = 0, \tag{12.20b}$$

где $u(x, y, t) \in \mathbb{R}^2$ представляет двумерное поле скоростей, а $p(x, y, t) \in \mathbb{R}^2$ – соответствующее поле давлений. Граничные условия таковы: (i) постоянное течение $u = (1, 0)^T$ при $x = -15$, т. е. на входе в канал, (ii) постоянное давление $p = 0$ при $x = 25$, т. е. на выходе из канала, (iii) граничные условия Неймана, т. е. $\partial u / \partial \mathbf{n} = 0$ на границе канала и цилиндра (с центром в точке $(x, y) = (0, 0)$ единичного радиуса).

Для каждого интересующего нас значения параметра *Re* выполняется сингулярное разложение матрицы данных с целью нахождения POD-мод. Хорошо известно, что для сравнительно малых значений числа Рейнольдса наблюдается быстрое убывание сингулярных значений, так что для характеристики динамики нужно всего несколько POD-мод. На рис. 12.19 показаны три доминирующие POD-моды для чисел Рейнольдса $Re = 40, 150, 300, 1000$. Заметим, что в качестве порога отсечения взято 99 % всей энергии (дисперсии), приходящейся на выбранные моды, что дает 1, 3, 3 и 9 POD-мод для представления динамики в изучаемых режимах. Если взять порог 99.9 %, то для учета изменчивости понадобится больше мод.

Для классификации числа Рейнольдса следует решить задачу оптимизации (12.19) и получить разреженный вектор коэффициентов **a**. Отметим, что каждый элемент **a** соответствует энергии одной POD-моды из нашей библиотеки. Для простоты мы выбрали в качестве точек интерполяции в матрице **P** некоторые минимумы и максимумы POD-мод. Для классификации суммируются абсолютные величины коэффициентов, соответствующих каждому числу Рейнольдса. Чтобы учесть тот факт, что чем больше число Рейнольдса, тем больше коэффициентов ему соответствует (для дисперсии 99.9 % числу $Re = 1000$ может соответствовать 16 мод, а числу 40 – всего одна), мы делим на квадратный корень из числа POD-мод в векторе **a**, соответствующих каждому числу Рейнольдса. Результатом классификации является режим, для которого получившаяся таким образом величина максимальна.

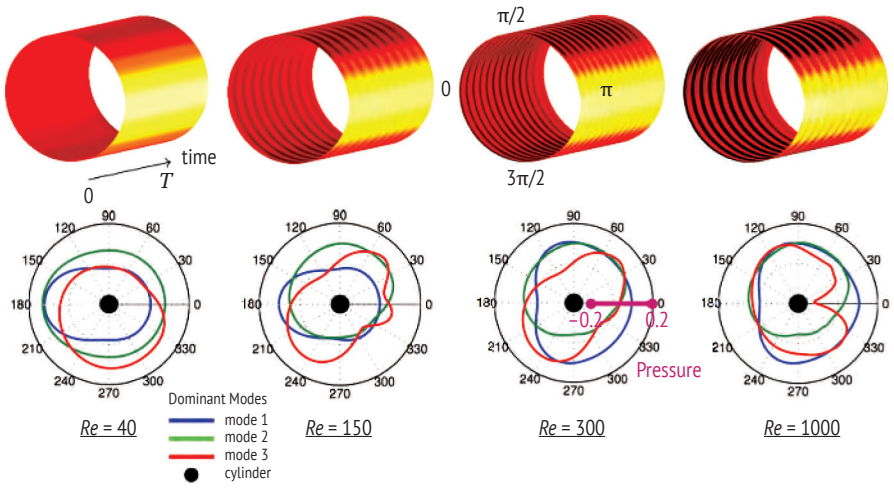

Рис. 12.19 ❖ Временная динамика поля давлений (сверху) для обтекающего цилиндр потока с числом Рейнольдса *Re* = 40, 150, 300 и 1000. Собранные снимки состояния показывают, что в динамике доминируют низкоразмерные структуры. Для каждого числа Рейнольдса показаны три доминирующие POD-моды давления в полярных координатах. Масштаб давления показан сиреневым цветом (справа внизу) (*взято из работы Kutz et al. [319]*)

Хотя верность классификации высока, многие ложные результаты связаны с ошибочным выбором числа Рейнольдса соседнего течения, т. е. вместо 1000 часто принимается 800. Это объясняется тем, что эти два числа Рейнольдса очень похожи, и алгоритм с трудом различает структуры их мод. На рис. 12.20 показаны схема разреженной конфигурации датчиков и реконструкция поля давлений при $Re = 1000$ с 15 датчиками. Качество классификации и реконструкции можно улучшить, воспользовавшись другими методами построения матрицы датчиков **P** [80, 98, 462, 281, 374, 89, 540]. Но как бы то ни было, этот пример демонстрирует использование методов, поощряющих разреженность, для выбора POD-мод (оптимизация по норме ℓ_1) и последующей реконструкции (проекция по норме ℓ_2).

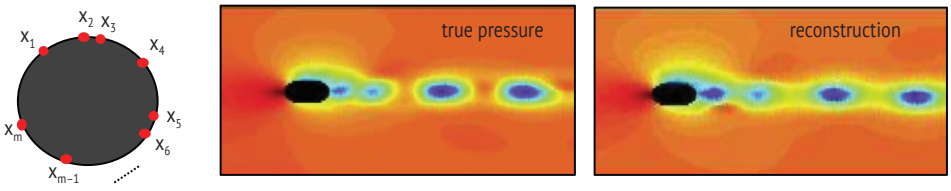

Рис. 12.20 ❖ Разреженное расположение *m* датчиков (слева) для классификации и реконструкции поля обтекания. Для выбора положений датчиков (точек интерполяции) можно использовать различные алгоритмы [80, 98, 462, 281, 374, 89, 540]. Матрица датчиков **P** определяет качество классификации и реконструкции для выбранного алгоритма (*взято из работы Kutz et al. [319]*)

Наконец, чтобы визуализировать весь процесс разреженного измерения и реконструкции более детально, на рис. 12.21 показана реконструкция числа Рейнольдса для нестационарного поля обтекания, а также реконструкции поля давлений и поля обтекания в избранные моменты времени. Отметим, что схема SRC в сочетании с библиотекой машинного обучения с учителем – эффективный метод характеризации течения с помощью разреженных измерений. Для больших чисел Рейнольдса точно классифицировать поле обтекания при таком малом количестве датчиков становится гораздо сложнее. Однако это еще не значит, что мы не можем реконструировать поле давлений, поскольку многие библиотечные элементы при больших числах Рейнольдса весьма схожи.

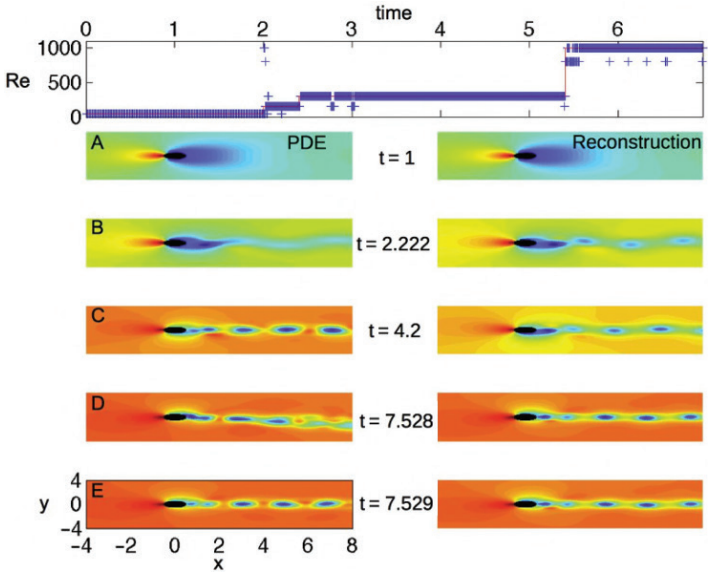

Рис. 12.21 ❖ Идентификация числа Рейнольдса и реконструкция поля давления для нестационарного течения с помощью разреженных измерений. На верхнем рисунке показано истинное число Рейнольдса, использованное в процессе полного моделирования (сплошная линия), а также моменты сжатых измерений (крестики). На рисунках A–D показана реконструкция поля давлений в четыре момента времени; видно, что, по крайней мере, качественно реконструкция верна (слева – смоделированное поле давлений, а справа – его реконструкция). Отметим, что чем больше число Рейнольдса, тем труднее классификация (*взято из работы Bright et al. [80]*)

Рекомендуемая литература

Учебники

(1) Certified reduced basis methods for parametrized partial differential equations, by J. Hesthaven, G. Rozza and B. Stamm, 2015 [244].

(2) **Reduced basis methods for partial differential equations: An introduction**, by A. Quarteroni, A. Manzoni and N. Federico, 2015 [442].
(3) **Model reduction and approximation: Theory and algorithms**, by P. Benner, A. Cohen, M. Ohlberger and K. Willcox, 2017 [54].

Статьи и обзоры

(1) **A survey of model reduction methods for parametric systems**, by P. Benner, S. Gugercin and K. Willcox, *SIAM Review*, 2015 [53].
(2) **Model reduction using proper orthogonal decomposition**, by S. Volkwein, *Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz*, 2011 [542].
(3) **Nonlinear model reduction for dynamical systems using sparse sensor locations from learned libraries**, by S. Sargsyan, S. L. Brunton and J. N. Kutz, *Physical Review E*, 2015 [462].
(4) **An online method for interpolating linear parametric reduced-order models**, by D. Amsallem and C. Farhat, *SIAM Journal of Scientific Computing*, 2011 [10].

Глоссарий

DMD-мода (также *динамическая мода*). Собственный вектор DMD-оператора **A**, дающего наилучшую аппроксимацию (см. *разложение по динамическим модам*). Эти моды пространственно когерентны и колеблются во времени с фиксированной частотой и скоростью возрастания или убывания.

Мах-пулинг. Стратегия понижающей передискретизации данных, позволяющая уменьшить размерность представления входа (изображения, матрицы выходов скрытого слоя и т. д.), а стало быть, делать предположения о признаках, содержащихся в передискретизованных подобластях.

Алгоритм реализации собственной системы (ERA). Метод идентификации систем, порождающий сбалансированные модели входа-выхода системы по данным импульсной характеристики. Показано, что при определенных условиях ERA строит модели, эквивалентные сбалансированному собственному ортогональному разложению и разложению по динамическим модам.

Амплитуда DMD. Амплитуда заданной DMD-моды, присутствующей в данных. Эти амплитуды могут интерпретироваться как значимость данной моды по аналогии с энергетическим спектром в БПФ.

Байесовский информационный критерий (BIC). Оценка относительного качества статистических моделей для имеющегося набора данных. Если задано множество моделей, то BIC оценивает качество каждой модели относительно остальных. Таким образом, BIC предлагает способ выбора модели.

Быстрое преобразование Фурье (БПФ). Алгоритм вычисления дискретного преобразования Фурье (ДПФ) за $O(n \log(n))$ операций. БПФ совершило революцию в современных вычислениях: обработке сигналов, сжатии и передаче данных.

Вейвлет. Функция или семейство функций, используемое для обобщения преобразования Фурье с целью аппроксимации более сложных многомасштабных сигналов.

Выпуклая оптимизация. Свод алгоритмов минимизации выпуклых функций на выпуклых множествах.

Гильбертово пространство. Обобщенное векторное пространство, на котором определено скалярное произведение. В этой книге под гильбертовым пространством, как правило, понимается бесконечномерное пространство функций. Такие пространства одновременно являются полными метрическими пространствами, и этой математической структуры достаточно для применения к функциям средств математического анализа.

Главная компонента. Пространственно коррелированная мода, присутствующая в наборе данных. Часто вычисляется с помощью сингулярного разложения данных после вычитания среднего.

Глубокое обучение. Класс алгоритмов машинного обучения, в которых обычно используются глубокие СНС для выделения и преобразования признаков. В глубоком обучении может применяться обучение с учителем (например, для классификации) и (или) обучение без учителя (например, анализ паттернов). Результатом является обучение нескольким уровням представления, соответствующим различным уровням абстракции; эти уровни образуют иерархию концепций.

Грамиан. Грамиан управляемости (наблюдаемости) определяет степень управляемости (наблюдаемости) состояния посредством управляющего воздействия (оценивания). Грамиан вводит скалярное произведение на пространстве состояний.

Динамическая система. Математическая модель динамической эволюции системы. Обычно динамическая система описывается в терминах обыкновенных дифференциальных уравнений или пространства состояний. Уравнения могут быть линейными или нелинейными, могут учитывать управляющие воздействия и представлять выходные сигналы в виде измерений состояния датчиками.

Закон Мура. Наблюдение, согласно которому плотность размещения транзисторов, а стало быть – быстродействие процессора, экспоненциально возрастает со временем. Закон Мура часто используется для прогнозирования вычислительной мощности и соответственно масштаба разрешимых задач.

Запаздывающие координаты. Дополненный набор координат, при построении которого рассматривается измерение в текущий момент времени, а также в несколько прошлых моментов, разделенных фиксированными промежутками. Запаздывающие координаты часто бывают полезны для реконструкции аттрактора в системах, где измерений недостаточно, как в теореме Такенса о погружении.

Идентификация системы. Процесс построения модели системы по данным измерений, возможно, произведенным после возмущения системы.

Информационный критерий Акаике (AIC). Оценка относительного качества статистических моделей для имеющегося набора данных. Если задано множество моделей, то AIC оценивает качество каждой модели относительно остальных. Таким образом, AIC предлагает способ выбора модели.

Классификация. Общий процесс, позволяющий выделить, разделить по категориям и понять идеи и объекты. Классификация – типичная задача для алгоритмов машинного обучения.

Кластеризация. Задача группировки множества объектов таким образом, чтобы объекты, попавшие в одну группу (кластер), были в каком-то смысле больше похожи друг на друга, чем на объекты из других групп. Это основная цель разведывательной добычи данных, часто применяется в статистическом анализе данных.

Когерентная структура. Пространственный паттерн, коррелированный с данными в системе.

Количественная оценка неопределенности. Теоретически обоснованная характеристика и метод управления неопределенности в технических системах. Часто включает применение к динамическим системам мощных инструментов теории вероятностей и математической статистики.

Линейная система. Система, в которой суперпозиция любых двух входов дает суперпозицию соответствующих выходов. Иными словами, удвоение входа приводит к удвоению выхода. Линейные стационарные динамические системы характеризуются линейными операторами, представляемыми в виде матриц.

Линейно-квадратичный регулятор (ЛКР). Оптимальный регулятор с пропорциональной обратной связью, который находит компромисс между двумя целями: управление состоянием при умеренном расходе энергии. Матрица пропорциональных коэффициентов усиления ищется путем решения алгебраического уравнения Риккати.

Марковская модель. Вероятностная динамическая система, в которой вектор состояния содержит вероятность того, что система окажется в некотором состоя-

нии; таким образом, сумма элементов этого вектора должна быть равна 1. Динамика определяется матрицей переходов, устроенной так, что сумма элементов в каждой строке равна 1.

Марковские параметры. Измерения выхода динамической системы в ответ на импульсный входной сигнал.

Матрица данных. Матрица, каждый столбец которой является снимком состояния системы в некоторый момент времени. Снимки могут быть *последовательными* во времени или браться из множества начальных условий либо экспериментов.

Машинное обучение. Набор статистических инструментов и алгоритмов, предназначенных для выделения доминирующих паттернов из данных. Обучение может быть с учителем и без учителя, а его целями являются, в частности, кластеризация, классификация и предсказание.

Метод главных компонент (PCA). Разложение матрицы данных в иерархию векторов главных компонент, упорядоченных по убыванию степени корреляции с данными. PCA реализуется с помощью сингулярного разложения данных после вычитания среднего. В этом случае каждое сингулярное значение представляет дисперсию соответствующей главной компоненты (сингулярного вектора) данных.

Метод полных наименьших квадратов. Алгоритм регрессии методом наименьших квадратов, в котором минимизируется ошибка как на входах, так и на выходах. Геометрически это соответствует поиску прямой, которая минимизирует сумму квадратов расстояний до всех точек, а не просто сумму квадратов расстояний до вертикальных прямых, проходящих через точки.

Многомасштабность. Наличие нескольких пространственных и (или) временных масштабов. Во многих системах, в частности при изучении турбулентности, встречаются пространственные и временные масштабы, различающиеся по величине на несколько порядков.

Модель пониженного порядка (ROM). Модель системы высокой размерности, выраженная в терминах состояния намного меньшей размерности. Обычно модель пониженного порядка призвана отыскать баланс между точностью и вычислительной сложностью модели.

Моментальный снимок. Одно многомерное измерение системы в некоторый момент времени. Несколько снимков, сделанных в последовательные моменты времени, можно организовать в виде столбцов матрицы данных.

Наблюдаемая функция. Функция, измеряющая какое-то свойство состояния системы. Наблюдаемые функции обычно являются элементами некоторого гильбертова пространства.

Наблюдаемость. Система называется наблюдаемой, если любое ее состояние можно оценить, зная историю показания имеющихся датчиков. Степень наблюдаемости определяется грамианом наблюдаемости.

Недоопределенная система. Система вида $\mathbf{Ax} = \mathbf{b}$, в которой уравнений меньше, чем неизвестных. В общем случае у такой системы бесконечно много решений $\mathbf{x}$, если только вектор $\mathbf{b}$ не принадлежит пространству столбцов матрицы $\mathbf{A}$.

Некогерентные измерения. Измерения, для которых мало скалярное произведение с базисными векторами разреживающего преобразования. Например, измерения одного пикселя (т. е. пространственные дельта-функции) некогерентны относительно базиса пространственного преобразования Фурье, поскольку возбуждают сразу все частоты и не соотносятся ни с какой конкретной частотой.

Низкий ранг. Свойство матрицы, означающее, что количество линейно независимых строк и столбцов мало по сравнению с размером матрицы. Для больших матриц данных ищутся аппроксимации низкого ранга.

Обратное распространение. Метод, применяемый для вычисления градиентов, необходим при обучении нейронных сетей. Основанное на правиле дифференцирования сложной функции, обратное распространение задействует композиционную структуру нейронной сети и ставит задачу обновления весов сети как задачу оптимизации. Часто используется для обучения глубоких нейронных сетей.

Оператор Купмана. Бесконечномерный линейный оператор, который экстраполирует функции измерения бесконечномерного гильбертова пространства на динамическую систему.

Оператор Перрона–Фробениуса. Оператор, сопряженный оператору Купмана, т. е. бесконечномерный оператор, который экстраполирует изменение функций плотности вероятности в динамической системе.

Оптимизация. Набор алгоритмов для нахождения «лучших из возможных» значений некоторой целевой функции. Математически задачей оптимизации является максимизация или минимизация вещественной функции путем систематического выбора входных значений из допустимого множества и вычисления значений функции. Различные обобщения теории и методов оптимизации составляют обширный раздел прикладной математики.

Парето-фронт. Распределение ресурсов, которое невозможно изменить так, что каждому индивидууму станет лучше (или все критерии улучшатся) и никому не станет хуже (ни один критерий не ухудшится).

Перекрестная проверка. Метод проверки модели на предмет того, насколько хорошо результаты статистического анализа обобщаются на независимый (зарезервированный) набор данных.

Переопределенная система. Система вида $\mathbf{Ax} = \mathbf{b}$, в которой уравнений больше, чем неизвестных. Обычно у переопределенной системы нет точного решения $\mathbf{x}$, если только вектор $\mathbf{b}$ не принадлежит пространству столбцов $\mathbf{A}$.

Преобразование Фурье. Изменение базиса с целью представить функцию бесконечным рядом, состоящим из синусов и косинусов.

Проекция Галеркина. Процесс, с помощью которого дифференциальные уравнения в частных производных сводятся к обыкновенным дифференциальным уравнениям в терминах динамики коэффициентов при ортогональных базисных модах, используемых для аппроксимации решения.

Пространство состояний. Множество всех возможных состояний системы. Часто пространством состояний является векторное пространство, например $\mathbb{R}^n$, хотя это может быть и гладкое многообразие M.

Псевдообращение. Псевдообращение – это обобщение операции обращения на неквадратные матрицы. Часто оно применяется для решения системы уравнений методом наименьших квадратов. Распространенным методом вычисления псевдообратной матрицы является сингулярное разложение: если имеется сингулярное разложение $\mathbf{X} = \mathbf{U\Sigma V}^*$, то псевдообратная матрица равна $\mathbf{X}^\dagger = \mathbf{V\Sigma}^{-1}\mathbf{U}^*$.

Разложение по динамическим модам (DMD). Основное спектральное разложение линейного оператора наилучшей аппроксимации $\mathbf{A} = \mathbf{X}'\mathbf{X}^\dagger$, которое экстраполирует матрицу данных $\mathbf{X}$ в будущую матрицу $\mathbf{X}'$. Собственные векторы $\mathbf{A}$ – это DMD-моды, а соответствующие им собственные значения определяют временную динамику этих мод.

Разреженная идентификация нелинейной динамики (SINDy). Метод, применяемый для одновременной идентификации нелинейной структуры и параметров динамической системы по данным. Для определения SINDy-моделей используются различные приемы разреженной оптимизации.

Разреженность. Вектор называется разреженным, если большинство его элементов в точности или приближенно равны нулю. Разреженность связана с наблюдением, согласно которому данные, как правило, представляются разреженными векторами в подходящем базисе, например базисе Фурье или POD.

Регрессия методом наименьших квадратов. Метод регрессии, при котором прямая или вектор, дающий наилучшую аппроксимацию, находится путем минимизации суммы квадратов ошибки между моделью и данными.

Регрессия. Статистическая модель, представляющая выходную переменную в терминах индикаторных переменных. Регрессия методом наименьших квадратов – это линейная регрессия, которая находит прямую, наилучшим образом аппроксимирующую данные; при обобщении на большее число измерений и полилинейную регрессию мы приходим к методу главных компонент. Нелинейная регрессия, динамическая регрессия и функциональная, или семантическая, регрессия используются для идентификации системы, упрощения модели и в машинном обучении.

Рекуррентная нейронная сеть (RNC). Класс нейронных сетей, в которых связи между блоками образуют направленную последовательность. Это позволяет обрабатывать серии событий во времени.

Робастная статистика. Методы построения хороших статистических оценок для данных, выбираемых из широкого спектра распределений вероятностей, особенно когда распределения не являются нормальными и имеются выбросы, негативно сказывающиеся на точности предсказаний.

Робастное управление. Метод управления, штрафующий за *худший сценарий* и тем самым поощряющий проектирование регуляторов, устойчивых к неопределенностям, возмущениям и немоделируемой динамике.

Сбалансированная модель входа-выхода. Модель, выраженная в системе координат, в которой состояния упорядочены одновременно по управляемости и наблюдаемости. Грамианы управляемости и наблюдаемости в такой системе равны и являются диагональными матрицами.

Сверточная нейронная сеть (CNC). Класс глубоких нейронных сетей прямого распространения, особенно удобных для анализа естественных изображений. Свертка обычно производится с помощью пространственного фильтра, который синтезирует локальную (в малой окрестности) пространственную информацию.

Свойство ограниченной изометрии (RIP). Свойство, благодаря которому матрица ведет себя как унитарное, или изометрическое, отображение на разреженных векторах. Иными словами, расстояние между любыми двумя разреженными векторами сохраняется, если эти векторы подвергаются отображению со свойством ограниченной изометрии.

Сжатие. Процесс уменьшения размера вектора или массива высокой размерности путем аппроксимации его разреженным вектором в преобразованном базисе. Например, в алгоритмах сжатия MP3 и JPG используется базис Фурье или вейвлетный базис для сжатия звуковых сигналов и изображений.

Сжатое измерение. Процесс реконструкции многомерного векторного сигнала по случайной выборке данных с использованием того факта, что многомерный

сигнал является разреженным в известном преобразованном базисе, например в базисе Фурье.

Сингулярное разложение (SVD). Пусть дана матрица $\mathbf{X} \in \mathbb{C}^{n \times m}$, тогда сингулярное разложение имеет вид $\mathbf{X} = \mathbf{U\Sigma V}^*$, где $\mathbf{U} \in \mathbb{C}^{n \times n}$, $\mathbf{\Sigma} \in \mathbb{C}^{n \times m}$ и $\mathbf{V} \in \mathbb{C}^{m \times m}$. Матрицы $\mathbf{U}$ и $\mathbf{V}$ унитарные, т. е. $\mathbf{UU}^* = \mathbf{U}^*\mathbf{U} = \mathbf{I}$ и $\mathbf{VV}^* = \mathbf{V}^*\mathbf{V} = \mathbf{I}$. Матрица $\mathbf{\Sigma}$ диагональная, элементы на диагонали – сингулярные значения, расположенные в порядке убывания. В результате порождается иерархическое разложение матрицы в сумму матриц ранга 1, определяемых произведением вектора-столбца (левого сингулярного вектора) на вектор-строку (комплексно сопряженный и транспонированный правый сингулярный вектор). Эти матрицы ранга 1 упорядочены по сингулярным значениям, так что первые r матриц ранга 1 образуют *наилучшую* аппроксимацию ранга r исходной матрицы в смысле среднеквадратической ошибки.

Скрытая марковская модель (CMM). Марковская модель, в которой имеется скрытое состояние, допускающее наблюдение только с помощью набора измерений, называемых эмиссиями.

Собственная функция Купмана. Собственная функция оператора Купмана. Эти собственные функции соответствуют измерениям пространства состояний динамической системы, образующим внутренние координаты. Иными словами, эти внутренние измерения линейно зависят от времени, хотя сама система нелинейна.

Собственное значение DMD. Собственное значение DMD-оператора $\mathbf{A}$, дающего наилучшую аппроксимацию (см. *разложение по динамическим модам*).

Собственное ортогональное разложение (POD). Разложение данных динамической системы в иерархию ортогональных мод, часто с использованием сингулярного разложения. Если данными являются результаты измерения скорости, например несжимаемой жидкости, то собственное ортогональное разложение упорядочивает моды по содержащемуся в них количеству энергии.

Сопряженное отображение. Для конечномерного линейного отображения (т. е. матрицы $\mathbf{A}$) сопряженным называется отображение $\mathbf{A}^*$, матрица которого является комплексно сопряженной и транспонированной $\mathbf{A}$. Сопряженным к бесконечномерному линейному оператору A называется такой оператор A^*, что для любых f и g $\langle Af, g\rangle = \langle f, A^*g\rangle$, где $\langle \cdot, \cdot \rangle$ – скалярное произведение.

Спектр мощности. Множество квадратов амплитуд коэффициентов преобразования Фурье сигнала. Мощность соответствует доле каждой частоты, необходимой для реконструкции сигнала.

Спектрограмма. Преобразование Фурье, вычисленное в скользящем окне, в результате чего получается график зависимости частоты от времени, показывающий, какие частоты активны в каждый момент времени. Спектрограмма полезна для характеризации непериодических сигналов, когда частотный состав меняется со временем, как в музыке.

Стохастический градиентный спуск. Метод, позволяющий аппроксимировать градиент одной точкой данных, а не всеми имеющимися данными. На каждом шаге градиентного спуска одна случайно выбранная точка используется для вычисления направления градиента.

Теория управления. Дисциплина, изучающая, как с помощью измерений и управляющих воздействий модифицировать динамическую систему, чтобы она отвечала желаемым техническим спецификациям.

Унитарная матрица. Матрица, для которой комплексно сопряженная и транспонированная матрица совпадает с обратной. Все собственные значения

унитарной матрицы располагаются на единичной окружности в комплексной плоскости, а преобразование координат, определяемое унитарной матрицей, сохраняет евклидово расстояние между векторами.

Управление с замкнутым контуром. Архитектура управления, в которой управляющее воздействие определяется с учетом информации о выходе системы, получаемой от датчиков.

Управление с обратной связью. Управление с замкнутым контуром, при котором датчики измеряют эффект управляющего воздействия, а затем информация с них подается обратно на приводы. Обратная связь необходима для робастного управления, чтобы быстро компенсировать неопределенность и неустойчивость модели.

Управление с прогнозирующими моделями (MPC). Вид оптимального управления, при котором на основе модели оптимизируется политика управления на конечном временном горизонте. Модели, применяемые в MPC, как правило, линейны, их можно определить эмпирически путем идентификации системы.

Управляемость. Система называется управляемой, если ее можно привести в любое состояние с помощью управляющих воздействий. Степень управляемости определяется грамианом управляемости.

Упреждающее управление. Управление, при котором датчики измеряют возмущения на входе в системе, и данная информация подается на приводы, чтобы заблаговременно эти возмущения компенсировать.

Фильтр Калмана. Метод оценивания, позволяющий реконструировать полное состояние динамической системы по временному ряду измерений выходов датчиков и входов приводов. Фильтр Калмана сам по себе является динамической системой, построенной так, чтобы имела место устойчивая сходимость к истинному состоянию наблюдаемой системы. Фильтр Калмана является оптимальным для линейных систем с гауссовым процессом и известной амплитудой шума измерений.

Эмиссия. Функции измерения в скрытой марковской модели.

Список литературы

[1] Ralph Abraham, Jerrold E. Marsden, and Tudor Ratiu. *Manifolds, Tensor Analysis, and Applications*, volume 75 of *Applied Mathematical Sciences*. Springer-Verlag, 1988.

[2] Ralph Abraham and Jerrold E. Marsden. *Foundations of Mechanics*, volume 36. Benjamin/Cummings Publishing Company Reading, Massachusetts, 1978.

[3] Mradul Agrawal, Sandeep Vidyashankar, and Ke Huang. On-chip implementation of ECoG signal data decoding in brain-computer interface. In *Mixed-Signal Testing Workshop (IMSTW), 2016 IEEE 21st International*, pages 1–6. IEEE, 2016.

[4] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[5] Hyo-Sung Ahn, YangQuan Chen, and Kevin L. Moore. Iterative learning control: Brief survey and categorization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1099–1121, 2007.

[6] Hirotugu Akaike. Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mathematics*, 21(1):243–247, 1969.

[7] Hirotugu Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.

[8] W. O. Amrein and Anne-Marie Berthier. On support properties of Lp-functions and their Fourier transforms. *Journal of Functional Analysis*, 24(3):258–267, 1977.

[9] David Amsallem, Julien Cortial, and Charbel Farhat. On-demand cfd-based aeroelastic predictions using a database of reduced-order bases and models. In *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, page 800, 2009.

[10] David Amsallem and Charbel Farhat. An online method for interpolating linear parametric reduced-order models. *SIAM Journal on Scientific Computing*, 33(5):2169–2198, 2011.

[11] David Amsallem, Matthew J. Zahr, and Kyle Washabaugh. Fast local reduced basis updates for the efficient reduction of nonlinear systems with hyper-reduction. *Advances in Computational Mathematics*, 41(5):1187–1230, 2015.

[12] Joakim Andén and Stéphane Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.

[13] Edward Anderson, Zhaojun Bai, Christian Bischof, Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, S. Hammerling, Alan McKenney, et al. *LAPACK Users' Guide*, volume 9. *SIAM*, 1999.

[14] Jeffrey L. Anderson. An ensemble adjustment Kalman filter for data assimilation. *Monthly Weather Review*, 129(12):2884–2903, 2001.

[15] Claus A. Andersson and Rasmus Bro. The n-way toolbox for matlab. *Chemometrics and Intelligent Laboratory Systems*, 52(1):1–4, 2000.

[16] Marc Antonini, Michel Barlaud, Pierre Mathieu, and Ingrid Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220, 1992.
[17] Athanasios C. Antoulas. *Approximation of large-scale dynamical systems*. *SIAM*, 2005.
[18] Hassan Arbabi and Igor Mezic´. Ergodic theory, dynamic mode decomposition and computation of spectral properties of the Koopman operator. *SIAM Journal on Applied Dynamical Systems,* 16 (4):2096–2126, 2017.
[19] Kartik B. Ariyur and Miroslav Krstic´. *Real-Time Optimization by Extremum-Seeking Control*. Wiley, Hoboken, New Jersey, 2003.
[20] Travis Askham and J. Nathan Kutz. Variable projection methods for an optimized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems,* 17 (1):380–416, 2018.
[21] Patricia Astrid. Fast reduced order modeling technique for large scale LTV systems. In *American Control Conference, 2004. Proceedings of the 2004*, volume 1, pages 762–767. IEEE, 2004.
[22] Karl Johan Aström and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2010.
[23] M. F. A. Azeez and A. F. Vakakis. Proper orthogonal decomposition (POD) of a class of vibroimpact oscillations. *Journal of Sound and Vibration*, 240(5): 859–889, 2001.
[24] K. Bache and M. Lichman. UCI machine learning repository, 2013.
[25] Brett W. Bader and Tamara G. Kolda. Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing*, 30(1):205–231, December 2007.
[26] S. Bagheri, L. Brandt, and D. S. Henningson. Input-output analysis, model reduction and control of the flat-plate boundary layer. *Journal of Fluid Mechanics*, 620:263–298, 2009.
[27] S. Bagheri, J. Hoepffner, P. J. Schmid, and D. S. Henningson. Input-output analysis and control design applied to a linear model of spatially developing flows. *Applied Mechanics Reviews*, 62(2):020803–1..27, 2009.
[28] Shervin Bagheri. Koopman-mode decomposition of the cylinder wake. *Journal of Fluid Mechanics*, 726:596–623, 2013.
[29] Z. Bai, S. L. Brunton, B. W. Brunton, J. N. Kutz, E. Kaiser, A. Spohn, and B. R. Noack. Data- driven methods in fluid dynamics: Sparse classification from experimental data. In *Invited Chapter for Whither Turbulence and Big Data in the 21st Century*, 2015.
[30] Z. Bai, E. Kaiser, J. L. Proctor, J. N. Kutz, and S. L. Brunton. Dynamic mode decomposition for compressive system identification. *arXiv preprint arXiv:*1710.07737, 2017.
[31] Zhe Bai, Thakshila Wimalajeewa, Zachary Berger, Guannan Wang, Mark Glauser, and Pramod K. Varshney. Low-dimensional approach for reconstruction of airfoil data via compressive sensing. *AIAA Journal*, 53(4):920–933, 2014.
[32] Maciej J. Balajewicz, Earl H. Dowell, and Bernd R. Noack. Low-dimensional modelling of high-Reynolds-number shear flows incorporating constraints from the Navier–Stokes equation. *Journal of Fluid Mechanics*, 729:285–308, 2013.

[33] Madhusudhanan Balasubramanian, Stanislav Zabic, Christopher Bowd, Hilary W. Thompson, Peter Wolenski, S. Sitharama Iyengar, Bijaya B. Karki, and Linda M. Zangwill. A framework for detecting glaucomatous progression in the optic nerve head of an eye using proper orthogonal decomposition. *IEEE Transactions on Information Technology in Biomedicine*, 13(5):781–793, 2009.

[34] B. Bamieh and L. Giarré. Identification of linear parameter varying models. *International Journal of Robust and Nonlinear Control*, 12:841–853, 2002.

[35] Andrzej Banaszuk, Kartik B. Ariyur, Miroslav Krstic´, and Clas A. Jacobson. An adaptive algorithm for control of combustion instability. *Automatica*, 40(11):1965–1972, 2004.

[36] Andrzej Banaszuk, Satish Narayanan, and Youping Zhang. Adaptive control of flow separation in a planar diffuser. *AIAA paper*, 617:2003, 2003.

[37] Andrzej Banaszuk, Youping Zhang, and Clas A. Jacobson. Adaptive control of combustion instability using extremum-seeking. In *American Control Conference, 2000. Proceedings of the 2000*, volume 1, pages 416–422. IEEE, 2000.

[38] S. P. Banks. Infinite-dimensional Carleman linearization, the Lie series and optimal control of non-linear partial differential equations. *International Journal of Systems Science*, 23(5):663– 675, 1992.

[39] R. G. Baraniuk. Compressive sensing. *IEEE Signal Processing Magazine*, 24(4):118–120, 2007.

[40] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010.

[41] Maxime Barrault, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T. Patera. An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[42] J. Basley, L. R. Pastur, N. Delprat, and F. Lusseyran. Space-time aspects of a three-dimensional multi-modulated open cavity flow. *Physics of Fluids (1994-present)*, 25(6):064105, 2013.

[43] J. Basley, L. R. Pastur, F. Lusseyran, T. M. Faure, and N. Delprat. Experimental investigation of global structures in an incompressible cavity flow using time-resolved PIV. *Experiments in Fluids*, 50(4):905–918, 2011.

[44] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317–330, 1983.

[45] J. F. Beaudoin, O. Cadot, J. L. Aider, and J. E. Wesfreid. Bluff-body drag reduction by extremum-seeking control. *Journal of Fluids and Structures*, 22:973–978, 2006.

[46] Jean-Francois Beaudoin, Olivier Cadot, Jean-Luc Aider, and José-Eduardo Wesfreid. Drag reduction of a bluff body using adaptive control methods. *Physics of Fluids*, 18(8):085107, 2006.

[47] R. Becker, R. King, R. Petz, and W. Nitsche. Adaptive closed-loop control on a high-lift configuration using extremum seeking. *AIAA Journal*, 45(6):1382–92, 2007.

[48] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):711–720, 1997.

[49] G. Bellani. Experimental studies of complex flows through image-based techniques. 2011.
[50] Brandt A. Belson, Jonathan H. Tu, and Clarence W. Rowley. Algorithm 945: modred – a parallelized model reduction library. *ACM Transactions on Mathematical Software*, 40(4):30, 2014.
[51] Michael Benedicks. On Fourier transforms of functions supported on sets of finite Lebesgue measure. *Journal of Mathematical Analysis and Applications*, 106(1):180–183, 1985.
[52] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, pages 153–160, 2007.
[53] P. Benner, S. Gugercin, and K Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.
[54] Peter Benner, Albert Cohen, Mario Ohlberger, and Karen Willcox. *Model Reduction and Approximation: Theory and Algorithms*, volume 15. *SIAM*, 2017.
[55] Peter Benner, Jing-Rebecca Li, and Thilo Penzl. Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems. *Numerical Linear Algebra with Applications*, 15(9):755–777, 2008.
[56] E. Berger, M. Sastuba, D. Vogt, B. Jung, and H. B. Amor. Estimation of perturbations in robotic behavior using dynamic mode decomposition. *Journal of Advanced Robotics*, 29(5):331–343, 2015.
[57] G. Berkooz, P. Holmes, and J. L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25:539–575, 1993.
[58] Gregory Beylkin, Ronald Coifman, and Vladimir Rokhlin. Fast wavelet transforms and numerical algorithms i. *Communications on Pure and Applied Mathematics*, 44(2):141–183, 1991.
[59] Stephen A. Billings. *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. John Wiley & Sons, 2013.
[60] P. Binetti, K. B. Ariyur, M. Krstic´, and F. Bernelli. Formation flight optimization using extremum seeking feedback. *Journal of Guidance, Control, and Dynamics*, 26(1):132–142, 2003.
[61] G. D. Birkhoff and B. O. Koopman. Recent contributions to the ergodic theory. *Proceedings of the National Academy of Sciences*, 18(3):279–282, 1932.
[62] George D. Birkhoff. Proof of the ergodic theorem. *Proceedings of the National Academy of Sciences*, 17(12):656–660, 1931.
[63] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford university press, 1995.
[64] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer New York, 2006.
[65] D. A. Bistrian and I. M. Navon. An improved algorithm for the shallow water equations model reduction: Dynamic mode decomposition vs POD. *International Journal for Numerical Methods in Fluids*, 2015.
[66] D. A. Bistrian and I. M. Navon. Randomized dynamic mode decomposition for non-intrusive reduced order modelling. *International Journal for Numerical Methods in Engineering*, 2016.

[67] Paola Bondi, Giuseppe Casalino, and Lucia Gambardella. On the iterative learning control theory for robotic manipulators. *IEEE Journal on Robotics and Automation*, 4(1):14–22, 1988.
[68] Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.
[69] Jorge Luis Borges. The library of Babel. *Collected Fictions*, 1998.
[70] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM, 1992.
[71] H. Boulard and Y. Kamp. Autoassociative memory by multilayer perceptron and singular values decomposition. *Biol Cybern*, 59:291–294, 1989.
[72] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.
[73] Stephen Boyd, Leon O. Chua, and Charles A. Desoer. Analytical foundations of Volterra series. *IMA Journal of Mathematical Control and Information*, 1(3):243–282, 1984.
[74] Stephen Boyd and Lieven Vandenberghe. *Convex Pptimization*. Cambridge University Press, 2009.
[75] Antoneta Iuliana Bratcu, Iulian Munteanu, Seddik Bacha, and Bertrand Raison. Maximum power point tracking of grid-connected photovoltaic arrays by using extremum seeking control. *CEAI*, 10(4):3–12, 2008.
[76] Leo Breiman. Better subset regression using the nonnegative garrote. *Technometrics*, 37(4):373–384, 1995.
[77] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
[78] Leo Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231, 2001.
[79] Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and Regression Trees*. CRC press, 1984.
[80] I. Bright, G. Lin, and J. N. Kutz. Compressive sensing and machine learning strategies for characterizing the flow around a cylinder with limited pressure measurements. *Physics of Fluids*, 25(127102):1–15, 2013.
[81] Ido Bright, Guang Lin, and J. Nathan Kutz. Classification of spatio-temporal data via asynchronous sparse sampling: Application to flow around a cylinder. *Multiscale Modeling & Simulation*, 14 (2), 823–838, 2016.
[82] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.
[83] Douglas Bristow, Marina Tharayil, Andrew G. Alleyne, et al. A survey of iterative learning control. *Control Systems, IEEE*, 26(3):96–114, 2006.
[84] Rasmus Bro. Parafac. tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, 38(2):149–171, 1997.
[85] Alexander Broad, T. D. Murphey, and Brenna Argall. Learning models for shared control of human-machine systems with unknown dynamics. *Robotics: Science and Systems Proceedings*, 2017.
[86] Roger W. Brockett. Volterra series and geometric control theory. *Automatica*, 12(2):167–176, 1976.

[87] David S. Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[88] D. S. Broomhead and R Jones. Time-series analysis. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 423, pages 103–121. The Royal Society, 1989.

[89] B. W. Brunton, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Sparse sensor placement optimization for classification. *SIAM Journal on Applied Mathematics*, 76(5):2099–2122, 2016.

[90] B. W. Brunton, L. A. Johnson, J. G. Ojemann, and J. N. Kutz. Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. *Journal of Neuroscience Methods*, 258:1–15, 2016.

[91] S. L. Brunton, B. W. Brunton, J. L. Proctor, E. Kaiser, and J. N. Kutz. Chaos as an intermittently forced linear system. *Nature Communications*, 8(19):1–9, 2017.

[92] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PLoS ONE*, 11(2):e0150171, 2016.

[93] S. L. Brunton, X. Fu, and J. N. Kutz. Extremum-seeking control of a mode-locked laser. *IEEE Journal of Quantum Electronics*, 49(10):852–861, 2013.

[94] S. L. Brunton and B. R. Noack. Closed-loop turbulence control: Progress and challenges. *Applied Mechanics Reviews*, 67:050801–1–050801–48, 2015.

[95] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[96] S. L. Brunton, J. L. Proctor, J. H. Tu, and J. N. Kutz. Compressed sensing and dynamic mode decomposition. *Journal of Computational Dynamics*, 2(2):165–191, 2015.

[97] S. L. Brunton and C. W. Rowley. Maximum power point tracking for photovoltaic optimization using ripple-based extremum seeking control. *IEEE Transactions on Power Electronics*, 25(10):2531–2540, 2010.

[98] S. L. Brunton, J. H. Tu, I. Bright, and J. N. Kutz. Compressive sensing and low-rank libraries for classification of bifurcation regimes in nonlinear dynamical systems. *SIAM Journal on Applied Dynamical Systems*, 13(4):1716–1732, 2014.

[99] Steven L. Brunton, Xing Fu, and J. Nathan Kutz. Self-tuning fiber lasers. *IEEE Journal of Selected Topics in Quantum Electronics*, 20(5), 2014.

[100] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Sparse identification of nonlinear dynamics with control (SINDYc). *IFAC NOLCOS*, 49(18):710–715, 2016.

[101] D. Buche, Peter Stoll, Rolf Dornberger, and Petros Koumoutsakos. Multi-objective evolutionary algorithm for the optimization of noisy combustion processes. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 32(4):460–473, 2002.

[102] Marko Budišic´ and Igor Mezic´. An approximate parametrization of the ergodic partition using time averaged observables. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 3162–3168. IEEE, 2009.

[103] Marko Budišic´ and Igor Mezic´. Geometry of the ergodic quotient reveals coherent structures in flows. *Physica D: Nonlinear Phenomena*, 241(15):1255–1269, 2012.
[104] Marko Budišic´, Ryan Mohr, and Igor Mezic´. Applied Koopmanism a). *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510, 2012.
[105] Kenneth P. Burnham and David R. Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer Science & Business Media, 2003.
[106] Peter A. Businger and Gene H. Golub. Algorithm 358: Singular value decomposition of a complex matrix [f1, 4, 5]. *Communications of the ACM*, 12(10):564–565, 1969.
[107] Eduardo F. Camacho and Carlos Bordons Alba. *Model Predictive Control*. Springer Science & Business Media, 2013.
[108] Erik Cambria, Guang-Bin Huang, Liyanaarachchi Lekamalage Chamara Kasun, Hongming Zhou, Chi Man Vong, Jiarun Lin, Jianping Yin, Zhiping Cai, Qiang Liu, Kuan Li, et al. Extreme learning machines [trends & controversies]. *IEEE Intelligent Systems*, 28(6):30–59, 2013.
[109] E. J. Candès. Compressive sensing. *Proceedings of the International Congress of Mathemat- ics*, 2006.
[110] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11–1–11–37, 2011.
[111] E. J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal recon- struction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.
[112] E. J. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications in Pure and Applied Mathematics*, 8(1207–1223), 59.
[113] E. J. Candès and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.
[114] E. J. Candès and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, pages 21–30, 2008.
[115] Emmanuel J. Candes and Terence Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215, 2005.
[116] Yanhua Cao, Jiang Zhu, Zhendong Luo, and IM Navon. Reduced-order modeling of the upper tropical pacific ocean model using proper orthogonal decomposition. *Computers & Mathematics with Applications*, 52(8):1373–1386, 2006.
[117] Yanhua Cao, Jiang Zhu, I Michael Navon, and Zhendong Luo. A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids*, 53(10):1571–1583, 2007.
[118] Kevin Carlberg, Matthew Barone, and Harbir Antil. Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction. *Journal of Computational Physics*, 330:693–734, 2017.
[119] Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. Efficient nonlinear model reduction via a least-squares Petrov–Galerkin projection and

compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011.

[120] Kevin Carlberg, Charbel Farhat, Julien Cortial, and David Amsallem. The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.

[121] Torsten Carleman. Application de la théorie des équations intégrales linéaires aux systémes d'équations différentielles non linéaires. *Acta Mathematica*, 59(1):63–87, 1932.

[122] Torsten Carleman. Sur la théorie de l'équation intégrodifférentielle de boltzmann. *Acta Mathematica*, 60(1):91–146, 1933.

[123] Torsten Carleman. Sur les systemes lineaires aux dérivées partielles du premier ordrea deux variables. *CR Acad. Sci. Paris*, 197:471–474, 1933.

[124] J. Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35:283–319, 1970.

[125] Rick Chartrand. Numerical differentiation of noisy, nonsmooth data. *ISRN Applied Mathematics*, 2011, 2011.

[126] Anindya Chatterjee. An introduction to the proper orthogonal decomposition. *Current Science*, 78(7):808–817, 2000.

[127] S. Chaturantabut and D. C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[128] K. K. Chen and C. W. Rowley. Normalized coprime robust stability and performance guarantees for reduced-order controllers. *IEEE Transactions on Automatic Control*, 58(4):1068–1073, 2013.

[129] K. K. Chen, J. H. Tu, and C. W. Rowley. Variants of dynamic mode decomposition: Boundary condition, Koopman, and Fourier analyses. *Journal of Nonlinear Science*, 22(6):887–915, 2012.

[130] Yangquan Chen, Kevin L Moore, and Hyo-Sung Ahn. Iterative learning control. In *Encyclopedia of the Sciences of Learning*, pages 1648–1652. Springer, 2012.

[131] Steve Cherry. Singular value decomposition analysis and canonical correlation analysis. *Journal of Climate*, 9(9):2003–2009, 1996.

[132] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder- decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[133] J. Y. Choi, M. Krstic´, K. B. Ariyur, and J. S. Lee. Extremum seeking control for discrete-time systems. *IEEE Transactions on Automatic Control*, 47(2):318–323, FEB 2002.

[134] Youngsoo Choi, David Amsallem, and Charbel Farhat. Gradient-based constrained optimization using a database of linear reduced-order models. *arXiv preprint arXiv:1506.07849*, 2015.

[135] T. Colonius and K. Taira. A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions. *Computer Methods in Applied Mechanics and Engineering*, 197:2131–2146, 2008.

[136] James W. Cooley, Peter A. W. Lewis, and Peter D. Welch. Historical notes on the fast Fourier transform. *Proceedings of the IEEE*, 55(10):1675–1677, 1967.
[137] James W. Cooley and John W Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.
[138] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
[139] Mark C. Cross and Pierre C. Hohenberg. Pattern formation outside of equilibrium. *Reviews of Modern Physics*, 65(3):851, 1993.
[140] James P. Crutchfield and Bruce S. McNamara. Equations of motion from a data series. *Complex Systems*, 1:417–452, 1987.
[141] Magnus Dam, Morten Brøns, Jens Juul Rasmussen, Volker Naulin, and Jan S. Hesthaven. Sparse identification of a predator-prey system from simulation data of a convection model. *Physics of Plasmas*, 24(2):022310, 2017.
[142] Bryan C. Daniels and Ilya Nemenman. Automated adaptive inference of phenomenological dynamical models. *Nature Communications*, 6, 2015.
[143] Bryan C. Daniels and Ilya Nemenman. Efficient inference of parsimonious phenomenological models of cellular dynamics using s-systems and alternating regression. *PloS one*, 10(3):e0119821, 2015.
[144] Suddhasattwa Das and Dimitrios Giannakis. Delay-coordinate maps and the spectra of Koopman operators. *arXiv preprint arXiv:1706.08544*, 2017.
[145] Ingrid Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE transactions on information theory*, 36(5):961–1005, 1990.
[146] Lawrence Davis et al. *Handbook of Genetic Algorithms*, volume 115. Van Nostrand Reinhold New York, 1991.
[147] Scott T. M. Dawson, Maziar S. Hemati, Matthew O. Williams, and Clarence W. Rowley. Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition. *Experiments in Fluids*, 57(3):1–19, 2016.
[148] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, pages 1–38, 1977.
[149] Santosh Devasia, Degang Chen, and Brad Paden. Nonlinear inversion-based output tracking. *Automatic Control, IEEE Transactions on*, 41(7):930–942, 1996.
[150] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289– 1306, 2006.
[151] D. L. Donoho and M. Gavish. Code supplement to "The optimal hard threshold for singular values is $4/\sqrt{3}$". http://purl.stanford.edu/vg705qn9070, 2014.
[152] David Donoho. 50 years of data science. In *Based on a Presentation at the Tukey Centennial Workshop*. NJ Princeton, 2015.
[153] David L. Donoho, Iain M Johnstone, Jeffrey C Hoch, and Alan S Stern. Maximum entropy and the nearly black object. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 41–81, 1992.
[154] David L. Donoho and Jain M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.
[155] J. C. Doyle. Guaranteed margins for LQG regulators. *IEEE Transactions on Automatic Control*, 23(4):756–757, 1978.

[156] J. C. Doyle, K. Glover, P. P. Khargonekar, and B. A. Francis. State-space solutions to standard H2 and H∞ control problems. *IEEE Transactions on Automatic Control*, 34(8):831–847, 1989.
[157] John C. Doyle, Bruce A. Francis, and Allen R. Tannenbaum. *Feedback Control Theory*. Courier Corporation, 2013.
[158] Petros Drineas and Michael W. Mahoney. A randomized algorithm for a tensor-based generalization of the singular value decomposition. *Linear Algebra and Its Applications*, 420(2-3):553–571, 2007.
[159] Zlatko Drmac and Serkan Gugercin. A new selection operator for the discrete empirical interpolation method – improved a priori error bound and extensions. *SIAM Journal on Scientific Computing*, 38(2):A631–A648, 2016.
[160] Qiang Du and Max Gunzburger. Model reduction by proper orthogonal decomposition coupled with centroidal voronoi tessellations (keynote). In *ASME 2002 Joint US-European Fluids Engineering Division Conference*, pages 1401–1406. American Society of Mechanical Engineers, 2002.
[161] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2000. [162] Jed A. Duersch and Ming Gu. Randomized QR with column pivoting. *SIAM Journal on Scientific Computing*, 39(4):C263–C291, 2017.
[163] D. Duke, D. Honnery, and J. Soria. Experimental investigation of nonlinear instabilities in annular liquid sheets. *Journal of Fluid Mechanics*, 691:594–604, 2012.
[164] Daniel Duke, Julio Soria, and Damon Honnery. An error analysis of the dynamic mode decomposition. *Experiments in Fluids*, 52(2):529–542, 2012.
[165] Geir. E. Dullerud and Fernando Paganini. *A Course in Robust Control Theory: A Convex Approach*. Texts in Applied Mathematics. Springer, Berlin, Heidelberg, 2000.
[166] R. Dunne and B. J. McKeon. Dynamic stall on a pitching and surging airfoil. *Experiments in Fluids*, 56(8):1–15, 2015.
[167] T. Duriez, S. L. Brunton, and B. R. Noack. *Machine Learning Control: Taming Nonlinear Dynamics and Turbulence*. Springer, 2016.
[168] Thomas Duriez, Vladimir Parezanovic´, Laurent Cordier, Bernd R. Noack, Joël Delville, Jean-Paul Bonnet, Marc Segond, and Markus Abel. Closed-loop turbulence control using machine learning. *arXiv preprint arXiv:1404.4589*, 2014.
[169] Thomas Duriez, Vladimir Parezanovic, Jean-Charles Laurentie, Carine Fourment, Joël Delville, Jean-Paul Bonnet, Laurent Cordier, Bernd R Noack, Marc Segond, Markus Abel, Nicolas Gautier, Jean-Luc Aider, Cedric Raibaudo, Christophe Cuvier, Michel Stanislas, and Steven L Brunton. Closed-loop control of experimental shear flows using machine learning. AIAA Paper 2014-2219, 7th Flow Control Conference, 2014.
[170] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
[171] Jens L. Eftang, Anthony T. Patera, and Einar M. Rønquist. An “hp” certified reduced basis method for parametrized elliptic partial differential equations. *SIAM Journal on Scientific Computing*, 32(6):3170–3200, 2010.
[172] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[173] Utku Eren, Anna Prach, Bas¸aran Bahadır Koçer, Saša V. Rakovic´, Erdal Kayacan, and Behçet Açıkmes¸e. Model predictive control in aerospace systems: Current state and opportunities. *Journal of Guidance, Control, and Dynamics*, 40(7):1541–1566, 2017.

[174] N. B. Erichson, S. L. Brunton, and J. N. Kutz. Compressed dynamic mode decomposition for real-time object detection. *Journal of Real-Time Image Processing*, 2016.

[175] N. B. Erichson, S. L. Brunton, and J. N. Kutz. Randomized dynamic mode decomposition. *arXiv preprint arXiv:1702.02912*, 2017.

[176] N. B. Erichson, K. Manohar, S. L. Brunton, and J. N. Kutz. Randomized CP tensor decomposition. *arXiv preprint arXiv:1703.09074*.

[177] N. B. Erichson, S. Voronin, S. L. Brunton, and J. N. Kutz. Randomized matrix decompositions using R. *arXiv preprint arXiv:1608.02148*, 2016.

[178] Trishan Esram, Jonathan W. Kimball, Philip T Krein, Patrick L. Chapman, and Pallab Midya. Dynamic maximum power point tracking of photovoltaic arrays using ripple correlation control. *Ieee Transactions On Power Electronics*, 21(5):1282–1291, September 2006.

[179] Richard Everson and Lawrence Sirovich. Karhunen–Loeve procedure for gappy data. *JOSA A*, 12(8):1657–1664, 1995.

[180] N. Fabbiane, O. Semeraro, S. Bagheri, and D. S. Henningson. Adaptive and model-based control theory applied to convectively unstable flows. *Appl. Mech. Rev.*, 66(6):060801–1–20, 2014.

[181] B. F. Feeny. On proper orthogonal co-ordinates as indicators of modal activity. *Journal of Sound and Vibration*, 255(5):805–817, 2002.

[182] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7(2):179–188, 1936.

[183] Ronald Aylmer Fisher. On the mathematical foundations of theoretical statistics. *Philo- sophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222:309–368, 1922.

[184] P. J. Fleming and R. C. Purshouse. Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice*, 10:1223–1241, 2002.

[185] Jean Baptiste Joseph Fourier. *The Analytical Theory of Heat*. The University Press, 1878.

[186] Joseph Fourier. *Theorie analytique de la chaleur, par M. Fourier*. Chez Firmin Didot, père et fils, 1822.

[187] J. E. Fowler. Compressive-projection principal component analysis. *IEEE Transactions on Image Processing*, 18(10):2230–2242, 2009.

[188] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[189] Jerome H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[190] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM*, 51(6):1025–1041, 2004.

[191] Xing Fu, Steven L. Brunton, and J. Nathan Kutz. Classification of birefringence in mode-locked fiber lasers using machine learning and sparse representation. *Optics Express*, 22(7):8585–8597, 2014.
[192] Koji Fukagata, Stefan Kern, Philippe Chatelain, Petros Koumoutsakos, and Nobuhide Kasagi. Evolutionary optimization of an anisotropic compliant surface for turbulent friction drag reduction. *Journal of Turbulence*, 9(35):1–17, 2008.
[193] F. Fukushima. A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetic*, 36:193–202, 1980.
[194] H. Gao, J. Lam, C. Wang, and Y. Wang. Delay-dependent output-feedback stabilisation of discrete-time systems with time-varying state delay. *IEE Proceedings-Control Theory and Applications*, 151(6):691–698, 2004.
[195] Carlos E. Garcia, David M. Prett, and Manfred Morari. Model predictive control: theory and practice: A survey. *Automatica*, 25(3):335–348, 1989.
[196] Jorge L. Garriga and Masoud Soroush. Model predictive control tuning methods: A review. *Industrial & Engineering Chemistry Research*, 49(8):3505–3515, 2010.
[197] Carl-Friedrich Gauss. *Theoria combinationis observationum erroribus minimis obnoxiae*, volume 1. Henricus Dieterich, 1823.
[198] C. F. Gauss. Nachlass: Theoria interpolationis methodo nova tractata, volume werke. *Königliche Gesellschaft der Wissenschaften, Göttingen*, 1866.
[199] Nicolas Gautier, J-L. Aider, Thomas Duriez, B. R. Noack, Marc Segond, and Markus Abel. Closed-loop separation control using machine learning. *Journal of Fluid Mechanics*, 770:442– 457, 2015.
[200] M. Gavish and D. L. Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Transactions on Information Theory*, 60(8):5040–5053, 2014.
[201] Mattia Gazzola, Oleg V. Vasilyev, and Petros Koumoutsakos. Shape optimization for drag reduction in linked bodies using evolution strategies. *Computers & Structures*, 89(11):1224– 1231, 2011.
[202] Gregor Gelbert, Jonas P. Moeck, Christian O. Paschereit, and Rudibert King. Advanced algorithms for gradient estimation in one-and two-parameter extremum seeking controllers. *Journal of Process Control*, 22(4):700–709, 2012.
[203] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(6):643–660, 2001.
[204] Jan J. Gerbrands. On the relationships between SVD, KLT and PCA. *Pattern Recognition*, 14(1):375–381, 1981.
[205] A. C. Gilbert and P. Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010.
[206] A. C. Gilbert, J. Y. Park, and M. B. Wakin. Sketched SVD: Recovering spectral features from compressive measurements. *ArXiv e-prints*, 2012.
[207] A. C. Gilbert, M. J. Strauss, and J. A. Tropp. A tutorial on fast Fourier sampling. *IEEE Signal Processing Magazine*, pages 57–66, 2008.
[208] Bryan Glaz, Li Liu, and Peretz P Friedmann. Reduced-order nonlinear unsteady aerodynamic modeling using a surrogate-based recurrence framework. *AIAA Journal*, 48(10):2418–2429, 2010.

[209] Philip J Goddard and Keith Glover. Controller approximation: approaches for preserving *H*∞ performance. *IEEE Transactions on Automatic Control*, 43(7):858–871, 1998.
[210] David E Goldberg. *Genetic Algorithms*. Pearson Education India, 2006.
[211] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerical Mathematics*, 14:403–420, 1970.
[212] Gene Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial & Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965.
[213] Gene Golub, Stephen Nash, and Charles Van Loan. A Hessenberg-Schur method for the problem *ax* + *xb* = *c*. *IEEE Transactions on Automatic Control*, 24(6):909–913, 1979.
[214] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*, volume 3. JHU Press, 2012.
[215] R. Gonzalez-Garcia, R. Rico-Martinez, and I. G. Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Comp. & Chem. Eng.*, 22:S965–S968, 1998.
[216] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.
[217] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
[218] Michael Grant, Stephen Boyd, and Yinyu Ye. Cvx: Matlab software for disciplined convex programming, 2008.
[219] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.
[220] Anne Greenbaum. *Iterative Methods for Solving Linear Systems*. *SIAM,* 1997.
[221] Mohinder S. Grewal. Kalman filtering. In *International Encyclopedia of Statistical Science*, pages 705–708. Springer, 2011.
[222] M. Grilli, P. J. Schmid, S. Hickel, and N. A. Adams. Analysis of unsteady behaviour in shockwave turbulent boundary layer interaction. *Journal of Fluid Mechanics*, 700:16–28, 2012.
[223] Jacob Grosek and J. Nathan Kutz. Dynamic mode decomposition for real-time background/foreground separation in video. *arXiv preprint arXiv:1404.7592*, 2014.
[224] Ming Gu. Subspace iteration randomization and singular value problems. *SIAM Journal on Scientific Computing*, 37(3):1139–1173, 2015.
[225] F. Gueniat, L. Mathelin, and L. Pastur. A dynamic mode decomposition approach for large and arbitrarily sampled systems. *Physics of Fluids*, 27(2):025113, 2015.
[226] Fredrik Gustafsson, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell, Jonas Jansson, Rickard Karlsson, and P-J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on signal processing*, 50(2):425–437, 2002.
[227] Alfred Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371, 1910.

[228] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217– 288, 2011.
[229] Nathan Halko, Per-Gunnar Martinsson, Yoel Shkolnisky, and Mark Tygert. An algorithm for the principal component analysis of large data sets. *SIAM Journal on Scientific Computing*, 33:2580–2594, 2011.
[230] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.
[231] Sven J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA Journal of Numerical Analysis*, 2(3):303–323, 1982.
[232] Sangbo Han and Brian Feeny. Application of proper orthogonal decomposition to structural vibration analysis. *Mechanical Systems and Signal Processing*, 17(5):989–1001, 2003.
[233] Nikolaus Hansen, André S. P. Niederberger, Lino Guzzella, and Petros Koumoutsakos. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Transactions on Evolutionary Computation*, 13(1):180–197, 2009.
[234] David Harrison Jr. and Daniel L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, 1978.
[235] Richard A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA working papers in phonetics*, 16:1–84, 1970. Available at www.psychology.uwo.ca/faculty/harshman/wpppfac0.pdf.
[236] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J. Friedman, and R. Tibshirani. *The Elements of Statistical Learning*, volume 2. Springer, 2009.
[237] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
[238] M. T. Heath, A. J. Laub, C. C. Paige, and R. C. Ward. Computing the singular value decomposition of a product of two matrices. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1147–1159, 1986.
[239] Michael Heideman, Don Johnson, and C Burrus. Gauss and the history of the fast Fourier transform. *IEEE ASSP Magazine*, 1(4):14–21, 1984.
[240] Werner Heisenberg. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. In *Original Scientific Papers Wissenschaftliche Originalarbeiten*, pages 478–504. Springer, 1985.
[241] Maziar S. Hemati, Clarence W. Rowley, Eric A. Deem, and Louis N. Cattafesta. De-biasing the dynamic mode decomposition for applied Koopman spectral analysis. *Theoretical and Computational Fluid Dynamics*, 31(4):349–368, 2017.
[242] Maziar S. Hemati, Matthew O. Williams, and Clarence W. Rowley. Dynamic mode decom- position for large and streaming datasets. *Physics of Fluids (1994-present)*, 26(11):111701, 2014.
[243] Kyle K. Herrity, Anna C. Gilbert, and Joel A. Tropp. Sparse approximation via iterative thresholding. In *Acoustics, Speech and Signal Processing, 2006.*

ICASSP 2006 Proceedings. 2006 IEEE International Conference on, volume 3, pages III–III. IEEE, 2006.

[244] Jan S. Hesthaven, Gianluigi Rozza, and Benjamin Stamm. Certified reduced basis methods for parametrized partial differential equations. *SpringerBriefs in Mathematics*, 2015.

[245] Tony Hey, Stewart Tansley, Kristin M. Tolle, et al. *The Fourth Paradigm: Data-Intensive Scien tific Discovery*, volume 1. Microsoft research Redmond, WA, 2009.

[246] Geoffrey E. Hinton and Terrence J. Sejnowski. Learning and releaming in boltzmann machines. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1(282-317):2, 1986.

[247] B. L. Ho and R. E. Kalman. Effective construction of linear state-variable models from input/output data. In *Proceedings of the 3rd Annual Allerton Conference on Circuit and System Theory*, pages 449–459, 1965.

[248] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[249] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[250] John H. Holland. *Adaptation in natural and artificial systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975.

[251] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, Cambridge, 2nd paperback edition, 2012.

[252] Philip Holmes and John Guckenheimer. *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, volume 42 of *Applied Mathematical Sciences*. Springer-Verlag, Berlin, Heidelberg, 1983.

[253] Eberhard Hopf. The partial differential equation $ut + uux = \mu uxx$. *Communications on Pure and Applied Mathematics*, 3(3):201–230, 1950.

[254] John J. Hopfield. Neural networks and physical systems with emergent collective computa- tional abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[255] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[256] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, September 1933.

[257] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520, October 1933.

[258] C. Huang, W. E. Anderson, M. E. Harvazinski, and V. Sankaran. Analysis of self-excited combustion instabilities using decomposition techniques. In *51st AIAA Aerospace Sciences Meeting*, pages 1–18, 2013.

[259] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962.

[260] Peter J. Huber. Robust statistics. In *International Encyclopedia of Statistical Science*, pages 1248–1251. Springer, 2011.

[261] S. J. Illingworth, A. S. Morgans, and C. W. Rowley. Feedback control of flow resonances using balanced reduced-order models. *Journal of Sound and Vibration*, 330(8):1567–1581, 2010.
[262] Eric Jacobsen and Richard Lyons. The sliding DFT. *IEEE Signal Processing Magazine*, 20(2):74–80, 2003.
[263] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.
[264] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer, 2013.
[265] M. C. Johnson, S. L. Brunton, N. B. Kundtz, and J. N. Kutz. Extremum-seeking control of a beam pattern of a reconfigurable holographic metamaterial antenna. *Journal of the Optical Society of America A*, 33(1):59–68, 2016.
[266] Richard A. Johnson and Dean Wichern. *Multivariate Analysis*. Wiley Online Library, 2002.
[267] W. B Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26(189-206):1, 1984.
[268] Ian Jolliffe. *Principal Component Analysis*. Wiley Online Library, 2005.
[269] Siddharth Joshi and Stephen Boyd. Sensor selection via convex optimization. *IEEE Transactions on Signal Processing*, 57(2):451–462, 2009.
[270] Mihailo R. Jovanovic´, Peter J. Schmid, and Joseph W. Nichols. Sparsity-promoting dynamic mode decomposition. *Physics of Fluids*, 26(2):024103, 2014.
[271] J. N. Juang. *Applied System Identification*. Prentice Hall PTR, Upper Saddle River, New Jersey, 1994.
[272] J. N. Juang and R. S. Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics*, 8(5):620– 627, 1985.
[273] J. N. Juang, M. Phan, L. G. Horta, and R. W. Longman. Identification of observer/Kalman filter Markov parameters: Theory and experiments. Technical Memorandum 104069, NASA, 1991.
[274] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Int. symp. aerospace/defense sensing, simul. and controls*, volume 3, pages 182–193. Orlando, FL, 1997.
[275] Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.
[276] E. Kaiser, J. N. Kutz, and S. L. Brunton. Data-driven discovery of Koopman eigenfunctions for control. *arXiv preprint arXiv:1707.01146*, 2017.
[277] Eurika Kaiser, J. Nathan Kutz, and Steven L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. To appear in *Proceedings of the Royal Society A*. *arXiv preprint arXiv:1711.05501*, 2017.
[278] Eurika Kaiser, Bernd R. Noack, Laurent Cordier, Andreas Spohn, Marc Segond, Markus Abel, Guillaume Daviller, Jan Östh, Siniša Krajnovic´, and Robert K Niven. Cluster-based reduced- order modelling of a mixing layer. *Journal of Fluid Mechanics*, 754:365–414, 2014.
[279] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

[280] K. Karhunen. Über lineare methoden in der wahrscheinlichkeitsrechnung, vol. 37. *Annales AcademiæScientiarum Fennicæ, Ser. A. I*, 1947.
[281] Kévin Kasper, Lionel Mathelin, and Hisham Abou-Kandil. A machine learning approach for constrained sensor placement. In *American Control Conference (ACC), 2015*, pages 4479–4484. IEEE, 2015.
[282] A. K. Kassam and L. N. Trefethen. Fourth-order time-stepping for stiff PDEs. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.
[283] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.
[284] Anthony R. Kellems, Saifon Chaturantabut, Danny C. Sorensen, and Steven J. Cox. Morphologically accurate reduced order modeling of spiking neurons. *Journal of Computational Neuroscience*, 28(3):477–494, 2010.
[285] J. Kepler. *Tabulae Rudolphinae, quibus Astronomicae scientiae, temporum longinquitate collapsae Restauratio continetur*. Ulm: Jonas Saur, 1627.
[286] Gaëtan Kerschen and Jean-Claude Golinval. Physical interpretation of the proper orthogonal modes using the singular value decomposition. *Journal of Sound and Vibration*, 249(5):849–865, 2002.
[287] Gaetan Kerschen, Jean-claude Golinval, Alexander F. Vakakis, and Lawrence A. Bergman. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview. *Nonlinear Dynamics*, 41(1-3):147–169, 2005.
[288] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, and C. Theodor- opoulos. Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level analysis. *Communications in Mathematical Science*, 1(4):715–762, 2003.
[289] N. J. Killingsworth and M. Krstc. PID tuning using extremum seeking: online, model-free performance optimization. *IEEE Control Systems Magazine*, February:70–79, 2006.
[290] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
[291] M. Kirby and L. Sirovich. Application of the Karhunen-Loève procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(1):103–108, 1990.
[292] V. C. Klema and A. J. Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control*, 25(2):164–176, 1980.
[293] Stefan Klus, Feliks Nüske, Péter Koltai, Hao Wu, Ioannis Kevrekidis, Christof Schütte, and Frank Noé. Data-driven model reduction and transfer operator approximation. *Journal of Nonlinear Science*, pages 1–26, 2018.
[294] Richard Koch. *The 80/20 Principle*. Nicholas Brealey Publishing, 1997.
[295] Richard Koch. *Living the 80/20 Way*. Audio-Tech Business Book Summaries, Incorporated, 2006.
[296] Richard Koch. *The 80/20 Principle: The Secret to Achieving More with Less*. Crown Business, 2011.
[297] Richard Koch. *The 80/20 Principle and 92 other Powerful Laws of Nature: the Science of Success*. Nicholas Brealey Publishing, 2013.

[298] Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1-3):1–6, 1998.
[299] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September 2009.
[300] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.
[301] B. O. Koopman and J.-v. Neumann. Dynamical systems of continuous spectra. *Proceedings of the National Academy of Sciences of the United States of America*, 18(3):255, 1932.
[302] Milan Korda and Igor Mezic´. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica,* 93:149–160, 2018.
[303] Milan Korda and Igor Mezic´. On convergence of extended dynamic mode decomposition to the Koopman operator. *Journal of Nonlinear Science,* 28(2): 687–710, 2018.
[304] Petros Koumoutsakos, Jonathan Freund, and David Parekh. Evolution strategies for automatic optimization of jet mixing. *AIAA Journal*, 39(5):967–969, 2001.
[305] Krzysztof Kowalski, Willi-Hans Steeb, and K. Kowalksi. *Nonlinear Dynamical Systems and Carleman Linearization*. World Scientific, 1991.
[306] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, volume 1. MIT press, 1992.
[307] John R. Koza, Forrest H. Bennett III, and Oscar Stiffelman. Genetic programming as a darwinian invention machine. In *Genetic Programming*, pages 93–108. Springer, 1999.
[308] Boris Kramer, Piyush Grover, Petros Boufounos, Mouhacine Benosman, and Saleh Nabi. Sparse sensing and dmd based identification of flow regimes and bifurcations in complex flows. *SIAM Journal on Applied Dynamical Systems,* 16(2):1164–1196, 2017.
[309] J. P. Krieger and M. Krstic. Extremum seeking based on atmospheric turbulence for aircraft endurance. *Journal of Guidance, Control, and Dynamics*, 34(6):1876–1885, 2011.
[310] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
[311] M. Krstic, A. Krupadanam, and C. Jacobson. Self-tuning control of a nonlinear model of combustion instabilities. *IEEE Tr. Contr. Syst. Technol.*, 7(4):424–436, 1999.
[312] M. Krstic´ and H. H. Wang. Stability of extremum seeking feedback for general nonlinear dynamic systems. *Automatica*, 36:595–601, 2000.
[313] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015.
[314] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.
[315] Karl Kunisch and Stefan Volkwein. Optimal snapshot location for computing pod basis functions. *ESAIM: Mathematical Modelling and Numerical Analysis*, 44(3):509–529, 2010.

[316] J. N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. Oxford University Press, 2013.
[317] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. *SIAM*, 2016.
[318] J. N. Kutz, X. Fu, and S. L. Brunton. Multi-resolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016.
[319] J. Nathan Kutz, Syuzanna Sargsyan, and Steven L Brunton. Leveraging sparsity and compressive sensing for reduced order modeling. In *Model Reduction of Parametrized Systems*, pages 301–315. Springer, 2017.
[320] Sanjay Lall, Jerrold E. Marsden, and Sonja Glavaški. Empirical model reduction of controlled nonlinear systems. In *IFAC World Congress*, volume F, pages 473–478. International Federation of Automatic Control, 1999.
[321] Sanjay Lall, Jerrold E. Marsden, and Sonja Glavaški. A subspace approach to balanced truncation for model reduction of nonlinear control systems. *International Journal of Robust and Nonlinear Control*, 12(6):519–535, 2002.
[322] Yueheng Lan and Igor Mezic´. Linearization in the large of nonlinear systems and Koopman operator spectrum. *Physica D: Nonlinear Phenomena*, 242(1):42–53, 2013.
[323] Alan Laub. A Schur method for solving algebraic Riccati equations. *IEEE Transactions on automatic control*, 24(6):913–921, 1979.
[324] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.
[325] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
[326] Jay H. Lee. Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, 9(3):415–424, 2011.
[327] K.C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(5):684–698, 2005.
[328] Adrien Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, 1805.
[329] V. Lenaerts, Gaëtan Kerschen, and Jean-Claude Golinval. Proper orthogonal decomposition for model updating of nonlinear mechanical systems. *Mechanical Systems and Signal Processing*, 15(1):31–43, 2001.
[330] Ian Lenz, Ross A. Knepper, and Ashutosh Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, 2015.
[331] R. Leyva, C. Alonso, I. Queinnec, A. Cid-Pastor, D. Lagrange, and L. Martinez-Salamero. MPPT of photovoltaic systems using extremum-seeking control. *Ieee Transactions On Aerospace and Electronic Systems*, 42(1):249–258, January 2006.
[332] Qianxiao Li, Felix Dietrich, Erik M. Bollt, and Ioannis G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10):103111, 2017.

[333] Y. C. Liang, H. P. Lee, S. P. Lim, W. Z. Lin, K. H. Lee, and C. G. Wu. Proper orthogonal decomposition and its applications-part i: Theory. *Journal of Sound and vibration*, 252(3):527–544, 2002.

[334] Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 581–588. ACM, 2013.

[335] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104:20167–20172, 2007.

[336] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[337] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[338] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 1999.

[339] Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[340] M. Loeve. *Probability Theory*. Van Nostrand, Princeton, NJ, 1955.

[341] J.-C. Loiseau and S. L. Brunton. Constrained sparse Galerkin regression. *Journal of Fluid Mechanics*, 838:42–67, 2018.

[342] J.-C. Loiseau, B. R. Noack, and S. L. Brunton. Sparse reduced-order modeling: sensor-based dynamics to full-state estimation. *Journal of Fluid Mechanics*, 844:459–490, 2018.

[343] Richard W. Longman. Iterative learning control and repetitive control for engineering practice. *International Journal of Control*, 73(10):930–954, 2000.

[344] E. N. Lorenz. Empirical orthogonal functions and statistical weather prediction. Technical report, Massachusetts Institute of Technology, December 1956.

[345] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963.

[346] D. M. Luchtenburg and C. W. Rowley. Model reduction using snapshot-based realizations. *Bulletin of the American Physical Society*, 56, 2011.

[347] J. L. Lumley. Toward a turbulent constitutive relation. *Journal of Fluid Mechanics*, 41(02):413–434, 1970.

[348] Bethany Lusch, Eric C. Chi, and J. Nathan Kutz. Shape constrained tensor decompositions using sparse representations in over-complete libraries. *arXiv preprint arXiv:1608.04674*, 2016.

[349] Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*. *arXiv preprint arXiv:1712.09707*, 2018.

[350] F. Lusseyran, F. Gueniat, J. Basley, C. L. Douay, L. R. Pastur, T. M. Faure, and P. J. Schmid. Flow coherent structures and frequency signature: application of the dynamic modes decomposition to open cavity flow. In *Journal of Physics: Conference Series*, volume 318, page 042036. IOP Publishing, 2011.

[351] Z. Ma, S. Ahuja, and C. W. Rowley. Reduced order models for control of fluids using the eigensystem realization algorithm. *Theor. Comput. Fluid Dyn.*, 25(1):233–247, 2011.

[352] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.
[353] Alan Mackey, Hayden Schaeffer, and Stanley Osher. On the compressive spectral method. *Multiscale Modeling & Simulation*, 12(4):1800–1827, 2014.
[354] Michael W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3:123–224, 2011.
[355] Andrew J. Majda and John Harlim. Physics constrained nonlinear regression models for time series. *Nonlinearity*, 26(1):201, 2012.
[356] Andrew J. Majda and Yoonsang Lee. Conceptual dynamical models for turbulence. *Proceedings of the National Academy of Sciences*, 111(18):6548–6553, 2014.
[357] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.
[358] Stéphane Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065):20150203, 2016.
[359] Stephane G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674– 693, 1989.
[360] John Mandel. Use of the singular value decomposition in regression analysis. *The American Statistician*, 36(1):15–24, 1982.
[361] Niall M. Mangan, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological, and Multi-Scale Communications*, 2(1):52–63, 2016.
[362] Niall M. Mangan, J. Nathan Kutz, Steven L. Brunton, and Joshua L. Proctor. Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A*, 473(2204):1–16, 2017.
[363] Jordan Mann and J. Nathan Kutz. Dynamic mode decomposition for financial trading strategies. *Quantitative Finance*, pages 1–13, 2016.
[364] K. Manohar, S. L. Brunton, and J. N. Kutz. Environmental identification in flight using sparse approximation of wing strain. *Journal of Fluids and Structures*, 70:162–180, 2017.
[365] K. Manohar, J. N. Kutz, and S. L. Brunton. Greedy Sensor and Actuator Placement Using Balanced Model Reduction. *Bulletin of the American Physical Society*, 2018.
[366] Krithika Manohar, Bingni W. Brunton, J. Nathan Kutz, and Steven L. Brunton. Data-driven sparse sensor placement. *IEEE Control Systems Magazine*, 38:63–86, 2018
[367] Krithika Manohar, Eurika Kaiser, S. L. Brunton, and J. N. Kutz. Optimized sampling for multi- scale dynamics. *SIAM Multiscale Modeling and Simulation. arXiv preprint arXiv:1712.05085*, 2017.
[368] Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. VAMPnets: Deep learning of molecular kinetics. *Nature Communications,* 9(1), 2018.
[369] J. E. Marsden and T. S. Ratiu. *Introduction to Mechanics and Symmetry*. Springer-Verlag, 2nd edition, 1999.
[370] Per-Gunnar Martinsson. Randomized methods for matrix computations and analysis of high dimensional data. *arXiv preprint arXiv:1607.01649*, 2016.

[371] Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis*, 30:47–68, 2011.
[372] John L. Maryak, James C. Spall, and Bryan D. Heydon. Use of the Kalman filter for inference in state-space models with unknown noise distributions. *IEEE Transactions on Automatic Control*, 49(1):87–90, 2004.
[373] L. Massa, R. Kumar, and P. Ravindran. Dynamic mode decomposition analysis of detonation waves. *Physics of Fluids (1994-present)*, 24(6):066101, 2012.
[374] Lionel Mathelin, Kévin Kasper, and Hisham Abou-Kandil. Observable dictionary learning for high-dimensional statistical inference. *Archives of Computational Methods in Engineering*, 25(1):103–120, 2018.
[375] R. Maury, M. Keonig, L. Cattafesta, P. Jordan, and J. Delville. Extremum-seeking control of jet noise. *Aeroacoustics*, 11(3&4):459–474, 2012.
[376] I. Mezic´. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1-3):309–325, 2005.
[377] I. Mezic´. Analysis of fluid flows via spectral properties of the Koopman operator. *Ann. Rev. Fluid Mech.*, 45:357–378, 2013.
[378] I. Mezic´. *Spectral Operator Methods in Dynamical Systems: Theory and Applications*. Springer, 2017.
[379] Igor Mezic´ and Andrzej Banaszuk. Comparison of systems with complex behavior. *Physica D: Nonlinear Phenomena*, 197(1):101–133, 2004.
[380] Igor Mezic´ and Stephen Wiggins. A method for visualization of invariant sets of dynamical systems based on the ergodic partition. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 9(1):213–218, 1999.
[381] Michele Milano and Petros Koumoutsakos. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182(1):1–26, 2002.
[382] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.
[383] Y. Mizuno, D. Duke, C. Atkinson, and J. Soria. Investigation of wall-bounded turbulent flow using dynamic mode decomposition. In *Journal of Physics: Conference Series*, volume 318, page 042040. IOP Publishing, 2011.
[384] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.
[385] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
[386] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
[387] J. P. Moeck, J.-F. Bourgouin, D. Durox, T. Schuller, and S. Candel. Tomographic recon- struction of heat release rate perturbations induced by helical modes in turbulent swirl flames. *Experiments in Fluids*, 54(4):1–17, 2013.
[388] B. C. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, AC-26(1):17–32, 1981.

[389] Calvin C. Moore. Ergodic theorem, ergodic theory, and statistical mechanics. *Proceedings of the National Academy of Sciences*, 112(7):1907–1911, 2015.
[390] Kevin L. Moore. *Iterative Learning Control for Deterministic Systems*. Springer Science & Business Media, 2012.
[391] Manfred Morari and Jay H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4):667–682, 1999.
[392] T. W. Muld, G. Efraimsson, and D. S. Henningson. Flow structures around a high-speed train extracted using proper orthogonal decomposition and dynamic mode decomposition. *Computers & Fluids*, 57:87–97, 2012.
[393] T. W. Muld, G. Efraimsson, and D. S. Henningson. Mode decomposition on surface-mounted cube. *Flow, Turbulence and Combustion*, 88(3):279–310, 2012.
[394] S. D. Müller, M Milano, and P. Koumoutsakos. Application of machine learning algorithms to flow modeling and optimization. *Annual Research Briefs*, pages 169–178, 1999.
[395] Iulian Munteanu, Antoneta Iuliana Bratcu, and Emil Ceanga. Wind turbulence used as searching signal for MPPT in variable-speed wind energy conversion systems. *Renewable Energy*, 34(1):322–327, January 2009.
[396] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.
[397] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
[398] D. Needell and J. A. Tropp. CoSaMP: iterative signal recovery from incomplete and inaccurate samples. *Communications of the ACM*, 53(12):93–100, 2010.
[399] J. v Neumann. Proof of the quasi-ergodic hypothesis. *Proceedings of the National Academy of Sciences*, 18(1):70–82, 1932.
[400] N. C. Nguyen, A. T. Patera, and J. Peraire. A best points interpolation method for efficient approximation of parametrized functions. *International Journal for Numerical Methods in Engineering*, 73(4):521–543, 2008.
[401] Yves Nievergelt and Y. Nievergelt. *Wavelets Made Easy*, volume 174. Springer, 1999.
[402] B. R. Noack, K. Afanasiev, M. Morzynski, G. Tadmor, and F. Thiele. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*, 497:335–363, 2003.
[403] B. R. Noack, T. Duriez, L. Cordier, M. Segond, M. Abel, S. L. Brunton, M. Morzyn´ski, J.-C. Laurentie, V. Parezanovic, and J.-P. Bonnet. Closed-loop turbulence control with machine learning methods. *Bulletin Am. Phys. Soc.*, 58(18):M25.0009, p. 418, 2013.
[404] Bernd R. Noack, Marek Morzynski, and Gilead Tadmor. *Reduced-Order Modelling for Flow Control*, volume 528. Springer Science & Business Media, 2011.
[405] Frank Noé and Feliks Nuske. A variational approach to modeling slow processes in stochastic dynamical systems. *Multiscale Modeling & Simulation*, 11(2):635–655, 2013.
[406] E. Noether. Invariante variationsprobleme nachr. d. könig. gesellsch. d. wiss. zu göttingen, math-phys. klasse 1918: 235-257. *English Reprint: physics/0503066, http://dx.doi.org/10.1080/00411457108231446*, page 57, 1918.

[407] Feliks Nüske, Bettina G. Keller, Guillermo Pérez-Hernández, Antonia S. J. S. Mey, and Frank Noé. Variational approach to molecular kinetics. *Journal of Chemical Theory and Computation*, 10(4):1739–1752, 2014.

[408] Feliks Nüske, Reinhold Schneider, Francesca Vitalini, and Frank Noé. Variational tensor approach for approximating the rare-event kinetics of macromolecular systems. *J. Chem. Phys.*, 144(5):054105, 2016.

[409] H. Nyquist. Certain topics in telegraph transmission theory. *Transactions of the A. I. E. E.*, pages 617–644, FEB 1928.

[410] Goro Obinata and Brian D. O. Anderson. *Model reduction for control system design*. Springer Science & Business Media, 2012.

[411] C. M. Ostoich, D. J. Bodony, and P. H. Geubelle. Interaction of a Mach 2.25 turbulent boundary layer with a fluttering panel using direct numerical simulation. *Physics of Fluids (1994-present)*, 25(11):110806, 2013.

[412] Samuel E. Otto and Clarence W. Rowley. Linearly-recurrent autoencoder networks for learning dynamics. *arXiv preprint arXiv:1712.01378*, 2017.

[413] Y. Ou, C. Xu, E. Schuster, T. C. Luce, J. R. Ferron, M. L. Walker, and D. A. Humphreys. Design and simulation of extremum-seeking open-loop optimal control of current profile in the DIII-D tokamak. *Plasma Physics and Controlled Fusion*, 50:115001–1–115001–24, 2008.

[414] Vidvuds Ozolin, š, Rongjie Lai, Russel Caflisch, and Stanley Osher. Compressed modes for variational problems in mathematics and physics. *Proceedings of the National Academy of Sciences*, 110(46):18368–18373, 2013.

[415] C. Pan, D. Yu, and J. Wang. Dynamical mode decomposition of Gurney flap wake flow. *Theoretical and Applied Mechanics Letters*, 1(1):012002, 2011.

[416] V. Parezanovic, J.-C. Laurentie, T. Duriez, C. Fourment, J. Delville, J.-P. Bonnet, L. Cordier, B. R. Noack, M. Segond, M. Abel, T. Shaqarin, and S. L. Brunton. Mixing layer manipulation experiment – from periodic forcing to machine learning closed-loop control. *Journal Flow Turbulence and Combustion*, 94(1):155–173, 2015.

[417] Vladimir Parezanovic´, Thomas Duriez, Laurent Cordier, Bernd R. Noack, Joël Delville, Jean- Paul Bonnet, Marc Segond, Markus Abel, and Steven L. Brunton. Closed-loop control of an experimental mixing layer using machine learning control. *arXiv preprint arXiv:1408.3259*, 2014.

[418] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(7–12):559–572, 1901.

[419] B. Peherstorfer, D. Butnaru, K. Willcox, and H.-J. Bungartz. Localized discrete empirical interpolation method. *SIAM Journal on Scientific Computing*, 36(1):A168–A192, 2014.

[420] Benjamin Peherstorfer and Karen Willcox. Detecting and adapting to parameter changes for reduced models of dynamic data-driven application systems. *Procedia Computer Science*, 51:2553–2562, 2015.

[421] Benjamin Peherstorfer and Karen Willcox. Dynamic data-driven reduced-order models. *Computer Methods in Applied Mechanics and Engineering*, 291:21–41, 2015.

[422] Benjamin Peherstorfer and Karen Willcox. Online adaptive model reduction for nonlinear systems via low-rank updates. *SIAM Journal on Scientific Computing*, 37(4):A2123–A2150, 2015.

[423] Sebastian Peitz and Stefan Klus. Koopman operator-based model reduction for switched- system control of PDEs. *arXiv preprint arXiv:1710.06759*, 2017.
[424] S. D. Pendergrass, J. N. Kutz, and S. L. Brunton. Streaming GPU singular value and dynamic mode decompositions. *arXiv preprint arXiv:1612.07875*, 2016.
[425] Roger Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge University Press, 1955.
[426] Roger Penrose and John Arthur Todd. On best approximate solutions of linear matrix equations. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 52, pages 17–19. Cambridge Univ Press, 1956.
[427] Lawrence Perko. *Differential Equations and Dynamical Systems*, volume 7. Springer Science & Business Media, 2013.
[428] M. Phan, L. G. Horta, J. N. Juang, and R. W. Longman. Linear system identification via an asymptotically stable observer. *Journal of Optimization Theory and Applications*, 79:59–86, 1993.
[429] Mark A. Pinsky. *Introduction to Fourier analysis and wavelets*, volume 102. American Mathematical Soc., 2002.
[430] T. Poggio. Deep learning: mathematics and neuroscience. *Views & Reviews, McGovern Center for Brains, Minds and Machines*, pages 1–7, 2016.
[431] Philippe Poncet, Georges-Henri Cottet, and Petros Koumoutsakos. Control of three-dimensional wakes using evolution strategies. *Comptes Rendus Mecanique*, 333(1):65–77, 2005.
[432] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing systems*, pages 1137–1144, 2007.
[433] J. L. Proctor, S. L. Brunton, B. W. Brunton, and J. N. Kutz. Exploiting sparsity and equation-free architectures in complex systems (invited review). *The European Physical Journal Special Topics*, 223(13):2665–2684, 2014.
[434] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.
[435] Joshua L. Proctor and Philip A. Eckhoff. Discovering dynamic patterns from infectious disease data using dynamic mode decomposition. *International Health*, 7(2):139–145, 2015.
[436] H. Qi and S. M. Hughes. Invariance of principal components under low-dimensional random projection of the data. IEEE International Conference on Image Processing, October 2012.
[437] Shie Qian and Dapang Chen. Discrete Gabor transform. *IEEE Transactions on Signal Processing*, 41(7):2429–2438, 1993.
[438] S. J. Qin and T. A. Badgwell. An overview of industrial model predictive control technology. In *AIChE Symposium Series*, volume 93, pages 232–256, 1997.
[439] S. Joe Qin and Thomas A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.
[440] Qing Qu, Ju Sun, and John Wright. Finding a sparse vector in a subspace: Linear sparsity using alternating directions. In *Advances in Neural Information Processing Systems 27*, pages 3401–3409, 2014.

[441] A. Quarteroni and G. Rozza. *Reduced Order Methods for Modeling and Computational Reduction*, volume 9 of *MS&A – Modeling, Simulation & Appplications*. Springer, 2013.
[442] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction*, volume 92. Springer, 2015.
[443] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
[444] J Ross Quinlan. *C4. 5: Programs for Machine Learning*. Elsevier, 2014.
[445] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.
[446] C. Radhakrishna Rao. The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society. Series B (Methodological)*, 10(2):159– 203, 1948.
[447] James B. Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems*, 20(3):38–52, 2000.
[448] Soumya Raychaudhuri, Joshua M. Stuart, and Russ B. Altman. Principal components analysis to summarize microarray experiments: application to sporulation time series. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, page 455. NIH Public Access, 2000.
[449] Rolf H. Reichle, Dennis B. McLaughlin, and Dara Entekhabi. Hydrologic data assimilation with the ensemble Kalman filter. *Monthly Weather Review*, 130(1):103–114, 2002.
[450] B. Ren, P. Frihauf, R. J. Rafac, and M. Krstic´. Laser pulse shaping via extremum seeking. *Control Engineering Practice*, 20:674–683, 2012.
[451] Branko Ristic, Sanjeev Arulampalam, and Neil James Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech house, 2004.
[452] Anthony John Roberts. *Model Emergent Dynamics in Complex Systems*. *SIAM*, 2014.
[453] Charles A. Rohde. Generalized inverses of partitioned matrices. *Journal of the Society for Industrial & Applied Mathematics*, 13(4):1033–1035, 1965.
[454] Vladimir Rokhlin, Arthur Szlam, and Mark Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31:1100–1124, 2009.
[455] C. W. Rowley, T. Colonius, and R. M. Murray. Model reduction for compressible flows using POD and Galerkin projection. *Physica D*, 189:115–129, 2004.
[456] C. W. Rowley, I. Mezic´, S. Bagheri, P. Schlatter, and D. S. Henningson. Spectral analysis of nonlinear flows. *J. Fluid Mech.*, 645:115–127, 2009.
[457] Clarence W. Rowley and Jerrold E. Marsden. Reconstruction equations and the Karhunen–Loève expansion for systems with symmetry. *Physica D: Nonlinear Phenomena*, 142(1):1–19, 2000.
[458] C.W. Rowley. Model reduction for fluids using balanced proper orthogonal decomposition. *Int. J. Bifurcation and Chaos*, 15(3):997–1013, 2005.
[459] S. Roy, J.-C. Hua, W. Barnhill, G. H. Gunaratne, and J. R. Gord. Deconvolution of reacting- flow dynamics using proper orthogonal and dynamic mode decompositions. *Physical Review E*, 91(1):013001, 2015.

[460] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(e1602614), 2017.

[461] Themistoklis P. Sapsis and Andrew J. Majda. Statistically accurate low-order models for uncertainty quantification in turbulent dynamical systems. *Proceedings of the National Academy of Sciences*, 110(34):13705–13710, 2013.

[462] S. Sargsyan, S. L. Brunton, and J. N. Kutz. Nonlinear model reduction for dynamical systems using sparse sensor locations from learned libraries. *Physical Review E*, 92(033304), 2015.

[463] S. Sarkar, S. Ganguly, A. Dalal, P. Saha, and S. Chakraborty. Mixed convective flow stability of nanofluids past a square cylinder by dynamic mode decomposition. *International Journal of Heat and Fluid Flow*, 44:624–634, 2013.

[464] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Foundations of Computer Science. 47th Annual IEEE Symposium on*, pages 143–152, 2006.

[465] T. Sayadi, P. J. Schmid, J. W. Nichols, and P. Moin. Reduced-order representation of near-wall structures in the late transitional boundary layer. *Journal of Fluid Mechanics*, 748:278–301, 2014.

[466] Taraneh Sayadi and Peter J. Schmid. Parallel data-driven decomposition algorithm for large-scale datasets: with application to transitional boundary layers. *Theoretical and Computational Fluid Dynamics*, pages 1–14, 2016.

[467] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher. Sparse dynamics for partial differential equations. *Proceedings of the National Academy of Sciences USA*, 110(17):6634–6639, 2013.

[468] Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. In *Proc. R. Soc. A*, volume 473, page 20160446. The Royal Society, 2017.

[469] Hayden Schaeffer and Scott G. McCalla. Sparse model selection via integral terms. *Physical Review E*, 96(2):023302, 2017.

[470] Robert E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[471] M. Schlegel, B. R. Noack, and G. Tadmor. Low-dimensional Galerkin models and control of transitional channel flow. Technical Report 01/2004, Hermann-Föttinger-Institut für Strömungsmechanik, Technische Universität Berlin, Germany, 2004.

[472] P. J. Schmid. Dynamic mode decomposition for numerical and experimental data. *J. Fluid. Mech*, 656:5–28, 2010.

[473] P. J. Schmid, L. Li, M. P. Juniper, and O. Pust. Applications of the dynamic mode decomposition. *Theoretical and Computational Fluid Dynamics*, 25 (1-4):249–259, 2011.

[474] P. J. Schmid and J. Sesterhenn. Dynamic mode decomposition of numerical and experimental data. In *61st Annual Meeting of the APS Division of Fluid Dynamics*. American Physical Society, November 2008.

[475] P. J. Schmid, D. Violato, and F. Scarano. Decomposition of time-resolved tomographic PIV. *Experiments in Fluids*, 52:1567–1579, 2012.

[476] E. Schmidt. Zur theorie der linearen und nichtlinearen integralgleichungen. i teil. entwicklung willkürlichen funktionen nach system vorgeschriebener. *Math. Ann.*, 3:433–476, 1907.

[477] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.
[478] Michael D. Schmidt, Ravishankar R. Vallabhajosyula, Jerry W. Jenkins, Jonathan E. Hood, Abhishek S. Soni, John P. Wikswo, and Hod Lipson. Automated refinement and inference of analytical models for metabolic networks. *Physical Biology*, 8(5):055011, 2011.
[479] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and beyond*. MIT press, 2002.
[480] Gideon Schwarz et al. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.
[481] A. Seena and H. J. Sung. Dynamic mode decomposition of turbulent cavity flows for self- sustained oscillations. *International Journal of Heat and Fluid Flow*, 32(6):1098–1110, 2011.
[482] Ervin Sejdic´, Igor Djurovic´, and Jin Jiang. Time–frequency feature representation using energy concentration: An overview of recent advances. *Digital Signal Processing*, 19(1):153–183, 2009.
[483] O. Semeraro, G. Bellani, and F. Lundell. Analysis of time-resolved PIV measurements of a confined turbulent jet using POD and Koopman modes. *Experiments in Fluids*, 53(5):1203–1220, 2012.
[484] Onofrio Semeraro, Francois Lusseyran, Luc Pastur, and Peter Jordan. Qualitative dynamics of wavepackets in turbulent jets. *Physical Review Fluids*, 2(9):094605, 2017.
[485] Gil Shabat, Yaniv Shmueli, Yariv Aizenbud, and Amir Averbuch. Randomized LU decomposition. *Applied and Computational Harmonic Analysis*, 2016.
[486] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.
[487] Ati S. Sharma, Igor Mezic´, and Beverley J. McKeon. Correspondence between Koopman mode decomposition, resolvent mode decomposition, and invariant solutions of the Navier-Stokes equations. *Physical Review Fluids*, 1(3):032402, 2016.
[488] Eli Shlizerman, Edwin Ding, Matthew O. Williams, and J. Nathan Kutz. The proper orthogonal decomposition for dimensionality reduction in mode-locked lasers and optical systems. *International Journal of Optics*, 2012, 2011.
[489] Valeria Simoncini. A new iterative method for solving large-scale Lyapunov matrix equations. *SIAM Journal on Scientific Computing*, 29(3):1268–1288, 2007.
[490] L. Sirovich. Turbulence and the dynamics of coherent structures, parts I-III. *Q. Appl. Math.*, XLV(3):561–590, 1987.
[491] L. Sirovich and M. Kirby. A low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 4(3):519–524, 1987.
[492] S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control*. Wiley, Chichester, 1996.
[493] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[494] Giovanni Solari, Luigi Carassale, and Federica Tubino. Proper orthogonal decomposition in wind engineering. part 1: A state-of-the-art and some prospects. *Wind and Structures*, 10(2):153–176, 2007.

[495] G. Song, F. Alizard, J.-C. Robinet, and X. Gloerfelt. Global and Koopman modes analysis of sound generation in mixing layers. *Physics of Fluids (1994-present)*, 25(12):124101, 2013.

[496] Danny C. Sorensen and Yunkai Zhou. Direct methods for matrix Sylvester and Lyapunov equations. *Journal of Applied Mathematics*, 2003(6):277–303, 2003.

[497] Mariia Sorokina, Stylianos Sygletos, and Sergei Turitsyn. Sparse identification for nonlinear optical communication systems: SINO method. *Optics Express*, 24(26):30433–30443, 2016.

[498] James C. Spall. The Kantorovich inequality for error analysis of the Kalman filter with unknown noise distributions. *Automatica*, 31(10):1513–1517, 1995.

[499] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[500] W-H. Steeb and F. Wilhelm. Non-linear autonomous systems of differential equations and Carleman linearization procedure. *Journal of Mathematical Analysis and Applications*, 77(2):601–611, 1980.

[501] Robert F. Stengel. *Optimal Control and Estimation*. Courier Corporation, 2012.

[502] Gilbert W. Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35(4):551–566, 1993.

[503] George Sugihara, Robert May, Hao Ye, Chih-hao Hsieh, Ethan Deyle, Michael Fogarty, and Stephan Munch. Detecting causality in complex ecosystems. *Science*, 338(6106):496–500, 2012.

[504] A. Surana. Koopman operator based observer synthesis for control-affine nonlinear systems. In *55th IEEE Conference on Decision and Control (CDC*, pages 6492–6499, 2016.

[505] Amit Surana and Andrzej Banaszuk. Linear observer synthesis for nonlinear systems using Koopman operator framework. *IFAC-PapersOnLine*, 49(18):716–723, 2016.

[506] Yoshihiko Susuki and Igor Mezic´. A prony approximation of Koopman mode decomposition. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 7022–7027. IEEE, 2015.

[507] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge, 1998.

[508] Adam Svenkeson, Bryan Glaz, Samuel Stanton, and Bruce J. West. Spectral decomposition of nonlinear systems with memory. *Phys. Rev. E*, 93:022211, Feb 2016.

[509] S. A. Svoronos, D. Papageorgiou, and C. Tsiligiannis. Discretization of nonlinear control systems via the Carleman linearization. *Chemical Engineering Science*, 49(19):3263–3267, 1994.

[510] D. L. Swets and J. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18(8):831–836, 1996.

[511] K. Taira and T. Colonius. The immersed boundary method: a projection approach. *Journal of Computational Physics*, 225(2):2118–2137, 2007.
[512] Naoya Takeishi, Yoshinobu Kawahara, Yasuo Tabei, and Takehisa Yairi. Bayesian dynamic mode decomposition. *Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.
[513] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning Koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems*, pages 1130–1140, 2017.
[514] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Subspace dynamic mode decomposition for stochastic Koopman analysis. *Physical Review*, E 96.3:033310, 2017.
[515] F. Takens. Detecting strange attractors in turbulence. *Lecture Notes in Mathematics*, 898:366– 381, 1981.
[516] Z. Q. Tang and N. Jiang. Dynamic mode decomposition of hairpin vortices generated by a hemisphere protuberance. *Science China Physics, Mechanics and Astronomy*, 55(1):118–124, 2012.
[517] Roy Taylor, J. Nathan Kutz, Kyle Morgan, and Brian Nelson. Dynamic mode decomposition for plasma diagnostics and validation. *Review of Scientific Instruments,* 89 (5):053501, 2018.
[518] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
[519] Zhou Ting and Jiang Hui. Eeg signal processing based on proper orthogonal decomposition. In *Audio, Language and Image Processing (ICALIP), 2012 International Conference on*, pages 636–640. IEEE, 2012.
[520] Santosh Tirunagari, Norman Poh, Kevin Wells, Miroslaw Bober, Isky Gorden, and David Windridge. Movement correction in DCE-MRI through windowed and reconstruction dynamic mode decomposition. *Machine Vision and Applications*, 28(3-4):393–407, 2017.
[521] Christopher Torrence and Gilbert P. Compo. A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society*, 79(1):61–78, 1998.
[522] Giang Tran and Rachel Ward. Exact recovery of chaotic systems from highly corrupted data. *Multiscale Modeling & Simulation,* 15 (3):1108–1129, 2017.
[523] Lloyd N. Trefethen. *Spectral methods in MATLAB*. *SIAM*, 2000.
[524] Lloyd N. Trefethen and David Bau III. *Numerical linear algebra*, volume 50. *SIAM*, 1997.
[525] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.
[526] J. A. Tropp. Recovery of short, complex linear combinations via *l*1 minimization. *IEEE Transactions on Information Theory*, 51(4):1568–1570, 2005.
[527] J. A. Tropp. Algorithms for simultaneous sparse approximation. part ii: Convex relaxation. *Signal Processing*, 86(3):589–602, 2006.
[528] J. A. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, 2006.
[529] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

[530] J. A. Tropp, A. C. Gilbert, and M. J. Strauss. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal Processing*, 86(3):572–588, 2006.

[531] J. A. Tropp, J. N. Laska, M. F. Duarte, J. K. Romberg, and R. G. Baraniuk. Beyond Nyquist: Efficient sampling of sparse bandlimited signals. *IEEE Transactions on Information Theory*, 56(1):520–544, 2010.

[532] Joel A. Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Randomized single-view algorithms for low-rank matrix approximation. *arXiv preprint arXiv:1609.00048*, 2016.

[533] J. H. Tu and C. W. Rowley. An improved algorithm for balanced POD through an analytic treatment of impulse response tails. *J. Comp. Phys.*, 231(16):5317–5333, 2012.

[534] J. H. Tu, C. W. Rowley, E. Aram, and R. Mittal. Koopman spectral analysis of separated flow over a finite-thickness flat plate with elliptical leading edge. *AIAA Paper 2011*, 2864, 2011.

[535] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. On dynamic mode decomposition: theory and applications. *J. Comp. Dyn.*, 1(2):391–421, 2014.

[536] Jonathan H. Tu, Clarence W. Rowley, J. Nathan Kutz, and Jessica K. Shang. Spectral analysis of fluid flows using sub-Nyquist-rate PIV data. *Experiments in Fluids*, 55(9):1–13, 2014.

[537] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[538] Rudolph Van Der Merwe. *Sigma-point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*. 2004.

[539] Charles Van Loan. *Computational Frameworks for the Fast Fourier Transform*. *SIAM*, 1992.

[540] Daniele Venturi and George Em Karniadakis. Gappy data and reconstruction procedures for flow past a cylinder. *Journal of Fluid Mechanics*, 519:315–336, 2004.

[541] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[542] Stefan Volkwein. Model reduction using proper orthogonal decomposition. *Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz. see http://www.uni-graz.at/imawww/volkwein/ POD.pdf*, 1025, 2011.

[543] Stefan Volkwein. Proper orthogonal decomposition: Theory and reduced-order modelling. *Lecture Notes, University of Konstanz*, 4:4, 2013.

[544] Sergey Voronin and Per-Gunnar Martinsson. RSVDPACK: Subroutines for computing partial singular value decompositions via randomized sampling on single core, multi core, and GPU architectures. *arXiv preprint arXiv: 1502.05366*, 2015.

[545] Avery Wang et al. An industrial strength audio search algorithm. In *Ismir*, volume 2003, pages 7–13. Washington, DC, 2003.

[546] H. H. Wang, M. Krstic´, and G. Bastin. Optimizing bioreactors by extremum seeking. *Adaptive Control and Signal Processing*, 13(8):651–669, 1999.

[547] H. H. Wang, S. Yeung, and M. Krstic´. Experimental application of extremum seeking on an axial-flow compressor. *IEEE Transactions on Control Systems Technology*, 8(2):300–309, 2000.
[548] W. X. Wang, R. Yang, Y. C. Lai, V. Kovanis, and C. Grebogi. Predicting catastrophes in nonlinear dynamical systems by compressive sensing. *Physical Review Letters*, 106:154101–1–154101–4, 2011.
[549] Zhu Wang, Imran Akhtar, Jeff Borggaard, and Traian Iliescu. Proper orthogonal decompo- sition closure models for turbulent flows: a numerical comparison. *Computer Methods in Applied Mechanics and Engineering*, 237:10–26, 2012.
[550] Christoph Wehmeyer and Frank Noé. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The Journal of Chemical Physics,* 148(24):241703, 2018.
[551] Greg Welch and Gary Bishop. An introduction to the Kalman filter, 1995.
[552] Peter Whitle. *Hypothesis Testing in Time Series Analysis*, volume 4. Almqvist & Wiksells, 1951.
[553] O. Wiederhold, R. King, B. R. Noack, L. Neuhaus, L. Neise, W. an Enghard, and M. Swoboda. Extensions of extremum-seeking control to improve the aerodynamic performance of axial turbomachines. In *39th AIAA Fluid Dynamics Conference*, pages 1–19, San Antonio, TX, USA, 2009. AIAA-Paper 092407.
[554] K. Willcox and J. Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA Journal*, 40(11):2323–2330, 2002.
[555] Karen Willcox. Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. *Computers & Fluids*, 35(2):208–226, 2006.
[556] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data-driven approximation of the Koopman operator: extending dynamic mode decomposition. *Journal of Nonlinear Science*, 6:1307–1346, 2015.
[557] Matthew O. Williams, Clarence W. Rowley, and Ioannis G. Kevrekidis. A kernel approach to data-driven Koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2015.
[558] Daniela M. Witten and Robert Tibshirani. Penalized classification using Fisher's linear discriminant. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(5):753–772, 2011.
[559] Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for the approximation of matrices. *Journal of Applied and Computational Harmonic Analysis*, 25:335–366, 2008.
[560] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(2):210–227, 2009.
[561] C. F. Jeff Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, pages 95–103, 1983.
[562] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, S. Yu Philip et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.
[563] Hao Ye, Richard J. Beamish, Sarah M. Glaser, Sue C. H. Grant, Chih-hao Hsieh, Laura J. Richards, Jon T. Schnute, and George Sugihara. Equation-

free mechanistic ecosystem forecasting using empirical dynamic modeling. *Proceedings of the National Academy of Sciences*, 112(13):E1569–E1576, 2015.

[564] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. *arXiv preprint arXiv:1708.06850*, 2017.

[565] B. Yildirim, C. Chryssostomidis, and G. E. Karniadakis. Efficient sensor placement for ocean measurements using low-dimensional concepts. *Ocean Modelling*, 27(3):160–173, 2009.

[566] Xiaoming Yuan and Junfeng Yang. Sparse and low-rank matrix decomposition via alternating direction methods. *preprint*, 12, 2009.

[567] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 2528–2535, 2010.

[568] C. Zhang and R. Ordó nez. Numerical optimization-based extremum seeking control with application to ABS design. *IEEE Transactions on Automatic Control*, 52(3):454–467, 2007.

[569] Hao Zhang, Clarence W. Rowley, Eric A. Deem, and Louis N. Cattafesta. Online dynamic mode decomposition for time-varying systems. *arXiv preprint arXiv:1707.02876*, 2017.

[570] T. Zhang, G. Kahn, S. Levine, and P. Abbeel. Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search. In *IEEE Robotics and Automation (ICRA)*, pages 528–535, 2016.

[571] Weiwei Zhang, Bobin Wang, Zhengyin Ye, and Jingge Quan. Efficient method for limit cycle flutter analysis based on nonlinear aerodynamic reduced-order models. *AIAA Journal*, 50(5):1019–1028, 2012.

[572] Sanjo Zlobec. An explicit form of the moore-penrose inverse of an arbitrary complex matrix. *SIAM Review*, 12(1):132–134, 1970.

[573] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Предметный указатель

C
CP-разложение, 77

D
DMD-мода (также динамическая мода), 294

E
EM-алгоритм, 223, 245

K
k ближайших соседей метод (kNN), 246
k-групповая перекрестная проверка, 192
k средних метод, 214, 244

L
LASSO, 142, 185, 193, 253, 260

M
max-пулинг, 269

N
N-мерные массивы, 76

P
PageRank, 247

А
Акаике информационный критерий, 200, 226, 319
Алгоритм реализации собственной системы (ERA), 303, 338, 413, 415
Ансамблевое обучение, 243
Аппроксимация кривыми, 161
Аппроксимация методом наименьших квадратов, 141, 163

Б
Байесовский информационный критерий, 200
Быстрое преобразование Фурье (БПФ), 32, 82, 92, 124, 465

В
Вейвлет, 113, 139
Выбор модели, 161, 164, 184, 191, 196
Выбросы, 52, 141, 151, 185
Выпуклая оптимизация, 133, 178, 366, 441

Г
Габора преобразование, 107
Галеркина проекционный метод, 308, 312, 398, 465, 470, 517
Ганкеля матрица, 82, 337, 410, 412, 416
Гильбертово пространство, 82, 320, 335
Глубокая сверточная нейронная сеть (ГСНС), 267
Глубокое обучение, 249
Градиентный спуск, 161, 169, 235, 256, 262
Грамиан, 362, 401
Гребневая регрессия, 185

Д
Датчик, 304, 339, 376, 379, 396, 416, 437, 449
Дендрограмма, 219
Джонсона–Линденштраусса лемма, 140
Динамическая система, 272, 286, 308, 320
Дискретное вейвлет-преобразование, 115
Дискретное косинусное преобразование, 135
Дискретный эмпирический метод интерполяции (DEIM), 495, 517

И
Идентификация системы, 294, 303, 313, 338, 397, 401, 415, 427, 438

К
Калмана фильтр, 369, 372, 379, 419, 458
Классификация, 58, 146, 244, 249, 263, 302, 452
Кластеризация, 57, 70, 238, 244
Когерентная структура, 33, 125, 294, 364, 399
Конструирование признаков, 204, 209, 268

Купмана оператор, 324, 333
Купмана собственная функция, 334

Л

Линейная система, 47
Линейно-квадратичный регулятор (ЛКР), 364, 444
Линейный дискриминант, 227

М

Марковские параметры, 420
Матрица данных, 65
Машинное обучение, 57, 116, 163, 207, 233, 262
Метод
  главных компонент (PCA), 37, 53, 71, 151, 412
  наименьших полных квадратов, 304
  опорных векторов (SVM), 233, 245
  переменных направлений, 175
Многомасштабность, 94, 304, 344
Многомасштабный анализ, 113
Модель
  гауссовой смеси (GMM), 223
  пониженного порядка (ROM), 302, 486, 517
Мура закон, 130

Н

Наблюдаемая функция, 321
Наблюдаемость, 82, 321, 357, 364, 379, 401, 407, 412
Наивный байесовский классификатор, 244, 246
Недоопределенная система, 34, 47, 129, 134, 146, 161, 180, 250
Нейронные сети, 170, 244, 249, 264, 340
Некогерентные измерения, 138
Неполное POD, 471, 494, 519
Низкий ранг, 37, 151, 204, 398, 470, 480, 494

О

Обратное распространение, 256, 260, 262
Обучающие данные, 227, 233, 270
Обучение
  без учителя, 203, 210, 244
  с учителем, 210
Оконное преобразование Фурье (преобразование Габора), 107

П

Парето-фронт, 188
Перекрестная проверка, 143, 190, 214, 227, 258, 304, 333
Переопределенная система, 47, 143, 161, 178, 310
Перрона–Фробениуса оператор, 293, 331
Понижение размерности, 279, 294, 344, 429, 467
Преобразование Фурье, 45, 82, 89, 331, 475
Принцип неопределенности, 112
Пространство состояний, 335, 398, 413
Псевдообращение, 47, 48, 134, 191, 253, 296

Р

Радиально-базисные функции, 237, 278
Разложение по динамическим модам (DMD), 294, 300
Разомкнутый контур, 441
Разреженная идентификация нелинейной динамики (SINDy), 308
Разреженная регрессия, 140, 309, 314, 319
Разреженное представление, 123, 129, 146, 301
Разреженность, 133
Рандомизированные алгоритмы, 71
Рандомизированное SVD, 72
Регрессия, 47, 140, 161, 210, 260, 302, 313, 415
Решающие деревья, 238
Робастная аппроксимация, 185
Робастное управление, 373, 384, 393
Робастный метод главных компонент (RPCA), 151

С

Сверточная нейронная сеть, 267
Свойство ограниченной изометрии, 131, 139
Сжатие, 38, 95, 113, 119, 137, 301
Сжатое измерение, 128, 136, 295
Сингулярное разложение (SVD), 32, 71, 139, 205, 294, 331, 399, 408, 469, 519
Скалярное произведение, 70, 77, 83, 137, 325, 331, 397, 471, 481, 496, 500
Случайный лес, 243
Смесовые модели, 223, 245
Снимок, 76, 295, 315, 364, 408, 429, 470, 475, 480, 518
Собственное ортогональное разложение (POD), 294, 398, 461, 517
Сопряженная система, 364, 409, 417, 424
Спектрограмма, 107
Стохастический градиентный спуск (СГС), 256, 264

Т

Тензоры, 77
Теория информации, 197
Тестовые данные, 216, 245

У

Унитарная матрица, 45
Управление
  на основе прогнозирующих моделей (MPC), 427
  с замкнутым контуром, 346, 349
  с обратной связью, 339, 345, 348, 397
  с поиском экстремума (ESC), 448, 455
Управляемость, 82, 357, 401, 407, 412
Упреждающее регулирование, 345, 392
Упрощение модели, 363, 397, 520
Усечение ранга, 204, 329
Усиление, 243, 246

Ф

Функция
  активации, 255, 263, 268
  стоимости, 347, 370, 378, 426, 437, 453

Ч

Частота дискретизации, 131

Ш

Шум, 133

Я

Ядерные методы, 236, 303, 333

Книги издательства «ДМК ПРЕСС»
можно купить оптом и в розницу
в книготорговой компании «Галактика»
(представляет интересы издательств
«ДМК ПРЕСС», «СОЛОН ПРЕСС», «КТК Галактика»).
Адрес: г. Москва, пр. Андропова, 38;
тел.: **(499) 782-38-89**, электронная почта: **books@alians-kniga.ru**.
При оформлении заказа следует указать адрес (полностью),
по которому должны быть высланы книги;
фамилию, имя и отчество получателя.
Желательно также указать свой телефон и электронный адрес.
Эти книги вы можете заказать и в интернет-магазине: **www.a-planeta.ru**.

Стивен Л. Брантон, Дж. Натан Куц

Анализ данных в науке и технике

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Зам. главного редактора *Сенченкова Е. А.*
Перевод *Слинкин А. А.*
Корректор *Синяева Г. И.*
Верстка *Чаннова А. А.*
Дизайн обложки *Мовчан А. Г.*

Гарнитура PT Serif. Печать цифровая.
Усл. печ. л. 44,04. Тираж 200 экз.

Веб-сайт издательства: **www.dmkpress.com**