Graduate Texts in Physics

Simon Širca
Martin Horvat

Computational Methods for Physicists

Compendium for Students

Springer

Graduate Texts in Physics

For further volumes:
www.springer.com/series/8431

Graduate Texts in Physics

Graduate Texts in Physics publishes core learning/teaching material for graduate- and advanced-level undergraduate courses on topics of current and emerging fields within physics, both pure and applied. These textbooks serve students at the MS- or PhD-level and their instructors as comprehensive sources of principles, definitions, derivations, experiments and applications (as relevant) for their mastery and teaching, respectively. International in scope and relevance, the textbooks correspond to course syllabi sufficiently to serve as required reading. Their didactic style, comprehensiveness and coverage of fundamental material also make them suitable as introductions or references for scientists entering, or requiring timely knowledge of, a research field.

Simon Širca • Martin Horvat

Computational Methods for Physicists

Compendium for Students

Springer

Simon Širca
Faculty of Mathematics and Physics
University of Ljubljana
Ljubljana, Slovenia

Martin Horvat
Faculty of Mathematics and Physics
University of Ljubljana
Ljubljana, Slovenia

ISSN 1868-4513
ISSN 1868-4521 (electronic)
Graduate Texts in Physics
ISBN 978-3-642-32477-2
ISBN 978-3-642-32478-9 (eBook)
DOI 10.1007/978-3-642-32478-9
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012951441

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Dedicated to our parents

Preface

This book evolved from short written homework instructions for the course in Computational Physics at the Department of Physics, University of Ljubljana. The feedback received from the students was used to gradually supplement the instructions by oral presentations in the classroom and additional material on the web. The heritage of this course, established and initially taught for a number of years by Professor Kodre, represented a basis onto which we attempted to span an even richer manifold, and to better elucidate the "exercises" from the mathematical, physical, as well as programming and computational viewpoints. The somewhat spartan instructions thus evolved into a much more general textbook which is intended primarily for third- and fourth-year physics students, and for Ph.D. students as an aid for all courses with a mathematical physics tinge. The book might also appeal to mathematics students. It was one of our local goals to modestly interweave physics and mathematics studies, and this is why the book steers between mathematical rigidity and more profane perspectives of numerical methods, while it tries to preserve the colorful content of the field of mathematical physics.

We were driven by the realization that physics students are often insufficiently prepared to face various obstacles they encounter in numerical solution or modeling of physical problems. Only a handful of them truly know how something can "actually be computed" or how their work can be efficiently controlled and its results reliably checked. Everyone can solve the matrix system $Ax = b$, but almost no one has an idea how to estimate the error and relate this estimate to the possible true error. They use explicit integrators of differential equations indiscriminately until they try to look closely at solutions of a problem as simple as $\ddot{x} = -x$. The direness of the situation is compounded by many commercial tools giving a false impression that all problems can be solved by a single keystroke. In parts of the text where basic approaches are discussed, we insist on seemingly ballast numerical details while, on the other hand, we did wish to offer at least some "serious" methods and illustrate them by manageable examples. The book swings back and forth between these extremes: it tries to be neither fully elementary nor encyclopedically complete, but at any rate representative—at least for the first-time reader.

The book is structured exactly with such gradations in mind: additional, "non-compulsory" chapters are marked with stars ⋆ and can be read by particularly motivated students or used as reference. Similarly, the $\odot$ symbols denote simpler tasks in the end-of-chapter problems, while more demanding ones are marked by the symbols $\bigoplus$. The purpose of the appendices is not merely to remove the superfluous contents from the main text, but to enhance programming efficiency (above all, Appendices B, C, E, I, and J). The sour apples we force our reader to bite are the lack of detailed derivations and references to formulas placed in remote parts of the text, although we tried to design the chapters as self-contained units. This style requires more concentration and consultation with literature on the reader's part, but makes the text more concise. In turn, the book does call for an inspired course tutor. In a typical one-semester course, she may hand-pick and fine-tune a dozen or so end-of-chapter problems and supply the necessary background, while the students may peruse the book as a convenient point of departure for work.

The end-of-chapter problems should resonate well with the majority of physics students. We scooped up topics from most varied disciplines and tried to embed them into the framework of the book. Chapters are concluded by relatively long lists of references, with the intent that the book will be useful also as a stepping stone for further study and as a decent vademecum.

In spite of all care, errors may have crept in. We shall be grateful to all readers turning our attention to any error they might spot, no matter how relevant. The Errata will be maintained at the book's web-page http://cmp.fmf.uni-lj.si, which also contains the data files needed in some of the problems.

We wish to express our gratitude to Professor Claus Ascheron, Senior Editor at Springer, for his effort in preparation and advancement of this book, as well as to Donatas Akmanavičius and his team for its meticulous production at VTeX.

The original text of the Slovenian edition was scrutinized by two physicists (Professors Alojz Kodre and Tomaž Prosen) as well as four mathematicians (Associate Professors Emil Žagar, Marjetka Krajnc, Gašper Jaklič, and Professor Valery Romanovski, who carefully examined the section on Gröbner bases). We thank them; from the navigation between the Scylla and Charibdis of these reviewers we emerged as better sailors and arrived happily, after years of roaming the stormy seas, to our Ithaca.

Ljubljana, Slovenia

Simon Širca
Martin Horvat

Contents

1 Basics of Numerical Analysis 1
1.1 Introduction 1
1.1.1 Finite-Precision Arithmetic 1
1.2 Approximation of Expressions 6
1.2.1 Optimal (Minimax) and Almost Optimal Approximations . 6
1.2.2 Rational (Padé) Approximation 9
1.2.3 Summation of Series by Using Padé Approximations (Wynn's ϵ-Algorithm) 12
1.2.4 Approximation of the Evolution Operator for a Hamiltonian System 14
1.3 Power and Asymptotic Expansion, Asymptotic Analysis 16
1.3.1 Power Expansion 17
1.3.2 Asymptotic Expansion 17
1.3.3 Asymptotic Analysis of Integrals by Integration by Parts . . 19
1.3.4 Asymptotic Analysis of Integrals by the Laplace Method . 21
1.3.5 Stationary-Phase Approximation 24
1.3.6 Differential Equations with Large Parameters 27
1.4 Summation of Finite and Infinite Series 31
1.4.1 Tests of Convergence 32
1.4.2 Summation of Series in Floating-Point Arithmetic 33
1.4.3 Acceleration of Convergence 36
1.4.4 Alternating Series 38
1.4.5 Levin's Transformations 42
1.4.6 Poisson Summation 44
1.4.7 Borel Summation 44
1.4.8 Abel Summation 45
1.5 Problems 46
1.5.1 Integral of the Gauss Distribution 46
1.5.2 Airy Functions 48
1.5.3 Bessel Functions 50

1.5.4 Alternating Series . . . 51
1.5.5 Coulomb Scattering Amplitude and Borel Resummation . . 52
References . . . 53

2 Solving Non-linear Equations . . . 57
2.1 Scalar Equations . . . 59
2.1.1 Bisection . . . 59
2.1.2 The Family of Newton's Methods and the Newton–Raphson Method . . . 60
2.1.3 The Secant Method and Its Relatives . . . 64
2.1.4 Müller's Method . . . 65
2.2 Vector Equations . . . 67
2.2.1 Newton–Raphson's Method . . . 67
2.2.2 Broyden's (Secant) Method . . . 69
2.3 Convergence Acceleration ⋆ . . . 72
2.4 Polynomial Equations of a Single Variable . . . 73
2.4.1 Locating the Regions Containing Zeros . . . 75
2.4.2 Descartes' Rule and the Sturm Method . . . 77
2.4.3 Newton's Sums and in Viète's Formulas . . . 79
2.4.4 Eliminating Multiple Zeros of the Polynomial . . . 80
2.4.5 Conditioning of the Computation of Zeros . . . 81
2.4.6 General Hints for the Computation of Zeros . . . 81
2.4.7 Bernoulli's Method . . . 82
2.4.8 Horner's Linear Method . . . 83
2.4.9 Bairstow's (Horner's Quadratic) Method . . . 84
2.4.10 Laguerre's Method . . . 87
2.4.11 Maehly–Newton–Raphson's Method . . . 88
2.5 Algebraic Equations of Several Variables ⋆ . . . 89
2.6 Problems . . . 94
2.6.1 Wien's Law and Lambert's Function . . . 94
2.6.2 Heisenberg's Model in the Mean-Field Approximation . . . 96
2.6.3 Energy Levels of Simple One-Dimensional Quantum Systems . . . 97
2.6.4 Propane Combustion in Air . . . 99
2.6.5 Fluid Flow Through Systems of Pipes . . . 100
2.6.6 Automated Assembly of Structures . . . 103
References . . . 106

3 Matrix Methods . . . 109
3.1 Basic Operations . . . 109
3.1.1 Matrix Multiplication . . . 109
3.1.2 Computing the Determinant . . . 111
3.2 Systems of Linear Equations $Ax = b$ . . . 111
3.2.1 Analysis of Errors . . . 111
3.2.2 Gauss Elimination . . . 113
3.2.3 Systems with Banded Matrices . . . 115

3.2.4 Toeplitz Systems . . . 115
3.2.5 Vandermonde Systems . . . 116
3.2.6 Condition Estimates for Matrix Inversion . . . 118
3.2.7 Sparse Matrices . . . 118
3.3 Linear Least-Square Problem and Orthogonalization . . . 119
3.3.1 The QR Decomposition . . . 120
3.3.2 Singular Value Decomposition (SVD) . . . 122
3.3.3 The Minimal Solution of the Least-Squares Problem . . . 126
3.4 Matrix Eigenvalue Problems . . . 127
3.4.1 Non-symmetric Problems . . . 128
3.4.2 Symmetric Problems . . . 130
3.4.3 Generalized Eigenvalue Problems . . . 133
3.4.4 Converting a Matrix to Its Jordan Form . . . 134
3.4.5 Eigenvalue Problems for Sparse Matrices . . . 136
3.5 Random Matrices ⋆ . . . 136
3.5.1 General Random Matrices . . . 136
3.5.2 Gaussian Orthogonal or Unitary Ensemble . . . 139
3.5.3 Cyclic Orthogonal and Unitary Ensemble . . . 142
3.6 Problems . . . 144
3.6.1 Percolation in a Random-Lattice Model . . . 144
3.6.2 Electric Circuits of Linear Elements . . . 146
3.6.3 Systems of Oscillators . . . 147
3.6.4 Image Compression by Singular Value Decomposition . . . 147
3.6.5 Eigenstates of Particles in the Anharmonic Potential . . . 148
3.6.6 Anderson Localization . . . 150
3.6.7 Spectra of Random Symmetric Matrices . . . 152
References . . . 154

4 Transformations of Functions and Signals . . . 159
4.1 Fourier Transformation . . . 159
4.2 Fourier Series . . . 161
4.2.1 Continuous Fourier Expansion . . . 161
4.2.2 Discrete Fourier Expansion . . . 163
4.2.3 Aliasing . . . 166
4.2.4 Leakage . . . 167
4.2.5 Fast Discrete Fourier Transformation (FFT) . . . 168
4.2.6 Multiplication of Polynomials by Using the FFT . . . 170
4.2.7 Power Spectral Density . . . 171
4.3 Transformations with Orthogonal Polynomials . . . 172
4.3.1 Legendre Polynomials . . . 174
4.3.2 Chebyshev Polynomials . . . 178
4.4 Laplace Transformation . . . 181
4.4.1 Use of Laplace Transformation with Differential Equations . . . 182

4.5 Hilbert Transformation ⋆ . . . 184
4.5.1 Analytic Signal . . . 186
4.5.2 Kramers–Kronig Relations . . . 187
4.5.3 Numerical Computation of the Continuous Hilbert Transform . . . 190
4.5.4 Discrete Hilbert Transformation . . . 192
4.6 Wavelet Transformation ⋆ . . . 195
4.6.1 Numerical Computation of the Wavelet Transform . . . 197
4.6.2 Discrete Wavelet Transform . . . 199
4.7 Problems . . . 200
4.7.1 Fourier Spectrum of Signals . . . 200
4.7.2 Fourier Analysis of the Doppler Effect . . . 200
4.7.3 Use of Laplace Transformation and Its Inverse . . . 201
4.7.4 Use of the Wavelet Transformation . . . 202
References . . . 203

5 Statistical Analysis and Modeling of Data . . . 207
5.1 Basic Data Analysis . . . 207
5.1.1 Probability Distributions . . . 207
5.1.2 Moments of Distributions . . . 208
5.1.3 Uncertainties of Moments of Distributions . . . 209
5.2 Robust Statistics . . . 210
5.2.1 Hunting for Outliers . . . 212
5.2.2 M-Estimates of Location . . . 213
5.2.3 M-Estimates of Scale . . . 216
5.3 Statistical Tests . . . 217
5.3.1 Computing the Confidence Interval for the Sample Mean . 217
5.3.2 Comparing the Means of Two Samples with Equal Variances . . . 218
5.3.3 Comparing the Means of Two Samples with Different Variances . . . 219
5.3.4 Determining the Confidence Interval for the Sample Variance . . . 220
5.3.5 Comparing Two Sample Variances . . . 221
5.3.6 Comparing Histogrammed Data to a Known Distribution . 223
5.3.7 Comparing Two Sets of Histogrammed Data . . . 224
5.3.8 Comparing Non-histogrammed Data to a Continuous Distribution . . . 224
5.4 Correlation . . . 225
5.4.1 Linear Correlation . . . 225
5.4.2 Non-parametric Correlation . . . 226
5.5 Linear and Non-linear Regression . . . 227
5.5.1 Linear Regression . . . 228
5.5.2 Regression with Orthogonal Polynomials . . . 229
5.5.3 Linear Regression (Fitting a Straight Line) . . . 230

5.5.4 Linear Regression (Fitting a Straight Line) with Errors in Both Coordinates . . . 232
5.5.5 Fitting a Constant . . . 233
5.5.6 Generalized Linear Regression by Using SVD . . . 236
5.5.7 Robust Methods for One-Dimensional Regression . . . 237
5.5.8 Non-linear Regression . . . 239
5.6 Multiple Linear Regression . . . 240
5.6.1 The Basic Method . . . 240
5.6.2 Principal-Component Multiple Regression . . . 242
5.7 Principal-Component Analysis . . . 244
5.7.1 Principal Components by Diagonalizing the Covariance Matrix . . . 246
5.7.2 Standardization of Data for PCA . . . 248
5.7.3 Principal Components from the SVD of the Data Matrix . . . 249
5.7.4 Improvements of PCA: Non-linearity, Robustness . . . 249
5.8 Cluster Analysis ⋆ . . . 249
5.8.1 Hierarchical Clustering . . . 250
5.8.2 Partitioning Methods: k-Means . . . 253
5.8.3 Gaussian Mixture Clustering and the EM Algorithm . . . 256
5.8.4 Spectral Methods . . . 258
5.9 Linear Discriminant Analysis ⋆ . . . 259
5.9.1 Binary Classification . . . 259
5.9.2 Logistic Discriminant Analysis . . . 261
5.9.3 Assignment to Multiple Classes . . . 262
5.10 Canonical Correlation Analysis ⋆ . . . 263
5.11 Factor Analysis ⋆ . . . 265
5.11.1 Determining the Factors and Weights from the Covariance Matrix . . . 266
5.11.2 Standardization of Data and Robust Factor Analysis . . . 269
5.12 Problems . . . 270
5.12.1 Multiple Regression . . . 270
5.12.2 Nutritional Value of Food . . . 270
5.12.3 Discrimination of Radar Signals from Ionospheric Reflections . . . 271
5.12.4 Canonical Correlation Analysis of Objects in the CDFS Area . . . 271
References . . . 273

6 Modeling and Analysis of Time Series . . . 277
6.1 Random Variables . . . 278
6.1.1 Basic Definitions . . . 278
6.1.2 Generation of Random Numbers . . . 279
6.2 Random Processes . . . 280
6.2.1 Basic Definitions . . . 280
6.3 Stable Distributions and Random Walks . . . 283
6.3.1 Central Limit Theorem . . . 283

6.3.2 Stable Distributions . . . 284
6.3.3 Generalized Central Limit Theorem . . . 287
6.3.4 Discrete-Time Random Walks . . . 287
6.3.5 Continuous-Time Random Walks . . . 290
6.4 Markov Chains ⋆ . . . 292
6.4.1 Discrete-Time or Classical Markov Chains . . . 292
6.4.2 Continuous-Time Markov Chains . . . 297
6.5 Noise . . . 299
6.5.1 Types of Noise . . . 300
6.5.2 Generation of Noise . . . 302
6.6 Time Correlation and Auto-Correlation . . . 304
6.6.1 Sample Correlations of Signals . . . 306
6.6.2 Representation of Time Correlations . . . 308
6.6.3 Fast Computation of Discrete Sample Correlations . . . 308
6.7 Auto-Regression Analysis of Discrete-Time Signals ⋆ . . . 310
6.7.1 Auto-Regression (AR) Model . . . 311
6.7.2 Use of AR Models . . . 314
6.7.3 Estimate of the Fourier Spectrum . . . 316
6.8 Independent Component Analysis ⋆ . . . 319
6.8.1 Estimate of the Separation Matrix and the FastICA Algorithm . . . 321
6.8.2 The FastICA Algorithm . . . 322
6.8.3 Stabilization of the FastICA Algorithm . . . 323
6.9 Problems . . . 324
6.9.1 Logistic Map . . . 324
6.9.2 Diffusion and Chaos in the Standard Map . . . 326
6.9.3 Phase Transitions in the Two-Dimensional Ising Model . . . 328
6.9.4 Independent Component Analysis . . . 329
References . . . 331

7 Initial-Value Problems for ODE . . . 335
7.1 Evolution Equations . . . 335
7.2 Explicit Euler's Methods . . . 337
7.3 Explicit Methods of the Runge–Kutta Type . . . 339
7.4 Errors of Explicit Methods . . . 340
7.4.1 Discretization and Round-Off Errors . . . 341
7.4.2 Consistency, Convergence, Stability . . . 342
7.4.3 Richardson Extrapolation . . . 343
7.4.4 Embedded Methods . . . 344
7.4.5 Automatic Step-Size Control . . . 346
7.5 Stability of One-Step Methods . . . 347
7.6 Extrapolation Methods ⋆ . . . 349
7.7 Multi-Step Methods ⋆ . . . 351
7.7.1 Predictor–Corrector Methods . . . 353
7.7.2 Stability of Multi-Step Methods . . . 354
7.7.3 Backward Differentiation Methods . . . 356

7.8 Conservative Second-Order Equations . . . 357
7.8.1 Runge–Kutta–Nyström Methods . . . 358
7.8.2 Multi-Step Methods . . . 359
7.9 Implicit Single-Step Methods . . . 359
7.9.1 Solution by Newton's Iteration . . . 362
7.9.2 Rosenbrock Linearization . . . 363
7.10 Stiff Problems . . . 365
7.11 Implicit Multi-Step Methods ⋆ . . . 367
7.12 Geometric Integration ⋆ . . . 368
7.12.1 Preservation of Invariants . . . 368
7.12.2 Preservation of the Symplectic Structure . . . 372
7.12.3 Reversibility and Symmetry . . . 373
7.12.4 Modified Hamiltonians and Equations of Motion . . . 374
7.13 Lie-Series Integration ⋆ . . . 375
7.13.1 Taylor Expansion of the Trajectory . . . 376
7.14 Problems . . . 380
7.14.1 Time Dependence of Filament Temperature . . . 380
7.14.2 Oblique Projectile Motion with Drag Force and Wind . . . 380
7.14.3 Influence of Fossil Fuels on Atmospheric CO_2 Content . . . 381
7.14.4 Synchronization of Globally Coupled Oscillators . . . 383
7.14.5 Excitation of Muscle Fibers . . . 384
7.14.6 Restricted Three-Body Problem (Arenstorf Orbits) . . . 386
7.14.7 Lorenz System . . . 388
7.14.8 Sine Pendulum . . . 389
7.14.9 Charged Particles in Electric and Magnetic Fields . . . 390
7.14.10 Chaotic Scattering . . . 391
7.14.11 Hydrogen Burning in the pp I Chain . . . 392
7.14.12 Oregonator . . . 394
7.14.13 Kepler's Problem . . . 395
7.14.14 Northern Lights . . . 396
7.14.15 Galactic Dynamics . . . 397
References . . . 398

8 Boundary-Value Problems for ODE . . . 401
8.1 Difference Methods for Scalar Boundary-Value Problems . . . 402
8.1.1 Consistency, Stability, and Convergence . . . 404
8.1.2 Non-linear Scalar Boundary-Value Problems . . . 405
8.2 Difference Methods for Systems of Boundary-Value Problems . . . 408
8.2.1 Linear Systems . . . 411
8.2.2 Schemes of Higher Orders . . . 411
8.3 Shooting Methods . . . 413
8.3.1 Second-Order Linear Equations . . . 414
8.3.2 Systems of Linear Second-Order Equations . . . 416
8.3.3 Non-linear Second-Order Equations . . . 418
8.3.4 Systems of Non-linear Equations . . . 419

8.3.5 Multiple (Parallel) Shooting . . . 421
8.4 Asymptotic Discretization Schemes ⋆ . . . 424
8.4.1 Discretization . . . 426
8.5 Collocation Methods ⋆ . . . 429
8.5.1 Scalar Linear Second-Order Boundary-Value Problems . . . 430
8.5.2 Scalar Linear Boundary-Value Problems of Higher Orders . . . 432
8.5.3 Scalar Non-linear Boundary-Value Problems of Higher Orders . . . 436
8.5.4 Systems of Boundary-Value Problems . . . 438
8.6 Weighted-Residual Methods ⋆ . . . 439
8.7 Boundary-Value Problems with Eigenvalues . . . 441
8.7.1 Difference Methods . . . 443
8.7.2 Shooting Methods with Prüfer Transformation . . . 446
8.7.3 Pruess Method . . . 449
8.7.4 Singular Sturm–Liouville Problems . . . 452
8.7.5 Eigenvalue-Dependent Boundary Conditions . . . 453
8.8 Isospectral Problems ⋆ . . . 454
8.9 Problems . . . 455
8.9.1 Gelfand–Bratu Equation . . . 455
8.9.2 Measles Epidemic . . . 456
8.9.3 Diffusion-Reaction Kinetics in a Catalytic Pellet . . . 457
8.9.4 Deflection of a Beam with Inhomogeneous Elastic Modulus . . . 459
8.9.5 A Boundary-Layer Problem . . . 459
8.9.6 Small Oscillations of an Inhomogeneous String . . . 460
8.9.7 One-Dimensional Schrödinger Equation . . . 462
8.9.8 A Fourth-Order Eigenvalue Problem . . . 463
References . . . 464

9 Difference Methods for One-Dimensional PDE . . . 467
9.1 Discretization of the Differential Equation . . . 469
9.2 Discretization of Initial and Boundary Conditions . . . 471
9.3 Consistency ⋆ . . . 473
9.4 Implicit Schemes . . . 475
9.5 Stability and Convergence ⋆ . . . 476
9.5.1 Initial-Value Problems . . . 476
9.5.2 Initial-Boundary-Value Problems . . . 479
9.6 Energy Estimates and Theorems on Maxima ⋆ . . . 481
9.6.1 Energy Estimates . . . 481
9.6.2 Theorems on Maxima . . . 482
9.7 Higher-Order Schemes . . . 484
9.8 Hyperbolic Equations . . . 485
9.8.1 Explicit Schemes . . . 486
9.8.2 Implicit Schemes . . . 489
9.8.3 Wave Equation . . . 490

9.9 Non-linear Equations and Equations of Mixed Type ⋆ 491
9.10 Dispersion and Dissipation ⋆ 494
9.11 Systems of Hyperbolic and Parabolic PDE ⋆ 497
9.12 Conservation Laws and High-Resolution Schemes ⋆ 500
9.12.1 High-Resolution Schemes 502
9.12.2 Linear Problem $v_t + cv_x = 0$ 504
9.12.3 Non-linear Conservation Laws of the Form $v_t + [F(v)]_x = 0$ 505
9.13 Problems 505
9.13.1 Diffusion Equation 505
9.13.2 Initial-Boundary Value Problem for $v_t + cv_x = 0$ 506
9.13.3 Dirichlet Problem for a System of Non-linear Hyperbolic PDE 507
9.13.4 Second-Order and Fourth-Order Wave Equations 508
9.13.5 Burgers Equation 509
9.13.6 The Shock-Tube Problem 511
9.13.7 Korteweg–de Vries Equation 512
9.13.8 Non-stationary Schrödinger Equation 513
9.13.9 Non-stationary Cubic Schrödinger Equation 515
References 517

10 Difference Methods for PDE in Several Dimensions 519
10.1 Parabolic and Hyperbolic PDE 519
10.1.1 Parabolic Equations 519
10.1.2 Explicit Scheme 520
10.1.3 Crank–Nicolson Scheme 522
10.1.4 Alternating Direction Implicit Schemes 523
10.1.5 Three Space Dimensions 526
10.1.6 Hyperbolic Equations 527
10.1.7 Explicit Schemes 527
10.1.8 Schemes for Equations in the Form of Conservation Laws 528
10.1.9 Implicit and ADI Schemes 529
10.2 Elliptic PDE 530
10.2.1 Dirichlet Boundary Conditions 530
10.2.2 Neumann Boundary Conditions 532
10.2.3 Mixed Boundary Conditions 532
10.2.4 Relaxation Methods 532
10.2.5 Conjugate Gradient Methods 537
10.3 High-Resolution Schemes ⋆ 537
10.4 Physically Motivated Discretizations 540
10.4.1 Two-Dimensional Diffusion Equation in Polar Coordinates 542
10.4.2 Two-Dimensional Poisson Equation in Polar Coordinates 544

10.5 Boundary Element Method ⋆ . . . 545
10.6 Finite-Element Method ⋆ . . . 549
10.6.1 One Space Dimension . . . 549
10.6.2 Two Space Dimensions . . . 553
10.7 Mimetic Discretizations ⋆ . . . 557
10.8 Multi-Grid and Mesh-Free Methods ⋆ . . . 557
10.8.1 A Mesh-Free Method Based on Radial Basis Functions . 559
10.9 Problems . . . 560
10.9.1 Two-Dimensional Diffusion Equation . . . 560
10.9.2 Non-linear Diffusion Equation . . . 563
10.9.3 Two-Dimensional Poisson Equation . . . 565
10.9.4 High-Resolution Schemes for the Advection Equation . 567
10.9.5 Two-Dimensional Diffusion Equation in Polar Coordinates . . . 568
10.9.6 Two-Dimensional Poisson Equation in Polar Coordinates . . . 568
10.9.7 Finite-Element Method . . . 569
10.9.8 Boundary Element Method for the Two-Dimensional Laplace Equation . . . 570
References . . . 571

11 Spectral Methods for PDE . . . 575
11.1 Spectral Representation of Spatial Derivatives . . . 577
11.1.1 Fourier Spectral Derivatives . . . 577
11.1.2 Legendre Spectral Derivatives . . . 580
11.1.3 Chebyshev Spectral Derivatives . . . 581
11.1.4 Computing the Chebyshev Spectral Derivative by Fourier Transformation . . . 583
11.2 Galerkin Methods . . . 586
11.2.1 Fourier–Galerkin . . . 586
11.2.2 Legendre–Galerkin . . . 587
11.2.3 Chebyshev–Galerkin . . . 589
11.2.4 Two Space Dimensions . . . 591
11.2.5 Non-stationary Problems . . . 591
11.3 Tau Methods . . . 594
11.3.1 Stationary Problems . . . 594
11.3.2 Non-stationary Problems . . . 596
11.4 Collocation Methods . . . 597
11.4.1 Stationary Problems . . . 598
11.4.2 Non-stationary Problems . . . 599
11.4.3 Spectral Elements: Collocation with B-Splines . . . 600
11.5 Non-linear Equations . . . 601
11.6 Time Integration ⋆ . . . 605
11.7 Semi-Infinite and Infinite Definition Domains ⋆ . . . 606
11.8 Complex Geometries ⋆ . . . 607

11.9 Problems . . . 607
11.9.1 Galerkin Methods for the Helmholtz Equation . . . 607
11.9.2 Galerkin Methods for the Advection Equation . . . 608
11.9.3 Galerkin Method for the Diffusion Equation . . . 609
11.9.4 Galerkin Method for the Poisson Equation: Poiseuille Law . . . 611
11.9.5 Legendre Tau Method for the Poisson Equation . . . 613
11.9.6 Collocation Methods for the Diffusion Equation I . . . 614
11.9.7 Collocation Methods for the Diffusion Equation II . . . 616
11.9.8 Burgers Equation . . . 617
References . . . 619

Appendix A Mathematical Tools . . . 621
A.1 Asymptotic Notation . . . 621
A.2 The Norms in Spaces $L^p(\Omega)$ and $L^p_w(\Omega)$, $1 \leq p \leq \infty$ . . . 622
A.3 Discrete Vector Norms . . . 623
A.4 Matrix and Operator Norms . . . 625
A.5 Eigenvalues of Tridiagonal Matrices . . . 626
A.6 Singular Values of X and Eigenvalues of $X^{\mathrm{T}}X$ and XX^{T} . . . 627
A.7 The "Square Root" of a Matrix . . . 628
References . . . 628

Appendix B Standard Numerical Data Types . . . 629
B.1 Real Numbers in Floating-Point Arithmetic . . . 629
B.1.1 Combining Types with Different Precisions . . . 632
B.2 Integer Numbers . . . 633
B.3 (Almost) Arbitrary Precision . . . 634
References . . . 635

Appendix C Generation of Pseudorandom Numbers . . . 637
C.1 Uniform Generators: From Integers to Reals . . . 637
C.2 Transformations Between Distributions . . . 638
C.2.1 Discrete Distribution . . . 639
C.2.2 Continuous Distribution . . . 640
C.3 Random Number Generators and Tests of Their Reliability . . . 646
C.3.1 Linear Generators . . . 646
C.3.2 Non-linear Generators . . . 648
C.3.3 Using and Testing Generators . . . 648
References . . . 649

Appendix D Convergence Theorems for Iterative Methods . . . 651
D.1 General Theorems . . . 651
D.2 Theorems for the Newton–Raphson Method . . . 653
References . . . 654

Appendix E Numerical Integration . . . 655
E.1 Gauss Quadrature . . . 657
E.1.1 Gauss–Kronrod Quadrature . . . 658
E.1.2 Quadrature in Two Dimensions . . . 659
E.2 Integration of Rapidly Oscillating Functions . . . 660
E.2.1 Asymptotic Method . . . 660
E.2.2 Filon's Method . . . 662
E.3 Integration of Singular Functions . . . 664
References . . . 665

Appendix F Fixed Points and Stability ⋆ . . . 667
F.1 Linear Stability . . . 667
F.2 Spurious Fixed Points . . . 669
F.3 Non-linear Stability . . . 671
References . . . 673

Appendix G Construction of Symplectic Integrators ⋆ . . . 675
References . . . 680

Appendix H Transforming PDE to Systems of ODE: Two Warnings . . . 681
H.1 Diffusion Equation . . . 681
H.2 Advection Equation . . . 684
References . . . 686

Appendix I Numerical Libraries, Auxiliary Tools, and Languages . . . 687
I.1 Important Numerical Libraries . . . 687
I.2 Basics of Program Compilation . . . 690
I.3 Using Libraries in C/C++ and Fortran . . . 691
I.3.1 Solving Systems of Equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ by Using the GSL Library . . . 691
I.3.2 Solving the System $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ in C/C++ Language and Fortran Libraries . . . 692
I.3.3 Solving the System $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ in Fortran95 by Using a Fortran77 Library . . . 694
I.4 Auxiliary Tools . . . 694
I.5 Choosing the Programming Language . . . 696
References . . . 697

Appendix J Measuring Program Execution Times on Linux Systems . . . 699
References . . . 702

Index . . . 703

Chapter 1
Basics of Numerical Analysis

1.1 Introduction

An ever increasing amount of computational work is being relegated to computers, and often we almost blindly assume that the obtained results are correct. At the same time, we wish to accelerate individual computation steps and improve their accuracy. Numerical computations should therefore be approached with a good measure of skepticism. Above all, we should try to understand the meaning of the results and the precision of operations between numerical data.

A prudent choice of appropriate algorithms is essential (see, for example, [1, 2]). In their implementation, we should be aware that the compiler may have its own "will" and has no "clue" about mathematical physics. In order to learn more about the essence of the computation and its natural limitations, we strive to simplify complex operations and restrict the tasks of functions to smaller, well-defined domains. It also makes sense to measure the execution time of programs (see Appendix J): large fluctuations in these well measurable quantities without modifications in running conditions typically point to a poorly designed program or a lack of understanding of the underlying problem.

1.1.1 Finite-Precision Arithmetic

The key models for computation with real numbers in finite precision are the *floating-point* and *fixed-point arithmetic*. A real number x in floating-point arithmetic with base β is represented by the approximation

$$\mathrm{fl}(x) = \pm\left(d_0 + d_1\beta^{-1} + d_2\beta^{-2} + \cdots + d_{p-1}\beta^{-(p-1)}\right) \cdot \beta^e \equiv \pm d_0.d_1 \ldots d_{p-1} \cdot \beta^e,$$

where $\{d_i\}_{i=0}^{p-1}$, $d_i \in \{0, 1, \ldots, \beta - 1\}$, is a set of p integers and the exponent e is within $[e_{\min}, e_{\max}]$. The expression $m = d_0.d_1 \ldots d_{p-1}$ is called the *significand*

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9_1, © Springer-Verlag Berlin Heidelberg 2012

Table 1.1 The smallest and largest exponents and approximate values of some important numbers representable in single- and double-precision floating-point arithmetic in base two, according to the IEEE 754 standard. Only positive values are listed

Precision	Single ("`float`")	Double ("`double`")
$e_{\max}$	127	1023
$e_{\min} = 1 - e_{\max}$	-126	-1022
Smallest normal number	$\approx 1.18 \times 10^{-38}$	$\approx 2.23 \times 10^{-308}$
Largest normal number	$\approx 3.40 \times 10^{38}$	$\approx 1.80 \times 10^{308}$
Smallest representable number	$\approx 1.40 \times 10^{-45}$	$\approx 4.94 \times 10^{-324}$
Machine precision, ε_{M}	$\approx 1.19 \times 10^{-7}$	$\approx 2.22 \times 10^{-16}$
Format size	32 bits	64 bits

or *mantissa*, while $f = 0.d_1 \ldots d_{p-1}$ is its *fractional part*. Here we are mostly interested in binary numbers ($\beta = 2$) which can be described by the fractional part f alone if we introduce two classes of numbers. The first class contains *normal* numbers with $d_0 = 1$; these numbers are represented as $\mathrm{fl}(x) = 1.f \cdot 2^e$, while the number zero is defined separately as $\mathrm{fl}(0) = 1.0 \cdot 2^{e_{\min}-1}$. The second class contains *subnormal* numbers, for which $d_0 = 0$. Subnormal numbers fall in the range between the number zero and the smallest positive normal number $2^{e_{\min}}$. They can be represented in the form $\mathrm{fl}(x) = 0.f \cdot 2^{e_{\min}}$. Data types with single ("`float`") and double ("`double`") precision, as well as algorithms for computation of basic operations between them are defined by the IEEE 754 standard; see Table 1.1, the details in Appendix B, as well as [3, 4].

Computations in fixed-point arithmetic (in which numbers are represented by a *fixed* value of e) are faster than those in floating-point arithmetic, and become relevant when working with a restricted range of values. They becomes useful on very specific architectures where large speed and small memory consumption are crucial (for example, in GPS devices or CNC machining tools). In scientific and engineering work, floating-point arithmetic dominates.

The elementary binary operations between floating-point numbers are addition, subtraction, multiplication, and division. We denote these operations by

$$+ : x \oplus y, \qquad - : x \ominus y, \qquad \times : x \otimes y, \qquad / : x \oslash y.$$

Since floating-point numbers have finite precision, the results of the operations $x + y$, $x - y$, $x \times y$, and x/y, computed with exact values of x and y, are not identical to the results of the corresponding operations in finite-precision arithmetic, $x \oplus y$, $x \ominus y$, $x \otimes y$, and $x \oslash y$. One of the key properties of finite-precision arithmetic is the non-associativity of addition and multiplication,

$$x \oplus (y \oplus z) \neq (x \oplus y) \oplus z, \qquad x \otimes (y \otimes z) \neq (x \otimes y) \otimes z.$$

This has important consequences, as demonstrated by the following examples.

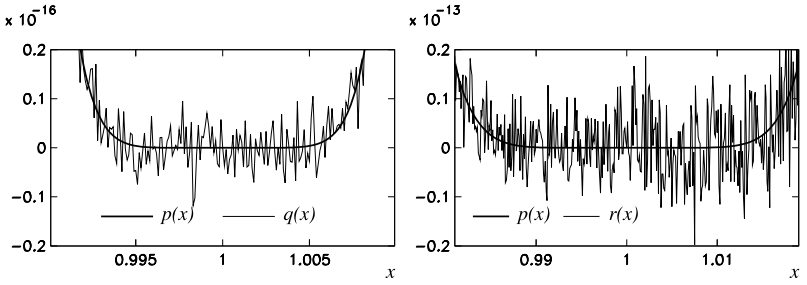


Fig. 1.1 Computation of $(1-x)^8$ in the vicinity of $x=1$ in double-precision floating-point arithmetic. [*Left*] Formula $p(x)=(1-x)^8$ by using `pow(1-x,8)` and formula $q(x)=1-8x+28x^2-\cdots$ by using simple multiplications of x. [*Right*] Computation of $r(x)$ with formula for $q(x)$, but by using `pow` functions for the powers

Example By writing a simple program in C or C++ you can convince yourself that in the case `float x=1e9; float y=-1e9; float z=1;` with the GNU compiler `c++` and option `-O2` you obtain $(\mathtt{x}\oplus\mathtt{y})\oplus\mathtt{z}=1$, while $\mathtt{x}\oplus(\mathtt{y}\oplus\mathtt{z})=0$. (Other compilers may behave differently.)

Example The effects of rounding errors can also be neatly demonstrated [5] by observing the results of three algebraically identical, but numerically different ways of computing the values of the polynomial $(1-x)^8$ in the vicinity of $x=1$. Let us denote the results of the three methods by $p(x)$, $q(x)$, and $r(x)$. We first compute

$$p(x)=(1-x)^8$$

by using the standard power function `pow(1-x, 8)` available in C or C++. The same polynomial can also be expanded as

$$q(x)=1-8x+28x^2-56x^3+70x^4-56x^5+28x^6-8x^7+x^8,$$

which we compute by simple multiplication, for example, $x^3=x\cdot x\cdot x$. Finally, we compute $r(x)$ just like $q(x)$, each term in a row, but evaluate the powers x^n by using the `pow(x, n)` function. Figure 1.1 shows the results of the three methods (be alert to the change of scale on the vertical axis).

Binary operations in floating-point arithmetic are thus performed with errors which depend on the arguments, the type of operation, the programming language, and the compiler. The precision of floating-point arithmetic, called the *machine precision* or *machine epsilon*, and denoted by ε_{M}, is defined as the difference between the representation of the number 1 and the representation of the nearest larger number. In single precision, we have $\varepsilon_{\mathrm{M}}=2^{-23}\approx 1.19\times 10^{-7}$, while in double precision $\varepsilon_{\mathrm{M}}=2^{-52}\approx 2.22\times 10^{-16}$. The precision of the arithmetic can also be defined as the largest ε for which $\mathrm{fl}(1+\varepsilon)=1$, but the value of ε obtained in this manner depends on the rounding method. The IEEE standard prescribes rounding to the

nearest representable result: in this case $\varepsilon \approx \varepsilon_{\mathrm{M}}/2$, which is known as *unit roundoff*. For arbitrary data or result of a basic binary operation between normal numbers we have

$$\big|\mathrm{fl}(x) - x\big| \le \frac{\varepsilon_{\mathrm{M}}}{2}|x|, \qquad \big|\mathrm{fl}(x \circ y) - x \circ y\big| \le \frac{\varepsilon_{\mathrm{M}}}{2}|x \circ y|,$$

where $\circ$ denotes a basic binary operation. The floating-point numbers therefore behave as exact values perturbed by a relative error:

$$\mathrm{fl}(x) = x(1+\delta), \quad |\delta| \le \frac{\varepsilon_{\mathrm{M}}}{2}.$$

When operations are performed between such numbers, a seemingly small perturbation δ can greatly increase the error of the result, leading to a *loss of significant digits* (one or more incorrect digits). As an example, we subtract two different numbers $x = \mathrm{fl}(x)(1+\delta_1)$ and $y = \mathrm{fl}(y)(1+\delta_2)$:

$$x - y = \big(\mathrm{fl}(x) - \mathrm{fl}(y)\big)(1+h), \quad h = \frac{\mathrm{fl}(x)\delta_1 - \mathrm{fl}(y)\delta_2}{\mathrm{fl}(x) - \mathrm{fl}(y)}.$$

The relative error is bounded by $|h| \le \varepsilon_{\mathrm{M}} \max\{|x|, |y|\}/|x-y|$ and can become very large if x and y are nearby.

Considering the effects of small perturbations also helps us analyze the relative errors of other basic operations and the precision of more complex algorithms. Special libraries like GMP [7] do allow for computations with almost arbitrary precision, but let us not use this possibility absent-mindedly: do not try to out-smart an unstable algorithm solely by increasing the arithmetic precision!

For a detailed description of how floating- or fixed-point arithmetic behave in typical algorithms, see [4, 8–10]; for details on floating-point data format according to the IEEE 754 standard see Appendix B. We conclude this section by advices for a stable and precise solution of a few seemingly trivial tasks [4, 6].

Branch Statements In programs, we tend to use simple branch statements like

$$\texttt{if } \big(|x_k - x_{k-1}| < \varepsilon_{\mathrm{abs}}\big) \texttt{ then } \ldots, \tag{1.1}$$

where, for example, $\varepsilon_{\mathrm{abs}} = 10^{-6}$. But for $\mathrm{fl}(x_k) = 10^{50}$ the nearest representable number in double precision is $\varepsilon_{\mathrm{M}}\,\mathrm{fl}(x_k) \approx 10^{34}$ away. By using an algorithm which returns x_k in double precision, the condition (1.1) will never be met, except in the trivial case of equal x_k and x_{k-1} (for example, due to rounding to zero). It is much better to use

$$\texttt{if } \big(|x_k - x_{k-1}| < \varepsilon_{\mathrm{abs}} + \varepsilon_{\mathrm{rel}} \max\{x_k, x_{k-1}\}\big) \texttt{ then } \ldots,$$

where $\varepsilon_{\mathrm{rel}} \approx \varepsilon_{\mathrm{M}}$. Similarly, we repeatedly encounter sign-change tests like

$$\texttt{if } (f_k f_{k-1} < 0) \texttt{ then } \ldots. \tag{1.2}$$

If $f_k = 10^{-200}$ and $f_{k-1} = -10^{-200}$, we expect $f_k f_{k-1} = -10^{-400} < 0$, but in double precision we get an underflow $\mathrm{fl}(-10^{-400}) = 0$ and the statement (1.2) goes the wrong way. If $f_k = 10^{200}$ and $f_{k-1} = -10^{200}$, we get an overflow. Let us rather check only the sign of the arguments! In C or C++ this can be accomplished by using the function `template <typename T> int sign(const T & val){ return int((val>0) - (val<0)); }`, and then comparing

$$\texttt{if}\ \big(\texttt{sign}(f_k) \neq \texttt{sign}(f_{k-1})\big)\ \texttt{then}\ \ldots.$$

Roots of the Quadratic Equation The quadratic equation $ax^2 + bx + c = 0$ has the roots

$$x_\pm = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{2c}{-b \mp \sqrt{b^2 - 4ac}},$$

related by $x_+ x_- = c/a$. In most cases the absolute value of one of the roots is larger than the other and is computed with a larger relative precision in floating-point arithmetic. Once b and $\sqrt{b^2 - 4ac}$ are known, it is therefore preferable from the numerical viewpoint to first compute the larger root $x_{\max}$ and then use $x_{\min} = c/(a x_{\max})$ to compute the smaller one. In a similar spirit we compute $\sqrt{1 + x^2} - 1$ for $|x| \ll 1$ where subtraction and potential loss of significant digits can be avoided by rewriting the expression as

$$\sqrt{1 + x^2} - 1 = \frac{(\sqrt{1 + x^2} - 1)(\sqrt{1 + x^2} + 1)}{\sqrt{1 + x^2} + 1} = \frac{x^2}{\sqrt{1 + x^2} + 1}.$$

Area of Triangle Heron's formula $S = \sqrt{d(d-a)(d-b)(d-c)}$ for the area of a triangle with sides $a \geq b \geq c$, where $d = (a + b + c)/2$, is very sensitive to round-off errors, in particular when one of the angles is larger than 90° and $a \approx b + c$. In such cases it is advisable to use the following formula which is accurate to $\approx 10\varepsilon_{\mathrm{M}}$:

$$S = \frac{1}{4}\sqrt{\big[a + (b + c)\big]\big[c - (a - b)\big]\big[c + (a - b)\big]\big[a + (b - c)\big]}.$$

Magnitude of Complex Number, Ratio of Complex Numbers The magnitude (absolute value) of a complex number $z = x + \mathrm{i}y$ is $|z| = (x^2 + y^2)^{1/2}$. Squaring and addition of large numbers, which both may lead to overflows, can be avoided by using the formula

$$|z| = \begin{cases} |x|\sqrt{1 + (|y|/|x|)^2}; & 0 < |y| < |x|, \\ |y|\sqrt{1 + (|x|/|y|)^2}; & 0 < |x| < |y|, \\ |x|\sqrt{2}; & \text{otherwise.} \end{cases}$$

We can exploit a similar trick to avoid overflows when computing the ratio of complex numbers $(a + \mathrm{i}b)/(c + \mathrm{i}d) = (ac + bd)/(c^2 + d^2) + \mathrm{i}(bc - ad)/(c^2 + d^2)$. If

this formula is applied directly, overflow may occur even though the correct result is within the allowed range. A safer way to compute the ratio is

$$\frac{a+\mathrm{i}b}{c+\mathrm{i}d} = \begin{cases} \frac{a+b(d/c)}{c+d(d/c)} + \mathrm{i}\frac{b-a(d/c)}{c+d(d/c)}; & |d| < |c|, \\ \frac{b+a(c/d)}{d+c(c/d)} - \mathrm{i}\frac{a-b(c/d)}{d+c(c/d)}; & |d| \geq |c|. \end{cases}$$

Natural Logarithm To compute $\log(1+x)$ at $0 \leq x < 3/4$ we recommend

$$\log(1+x) = \begin{cases} x; & 1 \oplus x = 1, \\ \frac{x\log(1+x)}{(1+x)-1}; & \text{otherwise}, \end{cases} \tag{1.3}$$

which has an error smaller than $5\varepsilon_{\mathrm{M}}$. Such a precise calculation finds its uses in economics for computation of interest rates where wrong results literally cost money. Let us assume we have some funds A and a small interest rate x for a short period of time. After n periods we have $A' = A(1+x)^n$. If $x \ll 1$, errors can accumulate in computing A' for $n \gg 1$. It is preferable to use the formula $A' = A\exp(n\log(1+x))$ and resort to (1.3).

Average of Two Numbers Even a simple expression like the arithmetic mean of two floating-point numbers, $x = (a+b)/2$, may overflow, and one should use $x = a + (b-a)/2$ or $a/2 + b/2$ instead.

1.2 Approximation of Expressions

Approximation is one of the key concepts of numerical analysis [11, 12]. In this book we only refer to approximations of scalar functions and expressions involving operators, and therefore only discuss these two examples.

1.2.1 Optimal (Minimax) and Almost Optimal Approximations

The optimal approximation of degree n of a continuous function f on the interval $[a,b]$ is defined as the polynomial $p_n^\star$ for which $E_n^\star \in \mathbb{R}$ exists such that

$$E_n^\star = \max_{a\leq x\leq b} \left|p_n^\star(x) - f(x)\right| \leq \max_{a\leq x\leq b} \left|p_n(x) - f(x)\right|, \tag{1.4}$$

where p_n is any polynomial of degree n. We are therefore seeking a polynomial $p_n^\star$ that minimizes the maximal error with respect to the function f. Such a polynomial represents an *optimal* or *minimax approximation*. The curve $p_8^\star$ in Fig. 1.2 (left) is the optimal approximation of the function $\mathrm{e}^x \sin(3\pi x)$ by a polynomial of degree eight ($n = 8$), while the corresponding curve in the right panel is the error of the approximation.

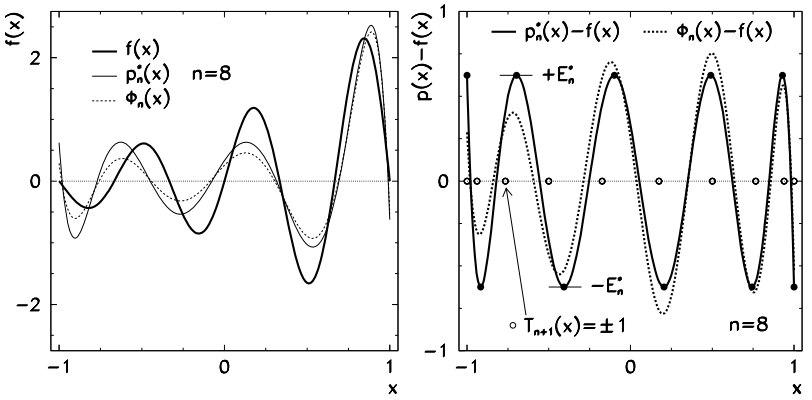


Fig. 1.2 Approximation of $f(x) = \mathrm{e}^x \sin(3\pi x)$ by an optimal polynomial and by a Chebyshev expansion. [*Left*] The function f, the optimal polynomial $p_8^\star$ of degree eight, and the expansion in terms of Chebyshev polynomials $\phi_8(x) = \sum_{k=0}^{8} \hat{\tau}_k T_k(x)$. [*Right*] Error of the approximations. The *symbols* • denote $(n+2)$ characteristic points at which the absolute value of the error $\pm E_n^\star = \pm|p_n^\star - f|$ is maximal, and between which the error changes its sign $(n+1)$-times. The *symbols* ∘ denote the points at which $T_{n+1}(x) = \pm 1$

The optimal polynomial approximation of a continuous function f on $[a, b]$ can be computed by the iterative Remes algorithm (see e.g. [13]). The basis for this procedure is the Borel equi-oscillation theorem which states that a degree-n polynomial $p_n^\star$ is the optimal approximation of f precisely when $(n+2)$ distinct points x_i exist, arranged as $a \le x_0 < x_1 < \cdots < x_{n+1} \le b$, for which

$$p_n^\star(x_i) - f(x_i) = \lambda(-1)^i E_n^\star, \quad i = 0, 1, \ldots, n+1,$$

and where λ has a fixed value of $+1$ or -1. The extremes $\pm E_n^\star$ of the error of the optimal approximation $p_n^\star$ therefore occur at the points x_i with the signs flipping back and forth (Fig. 1.2 (right)). This implies that the coefficients of $p_n(x) = \sum_{j=0}^{n} a_j x^j$ and the parameter E_n must fulfill the system of equations

$$\sum_{j=0}^{n} a_j x_i^j - f(x_i) = (-1)^i E_n, \quad i = 0, 1, \ldots, n+1, \tag{1.5}$$

$$\sum_{j=0}^{n} a_j x_i^j - f(x_i) = \text{extreme}. \tag{1.6}$$

If f is differentiable, we can rewrite the last equation as

$$\sum_{j=1}^{n} j a_j x_i^{j-1} - f'(x_i) = 0. \tag{1.7}$$

The iteration is started with an initial guess for x_i. We show in the following that a very good choice for x_i are the points at which the Chebyshev polynomials T_{n+1} are equal to ± 1, so

$$x_i = \cos\left(\frac{i\pi}{n+1}\right), \quad i = 0, 1, \ldots, n+1.$$

By solving the linear system (1.5) we obtain $n+1$ coefficients a_j and the parameter E_n, which are the first approximations for the coefficients of the optimal polynomial and the maximum error of $p - f$. These parameters are then used to solve the system (1.6) or (1.7): we search for points x_i at which the error $p - f$ has an extreme. This is the most difficult step in the procedure, especially when it needs to be automated. The new points x_i are then again used in (1.5) and we repeat the procedure until the difference between the consecutive values of E_n drops below a desired tolerance. At the end of the iteration the $\{a_j\}_{j=0}^n$ are the coefficients of the optimal polynomial $p_n^\star$, and $E_n = E_n^\star$.

Actually, the power basis $\{1, x, x^2, \ldots\}$ of $p_n^\star$ is a bad choice. The condition number of the matrix corresponding to the system (1.5) deteriorates exponentially with increasing n. Instead of polynomials, rational functions $p(x) = P_n(x)/Q_m(x)$ can be plugged into the Remes procedure, but the system of equations for the $(n+1)+(m+1)$ coefficients and the errors E becomes non-linear; it can be solved by linearization [14]. In the context of high-order optimal approximations, Chebyshev polynomials are also an attractive option [15].

Even without the Remes algorithm, the Chebyshev polynomials lead to an almost identical goal. Instead of searching for the optimal polynomial $p_n^\star$, we may be satisfied by finding an approximation q_n for which

$$\varepsilon = \max_{a \le x \le b} \left| p_n^\star(x) - q_n(x) \right|$$

with some small ε. Such an approximation is called an *almost optimal* or *near-minimax approximation*. Often it is much easier to find than the true optimal approximation. The most important among them is the approximation by Chebyshev polynomials

$$f(x) \approx \phi_n(x) = \sum_{k=0}^{n} \hat{\tau}_k T_k(x), \tag{1.8}$$

where the coefficients $\hat{\tau}_k$ are given in (4.39). We switch between the intervals $[a, b]$ and $[-1, 1]$ (on which Chebyshev polynomials are defined) by the transformations

$$t \mapsto x = 2\frac{t-a}{b-a} - 1, \qquad x \mapsto t = \frac{1-x}{2}\, a + \frac{1+x}{2}\, b.$$

The summation of the series (1.8) can be terminated at the same n that would be used in the optimal approximation. The error due to the series truncation is then determined by the term $\hat{\tau}_{n+1} T_{n+1}(x)$, which has interchanging extreme values of

$\pm\hat{\tau}_{n+1}$ at $(n+2)$ points on $[-1,1]$ and closely resembles the genuine optimal approximation (see Fig. 1.2 (left and right)).

Chebyshev polynomials have many pleasing and useful properties which we exploit heavily in function and signal transformations (Chap. 4) and spectral methods for partial differential equations (Chap. 11). In the context of optimal approximations, an important property of Chebyshev polynomials is their point-wise orthogonality (4.38). There exists a whole class of polynomials which are orthogonal on a discrete set of points, but only Chebyshev polynomials on $[-1,1]$ oscillate uniformly and can be generated by so few numerical operations (see recurrence (4.37)).

The shape of the optimal approximation depends on the norm in which its error is measured. Equation (1.4) defines the *uniform* optimal approximation in the "max"-norm $\|\cdot\|_\infty$. The computation of the optimal approximation for an arbitrary function, in particular if it is given only at specific points, can be cumbersome. The main nuisance is the large sensitivity of the Remes algorithm to individual values $f(x_i)$ that strongly deviate from the average (e.g. outliers in discrete-time signals). In physics we often resort to the optimal approximation in the Euclidean norm (A.2). In this case, the optimal approximation of the function f on $[a,b]$ is a function $p^\star$ for which $E^\star \in \mathbb{R}$ exists such that

$$E^\star = \int_a^b \big[p^\star(x) - f(x)\big]^2 \,\mathrm{d}x \le \int_a^b \big[p(x) - f(x)\big]^2 \,\mathrm{d}x \tag{1.9}$$

for any polynomial p of degree n (compare this expression to (1.4)). The form (1.9) is less sensitive to outliers mentioned above. We are of course referring to the method of least squares which we most often encounter in its discrete form, where we search for

$$\min_{p\in P_n} \sum_i \big[p(x_i) - f(x_i)\big]^2.$$

1.2.2 Rational (Padé) Approximation

Suppose that, on some interval, we wish to effectively approximate a function f possessing a power expansion

$$f(z) = \sum_{k=0}^{\infty} c_k z^k. \tag{1.10}$$

A Padé approximation of f is a rational function with a numerator of degree L and denominator of degree M, determined such that its power expansion matches the series (1.10) up to including the power $L+M$. In other words, if we can find a polynomial P_L of degree L and Q_M of degree M such that

$$f(z) = \frac{P_L(z)}{Q_M(z)} + \mathcal{O}\big(z^{L+M+1}\big), \quad Q_M(0) = 1,$$

then

$$[L/M]_f(z) = \frac{P_L(z)}{Q_M(z)} = \frac{a_0 + a_1 z + a_2 z^2 + \cdots + a_L z^L}{b_0 + b_1 z + b_2 z^2 + \cdots + b_M z^M}, \quad b_0 = 1,$$

defines a Padé approximation of order (L, M) of the function f. The coefficients a_k and b_k can be determined by equating the approximation $[L/M]_f$ with the power series for f and reading off the coefficients of the same powers of x:

$$\begin{aligned}&\big(b_0 + b_1 z + \cdots + b_M z^M\big)(c_0 + c_1 z + \cdots) \\ &\quad = a_0 + a_1 z + \cdots + a_L z^L + \mathcal{O}\big(z^{L+M+1}\big).\end{aligned}$$

By comparing the terms with powers $z^{L+1}, z^{L+2}, \ldots, z^{L+M}$ we obtain a system of equations for the coefficients b_k of the denominator Q_M, while by comparing terms with powers $z^0, z^1, \ldots, z^L$ we obtain explicit equations for the coefficients a_k of the numerator P_L. For example, with $L = M = 3$, we get

$$\begin{pmatrix} c_3 & c_2 & c_1 \\ c_4 & c_3 & c_2 \\ c_5 & c_4 & c_3 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} -c_4 \\ -c_5 \\ -c_6 \end{pmatrix}, \qquad \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} c_0 & 0 & 0 & 0 \\ c_1 & c_0 & 0 & 0 \\ c_2 & c_1 & c_0 & 0 \\ c_3 & c_2 & c_1 & c_0 \end{pmatrix} \begin{pmatrix} 1 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}.$$

Since $b_0 = 1$, the first line of the equation on the right tells us that the zeroth coefficient of P_L is equal to the zeroth coefficient of the power expansion of the function, $a_0 = c_0$. The bulk of the work is hidden in the matrix system on the left. Large Padé systems of this type are solved by robust algorithms like Gauss elimination with complete pivoting because, in most cases, we are interested in relatively low degrees L and M and because accuracy takes precedence over speed. In the sense of properties mentioned in the following, *diagonal Padé approximations*, in which $L = M$, are the most efficient.

Example (Adapted from [16], p. 4.) The function

$$f(z) = \sqrt{\frac{1 + z/2}{1 + 2z}} = \sum_{k=0}^{\infty} c_k z^k = 1 - \frac{3}{4} z + \frac{39}{32} z^2 - \frac{267}{128} z^3 + \frac{7563}{2048} z^4 - \cdots, \quad (1.11)$$

has the first two diagonal Padé approximations

$$[1/1]_f(z) = \frac{1 + \frac{7}{8} z}{1 + \frac{13}{8} z}, \qquad [2/2]_f(z) = \frac{1 + \frac{17}{8} z + \frac{61}{64} z^2}{1 + \frac{23}{8} z + \frac{121}{64} z^2}.$$

(Compute them!) The comparison of two power expansions and these Padé approximations is shown in Fig. 1.3, revealing a most delightful property: if some power series converges to a function with a convergence radius ρ, that is, for all $|z| < \rho$ and $0 < \rho < \infty$, then an appropriately chosen Padé approximation converges to f

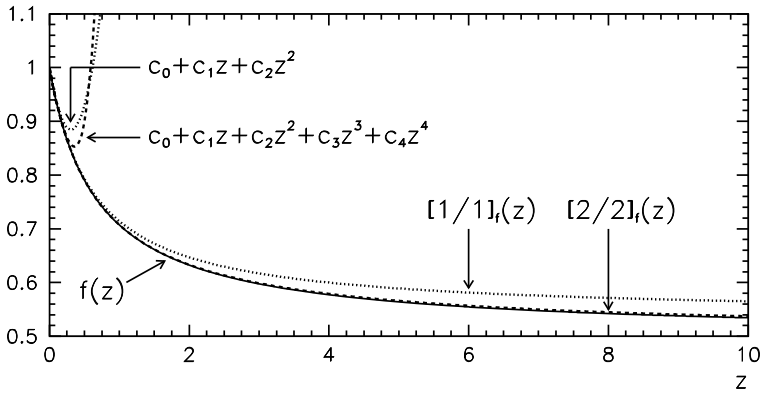


Fig. 1.3 Comparison of power expansions of degree two and four for the function (1.11) and the diagonal Padé approximations $[1/1]_f$ and $[2/2]_f$. The curves for f and $[2/2]_f$ are barely distinguishable in this plot

for all $z \in \mathcal{R}$, where the domain $\mathcal{R}$ is larger than the domain defined by $|z| < \rho$. The function f in example (1.11) has the limit $\lim_{z\to\infty} f(z) = 1/2$ and its power series has a convergence radius of only $\rho = 1/2$. On the other hand, both lowest-order diagonal Padé approximations are stable at infinity. Moreover, when the order of the approximation is increased, the correct limit $1/2$ is approached rapidly: $\lim_{z\to\infty}[1/1]_f(z) = 7/13 \approx 0.5385$ and $\lim_{z\to\infty}[2/2]_f(z) = 61/121 \approx 0.5041$. In other words, the Padé approximation tells us something about how the function behaves outside of the convergence radius of its power series, and ensures better asymptotics.

Example Take the function $f(z) = \mathrm{e}^z$, for which the lowest-order Padé approximations are listed in Table H.1. This table shows the Padé approximation in the context of solving partial differential equations and also summarizes the leading error terms. The Padé approach is also fruitful in approximating the time evolution of Hamiltonian operators: see example (1.17) below.

In certain cases the Padé approximation of a function f does not exist. For example, for $f(z) = 1 + z^2$ we would attempt to compute the Padé coefficients such that $(a_0 + a_1 z)/(b_0 + b_1 z) = 1 + z^2 + \mathcal{O}(z^3)$, and by comparing coefficients with equal powers of z we would get $a_0 = b_0$, $a_1 = b_1$ and $a_0 = 0$. This would lead to $[1/1]_f = (0 + a_1 z)/(0 + b_1 z) = 1 \neq 1 + z^2 + \mathcal{O}(z^3)$. One can encounter problems if the denominator of the Padé approximation is zero when $z = 0$. For details, see [16].

Padé approximations have numerous important transformation and invariance properties, of which *duality* and *unitarity* are the most relevant. Duality connects the Padé approximations for reciprocal functions:

$$g(z) = \big\{f(z)\big\}^{-1} \quad \text{and} \quad f(0) \neq 0 \quad \Leftrightarrow \quad [L/M]_g(z) = \big\{[M/L]_f(z)\big\}^{-1} \; \forall L, M.$$

In physical applications, unitarity is even more important, in particular in the theory of scattering matrices. Assume that $f(z) = \sum_{k=0}^{\infty} c_k z^k$ is unitary, so that $f(z) f^*(z) = 1$, and that $[M/M]_f$ is its diagonal Padé approximation. Then we also have

$$[M/M]_f(z)[M/M]_f^*(z) = 1,$$

where the symbol $*$ denotes complex conjugation of the coefficients of the approximation (not of the argument z). In approximations of time evolution of Hamiltonian operators (see (1.16)), preservation of unitarity is crucial.

1.2.3 Summation of Series by Using Padé Approximations (Wynn's ϵ-Algorithm)

Summation of series is the topic of Sect. 1.4, but here we wish to discuss an important method based on the Padé approximation. We have just witnessed how this approximation can be used to extend the convergence radius of a power series. But this very same approximation can be used to accelerate the summation of a series within its convergence radius. Assume that a series (in the region where it converges) can be approximated by a rational function f which is analytical in this region and has the form

$$f(z) = c_0 + c_1 z + c_2 z^2 + \cdots = \frac{P_L(z)}{Q_M(z)} \equiv [L/M]_f(z),$$

where $P_L(z) = a_0 + a_1 z + \cdots + a_L z^L$ and $Q_M(z) = 1 + b_1 z + \cdots + b_M z^M$. An efficient procedure to compute the diagonal approximations at a given z can be obtained by transformations between the values of $[L/M]_f(z)$ for different L and M. A connection between these approximations can be established which is known as the Wynn's ϵ-algorithm [17]. It can be written as a recurrence,

$$\epsilon(n, m+1) = \epsilon(n+1, m-1) + \frac{1}{\epsilon(n+1, m) - \epsilon(n, m)}. \tag{1.12}$$

We start the recurrence with $\epsilon(n, -1) = 0$ for $\forall n \geq 0$ and $\epsilon(n, 0)$ which contain the partial sums

$$\epsilon(n, 0) = S_n = \sum_{k=0}^{n} c_k z^k, \quad n \geq 0.$$

The initial state of the algorithm is given by the first two columns of the $\epsilon(n, m)$ table:

$$\begin{array}{cccccc}
\epsilon(0,-1)=0 & \underline{\epsilon(0,0)=S_0} & \epsilon(0,1) & \underline{\epsilon(0,2)} & \epsilon(0,3) & \underline{\epsilon(0,4)} & \cdots \\
\epsilon(1,-1)=0 & \epsilon(1,0)=S_1 & \epsilon(1,1) & \epsilon(1,2) & \epsilon(1,3) & \epsilon(1,4) & \cdots \\
\epsilon(2,-1)=0 & \epsilon(2,0)=S_2 & \epsilon(2,1) & \epsilon(2,2) & \epsilon(2,3) & \epsilon(2,4) & \cdots \\
\epsilon(3,-1)=0 & \epsilon(3,0)=S_3 & \epsilon(3,1) & \epsilon(3,2) & \epsilon(3,3) & \epsilon(3,4) & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots &
\end{array}$$

By repeated use of (1.12) we obtain all other entries, and these are related to Padé approximations of various orders. The diagonal approximations $[p/p]_f(z)$ at a given z can be read off from the underlined entries with even indices m:

$$[p/p]_f(z) = \epsilon(0, 2p), \quad p = 0, 1, \ldots.$$

We stop the algorithm when the value of the denominator of the fraction in (1.12) drops below a prescribed tolerance; since the convergence is very fast, this can be set to $\approx\varepsilon_{\mathrm{M}}$. Some authors recommend the Wynn procedure as the best general algorithm to accelerate the summation of any slowly converging series [18].

As an exercise, compare the extremely slow convergence of the partial sums $S_n = \sum_{k=0}^{n}(-1)^k/(k+1)$ with $\lim_{n\to\infty} S_n = \log 2$ (second column of the Wynn table) to the much faster convergence of the entries $\epsilon(0,0), \epsilon(0,2), \epsilon(0,4), \ldots$ in the first row of the table!

Example The Wynn procedure also enables us to evaluate asymptotic (divergent) series in the sense of Sect. 1.3.2. The exponential integral $f(z) = \mathrm{Ei}(z)$ for large positive z can be represented by the asymptotic series

$$\mathrm{Ei}(z) = \int_z^\infty \frac{\mathrm{e}^{-t}}{t}\,\mathrm{d}t \sim \frac{\mathrm{e}^{-z}}{z}\left[1 - \frac{1!}{z} + \frac{2!}{z^2} - \frac{3!}{z^3} + \cdots\right], \quad z \to \infty. \tag{1.13}$$

The partial sums of (1.13),

$$S_n = \frac{\mathrm{e}^{-z}}{z}\sum_{k=0}^{n}\frac{k!}{(-z)^k}, \quad n \geq 0, \tag{1.14}$$

do not converge for any z, as the upper two curves in Fig. 1.4 (left) indicate for $z = 2$ and $z = 4$. (At even higher z, the minimum of the error would just shift to the right and downwards.) The lower two curves in the same Figure show the error when the series is evaluated by the Wynn algorithm.

Example A physically more convincing use of the Wynn's method for divergent series is described in Problem 1.5.5. There, the function f represents the Coulomb scattering amplitude with a divergent power series in $z = \cos\theta$:

$$f(\theta) = \frac{1}{2\mathrm{i}k}\sum_{l=0}^{\infty}(2l+1)P_l(\cos\theta)\left(\mathrm{e}^{2\mathrm{i}\sigma_l} - 1\right). \tag{1.15}$$

The errors of the partial sums of (1.15) and of the Wynn approximations with respect to the exact amplitude are shown in Fig. 1.4 (right).

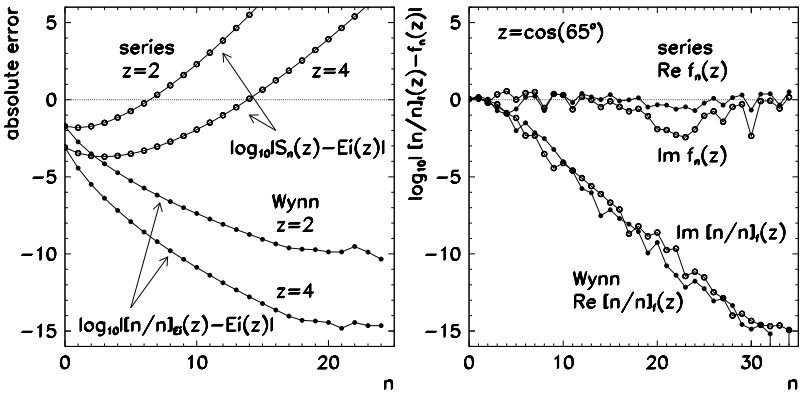


Fig. 1.4 Evaluation of asymptotic series by the Wynn algorithm. [*Left*] Errors of the partial sums (1.14) of (1.13) and of the Padé approximation $[n/n]_{\mathrm{Ei}}$ with respect to the exact values Ei(2) and Ei(4) as a function of the index n. [*Right*] Errors of the partial sums of (1.15) and of $[n/n]_f$ with respect to the exact amplitude (1.71) at $\theta = 65°$

1.2.4 Approximation of the Evolution Operator for a Hamiltonian System

The time evolution of a quantum Hamiltonian system from time t to time $t + \Delta t$ (for example, of a wave-function $\Psi(x,t)$ governed by the non-stationary Schrödinger equation) is given by

$$\Psi(x, t + \Delta t) = \mathrm{e}^{-\mathrm{i}\,H\Delta t/\hbar}\Psi(x,t). \tag{1.16}$$

In difference methods for partial differential equations (Chap. 9) we prefer to approximate the evolution operator $\exp[-\mathrm{i}H\Delta t/\hbar]$ by low-order unitary approximations. If the order of the approximation is too low, the solution can become noisy and meaningless even on short time scales. If H does not depend on time, a good way to improve the accuracy of the difference method in its temporal part is to approximate the exponential operator by a diagonal Padé approximation

$$\mathrm{e}^z = \sum_{k=0}^{\infty} c_k z^k = \frac{1 + a_1 z + \cdots + a_M z^M}{1 + b_1 z + \cdots + b_M z^M} + \mathcal{O}\big(z^{2M+1}\big), \tag{1.17}$$

where $c_k = 1/k!$, and the coefficients a_m and b_m are complex in general. The coefficients at a chosen order M are computed by the procedure described on p. 10. The numerator and the denominator of (1.17) can then be factorized as [16]

$$\mathrm{e}^z = \prod_{s=1}^{M}\left(\frac{1 - z/z_s^{(M)}}{1 + z/z_s^{*(M)}}\right) + \mathcal{O}\big(z^{2M+1}\big), \tag{1.18}$$

where $z_s^{(M)}$ are the zeros of the numerator, and $z_s^{*(M)}$ the zeros of the denominator (which are just their complex conjugates). For example, these zeros up to $M = 3$, in double precision, are

$$\begin{aligned} z_1^{(1)} &= -2, \\ z_{1,2}^{(2)} &= -3 \pm \mathrm{i}\sqrt{3}, \\ z_1^{(3)} &= -4.6443707092521712, \\ z_{2,3}^{(3)} &= -3.6778146453739144 \pm \mathrm{i}3.5087619195674433. \end{aligned}$$

At the lowest order ($M = 1$) the expression (1.18) with $z = -\mathrm{i}H\Delta t/\hbar$ simplifies to

$$\mathrm{e}^{-\mathrm{i}H\Delta t/\hbar} = \frac{1 - \frac{1}{2}\mathrm{i}H\Delta t/\hbar}{1 + \frac{1}{2}\mathrm{i}\,H\Delta t/\hbar} + \mathcal{O}\big(\Delta t^3\big).$$

In the context of (1.16) this can be read as

$$\left[1 + \frac{\mathrm{i}H\Delta t}{2\hbar}\right]\Psi(x, t + \Delta t) = \left[1 - \frac{\mathrm{i}H\Delta t}{2\hbar}\right]\Psi(x, t)$$

(see Problem 9.13.8). At higher orders, we get a product operator

$$\mathrm{e}^{-\mathrm{i}H\Delta t/\hbar} \approx \prod_{s=1}^{M} E_s^{(M)}, \quad E_s^{(M)} = \frac{1 + (\mathrm{i}\,H\Delta t/\hbar)/z_s^{(M)}}{1 - (\mathrm{i}\,H\Delta t/\hbar)/z_s^{*(M)}} \equiv \frac{R_s^{(M)}}{L_s^{(M)}},$$

which allows us to compute the time evolution as

$$\Psi(x, t_{n+1}) = E_M^{(M)} \cdots E_2^{(M)} E_1^{(M)} \Psi(x, t_n), \quad t_{n+1} - t_n = \Delta t.$$

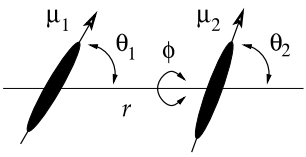
In a practical implementation, this implies a sequence of steps

$$\begin{aligned} L_1^{(M)}\Psi(x, t_{n+1/M}) &= R_1^{(M)}\Psi(x, t_n), \\ L_2^{(M)}\Psi(x, t_{n+2/M}) &= R_2^{(M)}\Psi(x, t_{n+1/M}), \\ &\cdots \quad \cdots \\ L_M^{(M)}\Psi(x, t_{n+1}) &= R_M^{(M)}\Psi(x, t_{n+(M-1)/M}). \end{aligned}$$

Note the symbolic notation: $t_{n+1} = t_n + \Delta t$ for each n, but the time advance in each line ($t_n \to t_{n+1/M}, t_{n+1/M} \to t_{n+2/M}, \ldots$) is not equidistant. If H does not depend on time, the sequence of steps is arbitrary. The band structure of matrices L and R depends on the spatial discretization of H: if it contains the kinetic energy operator $-(\hbar^2/2m)\partial^2/\partial x^2$, which is discretized in the form (9.11) on an equidistant spatial mesh, the matrices are tridiagonal; if it is discretized as in (9.12), they are pentadiagonal. Details can be found in [19]; the methods for time-dependent Hamiltonians are discussed in [20, 21].

1.3 Power and Asymptotic Expansion, Asymptotic Analysis

Power expansion is a tool to describe the behavior of a function in the vicinity of a specific point, while asymptotic expansion and analysis are languages in which we express the dependence of integrals and solutions of differential equations on parameters governing their behavior in the limit of very small or very large values. The basic notation (symbols $\mathcal{O}$, o and $\sim$) is given in Appendix A.1.



Example A nice instance of asymptotic analysis can be found in the study of the system of two freely rotating electric dipoles with an interaction of the form

$$V(r,\Omega)=\frac{\mu_1\mu_2}{4\pi\varepsilon_0 r^3}F(\Omega),\qquad \Omega=(\theta_1,\theta_2,\phi),$$

where $F(\Omega)=-2\cos\theta_1\cos\theta_2+\sin\theta_1\sin\theta_2\cos\phi$. One encounters such systems in studies of orientation of nitroaniline molecules in zeoliths. The relevant quantity is the interaction energy V weighted by the Boltzmann factor $\exp(-V/kT)$ and averaged over all possible orientations of the dipoles Ω,

$$\left\langle V\mathrm{e}^{-V/kT}\right\rangle=\frac{\mu_1\mu_2}{4\pi\varepsilon_0 r^3}\frac{\int \mathrm{d}\Omega\, F(\Omega)\,\mathrm{e}^{\lambda F(\Omega)}}{\int \mathrm{d}\Omega\,\mathrm{e}^{\lambda F(\Omega)}},\qquad \lambda=\frac{\mu_1\mu_2}{4\pi\varepsilon_0 r^3 kT},$$

where $\mathrm{d}\Omega=\sin\theta_1\sin\theta_2\,\mathrm{d}\theta_1\,\mathrm{d}\theta_2\,\mathrm{d}\phi$. By using standard collections of integrals [22, 23] the triple integrals can be reduced to a single integration [24, 25],

$$\left\langle V\mathrm{e}^{-V/kT}\right\rangle=-\frac{\mu_1\mu_2}{4\pi\varepsilon_0 r^3}\frac{\mathrm{d}}{\mathrm{d}\lambda}\log K(\lambda),\qquad K(\lambda)=\frac{8\pi}{\sqrt{3}\,\lambda}\int_1^2\frac{\sinh\lambda x}{\sqrt{x^2-1}}\,\mathrm{d}x.$$

We are interested in the behavior of $K(\lambda)$ for large values of the parameter λ, i.e. at fixed r and low temperatures T. By asymptotic analysis (see p. 23) we get

$$K(\lambda)\sim\frac{4\pi}{3}\frac{\mathrm{e}^{2\lambda}}{\lambda^2}\left(1+\frac{2}{3\lambda}+\frac{1}{\lambda^2}+\frac{22}{9\lambda^3}+\cdots\right) \tag{1.19}$$

or

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\log K(\lambda)=\frac{1}{K}\frac{\mathrm{d}K}{\mathrm{d}\lambda}\sim 2-\frac{2}{\lambda}-\frac{2}{3\lambda^2}+\cdots,\qquad \lambda\to\infty. \tag{1.20}$$

The terms in the expansion (1.20) have clear physical meanings. The leading term $+2$ does not depend on T and reflects the attraction of the dipoles in the state with $\theta_1=\theta_2=0$ that has the lowest energy, $\langle V\exp(-V/kT)\rangle=-\mu_1\mu_2/2\pi\varepsilon_0 r^3$. The second term (proportional to T) expresses the average potential energy of a pair of anisotropic two-dimensional oscillators [24, 25]. The third term (proportional to T^2) corresponds to the non-harmonic part of the potential.

1.3.1 Power Expansion

Assume that a real function f is at least $(n+1)$-times differentiable in the vicinity of the point x_0. The nth order *power (Taylor) expansion* of f is defined as

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + R_n(x) \tag{1.21}$$

with the remainder

$$R_n(x) = \int_{x_0}^{x} \frac{f^{(n+1)}(t)}{n!}(x-t)^n \,\mathrm{d}t = \frac{f^{(n+1)}(x^\star)}{(n+1)!}(x - x_0)^{n+1}, \quad x^\star \in [x_0, x].$$

In the last step we have used the mean value theorem [26]: for any continuous function g on the interval $[a, b]$ there exists a point $a \le x^\star \le b$ such that

$$\int_a^b g(x)\,\mathrm{d}x = g\big(x^\star\big)(b-a),$$

and $g(x^\star)$ is the average value of g. If at x_0 all derivatives of f exist and are finite, f can be expanded in the vicinity of x_0 in an infinite series. This series converges on the interval $(x_0 - r, x_0 + r)$ where r is the *convergence radius of the series*. It is given by the formulas (1.55).

The extension of the Taylor series to the complex plane and inclusion of negative powers of the arguments is called the *Laurent expansion*:

$$f(z) = \sum_{k\in\mathbb{Z}} a_k (z - z_0)^k, \quad a_k = \frac{1}{2\pi \mathrm{i}} \oint_\gamma \frac{f(z)}{(z - z_0)^{k+1}}\,\mathrm{d}z.$$

Here γ is an arbitrary closed contour encircling z_0 in the positive sense. According to the coefficients $\{a_{-k}\}_{k\in\mathbb{N}}$ of the negative powers, we have three distinct cases. The function f is *analytic* if all these coefficients are zero; it is *meromorphic* if there exists a minimal N such that $a_{-k} = 0$ for $\forall k \ge N$; if such N does not exist, we say that f has an *essential singularity* at z_0.

1.3.2 Asymptotic Expansion

The asymptotic expansion of a function is a series, the partial sum of which is an approximation of this function in the regime where the parameter describing the asymptotics becomes large or small. The classical example, used by virtually all textbooks (see, for example, [27]), is the gamma function. For small x we have the Laurent expansion

$$\Gamma(x) = \frac{1}{x} - \gamma + \left(\frac{\gamma^2}{2} + \frac{\pi^2}{12}\right)x + \cdots, \quad 0 < |x| < 1, \tag{1.22}$$

where $\gamma \approx 0.577216$ is the Euler constant. The expansion (1.22) converges for all x satisfying $0 < |x| < 1$, and both the left and the right side are functions of x. On the other hand, for large positive x, we have the expansion

$$\Gamma(x) \sim \mathrm{e}^{-x} x^x \sqrt{\frac{2\pi}{x}} \left(1 + \frac{1}{12x} + \frac{1}{288x^2} + \cdots\right), \quad x \to \infty, \tag{1.23}$$

which does not converge for *any* x! The right side of (1.23) represents an *asymptotic expansion* of the function $\Gamma(x)$ in the limit $x \to \infty$ and is not a function of x. It is merely a sequence of approximations for the function. The error due to the truncation of the series at order n and fixed x does *not* go to zero when n is increased, but it does vanish when $x \to \infty$ at fixed n.

An asymptotic series may converge or diverge. Colloquially, an asymptotic series usually means a divergent series. If the series converges, it can be summed up to an arbitrary term. But it makes no sense to sum a divergent asymptotic series; rather, we use such a series to obtain as good as possible an approximation to the function f. We sum the series only until the sequence of partial sums appears to converge, or stop the summation with the term just before the smallest one. (It turns out that in many asymptotic series the truncation error does not exceed the first omitted term.)

For a more general discussion of asymptotics at an arbitrary point x_0 we choose a sequence of functions $\{\phi_k\}_{k=0}^{\infty}$ with the property $\phi_{k+1}(x) = o(\phi_k(x))$ in the limit $x \to x_0$ and $\phi_k(x) \neq 0$ in the neighborhood of x_0, excluding x_0 itself. The sum $\sum_{k=0}^{\infty} c_k \phi_k(x)$ is called the *general asymptotic expansion* of f if

$$f(x) = \sum_{k=0}^{n} c_k \phi_k(x) + o\big(\phi_n(x)\big), \quad x \to x_0. \tag{1.24}$$

The limit point x_0 can be finite or infinite. With the chosen sequence of functions $\{\phi_k\}$, the expansion coefficients in (1.24) are given by

$$c_k = \lim_{x \to x_0} \frac{f(x) - \sum_{m=0}^{k-1} c_m \phi_m(x)}{\phi_k(x)}, \quad k = 0, 1, \ldots, n. \tag{1.25}$$

Example (See [27], p. 30) We seek an asymptotic expansion of the function

$$f(x) = \frac{1}{x} + \mathrm{e}^{-x} \left(1 - \frac{1}{x}\right)^{-1}, \quad x \to \infty, \tag{1.26}$$

based on the sequence of functions $\{\phi_k(x)\}_{k=0}^{\infty} = \{1, x^{-1}, x^{-2}, \ldots\}$. We compute the coefficients c_k of (1.24) by using (1.25): we get $c_0 = \lim_{x\to\infty} f(x)/\phi_0(x) = 0$ and $c_1 = \lim_{x\to\infty}(f(x) - c_0\phi_0(x))/\phi_1(x) = 1$, while $c_k = 0$ for $k \geq 2$. Based on the chosen sequence $\{\phi_k(x)\}_{k=0}^{\infty}$, the function f has the expansion

$$f(x) \sim c_1 \phi_1(x) = \frac{1}{x}, \quad x \to \infty. \tag{1.27}$$

The same function f can have another asymptotic expansion if a different sequence $\{\phi_k\}_{k=0}^{\infty}$ is chosen. If we select $\{\phi_k\}_{k=0}^{\infty} = \{1, x^{-1}, \mathrm{e}^{-x}, \mathrm{e}^{-x}x^{-1}, \mathrm{e}^{-x}x^{-2}, \ldots\}$, we get for the same f as before the coefficients $c_0 = 0$ and $c_k = 1$ for $k \geq 1$, thus

$$f(x) \sim \frac{1}{x} + \sum_{k=1}^{\infty} \mathrm{e}^{-x} x^{1-k}, \quad x \to \infty.$$

By reading this example backwards we realize that different functions may correspond to the same asymptotic expansion. Based on the sequence $\{1, x^{-1}, x^{-2}, \ldots\}$ both (1.26) and $f(x) = 1/x$ have identical expansions, namely (1.27).

1.3.3 Asymptotic Analysis of Integrals by Integration by Parts

For an arbitrary function f and positive $m \in \mathbb{N}$ let f_m (lower case) represent its mth derivative and F_m (upper case) its mth indefinite integral,

$$f_0 = f, \qquad f_m = \frac{\mathrm{d}^m f}{\mathrm{d}x^m}, \qquad \frac{\mathrm{d}F_m}{\mathrm{d}x} = F_{m-1}.$$

Suppose we are interested in the asymptotic behavior of the integral

$$I(\lambda) = \int_a^b g(\lambda, x) h(\lambda, x)\, \mathrm{d}x,$$

where the asymptotics is determined by the parameter λ. Let g be at least n-times differentiable and let h be integrable. By using integration by parts, $I(\lambda)$ can be written as the sum

$$I_n(\lambda) = \sum_{k=0}^{n-1} s_k(\lambda) + R_n(\lambda),$$

where the terms s_k and the remainder R_n are

$$s_k(\lambda) = (-1)^k \big[g_k(\lambda, b) H_{k+1}(\lambda, b) - g_k(\lambda, a) H_{k+1}(\lambda, a) \big],$$

$$R_n(\lambda) = (-1)^n \int_a^b g_n(\lambda, x) H_n(\lambda, x)\, \mathrm{d}x.$$

If g is $(n+1)$-times continuously differentiable, we have $R_n = s_n + R_{n+1}$, which can be used to estimate the value of the remainder in two cases [28].

1. If g and h are real and the products $g_n H_n$ and $g_{n+1} H_{n+1}$ have constant and equal signs on $[a, b]$, then R_n has the same sign as s_n and opposite than R_{n+1}, and $|R_n| \leq |s_n|$. Example:

$$g(\lambda, x) = (1 + \lambda x)^{-1}, \qquad h(x) = \exp(-x),$$

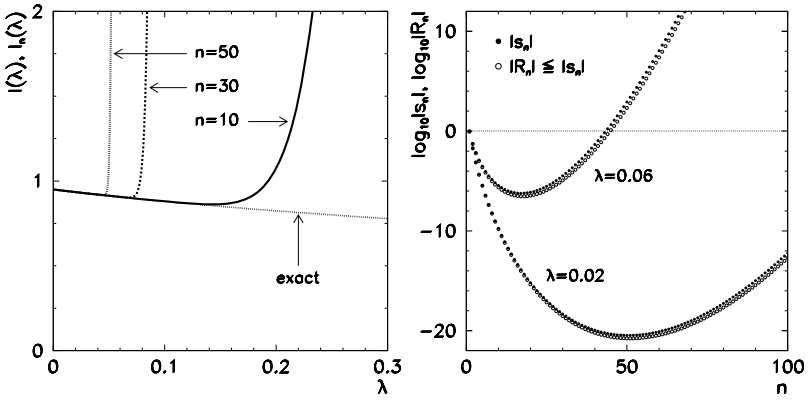


Fig. 1.5 Computation of the $I(\lambda) = \int_a^b \mathrm{e}^{-x}(1+\lambda x)^{-1}\,\mathrm{d}x$ on $[a,b] = [0,3]$ at small λ by asymptotic series. [*Left*] Divergent behavior of partial sums as a function of λ at $n = 10$, 30, and 50. [*Right*] The size of the first omitted term s_n and the remainder R_n as a function of n at $\lambda = 0.02$ and 0.06: we see that $|R_n| \le |s_n|$. (The error $|I(\lambda) - I_n(\lambda)|$ has the same qualitative behavior.) We stop the summation when we reach the minimum in this graph. This point also determines the smallest error one can achieve at a given λ

where $0 \le (-1)^m R_m \le (-1)^m s_m$. In the case $a = 0$ and $b = \infty$ we are dealing with the Euler integral $I(\lambda) = \int_0^\infty \mathrm{e}^{-x}(1+\lambda x)^{-1}\,\mathrm{d}x$, which can be represented as a finite alternating series and the remainder

$$I(\lambda) = \sum_{k=0}^{n} (-1)^k k!\,\lambda^k + (-\lambda)^{n+1}(n+1)! \int_0^\infty \frac{\mathrm{e}^{-x}}{(1+\lambda x)^{n+2}}\,\mathrm{d}x,$$

and which is closely related to the exponential integral Ei (see Fig. 1.5).

2. If g is real, $|H_{n+1}|$ an increasing function of x, and both g_n and g_{n+1} have constant and equal signs on $[a,b]$, or if g is real, $|H_{n+1}|$ is a decreasing function of x, and both g_n and g_{n+1} have constant but opposite signs on $[a,b]$, we have $|R_n| \le 2|s_n|$.

If λ is complex, we rename $x \mapsto x/\lambda$; thus $g(x) = (1+x)^{-1}$ and $h(\lambda, x) = \lambda^{-1}\exp(-x/\lambda)$. Then for $\mathrm{Re}\,\lambda > 0$, the functions g_n, g_{n+1} and H_{n+1} correspond to the second criterion of item 2 above, and we have $|R_n| \le 2|s_n|$. At any rate, the remainder R_n is on the order of the first omitted term, $R_n = \mathcal{O}(s_n)$.

The approach described above is particularly useful when h can be integrated easily, like in the case of $h(\lambda, x) = \exp(\lambda x)$ when $H_m(\lambda, x) = h(x)/\lambda^m$. This is the foundation of the asymptotic expansion of Laplace and Fourier integrals

$$L(\lambda) = \int_a^b \mathrm{e}^{-\lambda x}\phi(x)\,\mathrm{d}x, \qquad F(\lambda) = \int_a^b \mathrm{e}^{\mathrm{i}\lambda x}\phi(x)\,\mathrm{d}x,$$

in the limit $\lambda \to \pm\infty$, which are hard to compute by other means. The asymptotic expansion of the Fourier integral is

$$F_n(\lambda) = \sum_{k=0}^{n-1} \left(\frac{\mathrm{i}}{\lambda}\right)^{k+1} \left[\mathrm{e}^{\mathrm{i}\lambda a}\phi^{(k)}(a) - \mathrm{e}^{\mathrm{i}\lambda b}\phi^{(k)}(b)\right] + R_n(\lambda).$$

The remainder after the truncation of the series to n terms,

$$R_n(\lambda) = \left(\frac{\mathrm{i}}{\lambda}\right)^n \int_a^b \phi^{(n)}(x)\mathrm{e}^{\mathrm{i}\lambda x}\,\mathrm{d}x,$$

has an upper limit when dealing with finite intervals $[a, b]$. By integrating the remainder by parts one more time, we get [28]

$$\left|R_n(\lambda)\right| \le \lambda^{-n-1}\left[\left|\phi^{(n)}(a)\right| + \left|\phi^{(n)}(b)\right| + \int_a^b \left|\phi^{(n+1)}(x)\right|\mathrm{d}x\right] = \mathcal{O}\left(\lambda^{-n-1}\right).$$

1.3.4 Asymptotic Analysis of Integrals by the Laplace Method

Here we analyze integrals of the form

$$I(\lambda) = \int_a^b \phi(x)\mathrm{e}^{-\lambda h(x)}\,\mathrm{d}x \tag{1.28}$$

in the limit of large positive λ, where ϕ and h are real functions of a real variable x. Assume that h has a global minimum at one of the internal points ξ of the interval $[a, b]$, thus $h'(\xi) = 0$ and $h''(\xi) > 0$. Therefore $\exp(-\lambda h(x))$ reaches its maximum at ξ and in its vicinity we may expect the largest contribution to the integral (1.28). We expand h in the Taylor series around ξ,

$$h(x) = h(\xi) + \frac{1}{2}h''(\xi)(x-\xi)^2 + \mathcal{O}\left((x-\xi)^3\right), \tag{1.29}$$

while we take simply $\phi(x) \approx \phi(\xi)$. When these are used in the integral (1.28) and its integration limits are extended to $[-\infty, +\infty]$, we obtain

$$I(\lambda) \approx \int_a^b \phi(\xi)\,\mathrm{e}^{-\lambda[h(\xi)+h''(\xi)(x-\xi)^2/2]}\,\mathrm{d}x \approx \phi(\xi)\,\mathrm{e}^{-\lambda h(\xi)} \int_{-\infty}^{\infty} \mathrm{e}^{-\lambda h''(\xi)x^2/2}\,\mathrm{d}x.$$

This is the Laplace approximation, which is the leading term in the asymptotic expansion

$$I(\lambda) = \mathrm{e}^{-\lambda h(\xi)}\left[\phi(\xi)\sqrt{\frac{2\pi}{\lambda h''(\xi)}} + \mathcal{O}\left(\frac{1}{\lambda}\right)\right], \quad \lambda \to \infty. \tag{1.30}$$

If h reaches its maximum only at $x = a$ and $h'(a) > 0$, the leading term in the asymptotic expansion becomes

$$I(\lambda) = \mathrm{e}^{-\lambda h(a)} \left[\frac{\phi(a)}{h'(a)} \frac{1}{\lambda} + \mathcal{O}\left(\frac{1}{\lambda^2} \right) \right], \quad \lambda \to \infty, \tag{1.31}$$

while if it reaches its minimum only at $x = b$ and $h'(b) < 0$, we have

$$I(\lambda) = \mathrm{e}^{-\lambda h(b)} \left[-\frac{\phi(b)}{h'(b)} \frac{1}{\lambda} + \mathcal{O}\left(\frac{1}{\lambda^2} \right) \right], \quad \lambda \to \infty. \tag{1.32}$$

The asymptotics of the integrals of the form (1.28), where the dominant contributions originate in the minima of h, is given by expressions (1.30)–(1.32) [27]. Similar formulas can be derived for the case where $\exp(\lambda h(x))$ appears in the integral (1.28) instead of $\exp(-\lambda h(x))$.

Example We seek the leading term in the asymptotic expansion of the integral

$$I(\lambda) = \int_0^1 \frac{\exp(-\lambda[1 + x(1-x)])}{\sqrt{x^2+1}} \, \mathrm{d}x, \quad \lambda \to \infty.$$

In this case $h(x) = 1 + x(1-x)$ and $\phi(x) = 1/\sqrt{x^2+1}$. The function h reaches its minimum at both extreme points of the interval, $x = a = 0$ and $x = b = 1$, at which $h(a) = h(b) = 1$, $h'(a) = 1$, $h'(b) = -1$, $\phi(a) = 1$ and $\phi(b) = 1/\sqrt{2}$. The asymptotic expansion is therefore given by the sum of (1.31) and (1.32):

$$I(\lambda) = \mathrm{e}^{-\lambda} \left[\left(1 + \frac{1}{\sqrt{2}} \right) \frac{1}{\lambda} + \mathcal{O}\left(\frac{1}{\lambda^2} \right) \right], \quad \lambda \to \infty.$$

As an exercise, see how the integral behaves in the limit $\lambda \to -\infty$.

For better Laplace approximations, we need a more general expansion [29]. Assume that h has only one minimum on the interval $[a, b]$, at $x = a$; otherwise, we split the whole interval on suitable subintervals. Assume that h in the vicinity of $x = a$ (in limit $x \searrow a$) can be represented as

$$h(x) \sim h(a) + \sum_{s=0}^{\infty} a_s (x-a)^{s+\alpha}, \tag{1.33}$$

where $\alpha \in \mathbb{R}$ and $\alpha > 0$, $a_0 \neq 0$, while ϕ can be represented as

$$\phi(x) \sim \sum_{s=0}^{\infty} b_s (x-a)^{s+\beta-1}, \tag{1.34}$$

where $b_0 \neq 0$ and we require $\mathrm{Re}\, \beta > 0$ for the constant $\beta \in \mathbb{C}$. Let h' and ϕ be continuous around $x = a$ (except perhaps at $x = a$ itself). Under these assumptions, if

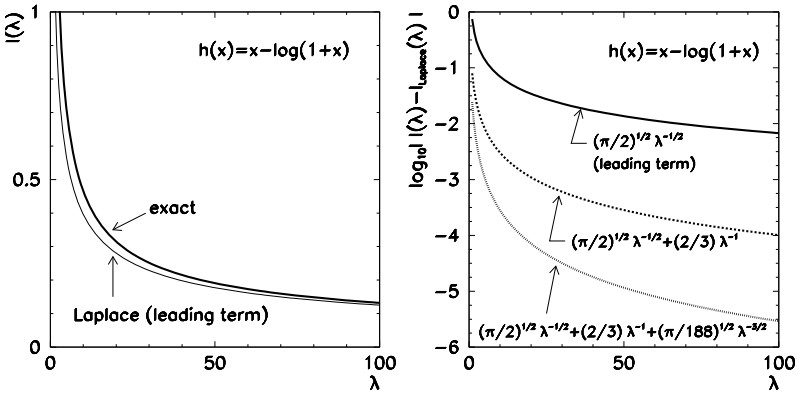


Fig. 1.6 Asymptotic behavior of $I(\lambda) = \int_0^\infty \phi(x)\exp[-\lambda h(x)]\,\mathrm{d}x$, where $\phi(x) = 1$ and $h(x) = x - \log(1+x)$. [*Left*] Exact dependence on λ and the leading Laplace approximation. [*Right*] The error of the approximation. The function h has a minimum at $a = 0$, where it can be expanded as $h(x) = x^2/2 - x^3/3 + x^4/4 - \cdots$. By comparison with (1.33) and (1.34) we get $\alpha = 2$, $a_s = (-1)^s/(s+2)$, $\beta = 1$, $b_0 = 1$, and $b_s = 0$, $s \geq 1$. From here we get the coefficients $c_0 = 1/\sqrt{2}$, $c_1 = 2/3$, and $c_2 = \sqrt{2}/12$ of (1.35)

the integral $I(\lambda)$ absolutely converges for all large enough λ, we have the asymptotic expansion

$$I(\lambda) \sim \mathrm{e}^{-\lambda h(a)} \sum_{s=0}^{\infty} \Gamma\left(\frac{s+\beta}{\alpha}\right) \frac{c_s}{\lambda^{(s+\beta)/\alpha}}, \quad \lambda \to \infty. \tag{1.35}$$

The coefficients c_s in the expansion (1.35) can be expressed by the coefficients a_k and b_k for $k \leq s$. Here we write the first three,

$$\begin{aligned} c_0 &= \frac{b_0}{\alpha a_0^{\beta/\alpha}}, \\ c_1 &= \left[\frac{b_1}{\alpha} - \frac{(\beta+1)a_1 b_0}{\alpha^2 a_0}\right] a_0^{-(\beta+1)/\alpha}, \\ c_2 &= \left[\frac{b_2}{\alpha} - \frac{(\beta+2)a_1 b_1}{\alpha^2 a_0} + \left\{(\alpha+\beta+2)a_1^2 - 2\alpha a_0 a_2\right\}\frac{(\beta+2)b_0}{2\alpha^3 a_0^2}\right] a_0^{-(\beta+2)/\alpha}, \end{aligned} \tag{1.36}$$

while the procedure to compute any c_k can be found in [29] and [30]. The estimate of the error due to the truncation of (1.35) to a finite number of terms is discussed by [31] in Sect. 3.9. The general Laplace method is illustrated in Fig. 1.6 and by the following example.

Example Let us revisit the calculation of the average interaction energy of two electric dipoles and its asymptotic behavior at low temperatures (1.19). We use the

Laplace method to analyze the integral

$$K(\lambda) = \frac{8\pi}{\sqrt{3}\lambda} \int_1^2 \frac{\sinh \lambda x}{\sqrt{x^2-1}}\,\mathrm{d}x = \frac{4\pi}{\sqrt{3}\lambda}\Bigg[\underbrace{\int_1^2 \frac{\mathrm{e}^{\lambda x}}{\sqrt{x^2-1}}\,\mathrm{d}x}_{I_1(\lambda)} - \underbrace{\int_1^2 \frac{\mathrm{e}^{-\lambda x}}{\sqrt{x^2-1}}\,\mathrm{d}x}_{I_2(\lambda)}\Bigg]$$

in the limit $\lambda = \mu_1\mu_2/(4\pi\varepsilon_0 r^3 kT) \to \infty$. By using $x \mapsto -x$ the first term can be rewritten as

$$I_1(\lambda) = \int_1^2 \frac{\mathrm{e}^{\lambda x}}{\sqrt{x^2-1}}\,\mathrm{d}x = \int_{-2}^{-1} \frac{\mathrm{e}^{-\lambda x}}{\sqrt{x^2-1}}\,\mathrm{d}x,$$

so that $h(x) = x$ and $\phi(x) = 1/\sqrt{x^2-1}$. The function h has a minimum at $x = a = -2$, as required by the assumptions for the expansion (1.35). Since h is so simple, its expansion (1.33) has only two terms,

$$h(x) = x = h(a) + \sum_{s=0}^{\infty} a_s (x-a)^{s+\alpha} = -2 + a_0\big(x-(-2)\big)^{0+\alpha} + 0 + 0 + \cdots,$$

from which we read off $\alpha = 1$, $a_0 = 1$, and $a_s = 0$ for $s \geq 1$. We expand ϕ around a in the Taylor series and compare it to the expansion (1.34),

$$\phi(x) = \frac{1}{\sqrt{x^2-1}} = \frac{1}{\sqrt{3}} + \frac{2(x-a)}{3\sqrt{3}} + \frac{(x-a)^2}{2\sqrt{3}} + \cdots = \sum_{s=0}^{\infty} b_s (x-a)^{s+\beta-1},$$

from which we infer $\beta = 1$, $b_0 = 1/\sqrt{3}$, $b_1 = 2/(3\sqrt{3})$, and $b_2 = 1/(2\sqrt{3})$. By using the coefficients a_s and b_s we compute c_s by (1.36) and use them in the expansion (1.35). We get $c_0 = b_0$, $c_1 = b_1$, $c_2 = b_2$. Finally, we have $-\lambda h(a) = 2\lambda$, thus

$$I_1(\lambda) = \mathrm{e}^{2\lambda}\left[\frac{1}{\sqrt{3}\lambda} + \frac{2}{3\sqrt{3}\lambda^2} + \frac{1}{\sqrt{3}\lambda^3} + \cdots\right],$$

from which (1.19) follows. The integral $I_2(\lambda)$ is already in the appropriate form, with the function $h(x) = x$ reaching its minimum at $x = a = 1$. Since $I_2(\lambda)$ behaves like $\mathrm{e}^{-\lambda}$ it is negligible in comparison to $I_1(\lambda)$ in the limit $\lambda \to \infty$.

1.3.5 Stationary-Phase Approximation

The method of *stationary-phase approximation* is used to deduce the asymptotic series for integrals of the form

$$I(\lambda) = \int_a^b \phi(x)\,\mathrm{e}^{\mathrm{i}\lambda h(x)}\,\mathrm{d}x, \tag{1.37}$$

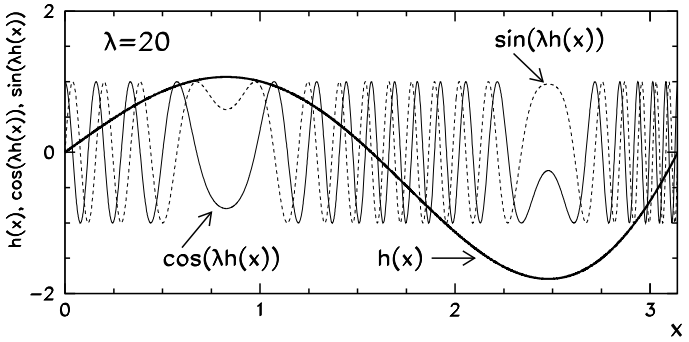


Fig. 1.7 The basic idea of stationary-phase approximation. At large λ the function $\exp(\mathrm{i}\lambda h(x))$ in (1.37) rapidly oscillates around zero. In the regions of large variations of $h(x)$ the contributions to the integral therefore largely cancel out, while there is much less cancellation where the variation is small (near maxima and minima)

where h is a real function of a real variable x. The integrand thus contains the exponential function with an imaginary argument, and we are interested in the behavior of the integral at $|\lambda| \gg 1$. In order to compute $I(\lambda)$ for negative arguments, we use the symmetry $I(\lambda)^\star = I(-\lambda)$.

The approximation can be established by realizing that the leading contribution to $I(\lambda)$ for $\lambda \to \infty$ comes from the integral over the points at which the *phase function h is stationary*, that is, $h'(x) = 0$. Assume that h has only one minimum ($h''(\xi) > 0$) or one maximum ($h''(\xi) < 0$) on the interval $[a, b]$, at ξ. We insert the expansion (1.29) into (1.37) and obtain

$$I(\lambda) \approx \int_a^b \phi(\xi)\, \mathrm{e}^{\mathrm{i}\lambda[h(\xi)+h''(\xi)(x-\xi)^2/2]}\, \mathrm{d}x = \phi(\xi)\, \mathrm{e}^{\mathrm{i}\lambda h(\xi)} \int_a^b \mathrm{e}^{\mathrm{i}\lambda h''(\xi)(x-\xi)^2/2}\, \mathrm{d}x.$$

We extend the integral on the right to the whole real axis and obtain the leading term of the stationary-phase approximation:

$$I(\lambda) \sim \phi(\xi) \sqrt{\frac{2\pi}{\lambda |h''(\xi)|}}\, \exp\left\{\mathrm{i}\left[\lambda h(\xi) + \frac{\pi}{4}\, \mathrm{sign}\big(h''(\xi)\big)\right]\right\}, \tag{1.38}$$

where we have assumed $h''(\xi) \neq 0$ and used $\int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}x^2}\, \mathrm{d}x = \sqrt{\pi}\, \mathrm{e}^{\mathrm{i}\pi/4}$.

Example We are interested in the leading asymptotic term of the integral

$$I(\lambda) = \int_0^\pi \phi(x)\, \mathrm{e}^{\mathrm{i}\,\lambda h(x)}\, \mathrm{d}x, \quad h(x) = \sin(2x)\, \mathrm{e}^{x^2/10},\ \phi(x) = \frac{1}{\sqrt{x^2+1}},$$

in the limit $\lambda \to \infty$. On $[0, \pi]$, the function h has a maximum at $\xi_1 \approx 0.8266$ and a minimum at $\xi_2 \approx 2.4776$ (see Fig. 1.7). At these points, $h''(\xi_1) \approx -4.0841$ and $h''(\xi_2) \approx 7.2550$. The asymptotics of the integral is determined by two contributions

of the form (1.38) which, at any $\lambda \gg 1$, are computed for $\xi = \xi_1$ and $\xi = \xi_2$, and then summed. With $\lambda = 20$, for example, we obtain

$$I(20) \approx (-0.0487 + 0.3566\,\mathrm{i}) + (-0.4814 + 0.2781\,\mathrm{i}) = -0.5301 + 0.6347\,\mathrm{i},$$

while $I(20) \approx -0.5290 + 0.6280\,\mathrm{i}$ by precise numerical integration. The leading-order stationary-phase approximation to the complex value of the integral thus leads to the error in the modulus of $\approx$0.5 % and in the phase of $\approx 0.2°$.

Higher terms of the stationary-phase approximation can be obtained by the generalization of this method [29]. Assume that h has a finite number of stationary points (zeros of $h'(x)$) on the integration interval. We split this interval into subintervals such that there is one stationary point at the lower edge of every subinterval. Let $[a, b]$ be such a subinterval, and let h be monotonously increasing on $[a, b]$, thus $h'(x) > 0$ for $x \in [a, b]$ (in the opposite case, we transform $x \mapsto -x$). Assume that h has the form

$$h(x) = h(a) + (x - a)^{\alpha} h_1(x), \quad h_1(a) \neq 0,$$

where h_1 is smooth on $[a, b]$ and $\alpha \geq 1$. Let ϕ have the form

$$\phi(x) = (x - a)^{\beta - 1} \phi_1(x),$$

where ϕ_1 is smooth on $[a, b]$ and $\beta \in (0, 1]$. Then the asymptotic expansion of the integral (1.37) in the limit $\lambda \to \infty$ is

$$I(\lambda) = \mathrm{e}^{\mathrm{i}\lambda h(a)} \left[\sum_{n=0}^{N-1} a_n \left(\frac{\mathrm{i}}{\lambda} \right)^{(n+\beta)/\alpha} \right] + R_N^{(1)} - \mathrm{e}^{\mathrm{i}\lambda h(b)} \left[\sum_{n=0}^{M-1} b_n \left(\frac{\mathrm{i}}{\lambda} \right)^{n+1} \right] + R_M^{(2)},$$

where $R_N^{(1)} = \mathcal{O}(\lambda^{-(N+\beta)/\alpha})$ and $R_M^{(2)} = \mathcal{O}(\lambda^{-M})$ are the remainders due to the truncation of the series (see [29] for details). By introducing new variables $t^{\alpha} = h(x) - h(a)$ the coefficients a_n can be determined as

$$a_n = \frac{1}{\alpha n!} \Gamma\left(\frac{n + \beta}{\alpha} \right) \left(\frac{\mathrm{d}}{\mathrm{d}t} \right)^n \left[\left(\frac{x - a}{t} \right)^{\beta - 1} \phi_1(x) \frac{\mathrm{d}x}{\mathrm{d}t} \right] \Bigg|_{t=0},$$

and the coefficients b_n as

$$b_n = \left(\frac{1}{h'(x)} \frac{\mathrm{d}}{\mathrm{d}x} \right)^n \left[\frac{\phi(x)}{h'(x)} \right] \Bigg|_{x=b}.$$

Only few instances of functions h and ϕ allow for a simple calculation of the coefficients a_n and b_n. We typically let this work be done by programs for symbolic computation like MATHEMATICA [32] (routine `InverseSeries`). In connection to the integration of rapidly oscillating functions see also Sect. E.2.

General integrals along a contour $\mathcal{C}$ in the complex plane

$$I(\lambda) = \int_{\mathcal{C}} g(z)\, \mathrm{e}^{\lambda f(z)}\, \mathrm{d}z, \quad \lambda \to \infty,$$

where f and g are analytic, can be computed by means of the *method of steepest descent* and by the *saddle-point method*, which are both similar to the Laplace method in spirit, but technically more complicated. For further information, we refer the reader to [29] and [33].

1.3.6 Differential Equations with Large Parameters

Asymptotic approaches are also applicable to the analysis of differential equations. For a physicist, second-order homogeneous equations

$$y''(x) + p(x,\lambda)y'(x) + q(x,\lambda)y(x) = 0 \tag{1.39}$$

in the limit $\lambda \to \infty$ may be particularly relevant. By using the ansatz $y(x) = z(x)\exp(-\frac{1}{2}\int p(x,\lambda)\,\mathrm{d}x)$ (1.39) can be put into the standard form

$$z''(x) + h(x,\lambda)z(x) = 0, \quad h(x,\lambda) = q(x,\lambda) - \frac{1}{2}p'(x,\lambda) - \frac{1}{4}p^2(x,\lambda). \tag{1.40}$$

Assume that $h(x,\lambda)$ has the Laurent expansion

$$h(x,\lambda) = \lambda^{2k}\sum_{n=0}^{\infty} h_n(x)\lambda^{-n}, \tag{1.41}$$

where k is a positive integer and $h_0 \neq 0$. By using this expansion, a large class of problems can be treated, in spite of the seemingly restrictive character of the leading term $\sim \lambda^{2k}$ in (1.41). Equation (1.39) has two types of solutions [28].

The First Type of the Solution has the form

$$z(x,\lambda) = A(x,\lambda)\,\mathrm{e}^{S(x,\lambda)}, \tag{1.42}$$

where we have introduced the amplitude function $A(x,\lambda)$ and the action function $S(x,\lambda)$. They are defined as series in the parameter λ:

$$A(x,\lambda) = \sum_{n=0}^{\infty} a_n(x)\lambda^{-n}, \qquad S(x,\lambda) = \lambda^k \sum_{n=0}^{k-1} b_n(x)\lambda^{-n}. \tag{1.43}$$

When we insert (1.42) in (1.40) and collect the terms with powers λ^{2k-n}, we get

$$\left(b_0'\right)^2 + h_0 = 0, \tag{1.44}$$

$$2b_0'b_m' + h_m + \sum_{n=1}^{m-1} b_n'b_{m-n}' = 0, \quad m = 1, 2, \ldots, k-1, \tag{1.45}$$

where $'$ denotes the derivative with respect to x. This is a system of differential equations for the coefficient functions b_n of the action $S(x, \lambda)$. Since $h_0 \neq 0$, squaring in (1.44) implies two possible signs for the derivative of the leading coefficient, $b_0' = \pm\sqrt{-h_0}$. These possibilities correspond to two linearly independent solutions of (1.40), as expected for a second-order equation. We then use the computed b_n in the equations for the coefficient functions a_n:

$$\begin{aligned} 2a_0'b_0' + a_0\left(b_0'' + h_k + \sum_{n=1}^{k-1} b_n'b_{k-n}'\right) &= 0, \\ 2a_n'b_0' + \sum_{m=0}^{n} a_{n-m}A_m + 2\sum_{m=1}^{n} a_{n-m}'b_m' + a_{n-k}'' &= 0, \quad n = 1, 2, \ldots. \end{aligned} \tag{1.46}$$

The functions h_n, a_n, and b_n are zero if the subscripts are outside of their ranges required by (1.41) and (1.43), thus $h_{-n} = a_{-n} = b_{-n} = b_{k-1+n} = 0$ for $\forall n \in \mathbb{N}$. We have also introduced

$$A_m = b_m'' + h_{k+m} + \sum_{l=m+1}^{k-1} b_l'b_{k+m-l}'.$$

The Second Type of the Solution of (1.40) has the form

$$z(x, \lambda) = \mathrm{e}^{Z(x,\lambda)}, \quad Z(x, \lambda) = \lambda^k \sum_{n=0}^{\infty} c_n(x)\lambda^{-n}, \tag{1.47}$$

where k is a positive integer. When the ansatz (1.47) is used in (1.40) and the terms with equal powers of λ are combined, we obtain

$$\left(c_0'\right)^2 + h_0 = 0, \tag{1.48}$$

$$2c_0'c_n' + h_n + \sum_{m=1}^{n-1} c_m'c_{n-m}' = 0, \quad n = 1, 2, \ldots, k-1, \tag{1.49}$$

$$2c_0'c_n' + h_n + \sum_{m=1}^{n-1} c_m'c_{n-m}' + c_{n-k}'' = 0, \quad n = k, k+1, \ldots, \tag{1.50}$$

while the functions c_{-n} vanish for $\forall n \in \mathbb{N}$. By determining the phase of the leading term (the derivative of which has two possible dependencies, $c_0' = \pm\sqrt{-h_0}$) this procedure yields two linearly independent solutions of (1.40).

For subscripts $0 \leq n \leq k-1$ the system of equations (1.44) and (1.45) for the functions b_n is the same as the system (1.48) and (1.49) for the functions c_n, so

$b_n = c_n$ for $0 \le n \le k-1$. By comparing (1.42) to (1.47) it becomes clear that the amplitude function of the first-type solution is just a formal expansion of the remainder of the action of the second-type solution,

$$A(x,\lambda) = \exp\left(\sum_{n=k}^{\infty} c_n(x)\lambda^{k-n}\right),$$

with functions $\{c_n\}_{n=k}^{\infty}$ determined by (1.50).

We have assumed that $h(x,\lambda)$ as a function of λ has a pole of even degree at infinity, $h(x,\lambda) = \mathcal{O}(\lambda^{2k})$. If the pole has an odd degree, $h(x,\lambda) = \mathcal{O}(\lambda^k)$, the solutions (1.42) and (1.47) are not valid. In this case we can introduce a new asymptotic parameter $\lambda' = \lambda^{1/2}$ and use λ' in the formulas derived above.

The procedure described here is called the WKB (Wentzel–Kramers–Brillouin) method. The special case $k = 1$ coincides with the problems of the Schrödinger equation for a particle of mass m in an one-dimensional potential V,

$$-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}\psi(x) + \big[V(x) - E\big]\psi(x) = 0. \tag{1.51}$$

In the limit of large energies $E \to \infty$ or small Planck constant $\hbar \to 0$ we speak of a *semi-classical* approach. The asymptotic parameter λ then represents high energies $E = \lambda^2$ or the smallness of $\hbar = \lambda^{-1}$. The WKB method is illustrated by the following example adapted from [27]; for details, see [34, 35].

Example The classical example of the WKB method in quantum mechanics is the calculation of the particle's wave-function in a space with linear potential [34]. In (1.51) this means $V(x) = \lambda^2 x$, and it can be rewritten as

$$z''(x) - \lambda^2 x z(x) = 0 \tag{1.52}$$

by a suitable change of variables. This equation is of the form (1.40) with $h(x,\lambda) = -\lambda^2 x$, but let us pretend for a moment that h is still general and has the expansion (1.41). To solve (1.52), we use the ansatz (1.42). First, we determine the leading coefficient of the action, b_0. From (1.44) it follows that

$$b_0'(x) = \pm\sqrt{-h_0(x)}, \qquad b_0''(x) = \mp\frac{h_0'(x)}{2\sqrt{-h_0(x)}}, \qquad b_0(x) = \pm\int_{x^\star}^{x}\sqrt{-h_0(t)}\,\mathrm{d}t.$$

The lower integration limit $x^\star$ will be determined in the following. We compute the coefficient a_0 of the amplitude function (1.43) by using (1.46), $2a_0'b_0' + a_0b_0'' + a_0h_1 = 0$. This is a differential equation for a_0:

$$\frac{\mathrm{d}a_0}{a_0} = \left(-\frac{1}{4}\frac{h_0'}{h_0} \mp \frac{h_1}{2\sqrt{-h_0}}\right)\mathrm{d}x.$$

We use $h_0'/h_0 = (\log h_0)'$, and obtain

$$a_0(x) = \frac{1}{[h_0(x)]^{1/4}} \exp\left\{\mp\frac{1}{2}\int_{x^\star}^{x} \frac{h_1(t)}{\sqrt{-h_0(t)}}\,\mathrm{d}t\right\}.$$

Since $k = 1$, the series (1.43) for $S(x, \lambda)$ contains only the term $\lambda^1 b_0 \lambda^0 = \lambda b_0$. The final structure of the solution to the leading order in λ is therefore simply $z(x, \lambda) = a_0(x)\exp\{\lambda b_0(x)\}$, but its precise form still depends on the sign of the coefficient function h_0 from the expansion (1.41).

The point $x^\star$, in which h_0 has a simple zero, ($h_0(x^\star) = 0$, $h_0'(x^\star) \neq 0$), is called the *turning* or *transition point*, since the physical character of the solution changes at this point. In the region where $h_0 > 0$, the expressions written above yield two linearly independent oscillatory solutions

$$z^{\pm}_{\mathrm{osc}}(x, \lambda) \approx \frac{1}{[h_0(x)]^{1/4}} \exp\left\{\pm\mathrm{i}\,\lambda \int_{x^\star}^{x} \sqrt{h_0(t)}\,\mathrm{d}t \pm \frac{\mathrm{i}}{2}\int_{x^\star}^{x} \frac{h_1(t)}{\sqrt{h_0(t)}}\,\mathrm{d}t\right\},$$

while in the region with $h_0 < 0$ we get exponentially increasing or decreasing solutions

$$z^{\pm}_{\mathrm{exp}}(x, \lambda) \approx \frac{1}{[|h_0(x)|]^{1/4}} \exp\left\{\pm\lambda \int_{x^\star}^{x} \sqrt{|h_0(t)|}\,\mathrm{d}t \mp \frac{1}{2}\int_{x^\star}^{x} \frac{h_1(t)}{\sqrt{|h_0(t)|}}\,\mathrm{d}t\right\}.$$

In order for the WKB analysis to be valid, some authors require that the whole function h, not just its leading term h_0, should have a zero at the turning point. It turns out that it is very hard to formulate an asymptotic analysis of the WKB type if h has a zero in the region being discussed while h_0 does not. The explicit demand that h_0 has a simple zero can thus be understood as a necessary condition for the applicability of the WKB method.

Let us reconsider (1.52). From (1.41) we read off $h_0(x) = -x$ and $h_n(x) = 0$ for $n \geq 1$. In the exponents of $z^{\pm}_{\mathrm{osc}}(x, \lambda)$ and $z^{\pm}_{\mathrm{exp}}(x, \lambda)$ only the first term appears, and the turning point is $x^\star = 0$. In its vicinity the WKB approximation fails (see Fig. 1.8). The solution of (1.52) in the WKB approximation to the left of the turning point ($x < x^\star$ and $h_0(x) > 0$) is a linear combination of the solutions z^{+}_{osc} and z^{-}_{osc}. The solution on the right ($x > x^\star$ and $h_0(x) < 0$) is a linear combination of z^{+}_{exp} and z^{-}_{exp}.

The method described above can be generalized to the case when h has a zero $x^\star$ of degree p. Assume that h is analytic at $x^\star$ and that in the vicinity of $x^\star$, in the limit $\lambda \to \infty$, it has the asymptotic expansion

$$h(x, \lambda) \sim C\,\lambda^{2k}\big(x - x^\star\big)^p, \quad C \in \mathbb{R}. \tag{1.53}$$

By substitution $x - x^\star = |C\,\lambda^{2k}|^{-1/(2+p)}t$ and by using (1.53) we rewrite (1.40) as

$$\frac{\mathrm{d}^2 z(t)}{\mathrm{d}t^2} + s t^p z(t) = 0, \quad s = \mathrm{sign}(C).$$

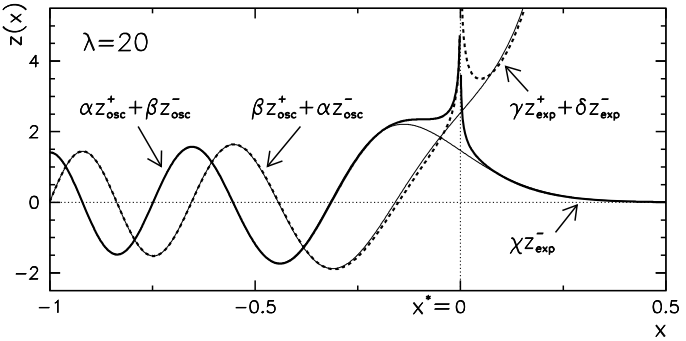


Fig. 1.8 The solution of $z''(x) - \lambda^2 x z(x) = 0$ with $\lambda = 20$ in the WKB approximation. The exact solutions (*thin lines*) are given by the Airy functions Ai and Bi (Problem 1.5.2). The coefficients α, β, γ, δ and χ in the linear combinations $z^{\pm}_{\rm osc}$ and $z^{\pm}_{\rm exp}$ are determined such that the WKB solutions match the exact solutions far from the turning point $x^\star = 0$

This equation is valid near $t = 0$. Its solutions are known [36] and can be expressed in terms of the Bessel functions of the first and second kind [22] as

$$z(t) = \sqrt{t} \begin{cases} C_1 J_{1/2q}(t^q/q) + C_2 Y_{1/2q}(t^q/q); & s = +1, \\ C_1 I_{1/2q}(t^q/q) + C_2 K_{1/2q}(t^q/q); & s = -1, \end{cases}$$

where $q = \frac{1}{2}(p+2)$. In seeking the solutions over a larger region, the constants C_1 and C_2 can be determined such that the solution near the turning point matches the solution far from the turning point in amplitude and phase.

For details on the formulation and use of asymptotic series see [28] and [29]. Connections of asymptotic series to special functions are discussed in the classic work [31].

1.4 Summation of Finite and Infinite Series

Physical quantities are often represented as infinite or finite series

$$S = \sum_{k=0}^{\infty} a_k, \qquad S_n = \sum_{k=0}^{n} a_k, \quad a_k \in \mathbb{R} \text{ or } a_k \in \mathbb{C}.$$

The sum S_n of the first $n+1$ terms of S is the *nth partial sum* of S. The series S *converges* if the sequence $\{S_n\}$ converges, i.e. if for any $\varepsilon > 0$ a $\kappa \in \mathbb{N}$ can be found such that for each $p \in \{0\} \cup \mathbb{N}$ we have $n > \kappa \implies |S_{n+p} - S_n| < \varepsilon$. A convergent sequence converges to a finite limit and this limit is its one and only cluster point. The series S is said to *diverge* if the sequence $\{S_n\}$ has a limit at infinity, has multiple cluster points or has no cluster points at all. Sometimes we carelessly interpret divergence as "convergence" to infinity.

General properties of series and summation methods are treated by the theory of summability [37, 38]. Further reading on modern techniques of symbolic summation of series to closed forms can be found in [39, 40].

1.4.1 Tests of Convergence

Tests of convergence are procedures used to identify sufficient conditions for convergence of infinite series $\sum_{k=0}^{\infty} a_k$. In many tests, we disregard the signs of the terms (if $a_k \in \mathbb{R}$) or their phases (if $a_k \in \mathbb{C}$) and only use their absolute values. This simplification is based on the Cauchy inequality $|\sum_k a_k| \leq \sum_k |a_k|$, from which we infer that a series converges if the corresponding series with absolute values of terms converges (*absolute convergence*). The necessary condition for the convergence of any series is $\lim_{k\to\infty} a_k = 0$.

Comparison Test For a given sequence $\{a_k\}_{k\in\mathbb{N}_0}$, where all $a_k \geq 0$, we find a sequence $\{b_k\}_{k\in\mathbb{N}_0}$. If there exists a $N \in \mathbb{N}_0$ such that $0 \leq a_k \leq b_k$ for all $k > N$ and the series $\sum_k b_k$ converges, the series $\sum_k a_k$ also converges. If at some $N \in \mathbb{N}_0$ we find $0 \leq b_k \leq a_k$ for all $k > N$ and the series $\sum_k b_k$ diverges, then the series $\sum_k a_k$ also diverges.

Quotient and Cauchy Square-Root Test In the quotient test we observe the upper limit of the quotient of the consecutive terms of the series,

$$\rho = \limsup_{k\to\infty} \left| \frac{a_{k+1}}{a_k} \right|,$$

while in the square-root test, we look at the upper limit of the square roots,

$$\rho = \limsup_{k\to\infty} |a_k|^{1/k}.$$

The series (absolutely) converges if $\rho < 1$, and (absolutely) diverges if $\rho > 1$. In the case $\rho = 1$ the test is inconclusive (the series may either converge or diverge).

Integral Test Assume we have a sequence $\{a_k\}_{k\in\mathbb{N}}$ where all $a_k \geq 0$, and there exists a continuous monotonously decreasing function f such that $f(k) = a_k$ for all $k \geq 1$. Then the series $\sum_{k=1}^{\infty} a_k$ and the integral $\int_1^{\infty} f(x)\,\mathrm{d}x$ either both converge or both diverge. In the case of convergence, the difference $R_n = S - S_n = \sum_{k=n+1}^{\infty} a_k$ satisfies $\int_{n+1}^{\infty} f(x)\,\mathrm{d}x \leq R_n \leq \int_n^{\infty} f(x)\,\mathrm{d}x$.

Kummer's and Raabe's Test To perform the Kummer's test, we need a sequence $\{a_k\}_{k\in\mathbb{N}_0}$, $a_k > 0$, and a sequence $\{b_k\}_{k\in\mathbb{N}_0}$, $b_k > 0$, from which we form the limit

$$\rho = \lim_{k\to\infty} \left(b_k \frac{a_k}{a_{k+1}} - b_{k+1} \right).$$

The series $\sum_k a_k$ converges if $\rho > 0$, while it diverges if $\rho < 0$ and the series $\sum_k 1/b_k$ diverges. If $\rho = 0$ the criterion is useless. In the case $b_k = k$ we obtain the Raabe's test, where we observe the limit

$$\rho = \lim_{k\to\infty}\left(k\frac{a_k}{a_{k+1}} - k - 1\right) = \lim_{k\to\infty}\left[k\left(\frac{a_k}{a_{k+1}} - 1\right)\right] - 1.$$

The series $\sum_k a_k$ converges if $\rho > 0$, and diverges if $\rho < 0$. In the case $\rho = 0$ the convergence or divergence cannot be ascertained.

Limit Comparison Test To a sequence $\{a_k\}_{k\in\mathbb{N}_0}$, $a_k > 0$, we find a sequence $\{b_k\}_{k\in\mathbb{N}_0}$, $b_k > 0$, such that the limit $\rho = \lim_{k\to\infty} a_k/b_k$ exists. If ρ is finite and $\rho \neq 0$, then both $\sum_k a_k$ and $\sum_k b_k$ either converge or diverge.

Leibniz's Test for Alternating Series An important class of real series is represented by alternating series $\sum_{k=0}^{\infty}(-1)^k a_k$ where $a_k \geq 0$ (the consecutive terms change signs). If a_k decrease monotonically and $\lim_{k\to\infty} a_k = 0$ holds true, the alternating series converges. The remainder can be bounded as $|S - S_n| \leq a_n$.

Most of the enumerated tests are adapted for analytic work, but very often we can also use them numerically to determine with large certainty whether a given series diverges or converges.

We are also interested in the convergence of the power series

$$\sum_{k=0}^{\infty} a_k(z - z_0)^k, \quad a_k, z, z_0 \in \mathbb{C}, \tag{1.54}$$

which is used to describe functions around a point z_0. Let us consider only absolute convergence and define the largest disk (circular region of points z in the complex plane) $\{z : |z - z_0| \leq r\}$, within which the series (1.54) absolutely converges. The disk radius r is the *convergence radius* and can be computed as

$$r = \left(\limsup_{k\to\infty} |a_k|^{1/k}\right)^{-1} \quad \text{or} \quad r = \limsup_{k\to\infty}\left|\frac{a_k}{a_{k+1}}\right|. \tag{1.55}$$

1.4.2 Summation of Series in Floating-Point Arithmetic

In floating-point arithmetic, the summation of real series $\sum_{k=0}^{n} a_k$ implies rounding errors. In particular for series with $n \to \infty$, precision is of utmost importance. Substantial work has been done in the minimization of summation errors (see [8, 41–43]). Here we list three most widely used summation methods that do not require more than $\mathcal{O}(n)$ of operations.

Simple Recursive Summation Assume that we have the values $\{a_k\}_{k=0}^{n}$ and wish to compute their sum $S = \sum_{k=0}^{n} a_k$. Most obviously, this can be accomplished by computing $\hat{S} = (\cdots(((a_0 \oplus a_1) \oplus a_2) \oplus a_3)\cdots \oplus a_{n-1}) \oplus a_n$, in a loop

Input: real numbers $a_0, a_1, \ldots, a_n$
$\hat{S} = a_0$;
for $k = 1$ **step** 1 **to** n **do**
  $\hat{S} = \hat{S} + a_k$;
end
Output: $\hat{S}$ is the numerical sum of numbers a_k

The deviation of the numerical sum $\hat{S}$ from the exact sum S strongly depends on how a_k are sorted. If they are unsorted, we have

$$|S - \hat{S}| \leq \frac{\varepsilon_{\mathrm{M}}}{2} n \sum_{k=0}^{n} |a_k| + \mathcal{O}\big(\varepsilon_{\mathrm{M}}^2\big), \tag{1.56}$$

where ε_{M} is the arithmetic precision (see p. 2) [42]. In simple summation one can therefore expect a loss of up to $\log_{10} n$ significant digits. The estimate for the upper limit of the error (not the error itself) is smallest when the terms are sorted as $|a_k| \leq |a_{k+1}|$. Sorting requires at least $\mathcal{O}(n \log n)$ additional operations.

Kahan's Algorithm A much better procedure to sum a series, by which the effect of rounding errors is greatly diminished, was proposed by Kahan [44]:

Input: real numbers $a_0, a_1, \ldots, a_n$
$\hat{S} = a_0$;
$c = 0$;
for $k = 1$ **step** 1 **to** n **do**
  $y = a_k - c$;
  $t = \hat{S} + y$;
  $c = (t - \hat{S}) - y$; // do not omit brackets
  $\hat{S} = t$;
end
Output: $\hat{S}$ is the numerical sum of numbers a_k

Algebraically, the value of c is zero, but in finite arithmetic it represents a large part of the lost precision when summing $t = \hat{S} + y$. It is added to the sum in the next step and by doing this, it compensates the rounding error from the previous step. The deviation of the numerical sum from the exact one satisfies

$$|S - \hat{S}| \leq \big(\varepsilon_{\mathrm{M}} + \mathcal{O}\big(n\varepsilon_{\mathrm{M}}^2\big)\big) \sum_{k=0}^{n} |a_k|. \tag{1.57}$$

According to (1.57), Kahan's summation is more precise than simple summation for $n\varepsilon_{\mathrm{M}}/2 \leq 1$. In practice, this applies to even larger n (see Fig. 1.9). In the implementation of the algorithm we should make sure that the compiler does not simplify it, since the essence of its strength is hidden in the rules of floating-point arithmetic. In C and C++ variables should be declared `volatile`.

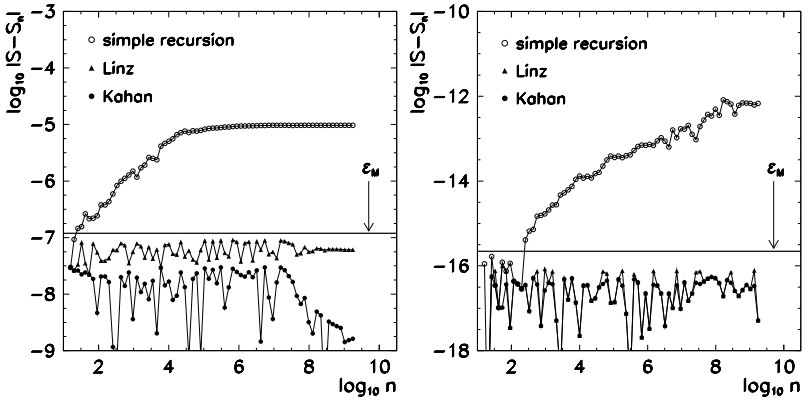


Fig. 1.9 Rounding errors in summing series with many terms. Shown is the absolute error of the numerical partial sums $S_n = \sum_{k=0}^{n} (-1)^k/(k+1)$ with respect to the limiting value $S_\infty = \log 2$. [*Left*] Summation in single-precision arithmetic. [*Right*] Summation in double-precision arithmetic. The horizontal lines correspond to $\varepsilon_M = 1.19 \times 10^{-7}$ (*left*) and $\varepsilon_M = 2.22 \times 10^{-16}$ (*right*)—see p. 2

Recursive Summation of Pairs Summation is an associative and commutative operation between real numbers. Algebraically, the order of summation is thus irrelevant, and this fact is exploited by the Linz procedure [45]. In the first step we sum the consecutive pairs of terms and obtain a new series. In this series we again sum the consecutive pairs and repeat this ($r = \lceil \log_2 n \rceil$)-times, until we are left with only one term, which represents the final sum:

Input: real numbers $a_0, a_1, \ldots, a_{n-1}$, where $n = 2^r$, $r \in \mathbb{N}$
$m = m' = n/2$;
for $k = 0$ **step** 1 **to** $m - 1$ **do**
  $S_{0,k} = a_{2k} + a_{2k+1}$;
end
for $j = 1$ **step** 1 **to** $r - 1$ **do**
  $m' = m'/2$;
  for $k = 0$ **step** 1 **to** $m' - 1$ **do**
    $S_{j,k} = S_{j-1,2k} + S_{j-1,2k+1}$;
  end
end
Output: $\hat{S} = S_{r-1,0}$ is the numerical sum of numbers a_k

Because each term a_k in the sum is touched only r-times, the deviation of the sum $\hat{S}$ from the exact value S is much smaller than in simple recursive summation:

$$|S - \hat{S}| \leq \frac{\varepsilon_M}{2} \log_2 n \sum_{k=0}^{n-1} |a_k|$$

(compare to (1.56)). For the intermediate sums $S_{j,k}$ we need additional computer memory to store $n/2$ real numbers, which is a bit wasteful compared to the simple and Kahan's summation which require only $\mathcal{O}(1)$ of memory. Linz's algorithm can be improved by compensating the numerical error and selecting the pairs in a more intricate manner. For details, consult [46].

In all three methods we specified the upper bounds for $|S - \hat{S}|$; for a given set of numbers $\{a_k\}$, all methods may be equally precise. In general, we recommend Linz's method unless pairs cannot be formed or this does not make much sense (for example, for relatively short series). On the other hand, Kahan's method, which is both simple and precise, never fails to enchant (see Fig. 1.9). For very precise summation, we resort to more sophisticated but slower methods like *distillation algorithms* described in [41, 47, 48].

1.4.3 Acceleration of Convergence

The convergence of the partial sums $S_n = \sum_{k=0}^{n} a_k$ to the limit $S = \lim_{n\to\infty} S_n$ may be slow. By "slow" we mean its leading behavior to be $|S_n - S| = \mathcal{O}(n^{-p})$ (power) or $|S_n - S| = \mathcal{O}((\log n)^{-p})$ (logarithmic) where $p > 0$ is the convergence order. Slow convergence is not desired since it implies large numerical costs and a potential accumulation of rounding errors.

We speak of "fast" convergence when it is better than "slow" according to the definition given above. Ideally, one would like to have exponential (geometric) convergence $|S_n - S| = \mathcal{O}(a^n)$ where $a \in [0, 1)$. In many cases, convergence can be accelerated by transforming the original series into another series that converges more rapidly. In the following, we describe a few basic approaches. A modern introduction to convergence acceleration with excellent examples and many hints can be found in [49]; for a more detailed review, see [50, 51].

Richardson Extrapolation Assume that we already know the order of convergence for a series $S = \sum_{k=0}^{\infty} a_k$, so that for its partial sums $S_n = \sum_{k=0}^{n} a_k$ we have

$$S = S_n + \frac{\alpha}{n^p} + \mathcal{O}\big(n^{-r}\big), \quad r > p > 0.$$

We think of the "value of the series" as being the value of the partial sum plus a correction with the known leading-order behavior α/n^p. By transforming

$$T_n^{(1)} = \frac{2^p S_{2n} - S_n}{2^p - 1} = S_{2n} + \frac{S_{2n} - S_n}{2^p - 1}$$

the term α/n^p can be eliminated and the terms $T_n^{(1)}$ give us a better estimate for the sum, for which we obtain $S = T^{(1)} + \mathcal{O}(n^{-r})$. The very same trick can be repeated—until this makes sense—by forming new sequences,

$$T_n^{(2)} = \frac{2^{p+1} T_{2n}^{(1)} - T_n^{(1)}}{2^{p+1} - 1}, \qquad T_n^{(3)} = \frac{2^{p+2} T_{2n}^{(2)} - T_n^{(2)}}{2^{p+2} - 1}, \quad \cdots.$$

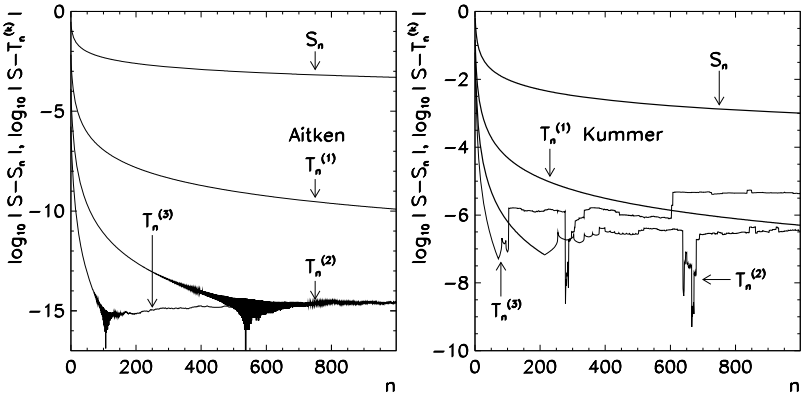


Fig. 1.10 Acceleration of convergence of partial sums. [*Left*] Aitken's method for the sum $S_n = \sum_{k=0}^{n} (-1)^k/(k+1)$ with the limit $S = \lim_{n\to\infty} S_n = \log 2$. Shown is the acceleration of this series with very slow (logarithmic) convergence by three-fold repetition of the Aitken's method. For typical series usually a single step of (1.58) suffices. [*Right*] Kummer's acceleration of the sums $S_n = \sum_{k=1}^{n} 1/k^2$ with the limit $S = \lim_{n\to\infty} S_n = \pi^2/6$ by using the auxiliary series $\sum_{k=1}^{\infty} 1/(k(k+1))$

Richardson's procedure is an example of a *linear extrapolation method* X, in which for partial sums S_n and T_n of two series we have $X(\lambda S_n + \mu T_n) = \lambda X(S_n) + \mu X(T_n)$. It is efficient if the partial sums S_n behave like polynomials in some sequence h_n, that is, $S_n = S + c_1 h_n^{p_1} + c_2 h_n^{p_2} + \cdots$ or $S_{n+1} = S + c_1 h_{n+1}^{p_1} + c_2 h_{n+1}^{p_2} + \cdots$, where the ratio h_{n+1}/h_n is constant. If this condition is not met, linear extrapolation may become inefficient or does not work at all. In such cases we resort to *semi-linear* or *non-linear extrapolation* [49].

Aitken's Method Aitken's method is one of the classical and most widely used ways to accelerate the convergence by non-linear extrapolation. Assume that we have a sequence of partial sums S_n with the limit $S = \lim_{n\to\infty} S_n$. We transform the sequence S_n into a new sequence

$$T_n^{(1)} = S_n - \frac{(S_{n+1} - S_n)^2}{S_{n+2} - 2S_{n+1} + S_n}, \quad n = 0, 1, 2, \ldots, \tag{1.58}$$

where the fraction should be evaluated exactly in the given form in order to minimize rounding errors. (Check that the transformed sequence (1.58) is identical to the column $\epsilon(n, 2)$ of the Wynn's table (1.12).) We repeat the process by using $T_n^{(1)}$ instead of S_n to form yet another, even more accelerated sequence $T_n^{(2)}$, and proceed thus until it continues to make sense as far as the rounding errors are concerned. Figure 1.10 (left) shows the comparison of convergence speeds for the unaccelerated partial sums S_n and the accelerated sequences $T_n^{(1)}$, $T_n^{(2)}$, and $T_n^{(3)}$.

Aitken's method is optimally suited for acceleration of linearly convergent sequences, for which $\lim_{n\to\infty}(S_{n+1} - S)/(S_n - S) = a$ with $-1 \le a < 1$. Such sequences originate in numerous numerical algorithms based on finite differences. In

some cases, we apply Aitken's formula to triplets of partial sums S_{n+p}, S_n, and S_{n-p}, where $p > 1$, because sometimes the geometric convergence of a series only becomes apparent at larger p; see also Sect. 2.3 and [52].

Kummer's Acceleration The basic idea of the Kummer's method of summing a convergent series $S = \sum_k a_k$ is to subtract from it another (auxiliary) convergent series $B = \sum_k b_k$ with the known limit B, such that

$$\lim_{k\to\infty} \frac{a_k}{b_k} = \rho \neq 0.$$

Then the original series can be transformed to

$$T = \sum_k a_k = \rho \sum_k b_k + \sum_k (a_k - \rho b_k) = \rho B + \sum_k \left(1 - \rho \frac{b_k}{a_k}\right) a_k. \tag{1.59}$$

The convergence of the series on the right is faster than the convergence of that on the left since $(1 - \rho b_k/a_k)$ tends to zero when $k \to \infty$. An example is the sum $S = \sum_{k=1}^{\infty} 1/k^2 = \pi^2/6$ from which we subtract $B = \sum_{k=1}^{\infty} 1/(k(k+1)) = 1$, thus $\rho = \lim_{k\to\infty} k(k+1)/k^2 = 1$. We use the terms a_k and b_k, as well as ρ and B, in (1.59), and get the transformed partial sum

$$T_n^{(1)} = 1 + \sum_{k=1}^{\infty} \left(1 - \frac{k^2}{k(k+1)}\right) \frac{1}{k^2},$$

which has a faster convergence than the original series. Again, the procedure can be invoked repeatedly (see [53] and Fig. 1.10 (right)).

1.4.4 Alternating Series

In alternating series the sign of the terms flips periodically,

$$S = a_0 - a_1 + a_2 - a_3 + \cdots = \sum_{k=0}^{\infty} (-1)^k a_k, \qquad S_n = \sum_{k=0}^{n} (-1)^k a_k.$$

In physics such examples can be encountered e.g. in electro-magnetism in problems with oppositely charged particles or currents flowing in opposite directions. An example is the calculation of the electric potential $U(x, y, z)$ of charges of opposite signs lying next to each other at distances a along the x-axis:

$$U(x, y, z) \propto \sum_{k=-\infty}^{\infty} \frac{(-1)^k}{\sqrt{(x + ka)^2 + y^2 + z^2}}.$$

Making a Monotonous Series Alternate For realistic physics problems, the results of series summation may be unpredictable. Simple recursive summation may suffer from large rounding errors. On the other hand, the acceleration of alternating series is typically more efficient than the acceleration of series with exclusively positive (or exclusively negative) terms. A monotonous sequence can be transformed into an alternating one by using the Van Wijngaarden's trick:

$$\sum_{k=0}^{\infty} a_k = \sum_{k=0}^{\infty} (-1)^k b_k, \quad b_k = \sum_{j=0}^{\infty} 2^j a_{2^j(k+1)-1}.$$

Euler's Transformation One of the oldest ways to accelerate the convergence of an alternating sequence by a linear combination of its terms is the *Euler transformation*. We rewrite the original sum $S = \sum_k (-1)^k a_k$ and its partial sum as

$$S = \sum_{k=0}^{\infty} (-1)^k \frac{\Delta^k a_0}{2^{k+1}}, \qquad S_n = \sum_{k=0}^{n} (-1)^k \frac{\Delta^k a_0}{2^{k+1}}, \tag{1.60}$$

where

$$\Delta^k a_0 = (-1)^k \sum_{j=0}^{k} (-1)^j \binom{k}{j} a_j.$$

If there exist $N \in \mathbb{N}$ and $C > 0$ such that $|\Delta^n a_0| \leq C$ for all $n > N$, the series (1.60) converges faster than geometrically with

$$|S - S_n| \leq \frac{C}{2^{n+1}}, \quad n > N.$$

In practical algorithms, we first form the partial sums

$$s_n^{(0)} = \sum_{k=0}^{n} (-1)^k a_k, \quad n = 0, 1, \ldots, N-1,$$

and recursively compute the *partial Euler transforms*

$$s_n^{(j+1)} = \frac{1}{2}\left(s_n^{(j)} + s_{n+1}^{(j)}\right), \quad j = 0, 1, \ldots. \tag{1.61}$$

The values $T_n \equiv s_0^{(n)}$ represent the improved (accelerated) approximations of the partial sums S_n (Fig. 1.11 (left)). The procedure is numerically demanding, since it requires $\mathcal{O}(n^2)$ operations and $\mathcal{O}(n)$ of memory for a complete transformation of a series with n terms. It turns out that the optimally precise results are obtained not by using the transform (1.61) with $j = N-1$ and $n = 0$, but with $j = \lceil 2N/3 \rceil$ and $n = \lceil N/3 \rceil$. An efficient implementation is given in [54].

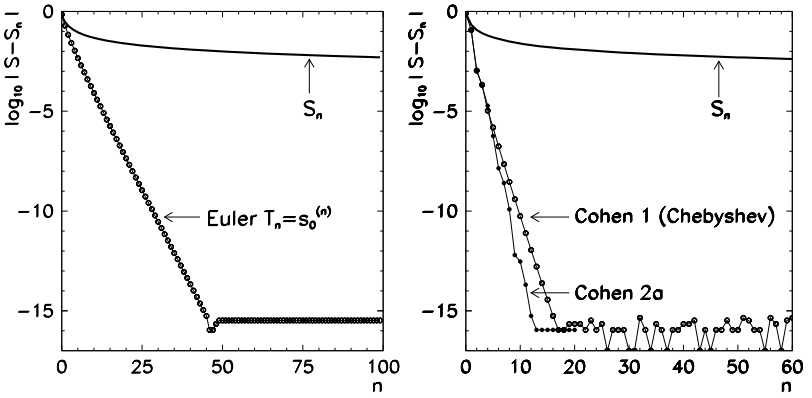


Fig. 1.11 Examples of acceleration of alternating series. Shown are the partial sums $S_n = \sum_{k=0}^{n} (-1)^k a_k$ without and with acceleration. [*Left*] Euler's method (1.61) for $a_k = 1/(k+1)$ (limit $S = \lim_{n\to\infty} S_n = \log 2$). [*Right*] Cohen–Villegas–Zagier's algorithm 1 from p. 42 by using Chebyshev polynomials (1.65) and algorithm 2a in [55] for $a_k = 1/(2k+1)$ (limit $S = \lim_{n\to\infty} S_n = \pi/4$). To compute the partial sum to ≈ 15 significant digits typically less than $\approx$10–20 terms of the accelerated series are required

Generalizing the Euler's Method Euler's transformation can be generalized by using the theory of measures. In this fresh approach to the summation of alternating series [55] we assume that for a series $\sum_{k=0}^{\infty} (-1)^k a_k$ there exists a positive function w such that the series terms a_k are its moments on the interval $[0, 1]$,

$$a_k = \int_0^1 x^k w(x)\,\mathrm{d}x. \tag{1.62}$$

The sum of the series can then be written as

$$S = \sum_{k=0}^{\infty} (-1)^k a_k = \int_0^1 \left(\sum_{k=0}^{\infty} (-1)^k x^k w(x) \right) \mathrm{d}x = \int_0^1 \frac{w(x)}{1+x}\,\mathrm{d}x.$$

(In the final summation formula the weight function does not appear.) In the last step, we have used the identity

$$\sum_{k=0}^{n-1} (-1)^k x^k = \frac{1-(-x)^n}{1+x}, \quad |x| < 1, \tag{1.63}$$

in the limit $n \to \infty$. We now choose a sequence of polynomials $\{P_n\}$, where P_n has a degree n and $P_n(-1) \neq 0$. To the sequence $\{P_n\}$ we assign the numbers

$$S_n = \frac{1}{P_n(-1)} \int_0^1 \frac{P_n(-1) - P_n(x)}{1+x}\, w(x)\,\mathrm{d}x.$$

The numbers S_n are linear combinations of the series terms a_k. This can be seen by inserting the expression for a general polynomial $P_n(x) = \sum_{k=0}^{n} p_k(-x)^k$ into the equation for S_n and observe (1.63) and a_k (1.62). We obtain

$$S_n = \frac{1}{d_n}\sum_{k=0}^{n-1}(-1)^k c_k^{(n)} a_k, \quad d_n = \sum_{k=0}^{n} p_k,\ c_k^{(n)} = \sum_{j=k+1}^{n} p_j.$$

The S_n defined in this way represent the partial sums of S that converge to S when n is increased. The difference between the partial sum S_n and the sum S can be constrained as

$$|S - S_n| \le \frac{1}{|P_n(-1)|}\int_0^1 \frac{|P_n(x)|}{1+x}\, w(x)\,\mathrm{d}x \le \frac{M_n}{|P_n(-1)|}|S|,$$

where $M_n = \sup_{x\in[0,1]}|P_n(x)|$ is the maximum value of the polynomial P_n on $[0, 1]$. The sufficient condition for the convergence of the partial sums S_n to S is therefore $\lim_{n\to\infty} M_n/P_n(-1) = 0$. The authors of [55] recommend to choose a sequence of polynomials $\{P_n\}$ such that $M_n/P_n(-1)$ converges to zero as quickly as possible. The following three choices are the most fruitful.

The first type of the polynomials P_n that may cross one's mind is

$$P_n(x) = (1-x)^n = \sum_{k=0}^{n}\binom{n}{k}(-x)^k, \qquad P_n(-1) = 2^n, \qquad M_n = 1.$$

Namely, the corresponding partial sums are

$$S_n = \frac{1}{2^n}\sum_{k=0}^{n-1}(-1)^k c_k^{(n)} a_k, \quad c_k^{(n)} = \sum_{j=k+1}^{n}\binom{n}{j}, \tag{1.64}$$

and they are identical to the partial sums of the Euler transform (1.60), except for a different subscripting (the sums (1.60) with subscript n are equal to the sums (1.64) with subscript $n + 1$). By this choice we obtain $|S - S_n| \le |S|/2^n$. Faster convergence, $|S - S_n| \le |S|/3^n$, can be obtained by using the polynomials

$$P_n(x) = (1-2x)^n = \sum_{k=0}^{n} 2^k\binom{n}{k}(-x)^k, \qquad P_n(-1) = 3^n, \qquad M_n = 1.$$

Here the partial sums have the form

$$S_n = \frac{1}{3^n}\sum_{k=0}^{n-1}(-1)^k c_k^{(n)} a_k, \quad c_k^{(n)} = \sum_{j=k+1}^{n} 2^j\binom{n}{j}.$$

A third choice is a special family of Chebyshev polynomials, which have other beneficial algebraic properties and are orthogonal. We define these polynomials im-

plicitly by $P_n(\sin^2 t) = \cos(2nt)$ or explicitly by

$$P_n(x) = T_n(1-2x) = \sum_{j=0}^{n} 4^j \frac{n}{n+j} \binom{n+j}{2j} (-x)^j,$$

where $T_n(x) = \cos(n \arccos x)$ are the standard Chebyshev polynomials of degree n on $[-1, 1]$. The polynomials of this sequence are computed by using the recurrence $P_{n+1}(x) = 2(1-2x)P_n(x) - P_{n-1}(x)$, which is initiated by $P_0(x) = 1$ and $P_1(x) = 1 - 2x$. For polynomials chosen in this way, one can show that $P_n(-1) = \frac{1}{2}[(3+\sqrt{8})^n + (3-\sqrt{8})^n]$ and $M_n = 1$. The partial sums

$$S_n = \frac{1}{P_n(-1)} \sum_{k=0}^{n-1} (-1)^k c_k^{(n)} a_k, \quad c_k^{(n)} = \sum_{j=k+1}^{n} 4^j \frac{n}{n+j} \binom{n+j}{2j}, \tag{1.65}$$

converge to the final sum as

$$|S - S_n| \le \frac{2|S|}{(3+\sqrt{8})^n} < \frac{2|S|}{5.828^n},$$

so we need to sum only $n \approx 1.31\,D$ terms for a precision of D significant digits! The coefficients $c_k^{(n)}$ and other constants can be computed iteratively and the whole computation of S_n can be implemented in a very compact algorithm [55]

Input: numbers $a_0, a_1, \ldots, a_{n-1}$ of an alternating series $\sum_{k=0}^{n-1} (-1)^k a_k$
$d = (3+\sqrt{8})^n$; $d = (d + 1/d)/2$;
$b = -1$; $c = -d$; $s = 0$;
for $k = 0$ **step** 1 **to** $n-1$ **do**
  $c = b - c$;
  $s = s + c\,a_k$;
  $b = (k+n)(k-n)b/((k+1/2)(k+1))$;
end
Output: partial sum $S_n = s/d$

This algorithm requires $\mathcal{O}(1)$ of memory and $\mathcal{O}(n)$ of CPU. Similar results can be obtained by using other families of orthogonal polynomials; the paper [55] describes further algorithms in which the coefficients of the partial sums cannot be generated as easily, but yield even faster convergence. For many types of sequences, these algorithms allow us to achieve convergence rates of $|S - S_n| \le |S|/7.89^n$, in some cases even the breath-taking $|S - S_n| \le |S|/17.93^n$. However, they require $\mathcal{O}(n)$ of memory and $\mathcal{O}(n^2)$ of CPU.

1.4.5 Levin's Transformations

Levin's transformations [56] are among the most generally useful, handy, and efficient methods to accelerate the convergence of series by semi-linear extrapolation.

We implement them by using divided differences which are computed recursively:

$$\delta^k f_n = \frac{\delta^{k-1} f_{n+1} - \delta^{k-1} f_n}{t_{n+k} - t_n}, \qquad \delta^0 f_n = f_n,$$

where $t_n = (n + n_0)^{-1}$ and we usually take $n_0 = 0$ or $n_0 = 1$. To compute the extrapolated partial sums we need the partial sums S_n and auxiliary functions ψ which depend on the terms of the sequence and its character (monotonous or alternating). We use the formula

$$S_{k,n} = \delta^k \left(\frac{S_n}{\psi(n)} \right) \left[\delta^k \left(\frac{1}{\psi(n)} \right) \right]^{-1}, \quad k = 1, 2, \ldots \tag{1.66}$$

and take $S_{k,0}$ (with $n_0 = 1$) or $S_{k,1}$ (with $n_0 = 0$) as the extrapolated sum. Levin's transformations differ by the functional forms of ψ. The best known are

$$T:\ \psi(n) = a_n, \qquad U:\ \psi(n) = (n + n_0)a_n, \qquad W:\ \psi(n) = a_n^2/(a_{n+1} - a_n).$$

The T-transformation is best for alternating series in which the partial sums behave as $S_n \sim r^n$, where r is the convergence ratio. The U-transformation works well with monotonous sequences for which $S_n \sim n^{-j}$ applies. The W-transformation can be used in either case regardless of the series type, although it is more sensitive to rounding errors than the U- and T-methods. The U-method is recommended [49] as a reliable way to speed up the summation of any series. The U-transformation, including the extrapolation to the limit and providing the remainder estimates, is implemented in the GSL library [57] in the `gsl_sum_levin_u_accel()` function.

Example Let us sum the slowly converging series $S_n = \sum_{k=0}^{n} (-1)^k/(k+1)$ with the limit $S = \lim_{n\to\infty} S_n = \log 2$. We choose the Levin's T-method and $n_0 = 0$, thus $t_n = n^{-1}$ and $\psi(n) = (-1)^n/(n+1)$. By using (1.66) with $n = 1$ we obtain

$k =$	1	$S_k =$	0.5	$S_{k,1} =$	0.7
	2		0.8333333333333334		0.6923076923076924
	3		0.5833333333333334		0.6932153392330384
	4		0.7833333333333333		0.6931436119116234
	5		0.6166666666666667		0.6931472579962513
	6		0.7595238095238096		0.6931471858853886
	7		0.6345238095238096		0.6931471799439380
	8		0.7456349206349208		0.6931471805844429
	9		0.6456349206349208		0.6931471805603313
	10		0.7365440115440117		0.6931471805598398.

While the partial sums S_k merely hint at convergence, the accelerated sum $S_{k,1}$ at $k = 10$ is already precise to 12 digits. See also Fig. 1.12 (left).

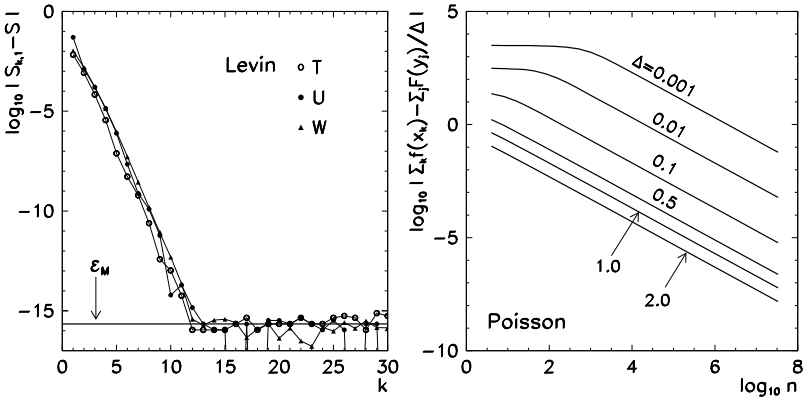


Fig. 1.12 [*Left*] Precision of Levin's methods T, U, and W in accelerating the convergence of the partial sums $S_n = \sum_{k=0}^{n} (-1)^k/(k+1)$ with the limit $S = \lim_{n\to\infty} S_n = \log 2$. [*Right*] Precision of the Poisson's summation formula (1.67) for $f(x) = 1/(1+x^2)$ with different samplings $x_k = k\,\Delta$ on the real axis, where $-n \le \{j,k\} \le n$ and $n \gg 1$

1.4.6 Poisson Summation

Often we encounter sums of function values f at equidistant points on the real axis,

$$S = \sum_{k\in\mathbb{Z}} f(x_k), \quad x_k = k\,\Delta,\ \Delta = x_{k+1} - x_k.$$

Assume that f is differentiable, that it decreases sufficiently fast at infinity, and that its Fourier transform

$$F(y) = \int_{-\infty}^{\infty} f(x)\, \mathrm{e}^{-\mathrm{i}\,2\pi xy}\, \mathrm{d}x$$

exists. The sum of the values $f(x_k)$ and the sum of the transforms $F(y_j)$, computed at $y_j = j/\Delta$, $j \in \mathbb{Z}$, are linked by the Poisson's summation formula (see Fig. 1.12 (right))

$$\sum_{k=-\infty}^{\infty} f(x_k) = \frac{1}{\Delta} \sum_{j=-\infty}^{\infty} F(y_j). \tag{1.67}$$

1.4.7 Borel Summation

Perturbative solutions in classical and quantum mechanics often appear as formally divergent series which, by appropriate means of summation, can yield a conditionally valid final result. Assume we have a sequence $\{a_k\}_{k\in\mathbb{N}_0}$ with the sum $\sum_{k=0}^{\infty} a_k$ that diverges. In the case when all a_k can be explicitly expressed as functions of

the index k, the series can be summed by *Borel resummation*. The original sum is *resummable* in the form

$$S = \lim_{\xi\to\infty} \mathrm{e}^{-\xi} \sum_{n=0}^{\infty} \frac{\xi^n}{n!} S_n, \quad S_n = \sum_{k=0}^{n} a_k, \tag{1.68}$$

if the corresponding limit exists. In practice, the summation parameter ξ is not allowed to go to infinity; rather, we try to locate a range of its values in which the value of S stabilizes when ξ is being increased.

The resummation (1.68) is defined in its differential form. Even more often, we use the integral form, in which the series terms a_k (not the partial sums S_n) are used:

$$S = \int_0^{\infty} \mathrm{e}^{-\xi} \left(\sum_{k=0}^{\infty} \frac{a_k \xi^k}{k!} \right) \mathrm{d}\xi.$$

This form is particularly useful when the function $f(\xi) = \sum_{k=0}^{\infty} a_k \xi^k / k!$ can be written in closed form or is well known in the region where it contributes most significantly to the integral $\int_0^{\infty} f(\xi)\, \mathrm{e}^{-\xi}\, \mathrm{d}\xi$. To do this, we can use the Padé approximation (see Sect. 1.2.2 and Problem 1.5.5).

1.4.8 Abel Summation

Assume that the series $S = \sum_{k=0}^{\infty} a_k$ formally diverges, but that the limit of the expression $S(x) = \sum_{k=0}^{\infty} x^k a_k$ still exists when $x \nearrow 1$. We can also introduce an auxiliary parameter ε such that $x = \mathrm{e}^{-\varepsilon}$ and observe the limit $\varepsilon \searrow 0$. Then the value

$$S_{\mathrm{A}} = \lim_{x\nearrow 1} S(x) = \lim_{x\nearrow 1} \sum_{k=0}^{\infty} x^k a_k = \lim_{\varepsilon\searrow 0} \sum_{k=0}^{\infty} \mathrm{e}^{-\varepsilon k} a_k$$

is called the Abel's generalized sum of the series S. Like with the Borel summation, we introduce an intermediate parameter to regularize a divergent series and then try to sum it in the hope that the generalized limit is finite.

Example The divergent series

$$S = \sum_{k=0}^{\infty} k \cos kr = \mathrm{Re} \sum_{k=0}^{\infty} k\, \mathrm{e}^{\mathrm{i}kr}, \quad 0 < r < 2\pi,$$

has the Abel sum

$$S_{\mathrm{A}} = \mathrm{Re} \lim_{x\nearrow 1} \sum_{k=0}^{\infty} x^k k\, \mathrm{e}^{\mathrm{i}kr} = \mathrm{Re} \lim_{x\nearrow 1} \sum_{k=0}^{\infty} k \left(x\, \mathrm{e}^{\mathrm{i}r} \right)^k = \mathrm{Re} \frac{\mathrm{e}^{\mathrm{i}r}}{(1 - \mathrm{e}^{\mathrm{i}r})^2} = -\frac{1}{4 \sin^2 r/2}.$$

As an exercise, change the parameter $0 < x < 1$ (approach $x \nearrow 1$) and the upper range of the sum, and watch what happens to the values S and S_A.

An analogous method can be used with divergent integrals. If the integral $S = \int_a^\infty f(x)\,\mathrm{d}x$ diverges, its generalized Abel sum is given by

$$S_\mathrm{A} = \lim_{\varepsilon \searrow 0} \int_a^\infty \mathrm{e}^{-\varepsilon x} f(x)\,\mathrm{d}x.$$

Example The divergent integral

$$S = \int_0^\infty \sqrt{x} \cos \alpha x\,\mathrm{d}x, \quad \alpha > 0$$

has the generalized value

$$S_\mathrm{A} = \lim_{\varepsilon \searrow 0} \int_0^\infty \mathrm{e}^{-\varepsilon x} \sqrt{x} \cos \alpha x\,\mathrm{d}x = \mathrm{Re} \lim_{\varepsilon \searrow 0} \int_0^\infty \sqrt{x}\,\mathrm{e}^{-(\varepsilon + \mathrm{i}\alpha)x}\,\mathrm{d}x = -\frac{\sqrt{\pi}}{(2\alpha)^{3/2}}.$$

For exercise, take $\alpha = 1$. Change the upper integration limit in the expressions for S and S_A, and force the limiting parameter ε to approach zero from above. Compare the results to the analytic limit $-\sqrt{\pi}/(2\alpha)^{3/2}$.

1.5 Problems

1.5.1 Integral of the Gauss Distribution

The standard (Gaussian) probability distribution $(1/\sqrt{2\pi})\,\exp(-x^2/2)$ pervades all branches of probability and statistics. Usually we need the probability over an interval, which can be computed by the integral

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp\bigl(-t^2\bigr)\,\mathrm{d}t, \qquad \mathrm{erfc}(x) = 1 - \mathrm{erf}(x). \tag{1.69}$$

Let us restrict the discussion to $x \geq 0$. The erf function monotonously increases and rapidly converges to 1 for large arguments. Its values are tabulated in textbooks, but for general purposes we wish to be able to compute them by exploiting different representations and approximations of the erf and erfc functions. If $\mathrm{erf}(x)$ is known, $\mathrm{erfc}(x)$ is also known (and vice versa), but it is preferable to compute the function having a smaller argument because the error is easier to control. We can switch between calculations of erf and erfc at the point $x \approx 0.4769362762044695$ where $\mathrm{erf}(x) = \mathrm{erfc}(x) = \frac{1}{2}$.

For small x, we can use the power expansion

$$\mathrm{erf}(x) = \frac{2x}{\sqrt{\pi}} \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{k!(2k+1)}.$$

The convergence radius of this series is infinite, but its terms alternate and without acceleration it is not suitable for the computation of $\operatorname{erf}(x)$ at x much larger than 1. For large arguments, $x \gg 1$, the asymptotic expansion is suitable:

$$\operatorname{erfc}(x) \sim \frac{\mathrm{e}^{-x^2}}{x\sqrt{\pi}}\left[1 + \sum_{k=1}^{\infty}(-1)^k \frac{(2k-1)!!}{(2x^2)^k}\right].$$

In the range of x where both the power and the asymptotic expansions provide a poor description of erf, a rational approximation

$$\operatorname{erfc}(x) = \mathrm{e}^{-x^2} \frac{\sum_{k=0}^{7} a_k x^k}{\sum_{k=0}^{7} b_k x^k}\big(1 + \varepsilon(x)\big)$$

may be used. The parameters of this formula are listed in the following table.

k	a_k	b_k
0	1.000000000000013	1.000000000000000
1	1.474885681937094	2.603264849035166
2	1.089127207353042	3.026597029346489
3	0.481934851516365	2.046071816911715
4	0.133025422885837	0.873486411474986
5	0.021627200301105	0.237214006125950
6	0.001630015433745	0.038334123870994
7	−0.000000000566405	0.002889083295887

The relative precision of this parameterization is $\varepsilon(x) < 10^{-14}$ on $x \in [0, 5]$. An elegant, fast, but almost impenetrable implementation of the power expansion of erf and of the computation of erfc by means of tabulated values with a precision of 14 to 16 digits in C can be found in [58]. The algorithms in the GNU C library are also based on rational approximations with many parameters [59, 60].

⊙ Examine the applicability and usefulness of different methods to compute $\operatorname{erf}(x)$. Watch the convergence of the power and asymptotic series. In the latter, sum the terms until the series appears to be converging. Does the convergence improve if Euler's method or Levin's U-transformation is used? By using all three ways of computation, write a program to calculate $\operatorname{erf}(x)$ in double precision on the whole real axis with an error less than 10^{-10}. Try to maximize the speed of the program. Show a comparison table of $\operatorname{erf}(x)$ for $x = 0\,(0.2)\,3$ and $\operatorname{erfc}(x)$ for $x = 3\,(0.5)\,8$.

The integral (1.69) can also be integrated numerically. What would be the required size of the subintervals in the Simpson's formula (Appendix E) in order to achieve a precision that would be comparable to the summation methods used above?

⊕ Write a program to compute the inverse function erf^{-1} with an absolute precision of 10^{-9}. Now that we are in possession of an efficient procedure to compute $\operatorname{erf}(x)$, the inverse can be found by finding the root of the equation

$$F(x) = \operatorname{erf}(x) - y = 0$$

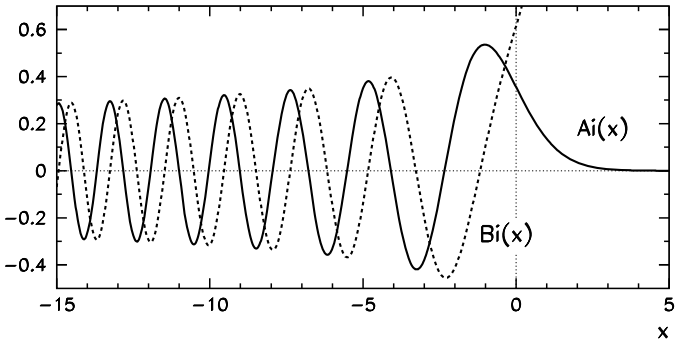


Fig. 1.13 Airy functions Ai and Bi for real arguments. Ai is bound everywhere, while Bi diverges at $x \to \infty$. The zeros of both functions occur only on the negative semi-axis

by using a method to solve non-linear equations. You can use bisection because erf is monotonous, but since the derivative $[\mathrm{erf}(x)]' = 2\mathrm{e}^{-x^2}/\sqrt{\pi}$ is also known, the much faster Newton's method can be used. Compare the computed $\mathrm{erf}^{-1}(y)$ for small values of y to the power expansion [61, 62]

$$\mathrm{erf}^{-1}(y) = \sum_{k=0}^{\infty} \frac{c_k}{2k+1} \left(\frac{\sqrt{\pi}}{2} y \right)^{2k+1}, \quad c_0 = 1,\ c_{k\geq 1} = \sum_{m=0}^{k-1} \frac{c_m c_{k-1-m}}{(m+1)(2m+1)}.$$

1.5.2 Airy Functions

In physics, the Airy functions Ai and Bi (Fig. 1.13) appear in optics and quantum mechanics [63]. They are defined as the independent solutions of the equation

$$y''(x) - xy(x) = 0$$

and have the integral representations

$$\mathrm{Ai}(x) = \frac{1}{\pi} \int_0^{\infty} \cos\bigl(t^3/3 + xt\bigr)\, \mathrm{d}t,$$
$$\mathrm{Bi}(x) = \frac{1}{\pi} \int_0^{\infty} \bigl[\mathrm{e}^{-t^3/3+xt} + \sin\bigl(t^3/3 + xt\bigr)\bigr]\, \mathrm{d}t.$$

For small x the functions Ai and Bi can be expressed by the Maclaurin series

$$\mathrm{Ai}(x) = \alpha f(x) - \beta g(x), \qquad \mathrm{Bi}(x) = \sqrt{3}\bigl[\alpha f(x) + \beta g(x)\bigr],$$

where, at $x = 0$, we have $\alpha = \mathrm{Ai}(0) = \mathrm{Bi}(0)/\sqrt{3} \approx 0.355028053887817239$ and $\beta = -\mathrm{Ai}'(0) = \mathrm{Bi}'(0)/\sqrt{3} \approx 0.258819403792806798$. The series for f and g are

$$f(x) = \sum_{k=0}^{\infty} \left(\frac{1}{3}\right)_k \frac{3^k x^{3k}}{(3k)!}, \qquad g(x) = \sum_{k=0}^{\infty} \left(\frac{2}{3}\right)_k \frac{3^k x^{3k+1}}{(3k+1)!},$$

where $(z)_n = \Gamma(z+n)/\,\Gamma(z)$ and $(z)_0 = 1$.

For large $|x|$ the Airy functions can be approximated by their asymptotic expansions. By substituting $\xi = \frac{2}{3}|x|^{3/2}$ and by using the asymptotic series

$$L(z) \sim \sum_{s=0}^{\infty} \frac{u_s}{z^s}, \qquad P(z) \sim \sum_{s=0}^{\infty} (-1)^s \frac{u_{2s}}{z^{2s}}, \qquad Q(z) \sim \sum_{s=0}^{\infty} (-1)^s \frac{u_{2s+1}}{z^{2s+1}},$$

with coefficients

$$u_s = \frac{\Gamma(3s + \frac{1}{2})}{54^s s! \Gamma(s + \frac{1}{2})},$$

we get, for large positive x,

$$\mathrm{Ai}(x) \sim \frac{\mathrm{e}^{-\xi}}{2\sqrt{\pi} x^{1/4}} L(-\xi), \qquad \mathrm{Bi}(x) \sim \frac{\mathrm{e}^{\xi}}{\sqrt{\pi} x^{1/4}} L(\xi),$$

while for large negative x we have

$$\mathrm{Ai}(x) \sim \frac{1}{\sqrt{\pi}(-x)^{1/4}} \big[\sin(\xi - \pi/4) Q(\xi) + \cos(\xi - \pi/4) P(\xi)\big],$$

$$\mathrm{Bi}(x) \sim \frac{1}{\sqrt{\pi}(-x)^{1/4}} \big[-\sin(\xi - \pi/4) P(\xi) + \cos(\xi - \pi/4) Q(\xi)\big].$$

⊙ Find an efficient procedure to compute the values of the Airy functions Ai and Bi on the real axis with a precision better than 10^{-10} by using a combination of the Maclaurin series and the asymptotic expansion. When estimating the errors, use programs that are capable of arbitrary-precision computations, e.g. MATHEMATICA.

⊕ The zeros of Ai have an important role in mathematical analysis when one tries to determine the intervals containing zeros of other special functions and orthogonal polynomials [64], as well as in physics in computation of energy spectra of quantum systems [34]. Compute the first hundred zeros $\{a_s\}_{s=1}^{100}$ of the Airy function Ai and the first hundred zeros $\{b_s\}_{s=1}^{100}$ of Bi at $x < 0$, and compare the computed values to the formulas

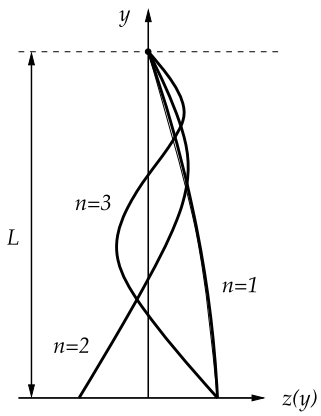
$$a_s = -f\left(\frac{3\pi(4s-1)}{8}\right), \qquad b_s = -f\left(\frac{3\pi(4s-3)}{8}\right), \quad s = 1, 2, \ldots,$$

where f has the asymptotic expansion [22]

$$f(z) \sim z^{2/3}\left(1 + \frac{5}{48}z^{-2} - \frac{5}{36}z^{-4} + \frac{77125}{82944}z^{-6} - \frac{108056875}{6967296}z^{-8} + \cdots\right).$$

1.5.3 Bessel Functions

Solving differential equations in circular or spherical geometry often leads to Bessel functions of the first kind J_ν and second kind Y_ν. A physical example which does not belong to this group but is related to it, is the problem of the oscillations of a freely hanging heavy rope in a constant gravitational field, as shown in the figure below.



A rope of length L is suspended from a ceiling and the origin of the y axis is placed at the free end of the rope. We study small deflections $z(y)$ of the rope in the field of the gravitational acceleration g. Three lowest eigenmodes are shown. The eigenmodes are described by the equation

$$\frac{\mathrm{d}}{\mathrm{d}y}\left(y\,\frac{\mathrm{d}z(y)}{\mathrm{d}y}\right) + \frac{\omega^2}{g}z(y) = 0,$$

where $z(y)$ is the deflection of the rope at y when oscillating with the angular frequency ω. The eigenfrequencies are determined by the equation and the boundary conditions $|z(0)| < \infty$ (deflection bounded at $y = 0$) and $z(L) = 0$ (fixed end at $y = L$). By substitution $t = 2\omega\sqrt{y/g}$ the differential equation can be transformed to $\ddot{z}(t) + \dot{z}(t)/t + z(t) = 0$. The solution of this equation is a linear combination of the Bessel functions J_0 and Y_0. With $\alpha = \sqrt{L/g}$ the boundary conditions become

$$\big|z(t=0)\big| < \infty, \qquad z(t = 2\omega\alpha) = 0.$$

The second condition eliminates Y_0 which is singular at the origin. The condition for the eigenfrequencies is then

$$\omega_n = \xi_{0,n}/(2\alpha), \quad n = 1, 2, \ldots,$$

where $\xi_{0,n}$ is the nth zero of J_0. The values of $J_0(x)$ for small x can be computed by using the power expansion

$$J_0(x) = \sum_{k=0}^{\infty} \frac{(-1)^k (x/2)^{2k}}{(k!)^2},$$

while at large enough arguments, we use the formula

$$J_0(x) = \sqrt{\frac{2}{\pi x}}\big[P(x)\cos(x-\pi/4) + Q(x)\sin(x-\pi/4)\big], \quad x \to \infty,$$

where $P(x)$ and $Q(x)$ are known in terms of the asymptotic series [65]

$$P(x) \sim \sum_{k=0}^{\infty} (-1)^k \frac{a_{2k}^2}{(2k)!(8x)^{2k}}, \qquad Q(x) \sim \sum_{k=0}^{\infty} (-1)^k \frac{a_{2k+1}^2}{(2k+1)!(8x)^{2k+1}}$$

with coefficients $a_n = (2n-1)!!$. (Note $(-1)!! = 0!! = 1$.) For intermediate x we compute $J_0(x)$ by using Bessel functions of higher orders J_n. In the limit $n \to \infty$ and constant x we have $J_n(x) \sim (x/2)^n/n!$, and for $x \sim 1$ and $n > 2x$ this is a very good approximation. Suppose we wish to calculate $J_0(x)$ at given x in arithmetic with precision ε_{M}. With the asymptotic approximation we determine an even $N \gg 2x$ such that $\varepsilon = J_N(x) \ll \varepsilon_{\mathrm{M}}$. This value of $J_N(x)$ is not normalized. Let us denote J_n temporarily by C_n and, for given x, start the iteration

$$C_{N+1} = 0, \qquad C_N = \varepsilon, \qquad C_{n-1}(x) = \frac{2n}{x} C_n(x) - C_{n+1}(x).$$

We obtain $C_0(x)$ which differs by a factor from the true value, $J_0(x) = C_0(x)/A$. The factor A is determined by using the identity $J_0(x) + 2\sum_{k=1}^{\infty} J_{2k}(x) = 1$:

$$A = C_0(x) + 2\sum_{k=1}^{N/2} C_{2k}(x).$$

⊙ Write a procedure to compute J_0 on the real axis to an absolute precision of 10^{-12} and compare it to the result of a tool of higher precision, like MATHEMATICA or MATLAB. Determine the first $N = 10000$ zeros $\{\xi_{0,n}\}_{n=1}^{N}$ of J_0 and compare them to the asymptotic formula [22, 31]

$$\xi_{0,n} \sim \beta + \frac{1}{8\beta} - \frac{31}{384\beta^3} + \frac{3779}{15360\beta^5} - \frac{6277237}{3440640\beta^7} + \cdots, \quad n \to \infty,$$

where $\beta = (n - 1/4)\pi$. Draw the first five eigenmodes of the oscillating rope.

1.5.4 Alternating Series

Some alternating series are literally famous, for example

$$\frac{\pi}{4} = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}, \qquad \log 2 = \sum_{k=0}^{\infty} \frac{(-1)^k}{k+1}, \qquad \frac{\pi^2}{12} = \sum_{k=0}^{\infty} \frac{(-1)^k}{(k+1)^2}.$$

Just as well-known, the natural algorithm has the expansions

$$\log x = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k}(x-1)^k, \quad |x-1| \le 1,\ x \neq 1,$$
$$\log x = 2\sum_{k=0}^{\infty} \frac{1}{2k+1}\left(\frac{x-1}{x+1}\right)^{2k+1}, \quad x > 0, \tag{1.70}$$

that enable us to compute $\log x$ on the whole positive semi-axis.

$\odot$ Compute the sums of these series to a precision of $\varepsilon = 10^{-7}$ by using different methods applicable to alternating series, and study their convergence. (The second series in (1.70) is an exception since it is not alternating.) Compare these methods to simple summation. Make a detailed analysis of the rounding errors first by using single-precision data types (type `float` in C or C++), and then by using double precision (type `double`).

$\bigoplus$ Sum the series given above by using a data type that allows for variable precision (for example, by using the GMP library mentioned in Appendix B.3). Draw a diagram of computational (CPU) times versus the required precision ε in the range $\log_{10}\varepsilon \in [-300, -4]$.

1.5.5 Coulomb Scattering Amplitude and Borel Resummation

An eloquent example [66] of trouble we may face by careless summation of divergent series is the Rutherford scattering amplitude for Coulomb scattering

$$f(\theta) = -\frac{\eta}{2k\sin^2(\theta/2)} \exp\bigl[\mathrm{i}\bigl(2\sigma_0 - \eta\log\sin^2(\theta/2)\bigr)\bigr]. \tag{1.71}$$

We know that the asymptotic expansion of this amplitude is

$$f(\theta) \sim \frac{1}{2\mathrm{i}k}\sum_{l=0}^{\infty}(2l+1)P_l(\cos\theta)\bigl(\mathrm{e}^{2\mathrm{i}\sigma_l} - 1\bigr), \tag{1.72}$$

where σ_l is the phase shift for Coulomb scattering in the partial wave with the orbital angular momentum quantum number l, and P_l is the Legendre polynomial of degree l (see (4.28)). The phase shift in the lth partial wave is

$$\sigma_l = \arg\Gamma(l+1+\mathrm{i}\eta).$$

In the limit $l \to \infty$ the phase shifts behave as $\sigma_l \sim \log l$, while the values $P_l(\cos\theta)$ at fixed θ fade out like $P_l(\cos\theta) \sim l^{-1/2}$. Therefore, for $l \to \infty$, the terms in the sum (1.72) oscillate within an envelope that is proportional to $l^{1/2}$, and the series diverges.

⊙ Sum the series (1.72) directly by summing the terms up to $l_{\max} = 100$ by using $\eta = 1$ and $k = 1\,\mathrm{fm}^{-1}$. Compute the sum for angles $30° \le \theta \le 180°$ in steps of 10° and compare the results to (1.71). Do not calculate the phase shifts by using the gamma function. Rather, use backward recurrence: start with the value of σ_l at $l = l_{\max}$, for which the Stirling approximation applies:

$$\sigma_l \sim \left(l + \frac{1}{2}\right)\beta + \eta \log\alpha - \eta - \frac{\sin\beta}{12\alpha} + \frac{\sin 3\beta}{360\alpha^3} - \frac{\sin 5\beta}{1260\alpha^5} + \frac{\sin 7\beta}{1680\alpha^7} - \frac{\sin 9\beta}{1188\alpha^9} + \cdots,$$

where

$$\alpha = \sqrt{(l+1)^2 + \eta^2}, \qquad \beta = \arctan\left(\frac{\eta}{l+1}\right).$$

Then use the recurrence to compute the phase shifts at lower l:

$$\sigma_l = \sigma_{l+1} - \arctan\left(\frac{\eta}{l+1}\right), \quad l = l_{\max} - 1, l_{\max} - 2, \ldots, 0.$$

Similarly, compute the Legendre polynomials by using the three-term recurrence formula $(l+1)P_{l+1}(x) = (2l+1)xP_l(x) - lP_{l-1}(x)$, which is initialized by $P_0(x) = 1$ and $P_1(1) = x$.

In addition, sum the series by Borel resummation in differential form (1.68). Compare the exact value (1.71) to the numerical one at angles $\theta = 10°$, 60°, 120°, and 150° with the parameter ξ in the range $5 \le \xi \le 100$ in steps of 5.

⊕ Compute the Rutherford sum by applying the Wynn algorithm (1.12). At scattering angles $\theta = 10°$, 60°, 120°, and 150°, calculate the diagonal Padé approximations $[n/n]$ for $0 \le n \le 20$. Stop the recurrence in the algorithm when the denominator of the fraction on the right side of (1.12) becomes equal to zero.

References

1. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 2nd edn. (MIT Press/McGraw-Hill, Cambridge/New York, 2001)
2. D.E. Knuth, *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*, 3rd edn. (Addison-Wesley, Reading, 1997)
3. IEEE Standard 754-2008 for Binary Floating-Point Arithmetic, IEEE 2008; see also http://grouper.ieee.org/groups/754/
4. D. Goldberg, What every computer scientist should know about floating-point arithmetic. ACM Comput. Surv. **23**, 5 (1991). An up-to-date version is accessible as Appendix D of the Sun Microsystems Numerical Computation Guide. http://docs.sun.com/source/806-3568
5. M.H. Holmes, *Introduction to Numerical Methods in Differential Equations* (Springer, New York, 2007) (Example in Appendix A.3.1)
6. D. O'Connor, *Floating Point Arithmetic*. Dublin Area Mathematics Colloquium, March 5, 2005.
7. GNU Multi Precision (GMP), free library for arbitrary precision arithmetic. http://gmplib.org
8. H.J. Wilkinson, *Rounding Errors in Algebraic Processes* (Dover, Mineola, 1994)
9. D.E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 2nd edn. (Addison-Wesley, Reading, 1980)

10. J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, 3rd edn. Texts in Applied Mathematics, vol. 12 (Springer, Berlin, 2002)
11. M.J.D. Powell, *Approximation Theory and Methods* (Cambridge University Press, Cambridge, 1981)
12. C. Hastings, *Approximations for Digital Computers* (Princeton University Press, Princeton, 1955), which is a pedagogical jewel; a seemingly simplistic outward appearance hides a true treasure-trove of ideas yielding one insight followed by another
13. W. Fraser, A survey of methods of computing minimax and near-minimax polynomial approximations for functions of a single variable. J. Assoc. Comput. Mach. **12**, 295 (1965)
14. H.M. Antia, *Numerical Methods for Scientists and Engineers*, 2nd edn. (Birkhäuser, Basel, 2002); see Sects. 9.11 and 9.12
15. R. Pachón, L.N. Trefethen, Barycentric-Remez algorithms for best polynomial approximation in the chebfun system. Oxford University Computing Laboratory, NAG Report No. 08/20
16. G.A. Baker, P. Graves-Morris, *Padé Approximants*, 2nd edn. Encyclopedia of Mathematics and Its Applications, vol. 59 (Cambridge University Press, Cambridge, 1996)
17. P. Wynn, On the convergence and stability of the epsilon algorithm. SIAM J. Numer. Anal. **3**, 91 (1966)
18. P.R. Graves-Morris, D.E. Roberts, A. Salam, The epsilon algorithm and related topics. J. Comput. Appl. Math. **12**, 51 (2000)
19. W. van Dijk, F.M. Toyama, Accurate numerical solutions of the time-dependent Schrödinger equation. Phys. Rev. E **75**, 036707 (2007)
20. K. Kormann, S. Holmgren, O. Karlsson, J. Chem. Phys. **128**, 184101 (2008)
21. C. Lubich, in *Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms*, ed. by J. Grotendorst, D. Marx, A. Muramatsu. NIC Series, vol. 10 (John von Neumann Institute for Computing, Jülich, 2002), p. 459
22. M. Abramowitz, I.A. Stegun, *Handbook of Mathematical Functions*, 10th edn. (Dover, Mineola, 1972)
23. I.S. Gradshteyn, I.M. Ryzhik, *Tables of Integrals, Series and Products* (Academic Press, New York, 1980)
24. P.C. Abbott, Asymptotic expansion of the Keesom integral. J. Phys. A, Math. Theor. **40**, 8599 (2007)
25. M. Battezzati, V. Magnasco, Asymptotic evaluation of the Keesom integral. J. Phys. A, Math. Theor. **37**, 9677 (2004)
26. T.M. Apostol, *Mathematical Analysis*, 2nd edn. (Addison-Wesley, Reading, 1974)
27. P.D. Miller, *Applied Asymptotic Analysis*. Graduate Studies in Mathematics, vol. 75 (Am. Math. Soc., Providence, 2006)
28. A. Erdélyi, *Asymptotic Expansions* (Dover, New York, 1987)
29. R. Wong, *Asymptotic Approximations of Integrals* (SIAM, Philadelphia, 2001)
30. J. Wojdylo, Computing the coefficients in Laplace's method. SIAM Rev. **48**, 76 (2006). While [29] in Sect. II.1 describes the classical way of computing the coefficients c_s by series inversion, this article discusses a modern explicit method
31. F.J.W. Olver, *Asymptotics and Special Functions* (Peters, Wellesley, 1997)
32. S. Wolfram, Wolfram Mathematica. http://www.wolfram.com
33. N. Bleistein, R.A. Handelsman, *Asymptotic Expansion of Integrals* (Holt, Reinhart and Winston, New York, 1975)
34. L.D. Landau, E.M. Lifshitz, *Course in Theoretical Physics, Vol. 3: Quantum Mechanics*, 3rd edn. (Pergamon, Oxford, 1991)
35. P.M. Morse, H. Feshbach, *Methods of Theoretical Physics*, vol. 1 (McGraw-Hill, Reading, 1953)
36. A.D. Polyanin, V.F. Zaitsev, *Handbook of Exact Solutions for Ordinary Differential Equations*, 2nd edn. (Chapman & Hall/CRC, Boca Raton, 2003), p. 215
37. J. Boos, *Classical and Modern Methods in Summability* (Oxford University Press, Oxford, 2000)

38. T.J.I'a. Bromwich, *An Introduction to the Theory of Infinite Series* (Macmillan, London, 1955). The collection of formulas and hints contained in the book remains indispensable
39. A. Sofo, *Computational Techniques for the Summation of Series* (Kluwer Academic/Plenum, New York, 2003)
40. M. Petkovšek, H. Wilf, D. Zeilberger, $A = B$ (Peters, Wellesley, 1996)
41. D.M. Priest, On properties of floating-point arithmetics: numerical stability and the cost of accurate computations. PhD thesis, University of California at Berkeley (1992)
42. N.J. Higham, The accuracy of floating point summation. SIAM J. Sci. Comput. **14**, 783 (1993)
43. J. Demmel, Y. Hida, Accurate and efficient floating point summation. SIAM J. Sci. Comput. **25**, 1214 (2003)
44. W. Kahan, Further remarks on reducing truncation errors. Commun. ACM **8**, 40 (1965)
45. P. Linz, Accurate floating-point summation. Commun. ACM **13**, 361 (1970)
46. T.O. Espelid, On floating-point summation. SIAM Rev. **37**, 603 (1995)
47. I.J. Anderson, A distillation algorithm for floating-point summation. SIAM J. Sci. Comput. **20**, 1797 (1999)
48. G. Bohlender, Floating point computation of functions with maximum accuracy. IEEE Trans. Comput. **26**, 621 (1977)
49. D. Laurie, Convergence acceleration, in *The SIAM 100-Digit Challenge. A Study in High-Accuracy Numerical Computing*, ed. by F. Bornemann, D. Laurie, S. Wagon, J. Waldvögel (SIAM, Philadelphia, 2004), pp. 227–261
50. C. Brezinski, M.R. Zaglia, *Extrapolation Methods* (North-Holland, Amsterdam, 1991)
51. C. Brezinski, Convergence acceleration during the 20th century. J. Comput. Appl. Math. **122**, 1 (2000)
52. E.J. Weniger, Nonlinear sequence transformations for the acceleration of convergence and the summation of divergent series. Comput. Phys. Rep. **10**, 189 (1989)
53. K. Knopp, *Theory and Application of Infinite Series* (Blackie, London, 1951)
54. http://www.nr.com/webnotes?5. The implementation described here is particularly attractive, as it automatically changes N and J such that the required maximum error is achieved most rapidly
55. H. Cohen, F.R. Villegas, D. Zagier, Convergence acceleration of alternating series. Exp. Math. **9**, 4 (2000)
56. H.H.H. Homeier, Scalar Levin-type sequence transformations. J. Comput. Appl. Math. **122**, 81 (2000)
57. GSL (GNU Scientific Library). http://www.gnu.org/software/gsl
58. G. Marsaglia, Evaluation of the normal distribution. J. Stat. Softw. **11**, 1 (2004)
59. J.F. Hart et al., *Computer Approximations* (Wiley, New York, 1968)
60. W.J. Cody, Rational Chebyshev approximations for the error function. Math. Comput. **23**, 631 (1969)
61. J.R. Philip, The function inverfc θ. Aust. J. Phys. **13**, 13 (1960)
62. L. Carlitz, The inverse of the error function. Pacific J. Math. **13** (1963)
63. O. Vallée, M. Soares, *Airy Functions and Applications to Physics* (Imperial College Press, London, 2004)
64. G. Szegö, *Orthogonal Polynomials* (Am. Math. Soc., Providence, 1939)
65. G.N. Watson, *Theory of Bessel Functions* (Cambridge University Press, Cambridge, 1922)
66. W.R. Gibbs, *Computation in Modern Physics* (World Scientific, Singapore, 1994). Sections 12.4 and 10.4

Chapter 2
Solving Non-linear Equations

In all areas of physics, mathematics, and engineering, we need to solve non-linear equations

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}, \quad \boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x}) \in X, \tag{2.1}$$

where X is a vector space ($X = \mathbb{R}^n$ or $X = \mathbb{C}^n$). We assume that the number of unknowns in (2.1) is equal to the number of equations. In this chapter the symbol $\boldsymbol{\xi}$ (ξ in scalar equations) denotes the exact solution of (2.1), so $\boldsymbol{f}(\boldsymbol{\xi}) = \boldsymbol{0}$.

Some famous non-linear equations have analytic solutions, like $x = a + b \sin x$ (Kepler's problem, see Example on p. 62 and [1]) or $x\mathrm{e}^x = a$ (solution given by the Lambert function, see [2]), but in general non-linear equations need to be solved numerically. Efficient methods for solving non-linear equations are built into many numerical libraries, but to achieve greater flexibility, we may be inclined to write our own program.

There are two groups of methods for solving non-linear equations. The first group comprises the methods that can be written in the iterative form

$$\boldsymbol{x}_{k+1} = \boldsymbol{\phi}(\boldsymbol{x}_k), \tag{2.2}$$

where $\boldsymbol{\phi}$ is the iteration function (mapping) and $\boldsymbol{\xi}$ is the fixed point of this mapping. The form of the function $\boldsymbol{\phi}$ in general depends on the function $\boldsymbol{f}$ and its derivatives. The iteration yields successive approximations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ of the exact solution $\boldsymbol{\xi}$, and we terminate the sequence when, at chosen $\varepsilon > 0$, we reach

$$\left\| \boldsymbol{x}_{k+1} - \boldsymbol{\phi}(\boldsymbol{x}_k) \right\| < \varepsilon.$$

These methods rest on the Banach contraction principle (see Appendix D), so they are equally useful and efficient in all Banach spaces: scalar problems, $X = \mathbb{R}$, are easily generalized to vector problems, $X = \mathbb{R}^n$. The typical representatives of this class are the Newton's method (Sect. 2.1.2) and the Müller's method (Sect. 2.1.4). The second group contains the methods that cannot be written in the form (2.2) and are hard to extend to multiple dimensions, e.g. the bisection (Sect. 2.1.1) or the secant method (Sect. 2.1.3).

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9_2,

Order of Convergence In methods of the form (2.2) the convergence of the sequence $\{\boldsymbol{x}_k\}$ to the fixed point $\boldsymbol{\xi}$ of $\boldsymbol{\phi}$ has different speeds. We say that the sequence $\{\boldsymbol{x}_k\}$ converges to $\boldsymbol{\xi}$ at least with order p if for any large enough k we have

$$\|\boldsymbol{e}_{k+1}\| \leq C\|\boldsymbol{e}_k\|^p, \quad \boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{\xi}, \tag{2.3}$$

where $0 < C < 1$ if $p = 1$. Ideally, we always strive to find the largest *order of convergence* p and the smallest *quotient* C. The three lowest orders ($p = 1$, 2, and 3) are called linear, quadratic, and cubic, respectively. If the initial error $\boldsymbol{e}_0$ is known, we may use (2.3) to determine the error of the kth approximation of the solution $\boldsymbol{\xi}$:

$$\|\boldsymbol{e}_k\| \leq C^{(p^k-1)/(p-1)}\|\boldsymbol{e}_0\|^{p^k}.$$

In linear convergence, the approach to the solution is exponential, and the speed of convergence depends exclusively on the magnitude of C. At higher orders, the approach is super-exponential and the speed also depends on p. The order and the quotient depend on the local properties of the mapping $\boldsymbol{\phi}$ (see Appendix D).

Order of the Root To each root of the equation we assign its own *order*. The order r of a root of $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ (i.e. of a zero of $\boldsymbol{f}$) is defined as the largest r for which there exist real positive constants α, β, and ε, such that

$$\alpha\|\boldsymbol{e}\|^r \leq \|\boldsymbol{f}(\boldsymbol{\xi} + \boldsymbol{e})\| \leq \beta\|\boldsymbol{e}\|^r, \quad \|\boldsymbol{e}\| \leq \varepsilon. \tag{2.4}$$

In general the orders of the roots are positive and real, while for analytic functions in the complex plane they are integer and named *root multiplicities*. If $r = 1$, the root is *simple*; otherwise, it is *multiple*. The root orders are important pieces of information in seeking possible or optimal ways to solve the equation.

Sensitivity of the Roots to Perturbations An important issue in solving the equation $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ is the sensitivity of its roots to the perturbation of the function $\boldsymbol{f}$ by another function $\boldsymbol{g}$. Let us consider only the scalar problem

$$f(x) + \varepsilon g(x) = 0, \tag{2.5}$$

where $\varepsilon \in \mathbb{R}$ is the perturbation parameter. Let f be an analytic function with the zero ξ of order m, and let $g(x) \neq 0$ near ξ. In this vicinity of ξ we use the power expansion of f, in which only terms with powers higher than $m - 1$ survive:

$$f(\xi + h) = h^m \frac{f^{(m)}(\xi)}{m!} + \mathcal{O}\big(h^p\big), \quad p > m. \tag{2.6}$$

In the absence of the perturbation ($\varepsilon = 0$), the root $\xi(\varepsilon)$ of (2.5) is simply $\xi = \xi(0)$. When $\varepsilon \neq 0$, we insert the expansion (2.6) into (2.5) and obtain the dependence of the root upon the perturbation: in leading order,

$$\xi(\varepsilon) \doteq \xi + \varepsilon^{1/m}\left(-\frac{m!g(\xi)}{f^{(m)}(\xi)}\right)^{1/m}.$$

Since the mth root is a multi-valued complex function, multiple zeros of $f + \varepsilon g$ may shift far away (in the complex plane) from zeros of the unperturbed function f. We are interested in the effect of numerical errors in $f(x)$ on the precision of the computed zero ξ. If $f(x)$ is known to a precision of δf, it follows that

$$\delta\xi \approx \left| \frac{m!}{f^{(m)}(\xi)} \delta f \right|^{1/m}.$$

The error of the zero, $\delta\xi$, may therefore become very large if the multiplicity of the zero, m, is large. If f is a polynomial, special estimates for the sensitivities of its coefficients to perturbations can be derived: see Sect. 2.4.5.

2.1 Scalar Equations

Finding the roots of the equation $f(x) = 0$, where f is a scalar function and x is its scalar argument, is among the oldest mathematical tasks for which a rich assortment of methods exists; here we discuss only the most famous ones.

2.1.1 Bisection

Bisection is the simplest and the most reliable numerical procedure to find a root of odd order. It is also used as a building block of other methods. Initially we isolate an interval $[a, b]$ on which the function f definitely has a zero. This means that in one of the points from this interval, f changes its sign,

$$f(a) f(b) < 0.$$

(Recall (1.2)!) In subsequent steps we gradually narrow the intervals containing the zero. In the kth step we use the point $c_k = (a_k + b_k)/2$ to divide the interval $[a_k, b_k]$, on which $f(a_k) f(b_k) < 0$, into subintervals $[a_k, c_k]$ and $[c_k, b_k]$. Only one of them contains the zero, and the one for which $f(a_{k+1}) f(b_{k+1}) < 0$ applies, is used as the interval $[a_{k+1}, b_{k+1}]$ to be divided in the next step. This procedure is repeated until the step n at which the condition $|a_n - b_n| \leq \varepsilon$ is fulfilled, and the final approximation of the zero is $\xi \approx (a_n + b_n)/2$. The lengths of the intervals decrease as $e_{k+1} = e_k/2$: the method has linear convergence. The approximate number of steps N needed to determine the zero to precision ε, is

$$N \approx \frac{\log((b-a)/\varepsilon)}{\log 2}.$$

Bisection cannot be used to find zeros of even orders, since $f(a_k) f(b_k) < 0$ does not hold true. Bisection can be generalized to multiple dimensions; see [3, 4].

2.1.2 The Family of Newton's Methods and the Newton–Raphson Method

The Newton family of methods contains methods for solving the equation $f(x) = 0$ based on power expansions of f and seeking its inverse f^{-1} around the point 0. Assume that ξ is the zero of f and S is an open neighborhood of ξ. Let f be at least $(n+1)$-times continuously differentiable in S and at least n-times in its closure $\overline{S}$. The function f can be written as a Taylor series (1.21) where $x, x_0 \in S$ and $x^\star \in [x, x_0]$. If $f' \neq 0$ in S, the implicit-function theorem tells us that to f there corresponds an inverse function g on $\Omega = f(S)$, which is at least $(n+1)$-times continuously differentiable on Ω and at least n-times on the closure $\overline{\Omega}$, and which has the form

$$g(u) = g(v) + g'(v)(u-v) + \cdots + \frac{g^{(n)}(v)}{n!}(u-v)^n + \frac{g^{(n+1)}(\delta)}{(n+1)!}(u-v)^{n+1}.$$

Here $u, v \in \Omega$ and $\delta \in [u, v]$. The derivatives of the inverse function are

$$g^{(n)}\big(f(x)\big) = \lim_{x_0 \to x} \frac{\mathrm{d}^{n-1}}{\mathrm{d}x^{n-1}} \left(\frac{x - x_0}{f(x) - f(x_0)} \right)^n;$$

the first derivative is $g'(f(x)) = 1/f'(x)$, while the second and the third are

$$g''\big(f(x)\big) = -\frac{f''(x)}{f'(x)^3}, \qquad g'''\big(f(x)\big) = \frac{3f''(x)^2 - f'(x)f'''(x)}{f'(x)^5}.$$

We expand g around $v = f(x)$, evaluate it at $u = f(\xi) = 0$, and end up with an expression for ξ which is valid for $f(x) \in \Omega$:

$$\xi = x + \underbrace{\sum_{k=1}^{n} \frac{g^{(k)}(f(x))(-f(x))^k}{k!}}_{\phi_n(x)} + \frac{g^{(n+1)}(\delta)}{(n+1)!}\big(-f(x)\big)^{n+1}, \quad \delta \in \big[0, f(x)\big].$$

The sum $\phi_n(x)$ can be used in the iteration of a method of order $n+1$ like

$$x_{k+1} = \phi_n(x_k),$$

as advertised at the outset (2.2). This iteration operates equally well on the real axis and in the complex plane, and can therefore be used to seek both real and complex roots. (If f is real and we wish to find a complex root, the initial approximation also needs to be complex.) The most important representatives of this family of methods are the Newton–Raphson method ($n = 1$),

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{2.7}$$

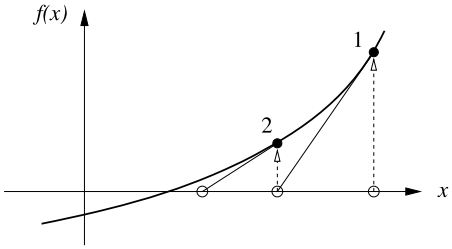


Fig. 2.1 Searching for the approximations of the zero by using the Newton–Raphson method. At each point of the iteration $(x_k, f(x_k))$ we compute the tangent to the function. The intersection of the tangent with the abscissa is the new approximation of the zero x_{k+1}. See also Fig. 2.2 (right)

(Fig. 2.1), and the modified Newton–Raphson method ($n = 2$), with the iteration

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} - \frac{f''(x_k) f(x_k)^2}{2 f'(x_k)^3}.$$

Methods of higher orders can be constructed by augmenting the iteration function, but their applicability is limited as the numerical computation, or even the existence of higher derivatives of f, are questionable. Methods of orders greater than two are rarely used.

The Newton–Raphson method with the iteration (2.7) is the most simple, generally useful and explored method from the Newton family. The search for the zero by using this formula is illustrated in Fig. 2.1. Due to this characteristic graphical interpretation this procedure is also known as the *tangent method*.

It is clear from this construction that the Newton–Raphson iteration has a quadratic convergence if the derivative $f'(x)$ exists in some open neighborhood S of ξ and is continuous and non-zero ($f'(x) \neq 0 \; \forall S$). This can be confirmed by expanding the iteration formula around ξ in powers of the difference $e_k = x_k - \xi$. We get

$$e_{k+1} \approx \frac{f''(\zeta)}{2 f'(x_k)} e_k^2, \quad \zeta = \zeta(x_k) \in S.$$

The method therefore has quadratic convergence when $\max_{x \in S} |f''(x)| \neq 0$. The speed of convergence is essentially driven by the ratio of the second and the first derivative evaluated at the zero of f. If the zero is multiple, the Newton–Raphson method slows down and becomes only linearly convergent. In such cases, quadratic convergence can be restored by writing the iteration (2.7) as

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

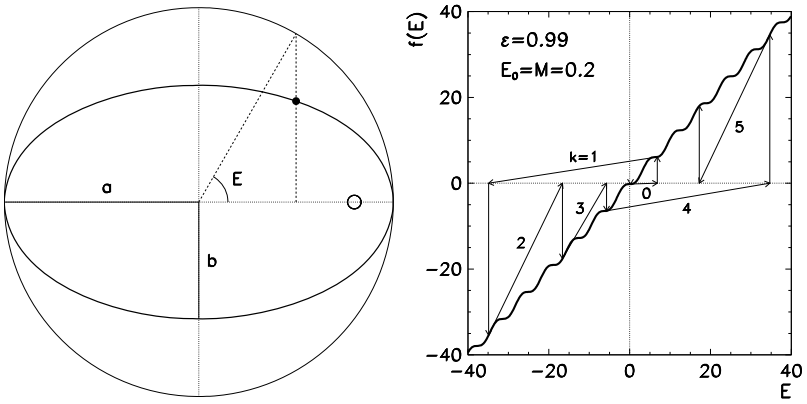


Fig. 2.2 Solving Kepler's equation by Newton's method. [*Left*] The definition of the anomaly E. [*Right*] Poor convergence of the approximations E_k with a strongly eccentric orbit and bad initial approximation E_0. The function $f(E) = E - M - \varepsilon \sin E$ is drawn

where m is the order of the zero of f. If the order of ξ is unknown and there is a chance that the zero is multiple, one should not try to solve $f(x) = 0$ but rather

$$u(x) = 0, \qquad u(x) = \frac{f(x)}{f'(x)}.$$

When u is inserted in (2.7), we obtain an iteration of the form

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

which also accommodates multiple zeros.

Under certain assumptions about f and the initial value x_0, the Newton–Raphson sequence is locally convergent, but it does not converge globally (for arbitrary initial values); see Example below. For some classes of f Newton's method can be tuned to become globally convergent [5]. Sufficient conditions for convergence are specified by Ostrowski theorems (Appendix D).

Example (Adapted from [6, 7]) Kepler's equation

$$E = M + \varepsilon \sin E \tag{2.8}$$

connects the parameters of the planet's orbital motion around the Sun: eccentric anomaly E, average anomaly M, and eccentricity $\varepsilon = \sqrt{1 - b^2/a^2}$ (Fig. 2.2 (left)). The time dependence of M is given by $M(t) = 2\pi(t - t_\mathrm{p})/T$, where T is the period and t_p is the time of the passage through the perihelium (the point on the elliptic orbit closest to the Sun). Various forms of analytic solutions exist [1].

Equation (2.8) can be solved by simple iteration, $E_{k+1} = M + \varepsilon \sin E_k$, which converges very slowly. It is therefore much better if the equation is rewritten in the

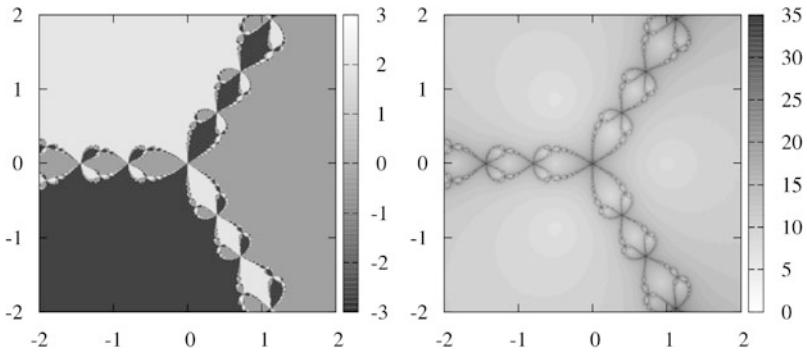


Fig. 2.3 Newton's method used to solve $z^3 - 1 = 0$ in the complex plane. [*Left*] The phase of the final value z as a function of the initial approximation given by the values $\mathrm{Re}\, z$ in $\mathrm{Im}\, z$ on the abscissa and the ordinate. The three shaded areas are the *basins of attraction* corresponding to the phases 0, $2\pi/3 \approx 2.094395$, and $-2\pi/3 \approx -2.094395$ to which the individual initial approximation is "attracted" [8]. [*Right*] The number of iterations needed to achieve convergence, as a function of the initial approximation

Newton form (2.7) with the function $f(E) = E - M - \varepsilon \sin E$. We get

$$E_{k+1} = \frac{M - \varepsilon[E_k \cos E_k - \sin E_k]}{1 - \varepsilon \cos E_k}, \quad k = 0, 1, 2, \ldots .$$

With a prudent choice of the initial approximation the sequence $\{E_k\}$ converges very rapidly: for $\varepsilon = 0.95$, $M = 245°$, and the initial approximation $E_0 = 215°$, we obtain the solution $E \approx 3.740501878977461$ to machine precision in just three steps; the simple iteration would require 131 steps for the same precision.

For small eccentricities, $\varepsilon \ll 1$, Newton's method is rather insensitive to the initial approximation, as in such cases $E = M + \varepsilon \sin E \approx M$, and the simple guess $E_0 = M$ does the job. For large ε, however, the sequence of the approximations $\{E_k\}$ with a poorly chosen initial value E_0 may diverge. Assume that we always set $E_0 = M$ initially. If $(\varepsilon, M) = (0.991, 0.13\pi)$ or $(0.993, 0.13\pi)$, the sequence $\{E_k\}$ converges, but for the values $(0.992, 0.13\pi)$ it diverges! Similar behavior can be observed near $(\varepsilon, M) = (0.99, 0.2)$ (Fig. 2.2 (right)). The reason for the divergence is the bad choice of E_0 combined with the tricky form of the function $f(E) = E - M - \varepsilon \sin E$ with almost horizontal parts where the tangent to f is "thrown" far away (see the jumps at $k = 1$ and $k = 4$). A good approximation for any ε, which can be used to avoid the pitfalls seen above, is $E_0 = \pi$ [6, 7].

Example We use Newton's method to compute the complex roots of $z^3 - 1 = 0$. The analytic solutions are 1, $\mathrm{e}^{\mathrm{i}2\pi/3}$, and $\mathrm{e}^{-\mathrm{i}2\pi/3}$. We choose the initial approximation $z_0 = x_0 + \mathrm{i}\, y_0 \in [-2, 2] \times [-2, 2]$ and observe the value of z at the end of the iteration $z_{k+1} = z_k - f(z_k)/f'(z_k)$, where $f(z) = z^3 - 1$. Figure 2.3 (left) shows the phase of the final value of z, while Fig. 2.3 (right) shows the number of steps needed to achieve convergence to machine precision.

2.1.3 The Secant Method and Its Relatives

Together with bisection, the secant method and its variants are the oldest known iterative procedures. From the pairs $(x_{k-1}, f(x_{k-1}))$ and $(x_k, f(x_k))$ with oppositely signed $f(x_{k-1})$ and $f(x_k)$ we determine the straight line intersecting the abscissa at x_{k+1}, and we compute the value of f at this very point, $f(x_{k+1})$. The process is repeated on the subinterval on which the function's values at its boundary points are oppositely signed, until convergence is achieved.

This method of finding the zero (regula falsi, see below) is just a special case in the family of methods based on the Newton–Raphson iteration where the exact derivative of the function is replaced by the approximate one:

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

When this approximation is inserted in (2.7), we obtain the iteration

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})} = \frac{f(x_k)x_{k-1} - f(x_{k-1})x_k}{f(x_k) - f(x_{k-1})}. \tag{2.9}$$

To compute x_{k+1} we need the information from *two* previous points, hence this method cannot be written in the one-point form $x_{k+1} = \phi(x_k)$. The iteration (2.9) can be used in various ways which yield the secant method, the method of false position (regula falsi), and the chord method. Among these, only the secant method has super-linear convergence with the optimal order $(1 + \sqrt{5})/2 \approx 1.618$. All other methods from this family have approximately linear convergence.

Secant Method This method does not require specific initial conditions (for example, it does not require the function to have opposite signs at the boundaries of the interval) and therefore does not always converge, in particular if the initial values are far from the zero. The secant method works best for functions with simple (not multiple) zeros. We use (2.9) directly; see Fig. 2.4 (left).

Regula Falsi To use this method the function should have opposite signs at the boundary points of $[a, b]$, $f(a)f(b) < 0$. Then (2.9) is used in the form

$$c = \frac{f(b)a - f(a)b}{f(b) - f(a)}, \tag{2.10}$$

where c is the new approximation of the zero. According to the value $f(c)$ we choose the next interval on which the zero is located, and assign $a = c$ or $b = c$, just like with bisection (see Fig. 2.4 (right)). With minimal modifications of (2.10) we can make the regula falsi method achieve super-linear convergence: see [9].

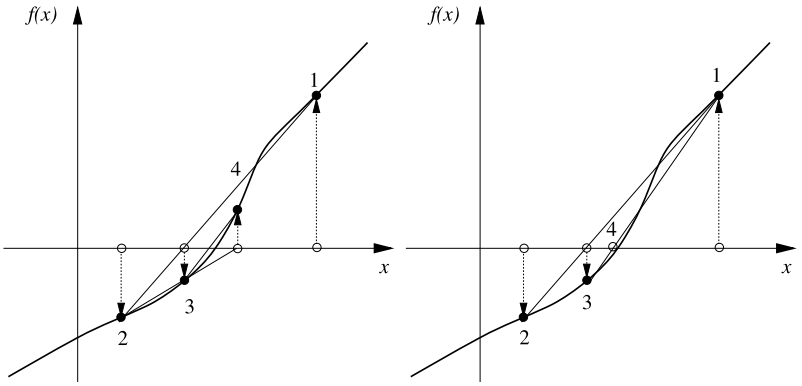


Fig. 2.4 Finding the approximations of the zeros of f by using the methods from the family defined by (2.9). [*Left*] Secant method. [*Right*] Method of false position

Chord Method This method is a simplified version of the regula falsi method. We also require $f(a)f(b) < 0$ (there is at least one zero on the interval) and that on $[a, b]$ the sign of the second derivative of f *does not* change. One of the ends of the interval may therefore be fixed and the iteration (2.9) can be refurbished according to the endpoint at which f satisfies $f(x)f''(x) > 0$. Two possible iteration procedures follow. If the point $x = a$ is fixed, we iterate

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

while if $x = b$ is fixed,

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

2.1.4 Müller's Method

Like the secant methods, Müller's method is a one-point iteration method *with memory*: in computing the approximation of the zero in the current step, one uses the interpolation of the values $f(x_k)$ and (approximations of) derivatives $f^{(j)}(x_k)$ at points accessed in previous steps. Using only the values would be best, but then at most quadratic convergence can be achieved. If derivatives are used to improve the interpolation, arbitrary orders of convergence can be achieved in principle, but this is not practical. For details, see [10, 11].

The most well-known form of the Müller method can be derived by generalizing the secant method: instead of the linear interpolation through the last two points, we use the quadratic interpolation through the last three points of the iteration $(x_k, f(x_k))$, $(x_{k-1}, f(x_{k-1}))$, and $(x_{k-2}, f(x_{k-2}))$. The next approximation

of the zero ξ of f is one of the zeros $x_\pm$ of the interpolation polynomial

$$y(x) = a(x - x_k)^2 + 2b(x - x_k) + f(x_k)$$

with the coefficients

$$a = \left[\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} - \frac{f(x_k) - f(x_{k-2})}{x_k - x_{k-2}} \right] \frac{1}{x_{k-1} - x_{k-2}},$$

$$b = \frac{1}{2} \left[\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} + \frac{f(x_k) - f(x_{k-2})}{x_k - x_{k-2}} - \frac{f(x_{k-1}) - f(x_{k-2})}{x_{k-1} - x_{k-2}} \right].$$

The zeros of the interpolation polynomial are

$$x_\pm = x_k + \frac{1}{a} \left[-b \pm \sqrt{b^2 - a f(x_k)} \right]$$

and are related by $x_+ x_- = (a x_k^2 - 2b x_k + f(x_k))/a$ and $(x_+ - x_k)(x_- - x_k) = f(x_k)/a$. The iteration step of the Müller method is then

$$x_{k+1} = x_\pm = x_k - \frac{f(x_k)}{b \pm \sqrt{b^2 - a f(x_k)}}.$$

We choose the sign which corresponds to the largest magnitude of the denominator since this minimizes the rounding errors. The iteration can be initialized with an arbitrary triplet of points x_0, x_1, and x_2 in the vicinity of ξ, but it is advisable to space them uniformly.

The Müller method also allows us to search for complex zeros, with an important advantage over the secant and Newton method: we enter the complex plane even if the initial approximation is on the real axis. On the other hand, this property may be a nuisance if we are searching only for strictly real zeros.

In the case that f is at least three times continuously differentiable in the vicinity of its simple zero ξ, and that the initial approximations x_0, x_1, and x_2 have been chosen close enough to ξ, one can show that for the sequence $\{x_k\}$, obtained in the Müller iteration the following holds true:

$$\lim_{k \to \infty} \frac{|x_k - \xi|}{|x_{k-1} - \xi|^p} = \left| \frac{1}{6} \frac{f'''(\xi)}{f'(\xi)} \right|^{(p-1)/2},$$

where the order of convergence is $p \approx 1.83929$. Since the coefficient $2b$ is equal to the derivative of the interpolation polynomial at x_k, Müller's iteration can also be defined—in analogy to the Newton method—as

$$x_{k+1} = x_k - \frac{f(x_k)}{2b}.$$

This simple iteration has the same order of convergence, but it becomes useless for $b \approx 0$. It can be put to good use, however, in the immediate vicinity of the real or complex zero or in cases where strictly real zeros are searched and we wish to avoid the jumps into the complex plane typical of the Müller method.

Further Reading In iterative approaches in this section, we have devoted more attention to the discussion of one-point methods, in which the values of the function (and its derivatives) at a single point are needed to compute the next approximation of the zero. This is the recommended and the most widely used practice. Methods with memory and multi-step methods are discussed in [10]. In general these methods are only locally convergent, and locating a good initial approximation of the zero is imperative. In very specific cases (e.g. in polynomial equations) globally convergent "*sure-fire*" methods exist that withstand practically arbitrary initial approximations. For a deeper insight see [5, 10, 12, 13].

2.2 Vector Equations

In contrast to scalar equations, the solutions of vector equations are sought in n-dimensional ($n > 1$) vector spaces. Equation (2.1) should therefore be read as

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}, \qquad \boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n \quad \big(\text{or } \boldsymbol{f} : \mathbb{C}^n \to \mathbb{C}^n\big), \qquad \boldsymbol{f} : \boldsymbol{x} \mapsto \boldsymbol{f}(\boldsymbol{x}).$$

The argument $\boldsymbol{x}$ of the function $\boldsymbol{f}$ and the solution of the equation $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$, its root $\boldsymbol{\xi}$, are vectors, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ and $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_n)^{\mathrm{T}}$. Similarly, a n-dimensional vector is used to represent the n scalar functions of $\boldsymbol{x}$:

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \\ \vdots \\ f_n(\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{pmatrix}.$$

In iterative solution of vector equations the approach to the root is no longer confined to a straight line. This introduces severe algorithmic complications and restricts the set of applicable methods. Here we follow [14] and introduce a generalization of the Newton–Raphson and secant method to multiple dimensions.

2.2.1 Newton–Raphson's Method

In the vector variant of the Newton–Raphson method we formally proceed in the same spirit as in the scalar case. In the vicinity of the root $\boldsymbol{\xi}$ the components of the function $\boldsymbol{f}$ have the Taylor expansions

$$0 = f_i(\boldsymbol{\xi}) \approx f_i(\boldsymbol{x}) + \sum_j \frac{\partial f_i(\boldsymbol{x})}{\partial x_j}(\xi_j - x_j) + \mathcal{O}\big(|\boldsymbol{\xi} - \boldsymbol{x}|^2\big). \tag{2.11}$$

By using the Jacobi matrix of derivatives of $\boldsymbol{f}$ with respect to $\boldsymbol{x}$,

$$J(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_n(\boldsymbol{x})}{\partial x_n} \end{bmatrix},$$

(2.11) can be rewritten as

$$\mathbf{0} = \boldsymbol{f}(\boldsymbol{\xi}) \approx \boldsymbol{f}(\boldsymbol{x}) + J(\boldsymbol{x})(\boldsymbol{\xi} - \boldsymbol{x}) + \cdots .$$

If $\boldsymbol{x}$ is close enough to $\boldsymbol{\xi}$, we may neglect the quadratic term in the expansion, and proclaim the solution $\boldsymbol{\zeta}$ of the equation $\boldsymbol{f}(\boldsymbol{x}) + J(\boldsymbol{x})(\boldsymbol{\zeta} - \boldsymbol{x}) = \mathbf{0}$ as the improved approximation of the root, $\boldsymbol{\zeta} = \boldsymbol{x} - J^{-1}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x})$. This leads to the general iteration step

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - J^{-1}(\boldsymbol{x}_k)\boldsymbol{f}(\boldsymbol{x}_k), \quad k = 0, 1, 2, \ldots, \tag{2.12}$$

where k is the iteration index.

It is sometimes very hard or even impossible to compute the Jacobi matrix analytically. In such cases, we resort to approximations of derivatives, like

$$\frac{\partial f_i(\boldsymbol{x})}{\partial x_j} \approx \frac{f_i(\boldsymbol{x} + h_j \boldsymbol{e}_j) - f_i(\boldsymbol{x})}{h_j},$$

where $\boldsymbol{e}_j$ is the unit vector along the jth coordinate axis (a n-dimensional vector with the jth component set to 1). If the chosen h_j is too small, the errors in computing the values of $\boldsymbol{f}(\boldsymbol{x})$ may become very large; if the h_j is too large, the derivative is locally poorly approximated. The optimal h_j can be estimated by

$$h_j^{(\mathrm{opt})} \left\| \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_j} \right\| \approx \sqrt{\varepsilon}\, \|\boldsymbol{f}(\boldsymbol{x})\| \quad \forall j,$$

where ε is the relative precision of the computation of the values $\boldsymbol{f}(\boldsymbol{x})$, for example, $\varepsilon = \varepsilon_{\mathrm{M}}$. By using this estimate, both $\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{f}(\boldsymbol{x} + h_j \boldsymbol{e}_j)$ are precise to about half of the digits allowed by the floating-point arithmetic being used.

The vector Newton method (2.12) is exact for affine functions $\boldsymbol{f}$. This means that the iterated values $\boldsymbol{x}$ are invariant with respect to a linear transformation $\boldsymbol{f} \to A\boldsymbol{f}$ with an arbitrary non-singular square matrix A, and that the iterated values of the method with the function $\boldsymbol{f}(A\boldsymbol{x})$ can be obtained by simply applying the method with the function $\boldsymbol{f}(\boldsymbol{x})$, and multiplying the vector of the iterated values by A^{-1}. The subsequent approximations converge to the root under the conditions of the Newton–Kantorovich theorem (see Appendix D and [5]).

2.2.2 Broyden's (Secant) Method

In vector methods of the secant type we exploit the idea that at each point of the iteration $\boldsymbol{x}_k$ the function $\boldsymbol{f}$ can locally be approximated by a linear model,

$$\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{f}(\boldsymbol{x}_k) + B_k(\boldsymbol{x} - \boldsymbol{x}_k).$$

From the viewpoint of the power expansion, the optimal choice is $B_k = J(\boldsymbol{x}_k)$, which corresponds to the Newton–Raphson method. But the matrix B_k may also be understood as an approximation of the Jacobi matrix which, at some step of the iteration, can be determined by the linear equation

$$B_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) = \boldsymbol{f}(\boldsymbol{x}_k) - \boldsymbol{f}(\boldsymbol{x}_{k-1}).$$

If the system of equations is linear, B_k is constant, while in non-linear systems, B_k changes during the iteration $k = 0, 1, \ldots$. The changes (or *updates*) of B_k may be very general or quite small. Various methods are characterized by different ways of constructing B_k. The most useful are known as *Broyden's methods* [15–17]; they are the generalizations of the secant method (2.9) to vector equations.

Broyden's methods belong to the class of methods in which the matrix B_k is only minimally changed during the steps of the iteration (*least-change secant update methods*). A nice, physically intuitive insight into the genesis of these methods is given by their author in [18]. Empirically, "good" and "bad" versions of the Broyden methods were discovered. In the *good Broyden method*, the approximation B_k of the Jacobi matrix in a single iteration step is calculated by

$$B_k = B_{k-1} + \frac{(\boldsymbol{y}_{k-1} - B_{k-1}\boldsymbol{s}_{k-1})\boldsymbol{s}_{k-1}^{\mathrm{T}}}{\boldsymbol{s}_{k-1}^{\mathrm{T}}\boldsymbol{s}_{k-1}},$$

where

$$\boldsymbol{s}_{k-1} = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}, \qquad \boldsymbol{y}_{k-1} = \boldsymbol{f}(\boldsymbol{x}_k) - \boldsymbol{f}(\boldsymbol{x}_{k-1}).$$

The update of B_{k-1} at each step has rank one: this means that B_{k-1} does not change in any direction that is orthogonal to the vector $\boldsymbol{s}_{k-1}$. By using the inverse of the updated matrix, B_k^{-1}, we then proceed with the next step of the iteration

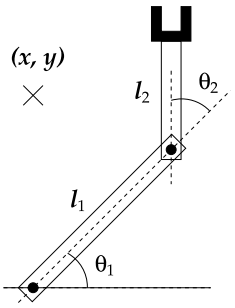
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - B_k^{-1}\boldsymbol{f}(\boldsymbol{x}_k). \tag{2.13}$$

In the program, this is implemented as a system of equations $B_k\boldsymbol{x}_{k+1} = B_k\boldsymbol{x}_k - \boldsymbol{f}(\boldsymbol{x}_k)$, which we solve for $\boldsymbol{x}_{k+1}$. We start the iteration by specifying the initial approximation of the solution $\boldsymbol{x}_0$, while B_0 is set to be the exact or the approximate Jacobi matrix $J(\boldsymbol{x}_0)$. One can also use the updates of the inverse matrix

$$B_k^{-1} = B_{k-1}^{-1} + \frac{(\boldsymbol{s}_{k-1} - B_{k-1}^{-1}\boldsymbol{y}_{k-1})\boldsymbol{s}_{k-1}^{\mathrm{T}}B_{k-1}^{-1}}{\boldsymbol{s}_{k-1}^{\mathrm{T}}B_{k-1}^{-1}\boldsymbol{y}_{k-1}}, \tag{2.14}$$

which is only allowed if $\boldsymbol{s}_{k-1}^{\mathrm{T}} B_{k-1}^{-1} \boldsymbol{y}_{k-1} \neq 0$. We need to calculate only one inverse of the matrix, B_0^{-1}, which we then update as in (2.14) and iterate (2.13).

If $\boldsymbol{f}$ is continuously differentiable, and we see that the approximations $\boldsymbol{x}_k$ converge to the root $\boldsymbol{\xi}$ (where $\boldsymbol{s}_k$ from some k on are linearly independent), we also find $\lim_{k\to\infty} B_k = J(\boldsymbol{\xi})$: in the iteration B_k approaches the Jacobi matrix $J(\boldsymbol{\xi})$ (Fig. 2.5 (right)). For a precise formulation of the theorem and its proof see [19].



Example In this classic problem, we seek the angles θ_1 and θ_2 at which the robot's arm with handles l_1 and l_2 reaches to the point (x, y). We are solving a system of non-linear equations $f_1(\theta_1, \theta_2) = 0$, $f_2(\theta_1, \theta_2) = 0$, with

$$f_1(\theta_1, \theta_2) = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) - x,$$
$$f_2(\theta_1, \theta_2) = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) - y.$$

The Jacobi matrix corresponding to this system is

$$J(\theta_1, \theta_2) = \begin{pmatrix} \partial f_1/\partial\theta_1 & \partial f_1/\partial\theta_2 \\ \partial f_2/\partial\theta_1 & \partial f_2/\partial\theta_2 \end{pmatrix}.$$

There may be an unique solution, multiple solutions, or the solution may not even exist. Let the lengths of the arms be $l_1 = l_2 = 1$ and the initial position of the arm be $(\theta_1, \theta_2) = (20°, 20°)$. We wish to reach the point $(x, y) = (0.7, 1.2)$.

The system can be solved by Broyden's updates (2.14) in the iteration (2.13) or its improved version (2.16) and (2.15). Since at each point $\boldsymbol{\theta} = (\theta_1, \theta_2)^{\mathrm{T}}$ we know the explicit form of the Jacobi matrix, the solution can also be computed by Newton's method (2.12). Figure 2.5 (left) shows the convergence of the sequence of approximations $\boldsymbol{\theta}_k$ for both iterations. The final position of the robotic arm is $(\theta_1, \theta_2) \approx (105.746°, 267.994°)$; if the initial approximation is $(20°, 100°)$ we obtain the second solution $\approx (13.741°, 92.006°)$. Figure 2.5 (right) shows the approach of the Broyden matrix to the exact Jacobi matrix.

For a successful iteration it is imperative that the matrix B_k is non-singular, which can be controlled by monitoring its determinant [20]. During the iteration, the determinant changes according to

$$\det(B_{k+1}) = \frac{\boldsymbol{s}_k^{\mathrm{T}} B_k^{-1} \boldsymbol{y}_k}{\boldsymbol{s}_k^{\mathrm{T}} \boldsymbol{s}_k} \det(B_k).$$

In the kth step, B_k may become singular and remain singular in all subsequent steps if $\boldsymbol{s}_k \perp B_k^{-1} \boldsymbol{y}_k$. Even approximate orthogonality and near-vanishing of the determinant may have disastrous consequences. One obtains a much better control of the stability of the Broyden method if the iteration is generalized as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \lambda_k B_k^{-1} \boldsymbol{f}(\boldsymbol{x}_k), \tag{2.15}$$

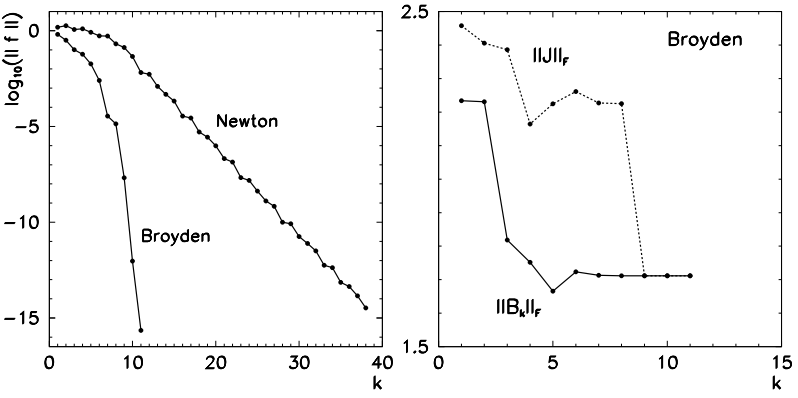


Fig. 2.5 The solution of the robotic arm problem. [*Left*] The deviation of $\boldsymbol{f}(\boldsymbol{\theta}_k)$ from the values $\mathbf{0}$ as a function of the iteration index in the Newton and Broyden method. (Other initial conditions may yield a rather different figure.) [*Right*] The convergence of the Broyden matrix to the exact Jacobi matrix as a function of the iteration index (measured in the Frobenius matrix norm)

$$B_k = B_{k-1} + \eta_{k-1} \frac{(\boldsymbol{y}_{k-1} - B_{k-1}\boldsymbol{s}_{k-1})\boldsymbol{s}_{k-1}^{\mathrm{T}}}{\boldsymbol{s}_{k-1}^{\mathrm{T}}\boldsymbol{s}_{k-1}}. \tag{2.16}$$

The control parameters η_k and λ_k can be adjusted during the iteration. Reasonable values are $\eta_k \in (0, 2)$ and $\lambda_k \in (0, 2)$. If η_k is calculated as

$$\eta_k = \begin{cases} 1; & |\gamma_k| \geq 1/10, \\ \frac{1-\mathrm{sign}(\gamma_k)/10}{1-\gamma_k}; & |\gamma_k| < 1/10, \end{cases} \qquad \gamma_k = \frac{\boldsymbol{s}_k^{\mathrm{T}} B_k^{-1} \boldsymbol{y}_k}{\boldsymbol{s}_k^{\mathrm{T}} \boldsymbol{s}_k},$$

which bounds $\eta_k \in [0.9, 1.1]$, it turns out [20] that the non-singularity of B_k is guaranteed; in addition, we establish $|\det(B_{k+1})| \geq |\det(B_k)|/10$. The sequence of approximations $\boldsymbol{x}_k$ or the deviations from the root $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{\xi}$ obtained by the generalized Broyden scheme satisfy

$$\frac{\|\boldsymbol{e}_{k+1}\|_2}{\|\boldsymbol{e}_k\|_2} \leq \frac{\varepsilon_k + |\lambda_k - 1|}{1 - \varepsilon_k},$$

where $\{\varepsilon_k\}$ is a sequence for which $\lim_{k\to\infty} \varepsilon_k = 0$. In the basic Broyden method ($\lambda_k = 1$) the right side of the inequality vanishes as $k \to \infty$, and the method is super-linear. This applies only near the solution. Farther away, it makes more sense to vary λ_k within a *line search*: We seek the largest λ_k such that

$$\left\| \boldsymbol{f}(\boldsymbol{x}_k - \lambda_k \boldsymbol{d}_k) \right\|_2 < \left\| \boldsymbol{f}(\boldsymbol{x}_k) \right\|_2, \quad \boldsymbol{d}_k = B_k^{-1} \boldsymbol{f}(\boldsymbol{x}_k),$$

or restrict the parameter λ_k to assume only the values of negative powers of 2,

$$\lambda_k = 2^{-j}, \quad j = \min\left\{ i \geq 0 : \left\| \boldsymbol{f}\left(\boldsymbol{x}_k - 2^{-i}\boldsymbol{d}_k\right) \right\|_2 < \left\| \boldsymbol{f}(\boldsymbol{x}_k) \right\|_2 \right\}.$$

We can also search for λ that minimizes $g(\lambda) = \|\boldsymbol{f}(\boldsymbol{x}_k - \lambda \boldsymbol{d}_k)\|_2$, but if this is implemented carelessly, we may face large numerical costs and slow convergence. A very good estimate for the optimal value of the parameter λ can be obtained by approximating $g(\lambda)$ by a parabolic interpolant through the points $g_0 = g(0)$, $g_t = g(t)$, $t \in (0, 1)$, and $g_1 = g(1)$:

$$g(\lambda) \approx g_0 + \frac{t^2(g_1 - g_0) + g_0 - g_t}{t(t-1)}\lambda - \frac{t(g_1 - g_0) + g_0 - g_t}{t(t-1)}\lambda^2.$$

Assume that λ at which $g(\lambda)$ has a minimum,

$$\lambda = \frac{1}{2}\frac{t^2(g_1 - g_0) + g_0 - g_t}{t(g_1 - g_0) + g_0 - g_t},$$

is a good approximation for the optimal value. If this approximation lies outside the interval $(0, 1)$, we discard it and make the next step by using $\lambda = 1$.

2.3 Convergence Acceleration ⋆

The convergence of iterative methods (and, more generally, all convergent sequences) can be accelerated by using *Aitken's procedures*. In the following, we present the most commonly known Aitken methods to accelerate the iterations or sequences with linear or quadratic convergence.

Assume we are seeking the solution ξ of the equation $f(x) = 0$. If the sequence $\{x_k\}$ of approximations of ξ converges linearly, we know that the differences $e_k = x_k - \xi$ near the root satisfy $e_{k+1} = \alpha e_k$ and $e_{k+2} = \alpha e_{k+1}$. When the former of these equations is divided by the latter, we get $e_{k+1}^2 = e_k e_{k+2}$. We insert $e_k = x_k - \xi$, solve for ξ, and get an improved approximation of the root,

$$X_k = x_k - \frac{(x_{k+1} - x_k)^2}{x_{k+2} - 2x_{k+1} + x_k}, \tag{2.17}$$

which can represent the final result or the initial value for the next step. (Compare (2.17) and (1.58)!) In the optimal case, the sequence $\{X_k\}$ converges *quadratically* to ξ. In general, however, one can guarantee only that the sequence $\{X_k\}$ converges *faster* than $\{x_k\}$. Due to its characteristic form of the numerator in (2.17) this method is known as the Aitken Δ^2 process.

Typically we do not accelerate quadratically convergent sequences. Yet this does become sensible when a single iteration step implies excessive numerical costs and we wish to squeeze as much as possible from the numerical procedure. As in the linear case, we divide the convergence equations in two subsequent steps, $e_{k+1} = \alpha e_k^2$ and $e_{k+2} = \alpha e_{k+1}^2$, thereby eliminating α. We get $e_{k+1}^3 = e_{k+2} e_k^2$, into which we insert $e_k = x_k - \xi$, and end up with a quadratic equation for ξ,

$$(3x_{k+1} - x_{k+2} - 2x_k)\xi^2 + \left(-3x_{k+1}^2 + 2x_k x_{k+2} + x_k^2\right)\xi + x_{k+1}^2 - x_{k+2}x_k^2 = 0.$$

To compute the improved approximation of the root, we take the average of its two solutions:

$$X_k = \frac{\xi_+ + \xi_-}{2} = \frac{x_k^2 + 2x_k x_{k+2} - 3x_{k+1}^2}{4x_k - 6x_{k+1} + 2x_{k+2}}.$$

2.4 Polynomial Equations of a Single Variable

Polynomials [21] are frequent guests in scientific and technical applications. In most cases, we are dealing with real polynomials of degree n, defined as

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = \sum_{k=0}^{n} a_k x^k, \tag{2.18}$$

where a_k are real coefficients and $a_n \neq 0$. The zeros (roots) of polynomial equations $p(x) = 0$ are assigned orders (multiplicities), just as the roots of equations $f(x) = 0$ with general functions f—see (2.4). A real polynomial may have real or complex zeros. Complex roots always appear in complex conjugate pairs.

Division of Polynomials One of the auxiliary tools for solving polynomial equations and simplifying polynomial expressions is the division of polynomials. When a polynomial u is divided by a non-zero polynomial v,

$$u(x) = u_n x^n + \cdots + u_1 x + u_0, \qquad v(x) = v_m x^m + \cdots + v_1 x + v_0,$$

we obtain the quotient q and the remainder r, $u(x) = q(x)v(x) + r(x)$, where

$$q(x) = q_{n-m} x^{n-m} + \cdots + q_1 x + q_0, \qquad r(x) = r_{m-1} x^{m-1} + \cdots + r_1 x + r_0,$$

and where degree(r) < degree(v). We divide the polynomials u and v by using the algorithm for *synthetic division* [22]:

Input: coefficients $\{u_0, \ldots, u_n\}$ and $\{v_0, \ldots, v_m\}$ of polynomials u and v
for $k = n - m$ **step** -1 **to** 0 **do**
  $q_k = u_{m+k}/v_m$;
  for $j = m + k - 1$ **step** -1 **to** k **do**
    $u_j = u_j - q_k v_{j-k}$
  end
end
Output: coefficients of the remainder, $\{r_0, \ldots, r_{m-1}\} = \{u_0, \ldots, u_{m-1}\}$, and of the quotient, $\{q_0, \ldots, q_{n-m}\}$

Computing the Value of the Polynomial The division of a polynomial by the linear function $x - \zeta$ leads to an elegant and economical method to compute the value of the polynomial at $x = \zeta$ by applying the rule $p(x) = (x - \zeta)q(x) + p(\zeta)$.

We write $p(\zeta)$ in factorized form $p(\zeta) = (\cdots((a_n\zeta + a_{n-1})\zeta + a_{n-2})\zeta + \cdots)\zeta + a_0$. From this we infer that synthetic division simplifies to the *Horner* or *Ruffini algorithm*, which can be memorized in terms of the table

$$\begin{array}{ccccccc} a_n & a_{n-1} & \cdots & a_2 & a_1 & a_0 \\ & b_n\zeta & \cdots & b_3\zeta & b_2\zeta & b_1\zeta \\ b_n & b_{n-1} & \cdots & b_2 & b_1 & b_0 = p(\zeta) \end{array}$$

The first row are the coefficients a_k of p. The third row contains the sums of the corresponding terms in the first and second rows. The second row contains the products of ζ with the coefficients b_k from the third row. The value $p(\zeta)$ is in the lower right corner. For example, compute the value of $p(x) = x^3 - 2x^2 + 3x - 4$ at $\zeta = 2$. With $a_3 = 1$, $a_2 = -2$, $a_1 = 3$, $a_0 = -4$, we get $b_3 = a_3 = 1$, $b_2 = a_2 + b_3\zeta = 0$, $b_1 = a_1 + b_2\zeta = 3$ and $b_0 = a_0 + b_1\zeta = p(\zeta) = 2$.

In a computer implementation we may use the following algorithm which supplies not only the value of the polynomial, but also its derivative at ζ:

Input: coefficients $\{a_0, \ldots, a_n\}$ of the polynomial p, value of ζ
$p = a_n$; $q = 0$;
for $i = n - 1$ **step** -1 **until** 0 **do**
  $q = p + q\zeta$;
  $p = a_i + p\zeta$;
end
Output: $p(\zeta) = p$ and $p'(\zeta) = q$

Horner's procedure can also be used to compute the numbers in various bases, for example, the value of a natural number in base b:

$$(a_n a_{n-1} \ldots a_1 a_0)_b = \sum_{k=0}^{n} a_k b^k, \quad a_k \in \{0, 1, \ldots, b-1\}.$$

This method is much faster than computing the powers and requires only $\mathcal{O}(n)$ arithmetic operations. The procedure can also be parallelized by splitting the polynomial into a part with odd and a part with even powers:

$$p(x) = p_{\text{even}}\big(x^2\big) + x p_{\text{odd}}\big(x^2\big), \quad p_{\text{even}}(t) = \sum_{k=0}^{\lfloor n/2 \rfloor} a_{2k} t^k, \;\; p_{\text{odd}}(t) = \sum_{k=0}^{\lceil n/2 \rceil - 1} a_{2k+1} t^k.$$

The numerical cost of Horner's scheme is optimal. However, for a generic polynomial, there exists an algorithm which requires only $\lfloor n/2 \rfloor + 1$ multiplications and n summations, but it cannot be found easily [22]. If floating-point arithmetic with precision ε_{M} is used (Table 1.1), the computed value $p_{\text{num}}(\zeta)$ differs from the exact value $p(\zeta)$, and the difference can be bounded by

$$\big|p_{\text{num}}(\zeta) - p(\zeta)\big| < \max_{0 \le k \le n} |a_k| \big[(1 + \varepsilon_{\text{M}})^{2n} - 1\big] \frac{|\zeta|^{n+1} - 1}{|\zeta| - 1}.$$

The coefficients a_k and the value ζ are assumed to be exact; the differences appear due to rounding errors in multiplication and summation. Clearly the precision deteriorates when the degree of the polynomial or its coefficients increase.

2.4.1 Locating the Regions Containing Zeros

In general, zeros of polynomials lie in bounded regions of the complex plane. In the following, we describe some generally useful methods used to determine the extent of these regions [23]. The classical estimates are due to Cauchy and Fujiwara: the magnitudes of all zeros $\xi_j \in \mathbb{C}$ of a polynomial of the form (2.18) are smaller than the only real zero r of the Cauchy polynomial

$$g(x) = x^n - \sum_{k=0}^{n-1} \left| \frac{a_k}{a_n} \right| x^k.$$

Thus r can be understood as the most conservative estimate for the radius of the disc in the complex plane that contains all zeros ξ_j. Cauchy's estimate is

$$r \leq \max\left\{ \left| \frac{a_0}{a_n} \right|, 1 + \left| \frac{a_1}{a_n} \right|, \ldots, 1 + \left| \frac{a_{n-1}}{a_n} \right| \right\}. \tag{2.19}$$

Much later, Fujiwara [24] derived the estimate

$$r \leq 2 \max\left\{ \left| \frac{a_0}{2a_n} \right|^{1/n}, \left| \frac{a_1}{a_n} \right|^{1/(n-1)}, \ldots, \left| \frac{a_{n-1}}{a_n} \right| \right\}, \tag{2.20}$$

which may result in a narrower bound. The estimates (2.19) and (2.20) are also applicable to complex polynomials and are thus also useful for complex zeros.

Determining the Interval with Real Zeros It can be shown [25] that arbitrary real numbers $\{v_1, v_2, \ldots, v_n\}$, for which we define $\alpha = \sum_{k=1}^{n} v_k$ and $\beta = \sum_{k=1}^{n} v_k^2$, lie on a closed interval with the boundary points

$$\frac{1}{n}\left[\alpha \pm \sqrt{(n-1)\left(n\beta - \alpha^2\right)}\right].$$

This property can be used for the determination of the interval containing real zeros of a real polynomial. Let a real polynomial of degree n,

$$p(x) = a_n \prod_{k=1}^{n} (x - \xi_k),$$

which is just a factorized form of (2.18), possess real zeros $\{\xi_1, \xi_2, \ldots, \xi_n\}$. For the numbers v_k we choose $v_k(x) = (x - \xi_k)^{-1}$, where $x \neq \xi_k$ for $\forall k$. It is clear from the

structure of the polynomial that α and β can be written as

$$\alpha = \sum_{k=1}^{n} v_k = \frac{p'(x)}{p(x)}, \qquad \beta = \sum_{k=1}^{n} v_k^2 = -\left(\frac{p'(x)}{p(x)}\right)' = \frac{p'(x)^2 - p(x)p''(x)}{p(x)^2}.$$

The values $v_k(x)$ for each x thus lie on the interval with the boundary points

$$u_\pm(x) = \frac{1}{np(x)}\Big[p'(x) \pm \sqrt{\big[(n-1)p'(x)\big]^2 - n(n-1)p(x)p''(x)}\Big],$$

thus

$$u_-(x) \le \frac{1}{x - \xi_k} \le u_+(x), \quad k = 1, 2, \ldots, n. \tag{2.21}$$

To u_-, v_k, and u_+, we apply the functional $\Phi : f(x) \mapsto \lim_{x\to\infty} x(xf(x) - 1)$ which maps $u_- \mapsto \Phi(u_-)$, $v_k \mapsto \xi_k$ and $u_+ \mapsto \Phi(u_+)$. In this mapping, the sense of the relation (2.21) does not change. From the inequality $\Phi(u_-) \le \xi_k \le \Phi(u_+)$ we therefore infer that the zeros ξ_k lie on the interval $[A_-, A_+]$, where

$$A_\pm = \Phi(u_\pm) = \frac{1}{na_n}\Big[-a_{n-1} \pm \sqrt{(n-1)^2 a_{n-1}^2 - 2n(n-1)a_n a_{n-2}}\Big], \tag{2.22}$$

if $a_n > 0$, or on the interval $[A_+, A_-]$, if $a_n < 0$.

Determining the Interval Containing at Least One Zero For each point x we can determine the interval $[x - r, x + r]$ containing at least one zero of the polynomial. We differentiate the sum α from the previous paragraph m times:

$$\alpha(x) = \frac{p'(x)}{p(x)} = \sum_{k=1}^{n} \frac{1}{x - \xi_k}, \qquad \alpha^{(m)}(x) = \sum_{k=1}^{n} \frac{(-1)^m (m+1)!}{(x - \xi_k)^{m+1}}.$$

This tells us that the derivative $\alpha^{(m)}$ is bound from above as

$$\left|\alpha^{(m)}(x)\right| \le \frac{n(m+1)!}{r^{m+1}},$$

where r measures the distance from x to the nearest zero. This inequality can be read as the estimate for the upper bound for r:

$$r \le \min_{m\in\mathbb{N}_0} \left(\frac{n(m+1)!}{|\alpha(x)^{(m)}|}\right)^{1/(m+1)}. \tag{2.23}$$

A meaningful estimate for r can already be obtained at orders $m = 0$, 1, or 2.

Example The real polynomial of degree $n = 5$,

$$p(x) = 3\prod_{k=1}^{5}(x - k) = 3x^5 - 45x^4 + 255x^3 - 675x^2 + 822x - 360, \tag{2.24}$$

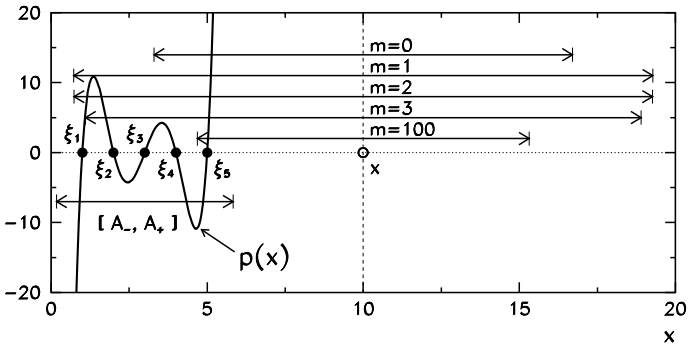


Fig. 2.6 Regions containing zeros of the polynomial (2.24). *Arrows* indicate the intervals centered at $x = 10$ which contain at least one zero according to (2.23). Also shown is the interval $[A_-, A_+]$ based on (2.22) which contains all zeros

has five real zeros $\xi_k = k$, $1 \le k \le 5$. Cauchy's estimate (2.19) is almost useless, as it tells us that all zeros are contained in the very wide interval $[-275, 275]$. Fujiwara's estimate (2.20) succeeds in narrowing down this range significantly, to $[-30, 30]$. By using (2.22) we get $[A_-, A_+] \approx [0.171573, 5.82843]$.

We also estimate the interval centered at $x = 10$ which contains at least one zero of p. We use (2.23) at $m = 0$, 1, 2, 3, and 100. At these orders, we calculate the intervals $[x - r, x + r] \approx [3.294, 16.706]$ ($m = 0$), $[0.721, 19.279]$ ($m = 1$), $[0.737, 19.264]$ ($m = 2$), $[1.098, 18.902]$ ($m = 3$), and $[4.682, 15.318]$ ($m = 100$), shown in Fig. 2.6. With $m = 100$ we clearly get a good estimate for the location of ξ_5, but computing the hundredth derivative of p'/p is unfeasible; in a practical situation we might simply keep the $m = 0$ estimate.

2.4.2 Descartes' Rule and the Sturm Method

Descartes' rule [13] allows us to determine the maximum number of positive and negative zeros of the polynomial of the form (2.18). According to this rule, the number of positive zeros N_+ (where each zero is counted as many times as its multiplicity) is equal to or smaller by an even integer than the number of sign changes M_+ in the sequence of coefficients $\{a_n, a_{n-1}, \ldots, a_1, a_0\}$,

$$N_+ = M_+ - 2i, \quad i \in \mathbb{N}_0.$$

The zero coefficients should be disregarded in the sequence. By the same token, the number of negative zeros N_- does not exceed the number of sign changes M_- in the sequence of coefficients $\{(-1)^n a_n, (-1)^{n-1} a_{n-1}, \ldots, -a_1, a_0\}$,

$$N_- = M_+ - 2j, \quad j \in \mathbb{N}_0.$$

The maximum possible number of all real zeros is therefore $N = N_+ + N_-$.

To compute the number of real zeros of the polynomial lying on the chosen interval $[a, b]$, and to determine the intervals containing the individual zeros, we use the Sturm method. Let us discuss polynomials with simple zeros only (multiple zeros can be eliminated, see Sect. 2.4.4). We first compute the derivative p'. The division of p by p' yields the remainder $-r_1$ which is used with the opposite sign in the following step. We then divide the derivative p' by r_1, yielding the remainder $-r_2$, which is again taken with the opposite sign. We repeat the procedure until the remainder is a constant:

$$\begin{aligned} p(x) &= p'(x)q_0(x) - r_1(x), \\ p'(x) &= r_1(x)q_1(x) - r_2(x), \\ r_1(x) &= r_2(x)q_2(x) - r_3(x), \\ &\cdots \quad \cdots \\ r_{k-1}(x) &= r_k(x)q_k(x) - r_{k+1}, \quad r_{k+1} = \text{const.} \end{aligned} \tag{2.25}$$

The sequence $S(x) = \{p(x), p'(x), r_1(x), r_2(x), \ldots\}$ is called the *Sturm sequence*. At given x we count the number of sign changes of its terms and denote this number by $M(x)$. The number of zeros on $[a, b]$, each of which is counted with the corresponding multiplicity, is then $N = |M(a) - M(b)|$. By dividing the real axis and computing N for such divisions, individual zeros can be isolated.

Example Define the real polynomials

$$p(x) = \prod_{k=1}^{7}(x - k), \qquad p_+(x) = p(x) + x^3. \tag{2.26}$$

The polynomial p contains the cubic term $-1960x^3$. What happens to the real zeros of p if the term $+1x^3$ is added to it? The polynomial p has seven zeros on the real axis: $\xi_k = k$, $1 \le k \le 7$ (Fig. 2.7 (left)). Let us choose the interval containing all zeros, e.g. $[a, b] = [0.5, 7.5]$. By Sturm's method we check that this interval indeed contains all seven zeros. Sturm's sequence, computed by using the procedure (2.25), has eight terms: $\{p(x), p'(x), r_1(x), \ldots, r_6(x)\}$. The values of the elements of this sequence at $x = a$ and $x = b$ are listed in the second and third columns of Table 2.1, respectively. In the second column we count $M(a) = 7$ sign changes, while there are no sign changes in the third, $M(b) = 0$. Therefore, p has $N = |M(a) - M(b)| = 7$ zeros on $[a, b]$.

A seemingly inconsequential perturbation $+1x^3$ distorts p to the extent that the perturbed polynomial p_+ has only three real zeros $\xi_1 \approx 0.999$, $\xi_2 \approx 2.085$, and $\xi_3 \approx 2.630$ (Fig. 2.7 (right)). Sturm's sequence for p_+ also contains eight terms, and their values at a and b are listed in the fourth and fifth columns of Table 2.1. From these columns, we now read off $M(a) = 5$ sign changes at $x = a$ and $M(b) = 2$ sign changes at $x = b$. On $[a, b]$ we therefore expect only $N = |M(a) - M(b)| = 3$ zeros of p_+, as confirmed by Fig. 2.7 (right).

Table 2.1 Values of the Sturm sequence polynomials at $x = a$ and $x = b$, and the number of sign changes in this sequence for the polynomials p and p_+ defined by (2.26)

f	Polynomial p		Polynomial p_+	
	$f(a)$	$f(b)$	$f(a)$	$f(b)$
p	−1055.74	1055.74	−1055.62	1477.62
p'	4128.23	4128.23	4128.98	4296.98
r_1	−1008.38	1008.38	−1008.88	670.88
r_2	2048.06	2048.06	2050.37	−187.88
r_3	−357.00	357.00	−118.83	−830.57
r_4	503.00	503.00	−5404.29	−7278.65
r_5	−54.55	54.55	−14.92	45.58
r_6	36.00	36.00	283.82	283.82
M	7	0	5	2

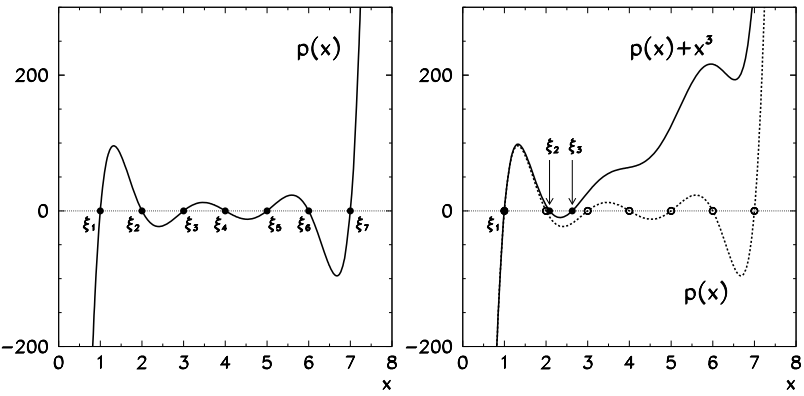


Fig. 2.7 Polynomials of degree seven defined by (2.26). [*Left*] Seven real zeros of the polynomial p. [*Right*] Three real zeros of the polynomial p_+

2.4.3 Newton's Sums and in Viète's Formulas

Solving polynomial equations is an important part of many physical problems, and their solutions often reflect the symmetry properties of these problems. If the zeros in certain expressions appear on equal footing, it may be easier to compute these expressions by using Newton's sums and Viète's formulas [26]. Both tools are also applicable as constraints in searching for zeros. For polynomials of the form (2.18) with zeros $\xi_1, \xi_2, \ldots, \xi_n$ Newton's sums are defined as

$$S_k = \sum_{l=1}^{n} \xi_l^k, \quad k = 1, 2, \ldots, n.$$

The S_k are the sums of the powers of the zeros. For $1 \le k \le n$ they are related by linear equations $a_n S_k + a_{n-1} S_{k-1} + \cdots + a_{n-k+1} S_1 + k a_{n-k} = 0$ which can be solved recursively,

$$S_k = -\frac{1}{a_n}[a_{n-1} S_{k-1} + \cdots + a_{n-k+1} S_1 + k a_{n-k}].$$

Viète's formulas relate the sums of the products of the zeros,

$$\widetilde{S}_k = \sum_{1 \le i_1 < \cdots < i_k \le n} \xi_{i_1} \cdots \xi_{i_k}, \quad (i_1, i_2, \ldots, i_k) \in \mathbb{N}_0^k,$$

which can be expressed by the coefficients of the polynomial as

$$\widetilde{S}_k = (-1)^k \frac{a_{n-k}}{a_n}.$$

2.4.4 Eliminating Multiple Zeros of the Polynomial

Many methods to compute the zeros may exhibit a dramatic drop in convergence speed and become inefficient in the case of multiple zeros. Let us discuss a m-fold zero,

$$\begin{aligned} p(x) &= (x-a)^m q_1(x), \\ p'(x) &= (x-a)^{m-1}\big[m q_1(x) + (x-a) q_1'(x)\big] \equiv (x-a)^{m-1} q_2(x), \end{aligned}$$

where q_1 and q_2 are non-constant polynomials that do not have a common divisor with the polynomials $p(x)$ and $(x-a)$. We see that the common divisor of p and p' contains the factor $(x-a)^{m-1}$, and an analogous situation can be observed for all multiple zeros. To eliminate multiple zeros, we compute the greatest common divisor of p and its derivative p', and divide it out from p. The polynomial obtained in this manner contains only simple zeros.

In principle, the *greatest common divisor* $\gcd(p, p')$ of the polynomials p and p' can be computed by using the Euclid algorithm (2.25). The procedure is repeated until the remainder r_{k+1} is equal to zero, and then

$$\gcd\big(p, p'\big)(x) = r_k(x);$$

on the other hand, if r_{k+1} is non-zero, the greatest common divisor of the polynomials is just a real number. This procedure may be numerically unstable [5] and we should avoid it if possible; if not, it should be accepted as necessary evil.

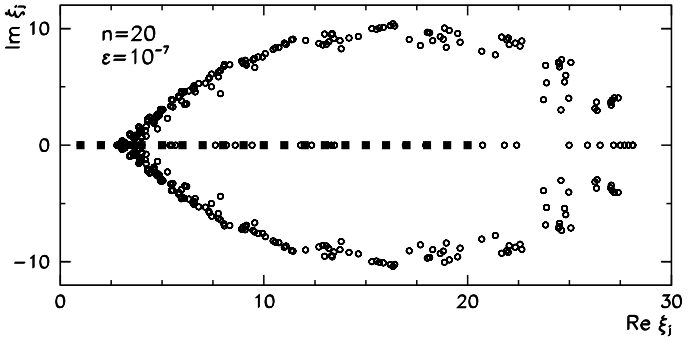


Fig. 2.8 Ill conditioning of the computation of zeros of the Wilkinson polynomial p_{20} of degree 20. *Full squares*: zeros of the exact polynomial lying on the real axis, $\xi_j = j$ for $1 \le j \le 20$. *Empty circles*: zeros of twenty different polynomials in which the coefficients are perturbed as $a_k \to a_k(1 + \varepsilon R_k)$, where $\varepsilon = 10^{-7}$ and R_k is a normally distributed random number (zero average and unit variance). The most sensitive zero is ξ_{15} and has the strongest dependence on the coefficient $a_{15} \approx 1.67 \times 10^9$. The problem is extremely ill-conditioned, as $\kappa \approx 5.1 \times 10^{13}$. In spite of the tiny change in the coefficients, real zeros shift deeply into the complex plane

2.4.5 Conditioning of the Computation of Zeros

Computing the zeros of polynomials is an ill-conditioned problem. If the coefficient a_k of the polynomial (2.18) is changed by an infinitesimal amount δa_k, the zero ξ_j shifts by

$$\delta\xi_j = -(\delta a_k)\xi_j^k / p'(\xi_j).$$

The sensitivity of the position of the zero with respect to the perturbations in the coefficients can be expressed by the ratio of the relative change of the zero ξ_j to the relative change of the coefficient a_k,

$$\kappa = \frac{|\delta\xi_j|}{|\xi_j|}\left[\frac{|\delta a_k|}{|a_k|}\right]^{-1} = \left|\frac{a_k \xi_j^{k-1}}{p'(\xi_j)}\right|,$$

which plays the role of a condition number [27]. An ill-conditioned problem exhibits large values of κ. The classical example of the Wilkinson polynomial $p_{20}(x) = \prod_{k=1}^{20}(x-k) = a_0 + a_1 x + \cdots + a_{19}x^{19} + x^{20}$ is shown in Fig. 2.8.

2.4.6 General Hints for the Computation of Zeros

Multiple polynomial zeros frequently stem from the symmetry properties of the physical problem. If this is the case, it is recommended to remove them "by hand"; otherwise, they can be removed by the procedure from Sect. 2.4.4. By using Sturm's method we first locate the intervals containing the individual zeros, which are then

computed one by one by using the methods described in the following. When some zero ξ is found, it can be used to decrease the degree of the polynomial $p(x)/(x-\xi) \to p(x)$ and thereby reduce the complexity of further calculations, a process known as *deflation*. However, deflation may be numerically unstable and we might wish to avoid it and harness the entire polynomial instead. To stabilize the computation, the zeros already computed can be used in the Maehly–Newton method (Sect. 2.4.11).

If multiple zeros are anticipated, an alternative way is to divide the polynomial p by its derivative p', yielding a rational function $R(x) = p(x)/p'(x)$. The zeros of R can then be computed by using one of the general methods, for example, by the Newton–Raphson method since the derivative of R is explicitly known, $R'(x) = 1 - p(x)p''(x)/p'(x)^2$.

The particular choice of methods presented in the following subsections is based on their general usefulness and pedagogical value. The most efficient "sure-fire" and "black-box" method currently on the market, which is implemented in major software packages like MATHEMATICA or MATLAB, is the *three-stage Jenkins–Traub method* [28, 29]. It is well-suited for the computation of zeros of real and complex polynomials, and has more than quadratic order, but it is too complicated to be discussed at the level of this textbook.

2.4.7 Bernoulli's Method

Bernoulli's method follows the same line of thought as the power methods used to compute the eigenvalues of matrices. It can be used to compute the largest zero (by magnitude) of the polynomial (2.18), which is in fact also the characteristic polynomial of the difference equation

$$a_n y_{n+k} + a_{n-1} y_{n+k-1} + \cdots + a_1 y_{k+1} + a_0 y_k = 0, \quad k = 0, 1, \ldots. \tag{2.27}$$

Assume that all zeros of the polynomial p are simple and that they are ordered as $|\xi_1| > |\xi_2| > \cdots > |\xi_n|$. If ξ is the root of $p(x) = 0$, it can be shown [30] that $y_k = \xi^k$ or any linear combination of the powers of the roots $y_k = \sum_j b_j \xi_j^k$ solve (2.27). The coefficients b_j are found by using the initial values $y_0 = c_0$, $y_1 = c_1, \ldots, y_{n-1} = c_{n-1}$, where c_k are arbitrary constants, which constitutes a system of n linear equations with a Vandermonde coefficient matrix. If we are interested only in the largest root (by magnitude) ξ_1, we iterate

$$y_{n+k} = -\frac{1}{a_n}[a_{n-1} y_{n+k-1} + \cdots + a_1 y_{k+1} + a_0 y_k], \quad k = 0, 1, \ldots$$

with arbitrary initial values $\{y_0, y_1, \ldots, y_n\}$. The sequence $\{y_k\}$ then almost always converges to this root geometrically with the ratio $|\xi_2/\xi_1|$,

$$\frac{y_{k+1}}{y_k} = \xi_1 + \mathcal{O}\big(|\xi_2/\xi_1|^k\big), \quad k \gg 1.$$

The procedure becomes a bit more involved for multiple zeros. Assume that $\{\xi_1, \xi_2, \ldots, \xi_h\}$ have multiplicities $\{m_1, m_2, \ldots, m_h\}$ and $|\xi_1| \geq |\xi_2| \geq \cdots \geq |\xi_h|$, where $\sum_{k=1}^{h} m_k = n$ and $m_k \geq 1$. Then the general solution of the difference equation is

$$y_k = \sum_{l=1}^{h} \sum_{i=0}^{m_l - 1} b_l^{(i)} k^i \xi_l^k.$$

As an example, let us discuss a double zero $\xi_2 = \xi_1^*$ (all other zeros being simple), such that $|\xi_1| = |\xi_2| > |\xi_3| > \cdots > |\xi_n|$. Then in y_k at $k \gg 1$ two terms dominate, $y_k \approx b_1 \xi_1^k + b_2 \xi_2^k$, where $b_2 = b_1^*$. We form the expressions

$$A_k = y_{k+1} y_{k-1} - y_k^2 \approx |b_1|^2 (\xi_1 - \xi_2)^2 (\xi_1 \xi_2)^{k-1},$$
$$B_k = y_{k+2} y_{k-1} - y_{k+1} y_k \approx |b_1|^2 (\xi_1 - \xi_2)^2 (\xi_1 + \xi_2)(\xi_1 \xi_2)^{k-1},$$

and observe the limits

$$\lim_{k \to \infty} \frac{B_k}{A_k} = \xi_1 + \xi_2 \equiv s, \qquad \lim_{k \to \infty} \frac{A_{k+1}}{A_k} = \xi_1 \xi_2 \equiv t.$$

Finally we obtain the complex conjugate roots ξ_1 and ξ_2 by solving the quadratic equation $\xi^2 - s\xi + t = 0$. For error estimates and other details see [30].

2.4.8 Horner's Linear Method

Assume that the interval $[a, b]$ contains an odd-order zero ξ of the polynomial p, so that $p(a)p(b) < 0$. We transform the polynomial p into a new polynomial p_0 such that the root lies on $[0, 1]$:

$$p_0(x) = p(Lx + a), \quad L = b - a.$$

In the following, we use the iteration

$$p_{k+1}(x) = p_k(x + d_k), \quad k = 0, 1, 2, \ldots \tag{2.28}$$

to shift the polynomial until its root coincides with the origin (see Fig. 2.9). At each step of the iteration, p_k has the form

$$p_k(x) = \sum_{i=0}^{n} a_i^{(k)} x^i.$$

We determine the shift d_k by computing the root of $d_k a_1^{(k)} + a_0^{(k)} = 0$, so

$$d_k = -a_0^{(k)} / a_1^{(k)}.$$

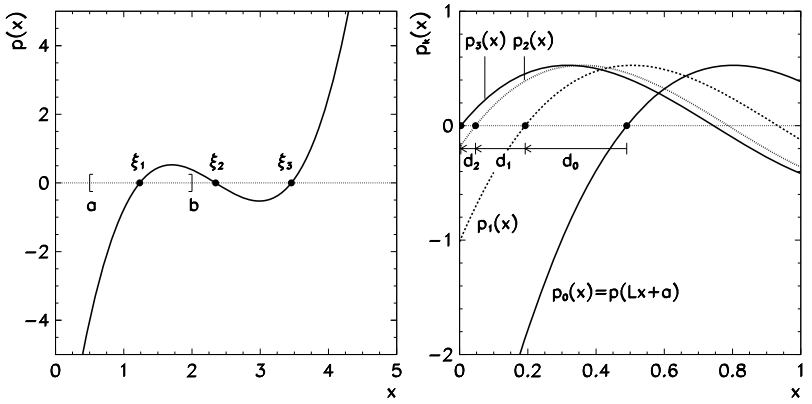


Fig. 2.9 Linear Horner's method. [*Left*] The polynomial $p(x) = (x - \xi_1)(x - \xi_2)(x - \xi_3)$ with the zeros $\xi_1 = 1.234$, $\xi_2 = 2.345$, and $\xi_3 = 3.456$. We attempt to compute ξ_1 which is assumed initially to lie on the interval $[a, b] = [0.5, 2.0]$. [*Right*] The sequence of the rescaled polynomials p_k, whose zeros in subsequent steps of the iterations (2.28) with shifts $d_0 \approx 0.297$, $d_1 \approx 0.146$, $d_2 \approx 0.043$, ... approach the origin. Judging from the first three approximations, the zero ξ_1 of p lies at $a + (b - a)(d_0 + d_1 + d_2) \approx 1.229$

We repeat the procedure until d_k becomes smaller than some required precision. The root of the original equation is then computed as the sum of all shifts,

$$\xi = a + L \sum_k d_k.$$

This method acquired its name from the extensive use of Horner's algorithm used to compute the shifts of the axis origin. By Horner's scheme, one shift is computed in $\mathcal{O}(n^2)$ operations, while the rescaling of the axis requires $\mathcal{O}(n)$.

2.4.9 Bairstow's (Horner's Quadratic) Method

A real polynomial p may have complex zeros which always occur in complex conjugate pairs. To compute such zeros we must use complex arithmetic, but most iterative methods with initial approximations on the real axis do not allow to be steered into the complex plane. Complex computation can be avoided by using the Bairstow method. The first observation leading to this method is that the zeros $\xi_{1,2}$ ($\xi_1 = \xi_2^*$) of the polynomial

$$d(x) = x^2 - rx - s, \tag{2.29}$$

are also zeros of the original polynomial

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,$$

if it is divisible by d with zero remainder. By dividing p by d, we generally get

$$p(x) = q(x)d(x) + ux + v,$$

where the degree of q is $n-2$. The coefficients u and v are functions of the parameters r and s of the polynomial d. We are therefore seeking a pair (r, s) such that the remainder vanishes:

$$u(r, s) = 0, \qquad v(r, s) = 0.$$

This constitutes a system of non-linear equations that can be solved by the Newton method. One step of the iteration to solve the system is

$$\begin{pmatrix} r_{k+1} \\ s_{k+1} \end{pmatrix} = \begin{pmatrix} r_k \\ s_k \end{pmatrix} - \begin{pmatrix} \partial_r u & \partial_s u \\ \partial_r v & \partial_s v \end{pmatrix}^{-1} \begin{pmatrix} u(r_k, s_k) \\ v(r_k, s_k) \end{pmatrix}, \tag{2.30}$$

where k is the iteration index. The Jacobi matrix contains the derivatives

$$\partial_r u = \frac{\partial u(r, s)}{\partial r}, \qquad \partial_s u = \frac{\partial u(r, s)}{\partial s}, \qquad \partial_r v = \frac{\partial v(r, s)}{\partial r}, \qquad \partial_s v = \frac{\partial v(r, s)}{\partial s},$$

which have to be computed in each step of the iteration at $(r, s) = (r_k, s_k)$. We obtain them by the following consideration: p does not depend on r or s, thus

$$\frac{\partial p(x)}{\partial r} \equiv 0 = d(x)\frac{\partial q(x)}{\partial r} - xq(x) + x\frac{\partial u}{\partial r} + \frac{\partial v}{\partial r},$$
$$\frac{\partial p(x)}{\partial s} \equiv 0 = d(x)\frac{\partial q(x)}{\partial s} - q(x) + x\frac{\partial u}{\partial s} + \frac{\partial v}{\partial s}.$$

At $x = \xi_1$ and $x = \xi_2$ (which are the zeros of d) these equations simplify to

$$-\xi_i q(\xi_i) + \xi_i \partial_r u + \partial_r v = 0, \qquad -q(\xi_i) + \xi_i \partial_s u + \partial_s v = 0, \quad i = 1, 2.$$

This is a system of four equations with four variables $\partial_r u$, $\partial_r v$, $\partial_s u$, and $\partial_s v$, which can be solved easily. The solution is even simpler if we once more divide q by d,

$$q(x) = q_1(x)d(x) + u_1 x + v_1,$$

since by recalling $d(\xi_1) = d(\xi_2) = 0$ we obtain

$$q(\xi_i) = u_1 \xi_i + v_1, \quad i = 1, 2.$$

Both divisions can be efficiently accomplished by using the Horner scheme at a cost of $\mathcal{O}(2n)$. In the non-degenerate case $\xi_1 \neq \xi_2 = \xi_1^*$ we get [5]

$$\partial_r u = r u_1 + v_1, \qquad \partial_r v = s u_1, \qquad \partial_s u = u_1, \qquad \partial_s v = v_1.$$

We run the iteration (2.30) until the sequence of the coefficients $\{r_k\}$ and $\{s_k\}$ converges to the final values r_∞ and s_∞ to within required precision. Ultimately, the zeros $\xi_{1,2}$ of the polynomial p are calculated by solving the quadratic equation

$$\xi^2 - r_\infty \xi - s_\infty = 0. \tag{2.31}$$

Table 2.2 Convergence of the parameters r and s of the polynomial d (2.29) and the Jacobi matrix in the iteration (2.30). With the initial approximation $(r, s) = (2, 0)$ the iteration converges to $(r, s) = (2, -5)$ which, according to (2.31), corresponds to two zeros of the polynomial (2.32), $\xi_{1,2} = 1 \pm 2\,\mathrm{i}$

k	r	s	$\partial_r u$	$\partial_s u$	$\partial_r v$	$\partial_s v$
0	2.5128	−3.3333	−65.000	35.000	0.000	−135.000
1	2.2144	−5.1768	−10.457	21.943	−73.143	−65.595
2	1.9788	−4.9591	9.170	21.782	−112.763	−39.065
3	1.9999	−4.9993	4.160	25.381	−125.864	−46.062
4	2.0000	−5.0000	4.989	25.004	−125.000	−45.015

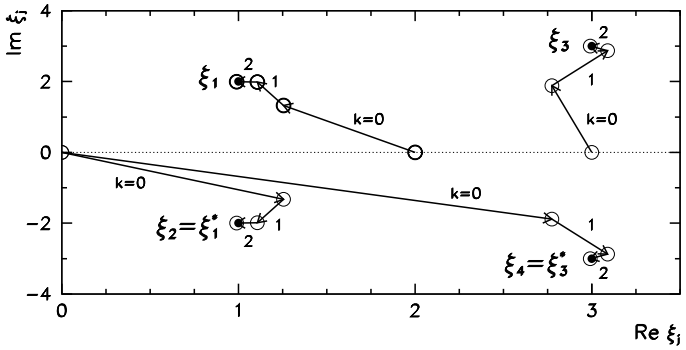


Fig. 2.10 Three Newton steps of the approach to $\xi_{1,2} = 1 \pm 2\,\mathrm{i}$ and $\xi_{3,4} = 3 \pm 3\,\mathrm{i}$ with the initial approximations $(r, s) = (2, 0)$ and $(r, s) = (3, 0)$, respectively

Example The polynomial

$$p(x) = x^5 - 13x^4 + 75x^3 - 241x^2 + 420x - 450 \tag{2.32}$$

has two pairs of complex conjugate zeros, $\xi_{1,2} = 1 \pm 2\,\mathrm{i}$ and $\xi_{3,4} = 3 \pm 3\,\mathrm{i}$, as well as a real zero $\xi_5 = 5$. By Bairstow's method we first compute the pair $\xi_{1,2}$. We use the initial approximation $(r, s) = (2, 0)$ for the parameters of d (2.29) to start the iteration (2.30). In each step we perform the division $p(x) = q(x)d(x) + ux + v$ to compute u and v, and $q(x) = q_1(x)d(x) + u_1 x + v_1$ to obtain u_1 and v_1, which we use to compute the elements of the Jacobi matrix $\partial_r u$, $\partial_s u$, $\partial_r v$, and $\partial_s v$ (Table 2.2). We use the final values of r and s in (2.31) to compute $\xi_{1,2} = 1 \pm 2\,\mathrm{i}$. The left part of Fig 2.10 illustrates the convergence to $\xi_{1,2}$. For the next pair of zeros, $\xi_{3,4}$, we act similarly. We divide the polynomial by the computed polynomial d (the remainder is zero), and repeat the Bairstow procedure. The convergence to the zeros $\xi_{3,4}$ is shown in the right part of Fig. 2.10.

2.4.10 Laguerre's Method

To a much larger extent than the methods mentioned so far, Laguerre's method utilizes the fact that the polynomial equations involved have degrees more than 1. It is suited for the computation of isolated simple zeros (multiple zeros may be eliminated by using the procedure from Sect. 2.4.4). Let the polynomial p be given in its factorized form

$$p(x) = a_n \prod_{k=1}^{n} (x - \xi_k)$$

(the leading coefficient a_n is irrelevant in this method). We define

$$G(x) = -\frac{p'(x)}{p(x)} = -\sum_{k=1}^{n} \frac{1}{x - \xi_k}, \qquad H(x) = G'(x) = \sum_{k=1}^{n} \frac{1}{(x - \xi_k)^2}. \tag{2.33}$$

When x is in the vicinity of some simple zero ξ_k, the kth term of the sums in (2.33) becomes very large, while the other terms change only unsubstantially. The sums of the remaining terms may therefore be approximated as

$$\sum_{j \neq k} \frac{1}{x - \xi_j} \approx \frac{n-1}{b}, \qquad \sum_{j \neq k} \frac{1}{(x - \xi_j)^2} \approx \frac{n-1}{b^2},$$

where b is a constant, and (2.33) rewritten as

$$G(x) \approx -\frac{1}{c(x)} - \frac{n-1}{b}, \qquad H(x) \approx \frac{1}{c(x)^2} + \frac{n-1}{b^2}, \quad c(x) = x - \xi_k.$$

We eliminate b and obtain a quadratic equation for c^{-1}, with the solutions

$$\frac{1}{c_{1,2}} = \frac{-G \pm \sqrt{(n-1)(nH - G^2)}}{n}.$$

By using the expressions for H and G we finally obtain

$$c_{1,2}(x) = \frac{np(x)}{p'(x) \pm \sqrt{(n^2 - n - 1)p'(x)^2 - n(n-1)p(x)p''(x)}}.$$

We approach the zero ξ_k by iterating

$$x_k^{(i+1)} = x_k^{(i)} - c\big(x_k^{(i)}\big), \qquad \lim_{i \to \infty} x_k^{(i)} = \xi_k,$$

where k is the index of the zero and i is the iteration index. To compute the correction $c(x)$ we use the *smaller* (in magnitude) of the solutions $c_1(x)$ or $c_2(x)$.

Laguerre's method is among the fastest available, as it converges cubically in the case of simple zeros. It can also be used to search for complex zeros. Its only deficiency is the diminished speed of convergence in the case of multiple zeros where it becomes only linear.

2.4.11 Maehly–Newton–Raphson's Method

This is a variant of the Newton–Raphson method where, in order to compute a zero of the polynomial, all previously determined zeros are considered, thus improving global stability. Let us discuss a polynomial p with simple zeros and assume that the zeros $\xi_1, \xi_2, \ldots, \xi_j$ have already been found to some precision. Define the function

$$p_j(x) = p(x) \Big/ \prod_{k=1}^{j} (x - \xi_k),$$

which shares the (as yet) undetermined zeros with the polynomial p and which weakens or completely eliminates the influence of previously determined zeros of p. In seeking the next zero, we use the Newton–Raphson method in the form

$$x_{i+1} = \Phi_j(x_i), \quad \Phi_j(x) = x - \frac{p(x)}{p'(x) - p_j(x)}.$$

This approach is useful with all functions for which the order of previously found zeros is known. Moreover, even though previous zeros have been determined only to finite precision, this affects neither the precision of the determination of the remaining zeros nor the quadratic convergence towards them. The mapping is not well defined for all previously found zeros or their approximate values, and we try to avoid them in subsequent computations. We do this by systematically searching for zeros from the largest to the smallest, using the estimates from Sect. 2.4.1 to determine the appropriate upper bounds.

The method just described is most suitable for computing the zeros of a polynomial with purely real zeros. In this case it can be merged with the *Newton double-step method* based on the theorem from Appendix D (see (D.6) and the corresponding text). The theorem has a practical consequence. Assume that we are seeking the largest zero ξ_1 and that the starting point of the iteration is $x_0 > \xi_1$. By using the Newton double-step method

$$x_{i+1} = x_i - 2\frac{p(x_i)}{p'(x_i)}, \quad i = 0, 1, \ldots$$

we obtain the sequence $x_0 \geq x_1 \geq \cdots \geq x_{i+1} \geq \xi_1$ with the limit $\lim_{i\to\infty} x_i = \xi_1$, and $p(x_0)p(x_i) \geq 0\ \forall i$; or else, in some step κ, we cross the zero such that

$$p(x_0)p(x_i) > 0 \quad (i = 0, 1, \ldots, \kappa - 1) \quad \text{and} \quad p(x_0)p(x_\kappa) < 0,$$

where $x_0 > x_1 > \cdots > x_{\kappa-1} > \xi_1 > x_\kappa > \xi_2$ applies. In this case we continue the iteration with the usual Newton method

$$x_{i+1} = x_i - \frac{p(x_i)}{p'(x_i)}, \qquad x_0 = x_\kappa, \quad i = 0, 1, \ldots,$$

and obtain the sequence $x_{i+1} \geq x_i \geq \cdots \geq \xi_1$ with the limit $\lim_{i\to\infty} x_i = \xi_1$. The following algorithm based on the Maehly–Newton–Raphson single- and double-step methods has been taken from [5]. It allows us to calculate the zeros of a degree-n polynomial possessing exclusively real zeros to a relative precision ε:

Input: polynomial p of degree n, derivative p', upper bound for the zeros x_0, precision ε, declared (uninitialized) variables ξ_j $(1 \leq j \leq n)$
for $j = 1$ **step** 1 **to** n **do**
  $m = 2$; $z = x_0$;
  ITER: $x = z$; $s = 0$;
  for $i = 1$ **step** 1 **to** $j - 1$ **do** $s = s + 1/(x - \xi_i)$;
  $z = x - mp(x)/[p'(x) - p(x)s]$;
  switch m **do**
    case 1*:*
      $e_{\max} = \varepsilon \max\{|x|, |z|\}$;
      $e = |z - x|$;
      if $((z < x) \vee (e \geq e_{\max}))$ **then goto** ITER;
    endsw
    case 2*:*
      if $(z \geq x)$ **then**
        $x_0 = z = x$; $m = 1$;
      end
      goto ITER;
    endsw
  endsw
  $\xi_j = x$;
end
Output: approximations of zeros $\xi_1, \xi_2, \ldots, \xi_n$

2.5 Algebraic Equations of Several Variables ⋆

Polynomial or algebraic equations [31] and their systems, like, for example,

$$x_1^2 - x_2 = 0, \qquad x_1^2 + x_2^2 = 1 \tag{2.34}$$

(the solution is the intersection of a parabola and a circle), or general systems

$$f_i(x_1, x_2, \ldots, x_n) = 0, \quad i = 1, 2, \ldots, m, \tag{2.35}$$

can be solved by algorithms from previous sections, assuming that such systems possess a finite number of solutions. Yet in some instances, their algebraic properties can be exploited so that they can be solved more elegantly. One such approach takes us to *Gröbner bases* that are used in a variety of applications, e.g. in automated assembly of mechanical systems [32], in seeking intersections of volumes in three-dimensional space, and in robot control. They are also useful in coding and

cryptography [33], as well as in logical and combinatorial problems [34]. Moreover, the theory of Gröbner bases provides tools to "solve" also systems with infinite sets of solutions [35].

Gröbner bases (invented by Buchberger) are a generalization of orthogonal bases known from linear algebra to algebraic expressions. This section conveys only their background and the basic idea. Mathematically precise definitions and pedagogically thorough introductions can be found in [36–39].

Monomials and Polynomial Terms The vantage point are polynomials in n variables $\{x_i\}_{i=1}^n$ with the coefficients from a ring K, e.g. $K = \mathbb{R}$ or $\mathbb{C}$, and the resulting algebraic structures [40]. Let $A = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}_0^n$ denote the vector in which the value of each component may be zero or a natural number. Then we can write a *monomial*

$$x^A = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n},$$

where $\alpha_j \in \mathbb{N}_0$, $1 \le j \le n$, and $\deg(x^A) = \sum_i \alpha_i$ is the degree of the monomial. The product of two monomials is $x^A x^B = x^{A+B}$, where $A + B$ is to be understood as a summation of vectors, e.g.

$$x^A = x_1^2 x_2^3 x_3, \qquad x^B = x_1 x_2^2 x_3^3, \qquad x^{A+B} = x_1^3 x_2^5 x_3^4,$$

since $A + B = (2, 3, 1) + (1, 2, 3) = (3, 5, 4)$. A monomial multiplied by a scalar $a \in K$, ax^A, is a *polynomial term* or simply *term*. By using this notation, a polynomial in n variables is compactly written as

$$f(x_1, \ldots, x_n) = \sum_A a_A x^A, \quad a_A \in K.$$

These are the basic building blocks of systems like (2.34) or (2.35). All polynomials of n variables with coefficients from K constitute a *polynomial ring*

$$K[x_1, x_2, \ldots, x_n] = \left\{ \sum_{A \in \mathbb{N}_0^n} a_A x^A : \forall a_A \in K \right\}.$$

Ordering of Monomials There is a way to distinguish among the multitude of variables and terms in each polynomial. To do this, one utilizes the symbols $\prec$ and $\succ$ to define the *total order* of monomials in $K[x_1, x_2, \ldots, x_n]$ as a relation possessing the properties: (i) a constant is the smallest monomial: $x^A \succ 1$ for all $x^A \neq 1$, and (ii) the ordering is multiplicative: $x^A \succ x^B \Rightarrow x^{A+C} \succ x^{B+C}$ for all x^C. The conditions of total order are satisfied by various orderings of monomials. For solving algebraic equations, the most relevant orderings are:

- *Lexicographic order*: $x^A \succ x^B$ if the first non-zero component of the vector $A - B$ is positive. Examples: $x_1^3 x_2^2 x_3^2 \succ x_1^2 x_2^4 x_3^4$ and $x_i \succ x_{i+1}$.
- *Graded lexicographic order*: $x^A \succ x^B$ if $\deg(x^A) \ge \deg(x^B)$ or $\deg(x^A) = \deg(x^B)$ and the *first* non-zero component of $A - B$ is *positive*. Example: $x_1^3 x_2^4 x_3^2 \succ x_1^3 x_2^2 x_3$. Another example: $x_1^2 \succ x_1 x_2 \succ x_2^2 \succ x_1 x_3 \succ x_2 x_3 \succ x_3^2$.

- *Graded reverse lexicographic order*: $x^A \succ x^B$ if $\deg(x^A) \geq \deg(x^B)$ or $\deg(x^A) = \deg(x^B)$ and the *last* non-zero component of $A - B$ is *negative*. Example: $x_1^3 x_2^4 x_3 \succ x_1 x_2^5 x_3$.

With the given ordering, one can assign to any polynomial f from the ring $K[x_1, x_2, \ldots, x_n]$ the corresponding largest monomial, which we call the *initial* or *leading monomial* and denote it by $\mathrm{LT}(f) = x^A$, where A is the maximal vector of powers. The corresponding leading coefficient a_A is denoted by $\mathrm{LC}(f) = a_A$.

Varieties and Ideals Let $\mathcal{F} = \{f_i\}_{i=1}^m$ be a set of polynomials from the ring $K[x_1, x_2, \ldots, x_n]$. At this point we establish the connection to the primary task of this section, which is solving systems of equations of the type (2.35). The set of all solutions of (2.35) is called the *algebraic variety* of $\mathcal{F}$ and is denoted by

$$\mathcal{V}(\mathcal{F}) = \big\{x = (x_1, x_2, \ldots, x_n) \in K^n : \ f(x) = 0, \ \forall f \in \mathcal{F}\big\}.$$

All possible (algebraic) linear combinations of polynomials in $\mathcal{F}$ span the subring of $K[x_1, x_2, \ldots, x_n]$,

$$\langle \mathcal{F} \rangle = \big\{p_1 f_1 + \cdots + p_m f_m : \ f_i \in \mathcal{F}, \ \forall p_i \in K[x_1, x_2, \ldots, x_n]\big\},$$

which is known as the *polynomial ideal* generated by $\mathcal{F}$. The same ideal can be generated by different sets of polynomials, but all polynomials from the ideal share the property that they vanish on the algebraic variety of any set of polynomials that generates this ideal. So in fact, $\mathcal{V}(\mathcal{F}) = \mathcal{V}(\langle \mathcal{F} \rangle)$, where $\mathcal{V}(\langle \mathcal{F} \rangle)$ is the variety of the ideal $\mathcal{F}$, i.e. the set of all common zeros of the polynomials of $\langle \mathcal{F} \rangle$,

$$\mathcal{V}(\mathcal{F}) = \mathcal{V}\big(\langle \mathcal{F} \rangle\big) = \big\{x \in K^n : \ f(x) = 0, \ \forall f \in \langle \mathcal{F} \rangle\big\}.$$

The basic idea behind Gröbner bases is to find a set of generators of a polynomial ideal that is "simple". In many cases a Gröbner basis allows one to determine the variety of an ideal easily.

Reduced Gröbner Basis Let $\mathcal{I}$ be a non-zero ideal, and let $\mathcal{G} = \{g_1, g_2, \ldots, g_M\}$ be a finite set of non-zero elements of $\mathcal{I}$. The set $\mathcal{G}$ is a Gröbner basis for $\mathcal{I}$ if the ideal generated by the leading terms of elements in $\mathcal{G}$ is equal to the ideal generated by the leading terms of elements in $\mathcal{I}$ (see [35], Theorem 1.2.16):

$$\big\langle \mathrm{LT}(\mathcal{G}) \big\rangle = \big\langle \mathrm{LT}(\mathcal{I}) \big\rangle.$$

A Gröbner basis is called *reduced* if $\mathrm{LC}(g_i) = 1$ for all i and no term of g_i is divisible by any $\mathrm{LT}(g_j)$ for $j \neq i$ (see [35], Definition 1.2.26). In loose parlance, the reduced Gröbner basis $\mathcal{G}$ belonging to the ideal $\mathcal{I}$, with adopted monomial ordering, is a unique set of polynomials that generates $\mathcal{I}$. If $\mathcal{I}$ is generated by polynomials from $\mathcal{F}$, then the sets $\mathcal{F}$ and $\mathcal{G}$ have equal algebraic varieties and equal ideals:

$$\mathcal{V}(\mathcal{F}) = \mathcal{V}(\mathcal{G}) \quad \text{and} \quad \langle \mathcal{F} \rangle = \langle \mathcal{G} \rangle.$$

Note that the Gröbner basis can be larger that the generating set, for instance, the ideal $\mathcal{I} = \langle x_1^2,\, x_1x_2 - x_2^2\rangle$ in lexicographic order, $x_1 \succ x_2$, has the Gröbner basis $\mathcal{G} = \{x_1^2,\, x_1x_2 + x_2^2,\, x_2^3\}$.

Solving Equations To solve the equations, we exploit the *elimination property* of Gröbner bases, which means the following. Let the variables be ordered as $x_1 \succ x_2 \succ \cdots \succ x_n$, presuming lexicographic ordering, and let the Gröbner basis $\mathcal{G}$ generate the ideal $\mathcal{I}$ in $K[x_1, x_2, \ldots, x_n]$. Then the generators of the intersection of $\mathcal{I}$ with the subring $K[x_k, x_{k+1}, \ldots, x_n]$ are given by the intersection of the Gröbner basis $\mathcal{G}$ with the subring $K[x_k, x_{k+1}, \ldots, x_n]$. A polynomial f lies in the subring $K[x_k, x_{k+1}, \ldots, x_n]$ precisely in the case that its leading term $\mathrm{LT}(f)$ is in this subring.

Let $\mathcal{I} = \langle\mathcal{F}\rangle$ be the ideal generated by the polynomials $\mathcal{F} = \{f_i\}_{i=1}^m$ of the original system of equations (2.35), and let $\mathcal{G}$ be its Gröbner basis with lexicographic ordering. Then (2.35) can be rewritten as

$$g_i(x_1, x_2, \ldots, x_n) = 0, \quad i = 1, 2, \ldots, M. \tag{2.36}$$

In many cases the systems with polynomials g_i turn out to be much simpler than those involving f_i, and this simplification is one of the main allures of Gröbner bases. In the case of a zero-dimensional ideal, i.e. when the variety consists of a finite number of points, the Gröbner basis (in a proper order of the elements) has the form

$$g_1(x_n), \quad g_2(x_n, x_{n-1}), \quad g_k(x_n, x_{n-1}, x_{n-2}), \quad \ldots.$$

The system of equations $g_i = 0$ for $i = 1, 2, \ldots, M$ can then be solved recursively: first one solves the equation $g_1 = 0$, then its solution is used in solving the second equation, $g_2 = 0$, and so on.

Given the set $\mathcal{F}$, the reduced Gröbner basis $\mathcal{G}$ of the ideal $\langle\mathcal{F}\rangle$ can be computed in several ways. The most widely known are the Buchberger algorithm [37, 42] and its improved implementation, the Faugère algorithm F_5 [43, 44]. These procedures are available in software packages like Singular (commands `groebner` or `slimgb`), Magna, MATHEMATICA (command `GroebnerBasis`) and Maple (command `gbasis`), as well as in the `sympy` library of Python. In these environments, Gröbner bases are implemented for symbolic solution of algebraic systems. Yet when seeking specific solutions of complex systems of equations, it may be preferable to avoid these built-in capabilities and utilize Gröbner bases directly. This allows us to hand-pick and control the solutions when the equations $g_1 = 0$, $g_2 = 0$, and so on, are solved in sequence.

Example Let us see how Gröbner bases are used to solve (2.34) over the real field $\mathbb{R}[x_1, x_2]$. In this case the Gröbner basis consists of two functions,

$$g_1 = -1 + x_2 + x_2^2, \qquad g_2 = x_1^2 - x_2.$$

In MATHEMATICA, for example, they can be obtained from the functions

$$f_1(x_1, x_2) = x_1^2 - x_2, \qquad f_2(x_1, x_2) = x_1^2 + x_2^2 - 1,$$

read off from (2.34) by using the command

```
GroebnerBasis[{x1^2 - x2, x1^2 + x2^2 - 1}, {x1, x2}].
```

The default ordering in this case is lexicographic, $x_1 \succ x_2$, and the command returns the Gröbner basis

```
{g1, g2} = {-1 + x2 + x2^2, x1^2 - x2}.
```

The function g_1 depends solely on x_2, so $g_1(x_2) = 0$ can be solved for x_2 explicitly, yielding

$$x_2 \in \left\{ \frac{-1 - \sqrt{5}}{2}, \frac{-1 + \sqrt{5}}{2} \right\}.$$

For each of these two values of x_2, we then solve the equation $g_2(x_1, x_2) = 0$, which involves only the x_1 variable, and realize that only $x_2 = (-1 + \sqrt{5})/2$ leads to a real value of x_1. This brings us to the solutions of the second equation,

$$x_1 \in \{\sqrt{x_2}, -\sqrt{x_2}\},$$

hence the two real solutions of (2.34) are approximately $(0.7862, 0.6180)$ and $(-0.7862, 0.6180)$.

In some instances it may happen that the number of equations M in (2.36) is exponentially larger than the number of equations m in (2.35). A typical reduced basis, however, has a dimension M that exceeds m at most by a few orders, and this represents only a minor problem in computations involving Gröbner bases. A much more severe issue is revealed by the following example.

Example Let us discuss the system of equations $f_1 = f_2 = f_3 = f_4 = 0$ with

$$\begin{aligned}
f_1(x_1, x_2, x_3) &= 8x_1^2x_2^2 + 5x_1x_2^3 + 3x_1^3x_3 + x_1^2x_2x_3, \\
f_2(x_1, x_2, x_3) &= x_1^5 + 2x_2^3x_3^2 + 13x_2^2x_3^3 + 5x_2x_3^4, \\
f_3(x_1, x_2, x_3) &= 8x_1^3 + 12x_2^3 + x_1x_3^2 + 3, \\
f_4(x_1, x_2, x_3) &= 7x_1^2x_2^4 + 18x_1x_2^3x_3^2 + x_2^3x_3^3.
\end{aligned}$$

The reduced Gröbner basis for the ideal generated by f_1, f_2, f_3, and f_4 over the field of rational numbers $\mathbb{Q}[x_1, x_2, x_3]$ is exceedingly simple [41]:

$$g_1 = x_3^2, \qquad g_2 = x_2^3 + \frac{1}{4}, \qquad g_3 = x_1.$$

The system of equations $f_1 = f_2 = f_3 = f_4 = 0$ is equivalent to the system $g_1 = g_2 = g_3 = 0$ which is apparently trivial to solve. However, the size of the polynomial coefficients appearing in the algorithm that produces the Gröbner basis itself may grow exponentially. In the discussed case, the following polynomial appears in intermediate computations:

$$x_2^3 - 1735906504290451290764747182\ldots.$$

The integer in the second term of this polynomial contains roughly 80000 digits, and is actually the numerator of a fraction that contains about the same number of digits in the denominator. This demonstrates that Gröbner bases, in particular over the field of rational numbers, may in some cases be extremely difficult to compute, and on the verge of being useless in practical applications. An interesting approach to overcoming this obstacle is described in [41].

2.6 Problems

2.6.1 Wien's Law and Lambert's Function

The distribution of the spectral energy density of black-body radiation [45] in terms of the wavelengths at temperature T is given by Planck's formula:

$$u(\lambda) = \frac{4\pi}{c}\frac{\mathrm{d}j}{\mathrm{d}\lambda} = \frac{8\pi hc}{\lambda^5}\frac{1}{\exp(hc/\lambda kT) - 1},$$

while the distribution in terms of the frequencies is given by

$$u(\nu) = \frac{4\pi}{c}\frac{\mathrm{d}j}{\mathrm{d}\nu} = \frac{8\pi h\nu^3}{c^3}\frac{1}{\exp(h\nu/kT) - 1}.$$

Both distributions have maxima that shift with temperature (Fig. 2.11). Wien's law describes the temperature dependence of the position of these maxima, and can be derived by solving the appropriate equations for the local extrema. In the case of the distribution in terms of the wavelengths, this translates to

$$\frac{\mathrm{d}^2 j}{\mathrm{d}\lambda^2} = 0 \quad \Longrightarrow \quad (x_\lambda - 5)\mathrm{e}^{x_\lambda} + 5 = 0, \tag{2.37}$$

while in the case of the frequency distribution, one has

$$\frac{\mathrm{d}^2 j}{\mathrm{d}\nu^2} = 0 \quad \Longrightarrow \quad (x_\nu - 3)\mathrm{e}^{x_\nu} + 3 = 0, \tag{2.38}$$

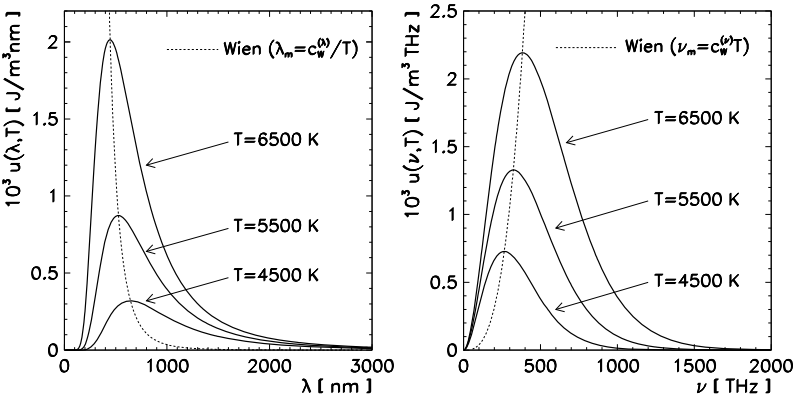


Fig. 2.11 Planck's law. [*Left*] Temperature dependence of the wavelength distribution of the spectral energy density. [*Right*] Temperature dependence of the frequency distribution. Also shown are the Wien curves connecting the distributions' maxima

where we have introduced $x_\lambda = hc/\lambda kT$ and $x_\nu = h\nu/kT$. These equations can be solved either analytically or numerically. The analytic solution is related to the Lambert function W_k, which represents different solutions of the equation

$$W_k \mathrm{e}^{W_k} = z, \quad z \in \mathbb{C}.$$

Lambert's functions are specified by the index k. We are concerned only about values $z \in \mathbb{R}$ and $W_k(z) \in \mathbb{R}$, for which two solutions are possible,

$$W_{-1}(z) \quad \text{at } z \in [-1/\mathrm{e}, 0], \qquad W_0(z) \quad \text{at } z \in [-1/\mathrm{e}, \infty),$$

shown in Fig. 2.12. The solutions of (2.37) and (2.38) can be expressed as

$$x_\lambda = 5 + W_0\big(-5\mathrm{e}^{-5}\big),$$
$$x_\nu = 3 + W_0\big(-3\mathrm{e}^{-3}\big).$$

In the Wien law, they appear as

$$\lambda_{\max} = \left(\frac{hc}{kx_\lambda}\right)\frac{1}{T} \equiv \frac{c_{\mathrm{W}}^{(\lambda)}}{T}, \qquad \nu_{\max} = \left(\frac{k}{h}\,x_\nu\right)T \equiv c_{\mathrm{W}}^{(\nu)}T. \tag{2.39}$$

$\odot$ Apply various numerical methods for the computation of zeros of nonlinear scalar functions to determine the values of W_0. Use simple iteration, the secant method, and the Newton–Raphson method. Optimize the computer program for speed and robustness. Draw the graph of W_0 for $x \in [-1/\mathrm{e}, 2]$ and calculate the constants x_λ and x_ν appearing in Wien's laws (2.39).

$\oplus$ Find the power expansion of the function W_0 in the vicinity of the point 0 by using the implicit-function theorem (p. 60). According to this theorem, for each

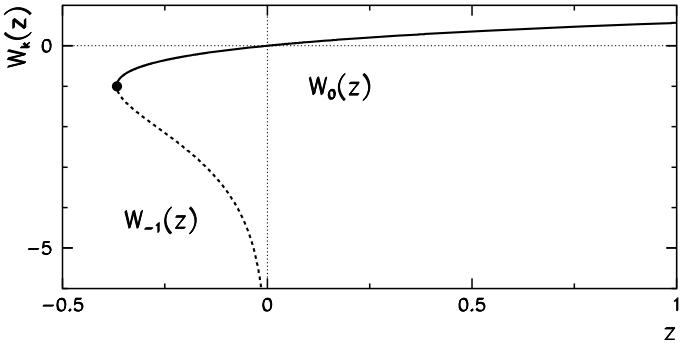


Fig. 2.12 Graphs of the Lambert functions W_0 and W_{-1}. For details see [46]

function f that is analytic near the point a, an inverse function g can be found, with the following power expansion near $f(a)$:

$$g(z) = a + \sum_{n=1}^{\infty} \lim_{x \to a} \left[\frac{\mathrm{d}^{n-1}}{\mathrm{d}x^{n-1}} \left(\frac{x-a}{f(x)-f(a)} \right)^n \right] \frac{(z-f(a))^n}{n!}.$$

For the Lambert function, $f(x) = x\mathrm{e}^x$ and $g(z) = W_0(z)$. Estimate the convergence radius of the power expansion of W_0 (use symbolic computation software).

2.6.2 Heisenberg's Model in the Mean-Field Approximation

Exact computations of thermo-dynamical equilibria of quantum spin systems are extremely time-consuming. One often resorts to simplified models in which knowledge from classical and quantum mechanics is combined. An instructive example is offered by the Heisenberg system of electron spins in a magnetic field [47]. The electrons are arranged along a circle. The energy of the system is given by the Hamiltonian

$$H = -\sum_{ij} J_{ij}\, \vec{s}_i \cdot \vec{s}_j + \gamma \vec{B} \cdot \sum_i \vec{s}_i, \quad \gamma = g\mu_\mathrm{B},$$

where $\mu_\mathrm{B} = e_0\hbar/(2m_\mathrm{e})$ is the Bohr magneton, $g = 2$ is the gyro-magnetic ratio of the electron, and $\vec{s}_i = (s_i^x, s_i^y, s_i^z)$ are the classical spin vectors with lengths $|\vec{s}_i| = 1/2$. The coupling between the spins J_{ij} has the property $J_{ij} = J(|i-j|)$. Matter with $J_{ij} > 0$ is ferromagnetic; matter with $J_{ij} < 0$ is anti-ferromagnetic. In the *mean-field approximation* (MFA) we substitute

$$\vec{s}_i \cdot \vec{s}_j \;\longrightarrow\; \langle\vec{s}\rangle \cdot \vec{s}_j + \langle\vec{s}\rangle \cdot \vec{s}_i - \left|\langle\vec{s}\rangle\right|^2,$$

where $\langle\vec{s}\rangle$ is the statistical average of the spin vectors, which is a macroscopic observable of the system. We will determine it by requiring thermodynamic equilibrium. The z-axis is pointing along $\vec{B}$ so that $\vec{B} = (0, 0, B)^{\mathrm{T}} \parallel (0, 0, \langle s^z\rangle)^{\mathrm{T}}$. In the MFA the Hamiltonian can be rewritten in the form

$$H_{\mathrm{MFA}} = -\gamma \sum_i B_{\mathrm{eff}} s_i^z, \quad B_{\mathrm{eff}} = B - \frac{J_0}{\gamma}\langle s^z\rangle, \; J_0 = \frac{1}{N}\sum_{i,j=1}^{N} J_{ij},$$

describing a system of decoupled spins in the effective magnetic field B_{eff}. We quantize the system again and allow only $s_i^z \in \{-1/2, 1/2\}$. The statistical sum Z corresponding to a single spin in a chain of N independent spins, is

$$Z = \exp(-\beta\gamma B_{\mathrm{eff}}/2) + \exp(\beta\gamma B_{\mathrm{eff}}/2) = 2\cosh(\beta\gamma B_{\mathrm{eff}}/2),$$

where $\beta^{-1} = k_{\mathrm{B}}T$. This sum can be used to compute the statistically averaged spin

$$\langle s^z\rangle = -\frac{1}{2}\tanh\left(\frac{1}{2}\beta\big(\gamma B - J_0\langle s^z\rangle\big)\right),$$

which should be interpreted as the self-consistent equation of the system. By appropriate substitutions, the equation can be cast in the form

$$z = \tanh(az - b),$$

where we have introduced the dimensionless quantities $z \propto \langle s^z\rangle$, $a \propto \beta J_0$, and $b \propto \beta\gamma B$.

$\bigodot$ Find the solutions for the average dimensionless spin z in the range of parameters $(a, b) \in [-2, 2] \times [-5, 5]$. Display the solutions in a manner clearly indicating the transitions where the character of the solution changes. Draw the curves of $z(a, b)$ at $b = 0$, 0.1, and 0.5, by using $a \in [0, 5]$.

$\bigoplus$ Discuss in more detail the region in the vicinity of the transition between the ferromagnetic phase ($|z| > 0$) and paramagnetic phase ($z = 0$). A well-known transition point is $(a, b) = (1, 0)$. Increase the strength of the magnetic field b and locate the point a at which the transition occurs. This gives you the dependence of a on b: plot it.

2.6.3 Energy Levels of Simple One-Dimensional Quantum Systems

Often, numerical methods are the only way to compute the spectra (eigenenergies) of quantum systems. This problem deals with one-dimensional systems for which the equations for the eigenenergies E or the corresponding wavenumbers k are relatively simple, but their determination requires us to solve transcendental equations. The non-relativistic Hamiltonian operator

$$\hat{H} = -\frac{\partial^2}{\partial x^2} + U(x),$$

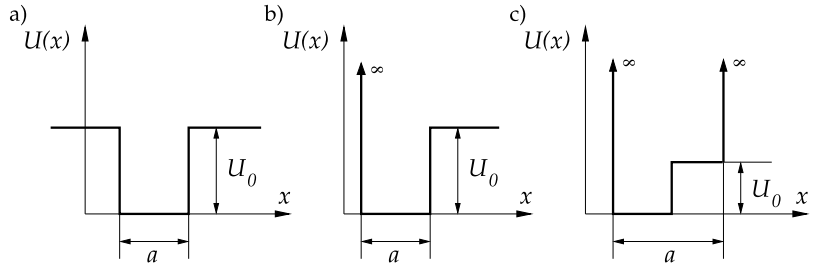


Fig. 2.13 Examples of one-dimensional potentials

contains the kinetic and the potential term. We are solving the stationary Schrödinger equation for the eigenenergies E and the eigenstates ψ_E,

$$\hat{H}\psi_E = E\psi_E, \quad E = k^2.$$

The spectrum may be discrete or continuous. Oscillation theorems tell us that energy degeneracies are impossible if the spectrum is discrete. Here we focus our attention to discrete spectra. The most simple systems involve piecewise constant potentials: three examples are shown in Fig. 2.13. These systems are described by the following eigenvalue equations: for a particle in the finite potential well with depth U_0 and width a (Fig. 2.13a):

$$\tan ka = \frac{2kk'}{k^2 - k'^2}, \quad k' = \sqrt{U_0 - k^2};$$

for a particle in a semi-infinite well with a step of height U_0 on one side (Fig. 2.13b):

$$k' \tan ka = -k, \quad k' = \sqrt{U_0 - k^2};$$

and for a particle in the infinite well with width a and a step of height U_0 reaching across one half of the well (Fig. 2.13c):

$$k' \tan \frac{1}{2}ka = -k \tan \frac{1}{2}k'a, \qquad k' = \sqrt{k^2 - U_0}.$$

We are interested in the states with energies exceeding U_0 (so $k^2 - U_0 > 0$). In the cases (a) and (b) we obtain a finite number of discrete energy states, while there are infinitely many in the case (c).

⊙ Find the energy states of particles in at least one of the cases specified above by using the method of bisection, the secant method, and the Newton–Raphson method. Set $U_0 = 1$. Compare the speed of convergence of all methods. Determine the order of convergence by plotting $\log|x_{k+1} - x_k|$ versus $\log|x_k - x_{k-1}|$, where x_k is the approximation of the root of the equation and k the iteration index. Consider the analytic structure of the equations when trying to figure out the initial approximations.

⊕ In the case (c) find all energy states starting from the lowest possible level and up to the level high enough that the asymptotic behavior of the solution can be ascertained. Confirm this behavior analytically.

2.6.4 Propane Combustion in Air

In concurrent chemical reactions proceeding in propane combustion in air,

$$\mathrm{C_3H_8} + \frac{R}{2}(4\mathrm{N_2} + \mathrm{O_2}) \quad \longrightarrow \quad \text{products},$$

different amounts x_i of the reaction products are formed, depending on the fraction R of air with respect to propane: CO_2 (x_1), H_2O (x_2), N_2 (x_3), CO (x_4), H_2 (x_5), H (x_6), OH (x_7), O (x_8), NO (x_9), and O_2 (x_{10}). The variables x_i correspond to the number of moles of the ith reaction product per mole of propane. A physical solution exists for $R > 3$. The equilibrium state at pressure 1 bar and temperature 2200 K is described by a system of non-linear equations [48]

$$\begin{aligned}
x_1 + x_4 - 3 &= 0, \\
2x_1 + x_2 + x_4 + x_7 + x_8 + x_9 + 2x_{10} - R &= 0, \quad R = 4.056734, \\
2x_2 + 2x_5 + x_6 + x_7 - 8 &= 0, \\
2x_3 + x_9 - 4R &= 0, \\
K_5 x_2 x_4 - x_1 x_5 &= 0, \quad K_5 = 0.193, \\
K_6\sqrt{x_2 x_4 S} - x_6\sqrt{x_1} &= 0, \quad K_6 = 0.002597, \\
K_7\sqrt{x_1 x_2 S} - x_7\sqrt{x_4} &= 0, \quad K_7 = 0.003448, \\
K_8 x_1 S - x_4 x_8 &= 0, \quad K_8 = 1.799 \times 10^{-5}, \\
K_9 x_1\sqrt{x_3 S} - x_4 x_9 &= 0, \quad K_9 = 2.155 \times 10^{-4}, \\
K_{10} x_1^2 S - x_4^2 x_{10} &= 0, \quad K_{10} = 3.846 \times 10^{-5},
\end{aligned}$$

where $S = \sum_{i=1}^{10} x_i$. (This sum can be taken as exact and inserted in the system for x_i, $1 \le i \le 10$, or it can be treated as an independent, eleventh equation.)

⊙ Solve the system of ten (or eleven) equations given above by using the Newton–Raphson and Broyden method for vector equations described in Sect. 2.2. Exploit the structure of the system in order to devise the best initial approximations for the iteration. How sensitive is the solution to the variations of the parameters R and K_i?

⊕ With poor initial approximations, negative arguments of the square roots may appear during the iteration. If you encounter such problems, replace

$$x_{i,\mathrm{new}}^2 = x_{i,\mathrm{old}}, \quad i = 1, 2, 3, 4,$$

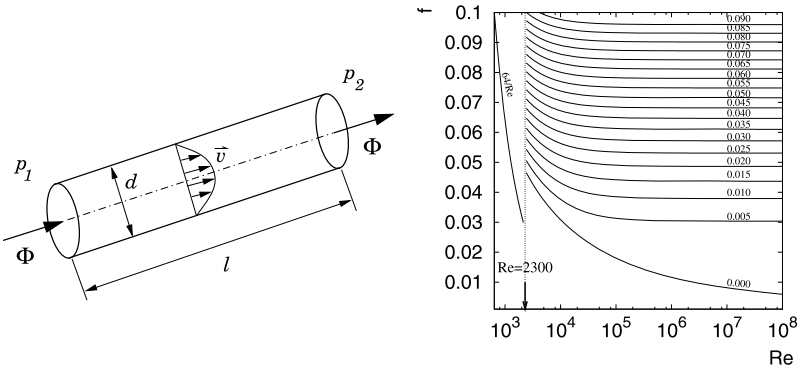


Fig. 2.14 [*Left*] Fluid flow in a pipe. The fluid enters at height z_1 and pressure p_1 with the average velocity $\overline{v}_1$, and exits at height z_2 and pressure p_2 with the velocity $\overline{v}_2$. The hydraulic diameter d for a pipe with cross-sectional area S and circumference o is defined by the ratio $4S/o$. [*Right*] Darcy's friction factor f as a function of the Reynolds number Re for different relative roughnesses e/d (simplified Moody diagram)

and repeat the calculation as before. You can also get rid of the negative arguments by taking their absolute values under all square roots, e.g. $\sqrt{|x_2 x_4 S|}$ instead of $\sqrt{x_2 x_4 S}$, or all equations are rewritten such that none of the x_i appears under the root sign. In all these approaches (see [48] for details), can you spot any differences in the convergence speed of Newton's or Broyden's methods?

2.6.5 Fluid Flow Through Systems of Pipes

Flow through pipes is not just a technological problem: it also offers lovely mathematical physics insights. Assume that the flow is unidirectional and occurs due to pressure gradients, neglecting any possible complex dynamics [49, 50]. Curved pipes are approximated by sequences of smoothly joined straight segments of length l and hydraulic diameter d (Fig. 2.14 (left)), along which Bernoulli's equation applies:

$$p_1 + \frac{1}{2}\rho\overline{v}_1^2 + \rho g z_1 = p_2 + \frac{1}{2}\rho\overline{v}_2^2 + \rho g z_2 + \Delta p,$$

where Δp is the pressure drop. The average velocity of the fluid $\overline{v}$, the cross-sectional area of the pipe S and the density ρ determine the mass flux $\Phi = \rho S\overline{v}$ which is constant along the pipe.

Due to the viscosity of the fluid, the velocity at the wall pipe is zero and the velocity across the pipe diameter is not constant. The pressure drops Δp along the pipe appear because of the viscosity, the roughness of the pipe walls, and the dynamics in the fluid itself. Phenomenologically, the pressure drop is given by the

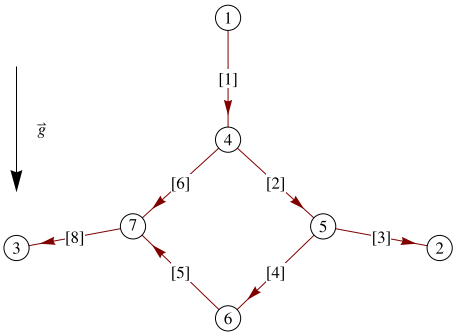


Fig. 2.15 The graph of a test pipe system. The vertices are denoted by indices in circles, and the connections are denoted by indices in brackets. We have N connections, M vertices, and m free ends. The remaining $M' = M - m$ vertices are located at known heights h_i and the losses in them are assumed to be negligible. The pressures in the vertices are denoted by p_i where i is the index of the vertex

Darcy–Weisbach equation [51]

$$\Delta p = f\frac{l}{2d}\rho\overline{v}|\overline{v}| = R\Phi|\Phi|, \quad R = f\frac{l}{2d\rho S^2},$$

where f is the Darcy friction factor and R the resistance coefficient. For long pipes R depends only on the relative roughness e/d and the Reynolds number $Re = \rho\overline{v}d/\mu = 4\Phi/(\pi\mu d)$, where μ is the dynamical viscosity of the fluid. The roughness e is defined as the standard deviation of the pipe radius, averaged over its interior surface. In the laminar regime of the flow ($Re < 2300$) f is determined by the Hagen–Poiseuille equation, $f = 64/Re$, while in the intermediate ($2300 < Re < 4000$) and in the turbulent regime ($Re > 4000$) we can read it off from the Moody diagram (Fig. 2.14 (right)). In the turbulent regime, a good approximation for f is given by the implicit Colebrook–White equation

$$\frac{1}{\sqrt{f}} = -2\log_{10}\left(\frac{e/d}{3.7} + \frac{2.51}{Re\sqrt{f}}\right).$$

In pipe systems the dominant pressure drops (dominant energy losses) are caused by long pipes; relatively small losses are caused by the joining elements or sharp turns. At the known flux $\Phi = \rho S\overline{v}$, the pressure drop along some element is given by the equation

$$\Delta p = K\frac{1}{2}\rho\overline{v}|\overline{v}|,$$

where $K = fl/d$ is the loss coefficient (e.g. 0.35 for a 45° elbow or 0.75 for a 90° elbow).

An example of a network of pipes in the gravitational field is illustrated by the graph of vertices and connections in Fig. 2.15. We are interested in the stationary flow along the pipes when the pressures at the free ends of the graph are known. For

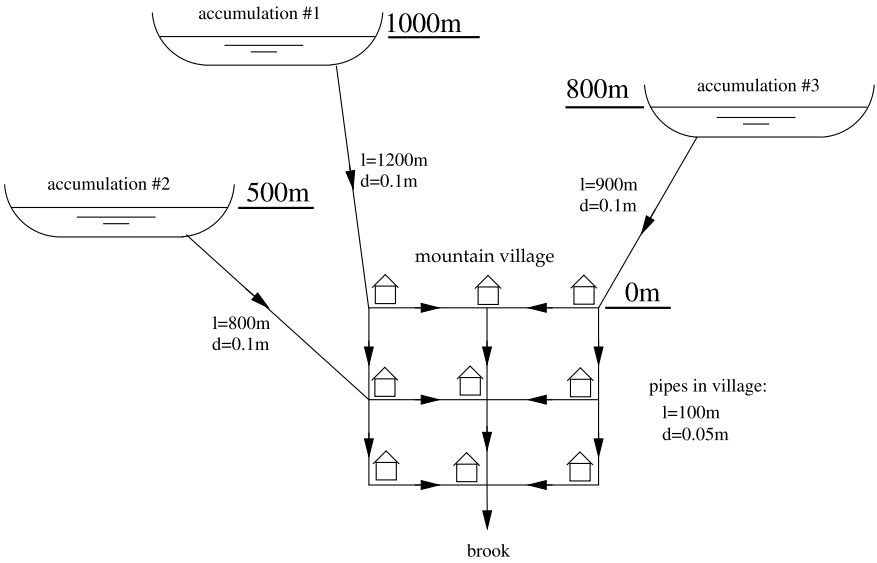


Fig. 2.16 A cartoon of the system of pipes of a mountain village

the mass flux Φ_j in the individual segment, Darcy–Weisbach equation applies. For the jth pipe connecting vertices i and k, it reads

$$R_j|\Phi_j|\Phi_j = \tilde{p}_i - \tilde{p}_k, \quad \tilde{p}_i = p_i + \rho g h_i,$$

where $\tilde{p}_i$ are the hydrostatic pressures. (The resistance coefficients R_j depend on $Re \propto \Phi_j$.) At each vertex, the sum of the incoming mass fluxes should be equal to the sum of the outgoing mass fluxes. By defining the vector of fluxes $\boldsymbol{\Phi} = (\Phi_i)_{i=1}^N$ all continuity equations can be written in the matrix form $A\boldsymbol{\Phi} = \mathbf{0}$, where the matrix $A \in \mathbb{R}^{M' \times N}$ contains the information about the connectedness of the graph. The hydrostatic pressures in the internal vertices $\{i_1, \ldots, i_{M'}\}$ are arranged in the vector $\tilde{\boldsymbol{p}} = (\tilde{p}_{i_1}, \ldots, \tilde{p}_{i_{M'}})^{\mathrm{T}}$. In addition, we define the diagonal matrix $\Sigma(\boldsymbol{\Phi}) = \mathrm{diag}(R_j|\Phi_j|)_{j=1}^N$. All Darcy–Weisbach equations for the system can then be written in the matrix form $\Sigma\boldsymbol{\Phi} + A^{\mathrm{T}}\tilde{\boldsymbol{p}} = \boldsymbol{P}$, where the vector $\boldsymbol{P} = (P_1, \ldots, P_N)^{\mathrm{T}}$ contains only the hydrostatic pressures at the free ends ($\rho g h$ for points at height h, e.g. the lakes in Fig. 2.16). The Darcy–Weisbach and the continuity equations must be solved simultaneously and can be rewritten as a single system of non-linear equations $F(\boldsymbol{x})\boldsymbol{x} = \boldsymbol{b}$, where

$$F(\boldsymbol{x}) = \begin{bmatrix} \Sigma(\boldsymbol{\Phi}) & A^{\mathrm{T}} \\ A & 0 \end{bmatrix}, \qquad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{\Phi} \\ \tilde{\boldsymbol{p}} \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{P} \\ \mathbf{0} \end{bmatrix}.$$

The equation can be solved by the iteration

$$\boldsymbol{x}_{n+1} = F_n^{-1}\boldsymbol{b}, \quad F_n = F(\boldsymbol{x}_n).$$

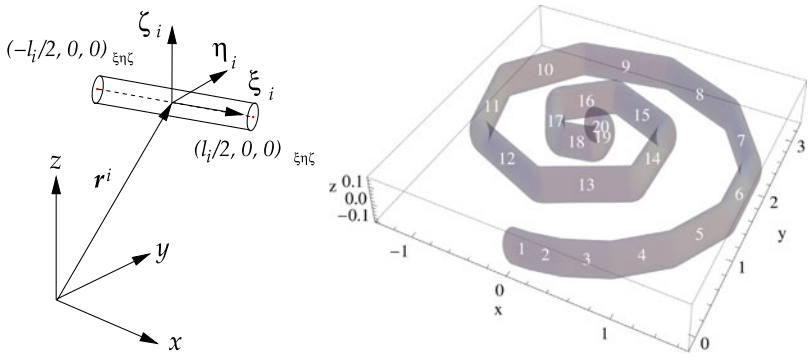


Fig. 2.17 [*Left*] A drawing of the ith rod with its local and global reference frames. [*Right*] An example of a linear structure assembled from rods of diameter $d = 0.2$

We form the initial matrix F_0 in the laminar regime where, for the jth pipe, we have $R_j|\Phi_j| = 32\mu l_j/(\rho d_j^2 S_j)$.

⊙ A mountain village is supplied by water from three accumulations lying at the mountain slopes. The whole village is at constant altitude. The consumption of water in the village is negligible; the remainder of the water drains into the brook. The cartoon of the system of pipes is shown in Fig. 2.16. The relative roughness of the pipes is $e/d = 0.01$. Draw the graph of this system and compute the mass fluxes through the individual pipes if the Colebrook–White equation is used to compute the friction factor. Neglect all losses.

⊕ Discuss the fluid flow in a random two-dimensional pipeline on a Cartesian grid. On this grid, select the origin whence a random walker starts his walk $\mathcal{N}$-times. The speed of the walk is one unit of length on the mesh per unit time. An individual walk takes $\mathcal{M}$ units of time. We interpret all $\mathcal{N}$ walks as a random pipeline composed of straight pipes of equal lengths. At the origin, attach a water reservoir with a constant over-pressure p_0. The loss coefficient in the pipes K is constant. (Choose the units such that $K = p_0 = 1$.) We are interested in the distribution of the flows in the pipes in the cases $\mathcal{N} = \mathcal{M} = 10$, $\mathcal{N} = \mathcal{M} = 100$, and $\mathcal{N} = \mathcal{M} = 1000$. Can you infer a scaling law from these distributions?

2.6.6 Automated Assembly of Structures

Because so many building blocks are involved, the simulation of robots, motion and collisions of vehicles, proteins and other complex systems of bodies require the use of automated, *computer-aided assembly* [32]. The manner in which individual components are connected is embodied in the equations known as *constraints*. Based on these constraints, the program computes the structure of the body in three-dimensional space. In virtually all physically relevant types of connections, the constraints can be written as systems of algebraic equations.

Here we discuss the assembly of a simple linear structure in the (x, y) plane in the reference space $\mathbb{R}^3$ with the coordinate axes $\{x, y, z\}$. We would like to automatically assemble a chain of m connected rods. We denote the rods by their indices $1 \le i \le m$ and their lengths by l_i. The $(i+1)$th rod should be rotated with respect to the ith rod by the angle θ_i. The center-of-mass of the ith rod is the origin of its local reference frame $\mathcal{S}^i$ with the axes $\{\xi_i, \eta_i, \zeta_i\}$ such that the ξ_i axis is along the rod, the ζ_i axis is aligned with the z axis of the global reference frame, and the edges of the rod are at $\boldsymbol{s}^i_\pm = (\pm l_i/2, 0, 0)^{\mathrm{T}}$ (see Fig. 2.17 (left)). The position of the ith rod is described by the vector $\boldsymbol{r}^i = (x^i, y^i, z^i)^{\mathrm{T}}$ from the origin of the frame $\mathcal{S}^i$, while its orientation is specified by the vector of Euler parameters $\boldsymbol{e}^i = (e^i_0, e^i_1, e^i_2, e^i_3)^{\mathrm{T}}$ [52]. The Euler parameters $\boldsymbol{e} = (e_0, e_1, e_2, e_3)^{\mathrm{T}}$ are used to define the rotation by an angle ϕ around an arbitrary axis $\hat{\boldsymbol{n}}$ ($\hat{\boldsymbol{n}}^{\mathrm{T}}\hat{\boldsymbol{n}} = 1$) as

$$e_0 = \cos\frac{\phi}{2}, \qquad \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} = \hat{\boldsymbol{n}} \sin\frac{\phi}{2}, \qquad \sum_{\rho=0}^{3} e_\rho^2 = 1.$$

The corresponding rotation matrix $R(\boldsymbol{e}) = [R_{ij}(\boldsymbol{e})]_{i,j=1}^3$ is then

$$R_{ij}(\boldsymbol{e}) = \delta_{i,j}\left(e_0^2 - e_k e_k\right) + 2e_i e_j + 2\varepsilon_{ijk} e_0 e_k,$$

where $\delta_{i,j}$ is the Kronecker delta, ε_{ijk} is the anti-symmetric tensor and Einstein's summation convention applies: at the right-hand side, we sum over all k.

All information on the ith rod are collected in the vector $\boldsymbol{q}^i = (\boldsymbol{r}^i, \boldsymbol{e}^i)$ of dimension 7. The condition

$$R_{33}\left(\boldsymbol{e}^i\right) - 1 = 0 \quad \forall i$$

restricts the assembly of the chain to the (x, y) plane. The joining of the end of the ith rod by the beginning of the $(i+1)$th rod is expressed by the constraint

$$R\left(\boldsymbol{e}^i\right)\boldsymbol{s}^i_+ + \boldsymbol{r}^i - R\left(\boldsymbol{e}^{i+1}\right)\boldsymbol{s}^{i+1}_- - \boldsymbol{r}^{i+1} = \boldsymbol{0}.$$

In our case $\hat{\boldsymbol{n}} = (0, 0, 1)^{\mathrm{T}}$, so the Euler parameters $\boldsymbol{e}^i$ for the ith rod depend only on the rotation angles of the individual rod, ϕ_i, and these are simply determined by the constraints $\phi_{i+1} - \phi_i - \theta_i = 0$. These constraints are not analytic in the parameters $\boldsymbol{r}^i$ and $\boldsymbol{e}^i$; due to the periodicity of the angles, they are also not unique. We therefore choose slightly more complicated constraints

$$R_{11}\left(\boldsymbol{e}^{i+1}\right) + \mathrm{i}\, R_{21}\left(\boldsymbol{e}^{i+1}\right) - \exp\left(\mathrm{i}\sum_{j=1}^{i}\theta_j\right)\left(R_{11}\left(\boldsymbol{e}^i\right) + \mathrm{i}\, R_{21}\left(\boldsymbol{e}^i\right)\right) = 0,$$

for $i = 1, 2, \ldots, m-1$, which are analytic. We exploit the fact that the projections of the rod's orientation on the x and y axes are hidden in the components of the rotation matrices R_{11} and R_{21}. Without loss of generality, the first rod may be oriented along the x axis, which implies an additional constraint $R_{11}(\boldsymbol{e}^1) - 1 = 0$.

⊙ Let the linear structure described above reside in the plane at $z = 0$ and let the beginning of the first rod coincide with the origin of the global reference frame. For some m, automatically assemble the structure corresponding to the lengths and angles

$$l_i = \sin\frac{\pi\, i}{m+1}, \qquad \theta_i = \frac{\pi\, i}{2(m+1)}, \quad i = 1, 2, \ldots, m.$$

The system can be described by the vector of positions and Euler parameters of all rods

$$\boldsymbol{q} = \big(\boldsymbol{q}^i\big)_{i=1}^m$$

of dimension $M = 7m$. All constraints of the system can be collected in the vector equation

$$\boldsymbol{F}(\boldsymbol{q}) = \big(F_i(\boldsymbol{q})\big)_{i=1}^N = \boldsymbol{0}, \tag{2.40}$$

which is analytic in $\boldsymbol{q}$ with $N \geq M$. The solution of this system requires us to find the global minimum of the quantity

$$\Phi(\boldsymbol{q}) = \boldsymbol{F}(\boldsymbol{q})^{\mathrm{T}} \boldsymbol{F}(\boldsymbol{q}), \tag{2.41}$$

which, in general, is troublesome. Here, we are only seeking the local minimum, so we are solving $\nabla\Phi(\boldsymbol{q}) = \boldsymbol{0}$. Since Φ is analytic in $\boldsymbol{q}$, its minimum can be found by using the Newton method of *unconstrained minimization* [14]. This method requires us to know the first derivative of (2.41),

$$\boldsymbol{D}(\boldsymbol{q}) = \big(\partial_i \Phi(\boldsymbol{q})\big)_{i=1}^M, \quad \partial_i \Phi(\boldsymbol{q}) = 2\big[\partial_i F_j(\boldsymbol{q})\big] F_j(\boldsymbol{q}),$$

where $\partial_i \equiv \partial/\partial q_i$, as well as the second derivative, given by the Hessian matrix,

$$H(\boldsymbol{q}) = \big[\partial_i \partial_j \Phi(\boldsymbol{q})\big]_{i,j=1}^M,$$
$$\partial_i \partial_j \Phi(\boldsymbol{q}) = 2\big[\partial_i \partial_j F_k(\boldsymbol{q})\big] F_k(\boldsymbol{q}) + 2\big[\partial_j F_k(\boldsymbol{q})\big] \partial_i F_k(\boldsymbol{q}).$$

(Einstein's summation convention applies throughout.) We choose a good enough initial approximation of the solution $\boldsymbol{q}^{(0)}$ and iterate

$$\boldsymbol{q}^{(k+1)} = \boldsymbol{q}^{(k)} + \big[H\big(\boldsymbol{q}^{(k)}\big)\big]^{-1} \boldsymbol{D}\big(\boldsymbol{q}^{(k)}\big). \tag{2.42}$$

This procedure is well suited for solving small or medium-sized systems where m is on the order of several 10 to several 100, and M on the order of 100 to 1000. We choose $m = 20$. The solution should appear as a structure shown in Fig. 2.17 (right). For larger systems, solving the system of equations

$$H\big(\boldsymbol{q}^{(k)}\big)\big(\boldsymbol{q}^{(k+1)} - \boldsymbol{q}^{(k)}\big) = \boldsymbol{D}\big(\boldsymbol{q}^{(k)}\big)$$

at each iteration step (2.42) may be too time-consuming and numerically unstable. In such cases, other methods should be called to rescue: see [14].

⊕ Another possibility to solve (2.40) leads through Gröbner bases (see Sect. 2.5). High numerical costs of this approach restrict m to about a few times 10; we choose $m = 3$. For a given set of functions $\{F_i(\boldsymbol{q})\}_{i=1}^{N}$ compute the reduced Gröbner basis $\mathcal{G} = \{g_i\}_{i=1}^{M}$ with lexicographic ordering. Solve the equations $g_i(\boldsymbol{q}) = 0$ for $i = 1, 2, \ldots, M$ in this very same index sequence, i.e. make use of the solutions of equations with index $j \in [1, i-1]$ in computing the solution of the equation with index i. Along the way, non-physical solutions can be eliminated. With such an approach, each equation $g_i(\boldsymbol{q}) = 0$ depends only on a single variable, and the index of the variable increases with the index of the equation under consideration. Hence, standard methods for seeking zeros of polynomials can be applied. If multiple solutions appear, they all describe the same spatial structure, which one easily confirms by drawing it.

References

1. A.E. Dubinov, I.N. Galidakis, Explicit solution of the Kepler equation. Phys. Part. Nucl. Lett. **4**, 213 (2007)
2. R. Luck, J.W. Stevens, Explicit solutions for transcendental equations. SIAM Rev. **44**, 227 (2002)
3. G.R. Wood, The bisection method in higher dimensions. Math. Program. **55**, 319 (1992)
4. W. Baritompa, Multidimensional bisection: a dual viewpoint. Comput. Math. Appl. **27**, 11 (1994)
5. J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, 3rd edn. Texts in Appl. Math., vol. 12 (Springer, Berlin, 2002)
6. E.D. Charles, J.B. Tatum, The convergence of Newton–Raphson iteration with Kepler's equation. Celest. Mech. Dyn. Astron. **69**, 357 (1998)
7. B.A. Conway, An improved algorithm due to Laguerre for the solution of Kepler's equation. Celest. Mech. **39**, 199 (1986)
8. H. Susanto, N. Karjanto, Newton's method's basins of attraction revisited. Appl. Comput. Math. **215**, 1084 (2009)
9. J.A. Ford, Improved algorithms of Illinois-type for the numerical solution of nonlinear equations. Technical report CSM-257, University of Essex (1995)
10. A. Ralston, H.S. Wilf, *Mathematical Methods of Digital Computers*, vol. 2 (Wiley, New York, 1967), Chap. 9
11. J.F. Traub, *Iterative Methods for the Solution of Equations* (Prentice-Hall, Englewood Cliffs, 1964)
12. W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 3rd edn. (Cambridge University Press, Cambridge, 2007). See also the equivalent handbooks in Fortran, Pascal and C, as well as http://www.nr.com
13. G. Dahlquist, Å. Björck, *Numerical Methods in Scientific Computing, Vols. 1 and 2* (SIAM, Philadelphia, 2008)
14. J.E. Dennis Jr., R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (SIAM, Philadelphia, 1996)
15. C.G. Broyden, A class of methods for solving nonlinear simultaneous equations. Math. Comput. **19**, 577 (1965)
16. J.J. Moré, J.A. Trangenstein, On the global convergence of Broyden's method. Math. Comput. **30**, 523 (1976)
17. D.M. Gay, Some convergence properties of Broyden's method. SIAM J. Numer. Anal. **16**, 623 (1979)

18. C.G. Broyden, On the discovery of the "good Broyden" method. Math. Program., Ser. B **87**, 209 (2000)
19. D. Li, J. Zeng, S. Zhou, Convergence of Broyden-like matrix. Appl. Math. Lett. **11**, 35 (1998)
20. J.M. Martínez, Practical quasi-Newton methods for solving nonlinear systems. J. Comput. Appl. Math. **124**, 97 (2000)
21. P. Henrici, *Applied and Computational Complex Analysis*, vol. 1 (Wiley, New York, 1974)
22. D.E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 2nd edn. (Addison-Wesley, Reading, 1980)
23. M. Marden, *The Geometry of Zeros of a Polynomial in the Complex Variable* (Am. Math. Soc., New York, 1949)
24. M. Fujiwara, Über die obere Schranke des absoluten Betrages der Wurzeln einer algebraischen Gleichung. Tohoku Math. J. **10**, 167 (1916)
25. D. Kincaid, W. Cheney, *Numerical Analysis. Mathematics of Scientific Computing* (Brooks/Cole, Belmont, 1991)
26. E.B. Vinberg, *A Course in Algebra* (Am. Math. Soc., Providence, 2003)
27. L.N. Trefethen, D. Bau, *Numerical Linear Algebra* (SIAM, Philadelphia, 1997)
28. M.A. Jenkins, J.F. Traub, A three-stage variable-shift iteration for polynomial zeros and its relation to generalized Rayleigh iteration. Numer. Math. **14**, 252 (1970)
29. M.A. Jenkins, J.F. Traub, A three-stage algorithm for real polynomials using quadratic iteration. SIAM J. Numer. Anal. **7**, 545 (1970)
30. E. Isaacson, H.B. Keller, *Analysis of Numerical Methods* (Wiley, New York, 1966)
31. H.J. Stetter, *Numerical Polynomial Algebra* (SIAM, Philadelphia, 2004)
32. E.J. Haug, *Computer Aided Kinematics and Dynamics of Mechanical Systems*, vol. 1 (Allyn and Bacon, Boston, 1989)
33. M. Sala, T. Mora, L. Perret, S. Sakata, C. Traverso (eds.), *Gröbner Bases, Coding, and Cryptography* (Springer, Berlin, 2009)
34. J. Gago-Vargas et al., Sudokus and Gröbner bases: not only a divertimento, in *CASC 2006*, ed. by V.G. Ganzha, E.W. Mayr, E.V. Vorozhtsov. Lecture Notes in Computer Science, vol. 4194 (Springer, Berlin, 2006), p. 155
35. V.R. Romanovski, D.S. Shafer, *The Center and Cyclicity Problems: A Computational Algebra Approach* (Birkhäuser, Boston, 2009)
36. W. Adams, P. Loustaunau, *An Introduction to Gröbner Bases*. Graduate Studies in Mathematics, vol. 3 (Am. Math. Soc., Providence, 1994)
37. D. Cox, J. Little, D. O'Shea, *Ideals, Varieties, and Algorithms*, 3rd edn. (Springer, Berlin, 2007)
38. E. Roanes-Lozano, E. Roanes-Macías, L.M. Laita, Some applications of Gröbner bases. Comput. Sci. Eng. **May/Jun**, 56 (2004)
39. E. Roanes-Lozano, E. Roanes-Macías, L.M. Laita, The geometry of algebraic systems and their exact solving using Gröbner bases. Comput. Sci. Eng. **March/April**, 76 (2004)
40. S. Mac Lane, G. Birkhoff, *Algebra*, 3rd edn. (Am. Math. Soc., Providence, 1999)
41. V.R. Romanovski, M. Prešern, An approach to solving systems of polynomials via modular arithmetics with applications. J. Comput. Appl. Math. **236**, 196 (2011)
42. B. Buchberger, An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. J. Symb. Comput. **41**, 475 (2006)
43. T. Stegers, Faugère's F5 algorithm revisited. Diploma thesis, Technische Universität Darmstadt (2005/2007)
44. J.C. Faugère, A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5), in *Proc. International Symposium on Symbolic and Algebraic Computation*, Lille, France (2002), p. 75
45. L. Schiff, *Quantum Mechanics* (McGraw-Hill, New York, 1968)
46. F. Chapeau-Blondeau, A. Monir, Numerical evaluation of the Lambert W-function and application to generation of generalized Gaussian noise with exponent $1/2$. IEEE Trans. Signal Process. **50**, 2160 (2002)

47. H. Gould, J. Tobochnik, *Statistical and Thermal Physics: With Computer Applications* (Princeton University Press, Princeton, 2010)
48. K. Meintjes, A.P. Morgan, Chemical equilibrium systems as numerical test problems. ACM Trans. Math. Softw. **16**, 143 (1990)
49. O. Reynolds, An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous and of the law of resistance in parallel channels. Proc. R. Soc. Lond. **35**, 84 (1883)
50. H. Faisst, B. Eckhardt, Sensitive dependence on initial conditions in transition to turbulence in pipe flow. J. Fluid Mech. **504**, 343 (2004)
51. B.E. Larock, R.W. Jeppson, G.Z. Watters, *Hydraulics of Pipeline Systems* (CRC Press, Boca Raton, 2002)
52. H. Goldstein, *Classical Mechanics*, 2nd edn. (Addison-Wesley, Reading, 1980)

Chapter 3
Matrix Methods

For physicist's needs, numerical linear algebra is so comprehensively covered by classic textbooks [1, 2] that another detailed description of the algorithms here would be pointless. To a larger extent than in other chapters we wish to merely set up road-signs between approaches to solving systems of linear equations, least-squares problems, and eigenvalue problems. Only Sect. 3.5 on random matrices conveys a somewhat different tone. Above all, we shall pay attention to classes of matrices for which analytical and numerical tools are particularly thoroughly developed and efficient.

To numerically solve problems in linear algebra, never write your own routines! This advice applies even to seemingly straightforward operations like matrix multiplication, and even more so to solving systems of equations $Ax = b$ or eigenvalue problems $Ax = \lambda x$. For these tasks we almost invariably use specialized libraries (see, for example, [3, 4] and Appendix I).

3.1 Basic Operations

3.1.1 Matrix Multiplication

Classical multiplication of $n \times n$ matrices in the form

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}, \quad i, j = 1, 2, \ldots, n, \tag{3.1}$$

where $A_{ik} B_{kj}$ is nested into three loops over i, j, and k, requires $F = 2n^3$ multiplications or additions and $M = n^3 + 3n^2$ memory accesses. The optimal ratio is $F/M \approx n/2$ [1], so this method of multiplication is not optimal. (Note that M may depend on processor architecture, peculiarities of the programming language, implementation, and compiler.) In standard libraries BLAS3 or LAPACK (see Appendix I) multiplication is realized in block form that also requires $F = 2n^3$ arithmetic operations, but has a better asymptotic ratio F/M for large n. For all forms

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9_3,

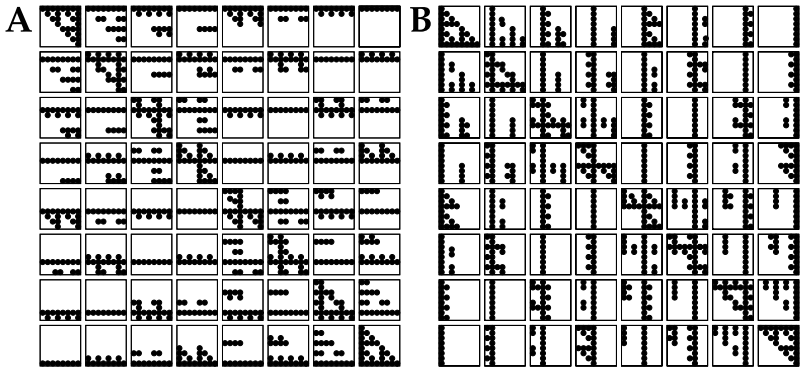


Fig. 3.1 Strassen's multiplication of 8×8 matrices. The *diagrams* show the elements of A and B which are accessed in memory in order to form the elements of C. For example, to get the C_{18} element (*upper right corners*) we need A_{1k} $(1 \le k \le 8)$ and B_{k8} $(1 \le k \le 8)$ as in classical multiplication (3.1), but a different pattern for other elements

of direct multiplication in floating-point arithmetic we have the estimate

$$\mathrm{fl}(AB) = AB + E, \quad |E| \le n\frac{\varepsilon_{\mathrm{M}}}{2}|A|\,|B| + \mathcal{O}\big(\varepsilon_{\mathrm{M}}^2\big), \tag{3.2}$$

where $|A|$ means $(|A|)_{ij} = |A_{ij}|$. We also have

$$\big\|\mathrm{fl}(AB) - AB\big\|_1 \le n\frac{\varepsilon_{\mathrm{M}}}{2}\|A\|_1\|B\|_1 + \mathcal{O}\big(\varepsilon_{\mathrm{M}}^2\big). \tag{3.3}$$

Faster algorithms exist, e.g. Strassen's [5] that splits the matrices A and B into smaller blocks which are then recursively multiplied and summed. The asymptotic cost of the algorithm is $4.70\,n^{2.81}$, so it really starts to soar when used with matrices of dimensions n in the hundreds or thousands. In some cases, Strassen's method may exhibit instabilities, but robust versions are contained in the fast version BLAS3 [6, 7] and we should use them if speed is our absolute priority. For optimally fast multiplication we need to understand the connection between hardware and software; see Fig. 3.1 and [8]. We can see in the anatomically detailed paper [9] just how deep this knowledge should be in order to design the best algorithms. For Strassen's multiplication the estimate (3.3) still applies while (3.2) is *almost always* true (see [2] and Chap. 23 in [10]).

The currently lowest asymptotic cost is claimed by the Coppersmith–Winograd algorithm [11] which is of order $\mathcal{O}(n^{2.38})$, but the error constant in front of $n^{2.38}$ is so large that the algorithm becomes useful only at very large n. A standard version is described in [12] and the best adaptive implementation in [13]. The theoretical lower limit of the numerical cost for any matrix multiplication algorithm is, of course, $2n^2$, since each element of A and B needs to be accessed at least once. See also [14–16].

3.1.2 Computing the Determinant

The determinant of a matrix A or its permutation PA should never be computed with the school-book method via sub-determinants. Rather, we use LU decomposition (3.7) and read off the determinant from the diagonal elements of U:

$$\det(PA) = (-1)^{\det(P)} \prod_{i=1}^{n} U_{ii}, \quad \det(P) = \text{order of permutation.}$$

3.2 Systems of Linear Equations $Ax = b$

This section deals with methods to solve systems of linear equations $Ax = b$, where A is a $n \times n$ matrix and b and x are vectors of dimension n. The routines from good linear algebra libraries (e.g. LAPACK) allow us to solve the system for r right-hand sides simultaneously (we are then solving $AX = B$ where B and X are $n \times r$ matrices). Solving $Ax = b$ is equivalent to "computing the inverse of A" although A^{-1} almost never needs to be explicitly known or computed. Table 3.1 summarizes some of the most widely used routines.

3.2.1 Analysis of Errors

Numerical errors in solving the problem $Ax = b$ can be analyzed in two ways. One way is to observe the sensitivity of the solution x to small perturbations of A or b. If the vector $\widehat{x}$ solves the perturbed equation $(A + \delta A)\widehat{x} = b + \delta b$ where $\|\delta A\| \le \varepsilon \|A\|$, $\|\delta b\| \le \varepsilon \|b\|$, and $\varepsilon \|A^{-1}\| \|A\| < 1$, the deviation $x - \widehat{x}$ from the exact solution can be bounded as

$$\frac{\|x - \widehat{x}\|}{\|\widehat{x}\|} \le \frac{\varepsilon}{1 - \varepsilon \|A^{-1}\| \|A\|} \left\{ \|A^{-1}\| \|A\| + \frac{\|A^{-1}\| \|b\|}{\|x\|} \right\} \le \frac{2\varepsilon \kappa(A)}{1 - \varepsilon \kappa(A)}, \qquad (3.4)$$

where $\kappa(A) \equiv \|A^{-1}\| \|A\|$ is the *condition number* of A with respect to matrix inversion (see Sect. 3.2.6). Matrices with large $\kappa(A)$ are *ill-conditioned*, those with small $\kappa(A)$ are *well-conditioned*. The value of the condition number depends on the norm in which it is measured. For example, in the Euclidean norm we have $\kappa_2(A) = \|A^{-1}\|_2 \|A\|_2 = \sigma_{\max}(A)/\sigma_{\min}(A)$ where $\sigma_{\max}(A)$ is the largest and $\sigma_{\min}(A)$ the smallest singular value of A (see Sect. 3.3.2). For nonsingular A the inverse value of the condition number $1/\kappa_p(A)$ measures the distance (in p-norm) to the nearest singular problem [17, 18],

$$\frac{1}{\kappa_p(A)} = \min_{A + \Delta A \text{ singular}} \frac{\|\Delta A\|_p}{\|A\|_p}.$$

Table 3.1 A collection of double-precision routines to solve systems of linear equations $Ax = b$ from the LAPACK, GSL and NUMERICAL RECIPES (third edition for C++) libraries, for different types of $n \times n$ matrices A. Routines from the LAPACK library allow the system to be solved simultaneously for r right-hand sides, while in GSL and NR3E we have $r = 1$. The k_l and k_u denote the number of sub-diagonals and super-diagonals in banded matrices ($k_l = k_u = k$ for symmetric matrices). Many routines from LAPACK are also available in single precision (first letter `S` instead of `D`) for real variables, as well as in double and single precision for complex variables (initial letters `Z` or `C`). For details, see [3, 4]. The GSL library also offers a few routines for complex variables

Matrix type	Numerical cost	LAPACK	GSL	NR3E
General	$\frac{2}{3}n^3 + 2n^2 r$	`DGESV` `DGESVX`	`gsl_linalg_LU_decomp` `gsl_linalg_LU_solve`	`LUdcmp`
General banded	$\mathcal{O}(nk_l(k_l + k_u) + n(2k_l + k_u)r)$ if $n \gg k_l, k_u$	`DGBSV` `DGBSVX`	/	`Bandec`
General tridiagonal	$\mathcal{O}(nr)$	`DGTSV` `DGTSVX`	/	`tridag`
Symmetric indefinite	$\frac{1}{3}n^3 + \mathcal{O}(n^2 r)$	`DSYSV` `DSYSVX`	/	/
Symmetric pos. definite	$\frac{1}{3}n^3 + \mathcal{O}(n^2 r)$	`DPOSV` `DPOSVX`	`gsl_linalg_cholesky_decomp` `gsl_linalg_cholesky_solve`	`Cholesky`
Symm. banded pos. definite	$\mathcal{O}(n(k+1)^2 + nkr)$ if $n \gg k$	`DPBSV` `DPBSVX`	/	/
Symm. tridiag. pos. definite	$\mathcal{O}(nr)$	`DPTSV` `DPTSVX`	`gsl_linalg_solve_symm_tridiag`	/
Toeplitz	$\mathcal{O}(n^2)$ $\mathcal{O}(n \log^2 n)$ [23, 24]	/	/	`toeplz`
Vandermonde	$\mathcal{O}(n^2)$ $\mathcal{O}(n \log^2 n)$ [27–33]	/	/	`vander`

Example A singular matrix has $\det(A) = 0$, but $\det(A) \approx 0$ does not necessarily mean that the problem $Ax = b$ is "almost singular". Let us elucidate this contrast by two examples ([2], p. 82). The upper-triangular $n \times n$ matrix U_n with values of 1 along the main diagonal and -1 on all super-diagonals, has $\det(U_n) = 1$ but $\kappa_\infty(U_n) = n2^{n-1}$. On the other hand, the $n \times n$ matrix $D_n = \mathrm{diag}(10^{-1}, 10^{-1}, \ldots, 10^{-1})$ has a condition number of $\kappa_p(D_n) = 1$ while $\det(D_n) = 10^{-n}$. Ill conditioning of the matrix (large κ) and the determinant being close to zero are therefore, in general, weakly correlated.

The bound (3.4) may over-estimate the actual error by many orders of magnitude, which does not hurt, but there is also no benefit. The estimate [10]

$$\frac{\|x - \widehat{x}\|_\infty}{\|\widehat{x}\|_\infty} \leq \frac{\||A^{-1}|(|r| + \gamma_{n+1}(|A||\widehat{x}| + |b|))\|_\infty}{\|\widehat{x}\|_\infty}, \tag{3.5}$$

where

$$r = A\widehat{x} - b, \qquad \gamma_n = \frac{n\varepsilon_\mathrm{M}}{1 - n\varepsilon_\mathrm{M}},$$

is much more useful. To evaluate (3.5) we need to compute the remainder r and the combination $|A||\widehat{x}| + |b|$ (trivial), as well as $|A^{-1}||r|$ (non-trivial, but there is no need to compute A^{-1} explicitly; see Sect. 3.2.6).

In the second way of analyzing errors we think backwards. The smallest ω for which $|\delta A| \leq \omega|A|$ and $|\delta b| \leq \omega|b|$ exist such that $(A + \delta A)\widehat{x} = b + \delta b$, is called the *relative backward error*. In other words, ω is the smallest relative change in any element of A or b such that $\widehat{x}$ represents the exact solution of the perturbed problem. The *lower* limit of the relative backward error is

$$\omega = \max_i \left\{ \frac{|r_i|}{(A\,|\widehat{x}| + b)_i} \right\}. \tag{3.6}$$

If we encounter a division of the type $|r_i|/0$ in (3.6) we assign a value of zero to the fraction if $|r_i| = 0$ or ∞ otherwise. Such cases occur in problems with sparse matrices when for some part of the matrix $A_{ij}x_j = 0$ for $\forall j$ and the denominator in (3.6) becomes zero or small enough to cause an overflow when computing the fraction. Error estimates built into library routines avoid this problem in several ways; see Sect. 7.7 in [10] for details.

3.2.2 Gauss Elimination

The most broadly useful algorithm to solve systems of equations $Ax = b$ with square matrices A without particular structure is the Gauss elimination with partial pivoting (GEPP). The core of the algorithm is the decomposition of the row-permuted matrix PA to a lower-triangular matrix L (values of 1 on the diagonal and non-zero

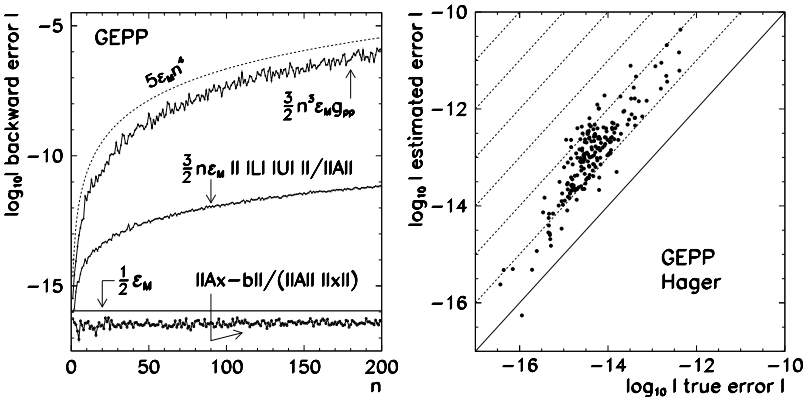


Fig. 3.2 Error estimates in solving the system of equations $Ax = b$ with random matrices by Gauss elimination with partial pivoting. [*Left*] Backward error versus matrix dimension. The pivot growth factor g_{pp} (3.8) typically does not increase faster than $\mathcal{O}(n)$. [*Right*] Hager's estimate compared to the true error. The error is over-estimated by about one order of magnitude; rare exceptions to this rough rule are mentioned in [1]

elements at most below the diagonal) and an upper-triangular matrix U (non-zero elements on the diagonal and above it) [2]:

$$PA = LU. \tag{3.7}$$

Partial pivoting enables the algorithm to proceed and prevents the absolute values of the elements of L to exceed unity. The condition number of A (Sect. 3.2.6) therefore remains comparable to those of L and U. In other words, by using partial pivoting we *almost always* [1] avoid numerical instabilities which might render the norm $\|PA - LU\|$ comparable to $\|A\|$. The measure for the stability of GEPP is the *pivot growth factor* $g_{\text{pp}} \equiv \|U\|_{\max}/\|A\|_{\max}$. For GEPP we have the backward error

$$\|\delta A\|_\infty \leq \frac{3}{2} g_{\text{pp}} n^3 \varepsilon_M \|A\|_\infty. \tag{3.8}$$

Figure 3.2 (left) shows the upper limits for the relative backward error $\frac{3}{2}g_{\text{pp}}n^3\varepsilon_{\text{M}}$ and $\frac{3}{2}n\varepsilon_{\text{M}}\| |L| \cdot |U| \|_\infty/\|A\|_\infty$ compared to the true error $\|Ax - b\|_\infty/(\|A\|_\infty\|x\|_\infty)$.

GEPP in double precision is implemented in the routines `DGESVX` (`ZGESVX` in complex) from LAPACK, `gsl_linalg_LU_decomp/_solve` from GSL or `LUdcmp` from NR3E (Table 3.1). In rare cases in which GEPP fails, we use elimination with complete pivoting (GECP) which is slower than GEPP: while GEPP requires $\frac{2}{3}n^3 + 2n^2 r$ operations (r is the number of right-hand sides in solving $AX = B$), finding the pivots in GECP implies additional $\mathcal{O}(n^3)$ operations ($\mathcal{O}(n^2)$ per step).

If A has special properties, the numerical cost may be reduced. For general or positive definite symmetric matrices the decomposition is accomplished by

one of the variants of the Cholesky method which requires $n^3/3 + \mathcal{O}(n^2 r)$ operations. We can use the `DSYSVX` and `DPOSVX` routines from LAPACK, the `gsl_linalg_cholesky_decomp/_solve` from GSL or `Cholesky` from NR3E.

3.2.3 Systems with Banded Matrices

Banded matrices have k_l sub-diagonals and k_u super-diagonals. For general banded matrices with a small bandwidth, $(n \gg k_\mathrm{l}, k_\mathrm{u})$ fast methods to solve $Ax = b$ exist, at a cost of $\mathcal{O}(nk_\mathrm{l}(k_\mathrm{l} + k_\mathrm{u}) + n(2k_\mathrm{l} + k_\mathrm{u}))$ (see Table 3.1). In double precision we recommend the use of the `DGBSVX` routine from LAPACK or the `Bandec` routine from NR3E.

Tridiagonal matrices are a special case of banded matrices. In physics, they can often be traced to the difference approximation of spatial or temporal differential operators on a mesh, like in (9.6) or (9.7). The three values at the neighboring mesh points appear in characteristic triplets in each matrix row. (If the difference involves more points, as in (9.12), we get a five- or more-diagonal matrix.) The algorithms for tridiagonal matrices (`DGTSV` and `DGTSVX` from LAPACK or `tridag` from NR3E) have the numerical cost of $\mathcal{O}(nr)$. Algorithms for symmetric tridiagonal positive-definite matrices (`DPTSV` and `DPTSVX` from LAPACK and `gsl_linalg_solve_symm_tridiag` from GSL) require $8nr$ operations. Such small costs motivate us to try to translate the physics problem such that tridiagonal or banded matrices appear instead of general ones.

3.2.4 Toeplitz Systems

Toeplitz matrices play a key role in problems of the theory of systems control, signal analysis, image processing, and in many related areas [19, 20]. Two typical cases of Toeplitz matrices are

$$T = \begin{pmatrix} t_0 & t_1 & t_2 & t_3 & \cdots \\ t_{-1} & t_0 & t_1 & t_2 & \cdots \\ t_{-2} & t_{-1} & t_0 & t_1 & \cdots \\ t_{-3} & t_{-2} & t_{-1} & t_0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \qquad T_k = \begin{pmatrix} 1 & t_1 & \cdots & t_{k-2} & t_{k-1} \\ t_1 & 1 & & & t_{k-2} \\ \vdots & & 1 & & \vdots \\ t_{k-2} & & & 1 & t_1 \\ t_{k-1} & t_{k-2} & \cdots & t_1 & 1 \end{pmatrix},$$

where T_k is positive definite. The common property of Toeplitz matrices is their insensitivity to a shift along a symmetry axis of the matrix. The physics background of this invariance are temporal or spatial shifts or displacements. Due to the characteristic appearance of these matrices we say that they possess *displacement structure* [21, 22]. The most relevant problems involving Toeplitz matrices are

the Yule–Walker equation $T_n x = -(t_1, t_2, \ldots, t_n)^{\mathrm{T}}$ in the theory of linear prediction (Sect. 6.7) and the solution of a general system $Tx = b$. Spatial derivatives in spectral methods for partial differential equations can also be represented by Toeplitz matrices (look carefully at the structure of the matrix (11.12) in Sect. 11.1).

Toeplitz systems can be solved by methods with a numerical cost of $\mathcal{O}(n^2)$. For general purposes, we recommend the use of the `toeplz` routine from NR3E. Super-fast $\mathcal{O}(n \log^2 n)$ Toeplitz methods exist [23, 24]. See also [25].

3.2.5 *Vandermonde Systems*

A Vandermonde matrix $V \in \mathbb{R}^{(n+1)\times(m+1)}$ for scalars $x_0, x_1, \ldots, x_n \in \mathbb{R}$ is defined by the elements $V_{ij} = (x_j)^i$ where $0 \le i \le m$ and $0 \le j \le n$. We encounter such matrices in two classes of problems: interpolation and quadrature. They are illustrated by the following two examples.

Example Assume we have a set of measurements $\{(x_j, y_j)\}_{j=0}^n$, the physics background of which is the function $y(x) = \sin(2\pi x) + 0.002\cos(100x)$ (a sine-wave with a tiny modulation or noise). We would like to interpolate the data by a polynomial $p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$ such that $p(x_j) = y_j$. We calculate the vector of coefficients $a = (a_0, a_1, \ldots, a_n)^{\mathrm{T}}$ by solving the system

$$V^{\mathrm{T}} a = y, \quad V = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_0^n & x_1^n & \cdots & x_n^n \end{pmatrix}, \; y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}, \tag{3.9}$$

where V is a square matrix. We sample the function y at $n+1$ randomly chosen points x_j from the interval $[0, 1]$ and solve the system (3.9) by Gauss elimination. The points are shown by the $\circ$ symbols and the corresponding polynomial by the (ran) curve in Fig. 3.3 (left). If we use equidistant points ($\bullet$) we get the (eq) curve, and with the Chebyshev nodes ($\diamond$) we get the (ceb) curve in the same figure.

On the non-symmetric interval $[0, 1]$ the interpolation works well close to the origin, but fails elsewhere. The reason for this is the imprecise calculation of the polynomial coefficients due to the large condition number $\kappa_2(V)$. By choosing the Chebyshev points x_j, $\kappa_2(V)$ can be greatly reduced, but interpolation is still poor. Its reliability can be enhanced by mapping the points $x_j \in [0, 1]$ to the symmetric interval $[-1, 1]$ by using $\xi_j = 2x_j - 1$ (Fig. 3.3 (right)). The randomly distributed points also work well, but this should not be a taken as a rule.

Example The second class of problems with Vandermonde matrices involves quadrature. If we know the sums $\{s_i\}_{i=0}^n$ of a quadrature formula $s_i = \sum_j w_j x_j^i$, the weights w_j are determined by solving $Vw = s$. Let us choose the points $\{x_j\}_{j=0}^n$

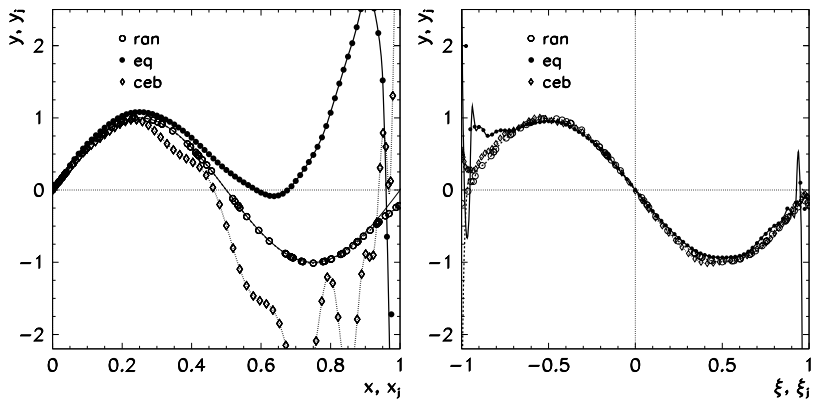


Fig. 3.3 The quality of the polynomial interpolation for different symmetries and choices of x_j. [*Left*] Interpolation at $\{(x_j, y(x_j))\}_{j=0}^{80}$ with randomly (ran) or uniformly (eq) distributed $x_j \in [0, 1]$. [*Right*] Interpolation at $\{(\xi_j = 2x_j - 1, y(x_j))\}_{j=0}^{80}$ with points ξ_j on a symmetric interval $[-1, 1]$. Also shown are the interpolation polynomials in the case when ξ_j are the roots of the Chebyshev polynomial of degree 80 (ceb)

from $[a, b]$ such that $a \le x_0 < x_1 < \cdots < x_{n-1} < x_n \le b$. We wish to determine such weights in the equation

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{j=0}^{n} w_j f(x_j),$$

that the formula will be exact for polynomials of degree $n-1$ at least, so

$$\begin{array}{rcccccccccc}
f(x)=1 & : & w_0 & + & w_1 & + \cdots + & w_n & = & \int_a^b \mathrm{d}x & = & s_0, \\
f(x)=x & : & w_0 x_0 & + & w_1 x_1 & + \cdots + & w_n x_n & = & \int_a^b x\,\mathrm{d}x & = & s_1, \\
\vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\
f(x)=x^{n-1} & : & w_0 x_0^{n-1} & + & w_1 x_1^{n-1} & + \cdots + & w_n x_n^{n-1} & = & \int_a^b x^{n-1}\,\mathrm{d}x & = & s_n.
\end{array}$$

The system is already in the form $Vw = s$ and can be solved, for example, by using the `vander` routine from the NR3E library.

Vandermonde systems are extremely ill-conditioned; the condition number $\kappa_2(V)$ grows exponentially with the matrix size. Even the lower limit of κ is known [26]. The algorithms to solve systems of the form $V^{\mathrm{T}}a = y$ and $Vw = s$ differ technically, but they have the same numerical cost. Systems of both types are best solved by Gauss elimination with complete pivoting which requires $\mathcal{O}(n^3)$ operations. Some Vandermonde systems can be solved by methods requiring only $\mathcal{O}(n^2)$ or even $\mathcal{O}(n \log^2 n)$ operations [27–33].

3.2.6 Condition Estimates for Matrix Inversion

The condition number of A corresponding to the sensitivity of $Ax = b$ to perturbations $(A + \delta A)\widehat{x} = b + \delta b$ depends on the type of these perturbations and on the matrices and vectors used in comparisons to δA and δb [10]. The error in the $\|\cdot\|_\infty$ norm, as in (3.4), corresponds to the condition number

$$\kappa_\infty(A) \equiv \left\|A^{-1}\right\|_\infty \|A\|_\infty.$$

For the error (3.6) with the $\|\cdot\|_\infty$ norm, the appropriate condition numbers are

$$\mathrm{cond}(A, x) = \left\|\left|A^{-1}\right||A||x|\right\|_\infty / \|x\|_\infty \quad \text{or} \quad \mathrm{cond}(A) = \left\|\left|A^{-1}\right||A|\right\|_\infty \le \kappa_\infty(A).$$

Regardless of the definition we must obtain $\|A^{-1}\|$ or $\||A^{-1}|\xi\|$ for $\xi \in \mathbb{R}^n$ in order to compute the condition number, and we would like to do this as precisely as possible without explicitly calculating the inverse A^{-1} which in general requires $\mathcal{O}(n^3)$ operations. Several *condition estimators* exist with a numerical cost of $\mathcal{O}(n^2)$. Among the most popular is the Hager's estimator [34] which is simple to code and suffices for almost all uses. Its more refined form [35, 36] is used in the LAPACK routines `xyyCON` and in the routines ending in the letter `X` from Table 3.1. This estimator returns the inverse value of the condition number `RCOND`. The comparison of the value of the estimator and the true error in solving a multitude of systems involving random matrices is shown in Fig. 3.2 (right). See also [37, 38] and Chap. 15 in [10].

Balancing If the problem $Ax = b$ is ill-conditioned, a more precise solution can be obtained by first solving the *scaled* or *balanced* problem $(D_1^{-1} A D_2) y = D_1^{-1} b$ where D_1 and D_2 are diagonal, and then calculate $x = D_2 y$. Balancing may do more harm than good (a nice example is given in [39]), and there is no general recipe. In addition, balancing may be used to precondition matrices in eigenvalue problems [40–42]. Practical routines can be found in LAPACK [3, 4].

3.2.7 Sparse Matrices

From a certain size upwards, in particular for sparse matrices, the methods of Sect. 3.2 become too time-demanding and memory-consuming. "Large" stands for $\approx n \times n = 10^6 \times 10^6$, a size one easily cooks up in multi-dimensional problems. (The horizon of "large" tends to move up by a factor 10 to 100 in n about every 10 years.) A matrix is considered to be sparse if it contains enough zeros that this fact can be exploited as the key circumstance allowing us to store the matrix and solve the system as economically as possible. To solve large sparse systems, two large classes of methods stand at our disposal: direct and iterative.

Direct methods for sparse matrices are much more demanding that those for dense matrices. One always tries to rearrange the sparse structure of the matrix to a

more condensed form or one with greater symmetry, and only then a step like a LU decomposition is performed. For a general introduction to direct methods for sparse systems, see Chap. 6 in [43]. A comprehensive list of program packages is given in [44, 45]. Benchmark comparisons of tools for large sparse *symmetric* systems have been compiled by [46, 47]. See also [48, 49].

The second choice are the iterative methods. Their classical representatives are the Gauss-Seidel and the SOR method. Among the more modern ones, we have multi-grid methods and conjugate-gradient methods. Typical users of iterative methods are engineering professionals, for which speed and orientation towards a functioning end-product are the chief guiding principles. The basic iterative methods are discussed in Sect. 10.2 and Sect. 10.8 (Table 10.1). Serious further reading can be found in Chaps. 8 and 9 of [43].

3.3 Linear Least-Square Problem and Orthogonalization

In the previous section we have discussed systems of the form $Ax = b$ where $A \in \mathbb{R}^{n\times n}$ and $x, b \in \mathbb{R}^n$. In this section, we are dealing with over-determined systems, in which there are more equations than unknowns, that is,

$$Ax = b, \quad A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^m,\ x \in \mathbb{R}^n,\ m \geq n.$$

Solving such systems can be understood as seeking a vector x that minimizes the remainder $r = Ax - b$ in some vector norm, e.g. $\|r\|_1$, $\|r\|_2$, or $\|r\|_\infty$. Due to its analytic properties (differentiability with respect to x) the $\|\cdot\|_2$ norm is most widely used. The problem of minimizing $\|Ax - b\|_2$ is also called the *linear least-squares (LS) problem*, as we attempt to minimize $[\sum_i r_i^2]^{1/2}$.

In the following we assume that the matrix $A \in \mathbb{R}^{m\times n}$ for $m \geq n$ has full rank, so it has n linearly independent columns. For rank-deficient or nearly rank-deficient matrices the matters become more complicated. For a matrix A with an exactly deficient rank the solution of the least-squares problems is not unique: if $\mathrm{rank}(A) = r < n$, there exists a $(n - r)$-dimensional set of vectors x that minimize $\|Ax - b\|_2$. From this set we have to choose the vector with the smallest norm which may be the only one to possess physical meaning (see Sect. 3.5 in [1], Sect. 5.5 in [2] and Sect. 3.3.3 in the following).

The basic method to solve the linear least-squares problem is to transform it to the *normal system*. We seek a vector x for which the gradient of $f(x) = \|Ax - b\|_2^2 = (Ax - b)^{\mathrm{T}}(Ax - b)$ is equal to zero, that is, a vector x such that $\partial f/\partial x_i = 0$ for $i = 1, 2, \ldots, n$. A short calculation brings us to the $n \times n$ normal system $A^{\mathrm{T}}Ax = A^{\mathrm{T}}b$ which has the solution

$$x_{\mathrm{LS}} = \left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}}b. \tag{3.10}$$

For matrices $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ with full rank the condition number for the least-squares problem is defined as the ratio of the largest and the smallest singular value, $\kappa_2(A) = \sigma_{\max}(A)/\sigma_{\min}(A)$ (see Sect. 3.3.2). The relative error of the vector $\widehat{x}_{\mathrm{LS}}$

computed by solving the normal system with respect to the true vector x_{LS} can be estimated as

$$\frac{\|\widehat{x}_{\mathrm{LS}} - x_{\mathrm{LS}}\|_2}{\|x_{\mathrm{LS}}\|_2} \approx \varepsilon_{\mathrm{M}} \kappa_2(A)^2. \tag{3.11}$$

Note that this estimate involves the *square* of the condition number. An additional concern is that solving the normal system may be unstable: this means that the computed $\widehat{x}_{\mathrm{LS}}$ does not necessarily minimize the "nearby" least-squares problem $\min_x \|(A + \delta A)x - (b + \delta b)\|_2$ where δA and δb are small perturbations. Still, the normal-system method is recommendable for well-conditioned matrices A as it comes at a numerical cost of just $(m + n/3)n^2$. In the NR3E library it is implemented in the `Fitlin` structure; see also Algorithm 5.3.1 in [2]. When higher precision and better stability are required, we resort to QR decomposition with column pivoting or singular-value decomposition (SVD).

3.3.1 The QR Decomposition

The QR decomposition is a method to orthogonalize a matrix $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ where A is decomposed to an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ $(Q^{\mathrm{T}}Q = QQ^{\mathrm{T}} = I)$ and an upper-triangular matrix $R \in \mathbb{R}^{m\times n}$ such that $A = QR$. The method with $Q \in \mathbb{R}^{m\times n}$ and $R \in \mathbb{R}^{n\times n}$ [1] is known as the "*thin*" variant of QR [2]. By using the QR decomposition, we hit two birds with one stone. If A has full rank n, its columns span a n-dimensional vector space $\mathcal{R}_n(A)$, and the first n columns of Q represent an orthonormal basis for this space; the decomposition and the computation of the basis vectors in double precision is accomplished by the `DGEQRF` and `DORGQR` routines from LAPACK. In addition, the QR decomposition also yields the solution of the least-squares problem. This becomes clear if the solution of the normal system (3.10) is rewritten as

$$x_{\mathrm{LS}} = \left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}}b = \left(R^{\mathrm{T}}Q^{\mathrm{T}}QR\right)^{-1}R^{\mathrm{T}}Q^{\mathrm{T}}b = R^{-1}\left(R^{\mathrm{T}}\right)^{-1}R^{\mathrm{T}}Q^{\mathrm{T}}b = R^{-1}Q^{\mathrm{T}}b.$$

When A is QR-decomposed, we obtain the solution x_{LS} by solving the upper-triangular system $Rx = Q^{\mathrm{T}}b$ by back-substitution. The whole process is implemented in the LAPACK's routine `DGELS` with improvements [50], the routines `gsl_linalg_QR_decomp` and `_lssolve` from GSL and, in the context of linear regression, `Fitlin` from NR3E (see Table 3.2). The lion's share of numerical work is hidden in the decomposition itself; the typical cost of these algorithms is $2n^2(m - n/3)$ to solve the least-squares problem, and an equal number of operations to construct the basis vectors for $\mathcal{R}_n(A)$ from Q.

There are other ways to orthogonalize a matrix $A \in \mathbb{R}^{m\times n}$. One of them is the classical Gram-Schmidt (GS) procedure, but it is unstable if the columns of A are only weakly dependent. In such cases, the GS procedure generates columns of Q which are not orthonormal, so that $\|I - Q^{\mathrm{T}}Q\| \gg \varepsilon$, and should not be used! The

Table 3.2 A collection of double-precision routines to solve the least-squares problem and its generalizations from the LAPACK, GSL, and NUMERICAL RECIPES (third edition for C++) libraries. In all cases we have $A \in \mathbb{R}^{m\times n}$ where $m \geq n$ (over-determined systems). The solution (3.10) by the normal-system methods is implemented in the `Fitlin` routine. The solution based on the QR decomposition is implemented in `DGELS` (LAPACK) and `gsl_linalg_QR_decomp/_solve` (GSL). The numerical cost of the LAPACK QR routines for complex variables is about four times higher than the cost of the corresponding routines for real variables. The methods exploiting singular-value decomposition (last line in the lower table) may also be applied to full-rank matrices. The numerical cost of these routines strongly depends on the implementation details. See also caption to Table 3.1

Full-rank matrices, $\text{rank}(A) = \min(m, n)$

Minimize	Constraint	Numerical cost	LAPACK	GSL	NR3E
$\min_x \|Ax - b\|_2$	/	$n^2(m + \frac{1}{3}n)$	/	/	`Fitlin`
$\min_x \|Ax - b\|_2$	/	$2n^2(m - \frac{1}{3}n)$	`DGELS`	`gsl_linalg_QR_decomp` `gsl_linalg_QR_lssolve`	/
$\min_x \|Ax - b\|_2$	$Bx = c$	$\leq 2n^2(2m + \frac{1}{3}n)$	`DGGLSE`	/	/
$\min_x \|y\|_2$	$Ax + By = c$	$2(\frac{2}{3}m^3 - \frac{1}{3}n^3) + 4nm^2$	`DGGGLM`	/	/
$\min_x \|B^{-1}(c - Ax)\|_2$	B square	$\frac{14}{3}n^3$	`DGGGLM`	/	/

Rank-deficient matrices, $\text{rank}(A) < \min(m, n)$

Minimize	Constraint	Numerical cost	LAPACK	GSL	NR3E
$\min_x \|Ax - b\|_2$	$\min_x \|x\|_2$	$\approx \mathcal{O}(mn^2) + \mathcal{O}(n^3)$ ([2], Sect. 5.4)	`DGELSY`	`gsl_linalg_QRPT_decomp` `gsl_linalg_QRPT_solve`	/
$\min_x \|Ax - b\|_2$	$\min_x \|x\|_2$	$\approx \mathcal{O}(mn^2) + \mathcal{O}(n^3)$ ([2], Sect. 5.5)	`DGELSS` `DGELSD`	`gsl_linalg_SV_decomp` `gsl_linalg_SV_solve`	`SVD, Fitsvd`

modified Gram-Schmidt procedure (mGS, Algorithm 5.2.5 in [2]) is much better: it can be used to generate columns of the matrix $Q_1 = [q_1, q_2, \ldots, q_n]$ for which

$$Q_1^{\mathrm{T}} Q_1 = I_n + E_{\mathrm{mGS}}, \qquad \|E_{\mathrm{mGS}}\|_2 \approx \varepsilon_{\mathrm{M}} \kappa_2(A). \tag{3.12}$$

If our requirements on orthonormality are very strict, we may use the mGS procedure only in cases when the condition number $\kappa_2(A)$ is small (columns of A well linearly independent). On the other hand, the mGS method is attractive due to its small cost of $\approx 2mn^2$.

Previously mentioned routines are based on Householder and Givens transformations which automatically ensure stability. In analogy to solving systems of equations $Ax = b$, the stability of least-squares methods is gauged by the sensitivity of the solution x of the problem $\min_x \|Ax - b\|_2$ with the remainder $r = b - Ax$ to small perturbations of A and b. Let $\widehat{x}$ minimize the perturbed problem, $\widehat{x} = \min_x \|(A + \delta A)x - (b + \delta b)\|_2$, and let $\widehat{r} = (b + \delta b) - (A + \delta A)\widehat{x}$ and

$$\varepsilon = \max\left\{ \frac{\|\delta A\|_2}{\|A\|_2}, \frac{\|\delta b\|_2}{\|b\|_2} \right\}.$$

(To get a feeling for what is going on, we may simply set $\varepsilon = \varepsilon_{\mathrm{M}}$.) Then the following holds [2]:

$$\frac{\|\widehat{x} - x\|_2}{\|x\|_2} < \varepsilon \left[\frac{2\|b\|_2}{\sqrt{\|b\|_2^2 - \rho_{\mathrm{LS}}^2}} \kappa_2(A) + \frac{\rho_{\mathrm{LS}}}{\sqrt{\|b\|_2^2 - \rho_{\mathrm{LS}}^2}} \kappa_2(A)^2 \right] + \mathcal{O}\left(\varepsilon^2\right),$$

where $\rho_{\mathrm{LS}} = \|Ax_{\mathrm{LS}} - b\|_2$ and we have assumed $\rho_{\mathrm{LS}} \neq \|b\|_2$. We see that the condition number $\kappa_2(A)$ appears linearly and quadratically in the relative error of the solution of the least-square problem. If the problem is well-conditioned and the remainders ρ_{LS} are small, the linear term dominates and the QR decomposition gives more precise results than the normal-system solution with the error (3.11) determined by the *square* of $\kappa_2(A)$. The QR decomposition based on Householder or Givens transformations also relieves us of the annoying presence of $\kappa_2(A)$ in the orthogonalization precision. Instead of (3.12)—see Fig. 3.4—we get

$$Q_1^{\mathrm{T}} Q_1 = I_n + E_{\mathrm{HQR}}, \quad \|E_{\mathrm{HQR}}\|_2 \approx \varepsilon_{\mathrm{M}}. \tag{3.13}$$

3.3.2 Singular Value Decomposition (SVD)

Orthogonalization of a matrix by singular value decomposition (SVD) [51] is one of the most remarkable methods of linear algebra with a limitless field of applications like data compression, linear least-square problems, and principal component analysis. If A is a real $m \times n$ matrix, there exist orthogonal matrices

$$U = [u_1, u_2, \ldots, u_m] \in \mathbb{R}^{m \times m} \quad \text{and} \quad V = [v_1, v_2, \ldots, v_n] \in \mathbb{R}^{n \times n},$$

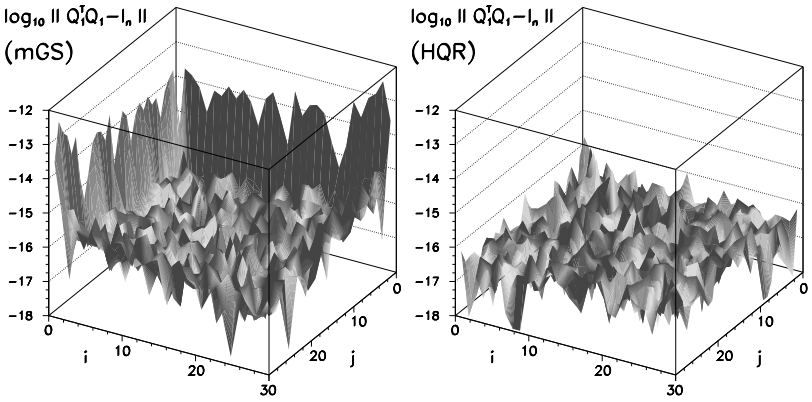


Fig. 3.4 The degree of orthonormality of the basis calculated by orthogonalizing a 30×30 matrix with the elements $A_{ij} = 1 + 0.01R$ where $R \in (-0.5, 0.5)$ is a uniformly distributed random number. The columns of A are weakly linearly dependent, so its condition number is large, $\kappa_2(A) = \sigma_{\max}(A)/\sigma_{\min}(A) \approx 6 \times 10^4$. [*Left*] Modified Gram–Schmidt procedure. Due to ill conditioning the error $\|Q_1^{\mathrm{T}} Q_1 - I_n\|$ increases (3.12). [*Right*] QR decomposition with Householder transformations. The error is essentially constrained by the machine precision (see (3.13))

such that A can be decomposed as

$$A = U \Sigma V^{\mathrm{T}}, \quad \Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}, \ p = \min\{m, n\},$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$. Like in the QR decomposition, SVD is also known in its "thin" variety in which $A \in \mathbb{R}^{m \times n}$ for $m \geq n$ can be decomposed as $A = U_1 \Sigma_1 V^{\mathrm{T}}$ where $U_1 \in \mathbb{R}^{m \times n}$ and $\Sigma_1 = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n) \in \mathbb{R}^{n \times n}$. The vectors u_k (or v_k) are known as left (or right) singular vectors of A, and the σ_k are known as singular values of A. We have $Av_i = \sigma_i u_i$ and $A^{\mathrm{T}} u_i = \sigma_i v_i$ for $1 \leq i \leq \min\{m, n\}$. The ratio of the largest and the smallest singular value determines the condition number of A, measured in the 2-norm,

$$\kappa_2(A) = \|A\|_2 \left\|A^{-1}\right\|_2 = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}. \tag{3.14}$$

The singular values are equal to the lengths of the semi-axes of the hyperellipsoid $\mathcal{E} = \{Ax : \|x\|_2 = 1\}$, and the condition number (3.14) is the ratio of the longest and shortest semi-axis (the elongation) of $\mathcal{E}$. We see this [52] if we represent an element of the unit sphere in $\mathbb{R}^n$ by the expansion $x = x_1 v_1 + \cdots + x_n v_n$ where $\sum_{i=1}^n x_i^2 = 1$, and observe its image $Ax = \sigma_1 x_1 u_1 + \cdots + \sigma_k x_k u_k$. The vector x is mapped to $y_1 u_1 + \cdots + y_k u_k$, where $y_i = \sigma_i x_i$, and

$$\frac{y_1^2}{\sigma_1^2} + \frac{y_2^2}{\sigma_2^2} + \cdots + \frac{y_k^2}{\sigma_k^2} = \sum_{i=1}^{k} x_i^2 \leq 1. \tag{3.15}$$

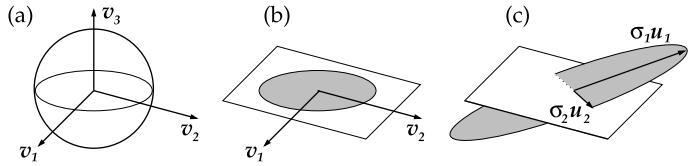


Fig. 3.5 Geometric interpretation of SVD [52] for a matrix $A \in \mathbb{R}^{m\times n}$ with $m = n = 3$ and $k = 2$. The mapping A collapses the $(n - k)$ dimensions of the space $\mathbb{R}^3$ (**a**). The unit sphere in the remaining k dimensions (**b**) is deformed into an ellipsoid by stretching and shrinking. This ellipsoid is finally embedded into $\mathbb{R}^m$ (**c**)

In other words, A maps the unit sphere in $\mathbb{R}^n$ into a k-dimensional hyper-ellipsoid with semi-axes of length σ_i in the directions u_i (Fig. 3.5). If A has full rank ($k = n$), the inequality in (3.15) becomes a strict equality and Ax resides on the surface of the hyper-ellipsoid.

The singular value decomposition unravels many important properties of a matrix related to its rank, condition, and norm. The singular values of A with rank $r < n$ are equal to zero from σ_r onwards, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_p = 0$, and the range of the mapping corresponding to this matrix is determined by the vectors $\{u_1, u_2, \ldots, u_r\}$. This brings us to the *SVD expansion*

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^{\mathrm{T}}, \tag{3.16}$$

which can be used to represent the matrix A by a limited set of singular values and vectors. The SVD expansion up to some rank $r < n$ is at the core of classical data reduction algorithms like, for instance, those used in image compression. An example is shown in Fig. 3.6; see also Problem 3.6.4.

LAPACK offers two double-precision routines for SVD, `DGESVD` and `DGESDD`. In the GSL library we have `gsl_linalg_SV_decomp`, while the NR3E offers `SVD`. Typical numerical costs are $\mathcal{O}(mn^2)$ for $m > n$ and $\mathcal{O}(m^2 n)$ for $m < n$. If A has full rank, SVD also yields the solution of the linear least-squares problem $\min_x \|Ax - b\|_2$: its solution is $x = V\Sigma^{-1}U^{\mathrm{T}}b$. The corresponding routines from the LAPACK, GSL, and NR3E libraries are listed in Table 3.2. The numerical cost of SVD is comparable to the cost of the QR decomposition.

Even though this is far too costly from the numerical standpoint, we may also use SVD to solve systems of equations $Ax = b$ with $n \times n$ matrices A, and study the sensitivity of the solutions to perturbations in A and b. If $A \in \mathbb{R}^{n\times n}$ is non-singular, $A = U\Sigma V^{\mathrm{T}}$ is its singular value decomposition, and $b \in \mathbb{R}^n$, we have

$$x = A^{-1}b = \left(U\Sigma V^{\mathrm{T}}\right)^{-1} b = \sum_{i=1}^{n} \frac{u_i^{\mathrm{T}} b}{\sigma_i} v_i .$$

Small perturbations in A and b may therefore cause large errors in x if σ_n and nearby singular values are small. With respect to conditioning of certain types of matrices,

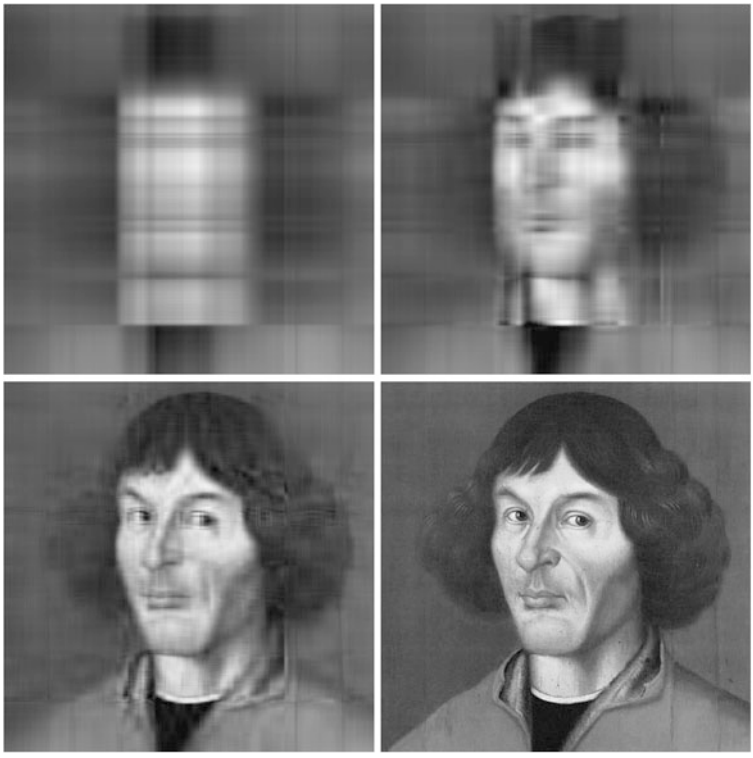


Fig. 3.6 Image compression by SVD. The original image is a 512×512 matrix of pixels with 256 gray levels. Shown are the approximations of rank 2 (99.6 % compression), rank 5 (99.0 %), rank 15 (97.1 %), and rank 100 (80.4 %). See Problem 3.6.4

singular values tend to have a larger "predictive power" than the usual eigenvalues λ_i (see Sect. 3.4). In general, we have

$$\sigma_{\min}(A) \leq \min_i |\lambda_i| \leq \max_i |\lambda_i| \leq \sigma_{\max}(A).$$

In particular with non-normal matrices, for which $A^{\mathrm{T}}A \neq AA^{\mathrm{T}}$ or $A^{\dagger}A \neq AA^{\dagger}$, it can happen that $\max_{i,j} |\lambda_i|/|\lambda_j| \ll \kappa_2(A)$.

A large gap between the largest and the smallest singular value of a matrix is usually an indication of its ill-conditioning. For some matrices, it may occur that from some index r upwards, the singular values are very small compared to the largest one (but not exactly zero), or that the spectrum exhibits two clearly separated sets of "large" and "small" singular values. Such a matrix has a *numerically defective rank*. Details can be found in [2].

The singular values of a matrix A are also intimately connected to its norms. If $A \in \mathbb{R}^{m\times n}$ has the decomposition $A = U\Sigma V^{\mathrm{T}}$, the following holds:

$$\|A\|_2 = \sigma_1, \qquad \|A\|_{\mathrm{F}}^2 = \sum_{i=1}^{p} \sigma_i^2 \quad \big(p = \min\{m, n\}\big),$$

$$\min_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sigma_n \quad (m \geq n).$$

Small perturbations δA of a matrix $A \in \mathbb{R}^{m \times n}$ $(m \geq n)$ induce changes in the singular values which are bounded by

$$\sum_{k=1}^{n} \big(\sigma_k(A + \delta A) - \sigma_k(A)\big)^2 \leq \|\delta A\|_{\mathrm{F}}^2.$$

3.3.3 The Minimal Solution of the Least-Squares Problem

If the matrix A is rank-deficient, the solution of the least-squares problem may become extremely sensitive to small perturbations $b \to b + \delta b$. The condition of such a problem can be improved by imposing an additional constraint on the solution, known as *regularization*. We require that the solution has a minimal norm. The vector x that solves the regularized problem $\min_x \|Ax - b\|_2$ is then known as the *minimal solution of the least-squares problem*. The minimal solution for a singular matrix A can be found by SVD. We decompose the matrix $A \in \mathbb{R}^{m \times n}$ with rank $r < n$ and $m \geq n$ as

$$A = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1 & V_2 \end{pmatrix}^{\mathrm{T}} = U_1 \Sigma_1 V_1^{\mathrm{T}},$$

where $\Sigma_1 \in \mathbb{R}^{r \times r}$ is non-singular, $U_1 \in \mathbb{R}^{m \times r}$, $U_2 \in \mathbb{R}^{m \times (n-r)}$, $V_1 \in \mathbb{R}^{n \times r}$, and $V_2 \in \mathbb{R}^{n \times (n-r)}$. All solutions x may then be written in the form

$$x = V_1 \Sigma_1^{-1} U_1^{\mathrm{T}} b + V_2 z, \tag{3.17}$$

where $z \in \mathbb{R}^{n-r}$ is an arbitrary vector. The solution x has a minimal norm $\|x\|_2$ precisely when $z = 0$. Its norm is bounded by $\|x\|_2 \leq \|b\|_2 / \sigma_{\min}(A)$. Hence the computation of the minimal solution is well-conditioned when the smallest positive singular value of A is not too small.

By using (3.17) the solution of the least-squares problem may be written in the unified form

$$x = A^+ b, \quad A^+ = V \Sigma^+ U^{\mathrm{T}}, \; \Sigma^+ = \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix}^+ = \begin{pmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{pmatrix},$$

where A^+ is the Moore–Penrose *generalized inverse* or *pseudo-inverse* of the matrix $A \in R^{m \times n}$ with $m \geq n$. If A is rank-deficient, the solution $x = A^+ b$ is unique and has the minimum norm. Details can be found in Sect. 3.5 of [1] and in Sect. 5.5 of [2]. The generalized inverse is put to practical use in Sect. 5.6.

3.4 Matrix Eigenvalue Problems

For a physicist, solving the linear eigenvalue problem

$$Ax = \lambda x, \quad x \neq 0, \tag{3.18}$$

especially in the context of real symmetric matrices, amounts to *diagonalizing a matrix*. In science and engineering, the eigenproblem (3.18) is ubiquitous: for example, solving systems of differential equations $\dot{x}(t) = Ax(t)$ or $\ddot{x}(t) = Ax(t)$ with suitable initial conditions can be translated to problems of the form (3.18) by the ansatz $x_i(t) = \exp(\lambda_i t)x_i(0)$. Equation (3.18) defines the *right eigenvectors* x and the corresponding eigenvalues. We may consider x as the direction which is insensitive to multiplication by A, and the corresponding eigenvalue λ is a representation of A in the subspace spanned by x. In this subspace, multiplying a vector by a matrix amounts to a rescaling along the eigenvector, thus

$$A^k x = \lambda^k x.$$

Similar reasoning applies to the *left eigenvectors* y, which are solutions of the problem

$$y^\dagger A = \lambda y^\dagger, \quad y \neq 0.$$

Eigenvalues of the matrix $A \in \mathbb{C}^{n\times n}$ are the roots of the characteristic polynomial $p(\lambda) = \det(\lambda I - A)$. In general, a real matrix may therefore possess complex eigenvalues. Due to potential numerical instabilities, individual λ_i should never be computed by seeking the roots of $p(\lambda) = 0$. Diagonalization algorithms convert the matrix A to a special form suitable for the computation of the eigenvalues and eigenvectors. The nature of this conversion depends on the symmetry properties of A.

We should not race ahead with diagonalization as some matrices are not diagonalizable at all. If $A \in \mathbb{C}^{n\times n}$ there exists a non-singular matrix X such that

$$X^{-1}AX = \operatorname{diag}(\lambda_1 I + N_1,\ \lambda_2 I + N_2,\ \ldots,\ \lambda_q I + N_q), \quad N_i \in \mathbb{C}^{n_i \times n_i}, \tag{3.19}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_q$ are distinct and all matrices N_i are strictly upper-triangular. The numbers n_i satisfy $n_1 + n_2 + \cdots + n_q = n$ and express the *algebraic multiplicity* of the individual λ_i. The *geometric multiplicity* of the eigenvalue λ_i is equal to the number of linearly independent eigenvectors belonging to λ_i. If the algebraic and the geometric multiplicities of λ_i are equal, the matrix A is diagonalizable or *non-defective*, and we can find a matrix $X \in \mathbb{C}^{n\times n}$ such that

$$X^{-1}AX = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n). \tag{3.20}$$

This can be done precisely when there exist scalars $\lambda_1, \lambda_2, \ldots, \lambda_n$ and linearly independent vectors $x_1, x_2, \ldots, x_n$ such that $Ax_i = \lambda_i x_i$ for $1 \leq i \leq n$. Then

$$\det(A) = \prod_{i=1}^{n} \lambda_i, \qquad \operatorname{tr}(A) = \sum_{i=1}^{n} A_{ii} = \sum_{i=1}^{n} \lambda_i.$$

The set of eigenvalues $\lambda(A) = \{\lambda_1, \lambda_2 \ldots, \lambda_n\}$ is called the *spectrum*. When some algebraic multiplicity λ_i exceeds the geometric multiplicity, the eigenvalue is said to be *defective*, and the matrix is also defective (or non-diagonalizable).

Example Check that the matrix $A = ((2,0,0),(0,2,0),(0,0,2))$ and the matrix $B = ((2,1,0),(0,2,1),(0,0,2))$ have the same characteristic polynomial $p(\lambda) = (\lambda - 2)^3$ which yields the eigenvalue $\lambda = 2$ of algebraic multiplicity 3. For the matrix A, the geometric multiplicity is also 3, as three independent eigenvectors, for example e_1, e_2, and e_3, may be assigned to this λ. On the other hand, only one independent eigenvector (e_1) can be found for B, so the geometric multiplicity of λ in this case is only 1. Hence B is deficient.

Normal matrices ($AA^{\mathrm{T}} = A^{\mathrm{T}}A$ or $AA^{\dagger} = A^{\dagger}A$) are diagonalizable (the opposite is not necessarily true), and tests of normality are frequently used as test of diagonalizability. This property can also be checked by using *minimal polynomials* [53]. Hermitian matrices ($A^{\dagger} = A$) and real symmetric matrices ($A^{\mathrm{T}} = A$) are normal, hence also diagonalizable, and have real eigenvalues.

3.4.1 Non-symmetric Problems

The basic tool for the diagonalization of dense non-symmetric matrices $A \in \mathbb{R}^{n\times n}$ or $A \in \mathbb{C}^{n\times n}$ is the iterative QR algorithm with implicit shifts [54, 55]. (The QR iteration is not the same as the QR decomposition discussed in Sect. 3.3.1. The QR decomposition is just its ingredient.) The QR algorithm is based on the conversion of A to the upper Hessenberg form and on the iterative computation of the real Schur form which ultimately yields the eigenvalues. The eigenvectors are finally computed by inverse iteration [2]. The QR algorithm for real non-symmetric matrices is implemented in the `DGEEVX` routine from LAPACK, in the `gsl_eigen_nonsymmv` routine from GSL, and in the `Unsymmeig` routine from NR3E. The corresponding LAPACK routine for complex matrices is `ZGEEVX`. The numerical cost of the QR algorithm is $\mathcal{O}(n^3)$. See Table 3.3.

The eigenvalues and eigenvectors of $A \in \mathbb{C}^{n\times n}$ are sensitive to small perturbations in the matrix. Let λ be a simple eigenvalue of A, and let x and y be its right and left eigenvectors normalized to unity, $\|x\|_2 = \|y\|_2 = 1$. Let $\lambda + \delta\lambda$ be the eigenvalue of the perturbed matrix $A + \delta A$. Then we have [1]

$$|\delta\lambda| \leq \frac{\|\delta A\|}{|y^{\dagger}x|} + \mathcal{O}\big(\|\delta A\|^2\big),$$

where $\kappa = 1/|y^{\dagger}x| = 1/\cos\theta(x,y)$ is the condition number for the eigenproblem with this eigenvalue. In general, therefore, perturbations $\|\delta A\|$ on the order of $\approx \varepsilon_{\mathrm{M}}$ cause errors on the order of $\approx \kappa\varepsilon_{\mathrm{M}}$ in simple eigenvalues. For multiple (degenerate) eigenvalues, the sensitivity to perturbations δA may be much more pronounced.

Table 3.3 A collection of double-precision routines to solve real eigenproblems from the LAPACK, GSL, and NUMERICAL RECIPES (third edition for C++) libraries. (The basic Jacobi method is described on p. 130.) The `DSYEVD`, `DSTEVD`, `DSYGVD`, and `DSBGVD` routines based on divide-and-conquer algorithms [59] and the `DSYEVR` routine based on relatively robust representations [60, 61] have the same asymptotic cost as the corresponding standard routines (last letter '`X`'), but their leading constant is much smaller, and they are therefore much faster. The complex routines are mentioned in the main text. See also caption to Table 3.1

Problem $Ax = \lambda x$ ($A \in \mathbb{R}^{n \times n}$)

Matrix A	Numerical cost	LAPACK	GSL	NR3E
General	$\mathcal{O}(n^3)$	`DGEESX, DGEEVX`	`gsl_eigen_nonsymmv`	`Unsymmeig`
Symmetric	$\mathcal{O}(n^3)$	`DSYEVX`	`gsl_eigen_symmv`	`Symmeig`
		`DSYEVD`		
		`DSYEVR`		
Symmetric banded	$\mathcal{O}(k_A n^2)$ for λ, $\mathcal{O}(n^3)$ for x	`DSBEVX`	/	/
		`DSBEVD`		
Symmetric tridiagonal	$\mathcal{O}(n^2)$ for λ, $\mathcal{O}(n^3)$ for x	`DSTEVX`	/	/
	$\mathcal{O}(n^2)$ for λ, $\mathcal{O}(n^3)$ for x	`DSTEVD`		
	$\mathcal{O}(n^2)$ for all λ and x	`DSTEGR/DSTEVR`		

Problem $Ax = \lambda Bx$ ($A, B \in \mathbb{R}^{n \times n}$, $\det(A - \lambda B) \not\equiv 0$)

Matrices A and B	Numerical cost	LAPACK	GSL	NR3E
General A, B	$\mathcal{O}(n^3)$	`DGGESX, DGGEVX`	`gsl_eigen_genv`	/
Symmetric A, B B pos. definite	$\mathcal{O}(n^3)$	`DSYGVX`	/	/
		`DSYGVD`		
Symmetric A, B A banded, B pos. definite	$\mathcal{O}(k_A n^2)$ for λ, $\mathcal{O}(n^3)$ for x	`DSBGVX`	/	/
		`DSBGVD`		

Namely, a large condition number $\kappa \gg 1$ implies that A with a simple eigenvalue λ is "close to" $A + \delta A$ which has a multiple eigenvalue, such that

$$\frac{\|\delta A\|_2}{\|A\|_2} \leq \frac{1}{\sqrt{\kappa^2 - 1}} \approx \frac{1}{\kappa}.$$

If A is non-normal, the value of κ *may be* very large, while for symmetric and Hermitian matrices, we have $|y^\dagger x| = 1$ and hence always $\kappa = 1$.

Example (Modified from [2]) The eigenvectors are particularly sensitive to perturbations in A if its simple eigenvalues lie close to each other or if there are multiple eigenvalues. Consider the matrices

$$A = \begin{pmatrix} 1.001 & 0.001 \\ 0.000 & 0.999 \end{pmatrix}, \qquad A + \delta A = \begin{pmatrix} 1.001 & 0.001 \\ 0.000 & 1.000 \end{pmatrix}.$$

The matrix A has eigenvalues 1.001 and 0.999. The eigenvector corresponding to the smaller eigenvalue is $(-0.44721, 0.89443)^{\mathrm{T}}$. If we slightly perturb A to $A + \delta A$ (10^{-3} change in lower right corner), the perturbed matrix has eigenvalues 1.001 and 1.000. Now, the eigenvector corresponding to the smaller of the eigenvalues is $(-0.70711, 0.70711)^{\mathrm{T}}$! For details, see [56, 57].

3.4.2 Symmetric Problems

The $Ax = \lambda x$ eigenproblems with real symmetric matrices ($A = A^{\mathrm{T}}$) possess numerous particular properties which allow for specially tailored, more efficient solution methods. The symmetry of A commonly originates in the natural symmetries of the underlying physics problem or its mathematical formulation. All eigenvalues of a symmetric matrix are real, and we can always find an orthogonal matrix Q such that

$$Q^{\mathrm{T}} A Q = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n), \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n,$$

and $\|A\|_2 = \max\{|\lambda_1|, |\lambda_n|\}$. Symmetric matrices are never defective, and their left eigenvectors are equal to their transposed right eigenvectors. Similarly, all eigenvalues of Hermitian matrices ($A^\dagger = A$) are real, and their left eigenvectors are the Hermitian conjugates of their right eigenvectors. Symmetric tridiagonal matrices—omnipresent in physics—with non-zero sub-diagonals can only have simple eigenvalues.

The oldest procedure to compute all eigenvalues and eigenvectors of dense symmetric matrices is the Jacobi method, in which a sequence of orthogonal transformations is used to gradually (to a required precision) extinguish all off-diagonal elements. The method has a cost of $\mathcal{O}(n^3)$ with a large leading constant, but it is extremely robust and may compute small eigenvalues to a better relative precision

than other, faster methods. Moreover, since the mentioned transformations are decoupled, it is well suited for parallelization. The basic version is implemented as `Jacobi` in the NR3E library and is recommended for matrices with sizes up to $n \approx 10$. A parallel implementation is given in [2].

By a sequence of Householder transformations, any symmetric matrix can be transformed to tridiagonal form from which eigenvalues and eigenvectors can be extracted. Fast methods therefore attempt to convert dense symmetric matrices to a tridiagonal form, which is then diagonalized. This conversion usually requires $\frac{4}{3}n^3 + \mathcal{O}(n^2)$ operations.

The basic method to compute all eigenvalues and (if needed) all eigenvectors of a symmetric tridiagonal matrix is the QR iteration which is a variant of the QR iteration for non-symmetric matrices. The computation of all eigenvalues requires $\approx 36n^2 + \mathcal{O}(n)$ operations, while the computation of all eigenvectors takes $\approx 9n^3 + \mathcal{O}(n^2)$ operations. The total cost of computing all eigenvalues of a dense symmetric matrix by tridiagonalization and QR iteration is $\frac{4}{3}n^3 + \mathcal{O}(n^2)$, and the total cost of computing all eigenvectors is $9n^3 + \mathcal{O}(n^2)$. Based on the anticipated rounding errors, QL iteration can be used instead of QR; consult [58] to see when this is appropriate.

To compute double-precision eigenvalues and eigenvectors of dense symmetric matrices by tridiagonalization and the QR algorithm, LAPACK offers the `DSYEVX` routine. The corresponding Hermitian routine is `ZHEEVX` (see Table 3.3 which also lists the numerical costs). The GSL library has `gsl_eigen_symmv` for real symmetric matrices and `gsl_eigen_hermv` for Hermitian matrices. The NR3E offers only a real implementation `Symmeig` but all $n \times n$ Hermitian eigenvalue problems can be transformed to $2n \times 2n$ real symmetric eigenvalue problem. Namely, the eigenproblem $Hz = \lambda z$ with a Hermitian matrix $H = A + \mathrm{i}\,B$ means $(A + \mathrm{i}\,B)(x + \mathrm{i}\,y) = \lambda(x + \mathrm{i}\,y)$ where A, B, x, and y are real, which can be written as

$$\begin{pmatrix} A & -B \\ B & A \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix}.$$

The matrix of this system is symmetric since $A^{\mathrm{T}} = A$ and $B^{\mathrm{T}} = -B$ if H is Hermitian ($H^{\dagger} = H$), and the eigenvalues and eigenvectors are duplicated.

A faster way to compute the eigenvalues and eigenvectors of dense real symmetric or Hermitian matrices leads through the "divide-and-conquer" algorithms [59]. In LAPACK they can be found in the `DSYEVD` routine (real symmetric) and the `ZHEEVD` (Hermitian). Methods based on *relatively robust representations*, RRR [60, 61] are even more efficient. LAPACK contains the `DSYEVR` routine (real symmetric) and the `ZHEEVR` routine (Hermitian).

For real symmetric tridiagonal matrices, LAPACK offers `DSTEVX` (QR iteration), `DSTEVD` ("divide-and-conquer") and `DSTEVR` (RRR). For symmetric banded matrices, there are `DSBEVX` (QR) and `DSBEVD` ("divide-and-conquer") for real matrices and the corresponding pair `ZHBEVX` and `ZHBEVD` for Hermitian matrices.

The sensitivity of eigenvalues and eigenvectors to small perturbations δA of the matrix A can be measured in analogy to the non-symmetric case. Let A be symmetric, with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ and eigenvectors $x_1, x_2, \ldots, x_n$, and

let δA be a symmetric perturbation matrix, so that the eigenvalues of $A + \delta A$ are $\widehat{\lambda}_1 \geq \widehat{\lambda}_2 \geq \cdots \geq \widehat{\lambda}_n$ and the corresponding eigenvectors are $\widehat{x}_1, \widehat{x}_2, \ldots, \widehat{x}_n$. If the perturbation is bounded by $\|\delta A\|_2 = \mathcal{O}(\varepsilon)\|A\|_2$, we have

$$|\widehat{\lambda}_i - \lambda_i| \leq \|\delta A\|_2 = \mathcal{O}(\varepsilon)\|A\|_2, \quad 1 \leq i \leq n. \tag{3.21}$$

If A is normal, $\|A\|_2 = \max_j |\lambda_j|$ also applies. The sensitivity to perturbations can also be measured by the angle θ_i between the eigenvector x_i of the unperturbed problem and the eigenvector $\widehat{x}_i$ of the perturbed problem. It can be shown that

$$\frac{1}{2}\sin 2\theta_i \leq \|\delta A\|_2 \Big[\min_{j \neq i} |\widehat{\lambda}_j - \widehat{\lambda}_i|\Big]^{-1}. \tag{3.22}$$

When the largest and the smallest eigenvalues differ by orders of magnitude, the bounds (3.21) and (3.22) tend to strongly over-estimate the errors. See [1] for stricter bounds.

Example We would like to compute the energy states of electrons in a two-dimensional periodic potential $V(x, y)$ with the period d in x- and y-directions in the presence of a constant magnetic field defined in terms of the vector potential $\boldsymbol{A} = B(0, -x, 0)^{\mathrm{T}}$. The corresponding Hamiltonian is

$$\hat{H} = \frac{1}{2m_{\mathrm{e}}}(\boldsymbol{p} - e\,\boldsymbol{A})^2 + V(x, y).$$

The ansatz for the wave-function of the electron is [62]

$$\psi(x, y) = \sum_{n,m \in \mathbb{Z}} a_{nm} \mathrm{e}^{-\mathrm{i}2\pi m\alpha x/d} \psi_0(x - nd, y - md),$$

where ψ_0 are the single-atom wave-functions and $\alpha = eBd^2/h$. The parameter α corresponds to the number of quanta of the magnetic flux per unit cell of the mesh. Because the Hamiltonian is periodic, the Bloch theorem [63] permits us to write the expansion coefficients a_{nm} in the form $a_{nm} = e^{\mathrm{i}m\phi} a_n$. The coupling between the neighboring mesh points is given by the overlap integrals

$$W_{nm} = \iint \psi_0(x - nd, y - md)\, \mathrm{e}^{\mathrm{i}2\pi m\alpha x/d}\, \hat{H}\Big|_{B=0} \psi_0(x, y)\, \mathrm{d}x\, \mathrm{d}y.$$

By using the ansatz ψ in the Schrödinger equation $\hat{H}\psi = E\psi$ and considering only the nearest-neighbor couplings ($W_{nm} = 0$ for $|n| > 1$ or $|m| > 1$) we obtain a tridiagonal system of equations for the expansion coefficients,

$$a_{n+1} + \lambda \cos(2\pi n\alpha - \phi) a_n + a_{n-1} = \varepsilon a_n, \quad 0 \leq n \leq q - 1. \tag{3.23}$$

We introduce the dimensionless energy $\varepsilon = E/W_{10}$ and the ratio of coupling strengths in x- and y-directions, $\lambda = 2W_{10}/W_{01}$. The solutions of (3.23) are qualitatively different for rational or irrational values of α. For rational values $\alpha = p/q$

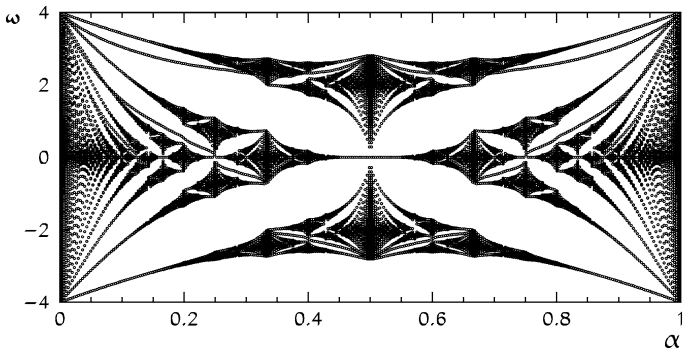


Fig. 3.7 Dimensionless eigenenergies ε of electrons in a two-dimensional periodic potential in the presence of a magnetic field B, as a function of $\alpha = eBd^2/h = p/q$, using Dirichlet boundary conditions for ψ and parameters $\lambda = 2$, $\phi = 0$, $q = 301$. In a crystal with $d \approx 0.2$ nm such patterns are not expected to appear until the field densities reach $B \approx 10^5$ T. But an almost identical image was seen in studies of microwave propagation through waveguides possessing periodic scattering centers [62]

the sequence of dimensionless energies $\{\varepsilon_n\}$ has a period q, while for irrational α the sequence $\{\varepsilon_n\}$ constitutes a fractal. In the special case $\lambda = 2$ it is known as Hofstadter's butterfly (see Fig. 3.7 and [64]).

3.4.3 Generalized Eigenvalue Problems

The textbook [1] describes the familiar physical example of masses connected by coupled springs with damping that can be described by the system of equations

$$M\ddot{x}(t) = -B\dot{x}(t) - Kx(t), \quad x(0) = x_0,\ \dot{x}(0) = \dot{x}_0, \tag{3.24}$$

where M is the mass matrix, B is the damping coefficient matrix and K is the string constant matrix. By using the ansatz $x(t) = \exp(\lambda t)x(0)$ (3.24) becomes a quadratic eigenvalue problem $(\lambda^2 M + \lambda B + K)x = 0$, but by introducing the vector $z = (x, y)^{\mathrm{T}}$ where $y = \dot{x}$, it can be transformed to the usual linear problem

$$\begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}B \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix}.$$

If M is ill-conditioned, it is advisable to rewrite this as

$$\begin{pmatrix} -K & -B \\ 0 & I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} 0 & M \\ I & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix},$$

which has the form of a *generalized eigenvalue problem*

$$Az = \lambda Bz, \quad z \neq 0. \tag{3.25}$$

Such eigenproblems can be solved by using the QZ algorithm which is a generalization of the QR algorithm from Sect. 3.4.1. In the case of general real matrices A and B we can use the `DGGEVX` from LAPACK or the `gsl_eigen_genv` from GSL. For general Hermitian matrices A and B where B is positive definite, we may use `ZGGEVX` from LAPACK or `gsl_eigen_genhermv` from GSL.

To solve (3.25) with symmetric A and B, where B is positive definite, LAPACK offers us the `DSYGVX` and `DSYGVD` routines. For banded A and positive definite B, we may take `DSBGVX` and `DSBGVD`. The corresponding Hermitian quartet of routines is `ZHEGVX`, `ZHEGVD`, `ZHBGVX`, and `ZHBGVD`. The numerical cost of these algorithms is $\mathcal{O}(n^3)$ (see Table 3.3). Implementations in the LAPACK library also allow us to solve related problems of the form $ABx = \lambda x$ and $BAx = \lambda x$. Further reading can be found in [1] (Sect. 4.5) and [2] (Sects. 7.7 and 8.7).

Example Generalized eigenproblems are commonly found in looking for stationary values of quadratic forms $x^{\mathrm{T}}Ax$ with symmetric matrices $A \in \mathbb{R}^{n\times n}$ and constraints

$$c^{\mathrm{T}}x = 0, \quad \|x\|_2 = \|c\|_2 = 1,\ x, c \in \mathbb{R}^n. \tag{3.26}$$

We use the function $\phi(x;\lambda,\mu) = x^{\mathrm{T}}Ax - \lambda(x^{\mathrm{T}}x - 1) + 2\mu\, x^{\mathrm{T}}c$ where λ and μ are the Lagrange multipliers [65]. When ϕ is differentiated with respect to x, we obtain

$$Ax - \lambda x + \mu c = 0. \tag{3.27}$$

This is an inhomogeneous eigenproblem [66] but it can be transformed to the homogeneous one. We determine μ by multiplying (3.27) from the left by c^{T}, and by using (3.26), we get $\mu = -c^{\mathrm{T}}Ax$. Hence

$$PAx = \lambda x, \quad P = I - cc^{\mathrm{T}},\ P^2 = I.$$

For eigenvalues of square matrices A and P we have $\lambda(PA) = \lambda(P^2A) = \lambda(PAP)$. It follows that by solving $PAPz = \lambda z$, where $x = Pz$, we simultaneously solve the problem (3.27). In general, the matrix PA is non-symmetric, but PAP is symmetric, and to solve $PAPz = \lambda z$, algorithms for symmetric eigenvalue problems can be used.

3.4.4 Converting a Matrix to Its Jordan Form

Matrix diagonalization finds one of the most common uses in solving systems of ordinary differential equations with constant coefficients

$$\frac{\mathrm{d}u}{\mathrm{d}t} = Au, \quad A \in \mathbb{C}^{n\times n},\ u \in \mathbb{C}^n. \tag{3.28}$$

By using a linear transformation $u = Xv$ the system can be rewritten as

$$X\frac{\mathrm{d}v}{\mathrm{d}t} = AXv \quad \text{or} \quad \frac{\mathrm{d}v}{\mathrm{d}t} = X^{-1}AXv = Jv. \tag{3.29}$$

If A is diagonalizable, i.e. if it can be decomposed as (3.20), the system (3.29) decouples to $\mathrm{d}v_i/\mathrm{d}t = \lambda_i v_i$. The general solution of this system is

$$v_i(t) = v_i(0)\,\mathrm{e}^{\lambda_i t} \quad \Longrightarrow \quad u(t) = \sum\nolimits_i v_i(0) x_i\,\mathrm{e}^{\lambda_i t}.$$

If A is defective, a complete decoupling by such a linear transformation cannot be achieved. But we can still simplify (3.28) significantly if the Jordan decomposition of A is known. The Jordan decomposition is a special form of (3.19),

$$X^{-1}AX = \operatorname{diag}\big(J_{n_1}(\lambda_1), J_{n_2}(\lambda_2), \ldots, J_{n_r}(\lambda_r)\big) = J,$$

where

$$J_{n_i}(\lambda_i) = \begin{pmatrix} \lambda_i & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & \lambda_i \end{pmatrix}^{n_i \times n_i}, \quad \sum_{i=1}^{r} n_i = n.$$

The matrix J is called the *Jordan canonical form*, and each sub-matrix $J_{n_i}(\lambda_i)$ is its Jordan block with the eigenvalue λ_i, where λ_i are not necessarily distinct. The number of blocks corresponding to an eigenvalue is equal to its geometric multiplicity, whereas their cumulative dimension is equal to its algebraic multiplicity. The decomposition is unique in the sense that only a permutation of the blocks J_{n_i} along the main diagonal of J is permissible. The general solution of (3.28) is

$$u(t) = X \exp(tJ) X^{-1} u(0).$$

The matrix $\exp(tJ)$ is block-diagonal, $\exp(tJ) = \operatorname{diag}(\exp(tJ_{n_i}(\lambda_i)))_{i=1}^{r}$, where

$$\exp\big(tJ_{n_i}(\lambda_i)\big) = \mathrm{e}^{\lambda_i t} \begin{pmatrix} 1 & t/1! & t^2/2! & \cdots & t^{n_i-1}/(n_i-1)! \\ 0 & 1 & t/1! & \cdots & t^{n_i-2}/(n_i-2)! \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & t/1! \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Only equations belonging to the same block remain coupled, and the eigenvalue multiplicities appear naturally. This is the main charm of the Jordan decomposition. However, in the real world—in floating-point arithmetic—the definition of the decomposition itself needs great care: we expect multiple eigenvalues λ_i on the main diagonal of a block, exact values of 1 on its first super-diagonal, and pure zeros elsewhere! Moreover, the numerical computation of the Jordan form is extremely ill-conditioned [67, 68]. The standard algorithm is described in [69]. Further reading on the computation of matrix exponents $\exp(A)$ and other functions of matrices $f(A)$ can be found in [70–72], [2] (Chap. 11), and [73].

3.4.5 Eigenvalue Problems for Sparse Matrices

Just like in solving systems of linear equations $Ax = b$ (Sect. 3.2.7), direct methods to compute eigenvalues and eigenvectors become too time-demanding and memory-consuming when matrices become too large. Again, iterative methods offer a way out. This remains true in the case of large sparse matrices unless their special structure is harnessed properly. Virtually all modern approaches to eigenproblems with large matrices are based on the methods of Krilov subspaces. A discussion of these methods is beyond the scope of this book. Initial reading is offered by [1]; the next steps might lead you to [74] or [43].

3.5 Random Matrices ⋆

Random matrices are matrices with elements that are chosen randomly. They find their use in hypothesis confirmation in theoretical physics, as in the theory of quantum chaos [75], information theory [76], finance [77], in numerical algorithms [78] and elsewhere [79]. The theory focuses on specific classes of such matrices [80]. For a physicist, Gaussian and cyclic (orthogonal or unitary) ensembles of random matrices are the most relevant. In this section we show how such matrices are generated and discuss their fundamental properties.

3.5.1 General Random Matrices

In certain applications we encounter square matrices with completely independent or weakly correlated random elements. Let the matrix M_n of dimension n have real or complex elements which are independent and identically distributed random numbers. Assume that the distribution of these numbers has zero mean, unit variance, and that all its higher moments are bounded. In addition, let $\{\lambda_i\}_{i=1}^n$ be the spectrum (set of eigenvalues) of M_n and

$$p_{M_n}(z) = \frac{1}{n}\sum_{i=1}^{n} \delta(z - \lambda_i), \quad z \in \mathbb{C},$$

the probability density of eigenvalues in the complex plane. Then $p_{M_n}(z)$, in the sense of the probability that a chosen eigenvalue is found near some specific point, converges to the uniform density on the unit circle when n is increased [81],

$$\lim_{n\to\infty} p_{M_n}(z) = \begin{cases} \pi^{-1}; & |z| \le 1, \\ 0; & \text{otherwise}, \end{cases} \tag{3.30}$$

which is known as the *circular law* or Girko's law [82]. (There is convincing numerical evidence that the circular law also applies to other classes of random matrices,

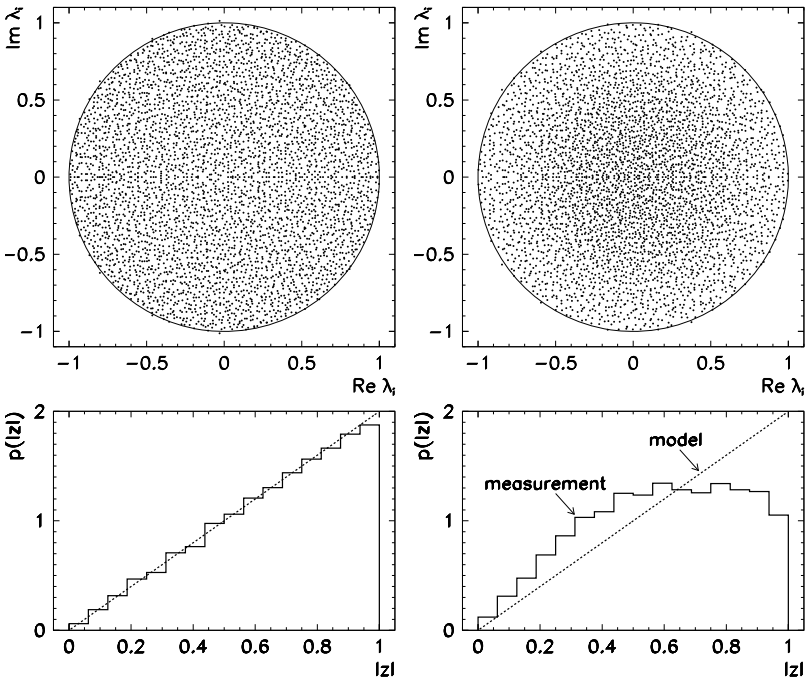


Fig. 3.8 Distribution of eigenvalues λ_i in the complex plane and the probability density p of the distribution of $|\lambda_i|$ for real matrices M_n of dimension $n = 4000$. [*Left*] Completely independent matrix elements $(M_n)_{ij}$. The eigenvalues are uniformly distributed within the unit circle. [*Right*] If the elements $(M_n)_{ij}$ are weakly correlated, the density of eigenvalues increases close to the origin and decreases near the unit circle compared to the uniform density. In both cases $\langle (M_n)_{ij} \rangle = 0$ and $\langle (M_n)_{ij}^2 \rangle = 1$

for instance, to Markov matrices [83].) The validity of the law (3.30) is illustrated in Fig. 3.8.

Marčenko–Pastur Theorem Stating this important theorem calls for some preliminaries. Assume that the probability density f of a random variable is known. The Stieltjes transformation [84] of f is

$$S_f(z) = \int_{-\infty}^{\infty} \frac{f(x)}{x - z} \, \mathrm{d}x, \quad \operatorname{Im} z \neq 0,$$

while its inverse is

$$f(x) = \frac{1}{\pi} \lim_{\xi \to 0} \operatorname{Im} S_f(x + \mathrm{i}\xi). \tag{3.31}$$

Let us define a $n \times n$ Hermitian matrix of the form

$$B_n = A_n + \frac{1}{n} X^\dagger T X,$$

where A_n is a $n \times n$ Hermitian matrix. When n is increased, let the distribution of its eigenvalues p_{A_n} *almost surely* converge to the distribution p_A with the corresponding Stieltjes transformation S_A. ("Almost sure convergence" means that the probability for the convergence of a sequence is equal to one [85].) Let X be a $m \times n$ complex matrix with independent and identically distributed elements X_{ij}, with the average $\langle X_{ij} \rangle = 0$ and variance $\langle |X_{ij}|^2 \rangle = 1$. The diagonal matrix $T = \mathrm{diag}(\tau_1, \tau_2, \ldots, \tau_m)$ should contain elements for which almost surely a limit probability density $H(\tau)$ exists when $m \to \infty$.

With the assumptions listed above, the Marčenko–Pastur theorem [86] tells us that the probability density p_{B_n}, when n is increased, almost surely converges to the limit distribution p_B, and the Stieltjes transformation S_B of this distribution satisfies the self-consistency relation

$$S_B(z) = S_A\left(z - c\int_{-\infty}^{\infty} \frac{\tau H(\tau)}{1 + \tau S_B(z)}\,\mathrm{d}\tau\right), \quad c = \lim_{n\to\infty} \frac{m(n)}{n} > 0. \tag{3.32}$$

Symmetric real or Hermitian complex matrices defined in terms of products $X^\dagger X$ are frequently seen in solutions of over-determined systems, in work with covariance matrices, or in applications of the singular value decomposition. In certain cases (see [87]) we may assume that the elements X_{ij} of X are independent and identically distributed with zero mean and unit variance. Then the Marčenko–Pastur theorem applies with $A_n = 0$, $S_A(z) = \int \delta(x)(x - z)^{-1}\,\mathrm{d}x = -z^{-1}$, $T = 1_m$, $H(\tau) = \delta(\tau - 1)$, and $B_n = \frac{1}{n} X^\dagger X$. When these quantities are inserted in (3.32), we obtain a quadratic equation $zS_B(z)^2 + S_B(z)(z - c + 1) + 1 = 0$. We solve it for S_B, and use the inverse Stieltjes transformation (3.31) to calculate

$$p_B(x) = \max\{0, 1 - c\}\delta(x) + \frac{\sqrt{(x - b_-)(b_+ - x)}}{2\pi x} I_{[b_-, b_+]}(x),$$

where $b_\pm = (1 \pm \sqrt{c})^2$, and $I_{[a,b]}(x) = 1$ for $x \in [a, b]$ or 0 otherwise. If X is a $n \times n$ (square) matrix, then $c = 1$, and the formula above simplifies to

$$p_B(x) = \frac{1}{2\pi x}\sqrt{x(4 - x)}\, I_{[0,2]}(x).$$

Marčenko–Pastur theorem assists us in many ways. In singular value decomposition of X, we find unitary matrices U and V such that $X = U\Sigma V^\dagger$ and $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_m)$, where $\{\sigma_i \geq 0\}_{i=1}^m$ is the set of singular values of X. From the decomposition formula we see that $X^\dagger X = V\Sigma^2 V^\dagger$, so the eigenvalues of $X^\dagger X$ are simply the squares of the singular values of X (see also Appendix A.6). One of the consequences is the *quarter-circular law*: it states that the probability density of the distribution of singular values of $\frac{1}{\sqrt{n}} X$ converges, in the sense of probability, to

$$p_\sigma(x) = \frac{2}{\pi}\sqrt{1 - \left(\frac{x}{2}\right)^2}\, I_{[0,2]}(x), \tag{3.33}$$

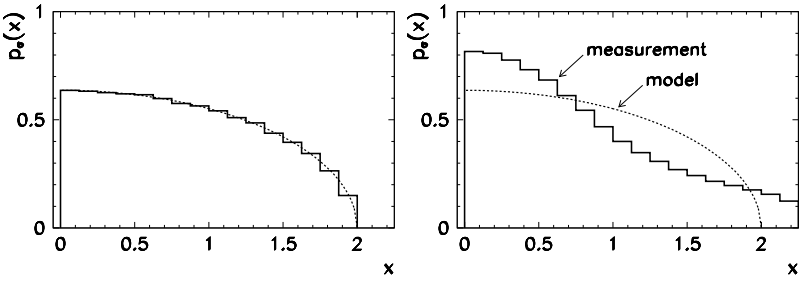


Fig. 3.9 Singular values of real matrices $\frac{1}{\sqrt{n}}X$ of dimension $n = 4000$ with uniformly distributed elements X_{ij} having zero mean ($\langle X_{ij}\rangle = 0$) and unit variance ($\langle X_{ij}^2\rangle = 1$). [*Left*] Distribution for independent elements X_{ij}. [*Right*] Distribution for weakly row-correlated X_{ij}. Due to the redundancy of the data the concentration of singular values is larger near the origin and some of them may actually exceed the value of 2

when n is increased. In plain words, the singular values of the random matrix $\frac{1}{\sqrt{n}}X$ do not exceed the value of 2. Equation (3.33) is illustrated by Fig. 3.9.

3.5.2 Gaussian Orthogonal or Unitary Ensemble

An ensemble of matrices is defined by the set of its elements (matrices) and by their distribution in the set. In the following, we are dealing with $n \times n$ matrices. The *Gaussian orthogonal ensemble* (GOE) consists of real symmetric matrices H with $n(n+1)/2$ free real parameters. The matrices from this set are distributed according to the probability density p in the space of matrices which is invariant to orthogonal transformations O in the sense

$$p(H) = p\big(O^{\mathrm{T}} H O\big).$$

The *Gaussian unitary ensemble* (GUE) consists of complex Hermitian matrices with n^2 free real parameters and a probability density p which is invariant with respect to unitary transformations U:

$$p(H) = p\big(U^{\dagger} H U\big).$$

The probability density p of the matrix H should be understood as the probability density of its independent elements. We can use the invariance condition to prove that the most general form of the density is $p(H) = \exp(a\,\mathrm{tr}(H^2) + b\,\mathrm{tr}(H) + c)$ where a and b are adjustable parameters and c is a normalization constant [75]. The ratio $b/(2a)$ represents the center of mass of the eigenvalues of H, which we are allowed to shift arbitrarily, so we set $b = 0$. The probability density of matrices belonging to either of the two ensembles is then

$$p(H) = \exp\big(a\,\mathrm{tr}\big(H^2\big) + c\big).$$

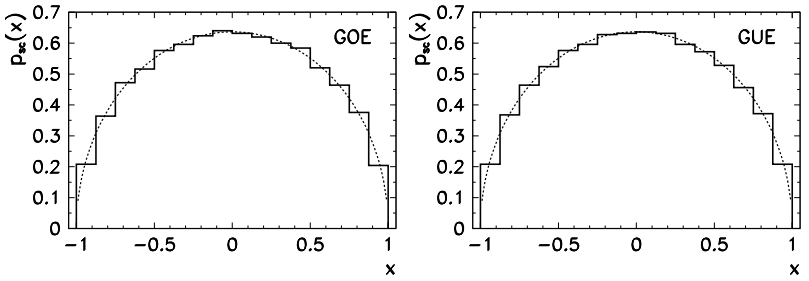


Fig. 3.10 Probability densities $p_{sc}(x)$ of the rescaled eigenvalues Λ' of one realization of a matrix of dimension $n = 2000$ [*Left*] from the GOE and [*Right*] from the GUE. The dotted lines represent the Wigner semi-circle (3.34)

For simplicity, let us choose $a = 1/2$. The elements of matrices H from the GOE or the GUE can be randomly generated by using the formulas

$$\begin{array}{lll} & \text{GOE} & \text{GUE} \\ i < j\text{:} & H_{ij} = 2^{-\frac{1}{2}}\, x_{ij}, & H_{ij} = 2^{-\frac{1}{2}}(x_{ij} + \mathrm{i}\, y_{ij}), \\ i = j\text{:} & H_{ii} = x_{ii}, & H_{ii} = x_{ii}, \end{array}$$

where x_{ij} and y_{ij} are distributed normally according to $N(0, 1)$.

In the limit of large n, the spectra of matrices from GOE and GUE possess certain well-defined statistical properties. Let H_n be a $n \times n$ matrix from GOE or GUE with the spectrum $\Lambda = \{\lambda_i\}_{i=1}^n$, and let us rescale the spectrum by using $\Lambda' = (a/n)^{1/2}\Lambda$ in the case of GOE, or by using $\Lambda' = (a/2n)^{1/2}\Lambda$ in the case of GUE. Then, if n is increased, the probability density of the distribution of the rescaled eigenvalues Λ' converges, in the sense of probability, to

$$p_{sc}(x) = \begin{cases} \frac{2}{\pi}\sqrt{1 - x^2}; & x \in [-1, 1], \\ 0; & \text{otherwise.} \end{cases} \tag{3.34}$$

Equation (3.34) establishes *Wigner's semi-circular law* [80], which tells us that we should expect the magnitudes of the rescaled eigenvalues not to exceed unity (see Fig. 3.10).

For random matrices, the fluctuations of the splittings between consecutive eigenvalues with respect to their local averages are very instructive. It turns out that the statistics of these fluctuations is equal to the statistics of the spectra of Hamiltonians corresponding to classically chaotic quantum systems [88]. The nature of the fluctuations becomes apparent when the spectrum is *unfolded*.

Unfolding of Matrix Spectra Assume that the spectrum Λ of a real symmetric or a Hermitian matrix is already known, and that the eigenvalues are sorted as $\{\lambda_i \in$

$\mathbb{R} : \lambda_{i+1} \geq \lambda_i\}_{i=1}^n$. We use the eigenvalues to construct the density

$$d(\lambda) = \sum_{i=1}^{n} \delta(\lambda - \lambda_i)$$

and compute the cumulative distribution

$$N(\lambda) = \int_{-\infty}^{\lambda} d\big(\lambda'\big)\,\mathrm{d}\lambda' = \#\{\lambda_i \leq \lambda\},$$

which measures the numbers of eigenvalues smaller than λ. This is a staircase-like, monotonously increasing function of λ. By using some method of local averaging, we split N into a smooth part $\overline{N}$ and an oscillatory part N_{osc},

$$N(\lambda) = \overline{N}(\lambda) + N_{\mathrm{osc}}(\lambda),$$

so that the local average of N_{osc} over a few (say, 10) neighboring eigenvalues is equal to zero or is negligible compared to the increase in N. We use the average increase of the number of eigenvalues $\overline{N}$ to define the average density of eigenvalues, $\overline{d}(\lambda) = \mathrm{d}\overline{N}(\lambda)/\mathrm{d}\lambda$. (In practice, this connection is often used in reverse to compute $\overline{N}$ if $\overline{d}$ is known.) For a large enough Δ we compute the average density:

$$\overline{d}(\lambda) = \frac{1}{2\Delta} \int_{-\Delta}^{\Delta} d(\lambda + t)\,\mathrm{d}t$$

and define

$$\overline{N}(\lambda) = \int_{-\infty}^{\lambda} \overline{d}\big(\lambda'\big)\,\mathrm{d}\lambda'.$$

The separation of the smooth and oscillatory parts is not unique, but in the limit of large matrices it does not influence significantly the conclusions about the statistics of the fluctuations. By using the function $\overline{N}$ the spectrum $\{\lambda_i\}_{i=1}^n$ is *unfolded* into

$$\widetilde{\lambda}_i = \overline{N}(\lambda_i), \quad i = 1, 2, \ldots, n,$$

so that the average distance between the neighboring eigenvalues $s_i = \widetilde{\lambda}_{i+1} - \widetilde{\lambda}_i$ locally (over a couple of consecutive eigenvalues) and globally equals 1. Finally, we are interested in the probability density of this average distance after the unfolding, which is

$$p_{\mathrm{s}}(x) = \frac{1}{n-1} \sum_{i=1}^{n-1} \delta(x - s_i).$$

For large matrices from GOE ($a = 1/(2n)$) or GUE ($a = 1/(4n)$) we may approximate $\overline{d}$ by the limit density (3.34), so $\overline{d}(x) = np_{\mathrm{sc}}(x)$ with $\Delta \to 0$, and we get

$$\overline{N}(\lambda) \sim n \int_{-1}^{\lambda} p_{\mathrm{sc}}\big(\lambda'\big)\,\mathrm{d}\lambda' = n\left(1 + \frac{1}{\pi}\big[\arcsin\lambda - \lambda\sqrt{1-\lambda^2}\big]\right), \quad \lambda \in [-1, 1].$$

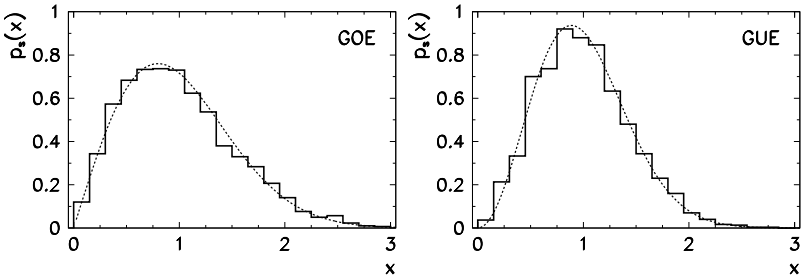


Fig. 3.11 Probability densities p_s of the distribution of the splittings between the subsequent eigenvalues of one realization of a matrix with $n = 2000$ after unfolding. [*Left*] Matrix from GOE. [*Right*] Matrix from GUE. The dotted curves correspond to the Wigner distributions (3.35) and (3.36)

With increasing n, the probability density p_s of the distances between neighboring unfolded eigenvalues converges, in the sense of probability, to the distribution

$$p_s^{\mathrm{GOE}}(x) \doteq \frac{\pi}{2} x \, \mathrm{e}^{-(\pi/4)x^2} \tag{3.35}$$

for matrices from GOE, or to the distribution

$$p_s^{\mathrm{GUE}}(x) \doteq \frac{32}{\pi^2} x^2 \, \mathrm{e}^{-(4/\pi)x^2} \tag{3.36}$$

for matrices from GUE. Formulas (3.35) and (3.36) are known as the Wigner distributions [80, 89]. Both distributions vanish at $x = 0$, which is a sign of the *avoidance of level crossing* or *level repulsion* (repulsion between neighboring eigenvalues), a mechanism which is statistically relatively stronger for matrices from GUE. Figure 3.11 shows the distributions p_s for large matrices from GOE and GUE, as well as the comparison to the forms (3.35) and (3.36). The connections between the Wigner distributions and the spectra of quantum-mechanical systems are discussed in Problem 3.6.7.

3.5.3 Cyclic Orthogonal and Unitary Ensemble

In certain physical problems, we need to comb the space in all orthogonal directions, and the procedure needs to be repeated for randomly chosen sets of directions. Such sets can be generated by a special class of random matrices discussed in this subsection. Again we restrict ourselves to $n \times n$ matrices.

The *cyclic orthogonal ensemble* (COE) consists of orthogonal matrices $O(n)$ while the *cyclic unitary ensemble* (CUE) consists of unitary matrices $U(n)$. Their measure is equal to the normalized Haar measure μ_H [80].

Haar Measure For a set of matrices Γ and the corresponding set of all its open subsets Σ, the Haar measure $\mu_{\mathrm{H}} : \Sigma \to \mathbb{R}$ is defined as

$$\begin{aligned} \mu_{\mathrm{H}}(\Gamma) &= 1, \\ \mu_{\mathrm{H}}(\gamma S) &= \mu_{\mathrm{H}}(S), \quad \forall \gamma \in \Gamma,\ \forall S \in \Sigma. \end{aligned}$$

If the parameterization of the matrices in Γ is known, we prefer to specify the differential of the measure at some point, $\gamma \in \Gamma$, hence $\mathrm{d}\mu_{\mathrm{H}}(\gamma) = p(\gamma)\,\mathrm{d}\gamma$, where $p(\gamma)$ is the probability density of the matrices over the set Γ. For example, the Haar measure for a set of one-dimensional unitary matrices is

$$U(1) = \left\{ \mathrm{e}^{\mathrm{i}\theta} :\ \theta \in [0, 2\pi) \right\}.$$

For 3×3 special orthogonal matrices, the Haar measure is

$$SO(3) = \left\{ R_z(\phi_2) R_x(\theta) R_z(\phi_1) :\ \theta \in [0, \pi),\ \phi_1, \phi_2 \in [0, 2\pi) \right\},$$

where $R_o(\alpha) \in \mathbb{R}^{3x3}$ is the rotation matrix for a rotation around the axis o by the angle α. Both sets are uniquely generated, so the differential of the Haar measure in the case of $U(1)$ matrices is equal to

$$\mathrm{d}\mu_{\mathrm{H}}(\gamma) = \frac{1}{2\pi}\,\mathrm{d}\theta, \quad \gamma \in U(1),$$

while for the $SO(3)$ matrices it is

$$\mathrm{d}\mu_{\mathrm{H}}(\gamma) = \frac{1}{8\pi^2} \sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi_1\,\mathrm{d}\phi_2, \quad \gamma \in SO(3).$$

Generating Matrices from COE There are many methods to generate a random orthogonal matrix from COE that preserves the Haar measure. The simplest way is to form a symmetric matrix A from GOE and diagonalize it,

$$A = O D O^{\mathrm{T}}, \quad O = [O_{ij}]_{i,j=1}^{n}.$$

By doing this, we obtain a random orthogonal matrix $O \in O(n)$, plus the eigenvalues of A contained in D. Assume that the diagonalization has set the leading coefficient of each column of O to a positive value, like it is done by LAPACK routines [3, 4]. This violates the condition that the matrices are chosen randomly with respect to the Haar measure. In this case, we randomly choose the signs $s_i \in \{-1, 1\}$ of the columns of O with equal probabilities $\mathcal{P}(s = \pm 1) = 1/2$, and finally generate the desired random orthogonal matrix from COE,

$$O_{\mathrm{rnd}} = [s_j O_{ij}]_{i,j=1}^{n},$$

which is chosen randomly with respect to the Haar measure.

Generating Matrices from CUE We proceed analogously in generating random unitary matrices from CUE. We generate a Hermitian matrix A from GUE and perform the spectral decomposition

$$A = U D U^{\dagger}, \quad U = [U_{ij}]_{i,j=1}^{n}.$$

By decomposing A we obtain a random unitary matrix $U \in U(n)$ and the diagonal matrix D containing the eigenvalues of A. Assume that we have used a diagonalization routine that sets the phase of the leading coefficient of individual eigenvectors to zero. To compensate for this regularity, we randomly pick the phases $\{\phi_i\}_{i=1}^{n}$ that are uniformly distributed on $[0, 2\pi)$, and let them modify the phases of the columns of U by using

$$U_{\mathrm{rnd}} = \left[\mathrm{e}^{\mathrm{i}\phi_j} U_{ij}\right]_{i,j=1}^{n}.$$

In this manner, we have generated a random unitary matrix from CUE which is chosen randomly with respect to the Haar measure.

When used with standard implementations of diagonalization algorithms, both methods mentioned here typically require $\mathcal{O}(n^3)$ operations. In spite of their simplicity, these procedures to generate random matrices from COE and CUE are just by a constant factor slower than the optimal methods based on Householder reflections (for GOE, [90]) or QR decomposition (for GUE, [91]). When a random orthogonal or unitary matrix X is not needed explicitly (for example, if already the products AX or XA are sufficient), the generation requires only $\mathcal{O}(n^2)$ operations.

Eigenvalues of orthogonal and unitary matrices reside on the unit circle. It should therefore not surprise us that the eigenvalues of matrices from COE and CUE in the limit of large dimensions are uniformly distributed on the unit circle. Interesting mathematical details can be found in [75] and [80].

3.6 Problems

3.6.1 Percolation in a Random-Lattice Model

Percolation is the passage of substances from one region to another in the process of filtering. In this Problem it is discussed in the context of random lattices which appear frequently in the physical descriptions of gels, polymers, glasses, and other disordered matter [92]. Mathematically, random lattices are graphs [93]. Here, we are dealing only with *discrete random lattices* in fixed geometries and possessing a certain degree of statistical disorder in their connectedness. One should distinguish the *lattice of connections*, where connections are randomly chosen, from the *lattice of sites*, in which the occupation of sites is random [94]. We shall study the electric conductivity of both types of lattice.

A lattice made of conductive material is placed between electrodes to which a constant voltage is applied. We randomly drop a fraction p of the connections from

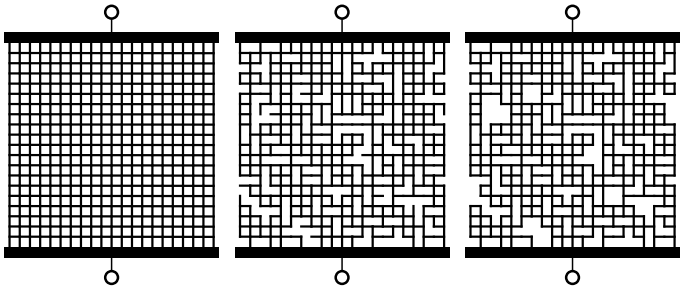


Fig. 3.12 Cartesian lattices of connections between the electrodes with an applied constant voltage. In the full lattice with 20×20 sites (*left*) we randomly drop $p = 20$ % of connections (*center*) and finally remove the unconnected ends (*right*)

the lattice, and remove the loose ends which cannot conduct. The lattice used in the calculations is shown in Fig. 3.12 (right). We would like to compute the current I flowing through the lattice, and determine how I changes—on the average—with p.

The lattice is in fact a circuit of N resistors $\{R_i\}_{i=1}^N$ and voltage sources $\{U_j\}_{j=1}^N$ connected at M nodes. Each connection can be oriented (positive direction from its beginning to its end). A voltage source with subscript j represents a point at the end of a free connection with a known electric potential U_j. Through a resistor R_i, a current I_i is flowing, and the sums of the incoming and outgoing currents should be equal at each node, which can be written as the matrix equation $AI = 0$ for the vector of currents $I = (I_i)_{i=1}^N$. The matrix $A \in \mathbb{R}^{M \times N}$ with the elements $A_{ij} \in \{-1, 1\}$ contains all information on the connectedness of the circuit (it is the *node-edge adjacency matrix* in graph theory).

For the ith connection between the potential e at the beginning of the connection and e' at its end, Ohm's law applies: $e + R_i I_i = e'$. At the edges of the lattice, the potential is given by the external voltage source and then e or e' are kept in the vector U; inside the lattice, e and e' are collected in the vector E. For all connections, Ohm's law can be written in the matrix form $RI + A^{\mathrm{T}} E = U$ where $R = \mathrm{diag}(R_i)_{i=1}^N$ is the resistance matrix, so all equations can be merged into one:

$$\begin{pmatrix} R & A^{\mathrm{T}} \\ A & 0 \end{pmatrix} \begin{pmatrix} I \\ E \end{pmatrix} = \begin{pmatrix} U \\ 0 \end{pmatrix}.$$

⊙ Calculate the average current $\langle I \rangle$ through 10×10, 20×20, and 50×50 lattices as a function of the fraction p of dropped connections. All connections have equal conductivities. To compute the total current, sum all components of I. Compute the average over 100 random lattices of equal size and normalize $\langle I \rangle$ to the current I_0 flowing through the complete lattice. The result should look approximately as in Fig. 3.13 (left). Draw the ratio of the number of connections P_{active} actually carrying current to the number of original connections P (Fig. 3.13 (right)). Determine the critical fraction of the dropped connections p_{c} at which the lattice, on the average, becomes globally unconnected and stops conducting.

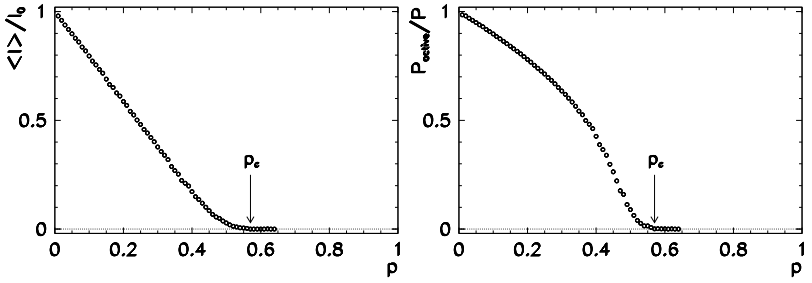


Fig. 3.13 [*Left*] The current flowing from the lattice of resistors as a function of the fraction p of dropped connections. [*Right*] The fraction of the connections actually carrying current. At the critical value $p_c \approx 0.57$ the lattice no longer conducts

⊕ To implement the lattice, we can also use nodes with attached connections spanning half of the inter-node distance. We randomly fill the sites of an empty lattice with nodes with attached connections. Eventually, an agglomeration of nodes builds up and the lattice is filled to a degree r. At some critical degree r_c the lattice becomes globally connected and starts to conduct. Compute $\langle I \rangle / I_0$ in dependence of r for the same lattices as in the first part of the Problem, and estimate r_c. What are the differences between the two approaches?

At some r the lattice contains connected clusters with typical extensions of $l = \max |r_i - r_j|$ where $(i, j) \in$ cluster. How does the average cluster size $\langle l \rangle$ change with r in the case of 100×100, 200×200, and 500×500 lattices? Compute the average over 100 different lattices of the same size.

3.6.2 Electric Circuits of Linear Elements

Electric circuits are systems of resistors, coils, capacitors, diodes, transistors, and other components. By Kirchhoff's laws, the sum of the currents at each node is zero, and the sum of the voltage drops in any closed loop of the circuit is zero. The time evolution of circuits is hard to predict if its elements are non-linear, and may even be chaotic [95]. If the circuit contains only linear elements, the usual relations $U = RI$, $U = -L\dot{I}$, and $U = e/C$ apply, and the system is analytically solvable.

⊙ Devise a connected electric circuit containing resistors, coils, and capacitors. The number of the elements n should be larger than 5. By using Kirchhoff's laws, obtain a homogeneous system of linear differential equations for the voltage drops $\{U_i\}_{i=1}^{n}$ on individual elements. Introduce new variables such that the problem can be reformulated as a single differential equation of order one. Let us denote all variables by $\{X_i\}_{i=1}^{m \geq n}$. For the solution of the equation, we use the ansatz $X_i = \exp(\mathrm{i}\omega t) Y_i$ where $Y_i \in \mathbb{C}$ are constants, so we obtain a non-homogeneous eigensystem. Find the spectrum of frequencies ω for this system by using any of the methods described in the text. Observe the changes in the spectrum when the values of the resistances (energy losses) are modified.

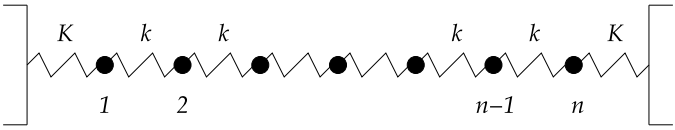


Fig. 3.14 A chain of n small masses connected by springs with elastic coefficients k. The chain is attached to the walls by springs with different coefficients K

⊕ Another option is to use the ansatz $U_i(t) = \exp(\mathrm{i}\omega t)V_i$ in the original equation, where $V_i \in \mathbb{C}$ are constants. This yields a linear homogeneous system of equations for the vector $v = \{V_i\}_{i=1}^n$ of the form $A(\omega)v = 0$. We expect non-trivial solutions at the frequencies ω which are zeros of the determinant of the matrix $A(\omega)$: $\det(A(\omega)) = 0$. In general, the polynomial $\det(A(\omega))$ is complex, and its roots can be found by general programs (see Appendix I).

3.6.3 Systems of Oscillators

Imagine a one-dimensional chain of n small spheres with masses m connected by springs with elastic coefficients k. At its ends, the chain is attached to the wall by springs with coefficients K (Fig. 3.14).

⊙ Analyze the spectrum of oscillations of this system when the ratio k/K is changed, and try to identify its thermodynamic limit $n \to \infty$. Discuss the spectrum in the case when one of the masses is heavier than the others. What happens if all but one sphere move without friction, while a single one of them is submerged in a liquid so that the linear drag law $F_\mathrm{u} = -6\pi\eta r v$ applies to it? Explore the dynamics of the spectrum of oscillations when the damping or the ratio η/k are increased!

⊕ Submerge the whole system in a liquid with viscosity η. Assume linear drag for all spheres and watch how the spectrum changes when η is increased. Set all elastic coefficients to be the same, $k = K$, and repeat the analysis.

3.6.4 Image Compression by Singular Value Decomposition

Singular value decomposition (SVD, Sect. 3.3.2) represents a particular choice of the orthonormal basis for the domain and range of the mapping $A : x \mapsto Ax$ such that the matrix in it becomes diagonal. In one form or another, SVD lies at the heart of many classical algorithms for data reduction by which we attempt to represent the data with as few parameters as possible. This Problem acquaints us with using SVD for image compression.

⊙ Perform the singular value decomposition of the 512×512 matrix representing the black-and-white image of Nicolas Copernicus (Fig. 3.6). The pixel map

(in shades of gray) can be found at the website of the book in `ASCII` format. Calculate the Frobenius norm of the matrix. Generate the SVD expansions (3.16) of rank 1, 2, 4, 8, 16, 32, 64, and 128. Convert the matrices of the lower ranks to the format of the original image and see if what you see is identifiable as a face or whether it is pleasant to the eye. How does the norm of the approximate matrix and its relative norm with respect to the original norm change with the rank of the approximation? Draw the magnitudes of the singular values as a function of their index. At any given rank of the expansion (3.16), compare the amount of memory needed to store the approximate image to the amount of memory needed to store the original image (the compression ratio). At what rank your naked eye perceives the approximate image as indistinguishable from the original? What does the image look like if *all but the first two* components are included in the singular expansion?

⊕ Perform the singular value decomposition of the Jackson Pollock's painting *Lavender Mist* which is represented by the 1024×749 matrix (see the website of the book). Again draw the magnitudes of the singular values as a function of their index, and calculate the relative error of the norm of the approximate matrix. Determine the rank needed to achieve a 10 % relative error between the approximate and the original matrix, as measured in the Frobenius norm. Compare your results to those obtained from the image of Copernicus. Do your conclusions change if you take a completely random matrix?

3.6.5 Eigenstates of Particles in the Anharmonic Potential

In this Problem (adapted from [96]) we study the one-dimensional linear harmonic oscillator (particle of mass m with kinetic energy $T(p) = p^2/(2m)$ in the quadratic potential $V(q) = m\omega^2 q^2/2$). The corresponding Hamiltonian is

$$H_0 = \frac{1}{2}\left(p^2 + q^2\right),$$

where energy is in units of $\hbar\omega$, momentum in units of $(\hbar m\omega)^{1/2}$, and linear dimensions in units of $(\hbar/m\omega)^{1/2}$. The eigenstates $|n\rangle$ of the unperturbed Hamiltonian H_0 are known from the introductory quantum mechanics courses: in coordinate representation, they are $|n\rangle = (2^n n!\sqrt{\pi})^{-1/2} \mathrm{e}^{-q^2/2} \mathcal{H}_n(q)$, where $\mathcal{H}_n$ are the Hermite polynomials. The eigenfunctions satisfy the stationary Schrödinger equation

$$H_0 |n^0\rangle = E_n^0 |n^0\rangle$$

with non-degenerate eigenenergies $E_n^0 = n + 1/2$ for $n = 0, 1, 2, \ldots$. The matrix $\langle i|H_0|j\rangle$ is obviously diagonal, with values $\delta_{i,j}(i + 1/2)$ on the diagonal. Now let us add the anharmonic term to the unperturbed Hamiltonian,

$$H = H_0 + \lambda q^4.$$

How does this perturbation modify the eigenenergies? We are looking for the matrix elements $\langle i|H|j\rangle$ of the *perturbed* Hamiltonian in the basis of the *unperturbed* wave-functions $|n^0\rangle$. In the calculation, we make use of the expectation value of the transition matrix element

$$q_{ij} = \langle i|q|j\rangle = \frac{1}{2}\sqrt{i + j + 1}\,\delta_{|i-j|,1},$$

which embodies the selection rule for electric dipole transitions between the levels of the harmonic oscillator. In the practical calculation the matrices q_{ij} and $\langle i|H|j\rangle$ of course need to be limited to finite sizes $N \times N$.

⊙ Use diagonalization methods to find the lowest eigenvalues (energies) and eigenfunctions (wave-functions) of the perturbed Hamiltonian $H = H_0 + \lambda q^4$ with the parameter $0 \le \lambda \le 1$. You are solving the eigenvalue problem

$$H|n\rangle = E_n|n\rangle.$$

The new (corrected) wave-functions $|n\rangle$ are, of course, linear combinations of the old (unperturbed) wave-functions $|n^0\rangle$. Make sure that $E_n \to E_n^0$ when $\lambda \to 0$. Examine the dependence of the results on the matrix dimension N and observe the convergence of eigenvalues at large N.

Instead of computing the matrix elements q_{ij} and understanding the perturbation matrix as $[q_{ij}]^4$, we might also wish to compute the transition matrix elements of the *square* of the coordinate, $q_{ij}^{(2)} = \langle i|q^2|j\rangle$ and understand the perturbation λq^4 as the square of the corresponding matrix, or simply compute the matrix elements of the fourth power of the coordinate, $q_{ij}^{(4)} = \langle i|q^4|j\rangle$ and take *this* matrix as the perturbation, that is,

$$\lambda q^4 \to \lambda[q_{ij}]^4 \quad \text{or} \quad \lambda q^4 \to \lambda\big[q_{ij}^{(2)}\big]^2 \quad \text{or} \quad \lambda q^4 \to \lambda\big[q_{ij}^{(4)}\big].$$

Identify the differences between these three methods! Use the equations

$$\langle i|q^2|j\rangle = \frac{1}{2}\big[\sqrt{j(j-1)}\delta_{i,j-2} + (2j+1)\delta_{i,j} + \sqrt{(j+1)(j+2)}\delta_{i,j+2}\big],$$

$$\begin{aligned}\langle i|q^4|j\rangle = \frac{1}{2^4}\sqrt{\frac{2^i\,i!}{2^j\,j!}}\big[&\delta_{i,j+4} + 4(2j+3)\delta_{i,j+2} + 12\big(2j^2+2j+1\big)\delta_{i,j} \\ &+ 16j\big(2j^2-3j+1\big)\delta_{i,j-2} + 16j\big(j^3-6j^2+11j-6\big)\delta_{i,j-4}\big],\end{aligned}$$

which are easy to derive from the recurrence relations for Hermite polynomials.

⊕ Compute the lowest eigenenergies and eigenfunctions for the problem in the "Mexican-hat" potential with two minima,

$$H = \frac{p^2}{2} - 2q^2 + \frac{q^4}{10}.$$

3.6.6 Anderson Localization

One of the important successes of 20th-century physics was the discovery of the localization of quantum states at impurities in magnets and disordered systems, which restrict the diffusion through such systems [97]. In this Problem we examine the localization in a chain of N harmonic oscillators with random masses $M_i > 0$, which is attached at its ends [98]. Let q_i be the deflection of the ith mass from equilibrium, $p_i = M_i \dot{q}_i$ its momentum, and K the elastic constant of the springs connecting the masses. The Hamiltonian of the system is

$$H = \frac{1}{2} \sum_{i=0}^{N-1} \frac{p_i^2}{2M_i} + \frac{K}{2} \sum_{i=0}^{N-1} (q_{i+1} - q_i)^2.$$

The boundary condition requires $q_{-1} = q_N = 0$. The equation of motion for the ith oscillator is $M_i \ddot{q}_i = K(q_{i+1} - 2q_i + q_{i-1})$. With the effective couplings $w_i = K/M_i$ and substitution $x_i = q_i/\sqrt{w_i}$ the system can be rewritten as

$$\frac{\mathrm{d}^2 x_i}{\mathrm{d}t^2} = \sqrt{w_i w_{i+1}}\, x_{i+1} - 2w_i x_i + \sqrt{w_i w_{i-1}}\, x_{i-1}, \quad i = 0, 1, \ldots, N-1,$$

or, in matrix form,

$$\ddot{z} = -Wz, \quad W = \begin{pmatrix} 2w_0 & -\sqrt{w_0 w_1} & 0 & 0 & \ldots \\ -\sqrt{w_0 w_1} & 2w_1 & -\sqrt{w_1 w_2} & 0 & \ldots \\ 0 & -\sqrt{w_1 w_2} & 2w_2 & -\sqrt{w_1 w_3} & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{pmatrix}, \tag{3.37}$$

where $z = (x_0, x_1, \ldots, x_{N-1})^{\mathrm{T}}$. By solving the eigenvalue problem

$$Wz = \omega^2 z, \quad z \neq 0,$$

we obtain the eigenvalues (angular frequencies) $\{\omega_i\}_{i=0}^{N-1}$ and the corresponding eigenvectors (eigenmodes) $\{z_i\}_{i=0}^{N-1}$ which are normalized as $\|z_i\|_2 = 1$. By using the known pairs $\{(\omega_i, z_i)\}_{i=0}^{N-1}$ we can rewrite any solution of the differential equation (3.37) as the sum

$$x(t) = \sum_{i=0}^{N-1} \left[a_i^{\mathrm{T}} z_i \, e^{+\mathrm{i}\omega_i t} + b_i^{\mathrm{T}} z_i \, e^{-\mathrm{i}\omega_i t} \right],$$

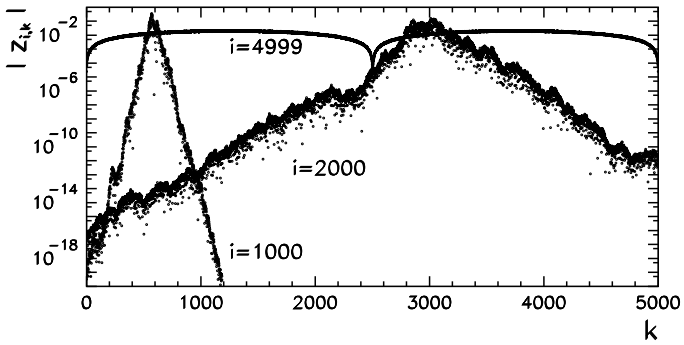


Fig. 3.15 Examples of eigenmodes in a slightly disordered harmonic chain with $N = 5000$ in the case when the weights w_i are uniformly distributed on [0.9, 1.1]

where the vectors $a_i, b_i \in \mathbb{C}^N$ are determined from the initial conditions $x(0)$ and $\dot{x}(0)$. We are interested in the eigenmodes. The energies ω_i^2 of the eigenmodes are homogeneous functions of the reciprocal masses w_i, so $(w_i \to \lambda w_i) \Leftrightarrow (\omega_i^2 \to \lambda\omega_i^2)$. This enables us to fix one moment of the distribution of the values w_i, for example, its average $\langle w \rangle = 1$. The high-energy eigenmodes typically have a limited range, which is known as the *localization of modes*. Some examples are shown in Fig. 3.15.

A physically relevant quantity is the localization length. It can be defined by the average of any quantity f with respect to the ith eigenmode $z_i = (z_{i,k})_{k=0}^{N-1}$,

$$\langle f \rangle_{\alpha,i} = \frac{\sum_k f(k)|z_{i,k}|^\alpha}{\sum_k |z_{i,k}|^\alpha},$$

where $z_{i,k}$ denotes the kth component of the ith mode, and $\alpha > 0$. Assuming that the eigenmode has an exponential localization $|z_{i,k}| \propto \exp(-|k-a|/\xi_i)$, the localization length ξ_i may be approximated by

$$\xi_i \approx \sqrt{\langle k^2 \rangle_{\alpha,i} - \langle k \rangle_{\alpha,i}^2}.$$

⊙ Analyze the chains of harmonic oscillators with different distributions of weights w_i for dimensions $N = 10^3$ and 10^4. First use the binomial distribution: to the weight w_i assign the value W_0 with probability p and the value W_1 with probability $1 - p$, so $\mathcal{P}(w_i = W_0) = p$ and $\mathcal{P}(w_i = W_1) = 1 - p$. Then use the exponential distribution: the weights w_i are distributed with the cumulative distribution $P(w_i \le w) = 1 - \exp(-\lambda w)$ at constant $\lambda > 0$. Check that for different λ the system of eigenmodes does not change significantly, and that similar conclusions apply for the distribution of $\lambda\omega_i^2$.

For each of these cases, draw the shapes of the eigenmodes. Plot z_i, x_i, and the distribution of the energies $\{\omega_i^2\}_{i=0}^{N-1}$, of the average position $\{\langle k \rangle_{\alpha=1,i}\}_{i=0}^{N-1}$, and of

the localization length $\{\xi_i\}_{i=0}^{N-1}$. Is there a statistically meaningful correlation between these quantities? To diagonalize the matrix W, use programs for tridiagonal matrices (see Table 3.3).

⊕ Study the spectral properties of a more abstract system of first-order differential equations

$$\dot{x}_i = -x_{i+1} + w_i x_i - x_{i-1}, \quad i = 0, 1, \ldots, N-1,$$

where $x_{-1} = x_N = 0$. Distribute the weights w_i around the value zero ($\langle w_i \rangle = 0$) with a standard deviation of $\langle w_i^2 \rangle^{1/2} = 1$. Find the eigenmodes by using the ansatz $x_i(t) = y_i \exp(\lambda t)$. Due to the symmetry of the matrix elements we have $\lambda \in \mathbb{R}$. Calculate the distribution of the localization lengths and eigenenergies in the case of uniformly and normally distributed weights w_i for $N = 10^2$, 10^3, and 10^4. Make sure that all eigenmodes are localized and that their centers-of-gravity are uniformly distributed over the modes.

3.6.7 Spectra of Random Symmetric Matrices

In quantum mechanics the state of the system is given by the vector $\psi \in \mathbb{C}^n$. Its time evolution is described by the Schrödinger equation

$$\mathrm{i}\hbar \frac{\mathrm{d}\psi}{\mathrm{d}t} = \widehat{H}\psi,$$

in which the Hamiltonian $\widehat{H}$ is represented by the Hermitian matrix H. In time-reversal symmetric systems a basis in the space of states exists such that H is real and symmetric. Without this symmetry, H is general Hermitian.

Quantum systems whose classical analogues are chaotic behave statistically in some aspects [75]. The paradigmatic classes of such systems are the particle in a multi-dimensional potential well of irregular shape or the system of strongly correlated particles like nucleons in heavy nuclei. In such systems, at times longer than $\mathcal{O}(\hbar^{-1})$, the wave-function becomes statistically similar to a *random wave* [99]. Moreover, the unfolded energy spectra E_i or the fluctuations of the splittings between the neighboring levels $(E_{i+1} - E_i)$ possess universal statistics (see Sect. 3.5.2 and [88]). In the case when the system is time-reversal symmetric, the splittings obey the statistics of the GOE matrix ensemble (3.35), while if there is no such invariance, they obey the statistics for GUE (3.36). This finding established the connection between the theory of random matrices and quantum physics.

⊙ Let us restrict ourselves to matrices from the GOE. Check (analytically and numerically) that the differences between the eigenenergies of 2×2 matrices from the GOE of the form $((a, b), (b, c))$ are distributed according to (3.35), where a, $b/\sqrt{2}$, and c are distributed randomly according to $N(0, 1)$. (Randomly generate a large number of such matrices, compute the eigenvalues for each of them, and plot their distribution.)

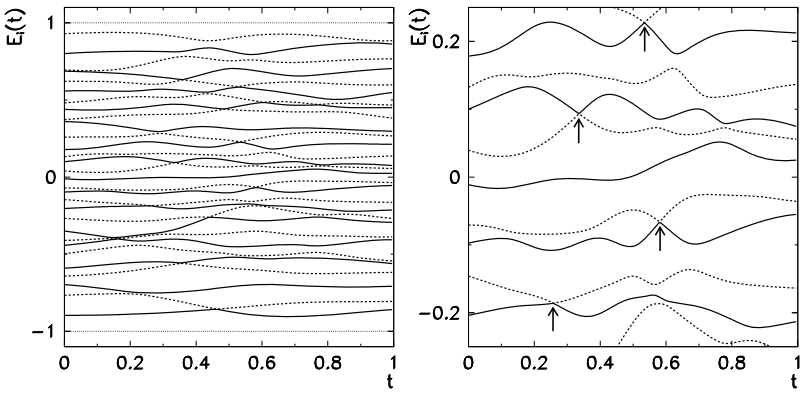


Fig. 3.16 [*Left*] Energy levels corresponding to the 30×30 Hamiltonian matrix in dependence of the parameter t. [*Right*] A blow-up of the energy axis near its origin. The *arrows* indicate the most prominent examples of the avoidance of crossing

Randomly generate the matrix $H \in \mathbb{R}^{n\times n}$ from the GOE with a size as large as possible ($n \approx 5000$), and find its eigenenergies by any method adapted to real symmetric matrices. Compute the distribution of the energies $\mathrm{d}p/\mathrm{d}x$ where $x = E/\sqrt{2n}$, and compare it to the Wigner's semi-circular distribution

$$p(E) \propto \sqrt{1 - E^2/(2n)}.$$

Unfold the energy spectrum $\{E_i\}_{i=1}^n$ by following the procedure outlined in Sect. 3.5.2: form the cumulative distribution $N(E) = \#\{E_i \le E\}$, separate N to its smooth and oscillatory part, $N(E) = \overline{N}(E) + N_{\mathrm{osc}}(E)$, and then unfold the spectrum into $\{\tilde{E}_i = \overline{N}(E_i)\}_{i=1}^n$ such that the average distance between the neighboring levels $S = \tilde{E}_{i+1} - \tilde{E}_i$ becomes locally (over a few levels) and globally equal to 1. Finally, compute the distribution density of the splittings (p_{s} as a function of S) between neighboring levels, and compare it to the distribution (3.35).

$\bigoplus$ Let A and B be $n \times n$ random matrices from the GOE. We use their linear combination to define the Hamiltonian

$$H(t) = \sqrt{\frac{n}{4\mathrm{tr}(\mathcal{H}(t)^2)}}\mathcal{H}(t), \quad \mathcal{H}(t) = (1-t)A + tB,\ t \in [0,1].$$

Compute the spectrum of H for $n = 10$, 50, and 100 as a function of the parameter t. You should observe the typical behavior of the energy levels similar to the one shown in Fig. 3.16. At some places, the levels appear to approach each other and then recede, the phenomenon known as the *avoidance of crossing* or *level repulsion*. This is a characteristic feature of classically chaotic systems and can already be inferred from the distributions (3.35) and (3.36) since $p(0) = 0$. Make sure that for each t and large enough n (say, $n = 2000$) the distributions of energies and energy splittings are as they should be for matrices from the GOE.

References

1. J.W. Demmel, *Applied Numerical Linear Algebra* (SIAM, Philadelphia, 1997)
2. G.H. Golub, C.F. Van Loan, *Matrix Computations*, 3rd edn. (Johns Hopkins University Press, Baltimore, 1996)
3. LAPACK, The Linear Algebra PACKage. http://www.netlib.org/lapack (version for Fortran 77), ../lapack95 (version for Fortran 95), ../clapack (version for C), ../lapack++ (version for C++), ../java/f2j (version for Java)
4. E. Anderson et al., *LAPACK Users' Guide*, 3rd edn. (SIAM, Philadelphia, 1999)
5. V. Strassen, Gaussian elimination is not optimal. Numer. Math. **13**, 354 (1969)
6. J. Demmel, N.J. Higham, Stability of block algorithms with fast level 3 BLAS. ACM Trans. Math. Softw. **18**, 274 (1992)
7. N.J. Higham, Exploiting fast matrix multiplication within the level 3 BLAS. ACM Trans. Math. Softw. **16**, 352 (1990)
8. S. Chatterjee, A.R. Lebeck, P.K. Patnala, M. Thottethodi, Recursive array layouts and fast matrix multiplication. IEEE Trans. Parallel Distrib. Syst. **13**, 1105 (2002)
9. K. Goto, R.A. van de Geijn, Anatomy of high-performance matrix multiplication. ACM Trans. Math. Softw. **34**, art. 12 (2008)
10. N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd edn. (SIAM, Philadelphia, 2002)
11. D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progressions. J. Symb. Comput. **9**, 251 (1990)
12. C.C. Douglas, M.A. Heroux, G. Slishman, R.M. Smith, GEMMW: a portable level-3 BLAS Winograd variant of Strassen's matrix-matrix multiply algorithm. J. Comput. Phys. **110**, 1 (1994)
13. P. D'Alberto, A. Nicolau, Adaptive Winograd's matrix multiplications. ACM Trans. Math. Softw. **36**, art. 3 (2009)
14. J. Demmel, I. Dumitriu, O. Holtz, R. Kleinberg, Fast matrix multiplication is stable. Numer. Math. **106**, 199 (2007)
15. J.W. Demmel, N.J. Higham, Stability of block algorithms with fast level-3 BLAS. ACM Trans. Math. Softw. **18**, 274 (1992)
16. D.H. Bailey, K. Lee, H.D. Simon, Using Strassen's algorithm to accelerate the solution of linear systems. J. Supercomput. **4**, 357 (1990)
17. J. Demmel, The componentwise distance to the nearest singular matrix. SIAM J. Matrix Anal. Appl. **13**, 10 (1992)
18. J.W. Demmel, On condition numbers and the distance to the nearest ill-posed problem. Numer. Math. **51**, 251 (1987)
19. R.M. Gray, Toeplitz and circulant matrices: a review. Found. Trends Commun. Inf. Theory **2**, 155 (2006)
20. J.R. Bunch, Stability of methods for solving Toeplitz systems of equations. SIAM J. Sci. Stat. Comput. **6**, 349 (1985)
21. T. Kailath, J. Chun, Generalized displacement structure for block-Toeplitz, Toeplitz-block, and Toeplitz-derived matrices. SIAM J. Matrix Anal. Appl. **15**, 114 (1994)
22. T. Kailath, A.H. Sayed, Displacement structure: theory and applications. SIAM Rev. **37**, 297 (1995)
23. G.S. Anmar, W.B. Gragg, Superfast solution of real positive definite Toeplitz systems. SIAM J. Matrix Anal. Appl. **9**, 61 (1988)
24. T.F. Chan, P.C. Hansen, A look-ahead Levinson algorithm for indefinite Toeplitz systems. SIAM J. Matrix Anal. Appl. **13**, 490 (1992)
25. M.K. Ng, *Iterative Methods for Toeplitz Systems* (Oxford University Press, Oxford, 2004)
26. W. Gautschi, G. Inglese, Lower bound for the conditional number of Vandermonde matrices. Numer. Math. **52**, 241 (1988)
27. Å. Björck, V. Pereyra, Solution of Vandermonde systems of equations. Math. Comput. **24**, 893 (1970)

28. Å. Björck, T. Elfving, Algorithms for confluent Vandermonde systems. Numer. Math. **21**, 130 (1973)
29. N.J. Higham, Fast solution of Vandermonde-like systems involving orthogonal polynomials. IMA J. Numer. Anal. **8**, 473 (1988)
30. D. Calvetti, L. Reichel, Fast inversion of Vandermonde-like matrices involving orthogonal polynomials. BIT Numer. Math. **33**, 473 (1993)
31. H. Lu, Fast solution of confluent Vandermonde linear systems. SIAM J. Matrix Anal. Appl. **15**, 1277 (1994)
32. H. Lu, Solution of Vandermonde-like systems and confluent Vandermonde-like systems. SIAM J. Matrix Anal. Appl. **17**, 127 (1996)
33. D. Bini, V.Y. Pan, *Polynomial and Matrix Computations, Vol. 1: Fundamental Algorithms* (Birkhäuser, Boston, 1994), Sect. 2.4
34. W.W. Hager, Condition estimates. SIAM J. Sci. Stat. Comput. **5**, 311 (1984). See also [1], Algorithm 2.5
35. N.J. Higham, FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation. ACM Trans. Math. Softw. **14**, 381 (1988)
36. N.J. Higham, F. Tisseur, A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra. SIAM J. Matrix Anal. Appl. **21**, 1185 (2000)
37. N.J. Higham, Experience with a matrix norm estimator. SIAM J. Sci. Stat. Comput. **11**, 804 (1990)
38. N.J. Higham, A survey of condition number estimation for triangular matrices. SIAM Rev. **29**, 575 (1987)
39. D.S. Watkins, A case where balancing is harmful. Electron. Trans. Numer. Anal. **23**, 1 (2006)
40. E.E. Osborne, On pre-conditioning of matrices. J. Assoc. Comput. Mach. **7**, 338 (1960)
41. B.N. Parlett, C. Reinsch, Balancing a matrix for calculation of eigenvalues and eigenvectors. Numer. Math. **13**, 293 (1969)
42. T.-Y. Chen, J.W. Demmel, Balancing sparse matrices for computing eigenvalues. Linear Algebra Appl. **309**, 261 (2000)
43. J.J. Dongarra, I.S. Duff, D.C. Sorensen, H.A. van der Vorst, *Numerical Linear Algebra for High-Performance Computers* (SIAM, Philadelphia, 1998)
44. X. Li, Direct solvers for sparse matrices. Sept. 2006 (accessible on the web)
45. M. Heath, E. Ng, B.W. Peyton, Parallel algorithms for sparse linear systems. SIAM Rev. **33**, 420 (1991)
46. N.I.M. Gould, J.A. Scott, Y. Hu, A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations. ACM Trans. Math. Softw. **33**, 10 (2007)
47. J.A. Scott, Y. Hu, Experiences of sparse direct symmetric solvers. ACM Trans. Math. Softw. **33**, 18 (2007)
48. E.J. Haunschmid, C.W. Ueberhuber, Direct solvers for sparse systems. TU Wien, SFB F011 "AURORA" Report, 1999
49. I.S. Duff, M.A. Heroux, R. Pozo, An overview of the sparse basic linear algebra subprograms: the new standard from the BLAS technical forum. ACM Trans. Math. Softw. **28**, 239 (2002)
50. J. Demmel, Y. Hida, E.J. Riedy, X.S. Li, Extra-precise iterative refinement for overdetermined least squares problems. ACM Trans. Math. Softw. **35**, 28 (2009). The routines `xGELS` offer only the basic solution without iterative improvement. This novelty is now offered by the `xGELS_X` and `xGELS_RFSX` routines
51. G. Golub, W. Kahan, Calculating the singular values and pseudo-inverse of a matrix. SIAM J. Numer. Anal. B **2**, 205 (1965)
52. D. Kalman, A singularly valuable decomposition: the SVD of a matrix. Coll. Math. J. **27**, 2 (1996)
53. M. Abate, When is a linear operator diagonalizable? Am. Math. Mon. **104**, 824 (1997)
54. D.S. Watkins, Understanding the QR algorithm. SIAM Rev. **24**, 427 (1982)
55. D.S. Watkins, The QR algorithm revisited. SIAM Rev. **50**, 133 (2008)

56. C. Van Loan, On estimating the condition of eigenvalues and eigenvectors. Linear Algebra Appl. **88/89**, 715 (1987)
57. J.H. Wilkinson, Sensitivity of eigenvalues. Util. Math. **25**, 5 (1984)
58. W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 3rd edn. (Cambridge University Press, Cambridge, 2007). See also the equivalent handbooks in Fortran, Pascal and C, as well as http://www.nr.com
59. J.J.M. Cuppen, A divide and conquer method for the symmetric tridiagonal eigenproblem. Numer. Math. **36**, 177 (1981)
60. I.S. Dhillon, B.N. Parlett, Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. Linear Algebra Appl. **387**, 1 (2004)
61. I.S. Dhillon, B.N. Parlett, C. Vömel, The design and implementation of the MRRR algorithm. ACM Trans. Math. Softw. **32**, 533 (2006)
62. H.-J. Stöckmann, *Quantum Chaos. An Introduction* (Cambridge University Press, Cambridge, 2006)
63. N.W. Ashcroft, N.D. Mermin, *Solid State Physics* (Harcourt College Publishers, Fort, Worth, 1976)
64. D.R. Hofstadter, Energy levels and wave functions of Bloch electrons in rational and irrational magnetic fields. Phys. Rev. B **14**, 2239 (1976)
65. G.H. Golub, Some modified matrix eigenvalue problems. SIAM Rev. **15**, 318 (1973)
66. R.M.M. Mattheij, G. Söderlind, On inhomogeneous eigenvalue problems. Linear Algebra Appl. **88/89**, 507 (1987)
67. G.H. Golub, J.H. Wilkinson, Ill-conditioned eigensystems and the computation of the Jordan canonical form. SIAM Rev. **18**, 578 (1976)
68. A. Bujosa, R. Criado, C. Vega, Jordan normal form via elementary transformations. SIAM Rev. **40**, 947 (1998)
69. B. Kågström, A. Ruhe, An algorithm for numerical computation of the Jordan normal form of a complex matrix. ACM Trans. Math. Softw. **6**, 398 (1980)
70. C. Moler, C. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Rev. **45**, 3 (2003)
71. C. Moler, C. Van Loan, Nineteen dubious ways to compute the exponential of a matrix. SIAM Rev. **20**, 801 (1978)
72. N.J. Higham, The scaling and squaring method for the matrix exponential revisited. SIAM Rev. **51**, 747 (2009)
73. N.J. Higham, *Functions of Matrices. Theory and Computation* (SIAM, Philadelphia, 2008)
74. W. Kerner, Large-scale complex eigenvalue problem. J. Comput. Phys. **85**, 1 (1989)
75. F. Haake, *Quantum Signatures of Chaos*, 3rd edn. (Springer, Berlin, 2006)
76. A.M. Tulino, S. Verdú, Random matrix theory and wireless communications. Found. Trends Commun. Inf. Theory **1**, 1 (2004)
77. V. Plerou et al., Random matrix approach to cross correlations in financial data. Phys. Rev. E **65**, 066126 (2002)
78. A. Edelman, N.R. Rao, Random matrix theory. Acta Numer. **14**, 233 (2005)
79. P. Bleher, A. Its, *Random Matrix Models and Their Applications* (Cambridge University Press, Cambridge, 2001)
80. M.L. Mehta, *Random Matrices*, 2nd edn. (Academic Press, San Diego, 1990)
81. T. Tao, V. Vu, From the Littlewood–Offord problem to the circular law: universality of the spectral distribution of random matrices. Bull. Am. Math. Soc. **46**, 377 (2009)
82. V.L. Girko, Circular law. Theory Probab. Appl. **29**, 694 (1985)
83. M. Horvat, The ensemble of random Markov matrices. J. Stat. Mech. **2009**, P07005 (2009)
84. D.V. Widder, The Stieltjes transform. Trans. Am. Math. Soc. **43**, 7 (1938)
85. R.M. Dudley, *Real Analysis and Probability* (Cambridge University Press, Cambridge, 2002)
86. V.A. Marčenko, L.A. Pastur, Distribution of eigenvalues for some sets of random matrices. Math. USSR Sb. **1**, 457 (1967)
87. R.R. Nadakuditi, Applied stochastic eigen-analysis. Doctoral dissertation. Massachusetts Institute of Technology, Cambridge, 1999. Accessible at http://hdl.handle.net/1912/1647

88. O. Bohigas, M.J. Giannoni, C. Schmit, Characterization of chaotic quantum spectra and universality of level fluctuation laws. Phys. Rev. Lett. **52**, 1 (1984)
89. E.P. Wigner, On the distribution of the roots of certain symmetric matrices. Ann. Math. **67**, 325 (1958)
90. G.W. Stewart, The efficient generation of random orthogonal matrices with an application to condition estimators. SIAM J. Numer. Anal. **17**, 403 (1980)
91. F. Mezzadri, How to generate random matrices from the classical compact groups. Not. Am. Math. Soc. **54**, 592 (2007)
92. S. Bunde, S. Havlin, *Fractals and Disordered Systems* (Springer, Berlin, 1991)
93. R.J. Wilson, J.J. Watkins, *Graphs: An Introductory Approach* (Wiley, New York, 1990)
94. D. Stauffer, A. Aharony, *Introduction to Percolation Theory*, 2nd edn. (Taylor & Francis, London, 1992)
95. G. Chen, T. Ueta, *Chaos in Circuits and Systems* (World Scientific, Singapore, 2002)
96. W. Kinzel, G. Reents, *Physics by Computer: Programming of Physical Problems Using Mathematica and C* (Springer, Berlin, 1997)
97. P.W. Anderson, Absence of diffusion in certain random lattices. Phys. Rev. **109**, 1492 (1958)
98. F.J. Dyson, The dynamics of a disordered linear chain. Phys. Rev. **92**, 1331 (1953)
99. M.V. Berry, Regular and irregular semiclassical wavefunctions. J. Phys. A **10**, 2083 (1977)

Chapter 4
Transformations of Functions and Signals

4.1 Fourier Transformation

The Fourier transformation $\mathcal{F}$ of the function f on the real axis is defined as

$$F(\omega) = \mathcal{F}[f](\omega) = \int_{-\infty}^{\infty} f(x)\, \mathrm{e}^{-\mathrm{i}\omega x}\, \mathrm{d}x. \tag{4.1}$$

The sufficient conditions for the existence of F are that f is absolutely integrable, i.e. $\int_{-\infty}^{\infty} |f(x)|\, \mathrm{d}x < \infty$, and that f is piece-wise continuous or has a finite number of discontinuities. The inverse transformation is

$$f(x) = \mathcal{F}^{-1}[F](x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)\, \mathrm{e}^{\mathrm{i}\omega x}\, \mathrm{d}\omega. \tag{4.2}$$

Changing the sign of the argument x of f (flipping the space or time coordinate) implies a change of the sign of the frequency ω in the transform:

$$g(x) = f(-x) \quad \Longleftrightarrow \quad G(\omega) = F(-\omega).$$

Fourier Theorem If the function f is piece-wise continuous on the interval $[-\pi, \pi]$, one can form the Fourier series

$$\frac{1}{2\pi} \sum_{n\in\mathbb{Z}} \int_{-\pi}^{\pi} f(x)\, \mathrm{e}^{\mathrm{i}n(\xi - x)}\, \mathrm{d}x = \frac{1}{2}\big[f(\xi + 0) + f(\xi - 0)\big], \quad \xi \in [-\pi, \pi].$$

It follows from this theorem that the function f which is continuous on $\mathbb{R}$ and its Fourier transform F for any $a \in \mathbb{R}$, $a > 0$, are related by the *Poisson sum*

$$\sum_{n\in\mathbb{Z}} F(n\,a) = \frac{2\pi}{a} \sum_{m\in\mathbb{Z}} f\left(\frac{2\pi m}{a}\right)$$

(see also (1.67)).

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9_4,

Parseval's Equality The integral of the square of the function f over x and the integral of the square of its transform F over ω are related by the Parseval's equality

$$\int_{-\infty}^{\infty} \big|f(x)\big|^2 \,\mathrm{d}x = \frac{1}{2\pi} \int_{-\infty}^{\infty} \big|F(\omega)\big|^2 \,\mathrm{d}\omega. \tag{4.3}$$

This invariance of the norm means that power is preserved in the transition from the temporal to the frequency representation of a signal. Parseval's equality is valid for all function expansions in Hilbert spaces; see e.g. [1].

Fourier Uncertainty Let f be normalized as $\int_{-\infty}^{\infty} |f(x)|^2 \,\mathrm{d}x = 1$. Then $|f|^2$ is non-negative and can be understood as a probability distribution of the signal with respect to x, with the first and second moments (average and variation)

$$\langle x \rangle = \int_{-\infty}^{\infty} x \big|f(x)\big|^2 \,\mathrm{d}x, \qquad \sigma_x^2 = \int_{-\infty}^{\infty} \big[x - \langle x \rangle\big]^2 \big|f(x)\big|^2 \,\mathrm{d}x.$$

By (4.3) the Fourier transform of f is also normalized, $\int_{-\infty}^{\infty} |F(\omega)|^2 \,\mathrm{d}\omega = 2\pi$. Hence $|F(\omega)|^2/(2\pi)$ is the power distribution density with respect to frequencies, with the first and second moments

$$\langle \omega \rangle = \frac{1}{2\pi} \int_{-\infty}^{\infty} \omega \big|F(\omega)\big|^2 \,\mathrm{d}\omega, \qquad \sigma_\omega^2 = \frac{1}{2\pi} \int_{-\infty}^{\infty} \big[\omega - \langle \omega \rangle\big]^2 \big|F(\omega)\big|^2 \,\mathrm{d}\omega.$$

The product of the variations in both representations of f is

$$\sigma_x^2 \sigma_\omega^2 \geq \frac{1}{4}. \tag{4.4}$$

Inequality (4.4) expresses the *Fourier uncertainty*: spatial or temporal variations of the signal are not independent of frequency variations. If variations decrease in the configuration space, they increase in the transform space, and vice versa. The lower bound of the uncertainty is achieved by Gaussian signals,

$$f(x) = \left(\frac{2s}{\pi}\right)^{1/4} \mathrm{e}^{-s(x-a)^2}, \quad s > 0,\ a \in \mathbb{R}. \tag{4.5}$$

We say that such signals have a *minimum possible width*.

Example Let us compute the Fourier uncertainty of two "bell"-shaped functions. The first example is

$$f(x) = \sqrt{\frac{2}{\pi}} \frac{1}{1+x^2}, \qquad F(\omega) = \sqrt{2\pi}\, \mathrm{e}^{-|\omega|}.$$

We compute $\langle x \rangle = 0$, $\sigma_x^2 = 1$, $\langle \omega \rangle = 0$, and $\sigma_\omega^2 = \frac{1}{2}$, thus $\sigma_x^2 \sigma_\omega^2 = \frac{1}{2}$. The second example is a narrow Gaussian shifted from the origin (see (4.5) with $s = 2$ and

$a = 5$):

$$f(x) = (4/\pi)^{1/4} \mathrm{e}^{-2(x-5)^2}, \qquad F(\omega) = \mathrm{e}^{-\frac{1}{8}\omega(\omega+40\mathrm{i})}.$$

We obtain $\langle x \rangle = 5$, $\sigma_x^2 = \frac{1}{8}$, $\langle \omega \rangle = 0$, and $\sigma_\omega^2 = 2$, hence $\sigma_x^2 \sigma_\omega^2 = \frac{1}{4}$, which indeed is the smallest possible product of variations, in accordance with (4.4).

Sampling Theorem Assume that the function (signal) f is continuous on the whole real axis and that (4.1) is its Fourier transform. The signal is uniformly sampled at the points spaced Δx apart, like $f_n = f(n\,\Delta x)$ for $n \in \mathbb{Z}$. If the range of the frequencies corresponding to the signal f is bounded, i.e. if

$$F(\omega) = 0 \quad \forall |\omega| \geq \omega_\mathrm{c} = \frac{\pi}{\Delta x}, \tag{4.6}$$

where ω_c is the Nyquist (critical) frequency, the complete signal f can be reconstructed *without loss of information* from the discrete sample $\{f_n : n \in \mathbb{Z}\}$ by using the formula

$$f(x) = \sum_{n \in \mathbb{Z}} f_n \frac{\sin \omega_\mathrm{c}(x - n\,\Delta x)}{\omega_\mathrm{c}(x - n\,\Delta x)}.$$

More on sampling can be found in the papers [2–4].

4.2 Fourier Series

4.2.1 Continuous Fourier Expansion

Following the notation of [5], the Fourier transform of the function f which is bounded and piece-wise continuous on the interval $[0, 2\pi]$, is

$$\widehat{f_k} = \frac{1}{2\pi} \int_0^{2\pi} f(x)\, \mathrm{e}^{-\mathrm{i}kx}\, \mathrm{d}x, \quad k = 0, \pm 1, \pm 2, \ldots. \tag{4.7}$$

The functions $\phi_k(x) = \mathrm{e}^{\mathrm{i}kx}$ and $\phi_k^*(x) = \mathrm{e}^{-\mathrm{i}kx}$ are orthogonal on $[0, 2\pi]$,

$$\int_0^{2\pi} \phi_j(x) \phi_k^*(x)\, \mathrm{d}x = 2\pi\, \delta_{j,k}.$$

The functions ϕ_k form the basis of the space $\mathcal{T}_N$ of trigonometric polynomials of degree $\leq N/2$. The Fourier cosine and sine transforms are the real and imaginary parts of the transform (4.7),

$$a_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) \cos kx\, \mathrm{d}x, \qquad b_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) \sin kx\, \mathrm{d}x,$$

where all three forms are related by $\widehat{f}_k = a_k - \mathrm{i}b_k$. In general, the functions f and their Fourier coefficients $\widehat{f}_k$ are complex. Specific symmetry properties of the signals correspond to specific properties of their transforms. The following cases are the most relevant:

$$\begin{aligned} \text{if} \quad & f \text{ is real}, && \widehat{f}_{-k} = \widehat{f}_k^*; \\ & f \text{ is real and even}, && \widehat{f}_k \text{ is real and even}; \\ & f \text{ is real and odd}, && \widehat{f}_k \text{ is imaginary and odd}. \end{aligned}$$

If f is real, the coefficients of its cosine and sine transform a_k and b_k are also real, and $\widehat{f}_{-k} = \widehat{f}_k^*$. The function f can be represented by the Fourier series

$$Sf(x) = \sum_{k=-\infty}^{\infty} \widehat{f}_k \phi_k(x), \tag{4.8}$$

with the coefficients given by (4.7), or by its truncated form,

$$S_N f(x) = \sum_{k=-N/2}^{N/2-1} \widehat{f}_k \phi_k(x). \tag{4.9}$$

Remark on Notation The normalization factor $(2\pi)^{-1}$ in (4.7) and (4.8) has changed its place relative to (4.1) and (4.2). In both cases the inverse transformation of the transform restores the original function. Other variants are widely used, e.g. swapping the roles of $\phi_k(x) = \mathrm{e}^{\mathrm{i}kx}$ and $\phi_k^*(x) = \mathrm{e}^{-\mathrm{i}kx}$: the opposite phases cause only a sign change in the imaginary components. These conventions partly originate in different interpretations of the sign of the frequencies: the Schrödinger equation $\mathrm{i}\hbar\partial\psi/\partial t = E\psi$ forces the physicist to associate $\mathrm{e}^{\mathrm{i}\omega t}$ with a quantum state with *negative* energy $E = -\hbar\omega$, while an electronics engineer will see it as a signal with *positive* frequency.

If the function f is continuous, periodic, and has a bounded variation on $[0, 2\pi]$, i.e. $\int_0^{2\pi} |f'(x)|\,\mathrm{d}x < \infty$, the series $Sf(x)$ uniformly converges to $f(x)$, so $\lim_{N\to\infty} \max_x |f(x) - S_N f(x)| = 0$. If f has a bounded variation on $[0, 2\pi]$, then $S_N f(x)$ converges to $(f(x+0) + f(x-0))/2$ for each $x \in [0, 2\pi]$. If f is continuous and periodic, the Fourier series does not necessarily converge at each $x \in [0, 2\pi]$. The Fourier series of $f \in L^2(0, 2\pi)$ converges to f in the sense

$$\lim_{N\to\infty} \int_0^{2\pi} \left| f(x) - S_N f(x) \right|^2 \mathrm{d}x = 0.$$

The truncation of $Sf(x)$ to $S_N f(x)$ containing a finite number of terms implies an error. Its magnitude is determined by the asymptotics of the Fourier coefficients,

$$\max_{0\le x\le 2\pi} \left| f(x) - S_N f(x) \right| \le \sum_{k<-N/2} |\widehat{f}_k| + \sum_{k>N/2-1} |\widehat{f}_k|.$$

If f is m-times ($m \geq 1$) continuously differentiable on $[0, 2\pi]$ and its jth derivative is periodic for all $j \leq m-2$, we have

$$\widehat{f_k} = \mathcal{O}\big(|k|^{-m}\big), \quad k \to \infty.$$

There are several methods to estimate the truncation error. We quote [5] for the estimates in function spaces $L^2(0, 2\pi)$ and $L^p(0, 2\pi)$ defined in Appendix A. In the norm in $L^2(0, 2\pi)$, $S_N f$ is the best approximation for f among all functions from $\mathcal{T}_N$. The estimate

$$\|f - S_N f\|_{L^2(0,2\pi)} \leq C N^{-m} \big\| f^{(m)} \big\|_{L^2(0,2\pi)}$$

is valid for $m \geq 0$ for any m-times differentiable function f.

4.2.2 Discrete Fourier Expansion

The Fourier expansion of a function f for which the values on $[0, 2\pi]$ are known at the points (nodes)

$$x_j = 2\pi j/N, \quad j = 0, 1, \ldots, N-1, \ N \text{ even}, \tag{4.10}$$

is represented by the coefficients of the *discrete Fourier transform* (DFT)

$$\widetilde{f_k} = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j)\, \mathrm{e}^{-\mathrm{i}kx_j}, \quad -N/2 \leq k \leq N/2 - 1. \tag{4.11}$$

Another Remark on Notation Since N is even, $\widetilde{f}_{-N/2} = \widetilde{f}_{N/2}$, and thus an expansion with $|k| \leq N/2$ is also sensible: for the coefficient at $k = N/2$ we then use the normalization $1/(2N)$ instead of $1/N$. The index k representing the frequency ω may also run from 0 to N. The lower portion of the spectrum ($1 \leq k \leq N/2 - 1$) then corresponds to the negative frequencies, $-\omega_\mathrm{c} < \omega < 0$, while the upper portion ($N/2 + 1 \leq k \leq N - 1$) corresponds to the positive ones, $0 < \omega < \omega_\mathrm{c}$, where $\omega_\mathrm{c} = N/2$. The value at $k = 0$ belongs to frequency zero (the average of the signal) while the value at $k = N$ maps to both ω_c and $-\omega_\mathrm{c}$, and can be omitted so that we have precisely N distinct coefficients (Fig. 4.1).

For real functions f it holds that

$$\widetilde{f}_{N-k} = \widetilde{f}_k^*, \quad \widetilde{f}_0 \in \mathbb{R}.$$

The function value at the point x_j is obtained by the inverse DFT, that is, by the sum

$$f(x_j) = \sum_{k=-N/2}^{N/2-1} \widetilde{f_k}\, \mathrm{e}^{\mathrm{i}kx_j}, \quad j = 0, 1, \ldots, N-1. \tag{4.12}$$

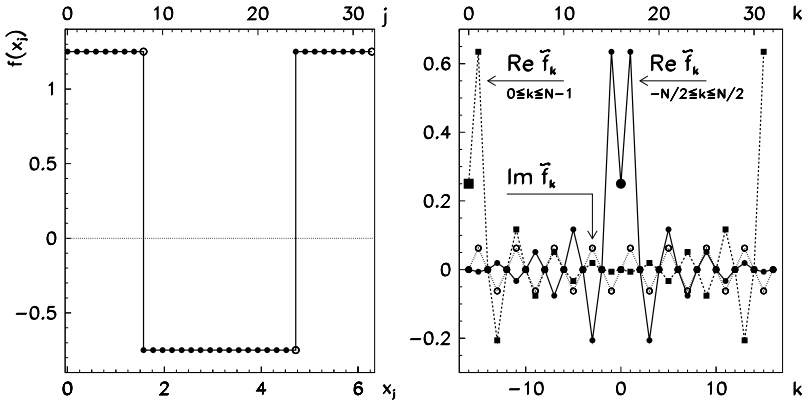


Fig. 4.1 Discrete Fourier transformation. [*Left*] The sum of the constant 0.25 and the square wave on $[0, 2\pi]$ is sampled at $N = 32$ points $x_j = 2\pi j/N$, $0 \le j \le N-1$ (the • *symbols*). At the points ∘ the function is not sampled! [*Right*] The real and imaginary parts of the Fourier coefficients for two different ways of indexing ($-N/2 \le k \le N/2$ or $0 \le k \le N-1$). The *thick symbols* at $k = 0$ denote the zero-frequency component, which is the average of the signal—precisely the 0.25 vertical shift on the *left figure* (If we subtract the average of the signal, the zeroth component of its transform is zero)

Let $I_N f$ denote the interpolant of the function f at N points. The discrete Fourier series

$$I_N f(x) = \sum_{k=-N/2}^{N/2-1} \widetilde{f}_k \, \mathrm{e}^{\mathrm{i}kx} \tag{4.13}$$

interpolates f, hence $I_N f(x_j) = f(x_j)$. The points x_j at which the value of f exactly coincides with the value of the interpolant $I_N f$ are known as the *Fourier collocation points*. The interpolant can also be written as

$$I_N f(x) = \sum_{j=0}^{N-1} f(x_j) g_j(x), \tag{4.14}$$

where

$$g_j(x) = \frac{1}{N} \sin \frac{N(x-x_j)}{2} \operatorname{ctg} \frac{x-x_j}{2} \tag{4.15}$$

are the characteristic Lagrange trigonometric polynomials with the property $g_j(x_i) = \delta_{i,j}$. The first three polynomials for $N = 8$ are shown in Fig. 4.2. An example of the trigonometric interpolation of a periodic function $f(x) = 2/(4 - 3\cos x)$ is shown in Fig. 4.3.

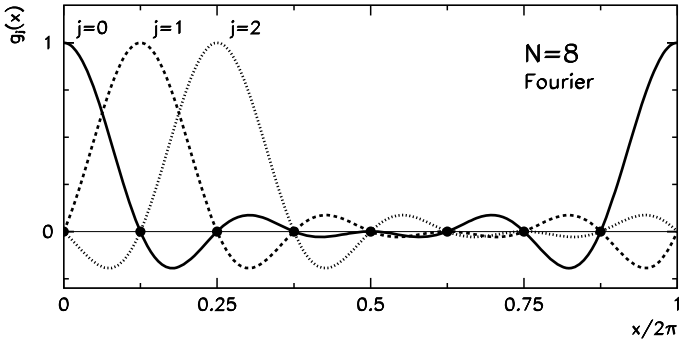


Fig. 4.2 The first three Lagrange trigonometric polynomials $g_j(x)$ in the case $N = 8$. The maxima are at the Fourier collocation points $x_j = 2\pi j/N$ ($j = 0, 1, \ldots, N-1$)

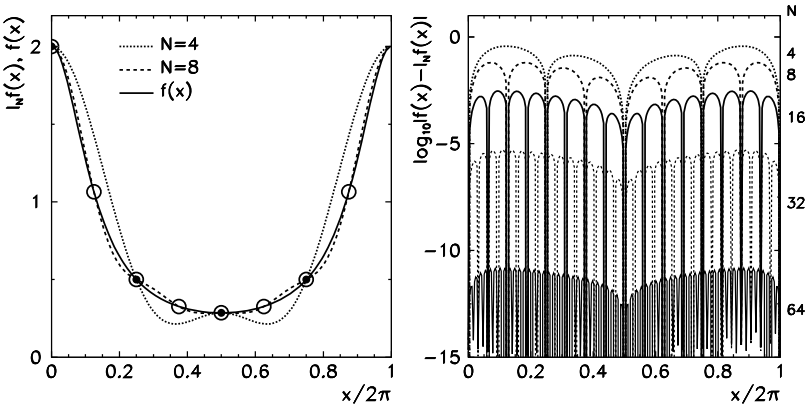


Fig. 4.3 Trigonometric interpolation of the function $f(x) = 2/(4 - 3\cos x)$. [*Left*] The interpolants $I_4 f$ (matching the function f at four points, *symbols* •) and $I_8 f$ (matching at eight points, symbols ∘). [*Right*] The error of the discrete Fourier series for $N = 4$, 8, 16, 32, and 64. Note the exceptionally rapid drop-off of the error with the increasing number of points (*right vertical axis*): it is known as spectral convergence and attains its full relevance in the spectral methods of Chap. 11. The interpolation property $I_N f(x_j) = f(x_j)$ is seen in the characteristic spikes of vanishing error

The quadrature formula

$$\frac{1}{2\pi}\int_0^{2\pi} f(x)\,\mathrm{d}x = \frac{1}{N}\sum_{j=0}^{N-1} f(x_j)$$

with the Fourier points (4.10) is exact for any function of the form $\mathrm{e}^{-\mathrm{i}kx}$ for $k \in \mathbb{Z}$, $|k| < N$ or any trigonometric polynomial of degree less than N which is a linear combination of such functions. The DFT can therefore also be understood as an approximation of the continuous Fourier transform of a periodic function f on the

interval $[0, 2\pi]$:

$$\frac{1}{2\pi}\int_0^{2\pi} f(x)\,\mathrm{e}^{-\mathrm{i}kx}\,\mathrm{d}x = \frac{1}{N}\sum_{j=0}^{N-1} f(x_j)\,\mathrm{e}^{-\mathrm{i}2\pi jk/N} + \mathcal{R}_N,$$

where $\mathcal{R}_N = -f''(\xi)(2\pi)^2/(12N^2)$ and $\xi \in [0, 2\pi]$.

4.2.3 Aliasing

The coefficients of the discrete Fourier transform (4.11) and the coefficients of the exact expansion (4.7) are related by

$$\widetilde{f_k} = \widehat{f_k} + \sum_{\substack{m=-\infty \\ m\neq 0}}^{\infty} \widehat{f}_{k+Nm}, \quad k = -N/2, \ldots, N/2-1. \tag{4.16}$$

By using (4.9) and (4.13) this can be written as $I_N f = S_N f + R_N f$. The remainder

$$R_N f = I_N f - S_N f = \sum_{k=-N/2}^{N/2-1} \left(\sum_{\substack{m=-\infty \\ m\neq 0}}^{\infty} \widehat{f}_{k+Nm} \right) \phi_k \tag{4.17}$$

is called the *aliasing error* and it measures the difference between the interpolation polynomial and the truncated Fourier series. Aliasing implies that the Fourier component with the wave-number $(k + Nm)$ behaves like the component with the wave-number k. Because the basis functions are periodic,

$$\phi_{k+Nm}(x_j) = \phi_k(x_j), \quad m \neq 0,$$

such components are indistinguishable (Fig. 4.4). In other words, on a discrete mesh, the kth Fourier component of the interpolant $I_N f$ depends not only on the kth component of f, but also on the higher components mimicking the kth.

The aliasing error is orthogonal to the truncation error $f - S_N f$, thus

$$\|f - I_N f\|^2 = \|f - S_N f\|^2 + \|R_N f\|^2.$$

The interpolation error is therefore always larger than the truncation error.

Example Knowing how to control aliasing has important practical consequences. The signals on standard audio compact discs are sampled at the frequency of $\nu_\mathrm{s} = 44.1$ kHz. The critical frequency (4.6) is then $\nu_\mathrm{c} = \nu_\mathrm{s}/2 = 22.05$ kHz. Such fine sampling prevents aliasing in the audible part of the spectrum and there is no distortion of the signal. Had we wished, however, to compress the signal, we should

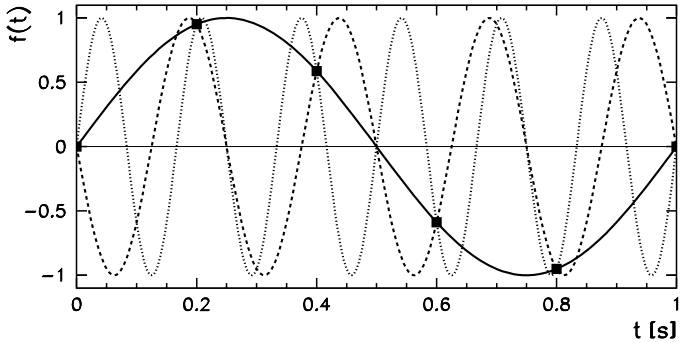


Fig. 4.4 A periodic signal $\sin(2\pi\nu t)$ with the frequency $\nu = 1\,\mathrm{s}^{-1}$ (*full line*) is sampled at $\nu_\mathrm{s} = 5\,\mathrm{s}^{-1}$ (*squares*): the critical frequency (4.6) is $\nu_\mathrm{c} = \omega_\mathrm{c}/(2\pi) = 2.5\ \mathrm{s}^{-1}$. We obtain exactly the same function values by sampling the functions $\sin(2\pi(\nu - \nu_\mathrm{s})t)$ (*dashed line*) or $\sin(2\pi(\nu + \nu_\mathrm{s})t)$ (*dotted line*). At this sampling rate, aliasing causes all three signals to be represented with the frequency ν in the discrete Fourier spectrum

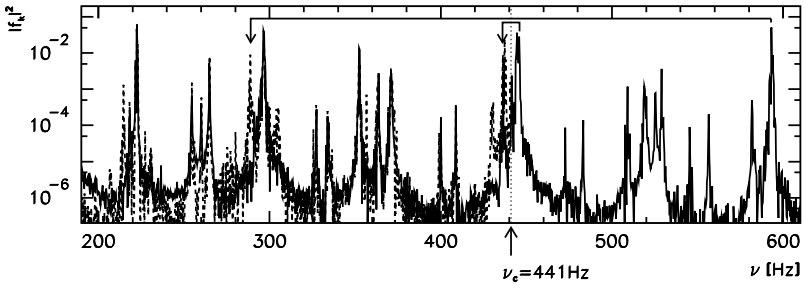


Fig. 4.5 Frequency spectrum of the concluding chord of the Toccata and Fugue for organ in d-minor, BWV 565, of J.S. Bach. *Full curve*: sampling at 44.1 kHz. *Dashed line*: sampling at a smaller frequency 882 Hz causes aliasing. The *arrows* indicate the peak at 593 Hz, which is mirrored across the critical frequency $\nu_\mathrm{c} = 441$ Hz onto the frequency $(441 - (593 - 441))$ Hz $= 289$ Hz, and the peak at 446 Hz, which maps onto 436 Hz

not do this by simply decreasing the sampling frequency, since this would map the high-frequency components into the low-frequency part of the spectrum and the signal would become distorted. Instead, we should use a filter to remove the high-frequency part of the spectrum and only then down-sample. Figure 4.5 shows the appearance of aliasing in the frequency spectrum of an acoustic signal sampled at 44.1 kHz and 882 Hz without filtering.

4.2.4 Leakage

The discrete Fourier transformation of realistic signals of course involves only finitely many values. From an infinite sequence we pick (multiply by one) only a

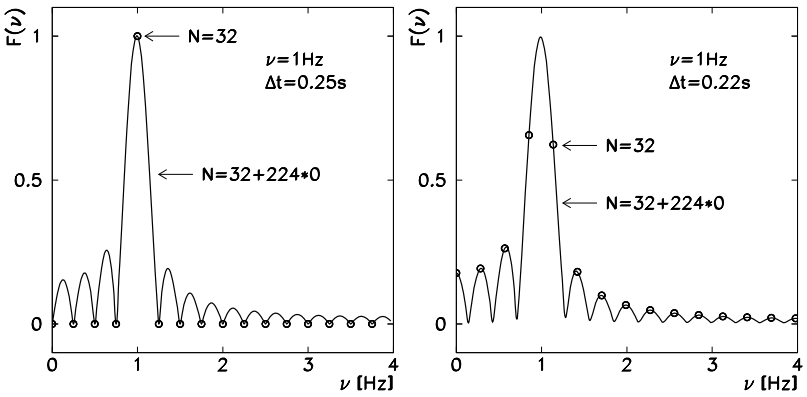


Fig. 4.6 Leakage in the frequency spectrum in the discrete Fourier transform of a sine wave with the frequency 1 Hz. [*Left*] Sampling at $N = 32$ points 0.25 s apart encompasses precisely four complete waves. The only non-zero component of the transform is the one corresponding to the frequency of 1 Hz. [*Right*] Sampling at $N = 32$ points 0.22 s apart covers only 3.52 waves. Many non-zero-frequency components appear. The *curves* connect the discrete transform of the same signals, except that the 32 original samples of the signal are followed by 224 zeros (total of 256 points). Adding zeros in the temporal domain is known as *zero padding* and improves the resolution in the frequency domain. In the limit $N \to \infty$ we approach the continuous Fourier transform

sample of length N, whereas the remaining values are dropped (multiplied by zero). Due to this restriction, known as *windowing*, the frequency spectrum exhibits the *leakage* phenomenon shown in Fig. 4.6 (adapted from [6]). To some extent, leakage can be controlled by using more sophisticated *window functions* that engage a larger portion of the signal and smoothly fade out instead of crude multiplication of the signal by one and the remainder by zero. The advantages and weaknesses of some classical window functions are discussed in [7].

4.2.5 Fast Discrete Fourier Transformation (FFT)

The discrete Fourier transformation (4.11) is a mapping between the vector spaces of dimensions N, $\mathcal{F}_N : \mathbb{C}^N \to \mathbb{C}^N$. Let us rewrite it in a more transparent form,

$$F = \mathcal{F}_N[f], \quad F_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j \, \mathrm{e}^{-2\pi \mathrm{i} jk/N}, \tag{4.18}$$

where we have denoted $f = \{f_j\}_{j=0}^{N-1}$ and $F = \{F_k\}_{k=0}^{N-1}$. Note that the indices j and k run symmetrically, both from 0 to $N - 1$. The inverse transformation is

$$f = \mathcal{F}_N^{-1}[F], \quad f_j = \sum_{k=0}^{N-1} F_k \, \mathrm{e}^{2\pi \mathrm{i} jk/N}.$$

Then (4.18) can be written as

$$F_k = \frac{1}{N} \sum_{j=0}^{N-1} W_N^{kj} f_j, \quad W_N = \mathrm{e}^{-2\pi \mathrm{i}/N}. \tag{4.19}$$

To evaluate the DFT by computing this sum we need $\mathcal{O}(N^2)$ operations. But precisely the same result can be achieved with far fewer operations by using the Cooley–Tukey algorithm. Let N be divisible by m. Then the sum can be split into m partial sums, and each of them runs over the elements f_j of the array f with the same modulus of the index j mod m:

$$F_k = \frac{1}{N} \sum_{l=0}^{m-1} W_N^{kl} \sum_{j=0}^{N/m-1} W_{N/m}^{kj} f_{mj+l}.$$

Let us denote by $f^{(l)} = \{f_{mj+l}\}_{j=0}^{N/m-1}$ the components of the array f which have the same modulus of the index with respect to m. We have thus recast the transform of the original array f of dimension N as a sum of m transforms of the shorter arrays $f^{(l)}$ of dimension N/m. This can be written symbolically as

$$\mathcal{F}_N[f]_k = \frac{1}{N} \sum_{l=0}^{m-1} W_N^{kl} \left(\frac{N}{m} \mathcal{F}_{N/m}\left[f^{(l)}\right] \right)_k.$$

This is a recursive computation of the DFT for the array f that follows the idea of *divide-and-conquer* algorithms. The array f for which the DFT should be computed is gradually broken down into sub-arrays, thus reducing the amount of necessary work. The optimal factorization is $N = 2^p$ ($p \in \mathbb{N}$) in which at each step the array is split into two sub-arrays containing elements of the original array with even and odd indices, respectively. This method requires $\mathcal{O}(N \log_2 N)$ operations for the full DFT instead of $\mathcal{O}(N^2)$ by direct summation, lending it the name *Fast Fourier Transformation*. Similar speeds are attainable by factorizing N to primes, e.g.

$$N = 2^{p_1} 3^{p_2} 5^{p_3} 7^{p_4}, \quad p_i \in \mathbb{N},$$

which is supported by all modern FFT libraries. Good implementations of the FFT are complicated, as the factorization should be carefully matched to the addressing of the arrays. The most famous library is the multiple-award winning FFTW (Fastest Fourier Transform in the West) [8]; see also [9, 10]. A comparison of the CPU cost of the standard DFT and FFT is illustrated in Fig. 4.7 (left).

Example Due to the fewer operations, FFT is not only essentially faster than the naive DFT; it is also more precise, as can be confirmed by a numerical experiment. We form the array $f = \{f_0, f_1, \ldots, f_{N-1}\}$ of random complex numbers. We apply the DFT to compute the transform of f, to which we apply the inverse DFT. Finally,

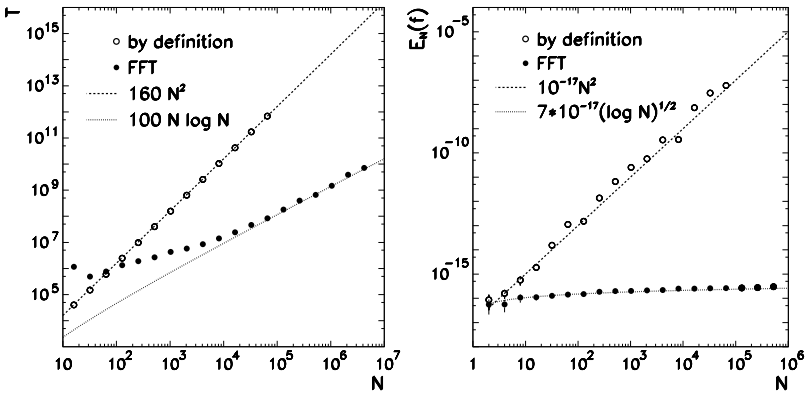


Fig. 4.7 [*Left*] The numerical cost (number of CPU cycles T) of the computation of the DFT by the basic definition (4.19) and by using the FFT, as a function of the sample size N. [*Right*] The average measure of deviation $E_N(h)$ for the computation of the DFT by definition and by using the FFT

we compute the deviation of the resulting array from the original array:

$$\Delta f = \left(\mathcal{F}_N^{-1} \circ \mathcal{F}_N\right) f - f.$$

In arithmetic with precision ε, we get $\Delta f \neq 0$. We define the average deviation as $E_N(f) = \langle \|\Delta f\|_2 / \|f\|_2 \rangle$, where the average $\langle\cdot\rangle$ is over a large set of random arrays. The results are shown in Fig. 4.7 (right). When the DFT is computed by (4.18), we get $E_N(f) \sim \mathcal{O}(\varepsilon N^2)$, while the FFT gives $E_N(f) \sim \mathcal{O}(\varepsilon\sqrt{\log N})$ [11]. In short, the FFT algorithm is unbeatable! All decent numerical libraries support the computation of the DFT by FFT algorithms (see Appendix I).

4.2.6 Multiplication of Polynomials by Using the FFT

Multiplication of polynomials is one of the tasks in computing with power bases, e.g. in expansions in powers of the perturbation parameters in classical and quantum mechanics. The multiplication of $p(x) = \sum_{i=0}^{n} a_i x^i$ and $q(x) = \sum_{i=0}^{m} b_i x^i$ in the form

$$q(x)p(x) = \sum_{i=0}^{n+m} c_i x^i, \quad c_i = \sum_{k=0}^{i} a_k b_{i-k},$$

requires $\mathcal{O}((n+1)(m+1))$ operations to determine the coefficients c_i. If the number of terms is large ($n, m \gg 1$) this process is slow and prone to rounding errors. A faster and a more precise way is offered by the FFT. We form two arrays of length $N = m + n + 1$. The coefficients of the polynomials p and q are stored at the begin-

ning of these arrays while the remaining elements are set to zero:

$$A = \{A_i\}_{i=0}^{N-1} = \{a_0, \ldots, a_n, \underbrace{0, \ldots, 0}_{m}\}, \qquad B = \{B_i\}_{i=0}^{N-1} = \{b_0, \ldots, b_m, \underbrace{0, \ldots, 0}_{n}\}.$$

The coefficients of the product are given by the convolution

$$c_i = \sum_{k=0}^{N-1} A_k B_{i-k}, \quad i = 0, 1, \ldots, N-1,$$

where we assume periodic boundary conditions, $A_k = A_{N+k}$, $B_k = B_{N+k}$. The convolution is then evaluated by first computing the FFT of the arrays A and B,

$$\hat{A} = \{\hat{A}_i\}_{i=0}^{N-1} = \mathcal{F}_N[A], \qquad \hat{B} = \{\hat{B}_i\}_{i=0}^{N-1} = \mathcal{F}_N[B],$$

multiplying the transforms component-wise into a new array $\hat{C} = \{\hat{A}_i \hat{B}_i\}_{i=0}^{N-1}$, and finally compute its inverse FFT,

$$C = \{C_i\}_{i=0}^{N-1} = N\mathcal{F}_N^{-1}[\hat{C}].$$

This procedure has a numerical cost of $\mathcal{O}(N \log_2 N)$ which, for $n, m \gg 1$, is much smaller than the cost of directly computing the sums of the products.

4.2.7 Power Spectral Density

The Fourier transformation can be seen as a decomposition of a signal to a linear combination of the functions $A_\omega e^{i\omega x}$. The quantity $|A_\omega|^2$ is the *signal power* at the given frequency ω. If we are dealing with real signals, we are mostly interested in the total power at the absolute value of the frequency, $|A_\omega|^2 + |A_{-\omega}|^2$ for $\omega \geq 0$.

For a continuous signal f with the Fourier transform (4.1), we define the *double-sided power spectral density* (PSD) as

$$S(\omega) = \big|F(\omega)\big|^2, \quad \omega \in \mathbb{R},$$

while the single-sided power spectral density is

$$S(\omega) = \big|F(-\omega)\big|^2 + \big|F(\omega)\big|^2, \quad \omega \in \mathbb{R}_+.$$

Often, the double-sided spectral density of a signal is defined via the single-sided density in which the power of a component with the frequency ω is equal to the power of the component with the frequency $-\omega$, and then $S(\omega) = 2|F(\omega)|^2$.

For discrete data $\{f_j\}$ with the transform (4.18) the discrete double-sided power spectral distribution $\{S_k\}$ is defined in analogy to the continuous case,

$$S_k = |F_k|^2, \quad k = 0, 1, \ldots, N-1.$$

The quantity S_k measures the power of the signal at the frequency $2\pi k/N$, while S_{N-k} corresponds to the frequency $-2\pi k/N$, where $k = 0, 1, \ldots, N/2 - 1$. For the single-sided distribution we sum over the powers of negative and positive frequencies. For odd N, the single-sided distribution $\{S_k\}$ is defined as

$$\begin{aligned} S_0 &= |F_0|^2, \\ 2S_k &= |F_k|^2 + |F_{N-k}|^2, \quad k = 1, 2, \ldots, (N-1)/2, \end{aligned}$$

while for even N it is given by

$$\begin{aligned} S_0 &= |F_0|^2, \\ 2S_k &= |F_k|^2 + |F_{N-k}|^2, \quad k = 1, 2, \ldots, N/2, \\ S_{N/2} &= |F_{N/2}|^2. \end{aligned}$$

In the discrete case, Parseval's equality applies in the form

$$\frac{1}{N} \sum_{j=0}^{N-1} |f_j|^2 = \sum_{k=0}^{N-1} |F_k|^2.$$

4.3 Transformations with Orthogonal Polynomials

Functions can also be expanded in terms of orthogonal polynomials [12]. Let us work in $\mathcal{P}_N$, the space of polynomials of degree $\leq N$, and restrict the discussion to Legendre and Chebyshev polynomials. The polynomials from these families are orthogonal on $[-1, 1]$ with respect to the weight function w. In the function space $L^2_w(-1, 1)$ we define the scalar product

$$\langle u, v \rangle_w = \int_{-1}^{1} u(x) v(x) w(x) \, \mathrm{d}x \tag{4.20}$$

and the norm $\|u\|_w = \sqrt{\langle u, u \rangle_w}$, where $w(x) = 1$ for Legendre and $w(x) = (1 - x^2)^{-1/2}$ for Chebyshev polynomials. Orthogonality means that

$$\int_{-1}^{1} p_i(x) p_j(x) w(x) \, \mathrm{d}x = \begin{cases} \text{const.}; & i = j, \\ 0; & i \neq j. \end{cases} \tag{4.21}$$

The expansion of the function u into an infinite series of orthogonal polynomials has the form

$$u(x) \equiv Su = \sum_{k=0}^{\infty} \widehat{u}_k p_k(x), \quad \widehat{u}_k = \frac{1}{\|p_k\|_w^2} \int_{-1}^{1} u(x) p_k(x) w(x) \, \mathrm{d}x.$$

Here $\widehat{u}_k$ are the expansion coefficients which may be understood as the transforms of u, by analogy to (4.8). For $N \in \mathbb{N}$ we have a finite series

$$S_N u \equiv \sum_{k=0}^{N} \widehat{u}_k p_k(x).$$

Relation to Quadrature Formulas Orthogonal polynomials are closely related to quadrature formulas. For any quadrature we need the *collocation points* or *nodes* $x_j \in [-1, 1]$ as well as the *quadrature weights* $w_j > 0$ for $j = 0, 1, \ldots, N$. With these quantities the integral of a continuous function can be approximated by the sum of the products of the weights and the function values at the nodes,

$$\int_{-1}^{1} f(x)w(x)\,\mathrm{d}x = \sum_{j=0}^{N} f(x_j)w_j + R_N. \tag{4.22}$$

The remainder R_N should be minimal or zero if f is a polynomial of a certain maximum degree. These degrees depend on the choice of the nodes.

In *Gauss quadrature* the nodes $x_0, x_1, \ldots, x_N$ are zeros of the orthogonal polynomial p_{N+1}, which lie in the interior of the interval $[-1, 1]$. The quadrature formula is then exact for polynomials of degree at most $2N + 1$. In *Gauss–Radau quadrature* the nodes are zeros of the polynomial $q(x) = p_{N+1}(x) + ap_N(x)$, where a is chosen such that $q(-1) = 0$. Then $x_0 = -1, x_1, x_2, \ldots, x_N$ are the zeros of q and the quadrature formula is exact for polynomials of degree at most $2N$. In *Gauss–Lobatto quadrature* we also include the point $x_N = 1$. This time we are seeking the $N + 1$ zeros of the polynomial $q(x) = p_{N+1}(x) + ap_N(x) + bp_{N-1}(x)$, where a and b are chosen such that $q(-1) = q(1) = 0$. The quadrature is then exact for polynomials of degree at most $2N - 1$. The weights w_j depend on the class of the orthogonal polynomials.

Discrete Transformation Following the notational conventions of [5], we define the discrete transformation with orthogonal polynomials as

$$u(x_j) = \sum_{k=0}^{N} \widetilde{u}_k p_k(x_j), \tag{4.23}$$

while its inverse is

$$\widetilde{u}_k = \frac{1}{\gamma_k} \sum_{j=0}^{N} u(x_j)p_k(x_j)w_j, \quad \gamma_k = \sum_{j=0}^{N} p_k^2(x_j)w_j. \tag{4.24}$$

Equations (4.23) and (4.24) for polynomial transforms are analogous to the discrete Fourier transforms with trigonometric polynomials.

In collocation methods (for example, for the solution of partial differential equations in Chap. 11) we can represent a smooth function u on the interval $[-1, 1]$ by its

discrete values at the collocation points, while the derivatives of u are approximated by the derivatives of its interpolation polynomial. The interpolation polynomial is an element of $\mathcal{P}_N$, and its values at the collocation points are equal to the function values, $I_N u(x_j) = u(x_j)$ for $0 \le j \le N$. It is formed by the sum

$$I_N u(x) = \sum_{k=0}^{N} \widetilde{u}_k p_k(x). \tag{4.25}$$

It can be shown—just like in Fourier expansions—that the interpolant $I_N u$ is the projection of u on the space $\mathcal{P}_N$ with respect to the scalar product

$$\langle u, v \rangle_N = \sum_{j=0}^{N} u(x_j) v(x_j) w_j.$$

This means that $\langle I_N u, v \rangle_N = \langle u, v \rangle_N$ for any continuous function v. Expression (4.23) can therefore also be read as $\gamma_k \widetilde{u}_k = \langle u, p_k \rangle_N$, and the orthogonality relation as $\langle p_j, p_k \rangle_N = \gamma_k \delta_{j,k}$ for $0 \le k \le N$. The relation between the discrete polynomial coefficients and the coefficients of the continuous expansion is

$$\widetilde{u}_k = \widehat{u}_k + \frac{1}{\gamma_k} \sum_{j>N} \langle p_j, p_k \rangle_N \widehat{u}_j. \tag{4.26}$$

In general $\langle p_j, p_k \rangle_N \neq 0$ for $j > N$, so the kth component of the interpolant $I_N u$ depends on the kth component of u and *all* components with indices $k > N$. We can again rewrite (4.26) in the form $I_N u = S_N u + R_N u$, where in

$$R_N u = I_N u - S_N u = \sum_{k=0}^{N} \left(\frac{1}{\gamma_k} \sum_{j>N} \langle p_j, p_k \rangle_N \widehat{u}_j \right) p_k \tag{4.27}$$

we again recognize the aliasing error (compare (4.26) and (4.27) to the corresponding (4.16) and (4.17) for the DFT). The aliasing error is orthogonal to the series truncation error $u - S_N u$, hence

$$\|u - I_N u\|_w^2 = \|u - S_N u\|_w^2 + \|R_N u\|_w^2.$$

4.3.1 Legendre Polynomials

Legendre polynomials P_k for $k = 0, 1, \ldots$ are the eigenfunctions of the singular Sturm–Liouville problem

$$\left(\left(1 - x^2\right) P_k'(x) \right)' + k(k+1) P_k(x) = 0, \quad x \in [-1, 1],$$

which is of the form (8.84) with $p(x) = 1 - x^2$, $q(x) = 0$, and $w(x) = 1$. The polynomials P_k for even (odd) k are even (odd). With the normalization $P_k(1) = 1$

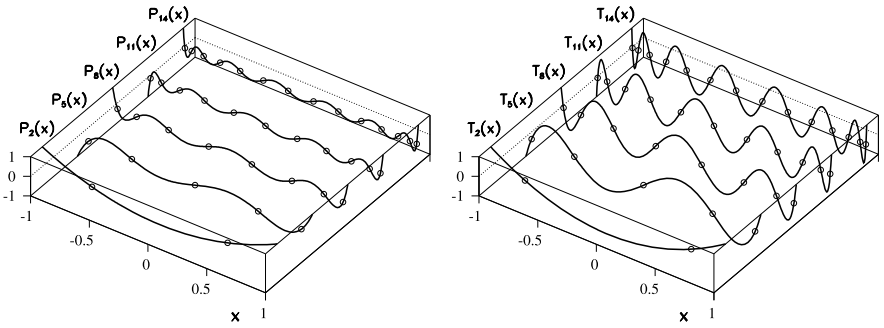


Fig. 4.8 Orthogonal polynomials used in the transformations of functions, collocation methods, and spectral methods for partial differential equations (Chap. 11). [*Left*] Legendre polynomials P_k. [*Right*] Chebyshev polynomials T_k. Shown are the polynomials of degree $k \in \{2, 5, 8, 11, 14\}$. There is a characteristic clustering of the zeros in the vicinity of the boundary points $x = -1$ and $x = 1$

they can be generally written as

$$P_k(x) = \frac{1}{2^k} \sum_{l=0}^{\lfloor k/2 \rfloor} (-1)^l \binom{k}{l} \binom{2k-2l}{k} x^{k-2l}, \tag{4.28}$$

where $\lfloor k/2 \rfloor$ is the integer part of $k/2$. Legendre polynomials possess a three-term recurrence relation

$$(k+1)P_{k+1}(x) = (2k+1)x\,P_k(x) - k\,P_{k-1}(x), \quad P_0(x) = 1,\ P_1(x) = x,$$

which is a great tool to actually compute the polynomial values. Some typical polynomials P_k are shown in Fig. 4.8 (left).

By using the polynomials P_k any function $u \in L^2(-1, 1)$ can be expanded in a series

$$u(x) = \sum_{k=0}^{\infty} \widehat{u}_k P_k(x), \quad \widehat{u}_k = \frac{2k+1}{2} \int_{-1}^{1} u(x) P_k(x)\,\mathrm{d}x. \tag{4.29}$$

The discrete expansion in terms of Legendre polynomials is defined by (4.23) and (4.24) by inserting $p_k(x) = P_k(x)$, while the collocation points x_j, the weights w_j, and the normalization factors γ_k depend on the type of the quadrature (4.22). For quadrature with Legendre polynomials there are no explicit formulas for the collocation points x_j; we must compute the zeros of the corresponding polynomials.

Gauss For Legendre–Gauss quadrature the collocation points x_j ($j = 0, 1, \ldots, N$) are the roots of the equation $P_{N+1}(x_j) = 0$, while the corresponding weights and normalization factors are

$$w_j = \frac{2}{(1-x_j^2)[P'_{N+1}(x_j)]^2}, \qquad \gamma_k = \frac{2}{2k+1}. \tag{4.30}$$

Gauss–Radau The collocation points for Legendre–Gauss–Radau quadrature are the roots of the equation $P_N(x_j) + P_{N+1}(x_j) = 0$, and the weights are

$$w_0 = \frac{2}{(N+1)^2}, \qquad w_j = \frac{1 - x_j}{(N+1)^2[P_N(x_j)]^2}, \quad j = 1, 2, \ldots, N. \tag{4.31}$$

The normalization factors are $\gamma_k = 2/(2k+1)$.

Gauss–Lobatto The most frequently used Legendre–Gauss–Lobatto quadrature explicitly includes both endpoints of the interval, $x_0 = -1$ and $x_N = 1$, while the interior points x_j are the roots of the equation $P'_N(x_j) = 0$:

$$x_j = \begin{cases} -1; & j = 0, \\ \text{solutions of } P'_N(x_j) = 0; & 1 \le j \le N-1, \\ 1; & j = N. \end{cases} \tag{4.32}$$

In this case the weights are

$$w_j = \frac{2}{N(N+1)} \frac{1}{[P_N(x_j)]^2}, \quad j = 0, 1, \ldots, N. \tag{4.33}$$

The normalization factors are $\gamma_k = 2/(2k+1)$ for $k < N$ and $\gamma_N = 2/N$.

Example Let us determine the discrete Legendre–Gauss expansion of the function

$$u(x) = x \cos\bigl((\pi x)^2\bigr)$$

by using ten points ($N = 9$). The collocation points x_j are the roots of the equation $P_{N+1}(x_j) = 0$ (all values rounded to four digits):

$$-x_0 = x_9 = 0.9739, \qquad -x_1 = x_8 = 0.8651, \qquad -x_2 = x_7 = 0.6794,$$
$$-x_3 = x_6 = 0.4334, \qquad -x_4 = x_5 = 0.1489.$$

The weights w_j are given by (4.30):

$$w_0 = w_9 = 0.0667, \qquad w_1 = w_8 = 0.1495, \qquad w_2 = w_7 = 0.2191,$$
$$w_3 = w_6 = 0.2693, \qquad w_4 = w_5 = 0.2955.$$

The coefficients (4.24) of the discrete expansion (4.23) are then

$$\widetilde{u}_1 = -0.1084, \qquad \widetilde{u}_3 = -0.1784, \qquad \widetilde{u}_5 = -0.4807,$$
$$\widetilde{u}_7 = -1.0136, \qquad \widetilde{u}_9 = -0.0923,$$

while $\widetilde{u}_0 = \widetilde{u}_2 = \widetilde{u}_4 = \widetilde{u}_6 = \widetilde{u}_8 = 0$: since the function u is odd, its expansion also involves only odd Legendre polynomials. Figure 4.9 (left) shows the points x_j, the function u, and its interpolant $I_N u$ (4.25).

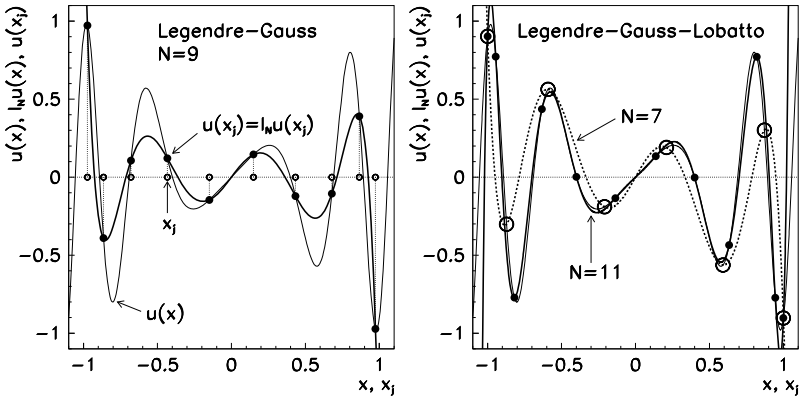


Fig. 4.9 Interpolation polynomials corresponding to the transformation of functions by Legendre polynomials. [*Left*] Computation at the Legendre–Gauss nodes with $N = 9$. [*Right*] Computation at the Legendre–Gauss nodes with $N = 7$ and $N = 11$

We follow the same path in Legendre–Gauss–Lobatto collocation, where the endpoints $-x_0 = x_N = 1$ are always included. Let us take eight points ($N = 7$). The nodes are given by (4.32),

$$-x_0 = x_7 = 1.0000, \qquad -x_1 = x_6 = 0.8717,$$
$$-x_2 = x_5 = 0.5917, \qquad -x_3 = x_4 = 0.2093,$$

while the weights are given by (4.33),

$$-w_0 = w_7 = 0.0357, \qquad -w_1 = w_6 = 0.2107,$$
$$-w_2 = w_5 = 0.3411, \qquad -w_3 = w_4 = 0.4125.$$

The seventh-degree interpolation polynomial $I_7 u$ of u is shown in Fig. 4.9 (right). The same figure also shows the eleventh-degree interpolant $I_{11} u$.

In the Gauss–Lobatto case, the polynomial $I_N u$ that interpolates u and matches it at the points x_j can be written in two ways: as an expansion in $P_k(x)$ with the coefficients $\widetilde{u}_k$, or in the form of the expansion in Lagrange interpolation polynomials where the coefficients are given by the values of u at x_j:

$$I_N u(x) = \sum_{k=0}^{N} \widetilde{u}_k P_k(x) = \sum_{j=0}^{N} u(x_j) l_j(x), \tag{4.34}$$

where

$$l_j(x) = \frac{1}{N(N+1)} \frac{x^2 - 1}{x - x_j} \frac{P'_N(x)}{P_N(x_j)} \tag{4.35}$$

is the Lagrange interpolation polynomial. (At $x = x_j$ both its numerator and denominator are zero and l'Hôspital is called to rescue.) In the discrete Fourier transformation we have expressed the interpolant $I_N u$ by the sum (4.14) over trigonometric polynomials (4.15). Expressions (4.34) and (4.35) are their equivalents for the discrete Legendre transformation.

4.3.2 Chebyshev Polynomials

Chebyshev polynomials T_k for $k = 0, 1, \ldots$ [13] are the eigenfunctions of the singular Sturm–Liouville problem

$$\left(\sqrt{1-x^2}\,T_k'(x)\right)' + \frac{k^2}{\sqrt{1-x^2}}\,T_k(x) = 0, \quad x \in [-1, 1],$$

which is of the form (8.84) with $p(x) = (1-x^2)^{1/2}$, $q(x) = 0$, and $w(x) = (1-x^2)^{-1/2}$. The polynomials T_k for even (odd) k are even (odd). With the normalization $T_k(1) = 1$ they are defined as

$$T_k(x) = \cos(k \arccos x) = \sum_{i=0}^{\lfloor k/2 \rfloor} \binom{k}{2i} \left(x^2 - 1\right)^i x^{k-2i}. \tag{4.36}$$

Some typical examples are shown in Fig. 4.8 (right). The polynomials are related by the three-term recurrence

$$T_{k+1}(x) = 2x T_k(x) - T_{k-1}(x), \quad T_0(x) = 1,\ T_1(x) = x, \tag{4.37}$$

which allows us to compute the values $T_n(x)$ in just $\mathcal{O}(n)$ operations. Chebyshev polynomials are orthogonal on the interval $[-1, 1]$ with respect to the weight $(1-x^2)^{-1/2}$,

$$\int_{-1}^{1} T_k(x) T_l(x) \frac{\mathrm{d}x}{\sqrt{1-x^2}} = \begin{cases} 0; & k \neq l, \\ \pi/2; & k = l \neq 0, \\ \pi; & k = l = 0, \end{cases}$$

but they also possess the peculiar property of *orthogonality by points*: on the discrete set of points

$$x_j = \cos \frac{(2j+1)\pi}{2N}, \quad j = 0, 1, \ldots, N-1,$$

corresponding to the N zeros of the polynomial T_N, Chebyshev polynomials are orthogonal in the sense of the sum

$$\sum_{j=0}^{N-1} T_k(x_j) T_l(x_j) = \begin{cases} 0; & k \neq l, \\ N/2; & k = l \neq 0, \\ N; & k = l = 0. \end{cases} \tag{4.38}$$

(More general formulas can be found in (1.141) and (1.144) in [13].) Any function $u \in L^2(-1, 1)$ can be expanded in terms of Chebyshev polynomials T_k as

$$u(x) = \sum_{k=0}^{\infty} \widehat{u}_k T_k(x), \quad \widehat{u}_k = \frac{2}{\pi c_k} \int_{-1}^{1} u(x) T_k(x) w(x)\, \mathrm{d}x, \tag{4.39}$$

where $c_0 = 2$ and $c_k = 1$ for $k \geq 1$. In contrast to quadrature involving Legendre polynomials, the nodes and the weights for Chebyshev quadrature are given by explicit formulas which are given in the following. Note that the nodes are indexed such that the values of x_j decrease when j increases.

Gauss For Chebyshev–Gauss collocation the nodes and the weights are

$$x_j = \cos \frac{(2j+1)\pi}{2N+2}, \qquad w_j = \frac{\pi}{N+1}, \quad j = 0, 1, \ldots, N,$$

while the factors γ_k in (4.24) are equal to $\gamma_k = \pi c_k/2$ for $0 \leq k < N$ and $\gamma_N = \pi/2$.

Gauss–Radau For Chebyshev–Gauss–Radau collocation we have

$$x_j = \cos \frac{2j\pi}{2N+1}, \qquad w_j = \begin{cases} \frac{\pi}{2N+1}; & j = 0, \\ \frac{2\pi}{2N+2}; & j = 1, 2, \ldots, N, \end{cases}$$

while $\gamma_k = \pi c_k/2$ for $0 \leq k < N$ and $\gamma_N = \pi/2$.

Gauss–Lobatto The most frequently used Gauss–Lobatto collocation includes the endpoints $x_0 = 1$ and $x_N = -1$. Here, the nodes and the weights are given by

$$x_j = \cos \frac{j\pi}{N}, \qquad w_j = \begin{cases} \frac{\pi}{2N}; & j = 0, N, \\ \frac{\pi}{N}; & j = 1, 2, \ldots, N-1. \end{cases} \tag{4.40}$$

The normalization factors are $\gamma_k = \pi c_k/2$ for $0 \leq k < N$ and $\gamma_N = \pi$.

Discrete Chebyshev transformation on the interval $x \in [-1, 1]$ is most often associated with Gauss–Lobatto collocation, and this is the only case we discuss henceforth. The expansion of the function u in a finite series has the form

$$I_N u(x) = \sum_{k=0}^{N} \widetilde{u}_k T_k(x), \quad \widetilde{u}_k = \frac{2}{N \overline{c}_k} \sum_{j=0}^{N} \frac{1}{\overline{c}_j} u(x_j) T_k(x_j), \tag{4.41}$$

where $\overline{c}_0 = \overline{c}_N = 2$ and $\overline{c}_j = 1$ for $j = 1, 2, \ldots, N-1$. This is a discrete equivalent of the continuous expansion defined in (4.39). At the Gauss–Lobatto quadrature nodes (4.40) the expansion can be rewritten as

$$I_N u(x_j) = \sum_{k=0}^{N} \widetilde{u}_k \cos \frac{\pi j k}{N}, \quad \widetilde{u}_k = \frac{2}{N \overline{c}_k} \sum_{j=0}^{N} \frac{1}{\overline{c}_j} u(x_j) \cos \frac{\pi j k}{N},$$

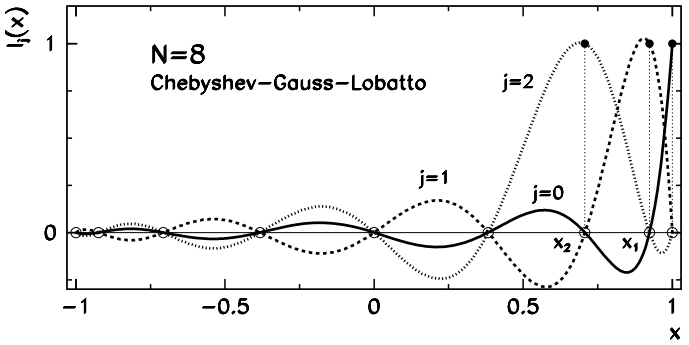


Fig. 4.10 The first three Lagrange interpolation polynomials $l_j(x)$ for Chebyshev–Gauss–Lobatto collocation with $N = 8$. The collocation points $x_j = \cos(\pi j/N)$, at which $l_j(x_k) = \delta_{j,k}$ holds, follow from right to left for indices $j = 0, 1, \ldots, N$

which is nothing but the cosine transform. The transliteration of the Chebyshev transformation to the Fourier transformation makes a great impact in spectral methods (Sect. 11.1.3).

Example As in the Legendre case (4.34), the transformation with Chebyshev polynomials allows us to write the interpolation polynomial in two ways: by using (4.41) or by expanding it in terms of different interpolation polynomials,

$$I_N u(x) = \sum_{k=0}^{N} \widetilde{u}_k T_k(x) = \sum_{j=0}^{N} u(x_j) l_j(x),$$

where

$$l_j(x) = \frac{1}{\overline{c}_j} \frac{(-1)^j}{N^2} \frac{x^2 - 1}{x - x_j} T'_N(x) \tag{4.42}$$

is the interpolation polynomial for Gauss–Lobatto collocation (4.40). The polynomials l_0, l_1, and l_2 for $N = 8$ are shown in Fig. 4.10. A correct choice of the collocation points is of key importance. Namely, on the interval $[-1, 1]$, a different Lagrange interpolation polynomial with the general form

$$l_j(x) = \frac{q_N(x)}{(x - x_j) q'_N(x_j)}, \qquad q_N(x) = \prod_{j=0}^{N} (x - x_j), \tag{4.43}$$

can be spanned through any set of function values at the $N + 1$ nodes. But for the chosen class of orthogonal polynomials, only one set of nodes is optimal: for Chebyshev polynomials with Gauss–Lobatto collocation these are precisely the points (4.40). Figure 4.11 shows the classic comparison between the interpolation of the function at equidistant points according to (4.43) and the interpolation at the Gauss–Lobatto points (4.40), which prevents the wild oscillations of the interpolant.

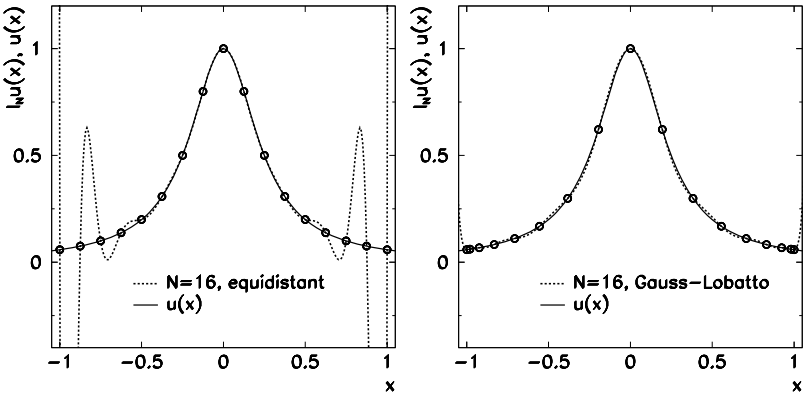


Fig. 4.11 Lagrange interpolation of the function $u(x) = 1/(1 + 16x^2)$ on the interval $[-1, 1]$. [*Left*] The interpolant $I_{16}u$ through 17 equidistant points $x_j = 2j/N - 1$ ($j = 0, 1, \ldots, N$). When the number of points N is increased, the error $\|u - I_N u\|$ exponentially increases, the annoyance known as the Runge phenomenon. [*Right*] The interpolant $I_{16}u$ through 17 Gauss–Lobatto points (4.40). By increasing N the error $\|u - I_N u\|$ exponentially decreases, which can be exploited in spectral methods (Chap. 11)

Table 4.1 Laplace transforms of some common functions. The transforms of the products $\mathcal{L}[\theta(t-c)f(t-c)] = \mathrm{e}^{-cs}F(s)$, $\mathcal{L}[\mathrm{e}^{ct}f(t)] = F(s-c)$, and $\mathcal{L}[(-t)^n f(t)] = F^{(n)}(s)$ are also very useful. For a much more complete list see Sect. 29.3 in [16]

$f(t) = \mathcal{L}^{-1}[F]$	$F(s) = \mathcal{L}[f]$	Assumptions
t^p	$\Gamma(p+1)/s^{p+1}$	$p > -1$, $\mathrm{Re}\{s\} > 0$
e^{at}	$1/(s-a)$	$\mathrm{Re}\{s\} > a$
$\sin at$	$a/(s^2+a^2)$	$\mathrm{Re}\{s\} > 0$
$\cos at$	$s/(s^2+a^2)$	$\mathrm{Re}\{s\} > 0$
$\delta(t-c)$	e^{-cs}	$\mathrm{Re}\{s\} > 0$
$\theta(t-c)$	e^{-cs}/s	$\mathrm{Re}\{s\} > 0$

4.4 Laplace Transformation

The continuous Laplace transformation of the function f is defined as

$$F(s) = \mathcal{L}[f](s) = \int_0^\infty \mathrm{e}^{-st} f(t)\,\mathrm{d}t, \quad s \in \mathbb{C}.$$

The sufficient conditions for the existence of the transform $\mathcal{L}[f]$ are that f is piecewise continuous on $\mathbb{R}_+$ and that f is of exponential order in the limit $t \to \infty$: this means that real constants $C > 0$, a, and $T > 0$ exist such that $|f(t)| \le C\mathrm{e}^{at}$ for $\forall t > T$. The transformation is linear, $\mathcal{L}[c_1 f_1 + c_2 f_2] = c_1\mathcal{L}[f_1] + c_2\mathcal{L}[f_2]$. The transforms of some typical functions are enumerated in Table 4.1. The algorithms for the fast discrete Laplace transformation are described in [14, 15].

Laplace transforms of real functions for which we do not know the suitable elementary integral, can be computed by using the Gauss–Laguerre quadrature of high

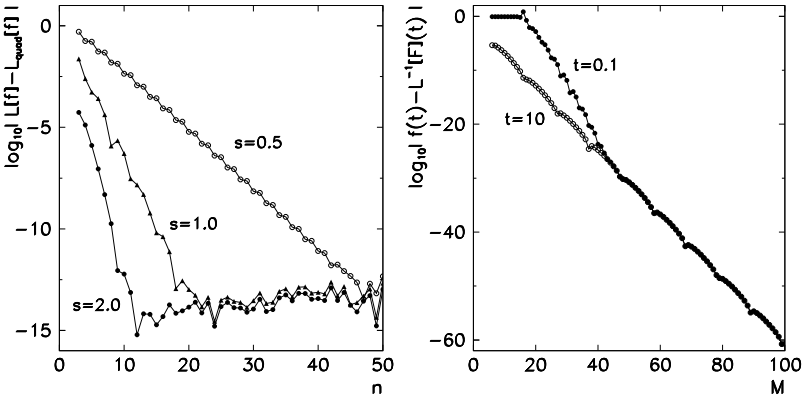


Fig. 4.12 [*Left*] The precision of the quadrature formula (4.44) for the Laplace transformation of the function $f(t) = \cos t$ (the exact transform is $F(s) = s/(s^2 + 1)$). [*Right*] The precision of the FT algorithm from [19] to compute the inverse Laplace transform $F(s) = s/(s^2 + 1)$ to arbitrary precision (the exact solution is $f(t) = \cos t$). Shown is the error $\log_{10} |f(t) - \mathcal{L}^{-1}[F](t)|$ as a function of the number of terms in the quadrature sum for two different t

order (see Fig. 4.12 (left)). Assuming that f can be sufficiently well described by a polynomial, the transform with $s > 0$ can be computed as

$$\mathcal{L}[f](s) = \frac{1}{s} \sum_{i=1}^{n} w_i \, f(x_i/s) + R_n, \quad R_n = \frac{(n!)^2}{(2n)!} f^{(2n)}(\xi), \tag{4.44}$$

where $\{x_i\}_{i=1}^n$ are the zeros of the Laguerre polynomial $L_n(x)$, $\{w_i\}_{i=1}^n$ are the quadrature weights, and $\xi \in \mathbb{R}$. From the zeros x_i we compute the weights as

$$w_i = \frac{x_i}{[(n+1)L_{n+1}(x_i)]^2}.$$

Many modern numerical libraries include the generators of zeros and weights, and they are mostly based on the algorithms presented in [17]. This approach allows for a stable and precise calculation of the zeros and weights in double precision up to order $n \approx 40$ (up to $n = 15$ they are listed on p. 923 of [16]). The quadrature (4.44) at small values of s fails, in particular with oscillatory functions f, because the value $f(x_i/s)$ changes too quickly; in such cases it is preferable to use the formula

$$\mathcal{L}[f](s) \approx \sum_{i=1}^{n} w_i \, \mathrm{e}^{(1-s)x_i} f(x_i).$$

4.4.1 Use of Laplace Transformation with Differential Equations

One of the most fruitful application areas of the Laplace transformation is the study of electric circuits and mechanical systems where we wish to understand the solu-

tions of linear differential equations with discontinuous or impulse forcing terms. The generic example is the equation $\ddot{x} + \beta\dot{x} + \omega_0^2 x = f(t)$ with the Heaviside (step) function $f(t) = \theta(t)$ or with the impulse $f(t) = \delta(t)$. The kernel of the Laplace transformation e^{-st} determines the natural scale of the physical process.

Laplace transformation draws its true strength from the relations between the transforms of the function f itself and the transforms of its derivatives. For a piecewise continuous f' we have

$$\mathcal{L}\big[f'\big] = s\mathcal{L}[f] - f(0)$$

or, with similar assumptions for the higher derivatives [18],

$$\mathcal{L}\big[f^{(n)}\big] = s^n\mathcal{L}[f] - s^{n-1}f(0) - \cdots - sf^{(n-2)}(0) - f^{(n-1)}(0). \tag{4.45}$$

By using the relation (4.45) in the initial-value problem with constant coefficients

$$a\ddot{x} + b\dot{x} + cx = f(t),$$

with given initial conditions $x(0)$ and $\dot{x}(0)$, we obtain

$$a\big[s^2X(s) - sx(0) - \dot{x}(0)\big] + b\big[sX(s) - x(0)\big] + cX(s) = F(s). \tag{4.46}$$

Instead of solving the differential equation for $x(t)$ we have succeeded in rephrasing the problem in terms of an algebraic equation for $X(s)$ into which the initial conditions have already been "built in". From the function $X(s)$ we then obtain the solution $x(t)$ by computing the inverse Laplace transform.

Formally, the inverse Laplace transform is given by the formula

$$\mathcal{L}^{-1}[F](t) = \frac{1}{2\pi\,\mathrm{i}}\int_C \mathrm{e}^{st}F(s)\,\mathrm{d}s, \tag{4.47}$$

where C is a vertical line in the complex plane, defined by $C = \xi + \mathrm{i}\eta$, and ξ is chosen such that all singularities of the transform $F(s)$ lie to the left of C. In other words, F is analytic on the half-plane $\mathrm{Re}\{s\} > \xi$. The sufficient conditions for the existence of the inverse are

$$\lim_{s\to\infty} F(s) = 0, \qquad \lim_{s\to\infty} \big|sF(s)\big| < \infty.$$

The inverse Laplace transformation is linear.

The world of the inverse Laplace transformation is not rosy: the expression for $X(s)$, which we read off from (4.46), first has to be reshuffled such that in its various parts of the functions $F(s)$ from Table 4.1 are identified, and these are then associated with the corresponding parts of the solution $x(t)$. For example,

$$X(s) = \frac{s-1}{s^2 - s - 2} = \frac{1}{3}\frac{1}{(s-2)} + \frac{2}{3}\frac{1}{(s+1)},$$

which corresponds to

$$x(t) = \frac{1}{3}\,\mathrm{e}^{2t} + \frac{2}{3}\,\mathrm{e}^{-t}$$

(see the second row of Table 4.1). This was an easy example, as the available set of pairs $F(s)$ and $f(t)$ for which the table of known elementary functions and their transforms is read from right to left, is quickly exhausted. The inverse transform then needs to be computed numerically (see Problem 4.7.3).

The numerical computation of the inverse Laplace transform is described in an immense body of papers. Excellent insight is offered by [20], [21, 22], and [23]. One of the best approaches is a deformation of the curve C in the integral (4.47) such that the integral converges as quickly as possible, and the use of the corresponding quadrature formulas [24, 25]. Another good way is to rewrite the integral as a Fourier series which can be computed by the FFT algorithm [26–29]. A very robust and simple algorithm for the inverse transformation at arbitrary arithmetic precision can be found in [19]; see also the example in Fig. 4.12 (right).

4.5 Hilbert Transformation ⋆

The Hilbert transformation $\mathcal{H}$ of the function $s : \mathbb{R} \to \mathbb{R}$ is defined as the convolution of the function s with the function $h(t) = 1/(\pi t)$:

$$\hat{s}(t) = \mathcal{H}[s](t) = -(s * h)(t) = \frac{1}{\pi} P \int_{-\infty}^{\infty} \frac{s(\tau)}{\tau - t}\,\mathrm{d}\tau, \tag{4.48}$$

where $*$ denotes the convolution. The integral should be understood as the Cauchy principal value (symbol P). The principal value integral of the function f over the interval $[a, b]$, on which f has a singularity at $c \in [a, b]$, is defined as

$$P \int_a^b f(t)\,\mathrm{d}t = \lim_{\varepsilon \searrow 0} \left(\int_a^{c-\varepsilon} f(t)\,\mathrm{d}t + \int_{c+\varepsilon}^b f(t)\,\mathrm{d}t \right).$$

The value of the transform does not change if an arbitrary constant is added to the function, $\mathcal{H}[s + \mathrm{const}] = \mathcal{H}[s]$. The transform (4.48) of the function $s \in L^p(\mathbb{R})$ for $1 < p < \infty$ exists only in the case that s tends to zero at positive and negative infinity [30], $\lim_{t \to \pm\infty} s(t) = 0$. The Hilbert transformation is linear and bounded [31]:

$$\left\| \mathcal{H}[s] \right\|_p \le C_p \|s\|_p, \quad C_p = \begin{cases} \tan(\pi/2p); & p = 1, 2, \\ \tan(\pi/2p)^{-1}; & p > 2. \end{cases}$$

The inverse Hilbert transformation is

$$s(t) = \mathcal{H}^{-1}[\hat{s}](t) = (\hat{s} * h)(t) = -\frac{1}{\pi} P \int_{-\infty}^{\infty} \frac{\hat{s}(\tau)}{\tau - t}\,\mathrm{d}\tau,$$

hence $\mathcal{H}^{-1} = -\mathcal{H}$ and $\mathcal{H}^2 = -\mathrm{id}$.

Relation to the Fourier Transform The Fourier transforms (4.1) of the signal s, $S(\omega) = \mathcal{F}[s](\omega)$, and of the function $h(t) = 1/(\pi t)$,

$$H(\omega) = \mathcal{F}[h](\omega) = -\mathrm{i}\,\mathrm{sign}(\omega), \qquad \mathrm{sign}(\omega) = \begin{cases} 1; & \omega > 0, \\ 0; & \omega = 0, \\ -1; & \omega < 0, \end{cases}$$

are related to the Hilbert transform of s by

$$\hat{s} = -\mathcal{F}^{-1}[SH], \tag{4.49}$$

where $\mathcal{F}^{-1}$ is the inverse Fourier transformation (4.2). The norm of the function $s \in L^2(\mathbb{R})$ with the Fourier transform $S \in L^2(\mathbb{R})$ is the same in all three representations,

$$\int_{-\infty}^{\infty} \big|\mathcal{H}[s](t)\big|^2 \,\mathrm{d}t = \frac{1}{2\pi} \int_{-\infty}^{\infty} \big|\mathrm{sign}(\omega) S(\omega)\big|^2 \,\mathrm{d}\omega = \lim_{\varepsilon \searrow 0} \left(\int_{-\infty}^{-\varepsilon} + \int_{\varepsilon}^{\infty} \right) \big|s(t)\big|^2 \,\mathrm{d}t,$$

where we have used (4.49) and Parseval's equality (4.3) for the Fourier transform. The Hilbert transformation therefore preserves the total power of the signal.

A real signal s and its Hilbert transform $\hat{s} = \mathcal{H}[s]$ are orthogonal, i.e.

$$\int_{-\infty}^{\infty} s(t)\hat{s}(t) \,\mathrm{d}t = \frac{\mathrm{i}}{2\pi} \int_{-\infty}^{\infty} \mathrm{sign}(\omega) \big|S(\omega)\big|^2 \,\mathrm{d}\omega = 0,$$

if s, $\hat{s}$, and the Fourier transform $S = \mathcal{F}[s]$ are in $L^1(\mathbb{R})$ or $L^2(\mathbb{R})$ (we assumed $S(\omega)^* = S(-\omega)$). A similar conclusion can be made by examining (4.49) for the Fourier modes, since for the functions $\cos \omega t$ and $\sin \omega t$, which are orthogonal for all $\omega \neq 0$, we have

$$\mathcal{H}[\cos \omega t] = -\mathrm{sign}(\omega) \sin \omega t, \qquad \mathcal{H}[\sin \omega t] = \mathrm{sign}(\omega) \cos \omega t. \tag{4.50}$$

Warning The numerical computation of the Hilbert transform of the function f on the whole real axis may be very problematic (see Fig. 4.13). Even small perturbations $\varepsilon(t)$ that are not in the space $L^p(\mathbb{R})$, $p > 1$ may cause a singularity in $\mathcal{H}[f + \varepsilon]$. The problems of this type are not unique to the Hilbert transformation, but because its kernel $h(t) = 1/(\pi t)$ is singular, the instabilities become more pronounced. We therefore often resort to simplifications; a few guidelines can be found in [32]. The Hilbert transform $\mathcal{H}[f](t)$ at large parameters t can be elegantly computed by its asymptotic expansion presented in [33] (in this case the asymptotics of f has to be known). We discuss the methods for numerical computation of the continuous Hilbert transform in Sect. 4.5.3, while the discrete transform is discussed in Sect. 4.5.4.

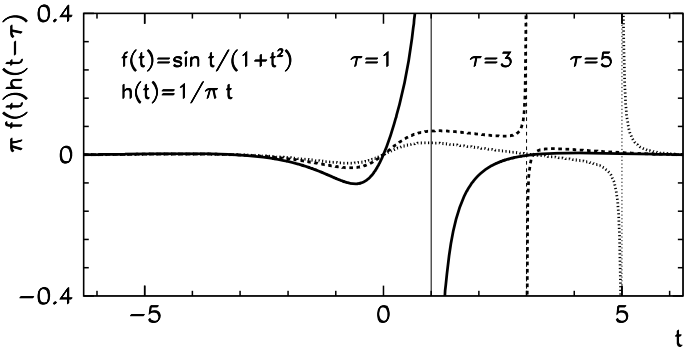


Fig. 4.13 The integrands for the computation of the Hilbert transform of $f(t) = \sin t/(1+t^2)$ for three values $\tau = 1$, $\tau = 3$, and $\tau = 5$. We integrate over the poles migrating along the real axis and "sample" f in a very sensitive manner. See also Sect. E.3

4.5.1 Analytic Signal

Hilbert transformation is a commonly used tool in the analysis of signals where it is used for their "complexification". This means that to a real signal (function) $s(t)$ is assigned a complex function

$$s^{\#}(t) = s(t) - \mathrm{i}\,\mathcal{H}[s](t), \tag{4.51}$$

known as the *analytic signal* (s and $\hat{s} = \mathcal{H}[s]$ are real functions). If the signal is $s(t) = \cos\omega t$, $\omega > 0$, the analytic signal is $s^{\#}(t) = \cos\omega t + \mathrm{i}\sin\omega t = \exp(\mathrm{i}\omega t)$, as can be inferred from (4.50). The Fourier transform of the analytic signal is

$$S^{\#}(\omega) = S(\omega)\big[1 + \mathrm{sign}(\omega)\big],$$

where $S(\omega) = \mathcal{F}[s](\omega)$ is the Fourier transform (4.1) of the signal s. The function $S^{\#}$ is zero for negative frequencies $\omega < 0$: this is a fundamental property of the analytic signals. If the function $s^{\#}$ is analytic on the upper complex half-plane and satisfies the Cauchy integral equation

$$P\int_{-\infty}^{\infty} \frac{s^{\#}(\xi)}{\xi - x}\,\mathrm{d}\xi = \mathrm{i}\,\pi s^{\#}(x), \tag{4.52}$$

we are referring to a *strongly analytic signal*. In addition to analyticity, the sufficient condition for the validity of the integral equation is that the function falls off quickly enough in the upper complex half-plane. If the signal $s^{\#}$ is strongly analytic, we may insert $s^{\#} = s - \mathrm{i}\,\hat{s}$ into (4.52) and obtain the relations between the real and imaginary parts of the analytic signal,

$$\mathcal{H}[s] = \hat{s}, \qquad \mathcal{H}[\hat{s}] = -s.$$

The analytic signal can be represented in the complex plane as

$$s^{\#}(t) = A(t)\,\mathrm{e}^{\mathrm{i}\phi(t)},$$

where

$$A(t) = \left|s^{\#}(t)\right| = \sqrt{\big(s(t)\big)^2 + \big(\mathcal{H}[s](t)\big)^2} \tag{4.53}$$

is the analytic amplitude, and

$$\phi(t) = \arg\big(s^{\#}(t)\big) = \operatorname{atan}\left(\frac{\operatorname{Im} s^{\#}(t)}{\operatorname{Re} s^{\#}(t)}\right) \tag{4.54}$$

is the analytic phase. In this representation we define various *instantaneous* quantities [34], like the *instantaneous signal power* $E_{\mathrm{i}}(t) = |A(t)|^2$ and the *instantaneous complex phase* $\Phi_{\mathrm{i}}(t) = \log s^{\#}(t) = \log A(t) + \mathrm{i}\phi(t)$. By using the time derivative of the complex phase we also define the *instantaneous complex frequency* $\Omega_{\mathrm{i}}(t) = \dot{\Phi}_{\mathrm{i}}(t) = \alpha_{\mathrm{i}}(t) + \mathrm{i}\beta_{\mathrm{i}}(t)$, where $\alpha_{\mathrm{i}}(t) = \dot{A}(t)/A(t)$ is the *instantaneous radial frequency* and $\beta_{\mathrm{i}}(t) = \dot{\phi}(t)$ is the *instantaneous angular frequency*. These quantities help us in unraveling the characteristics of the signal s which are not apparent from observing just the time dependence of $s(t)$.

Example Hilbert transformation is used as a tool in the analysis of brain potentials for the study of brain states and functions. The signals used in the analysis are obtained by digitizing the electro-encephalograph (EEG) recordings. An example of the EEG recording during an epileptic seizure is shown in Fig. 4.14 (top left). We use the sampled signal s_n to form the Hilbert transform $\mathcal{H}[s]_n$ (Fig. 4.14 (top right)). By using (4.53) and (4.54) we then compute the analytic amplitude and phase (Fig. 4.14 (bottom left and right)).

The actual (physiological) changes in the brain states can then be inferred from the sudden increases or abrupt drops in the analytic amplitude, jumps in the analytic phase or from the behavior of other observables formed from the Fourier and Hilbert transform of the original signal, for example, the instantaneous frequencies from (4.64). (However, similar effect may be caused by time-dependent interferences between different frequency components of the signal.) A good guide to work in this field can be found in the papers [35–38].

4.5.2 Kramers–Kronig Relations

Relations between real and imaginary parts of the Fourier transforms play prominent roles in many areas of physics. These relations have the form of Hilbert transforms. Let $s(t)$ be a real signal with the complex Fourier transform $S(\omega) = \mathcal{F}[s](\omega)$ which

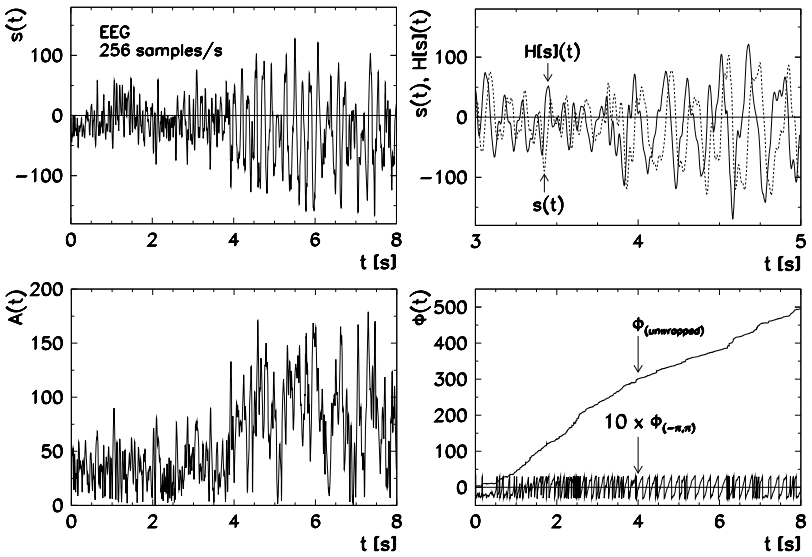


Fig. 4.14 Analysis of electro-encephalograms by using Hilbert transformation. [*Top left*] The raw signal s (2048 values, 256 samples per second) [39]. [*Top right*] The signal s and its Hilbert transform $\mathcal{H}[s]$ at $t \approx 4$ s where the nature of the signal changes. [*Bottom left*] The analytic amplitude A (see (4.53)). [*Bottom right*] The analytic phase ϕ (see (4.54)). The *lower part of the figure* shows the phase on the interval $(-\pi, \pi)$, while there is no such constraint in the *upper part* (at each crossing of the branch cut in the complex plane for the arctan function we add 2π to the phase)

is an analytic function on the upper half-plane of the complex variable ω. If S satisfies (4.52), its real and imaginary parts are related by

$$\operatorname{Re} S(\omega) = \frac{1}{\pi} P \int_{-\infty}^{\infty} \frac{\operatorname{Im} S(\omega')}{\omega' - \omega} \, \mathrm{d}\omega', \qquad \operatorname{Im} S(\omega) = -\frac{1}{\pi} P \int_{-\infty}^{\infty} \frac{\operatorname{Re} S(\omega')}{\omega' - \omega} \, \mathrm{d}\omega',$$

in short, $\operatorname{Re} S(\omega) = \mathcal{H}[\operatorname{Im} S](\omega)$ and $\operatorname{Im} S(\omega) = -\mathcal{H}[\operatorname{Re} S](\omega)$. These equations are known as the *Kramers–Kronig (dispersion) relations*. Relations of this type can be formulated for all Fourier transforms of signals which appear in the descriptions of the system's response to an external perturbation. They are very general: their derivation requires only that the response of the system is causal. This means that the signal s at times $t > 0$ is a consequence of the events occurring at $t \leq 0$.

Example The most famous Kramers–Kronig relations connect the real and imaginary parts of the electric susceptibility $\chi(\omega) = \operatorname{Re} \chi(\omega) + \mathrm{i} \operatorname{Im} \chi(\omega)$ or the refraction index $N(\omega) = n(\omega) + \mathrm{i}\kappa(\omega)$ [40]. The polarization $\boldsymbol{P}$ represents the linear response of the matter to the electric field $\boldsymbol{E}$,

$$\boldsymbol{P}(t) = \varepsilon_0 \int_{-\infty}^{\infty} \chi\big(t - t'\big) \boldsymbol{E}\big(t'\big) \, \mathrm{d}t', \quad \chi(\tau) = 0 \text{ at } \tau < 0.$$

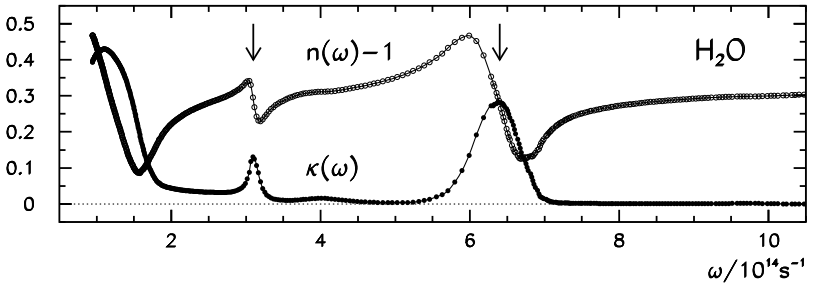


Fig. 4.15 Frequency dependence of the real and imaginary part of the complex refraction index of water in the range of micrometer wavelengths. They are related by (4.56). The *arrows* indicate the regions of anomalous dispersion (a sudden drop of the real part) and absorption (a rapid increase of the imaginary part)

In the frequency representation this means $\boldsymbol{P}(\omega) = \varepsilon_0 \chi(\omega) \boldsymbol{E}(\omega)$, where $\chi(\omega)$ is the susceptibility which depends on the frequency of the external perturbation $\boldsymbol{E}(\omega)$. Susceptibility χ satisfies the above Kramers–Kronig relations, which can be further simplified by using symmetry properties. We have $\chi(\omega)^* = \chi(-\omega)$, or $\operatorname{Re}\chi(-\omega) = \operatorname{Re}\chi(\omega)$ and $\operatorname{Im}\chi(-\omega) = -\operatorname{Im}\chi(\omega)$, so the integration range can be narrowed down to positive frequencies only,

$$\begin{aligned} \operatorname{Re}\chi(\omega) &= \frac{2}{\pi} P \int_0^\infty \frac{\omega' \operatorname{Im}\chi(\omega')}{\omega'^2 - \omega^2} \,\mathrm{d}\omega', \\ \operatorname{Im}\chi(\omega) &= -\frac{2}{\pi} P \int_0^\infty \frac{\omega \operatorname{Re}\chi(\omega')}{\omega'^2 - \omega^2} \,\mathrm{d}\omega'. \end{aligned} \tag{4.55}$$

We need only one more step to find the relations between the real (dispersive) and the imaginary (absorption) part of the complex refraction index. In the limit of small susceptibilities we have

$$N(\omega) = \sqrt{1 + \operatorname{Re}\chi(\omega) + \mathrm{i}\operatorname{Im}\chi(\omega)} \approx \underbrace{1 + \frac{1}{2}\operatorname{Re}\chi(\omega)}_{n(\omega)} + \underbrace{\frac{\mathrm{i}}{2}\operatorname{Im}\chi(\omega)}_{\mathrm{i}\kappa(\omega)}.$$

From (4.55) it then follows that

$$n(\omega) = 1 + \frac{2}{\pi} P \int_0^\infty \frac{\omega' \kappa(\omega')}{\omega'^2 - \omega^2} \,\mathrm{d}\omega'. \tag{4.56}$$

It is relatively hard to measure the frequency dependence of the real part of the refraction index n. But by using the derived relations it can be computed from the imaginary part κ which can be determined easily, just by measuring the ratio of the incident and transmitted wave intensities at different frequencies. As an example, Fig. 4.15 shows $n(\omega)$ and $\kappa(\omega)$ for infrared light in water.

4.5.3 Numerical Computation of the Continuous Hilbert Transform

The numerical computation of the Hilbert transform is almost always a tough nut to crack, as the transformation involves the integral of a singular function. Here we mention a few strategies for a quick and stable computation. More information can be found in the review article [41].

Transforming the Integrand to a Sum of Orthogonal Polynomials Let us first generalize the definition of the Hilbert transformation to integrals over the interval $I \subset \mathbb{R}$,

$$\mathcal{H}[f](y) = \frac{1}{\pi} P \int_I \frac{f(x)}{x-y} \, \mathrm{d}x.$$

If the interval $I = [a, b]$ is finite, we can rewrite the integral as

$$\mathcal{H}[f](y) = \frac{1}{\pi} f(y) \log \left| \frac{b-y}{a-y} \right| + \frac{1}{\pi} \int_a^b \frac{f(x) - f(y)}{x-y} \, \mathrm{d}x,$$

which eliminates the singularity. We write the integrand as the product of a positive weight function w and the remaining factor, $f(x) - f(y) = w(x)g(x; y)$. With foresight, we would like to find a particular pair (interval I, weight function w) that corresponds to the definition domain and the weight function of some system of orthogonal polynomials [12]. Then we could approximate the function g by the expansion

$$g(x; y) \approx \sum_{j=0}^{n} d_j(y) q_j(x),$$

where $\{q_j\}_{j \in \mathbb{N}_0}$ are orthogonal polynomials, and compute the Hilbert transform as the weighted sum of the transforms of the individual terms in this expansion:

$$\mathcal{H}[wg](y) \approx \sum_{j=0}^{n} d_j(y) \psi_j(y), \quad \psi_j(y) = \mathcal{H}[wq_j](y).$$

All systems of orthogonal polynomials q_j possess three-term recurrence relations which also apply to the functions ψ_j, e.g. $\psi_{j+1}(x) = (A_j x + B_j)\psi_j(x) + C_j \psi_{j-i}(x)$, where A_j, B_j, and C_j are constants that do not depend on x. This relation between q_j and ψ_j is based on the property of the Hilbert transformation

$$\mathcal{H}[xf](y) = y\mathcal{H}[f](y) + \frac{1}{\pi} \int_{-\infty}^{\infty} f(x) \, \mathrm{d}x$$

and the orthogonality of the polynomials q_j to a constant, $\int_{-\infty}^{\infty} q_j(x) w(x) \, \mathrm{d}x = 0$ for $j > 0$. Using the recurrence to compute ψ_j speeds up tremendously the computation of the transform $\mathcal{H}[wf](y)$. For the computation with the Chebyshev polynomials (weight $w(x) = (1 - x^2)^{-1/2}$ and $I = [-1, 1]$) see [42]; for the computation

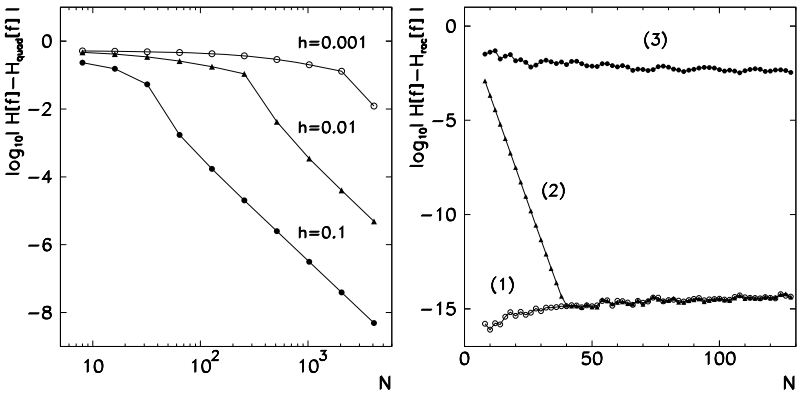


Fig. 4.16 The numerical computation of the Hilbert transform. [*Left*] The precision of the formula (4.57) in dependence of N and the width of the interval h, for the function $f(x) = (\sin x)/(1 + x^2)$. [*Right*] The precision of the algorithm (4.60) with the rational approximation (4.58) in dependence of N for three different functions: (1) $f(x) = 1/(1 + x^2)$, (2) $f(x) = 1/(1 + x^4)$, and (3) $f(x) = (\sin x)/(1 + x^2)$. Since the function (3) oscillates, we should obviously try a bit harder: see [45]

with the Hermite polynomials ($w(x) = \mathrm{e}^{-x^2}$ and $I = \mathbb{R}$) and the generalized Laguerre polynomials ($w(x) = x^\alpha \mathrm{e}^{-x}$ and $I = \mathbb{R}_+$) see [43].

Quadrature Formulas If our knowledge about the behavior of the integrand is very limited, the Hilbert transform can be computed by using quadrature formulas. A simple one is

$$\mathcal{H}[f](y) \approx \frac{2}{\pi} \sum_{n\in\mathbb{Z}} \frac{f(y + (2n + 1)h)}{2n + 1}, \tag{4.57}$$

which has been suggested and analyzed in [44]. Of course, in a practical implementation, we restrict $|n| \le N$, see Fig. 4.16 (left).

Collocation Method A slightly different approach to the Hilbert transformation has been charted by [46] and it exploits its spectral properties. (Note that in this part of the text we define $\mathrm{sign}(0) = 1$.) The authors use the functions

$$\rho_n(x) = \frac{(1 + \mathrm{i}\, x)^n}{(1 - \mathrm{i}\, x)^{n+1}}, \quad n \in \mathbb{Z}, \tag{4.58}$$

which satisfy $\mathcal{H}[\rho_n](x) = \mathrm{i}\,\mathrm{sign}(n)\rho_n(x)$ and are therefore the eigenfunctions of the Hilbert transformation. The functions ρ_n on $L^2(\mathbb{R})$ form a complete orthogonal basis and fulfill $\int_{-\infty}^{\infty} \rho_m^*(x)\rho_n(x)\,\mathrm{d}x = \pi\,\delta_{m,n}$. An arbitrary function $f \in L^2(\mathbb{R})$ can be expanded in terms of these basis functions as

$$f(x) = \sum_{n\in\mathbb{Z}} a_n \rho_n(x), \quad a_n = \frac{1}{\pi} \int_{-\infty}^{\infty} \rho_n(x)^* f(x)\,\mathrm{d}x. \tag{4.59}$$

The Hilbert transform of the function f is then

$$\mathcal{H}[f](x) = \mathrm{i} \sum_{n\in\mathbb{Z}} \operatorname{sign}(n) a_n \rho_n(x) \approx \mathrm{i} \sum_{n=-N}^{N-1} \operatorname{sign}(n) a_n \rho_n(x). \tag{4.60}$$

If the function f is real, the expansion coefficients satisfy $a_n = a^*_{-n-1}$, so one needs to compute only half of the coefficients a_n for $n = 0, 1, \ldots, N-1$. The coefficients a_n are given by (4.59). Alternatively, one can use the substitution $x = \tan(\phi/2)$ to rewrite the very same expression as

$$a_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} g_n(\phi)\, \mathrm{d}\phi, \quad g_n(\phi) = \left(1 - \mathrm{i} \tan\frac{\phi}{2}\right) f\left(\tan\frac{\phi}{2}\right) \mathrm{e}^{-\mathrm{i}n\phi}.$$

The function g is periodic on $[-\pi, \pi]$ and $\lim_{\phi\to\pm\pi} g(\phi) = 0$ is assumed. The approximation of the integral in the above expression can be obtained by various integration methods, for example, by using the trapezoidal rule with the points $\phi_j = \pi j/N$ for $|j| < N$. This results in the approximation

$$a_n \approx \frac{1}{2N} \sum_{j=-N+1}^{N-1} g_n(\phi_j).$$

By evaluating this formula with the fast Fourier transform we can compute the coefficients a_n for all n at once, requiring only $\mathcal{O}(N \log N)$ operations. This approach is reliable in cases where the coefficients a_n rapidly decrease as n increases. This behavior is typical for functions f that themselves fall off rapidly at infinity (see Fig. 4.16 (right)). Details can be found in [46].

4.5.4 Discrete Hilbert Transformation

We sample a continuous signal s equidistantly in t in steps of Δ and obtain the discrete signal $\{s_k\}_{k\in\mathbb{Z}}$ to which we assign a discrete Hilbert transform $\{\hat{s}_k\}_{k\in\mathbb{Z}}$. While there are several ways to do this, all of them reproduce the continuous transform (4.48) in the limit $\Delta \to 0$ if s is smooth enough. The discrete Hilbert transform is an arbitrarily good approximation of the continuous one.

Time-Domain Approach The discrete variant of the Hilbert transform for a finitely long signal $\{s_k\}$, specified at times $t_k = k\Delta$, can be cast in the form

$$\hat{s}_n = \mathcal{H}_N[s]_n = \frac{1}{\pi} \sum_{m=1}^{N} \frac{s_{n+m} - s_{n-m}}{m},$$

where N is the number of points in the sample to the left and right of t_n which are included in the sum. The discrete transform reverts to the continuous one when

$N \to \infty$ and $\Delta \to 0$. "Infinitely long" signals occur when there is an incessant flow of data from the measuring apparatus and we wish to process them in real time, or if the amount of data is large compared to the available memory. We should therefore always estimate the extent of the signal with respect to some reference point that we still wish to use in the computation of the transform.

It is possible to compute the discrete Hilbert transform by numerically evaluating the integral in the continuous transform (4.48). We either use the interpolant of the signal s or a chosen model function is fitted to the signal: both can be simply and stably convoluted with the function $h(t) = 1/(\pi t)$. By linear interpolation of the signal values [47] we obtain the discrete transform

$$\begin{aligned}\mathcal{H}_N[s]_n = -\frac{1}{\pi}\Bigg\{ &\sum_{k=0}^{n-2} s_k \phi(k-n+1) + (s_k - s_{k+1})\big[1 + (n-k)\phi(n-k)\big] \\ &+ s_{n-1} - s_{n+1} + \sum_{k=n+2}^{N-1} s_k \phi(k-n) \\ &+ (s_{k-1} - s_k)\big[1 + (k-n)\phi(k-n)\big]\Bigg\},\end{aligned}$$

where we sample the signal s_k at $k\Delta t$ $(k = 0, 1, \ldots, N-1)$ and $\phi(k) = \log(1 - 1/k)$. This computation requires $\mathcal{O}(N^2)$ operations. Boche's approach [48] which is tailored to signals of limited bandwidth, is also found in practice.

Fourier-Domain Approach The Fourier approach is fruitful for periodic signals $\{s_k\}_{k=0}^{N-1}$ (for which $s_{k+N} = s_k$). The discrete Fourier transform of the signal is

$$S_n = \mathcal{F}_N[s]_n = \frac{1}{N}\sum_{k=0}^{N-1} s_k \, \mathrm{e}^{-\mathrm{i}2\pi nk/N}.$$

We define the discrete Hilbert transform for such a signal as the convolution

$$\hat{s}_n = \mathcal{H}_N[s]_n = \sum_{k=0}^{N-1} h_{n-k} s_k = \sum_{k=0}^{N-1} h_k s_{n-k}, \tag{4.61}$$

where $\{h_k\}_{k=0}^{N-1}$ is the convolution kernel and we assume $h_{k+N} = h_k$, $s_{k+N} = s_k$. We determine the kernel by translating the property (4.50) of the continuous Hilbert transform to discrete language, namely $\mathcal{H}[e](t) = \mathrm{i}\,\mathrm{sign}(\omega)e(t)$, where $e(t) = \exp(\mathrm{i}\,\omega t)$. This means

$$\mathcal{H}_N\big[e^{(k)}\big]_n = \mathrm{i}\,\mathrm{sign}(N-2k)\, e_n^{(k)}, \quad e_n^{(k)} = \exp(\mathrm{i}\,2\pi kn/N). \tag{4.62}$$

We have already taken into account the definitions of the negative and positive frequencies in the discrete Fourier transform. By using the relation between the discrete

Fourier transform and the convolution, we can express the Hilbert transform (4.61) as

$$\mathcal{H}_N[s]_n = -N\mathcal{F}_N^{-1}[F]_n, \quad F = \{H_i S_i\}_{i=0}^{N-1},$$

where we have denoted $S = \mathcal{F}_N[s]$ and $H = \mathcal{F}_N[h]$. Condition (4.62) is satisfied if $H_k = -\mathrm{i}\,\mathrm{sign}(N-2k)/N$. After using (4.62) the path to the discrete Hilbert transform becomes clear: we compute the components of the Fourier transform S_k of the signal s, multiply them by $\mathrm{i}\,\mathrm{sign}(N-2k)$, and compute the inverse Fourier transform of the product:

$$\mathcal{H}_N[s]_n = \mathrm{i}\sum_{k=0}^{N-1} S_k\,\mathrm{sign}(N-2k)\,\mathrm{e}^{\mathrm{i}2\pi nk/N}. \tag{4.63}$$

By analogy with the continuous case (4.51) we define the complex analytic signal corresponding to the discrete Hilbert transform:

$$s_n^{\#} = s_n - \mathrm{i}\,\hat{s}_n, \quad \hat{s}_n = \mathcal{H}_N[s]_n.$$

Its real and imaginary parts can again be used to elegantly compute the instantaneous quantities (see p. 186). For example, the instantaneous radial frequency α_n and the instantaneous angular frequency β_n become

$$\alpha_n = \frac{s_n' s_n + \hat{s}_n' \hat{s}_n}{s_n^2 + \hat{s}_n^2}, \qquad \beta_n = \frac{\hat{s}_n' s_n - s_n' \hat{s}_n}{s_n^2 + \hat{s}_n^2}, \tag{4.64}$$

where $'$ denotes the time derivative. Note that even these derivatives can be computed by using the discrete Fourier transformation. If the array $\{f_k\}_{k=0}^{N-1}$ corresponds to the discrete Fourier transform $\{F_n\}_{n=0}^{N-1}$, the time derivative of the component f_k is given by

$$f_k' = \sum_{n=0}^{N-1} \omega_n F_n\,\mathrm{e}^{\mathrm{i}2\pi nk/N}, \quad \omega_n = \frac{2\pi}{N}\left[\left(n - \frac{N}{2}\right)\mathrm{sign}(N-2n) + \frac{N}{2}\right],$$

where the physical angular frequencies ω_k belong to the individual Fourier modes. Only $\mathcal{O}(N\log N)$ operations are needed if FFT is used to compute the discrete Hilbert transform (4.63).

The Fourier-domain approach is the fastest among the possibilities described here, but it does have its deficiencies. The signal is periodic but rapid changes in the signal may be aliased in the frequency spectrum and therefore misinterpreted when the convolution is performed. It is not wise to use the Fourier approach if the Hilbert transform falls off too slowly at infinity as the finite Fourier series is not optimal for the description of the long tails. A detailed discussion of the variants of the Hilbert transform with emphasis on signal processing applications can be found in [34, 49, 50] and in the monumental work [51].

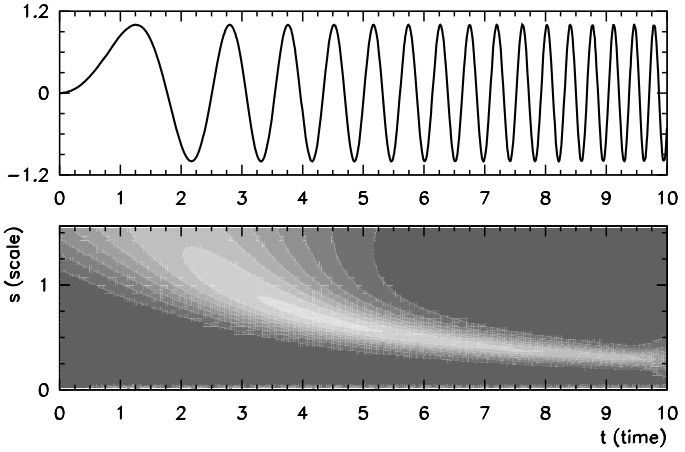


Fig. 4.17 The basic idea of the continuous wavelet transform. [*Top*] The signal $f(t) = \sin \omega t$, $\omega \propto t$. [*Bottom*] In this portion of the signal, the continuous wavelet transformation detects large structures at short times (where the waves have a typical scale $s \approx 1.5$) and small structures at long times (scale $s \approx 0.3$). The frequency and the scale of the oscillations are inversely proportional, which generates the typical curvature of the transform ($s \propto \omega^{-1} \propto t^{-1}$)

4.6 Wavelet Transformation ⋆

We may think of the *wavelet transformation* of a signal as an extension of its Fourier analysis, through which not only the strengths of the signal's frequency components are determined, but also the times at which these components occur. The classic example in Fig. 4.17 illustrates this basic idea for the signal $\sin(t^2)$ whose frequency linearly increases with time. By using the wavelet transformation we can also locate changes in the signal that are not immediately apparent from its temporal behavior alone (Fig. 4.18).

The *continuous wavelet transform* (CWT) of the function f is defined as

$$L_\psi[f](s,t) = \frac{1}{\sqrt{c_\psi s}} \int_{-\infty}^{\infty} f(\tau)\psi^*\left(\frac{\tau - t}{s}\right) \mathrm{d}\tau, \quad t \in \mathbb{R},\ s \neq 0,$$

where t is the time at which a feature of scale s is observed in the function f, and c_ψ is the normalization constant. The function ψ, whose properties are given in the following, should allow us to change the parameter s (the typical scale of a structure in the signal f) as well as its shift t with respect to the signal f. By denoting

$$\psi_s(t) = \psi^*(-t/s)$$

we can rewrite the definition in the form of a convolution

$$L_\psi[f](s,t) = \frac{1}{\sqrt{c_\psi s}} \int_{-\infty}^{\infty} f(\tau)\psi_s(t - \tau)\,\mathrm{d}\tau. \tag{4.65}$$

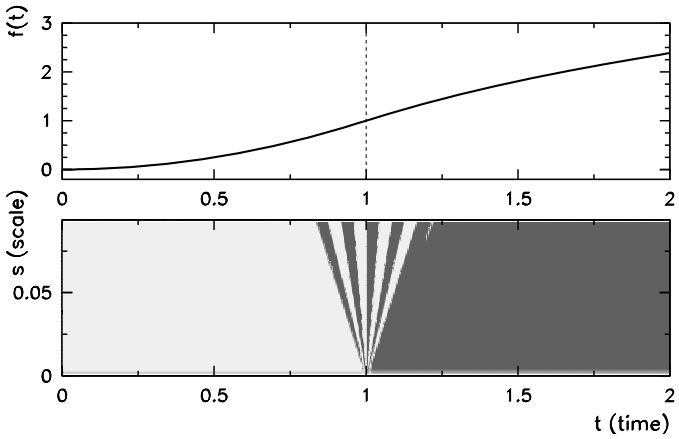


Fig. 4.18 Continuous transform of a real signal with a complex Morlet wavelet (4.68) [*Top*] The signal $f(t) = t^2$ $(t < 1)$ or $f(t) = 1 + 2\log t$ $(t \geq 1)$ is continuous and has a continuous first derivative at $t = 1$ while its second derivative is discontinuous. [*Bottom*] The phase of the wavelet transform in the vicinity of $t = 1$ oscillates a couple of times, and reveals the location of the critical point when the scale s is decreased

The function ψ should satisfy certain conditions. Its "energy" should be bounded, which means $\int_{-\infty}^{\infty} |\psi(t)|^2 \, \mathrm{d}t < \infty$, and the weighted integral of its spectral density (the square of the Fourier transform $\hat{\psi}$) should be finite:

$$c_\psi = 2\pi \int_{-\infty}^{\infty} \frac{1}{|\omega|} \left|\hat{\psi}(\omega)\right|^2 \mathrm{d}\omega < \infty.$$

The functions ψ found in the literature usually fulfill this *admissibility condition* by design. Moreover, we require the functions ψ to fulfill

$$\int_{-\infty}^{\infty} \psi(\eta) \, \mathrm{d}\eta = 0. \tag{4.66}$$

The functions ψ therefore oscillate around the abscissa and fall off rapidly at large distances from the origin, giving them the appearance of small waves and the nickname *wavelets*. The simplest wavelet is the Haar function

$$\psi_{\mathrm{Haar}}(\eta) = \begin{cases} 1; & 0 \leq \eta < 1/2, \\ -1; & 1/2 \leq \eta < 1, \\ 0; & \text{otherwise.} \end{cases}$$

The *derivative of Gaussian* wavelets DOG(m) are also very simple to use. We obtain them by successive derivatives of the Gauss function,

$$\psi_{\mathrm{DOG}(m)}(\eta) = \frac{(-1)^{m+1}}{\sqrt{\Gamma(m+1/2)}} \frac{\mathrm{d}^m}{\mathrm{d}\eta^m} \left(\mathrm{e}^{-\eta^2/2}\right). \tag{4.67}$$

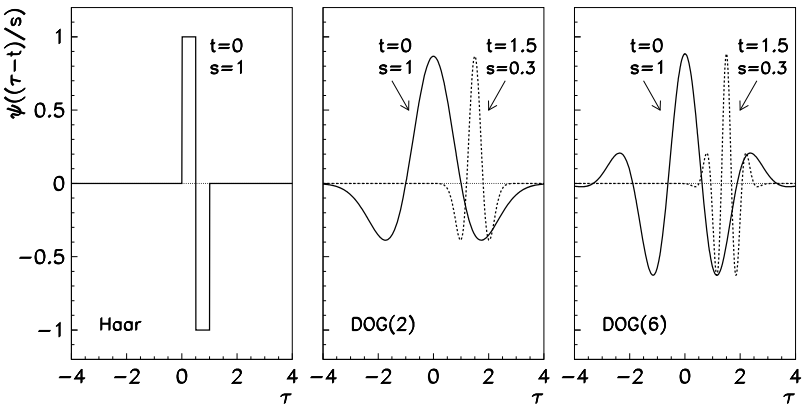


Fig. 4.19 Examples of wavelets used in the continuous wavelet transformation. By horizontal shifts and changes of scale the wavelet probes the features of the investigated signal and the times at which these features appear. [*Left*] Haar wavelet. [*Center*] The DOG(2) wavelet known as the "Mexican hat". [*Right*] The DOG(6) wavelet

Another useful wavelet is the complex Morlet wavelet

$$\psi_{\text{Morlet}}(\eta) = \pi^{-1/4}\, \mathrm{e}^{\mathrm{i}\omega_0\eta}\, \mathrm{e}^{-\eta^2/2}, \quad \omega_0 \in [5, 6]. \tag{4.68}$$

(For the Morlet wavelet (4.66) is not exactly fulfilled; the absolute precision to which the equality is valid improves when ω_0 is increased, and amounts to at least $\approx 10^{-5}$ for $\omega_0 > 5$.) When complex wavelets are used, the corresponding transforms should be specified in terms of their magnitudes and phases (see Fig. 4.18). Some typical wavelets are shown in Fig. 4.19.

4.6.1 Numerical Computation of the Wavelet Transform

The continuous wavelet transform (4.65) of the discrete values of the signal f_k with the chosen scale parameter s is evaluated by computing the convolution sum [52]

$$L_\psi[f](s, t_n) = \frac{1}{\sqrt{c_\psi s}} \sum_{k=0}^{N-1} f_k \psi^*\left(\frac{(k-n)\Delta t}{s}\right), \quad n = 0, 1, \ldots, N-1.$$

We take the values of s from an arbitrary set $\{s_m\}_{m=0}^{M-1}$ with some $M < N$. The most simple choice is $s_m = (m+1)\Delta t$. The computation of the sum becomes unacceptably slow for large N and M as the time cost increases as $\mathcal{O}(MN^2)$. Since the convolution of two functions in configuration space is equivalent to the multiplication of their Fourier transforms in the Fourier space, the CWT can be computed by using the fast Fourier transformation (FFT).

Wavelets Given as Continuous Functions The procedure is simple when wavelet functions exist in closed forms and for which the analytic form of their Fourier transforms in known. First we use the FFT to compute the Fourier transform F of the signal f which has been sampled at N points with uniform spacings Δt:

$$F_k = \mathcal{F}_N[f]_k = \frac{1}{N}\sum_{n=0}^{N-1} f_n \, \mathrm{e}^{-\mathrm{i}\,2\pi kn/N}. \tag{4.69}$$

If the wavelet is given by the function $\psi(t/s)$ in configuration space, its correlate in Fourier space (in the continuous limit) is the function $\hat{\psi}(s\omega)$. For example, the family of wavelets (4.67) corresponds to the family of transforms

$$\hat{\psi}_{\mathrm{DOG}(m)}(s\omega) = \frac{-\mathrm{i}^m}{\sqrt{\Gamma(m+1/2)}}(s\omega)^m \, \mathrm{e}^{-(s\omega)^2/2}.$$

The wavelet transform is then evaluated by using the inverse FFT to compute the sum

$$L_\psi[f](s, t_n) = \frac{1}{\sqrt{c_\psi s}}\sum_{k=0}^{N-1} F_k \hat{\psi}^*(s\omega_k) \, \mathrm{e}^{\mathrm{i}\,\omega_k n \Delta t},$$

where

$$\omega_k = \begin{cases} 2\pi k/(N\Delta t); & k \le N/2, \\ -2\pi k/(N\Delta t); & k > N/2. \end{cases}$$

The numerical cost of this procedure is $\mathcal{O}(MN \log N)$.

Wavelets Given at Discrete Points If the wavelet is not specified as an analytic function or if its Fourier representation is not known, we need to compute both the discrete Fourier transform of the sampled signal (4.69) and the discrete Fourier transform of the wavelet at scale s, i.e.

$$Y_k = \mathcal{F}_N[\psi_s]_k = \frac{1}{N}\sum_{n=0}^{N-1} \psi^*(-n\Delta t/s) \, \mathrm{e}^{-\mathrm{i}\,2\pi kn/N}.$$

The convolution is at the heart of this procedure and the way the sampling is done requires some attention. For even wavelets it makes sense to adopt the sampling which preserves the symmetry of the wavelet about its origin. The wavelet and the signal are sampled as shown in Fig. 4.20.

We obtain the wavelet transform by multiplying the arrays $F_k \equiv \mathcal{F}_N[f]_k$ and $Y_k \equiv \mathcal{F}_N[\psi_s]_k$ component-wise, diving the result by $\sqrt{c_\psi s}$, and computing the inverse Fourier transform of the product array W_k:

$$L_\psi[f](s) = N\mathcal{F}_N^{-1}[W], \qquad W_k \equiv \frac{1}{\sqrt{c_\psi s}} F_k Y_k.$$

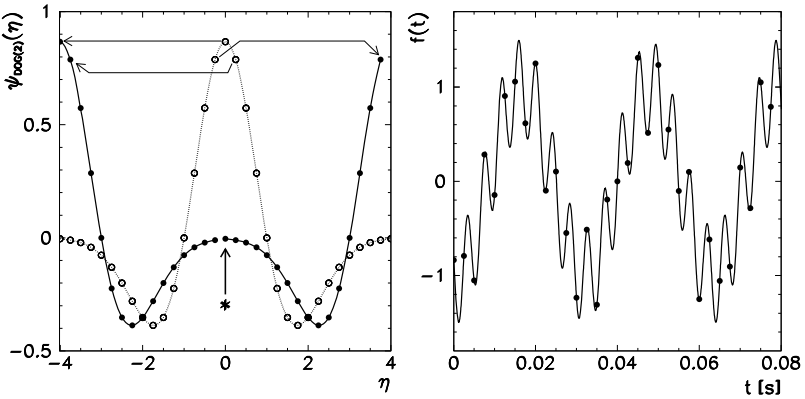


Fig. 4.20 Sampling at $N = 32$ points for the computation of the continuous wavelet transform. [*Left*] Periodic sampling of the wavelet in its natural (dimensionless) scale at the maximum scaling parameter s. The wavelet is sampled at N points on $[\eta_{\min}, \eta_{\max}] = [-4, 4]$. When we wish to reduce the parameter s, we insert zeros at the location marked by the thick arrow and the symbol $*$. [*Right*] Sampling of the signal at N points of the physical (time) scale or the interval $[0, N\Delta t]$. The relation between the dimensionless variable η and the physical time t is $\eta = t(\eta_{\max} - \eta_{\min})/(N\Delta t)$. For details see [53]

The inverse procedure is at hand: we reconstruct the original signal from the continuous wavelet transform by deconvolution, i.e. by dividing the Fourier representation of the transform by the Fourier representation of the wavelet. We form the arrays $L_k \equiv \mathcal{F}_N[L_\psi[f](s)]_k$ and $Y_k \equiv \mathcal{F}_N[\psi_s]_k$, divide them, multiply them by $\sqrt{c_\psi s}$, and compute the inverse Fourier transform of the quotient array:

$$f = N^{-1}\mathcal{F}_N^{-1}[F], \quad F_k \equiv \sqrt{c_\psi s}(L_k/Y_k).$$

(The parameter s obviously has to be chosen such that $Y_k \neq 0$ for all k.)

4.6.2 Discrete Wavelet Transform

Even though we have sampled the signal and the wavelet at discrete points only, the transform described above can still be considered continuous, since the parameter s and the time axis span all $M \times N$ values. The continuous wavelet transform of a function sampled at N points has $M \times N$ values and is therefore highly redundant. In contrast, the wavelet transform that represents the signal uniquely at just $\approx N$ points, is known as the *discrete wavelet transform* (DWT). It lies at the heart of modern data (de)compression algorithms (an example with the CWT is given in Problem 4.7.4). The treatment of the DWT is beyond the scope of this book; excellent introductory texts are [54–57]. A comparison of the wavelet transform and the Fourier transform is discussed in [58].

4.7 Problems

4.7.1 Fourier Spectrum of Signals

The discrete Fourier transformation (DFT, Sect. 4.2.2) is the fundamental tool of signal analysis. First we confirm the formulas for known pairs of periodic signals $f = \{f_j\}_{j=0}^{N-1}$ and their transforms $F = \{F_k\}_{k=0}^{N-1} = \mathcal{F}_N[f]$, for example,

$$f_j = \mathrm{e}^{\mathrm{i}aj/N} \quad \Leftrightarrow \quad F_k = \frac{1}{N}\left\{\left(\mathrm{e}^{\mathrm{i}a} - 1\right)/\left(\mathrm{e}^{\mathrm{i}(a-2k\pi)/N} - 1\right)\right\},$$
$$\mathrm{Re}\, f_j, \mathrm{Im}\, f_j \sim \mathcal{N}(0,1) \quad \Leftrightarrow \quad \mathrm{Re}\, F_k, \mathrm{Im}\, F_k \sim \mathcal{N}(0,N),$$

where $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution with average μ and variance σ^2.

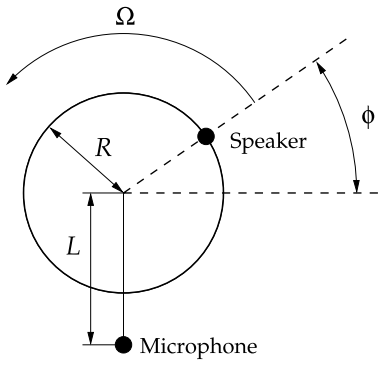
⊙ Compute the Fourier transforms of simple signals, in which several frequencies are represented. Compare the results in the case when the sample is periodic on the interval (the chosen frequencies are integer multiples of the fundamental frequency), to those cases when it is not. Observe the effect of aliasing for a sample that includes frequencies above the critical value. Show that for real signals $F_{N-k} = F_k^*$ and that $\mathcal{F}_N$ and $\mathcal{F}_N^{-1}$ are truly inverse operations.

Perform a Fourier analysis of the sound generated by tapping on a rectangular wood-box resonator, creating transient acoustic waves in its interior. Plot the power spectral density and identify the eigenmodes of the resonator from its most prominent peaks. Analyze the sound of the tuning fork attached to the resonator of a guitar. (The signals can be found at the book's web-page.)

⊕ Fourier-analyze the concluding chord of the Toccata and Fugue for organ in d-minor, BWV 565, of J.S. Bach (Fig. 4.5). The signal from the audio CD has been sampled at 44100 Hz, 11025 Hz, 5512 Hz, 2756 Hz, 1378 Hz, and 882 Hz. By listening to the recordings in the `.mp3` format find out what happens when the sampling frequency is reduced, then try to confirm this by Fourier analysis.

4.7.2 Fourier Analysis of the Doppler Effect

Here we discuss the experiment set up in Salzburg, the birthplace of C. Doppler (1805–1853). A speaker is attached to a rotating wheel with radius R, and there is a microphone in the plane of rotation at a distance L from the center of the wheel, as shown in the figure. The rotation angular frequency is $\Omega = v/R$, where v is the tangential velocity. The radius vector $\boldsymbol{r}(t)$ describes the relative position of the microphone with respect to the speaker. The speaker periodically approaches and recedes from the microphone (period $2\pi/\Omega$).



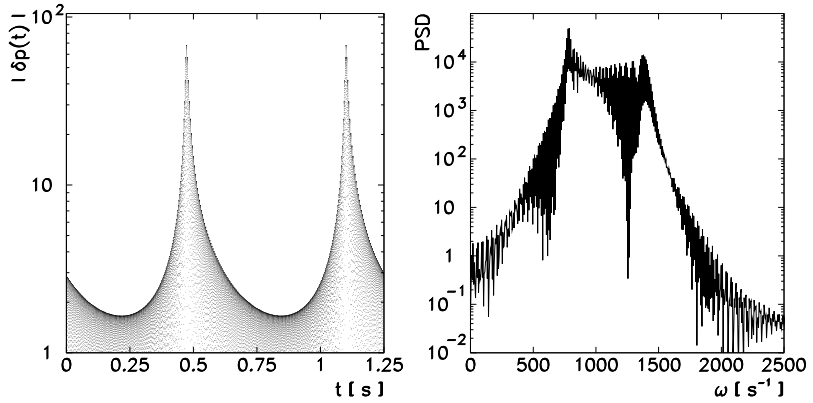


Fig. 4.21 [*Left*] An example of the signal at $v = 100\,\mathrm{m/s}$, $R = 10\,\mathrm{m}$, $L = 10.5\,\mathrm{m}$, $\omega = 1000/\mathrm{s}$, and $c = 340\,\mathrm{m/s}$. [*Right*] The Fourier power spectral density. The peaks in the spectrum occur approximately at $\omega/(1 + v/c) = 773\,\mathrm{Hz}$ and $\omega/(1 - v/c) = 1417\,\mathrm{Hz}$

The microphone is used to measure the pressure differences $\delta p(t)$ given by the formula

$$\delta p(t) = A\frac{\cos[\omega(t - x(\Omega t)/c)]}{x(\Omega t)}.$$

Here A is the amplitude, ω is the angular frequency of the emitted sound and c is its velocity. The function $x(t)$ is defined implicitly as the positive solution of the equation $l^2 + r^2 + 2lr\sin(t - x) = x^2$, where $l = \Omega L/c$ and $r = \Omega R/c$.

$\odot$ Let $v = 100\,\mathrm{m/s}$ and $A = 1$. Sample the signal $\delta p(t)$ at time intervals $\Delta t = 1/(\Omega N)$ over two periods. This gives you the discrete signal $f_j = \delta p(j\,\Delta t)$ at $j = 0, 1, \ldots, N - 1$. Compute the DFT and the Hilbert transform of the discrete signal in the limits $L \ll R$, $L \gg R$, and $L \approx R$, and analyze its single-sided power spectral density (PSD), its instantaneous amplitude, and instantaneous frequency. An example of the signal and its spectral density is shown in Fig. 4.21. Compare the results to the classical Doppler prediction.

$\oplus$ Analyze the PSD of the signal at smaller time windows which open just slightly ahead of the time when the speaker is nearest to the microphone, and close just after that.

4.7.3 Use of Laplace Transformation and Its Inverse

The time dependence of the charge on a capacitor connected in series to a electric generator, a resistor, and a coil, is described by the differential equation

$$L\frac{\mathrm{d}^2Q}{\mathrm{d}t^2} + R\frac{\mathrm{d}Q}{\mathrm{d}t} + \frac{1}{C}Q = U_\mathrm{g}(t)$$

or, in dimensionless form, at given R, C and L, with $R\sqrt{C/L} = 1$,

$$\ddot{x} + \dot{x} + x = g(t).$$

With initial conditions $x(0) = \dot{x}(0) = 0$ the source generates a voltage pulse of the form

$$g(t) = 1 - \theta(t - \pi) = \begin{cases} 1; & 0 \le t < \pi, \\ 0; & t \ge \pi. \end{cases}$$

By using the properties (4.45) and Table 4.1, and by taking the initial conditions into account, the Laplace transform of the differential equation is

$$s^2 X(s) + sX(s) + X(s) = \mathcal{L}[1] - \mathcal{L}\big[\theta(t - \pi)\big] = \big(1 - \mathrm{e}^{-\pi s}\big)/s$$

or

$$X(s) \equiv \big(1 - \mathrm{e}^{-\pi s}\big)\, H(s), \quad H(s) = \frac{1}{s(s^2 + s + 1)} = \frac{1}{s} - \frac{(s + \frac{1}{2}) + \frac{1}{2}}{(s + \frac{1}{2})^2 + \frac{3}{4}}.$$

Let us denote $h(t) = \mathcal{L}^{-1}[H]$ and remember that the inverse Laplace transformation is linear. Then by looking at Table 4.1 once more and considering the property $\mathcal{L}[\mathrm{e}^{ct} f(t)] = F(s - c)$, we find the analytic solution

$$x(t) = \mathcal{L}^{-1}[X] = \mathcal{L}^{-1}[H] - \mathrm{e}^{-\pi s}\mathcal{L}^{-1}[H] = h(t) - h(t - \pi)\theta(t - \pi),$$

where

$$h(t) = \mathcal{L}^{-1}[H] = 1 - \frac{1}{3}\,\mathrm{e}^{-t/2}\left(3\cos\frac{\sqrt{3}}{2}\,t + \sqrt{3}\sin\frac{\sqrt{3}}{2}\,t\right).$$

⊙ Use the inverse Laplace transformation to find the solution $x(t)$ from the function $X(s) = (1 - \mathrm{e}^{-\pi s})H(s)$. Since a discontinuous function $g(t)$ appears in the time domain, the numerical inverse should be computed by dedicated algorithms specially tailored to such functions. You may use the methods from [27].

4.7.4 Use of the Wavelet Transformation

The continuous wavelet transformation is one of the basic tools in signal processing and analysis. We use it to determine the frequency components of signals and the instances in time when individual components actually occur: this is its main advantage over the usual Fourier transformation.

⊙ Use the continuous wavelet transformation on a simple periodic signal to which components with higher frequencies have been admixed at certain subinter-

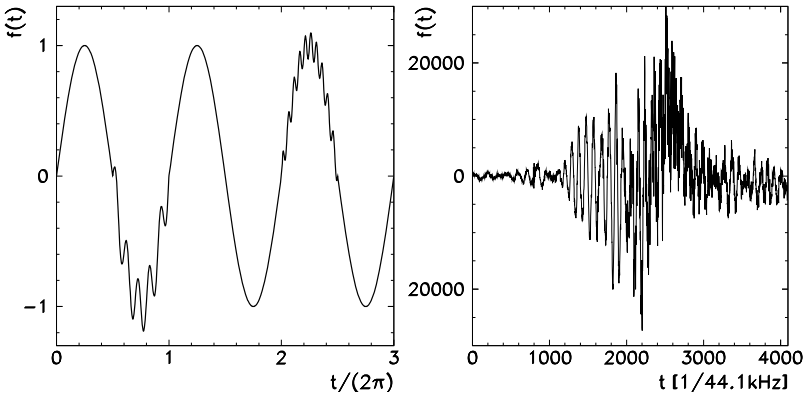


Fig. 4.22 [*Left*] The signal (4.70) with three frequency components, two of which appear only occasionally. [*Right*] A recording of the intensity of sound caused by quickly waving a bamboo-stick through air

vals, for example,

$$f(t) = \begin{cases} \sin t + 0.2 \sin 10t; & \pi \le t \le 2\pi, \\ \sin t + 0.1 \sin 20t; & 4\pi \le t \le 5\pi, \\ \sin t; & \text{otherwise} \end{cases} \tag{4.70}$$

(see Fig. 4.22 (left)). You can additionally perturb the signal by mixing it with noise of various amplitudes. How do such admixtures influence the quality (the resolution capability) of the wavelet transform? Use different types of wavelet functions. What differences can you observe?

⊕ The continuous wavelet transformation also gives us some basic feeling for the properties of compression and decompression of data. Figure 4.22 (right) shows a typical acoustic signal. (The file can be found at the book's web-page.) By using the procedure described in Sect. 4.6.1 compute the continuous wavelet transform for this signal. Set to zero all components in the transform that are below a certain chosen threshold (for example, keep only 20, 10, or 5 % of the strongest components). Then perform the inverse wavelet transformation and compare the result to the original signal. How has it changed? Do you reach the same conclusion if you repeat the exercise with the Fourier transform? (Use the FFT to map the signal into Fourier space, keep 20, 10, or 5 % of the strongest components, and compute the inverse FFT.)

References

1. B.D. MacCluer, *Elementary Functional Analysis* (Springer, New York, 2010)
2. C. Shannon, Communication in the presence of noise. Proc. IEEE **86**, 447 (1998) (reprint)
3. M. Unser, Sampling-50 years after Shannon. Proc. IEEE **88**, 569 (2000)

4. H. Nyquist, Certain topics in telegraph transmission theory. Proc. IEEE **90**, 280 (2002) (reprint)
5. C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zang, *Spectral Methods. Fundamentals in Single Domains* (Springer, Berlin, 2006)
6. D. Donnelly, B. Rust, The fast Fourier transform for experimentalists, part I: concepts. Comput. Sci. Eng. **Mar/Apr**, 80 (2005)
7. F.J. Harris, On the use of windows for harmonic analysis with the discrete Fourier transform. Proc. IEEE **66**, 51 (1978)
8. M. Frigo, S.G. Johnson, The design and implementation of FFTW3. Proc. IEEE **93**, 216 (2005); documentation can be found at http://www.fftw.org
9. W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 3rd edn. (Cambridge University Press, Cambridge, 2007). See also the equivalent handbooks in Fortran, Pascal and C, as well as http://www.nr.com
10. P. Duhamel, M. Vetterli, Fast Fourier transforms: a tutorial review and a state of the art. Signal Process. **19**, 259 (1990)
11. J.C. Schatzman, Accuracy of the discrete Fourier transform and the fast Fourier transform. SIAM J. Sci. Comput. **17**, 1150 (1996)
12. G. Szegö, *Orthogonal Polynomials* (Am. Math. Soc., Providence, 1939)
13. T.J. Rivlin, *The Chebyshev Polynomials* (Wiley, New York, 1974)
14. V. Rokhlin, A fast algorithm for discrete Laplace transformation. J. Complex. **4**, 12 (1988)
15. J. Strain, A fast Laplace transform based on Laguerre functions. Math. Comput. **58**, 275 (1992)
16. M. Abramowitz, I.A. Stegun, *Handbook of Mathematical Functions*, 10th edn. (Dover, Mineola, 1972)
17. S. Zhang, J. Jin, *Computation of Special Functions* (Wiley-Interscience, New York, 1996)
18. W.E. Boyce, R.C. DiPrima, *Elementary Differential Equations and Boundary Value Problems*, 5th edn. (Wiley, New York, 1992)
19. J. Abate, P.P. Valkó, Multi-precision Laplace transform inversion. Int. J. Numer. Methods Eng. **60**, 979 (2004)
20. A.M. Cohen, *Numerical Methods for Laplace Transform Inversion* (Springer, New York, 2007)
21. B. Davies, B. Martin, Numerical inversion of the Laplace transform: a survey and comparison of methods. J. Comput. Phys. **33**, 1 (1979)
22. D.G. Duffy, On the numerical inversion of Laplace transforms: comparison of three new methods on characteristic problems from applications. ACM Trans. Math. Softw. **19**, 333 (1993)
23. C.L. Epstein, J. Schotland, The bad truth about Laplace's transform. SIAM Rev. **50**, 504 (2008)
24. R. Piessens, Gaussian quadrature formulas for the numerical integration of Bromwich's integral and the inversion of the Laplace transform. J. Eng. Math. **5**, 1 (1971)
25. L.N. Trefethen, J.A.C. Weideman, T. Schmelzer, Talbot quadratures and rational approximations. BIT Numer. Math. **46**, 653 (2006)
26. J. Abate, W. Whitt, The Fourier-series method for inverting transforms of probability distributions. Queueing Syst. **10**, 5 (1992)
27. P. den Iseger, Numerical transform inversion using Gaussian quadrature. Probab. Eng. Inf. Sci. **20**, 1 (2006)
28. A. Yonemoto, T. Hisikado, K. Okumura, Accuracy improvement of the FFT-based numerical inversion of Laplace transforms. IEEE Proc., Circuits Devices Syst. **150**, 399 (2003)
29. J. Abate, G.L. Choudhury, On the Laguerre method for numerically inverting Laplace transforms. INFORMS J. Comput. **8**, 413 (1996). This paper presents efficient algorithms based on the expansion of the function f in terms of Laguerre polynomials and the corresponding computation of $\mathcal{L}[f]$ and $\mathcal{L}^{-1}[F]$
30. E. Titchmarsh, *Introduction to the Theory of Fourier Integrals*, 2nd edn. (Clarendon, Oxford, 1948)
31. L. Grafakos, *Classical and Modern Fourier Analysis* (Prentice Hall, New Jersey, 2003)

32. M.L. Glasser, Some useful properties of the Hilbert transform. SIAM J. Math. Anal. **15**, 1228 (1984)
33. R. Wong, Asymptotic expansion of the Hilbert transform. SIAM J. Math. Anal. **11**, 92 (1980)
34. S.L. Hahn, *Hilbert Transform in Signal Processing* (Artech House, Boston, 1996)
35. W.J. Freeman, Origin, structure, and role of background EEG activity, part 1: analytic amplitude. Clin. Neurophysiol. **115**, 2077 (2004)
36. W.J. Freeman, Origin, structure, and role of background EEG activity, part 2: analytic phase. Clin. Neurophysiol. **115**, 2089 (2004)
37. W.J. Freeman, Origin, structure, and role of background EEG activity, part 3: neural frame classification. Clin. Neurophysiol. **116**, 1118 (2005)
38. W.J. Freeman, Origin, structure, and role of background EEG activity, part 4: neural frame simulation. Clin. Neurophysiol. **117**, 572 (2006)
39. M. West, A. Krystal, EEG, Duke University
40. C. Kittel, *Introduction to Solid State Physics*, 8th edn. (Wiley, New York, 2005)
41. M. Nandagopal, N. Arunajadai, On finite evaluation of finite Hilbert transform. Comput. Sci. Eng. **9**, 90 (2007)
42. T. Hasegawa, T. Torii, Hilbert and Hadamard transforms by generalized Chebyshev expansion. J. Comput. Appl. Math. **51**, 71 (1994)
43. W. Gautschi, J. Waldvogel, Computing the Hilbert transform of the generalized Laguerre and Hermite weight functions. BIT Numer. Math. **41**, 490 (2001)
44. V.R. Kress, E. Martensen, Anwendung der Rechteckregel auf die reelle Hilberttransformation mit unendlichem Intervall. Z. Angew. Math. Mech. **50**, 61 (1970)
45. F.W. King, G.J. Smethells, G.T. Helleloid, P.J. Pelzl, Numerical evaluation of Hilbert transforms for oscillatory functions: a convergence accelerator approach. Comput. Phys. Commun. **145**, 256 (2002)
46. J.A.C. Weideman, Computing the Hilbert transform on the real line. Math. Comput. **64**, 745 (1995). Attention: the authors use a non-standard definition $\mathrm{sign}(0) = 1$; in Eq. (22) correct $1/N \to 1/(2N)$
47. X. Wang, Numerical implementation of the Hilbert transform. PhD thesis, University of Saskatchewan, Saskatoon, 2006
48. H. Boche, M. Protzmann, A new algorithm for the reconstruction of the band-limited functions and their Hilbert transform. IEEE Trans. Instrum. Meas. **46**, 442 (1997)
49. A.V. Oppenheim, R.W. Schafer, *Discrete-time Signal Processing*, 2nd edn. (Prentice Hall, New Jersey, 1989)
50. R. Bracewell, *The Fourier Transform and Its Applications*, 2nd edn. (McGraw-Hill, Reading, 1986)
51. F.W. King, *Hilbert Transforms, Vols. 1 & 2*. Encyclopedia of Mathematics and Its Applications, vols. 124 & 125 (Cambridge University Press, Cambridge, 2009)
52. C. Torrence, G.P. Compo, A practical guide to wavelet analysis. Bull. Am. Meteorol. Soc. **79**, 61 (1998)
53. D. Jordan, R.W. Miksad, E.J. Powers, Implementation of the continuous wavelet transform for digital time series analysis. Rev. Sci. Instrum. **68**, 1484 (1997)
54. P.S. Addison, *The Illustrated Wavelet Transform Handbook* (Institute of Physics, Bristol, 2002)
55. H.-G. Stark, *Wavelets and Signal Processing. An Application-Based Introduction* (Springer, Berlin, 2005)
56. G. Kaiser, *A Friendly Guide to Wavelets* (Birkhäuser, Boston, 1994)
57. G. Kaiser, Physical wavelets and their sources: real physics in complex space-time. J. Phys. A, Math. Gen. **36**, R291 (2003). A physicist might find particular pleasure in physical (acoustic and electro-magnetic) wavelets discussed in this paper
58. J.F. Kirby, Which wavelet best reproduces the Fourier power spectrum? Comput. Geosci. **31**, 846 (2005)

Chapter 5
Statistical Analysis and Modeling of Data

We record the outcomes of physical measurements as signals (sequences of values), where we are not interested in each value in particular but the characteristics of the signal as a whole. Signals can be analyzed in the statistical sense, where the time ordering of data is irrelevant, or in the functional sense, where it becomes essential: then we imagine that the signal (the measurement of a quantity) originates in the source (the dynamical system), and we may be able to infer the properties of that system from the properties of the signal. In the following two chapters we introduce the basic methods of both approaches [1].

5.1 Basic Data Analysis

5.1.1 Probability Distributions

Figure 5.1 (left) shows the probability density function

$$p(x) = \sqrt{\frac{2}{\pi}} \frac{1}{\sigma^3} x^2 \exp\left(-x^2/2\sigma^2\right), \quad \sigma = 2, \tag{5.1}$$

describing the classical Maxwell velocity distribution, while Fig. 5.1 (right) shows the corresponding cumulative distribution function $P(x)$. Both have their usual meanings: the value $P(x)$ is equal to the probability $\mathcal{P}$ that a random variable X has a value smaller than x,

$$P(x) = \mathcal{P}(X < x), \qquad 1 - P(x) = \mathcal{P}(X \geq x), \quad P(0) = 0, \; P(\infty) = 1,$$

while the probability density $p(x) = \mathrm{d}P(x)/\mathrm{d}x$ is a measure of probability for X assuming a value on the interval $[x, x + \mathrm{d}x)$,

$$\mathcal{P}(a \leq X < b) = \int_a^b p(x)\,\mathrm{d}x = P(b) - P(a).$$

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9_5,

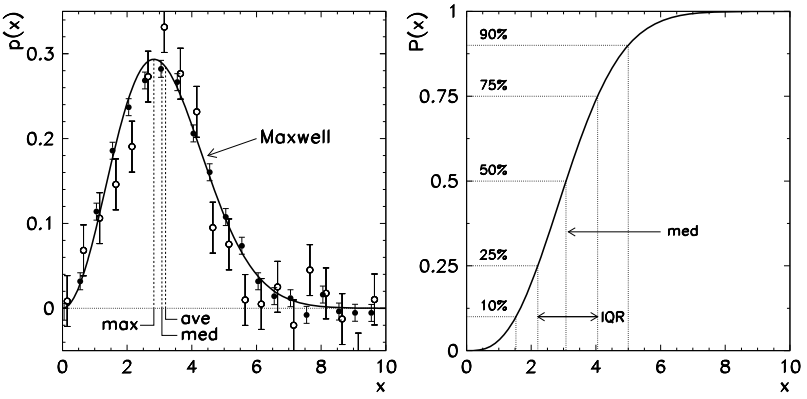


Fig. 5.1 Maxwell's distribution. [*Left*] Probability density (5.1) with the maximum (max), median (med), and average (ave) drawn in. [*Right*] Cumulative distribution function $P(x) = \int_0^x p(\xi)\,\mathrm{d}\xi$. Shown are the first and last decile, the first and last quartile with the inter-quartile range (IQR), and the median

A physicist dealing with the analysis of finite-size data samples typically encounters just a handful of probability distributions. But even within this limited set she sometimes finds it difficult to judge whether the measurements are in agreement with this or that known distribution, or if perhaps there is an unexpected deviation from it.

In addition to the analytic form of the probability density, Fig. 5.1 (left) also illustrates a typical situation with two sets of measurements (full and empty circles with uncertainties) generating the usual every-day questions. Are the two sets of data mutually consistent? How can they be compared at all? Is the arithmetic mean (ave), the median (med), or the maximum of the distribution (max) the best measure of consistency? Are the two sets of data scattered similarly? Is either of the sets consistent with the distribution that is expected (in this case, Maxwell's)? What can we do if a couple of data values appear to lie very far from the rest?

5.1.2 Moments of Distributions

The basic quantities that can be computed from the data set (the sample) $\boldsymbol{x} = \{x_i\}_{i=0}^{n-1}$ gathered from a very large population, are the arithmetic mean $\overline{x}$ and the variance (dispersion) s_x^2 or the standard deviation s_x,

$$\overline{x} = \frac{1}{n}\sum_{i=0}^{n-1} x_i, \qquad s_x^2 = \frac{1}{n-1}\sum_{i=0}^{n-1}(x_i - \overline{x})^2, \qquad s_x = \sqrt{s_x^2}. \tag{5.2}$$

The sample mean $\overline{x}$ is a non-biased estimate for the true average value of the whole population, $\langle x \rangle$, and the sample variance s_x^2 is a non-biased estimate for the true variance of the population, σ_x^2.

The average of an observable f with respect to the probability density p is

$$\langle f \rangle = \int f(x)\, p(x)\, \mathrm{d}x. \tag{5.3}$$

If the data x_i come from the population in which the elements are distributed according to the probability density p, the measures (5.2) are the approximations of their first two central moments

$$\langle x \rangle = \lim_{n\to\infty} \overline{x}, \qquad \sigma_x^2 = \left\langle \big(x - \langle x \rangle\big)^2 \right\rangle = \lim_{n\to\infty} s_x^2.$$

The convergence of the statistical measures $\overline{x}$ and s_x with increasing n depends on the properties of the data distribution. In certain cases these measures do not even exist. Assume that the data is spread throughout the whole real axis and that their probability density function p falls off as $\mathcal{O}(|x|^{-\rho})$, where $\rho > 1$. Then the average does not exist for $\rho \leq 2$ and the deviation does not exist for $\rho \leq 3$. If the first and the second central moments of a random variable are bounded and $n \gg 1$, the exact values can be related to the measured ones within some statistical confidence interval (see Sect. 5.1.3).

When we are dealing with histogrammed and non-normalized data, as in Fig. 5.1 (left), the estimates $\overline{x}$ and s_x^2 in the sense of the definition (5.3) can be computed by the weighted averages

$$\overline{x} = \frac{\sum_{i=0}^{n-1} x_i\, p(x_i)}{\sum_{i=0}^{n-1} p(x_i)}, \qquad s_x^2 = \frac{n}{n-1} \frac{\sum_{i=0}^{n-1} (x_i - \overline{x})^2\, p(x_i)}{\sum_{i=0}^{n-1} p(x_i)}. \tag{5.4}$$

5.1.3 Uncertainties of Moments of Distributions

In most cases we do not know the statistical nature of the individual measurements' uncertainties. We do assume that the errors behave randomly, so that each measurement consists of a true value $\langle x \rangle$ and a random error Δx_i,

$$x_i = \langle x \rangle + \Delta x_i, \tag{5.5}$$

but we do not know the random process that generates the error Δx_i. Physicists usually assume that Δx_i are normally distributed ("Gaussian"), so that the random variables of measurements x_i possess the probability density

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \langle x \rangle)^2}{2\sigma^2} \right), \tag{5.6}$$

which is defined by its moments $\langle x \rangle$ and $\sigma^2 = \langle (x - \langle x \rangle)^2 \rangle$. Then, for $n \gg 1$, $\overline{x}$ is also normally distributed, and for the estimates (5.2) we have the uncertainties

$$\Delta\overline{x} = \frac{s_x}{\sqrt{n}}, \qquad \Delta s_x^2 = s_x^2 \sqrt{\frac{2}{n-1}}, \qquad \Delta s_x = \frac{s_x}{\sqrt{2(n-1)}}. \tag{5.7}$$

We give the measurement results in the form $\langle x\rangle = \overline{x} \pm \Delta\overline{x}$, $\sigma^2 = s_x^2 \pm \Delta s_x^2$, $\sigma = s_x \pm \Delta s_x$, where the uncertainties correspond to confidence intervals (more will be said in Sect. 5.3 at Fig. 5.5). Note, however, that any sufficiently smooth observable f of the quantity x is distributed approximately normally, with the confidence interval

$$f\big(\langle x\rangle\big) \pm \left|\frac{\mathrm{d}f}{\mathrm{d}x}\big(\langle x\rangle\big)\right| \Delta x.$$

Only the local behavior of f near $\langle x\rangle$ is relevant for this estimate. The case of two noisy and statistically independent quantities $x \pm \Delta x$ and $y \pm \Delta y$ can be described by a function of two variables $f(x, y)$, and the variance computed as

$$(\Delta f)^2 = \left(\frac{\partial f(x,y)}{\partial x}\right)^2 (\Delta x)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2 (\Delta y)^2.$$

The estimates (5.2) and their uncertainties (5.7) apply to n independent measurements with equal measurement errors, from which we wish to infer the unknown quantities $\langle x\rangle$ and σ_x. If the values x_i have different but *known* errors σ_i, the weighted averaging (5.4) results in the non-biased estimate for $\langle x\rangle$ and its uncertainty

$$\overline{x} = \frac{1}{w}\sum_{i=0}^{n-1} w_i x_i, \quad w_i = \frac{1}{\sigma_i^2}, \; w = \sum_{i=0}^{n-1} w_i, \; \Delta\overline{x} = \frac{1}{\sqrt{w}}. \tag{5.8}$$

5.2 Robust Statistics

Sometimes the population sample contains values that clearly deviate from the majority of other values. They are known as *outliers* [2]. The outliers may represent errors in the measurement or genuine data that indeed strongly deviate from the rest. (Early data from the Nimbus 7 satellite hinted at the hole in the ozone layer above Antarctica, but the researchers discarded them as instrumental errors [3].) Robust statistics is a methodology to determine the parameters that characterize well the samples with relatively few outliers [4]. "Robustness" implies a small sensitivity of the estimated mean and variance to the inclusion or exclusion of some or all outliers from the sample.

Example The classical case leading us to consider the use of robust methods are the 24 measurements of copper content in bread flour [5, 6],

2.20 2.20 2.40 2.40 2.50 2.70 2.80 2.90 3.03 3.03 3.10 3.37
3.40 3.40 3.40 3.50 3.60 3.70 3.70 3.70 3.70 3.77 5.28 28.95,

shown in Fig. 5.2. The arithmetic mean of the whole sample is $\overline{x} = 4.28$ and the standard deviation is $s_x = 5.30$. If the value 28.95 is excluded from the sample, we

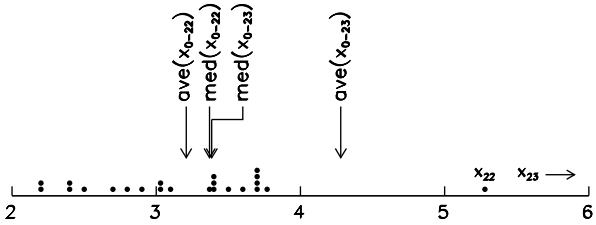


Fig. 5.2 The sample (24 values) of copper content in flour in µg/g. The value $x_{23} = 28.95$ (and potentially $x_{22} = 5.28$) represent outliers. The median (med) is much less sensitive to the exclusion of the extreme outlier than the arithmetic mean (ave)

get $\overline{x} = 3.21$ and $s_x = 0.69$. A single outlier therefore substantially modifies both $\overline{x}$ and s_x, so clearly these two quantities do not give us robust estimates for the properties of the complete population.

A much more robust measure for the "center" of the sample $\boldsymbol{x} = \{x_i\}_{i=0}^{n-1}$ is the *median*. It is defined as the value that divides the ordered sample $(x_i \leq x_{i+1})$ in two, such that its left and right portions contain half of the data each,

$$\text{median}(\boldsymbol{x}) = \begin{cases} x_{(n-1)/2}; & n \text{ odd}, \\ \frac{1}{2}(x_{n/2-1} + x_{n/2}); & n \text{ even}. \end{cases}$$

If the probability density has the form $p(x - \mu)$, where $\mu = \overline{x} = \text{median}(\boldsymbol{x})$, the variance of the sample median tends to $1/[4np(0)^2]$ if $p(0) > 0$ and $n \gg 1$. The spread of the data around the median can be estimated by the *median absolute deviation* (MAD)

$$\text{MAD}(\boldsymbol{x}) = \text{median}\big(\big|\boldsymbol{x} - \mathbf{1}_n \text{median}(\boldsymbol{x})\big|\big), \tag{5.9}$$

where $\mathbf{1}_n = \{1, 1, \ldots, 1\}$ is the sequence of values 1 with length n. It makes sense to define a quantity that can be directly compared to the standard deviation,

$$\text{MADN}(\boldsymbol{x}) = 1.4826\, \text{MAD}(\boldsymbol{x}).$$

Namely, the factor 1.4826 is determined such that for the normal distribution $N(\mu, \sigma^2)$, $\text{MADN} = \sigma$ holds. The values for all data in the flour sample are median = 3.385 and MADN = 0.526, while they are 3.37 and 0.504 if the outlier at 28.95 is excluded. Both values change only insignificantly (see Fig. 5.2).

It is clear from the uncertainties (5.7) that the arithmetic mean for $n \gg 1$ is distributed normally as $N(\mu, \sigma^2/n)$ if the measurement errors in (5.5) are distributed normally as $N(0, \sigma^2)$. With the same assumption for the errors, the median is distributed normally as $N(\mu, (\pi/2)\sigma^2/n)$, so the variance of the median for the normal distribution is $\pi/2 \approx 1.571$-times larger than the variance of the arithmetic mean. We say that the median in the case of the normal distribution has *low asymptotic efficiency* $\text{eff} \approx 1/1.571 \approx 64\,\%$.

Estimating the "center" and "spread" of data by using the median and MADN only weakly depends on the values in the distribution tails, but the computation of

Fig. 5.3 The box diagram for a visual identification of outliers. The outliers can be expected outside of the interval $[X_-, X_+] = [Q_1 - \frac{3}{2}\,\mathrm{IQR},\, Q_3 + \frac{3}{2}\,\mathrm{IQR}]$

both estimates is numerically more costly than the computation of $\overline{x}$ and s_x^2, since sorting the data requires $\mathcal{O}(n \log n)$ operations and $\mathcal{O}(n)$ of memory.

5.2.1 Hunting for Outliers

Chasing outliers in one dimension is but a tiny subgroup of activities related to the search for subsets of data that do not behave according to expectations or deviate from the bulk of the data set. In literature, these activities are known under the names *anomaly detection*, *outlier detection*, *novelty detection*, or *exception mining* (see the review articles [7–10]), and represent one of the tools of *data mining*.

The school-book way of eliminating outliers is the "3σ-rule". It suggests us to eliminate all data deviating from the sample mean $\overline{x}$ by more than $\pm 3s_x$, or to assign all such data the values $\overline{x} \pm 3s_x$. This method has numerous deficiencies. For example, it forces us to needlessly remove approximately three data points from an impeccable normally distributed sample of size $n = 1000$, since the interval $[\overline{x} - 3s_x, \overline{x} + 3s_x]$ for large n includes 99.7 % of data. Moreover, the computation of the mean and the variance themselves is sensitive to outliers. Instead, it is preferable to use the criterion

$$x_i \text{ outlier} \quad \Longleftrightarrow \quad \left| \frac{x_i - \mathrm{median}(\boldsymbol{x})}{\mathrm{MADN}(\boldsymbol{x})} \right| > 3.5. \tag{5.10}$$

A simple way to visually identify the outlier candidates is to draw a *box diagram*. First we compute the median of the sample and the first and third quartiles Q_1 and Q_3: this splits the sample into four compartments with a fourth of the data in each of them. Then the *inter-quartile range* (IQR) is formed, along with the boundaries X_- and X_+ beyond which outliers can be expected,

$$\mathrm{IQR} = Q_3 - Q_1, \qquad X_- = Q_1 - \frac{3}{2}\,\mathrm{IQR}, \qquad X_+ = Q_3 + \frac{3}{2}\,\mathrm{IQR}.$$

This method identifies the values x_{22} and x_{23} from the flour sample as outliers (see Fig. 5.3). The interval $[X_-, X_+]$ for large n contains 99.3 % of all data, and for the normal distribution, the method is approximately equivalent to the "3σ-rule".

In general, all statistical methods for the identification of outliers in the sample $\boldsymbol{x}$ already assume a certain probability distribution for the whole population (in most cases, normal). The outliers are sought in the range

$$\left\{x : |x - \widehat{\mu}_n| > \widehat{\sigma}_n g(n, \alpha_n)\right\}, \tag{5.11}$$

where $\widehat{\mu}_n$ is an estimate for the population average, $\widehat{\sigma}_n$ is the estimate for its dispersion, and the function $g(n, \alpha_n)$ depends on the sample size n. With certain approximations, all values x lying within (5.11) are considered as outliers with a probability of $1 - \alpha_n$. The criterion (5.10) is just (5.11) with $\widehat{\mu}_n = \text{median}(\boldsymbol{x})$, $\widehat{\sigma}_n = \text{MADN}(\boldsymbol{x})$, and $g(n, \alpha_n) = 3.5$, and is known as the Hampel identifier. A more precise definition of these quantities is given in [11, 12] from which we take (with minor simplifications) the formulas valid for $n \geq 10$:

$$g(n, 0.05) \approx 2.9 + 12.5(n - 5.5)^{-0.572}, \qquad g(n, 0.01) \approx 3.8 + 25.3(n - 7.0)^{-0.601}.$$

The criterion (5.11) also performs well if $\widehat{\mu}_n$ is set to the center of the data, while $\widehat{\sigma}_n$ is set to the length of the *shortest half-sample* of the size $[n/2] + 1$ (Rousseeuw identifier). The older methods which are still in broad use—partly due to ignorance of better options—are described in [13] and [14, 15].

5.2.2 M-Estimates of Location

Eliminating outliers is not the best idea since this procedure inevitably implies a loss of information [16]. In modern approaches we tend to keep the outliers but take them into account with a smaller weight. One such approach is followed by M-estimates [17]: these methods attempt to estimate the properties of data sets $\{x_i\}_{i=0}^{n-1}$ by satisfactorily and stably characterizing the majority of the data in the sample, regardless of the presence or absence of outliers. In general, the *M-estimate of location* $\widehat{\mu}$ that tells us something about the location of the bulk of the data, is defined as

$$\widehat{\mu} = \arg\min_{\mu} \sum_{i=0}^{n-1} \rho(x_i - \mu). \tag{5.12}$$

This notation means that the parameter $\widehat{\mu}$ is equal to μ which minimizes the sum on the right-hand side. Here $\rho(x) = -\log p(x)$, $p(x) = P'(x)$, and $P(x)$ is the cumulative distribution function, according to which the error Δx_i in (5.5) is distributed. The definition (5.12) follows from the requirement that the estimate $\widehat{\mu}$ maximizes the *likelihood function*; for details see [17]. If ρ is differentiable and $\psi(x) = \rho'(x)$, we can differentiate (5.12) and obtain

$$0 = \sum_{i=0}^{n-1} \psi(x_i - \widehat{\mu}). \tag{5.13}$$

Example In the measurement error Δx_i is normally distributed, as it is usually assumed, then $p(x) = \exp(-x^2/2)/\sqrt{2\pi}$, $\rho(x) = x^2/2$, and $\psi(x) = x$, so (5.12) and (5.13) simplify to

$$\widehat{\mu} = \arg\min_{\mu} \sum_{i=0}^{n-1} (x_i - \mu)^2, \qquad \sum_{i=0}^{n-1} (x_i - \widehat{\mu}) = 0.$$

Both equations relate well-known stories: the latter is obviously solved by the mean, $\widehat{\mu} = \overline{x}$, while the former tells us that this $\overline{x}$ also minimizes the sum of the squares of the differences between the measured data and $\overline{x}$. Similarly, one can show that the median minimizes the sum $\sum_{i=0}^{n-1} |x_i - \mu|$. The arithmetic mean and the median are just special cases of M-estimates.

M-estimates are devised as weighted averages and as such they are also computed [17]. For most realistic distributions $\psi(0) = 0$ and $\psi'(0)$ exists, so that ψ is approximately linear near the origin. Equation (5.13) can then be written as

$$\sum_{i=0}^{n-1} (x_i - \widehat{\mu}) W(x_i - \widehat{\mu}) = 0$$

or

$$\widehat{\mu} = \frac{\sum_{i=0}^{n-1} w_i x_i}{\sum_{i=0}^{n-1} w_i}, \quad w_i = W(x_i - \widehat{\mu}), \tag{5.14}$$

where W is the weight function. The essence of a good M-estimate is the choice of a weight function that sufficiently damps the outliers at large $|x|$. Many functions are in use, based on the presumed distributions of the measurement errors. In terms of robustness, good representatives are the Tukey function

$$\psi(t) = \begin{cases} t(1 - t^2/\tau^2)^2; & |t| \le \tau, \\ 0; & |t| > \tau, \end{cases} \qquad W(t) = \begin{cases} \psi(t)/t; & |t| \le \tau, \\ 0; & |t| > \tau \end{cases} \tag{5.15}$$

and the Cauchy function

$$\psi(t) = \frac{t}{1 + \frac{1}{2}t^2}, \qquad W(t) = \psi(t)/t, \tag{5.16}$$

shown in Fig. 5.4. When τ is increased in the Tukey function, more distant outliers are admitted while the robustness of the M-estimate is reduced; on the other hand, its asymptotic efficiency increases (and vice versa). The values $\tau = 3.44$, 3.88, and 4.68 correspond to asymptotic efficiencies eff = 85, 90, and 95 % (asymptotic efficiency has been defined on p. 211).

Equation (5.14) is implicit since the desired parameter $\widehat{\mu}$ occurs at left and in the weights w_i, but it can be solved iteratively [17]. The convergence is guaranteed by favorable properties of the function W. If we wish to compute the M-estimate of

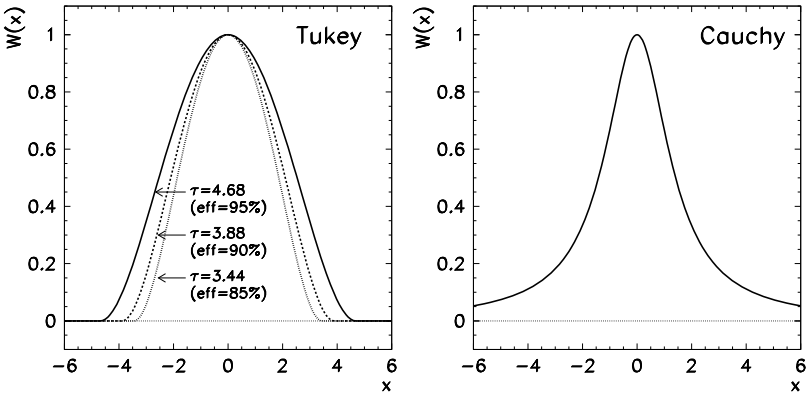


Fig. 5.4 Weight functions for M-estimates of location. [*Left*] Tukey function (5.15). The parameter τ determines the asymptotic efficiency. [*Right*] Cauchy function (5.16)

location in the case when the dispersion of the data $\widehat{\sigma}$ is already known, we define the weights

$$w_i^{(k)} = W\left(\frac{x_i - \widehat{\mu}^{(k)}}{\widehat{\sigma}}\right), \quad i = 0, 1, \ldots, n-1, \tag{5.17}$$

and rewrite (5.14) as an iteration

$$\widehat{\mu}^{(k+1)} = \frac{\sum_{i=0}^{n-1} w_i^{(k)} x_i}{\sum_{i=0}^{n-1} w_i^{(k)}}, \tag{5.18}$$

where k is the iteration index.
The algorithm to compute the M-estimate of location is then

Input: Values $\boldsymbol{x} = \{x_i\}_{i=0}^{n-1}$ and the relative precision of M-estimate ε
$k = 0$; $\widehat{\mu}^{(k)} = \text{median}(\boldsymbol{x})$; $\widehat{\sigma} = \text{MADN}(\boldsymbol{x})$;
while $(|\widehat{\mu}^{(k+1)} - \widehat{\mu}^{(k)}| > \varepsilon\widehat{\sigma}|)$ **do**
  Compute the weights (5.17) with W from (5.15) or (5.16);
  Compute the estimate (5.18);
  $k = k + 1$;
end
Output: M-estimate of location $\widehat{\mu}$

For the data in Fig. 5.2 and the Tukey function (parameter $\tau = 4.68$, asymptotic efficiency eff $= 95$ %) we get $\widehat{\mu} = 3.144$ for all data, $\widehat{\mu} = 3.143$ by omitting the extreme value 28.95, and $\widehat{\mu} = 3.127$ by omitting the value 5.28. (Check these numbers and monitor the usual mean $\overline{x}$ during this procedure!)

5.2.3 M-Estimates of Scale

Robust estimates of data dispersion are called *M-estimates of scale*. Analogously to M-estimates of location they are introduced through maximum likelihood functions [17]. In contrast to (5.5) one imagines that the measurements x_i are distributed around zero with some error $\sigma \Delta x_i$ and that Δx_i are distributed according to some probability density p, while σ is an unknown parameter measuring the scale of this error. The M-estimate of scale $\widehat{\sigma}$ is defined as

$$\widehat{\sigma} = \arg\max_{\sigma} \frac{1}{\sigma^n} \prod_{i=0}^{n-1} p\left(\frac{x_i}{\sigma}\right). \tag{5.19}$$

The parameter $\widehat{\sigma}$ is equal to σ maximizing the product at the right, while $\rho(x) = x\psi(x)$ and $\psi(x) = -p(x)'/p(x)$. If the error is normally distributed (as (5.6) with $\langle x \rangle = 0$ and $\sigma = 1$), we have $\rho(x) = x^2$ and (5.19) is solved by the *root mean square* $\widehat{\sigma} = \mathrm{RMS}(\boldsymbol{x}) = (\langle \boldsymbol{x}^2 \rangle)^{1/2}$. The measure of dispersion $\mathrm{RMS}(\boldsymbol{x})$ is therefore just a special case of M-estimates of scale.

In practice, we usually do not know the true probability density p for the distribution of errors. Still, we would like to ensure that the estimate of dispersion is robust with respect to potential outliers in the sample. M-estimates of scale are devised analogously to M-estimates of location, as weighted averages,

$$\widehat{\sigma} = \sqrt{\frac{c}{n} \sum_{i=0}^{n-1} w_i x_i^2}, \quad w_i = W\left(\frac{x_i}{\widehat{\sigma}}\right), \tag{5.20}$$

where the function W and the constant c depend on the model that we use to understand the source or the probability density of the errors Δx_i. If little is known about it, the following empirical formulas can be recommended [17]:

$$W(t) = \min\left\{1/t^2, t^4 - 3t^2 + 3\right\}, \qquad c = 2. \tag{5.21}$$

The parameter $\widehat{\sigma}$ in (5.20) again occurs at the left and in the weights w_i, so the equation is solved iteratively,

$$\widehat{\sigma}^{(k+1)} = \sqrt{\frac{c}{n} \sum_{i=0}^{n-1} w_i^{(k)} x_i^2}, \quad w_i^{(k)} = W\left(\frac{x_i}{\widehat{\sigma}^{(k)}}\right),$$

where k is the iteration index. We start the iteration with $\widehat{\sigma}^{(0)} = \mathrm{MADN}(\boldsymbol{x})$ and terminate it when $|\widehat{\sigma}^{(k+1)} - \widehat{\sigma}^{(k)}| < \varepsilon \widehat{\sigma}^{(k)}$. For normally distributed errors Δx_i, the estimate $\widehat{\sigma}$ becomes the standard deviation if we divide it by 1.56 [17].

5.3 Statistical Tests

In this section we describe basic statistical tests useful in quantifying the properties of various data samples or comparing the samples to each other. Some of the questions posed in Fig. 5.1 (left) will find their answers here. These standard procedures are founded and explained in greater detail in all good statistics textbooks; see e.g. [18].

5.3.1 Computing the Confidence Interval for the Sample Mean

For a set of uncorrelated data $\{x_i\}_{i=0}^{n-1}$ we have used (5.2) to compute the mean $\overline{x}$ and variance s_x^2. Assume that the measured data are normally distributed. We are seeking a quantitative measure to determine the quality of the approximation $\overline{x}$ for the true average $\langle x\rangle$. To do this, we use $\overline{x}$ and s_x^2 to form the statistic

$$t = \frac{\overline{x} - \langle x\rangle}{\sqrt{s_x^2}}\sqrt{n}.$$

If x_i are normally distributed according to $N(\langle x\rangle, \sigma^2)$, the statistic t is distributed according to $S(t; n-1)$ [19], where

$$S(t;\nu) = \frac{\mathrm{d}P}{\mathrm{d}t}(t;\nu) = \frac{1}{\sqrt{\nu}B(\frac{\nu}{2},\frac{1}{2})}\left(1+\frac{t^2}{\nu}\right)^{-(\nu+1)/2} \tag{5.22}$$

is the Student's distribution for ν *degrees of freedom* and $B(a,b)$ is the usual beta function. The number ν is not necessarily integer and it depends on the number of measurements and of the type of the statistical test. Examples are given later on. The integral of the probability density (5.22) enables us to determine the intervals on which, with some probability $1-\alpha$ chosen in advance, we may expect to find the values of t or $\langle x\rangle$, while with probability α the value $\langle x\rangle$ will be outside the corresponding interval. The Student's probability density function is even in t, so we define symmetric limits t_- and t_+ such that

$$\int_{t_-}^{t_+} \frac{\mathrm{d}P}{\mathrm{d}t}\,\mathrm{d}t = 1-\alpha, \quad -t_- = t_+ = t_*. \tag{5.23}$$

This means that we may believe, at *confidence level* $1-\alpha$, that $|t| \le t_*$, or that we may expect $|t| > t_\star$ with probability (*risk level*) α. The statistic t has therefore been bounded by the conditions

$$t_- \le \frac{\overline{x} - \langle x\rangle}{s_x}\sqrt{n} \le t_+.$$

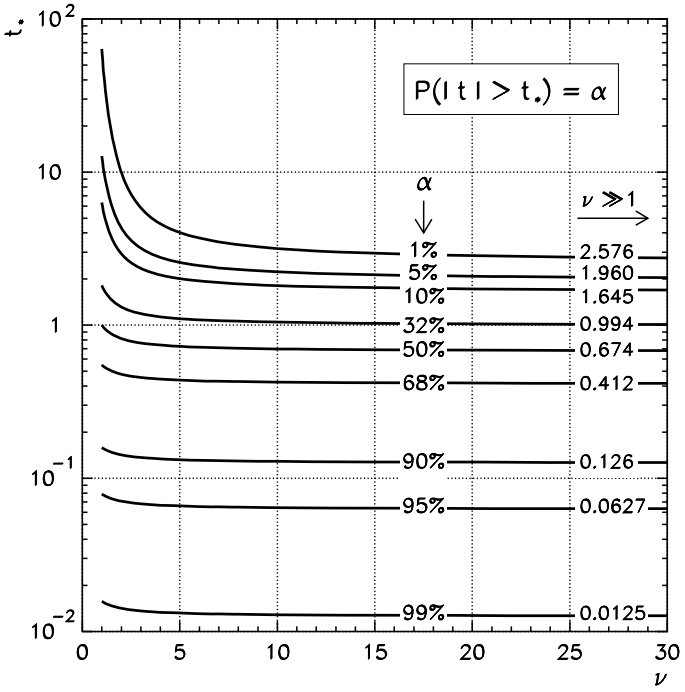


Fig. 5.5 Determining the confidence interval. The value t_* in (5.24) as a function of the number of degrees of freedom ν for various α. The values t_* in the limit $\nu \gg 1$ are shown at the extreme right. The risk level 31.7 % (confidence level 68.3 %) corresponds to $t_* \approx 1$ and the confidence interval (5.24) is $[\overline{x} - s_x/\sqrt{n}, \overline{x} + s_x/\sqrt{n}]$, as told by (5.7)

In other words, the true average of a large population, from which the sample $\{x_i\}_{i=0}^{n-1}$ has been acquired, can be estimated by $\langle x \rangle = \overline{x}$, and $\langle x \rangle$ will be found with probability $1 - \alpha$ on the confidence interval

$$\left[\overline{x} - \frac{t_* s_x}{\sqrt{n}}, \overline{x} + \frac{t_* s_x}{\sqrt{n}}\right]. \tag{5.24}$$

Figure 5.5 shows t_* in dependence of the number of degrees of freedom ν, for various values of α.

5.3.2 Comparing the Means of Two Samples with Equal Variances

The Student's t-test frequently appears in a different disguise. Assume we have two sets of data with equal variances, as would perhaps occur when using the same apparatus to measure a quantity which, we suspect, has changed between the two

sets of measurements. From the samples

$$\boldsymbol{x} = \{x_i\}_{i=0}^{n_x-1}, \qquad \boldsymbol{y} = \{y_i\}_{i=0}^{n_y-1},$$

we first form the means $\overline{x}$ and $\overline{y}$, then the statistics s_d and t_d,

$$s_\mathrm{d} = \left[\frac{\sum_{i=0}^{n_x-1}(x_i-\overline{x})^2 + \sum_{i=0}^{n_y-1}(y_i-\overline{y})^2}{n_x+n_y-2}\left(\frac{1}{n_x}+\frac{1}{n_y}\right)\right]^{1/2}, \qquad t_\mathrm{d} = \frac{\overline{x}-\overline{y}}{s_\mathrm{d}}.$$

With t_d we compute the value $0 \le \alpha \le 1$ by using the formula

$$\alpha = 1 - \int_{-|t_\mathrm{d}|}^{|t_\mathrm{d}|} \frac{\mathrm{d}P}{\mathrm{d}t}(t;\nu)\,\mathrm{d}t = \frac{B_x(\nu/2,1/2)}{B(\nu/2,1/2)}, \quad x = \frac{\nu}{\nu+t_\mathrm{d}^2}, \tag{5.25}$$

where $B(a,b)$ is the beta and $B_x(a,b)$ the incomplete beta function (accessible in standard numerical libraries) and we set $\nu = n_x + n_y - 2$. The value α ($0 \le \alpha \le 1$) measures the probability that for samples $\boldsymbol{x}$ and $\boldsymbol{y}$, the situation $|t| \ge |t_\mathrm{d}|$ occurs by chance. The smaller the value of α, the more statistically significant is the estimated difference of the means $\overline{x} - \overline{y}$. Computing α by (5.25) is simpler than the inverse task (5.23) of computing the integration limits t_* for a given α.

5.3.3 *Comparing the Means of Two Samples with Different Variances*

The Student's t-test can also be used on two samples with different variances. We face this situation when a quantity is measured by different devices (for example, one more and one less precise) and we wish to know whether the average of this quantity has changed between the two experiments. From the samples

$$\boldsymbol{x} = \{x_i\}_{i=0}^{n_x-1}, \qquad \boldsymbol{y} = \{y_i\}_{i=0}^{n_y-1},$$

we form the means $\overline{x}$ and $\overline{y}$, the variances s_x^2 and s_y^2, and the statistic

$$t = \frac{\overline{x}-\overline{y}}{\sqrt{s_x^2/n_x + s_y^2/n_y}}.$$

The statistic t is distributed approximately according to the Student's distribution (5.22) with the number of degrees of freedom

$$\nu = \left[\frac{s_x^2}{n_x}+\frac{s_y^2}{n_y}\right]^2\left[\frac{1}{n_x-1}\left(\frac{s_x^2}{n_x}\right)^2 + \frac{1}{n_y-1}\left(\frac{s_y^2}{n_y}\right)^2\right]^{-1}.$$

5.3.4 Determining the Confidence Interval for the Sample Variance

This unit complements Sect. 5.3.1 where we have determined the confidence interval for the mean of presumably normally distributed uncorrelated data $\{x_i\}_{i=0}^{n-1}$. Again we compute the sample mean and variance, $\overline{x}$ and s_x^2, and form the statistic

$$\chi^2 = \frac{(n-1)s_x^2}{\Sigma^2}.$$

Here Σ^2 is the sought parameter for which the confidence interval should be found, and thus allow us to gauge the reliability of the dispersion estimate s_x^2. The statistic χ^2 is distributed with the density $\mathrm{d}P/\mathrm{d}\chi^2(\chi^2; n-1)$, where

$$\frac{\mathrm{d}P}{\mathrm{d}\chi^2}(\chi^2; \nu) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)}(\chi^2)^{\nu/2-1}\mathrm{e}^{-\chi^2/2}, \quad \chi^2 \geq 0, \tag{5.26}$$

and where ν is the number of degrees of freedom. For large ν the χ^2 distribution resembles the Gaussian distribution with the mean ν and variance 2ν. We define the limits χ_-^2 and χ_+^2 such that the integrals of the extreme lower and the extreme upper tail of the distribution are equal,

$$\int_0^{\chi_-^2} \frac{\mathrm{d}P}{\mathrm{d}\chi^2}\,\mathrm{d}\chi^2 = \int_{\chi_+^2}^{\infty} \frac{\mathrm{d}P}{\mathrm{d}\chi^2}\,\mathrm{d}\chi^2 = \frac{\alpha}{2}. \tag{5.27}$$

Here α is the chosen risk level quantifying the expectation that χ^2 will by chance lie outside of the confidence interval, that is, below χ_-^2 or above χ_+^2. (Compare (5.27) to (5.23).) The variance estimate is $\Sigma^2 = s_x^2$ and we anticipate, at $1-\alpha$ confidence level, that the true variance σ_x^2 is within the confidence interval

$$\frac{(n-1)s_x^2}{\chi_+^2} \leq \sigma_x^2 \leq \frac{(n-1)s_x^2}{\chi_-^2}.$$

The values χ_-^2 and χ_+^2 solving (5.27) can be computed by standard numerical packages, while for everyday use they can be read off from Figs. 5.6 or 5.12.

Example Assume that we have used (5.2) to estimate the variance s_x^2 of a sample $\{x_i\}$ of $n = 11$ values. We would like to determine the interval to which, at 90 % confidence level, the true variance σ_x^2 can be constrained. We use Fig. 5.6 to locate the curve corresponding to $\nu = n - 1 = 10$ degrees of freedom. The χ_-^2 and χ_+^2 are found as values of χ^2 on the abscissa where the horizontal lines at $\alpha = 0.05$ and $\alpha = 0.95$ intersect this curve: in this case $\approx$18.3 and $\approx$3.94. The risk level of 10 % thus corresponds to the confidence interval

$$\frac{10s_x^2}{18.3} \leq \sigma_x^2 \leq \frac{10s_x^2}{3.94}. \tag{5.28}$$

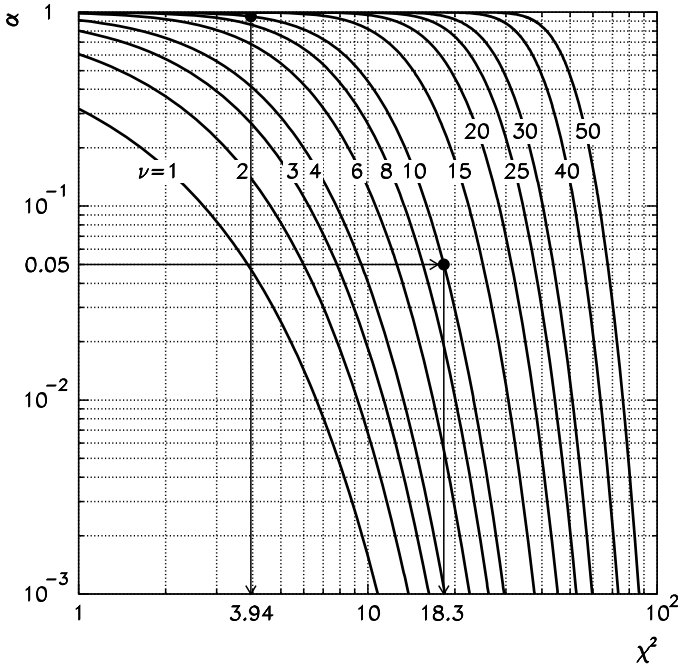


Fig. 5.6 The solution of $P(\chi^2 > \chi^2_+) = \alpha$ and $P(\chi^2 < \chi^2_-) = \alpha$. The *symbols* • denote the points $(\chi^2_-, 1-\alpha) = (3.94, 0.95)$ and $(\chi^2_+, \alpha) = (18.3, 0.05)$ from Example (5.28)

See also Fig. 5.12.

5.3.5 *Comparing Two Sample Variances*

Another standard task is the comparison of variances of two data samples

$$\boldsymbol{x} = \{x_i\}_{i=0}^{n_x-1}, \qquad \boldsymbol{y} = \{y_i\}_{i=0}^{n_y-1}.$$

We compute their sample variances s_x^2 and s_y^2, and form the ratio

$$F = s_x^2/s_y^2. \tag{5.29}$$

The larger variance should be put in the numerator and the smaller in the denominator. (If needed, rename the data $\boldsymbol{x} \leftrightarrow \boldsymbol{y}$.) The ratio F is distributed according to $\mathrm{d}P/\mathrm{d}F(n_x - 1, n_y - 1)$, where

$$\frac{\mathrm{d}P}{\mathrm{d}F}(\nu_1, \nu_2) = \left(\frac{\nu_1}{\nu_2}\right)^{\nu_1/2} \frac{\Gamma((\nu_1+\nu_2)/2)}{\Gamma(\nu_1/2)\Gamma(\nu_2/2)} F^{\nu_1/2-1} \left(1 + \frac{\nu_1}{\nu_2} F\right)^{-(\nu_1+\nu_2)/2}.$$

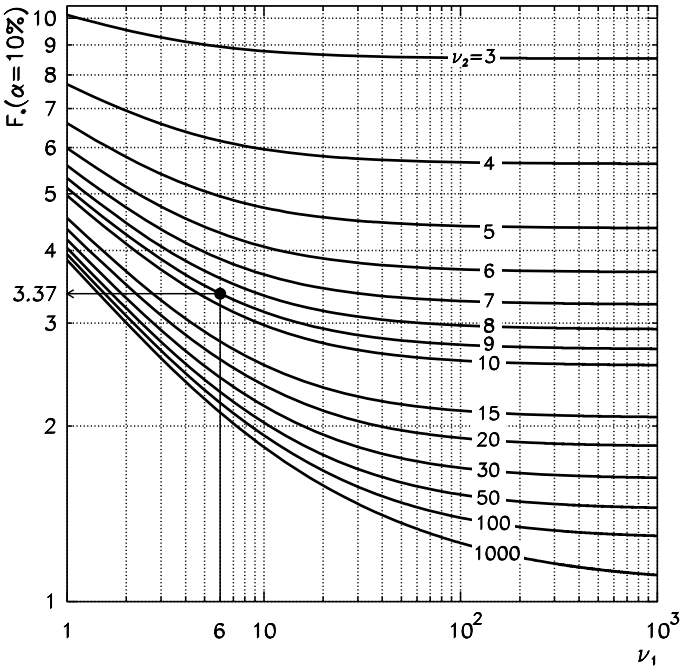


Fig. 5.7 The solutions of (5.30) for $\alpha = 10$ % for various degrees of freedom ν_1 and ν_2. The *symbol* • denotes the value $F_* = 3.37$ ($\nu_1 = 6$, $\nu_2 = 9$) for Example (5.31)

Similar to the previous tests, we define the limit F_* for which

$$\int_0^{F_*} \frac{\mathrm{d}P}{\mathrm{d}F}(\nu_1, \nu_2)\,\mathrm{d}F = 1 - \frac{\alpha}{2}, \tag{5.30}$$

and where α is the confidence level or *significance*. The values F_* at $\alpha = 10$ % in dependence of ν_1 and ν_2 are shown in Fig. 5.7. The assumption that the true variances σ_x^2 and σ_y^2 are equal should be rejected at confidence level α if the value F from (5.29) is larger than F_*.

Example (Taken from [19], p. 217) We have two samples of sizes $n_x = 10$ and $n_y = 7$,

$$\begin{aligned} \boldsymbol{x} &= \{100, 101, 103, 98, 97, 98, 102, 101, 99, 101\}, \\ \boldsymbol{y} &= \{97, 102, 103, 96, 100, 101, 100\}, \end{aligned} \tag{5.31}$$

with the means $\overline{x} = 100.0$ and $\overline{y} = 99.8$, and the sample variances $s_x^2 = 3.78$ and $s_y^2 = 6.50$, respectively. Is the deviation of the variance ratio $F = s_y^2/s_x^2 = 1.72$ from unity statistically significant, at significance $\alpha = 10$ %? From Fig. 5.7 we read off the value F_* along the curve $\nu_2 = \nu_x = n_x - 1 = 9$ at the abscissa $\nu_1 = \nu_y = n_y -$

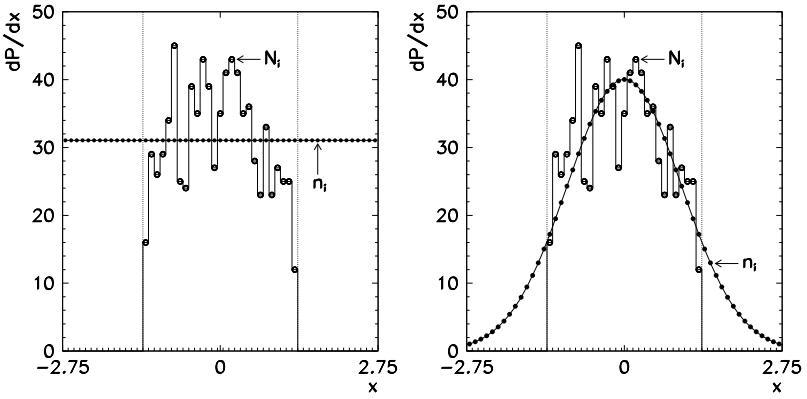


Fig. 5.8 Consistency of histogrammed data and a known distribution. [*Left*] Comparison to the uniform distribution. [*Right*] Comparison to the normal distribution

$1 = 6$. We get $F_* = 3.37$. Since $F > F_*$ *does not* hold, the hypothesis $\sigma_x^2 = \sigma_y^2$ cannot be discarded.

5.3.6 Comparing Histogrammed Data to a Known Distribution

Assume that we can measure the distribution of a physical quantity with respect to x that may have values from $[-2.75, 2.75]$. Suppose that the experimental apparatus restricts us in a way that only allows us to acquire data on a narrower interval $[-1.35, 1.35]$. We display the results on this interval as a histogram with $\mathcal{N} = 27$ bins, as shown in Fig. 5.8. Is the measured distribution consistent with the theoretically predicted uniform distribution?

We use N_i to denote the number of measured events in the ith bin, while n_i denotes the corresponding theoretically anticipated number. We form the statistic

$$\chi^2 = \sum_{i=1}^{\mathcal{N}} \frac{(N_i - n_i)^2}{n_i},$$

where the sum runs over all bins $\mathcal{N}$. The number of all events is $N = \sum_{i=1}^{\mathcal{N}} N_i$. In the limit $N \gg 1$, χ^2 is distributed according to $\mathrm{d}P/\mathrm{d}\chi^2(\chi^2; \mathcal{N} - 1)$ (see (5.26)). We choose a risk level α at which the assumed distribution is discarded even though it is correct. With the chosen α we determine χ_+^2 in the cumulative distribution

$$\int_{\chi_+^2}^{\infty} \frac{\mathrm{d}P}{\mathrm{d}\chi^2}\left(\chi^2; \mathcal{N} - 1\right) \mathrm{d}\chi^2 = \alpha.$$

If the assumed distribution with the values n_i is consistent with the measured data N_i, we may expect $\chi^2 < \chi_+^2$ at confidence level $1 - \alpha$, while the outcome $\chi^2 >$

χ^2_+ leads to the conclusion that the theoretical distribution is inconsistent with the experimental one. The *reduced value* $\chi^2_+/(\mathcal{N}-1)$ for some typical values of α can be read off from Fig. 5.12.

Example Let us discuss the data in Fig. 5.8. The total number of events in all bins is $N = 838$. Is the measured distribution consistent with the uniform distribution, according to which we would expect $n_i = N/\mathcal{N} = 31.04$? We compute $\chi^2 = 59.38$ or, for $\mathcal{N} - 1 = 26$ degrees of freedom, $\chi^2/(\mathcal{N}-1) = 2.28$. At $\alpha = 5$ % we read off $\chi^2_+/(\mathcal{N}-1) \approx 1.5$ from Fig. 5.12, while for $\alpha = 1$ % we find $\chi^2_+/(\mathcal{N}-1) \approx 1.8$. In any case we obtain $\chi^2 > \chi^2_+$ which points to the conclusion that the uniform distribution is inconsistent with the measured distribution.

We have narrowed the range in Fig. 5.8 on purpose: this is what one often finds in practice, forcing us to see the data as nothing but "a constant". What do we get with the values n_i corresponding to the normal distribution $N(0, 1)$? Now we compute $\chi^2 = 35.66$ or $\chi^2/(\mathcal{N}-1) = 1.37$, which is less than $\chi^2_+(\alpha = 5\ \%)$ and $\chi^2_+(\alpha = 1\ \%)$. At confidence level of at least 99 % we may therefore claim that the measured and the assumed theoretical distribution are consistent.

The value of χ^2 depends on the number of bins $\mathcal{N}$. The classic choice [20] for the number of bins $\mathcal{N}_{\text{opt}}$ that makes the χ^2-test optimally sensitive is $\mathcal{N}_{\text{opt}} \approx 4\, N^{2/5}$ (at $\alpha = 1$ %), but see [21, 22] for modern alternatives.

5.3.7 Comparing Two Sets of Histogrammed Data

The χ^2-test described above can also be used to question the mutual consistency of two sets of histogrammed data N_i and M_i in bins $i = 1, 2, \ldots, \mathcal{N}$. In this case, the statistic χ^2 should be defined as

$$\chi^2 = \sum_{i=1}^{\mathcal{N}} \frac{(\sqrt{M/N}N_i - \sqrt{N/M}M_i)^2}{N_i + M_i}, \quad N = \sum_{i=1}^{\mathcal{N}} N_i,\ M = \sum_{i=1}^{\mathcal{N}} M_i.$$

In general $N \neq M$. The χ^2-test with the chosen risk level α is performed as before, with the number of degrees of freedom $\mathcal{N} - 1$. The values $\chi^2 > \chi^2_+$ indicate that the measured data N_i and M_i originate in different distributions.

5.3.8 Comparing Non-histogrammed Data to a Continuous Distribution

If we wish to compare non-histogrammed data to a continuous distribution on the domain that is identical to the data range, we resort to the Kolmogorov–Smirnov

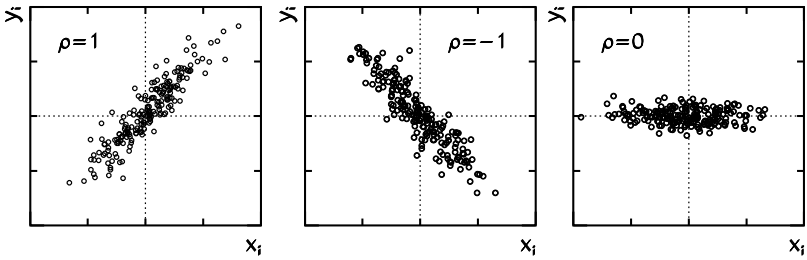


Fig. 5.9 Typical images of almost complete correlation ($\rho \approx 1$), almost complete anti-correlation ($\rho \approx -1$), and almost uncorrelated data ($\rho \approx 0$) in the (x_i, y_i) plane

test. As the statistic, this test uses the maximum distance between the cumulative distributions of the data and the assumed continuous distribution. (Both the data and the distribution to which they are being compared can always be histogrammed, so the previously described χ^2-test can also be applied, but direct comparison has certain advantages. For details, see [23].)

5.4 Correlation

Here we discuss measures of correlation between data sets. The correlation strength is measured by correlation coefficients, while we use suitable statistics to confirm whether the observed correlation is statistically significant or not.

5.4.1 Linear Correlation

The basic measure for the degree of correlation between two data sets is the linear correlation coefficient ρ. The correlatedness of two-dimensional data sometimes simply "pops out": typical images in the (x_i, y_i) plane for correlation coefficients $\rho \approx 1$ (almost complete positive correlation), $\rho \approx -1$ (almost complete anti-correlation), or $\rho \approx 0$ (uncorrelated data) are shown in Fig. 5.9.

The estimate for the linear correlation between the data $\{x_i\}_{i=0}^{n-1}$ and $\{y_i\}_{i=0}^{n-1}$ is

$$\widehat{\rho} = \frac{\sum_{i=0}^{n-1}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=0}^{n-1}(x_i - \overline{x})^2}\sqrt{\sum_{i=0}^{n-1}(y_i - \overline{y})^2}}, \quad -1 \le \widehat{\rho} \le 1. \tag{5.32}$$

The coefficient (5.32) is appropriate for the estimate of the degree of correlation once we have already confirmed that the correlation exists and that it has a certain statistical significance. Just like for the sample mean and variance, we can determine the confidence interval for the sample correlation coefficient $\widehat{\rho}$. In order to do this,

we use the statistic

$$z = \frac{1}{2} \ln \frac{1+\widehat{\rho}}{1-\widehat{\rho}} = \operatorname{Atanh} \widehat{\rho}$$

and assume that the measured data x_i and y_i are distributed according to the binormal (two-dimensional normal) distribution. For sample sizes n of more than a few times 10, the statistic z is then approximately normally distributed, with

$$N\left(\overline{z}, \sigma_z^2\right) = N\left(\frac{1}{2}\left[\ln \frac{1+\rho}{1-\rho} + \frac{\rho}{n-1}\right], \frac{1}{n-3}\right),$$

where ρ is the true correlation coefficient. The best estimate for the correlation coefficient is simply $\rho = \widehat{\rho}$, while the significance level at which we may claim that the measured $\widehat{\rho}$ differs from ρ, is given by

$$\alpha = 1 - \operatorname{erf}\left(\frac{|z - \overline{z}|\sqrt{n-3}}{\sqrt{2}}\right).$$

In determining whether the measurements of the quantities x_i and y_i from two time periods ("1" and "2") are correlated differently, we compare the correlation coefficients $\widehat{\rho}_1$ and $\widehat{\rho}_2$. The statistical significance of the difference between $\widehat{\rho}_1$ and $\widehat{\rho}_2$ is

$$\alpha = 1 - \operatorname{erf}\left(\frac{|z_1 - z_2|}{\sqrt{2}} \sqrt{\frac{(n_1-3)(n_2-3)}{n_1+n_2-6}}\right).$$

We may also ask the inverse question: to what confidence interval $[\rho_-, \rho_+]$ will the correlation coefficient be restricted at confidence level $1-\alpha$? For $1-\alpha \approx 96\ \%$, which is appropriate for everyday use, the values ρ_- and ρ_+ can be computed as

$$\rho_- = \tanh\left(\operatorname{Atanh} \widehat{\rho} - \frac{2}{\sqrt{n}}\right), \qquad \rho_+ = \tanh\left(\operatorname{Atanh} \widehat{\rho} + \frac{2}{\sqrt{n}}\right).$$

5.4.2 Non-parametric Correlation

The formula for the linear correlation coefficient (5.32) involves the sample means $\overline{x}$ and $\overline{y}$ which are strongly sensitive to the presence of outliers (see Sect. 5.2). We need a more robust tool. One option is to define the correlation by referring only to the positions (ranks) r_i and s_i that a certain x_i and some y_i occupy in the ordered samples $\boldsymbol{x}$ and $\boldsymbol{y}$. When more (for instance, m) equal values share m positions, we assign to all these values an average rank that they would have, had they been only slightly different. In addition, we compute the mean ranks $\overline{r} = (\sum_{i=1}^{n} r_i)/n$ and $\overline{s} = (\sum_{i=1}^{n} s_i)/n$ (the ranks are indexed from 1 upwards).

Example Determine the mean rank of the sample $\{x_i\}_{i=0}^{7} = \{2, 3, 9, 3, 4, 9, 7, 3\}$! We first order the sample and obtain the array $\{x_0, x_1, x_3, x_7, x_4, x_6, x_2, x_5\}$. The values $x_1 = x_3 = x_7 = 3$ share the ranks 2 to 4, so their mean rank is $(2 + 3 + 4)/3 = 3$. The values $x_2 = x_5 = 9$ share the ranks 7 and 8, so their rank is 7.5. Finally we obtain $\{r_i\}_{i=1}^{8} = \{1, 3, 3, 3, 5, 6, 7.5, 7.5\}$ and $\overline{r} = 4.5$.

We use the ranks r_i and s_i as well as the mean ranks $\overline{r}$ and $\overline{s}$ to define the *rank correlation coefficient*

$$\widehat{\rho}_{\mathrm{p}} = \frac{\sum_{i=1}^{n}(r_i - \overline{r})(s_i - \overline{s})}{\sqrt{\sum_{i=1}^{n}(r_i - \overline{r})^2}\sqrt{\sum_{i=1}^{n}(s_i - \overline{s})^2}}. \tag{5.33}$$

In computing $\widehat{\rho}_{\mathrm{p}}$ we are only referring to the mutual placement of the data, hence this type of correlation estimate is called non-parametric. The distribution of the ranked data is uniform, and if the sample contains relatively few repeat values, the estimate (5.33) is much more robust than (5.32). The statistical significance of a measured correlation coefficient $\widehat{\rho}_{\mathrm{p}} \neq 0$ can be established by the t-test. We form the statistic

$$t_{\mathrm{p}} = \widehat{\rho}_{\mathrm{p}}\sqrt{\frac{n-2}{1-\widehat{\rho}_{\mathrm{p}}^2}},$$

which is distributed approximately according to the Student's distribution (5.22) with $n - 2$ degrees of freedom. The confidence level at which the assumption that the measured correlation coefficient $\widehat{\rho}_{\mathrm{p}}$ is equal to the true coefficient ρ_{p} can be discarded, is computed by (5.25) in which we use t_{p} instead of t_{d} and set $\nu = n - 2$.

5.5 Linear and Non-linear Regression

On an almost daily basis, we encounter the problem of fitting a smooth curve to a set of values

$$(x_i, y_i), \quad i = 0, 1, \ldots, n-1. \tag{5.34}$$

The curve fitted to the data is specified by its analytic form that contains a certain set of model parameters. The fitting procedure as well as the final values of these parameters should somehow reflect the precision (the uncertainties) of the data. The search for the appropriate model curve is known as *regression*. According to the linear or non-linear dependence of the fitting curve with respect to the model parameters, we distinguish linear and non-linear regression.

5.5.1 Linear Regression

In linear regression, the fitting curve (model) linearly depends on the parameters characterizing it. We use this type of regression when we wish to find a polynomial

$$f(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0,$$

that fits the data (5.34) as well as possible. One assumes that the points y_i are realizations of a random variable that is distributed around the unknown exact value with an absolute error σ_i. We wish to perform the fitting such that the sum of the squares of the errors $(y_i - f(x_i))^2$ with respect to the uncertainties σ_i will be as small as possible. If the fitting function is linear ($a_i = 0$ for $i \geq 2$), we are dealing with "straight-line" linear regression, while if it is a general polynomial, we are referring to polynomial (or general linear) regression, as the dependence on the model parameter is still linear.

The posed problem does not have a unique solution, since many measures of deviation can be devised. However, the *least-squares method* mentioned above is by far the most popular. In this case, we are seeking the parameters a_j ($j = 0, 1, \ldots, m$) that minimize the weighted sum of the squares of differences between the polynomial $f(x_i)$ and the values y_i,

$$\chi^2 = \sum_{i=0}^{n-1} \frac{(y_i - f(x_i))^2}{\sigma_i^2}. \tag{5.35}$$

The deviation is "punished" in proportion to the inverse of the absolute error in y_i, i.e. σ_i. The measure of deviation χ^2 is minimized by fulfilling the requirements

$$\frac{\partial \chi^2}{\partial a_j} = 0, \quad j = 0, 1, \ldots, m. \tag{5.36}$$

This translates into a system of linear equations for the parameters a_j,

$$\sum_{j=0}^{m} \Sigma_{kj} a_j = b_k, \quad \Sigma_{kj} = \sum_{i=0}^{n-1} \frac{x_i^{k+j}}{\sigma_i^2}, \; b_k = \sum_{i=0}^{n-1} \frac{x_i^k y_i}{\sigma_i^2}.$$

We introduce the vectors of parameters $a = (a_j)_{j=0}^m$ and coefficients $b = (b_j)_{j=0}^m$, as well as the matrix $\Sigma = [\Sigma_{kj}]_{k,j=0}^m$, so the system can be written in matrix form,

$$\Sigma a = b \quad \text{or} \quad a = \Sigma^{-1} b. \tag{5.37}$$

By using the Vandermonde matrix $V = [V_{ij}]$, where $V_{ij} = x_i^j$ ($0 \leq i \leq n-1$, $0 \leq j \leq m$), and the weight matrix $D = \mathrm{diag}(\sigma_i^{-2})_{i=0}^{n-1}$, the system is written as

$$V^{\mathrm{T}} D V a = V^{\mathrm{T}} D y, \qquad \Sigma = V^{\mathrm{T}} D V, \qquad b = V^{\mathrm{T}} D y,$$

where $y = (y_i)_{i=0}^{n-1}$. The parameters a_j depend on the values y_i which are statistically distributed around the exact values. Assuming that y_i are good approximations of the exact values, the variance of a_j can be estimated as

$$\sigma^2(a_j) = \left(\Sigma^{-1}\right)_{jj}. \tag{5.38}$$

The matrix Σ tends to be very poorly conditioned, since its condition number grows exponentially with its dimension, $\kappa(\Sigma) = C\exp(\mathcal{O}(n))$ (Sect. 3.2.5). If we assume $\kappa(\Sigma) = C\exp(\alpha n)$, the desired relative precision of the coefficients a_j is ε, and the arithmetic precision is ε_{M}, we may include polynomials of degrees $n \le (1/\alpha)\log(\varepsilon_{\mathrm{M}}/C\varepsilon)$. In double-precision floating-point arithmetic this typically means $n \le 10$.

5.5.2 Regression with Orthogonal Polynomials

At least some stability problems can be avoided if the points $\{x_i\}_{i=0}^{n-1}$ coincide with the definition domain of some system of orthogonal polynomials. The most useful in regression are *orthogonal polynomials of a discrete variable*. These are polynomials $\{p_k(x) : k = \mathrm{degree}(p_k)\}_{k=0}^{m}$ that are linearly independent and orthogonal on a discrete set of points $\{x_i\}$ in the sense

$$\sum_{i=0}^{n-1} \frac{1}{\sigma_i^2} p_k(x_i)\, p_l(x_i) = A_k \delta_{k,l}$$

with some weight $1/\sigma_i^2$. The model function fitted to the data can be designed as the linear combination $f(x) = \sum_{k=0}^{m} a_k p_k(x)$. We determine the expansion coefficients a_k by minimizing the measure of deviation (5.35). We obtain

$$\chi^2 = \sum_{k=0}^{m} a_k^2 A_k - 2\sum_{k=0}^{m} a_k B_k + C, \quad B_k = \sum_{i=0}^{n-1} \frac{p_k(x_i) y_i}{\sigma_i^2}, \; C = \sum_{i=0}^{n-1} \frac{y_i^2}{\sigma_i^2}.$$

From the condition for the minimum $\partial\chi^2/\partial a_k = 0$ we get $2a_k A_k - 2B_k = 0$ or

$$a_k = \frac{B_k}{A_k} \quad \text{and} \quad \chi^2 = C - \sum_{k=0}^{m} \frac{B_k^2}{A_k} = \min.$$

Example A most popular system of orthogonal polynomials of a discrete variable are the Chebyshev polynomials (4.36) that exhibit orthogonality by points (see (4.38)). A linear combination $f(x) = \sum_{k=0}^{m} a_k T_k(x)$ of these polynomials minimizes the measure of deviation $\chi^2 = \sum_{i=0}^{n-1} (y_i - f(x_i))^2$ with the coefficients

$$a_0 = \frac{1}{n}\sum_{i=0}^{n-1} y(x_i), \qquad a_k = \frac{2}{n}\sum_{i=0}^{n-1} y(x_i) T_k(x_i), \quad 1 \le k \le m,$$

which is known as the Chebyshev approximation formula. (The problem is well defined for $m+1 \leq n$. When $m+1=n$ the function f interpolates the data y_i and therefore $\chi^2 = 0$.) Chebyshev polynomials have good approximation properties (Sect. 1.2.1) and can be efficiently computed by (4.37), and are therefore frequently used for the solution of least-squares problems [24].

5.5.3 Linear Regression (Fitting a Straight Line)

In the case of linear regression we are seeking the straight line $f(x) = a_1 x + a_0$ that best fits the data (x_i, y_i) with known uncertainties (errors) σ_i. When χ^2 (see (5.35)) is minimized with respect to the parameters a_0 and a_1, we obtain an analytically solvable system of equations

$$\begin{aligned} a_0 S + a_1 S_x &= S_y, \\ a_0 S_x + a_1 S_{xx} &= S_{xy}, \end{aligned}$$

where we have denoted

$$S = \sum_{i=0}^{n-1} \frac{1}{\sigma_i^2}, \qquad S_x = \sum_{i=0}^{n-1} \frac{x_i}{\sigma_i^2}, \qquad S_{xx} = \sum_{i=0}^{n-1} \frac{x_i^2}{\sigma_i^2},$$

$$S_{xy} = \sum_{i=0}^{n-1} \frac{x_i y_i}{\sigma_i^2}, \qquad S_y = \sum_{i=0}^{n-1} \frac{y_i}{\sigma_i^2}.$$

In this system, we immediately recognize the matrix Σ from (5.37) and its inverse,

$$\Sigma = \begin{pmatrix} S & S_x \\ S_x & S_{xx} \end{pmatrix}, \qquad \Sigma^{-1} = \frac{1}{\det(\Sigma)} \begin{pmatrix} S_{xx} & -S_x \\ -S_x & S \end{pmatrix},$$

where we have assumed $\det(\Sigma) = S_{xx} S - S_x^2 \neq 0$. From (5.37) it follows that the coefficients a_0 and a_1 minimizing the measure of deviation χ^2 are

$$a_0 = \frac{S_{xx} S_y - S_x S_{xy}}{S_{xx} S - S_x^2}, \qquad a_1 = \frac{S S_{xy} - S_x S_y}{S_{xx} S - S_x^2}.$$

An example of fitting is shown in Fig. 5.10 (left).

If the errors are constant ($\sigma_i = \sigma$), the parameters of the straight line are $a_1 = \mathrm{cov}(x, y)/s_x^2$, $a_0 = \overline{y} - a_1 \overline{x}$, where $\overline{x} = (\sum_{i=0}^{n-1} x_i)/n$ and $\overline{y} = (\sum_{i=0}^{n-1} y_i)/n$ are the arithmetic means of the data, while

$$s_x^2 = \frac{1}{n-1} \sum_{i=0}^{n-1} (x_i - \overline{x})^2 \quad \text{and} \quad \mathrm{cov}(x, y) = \frac{1}{n-1} \sum_{i=0}^{n-1} (x_i - \overline{x})(y_i - \overline{y})$$

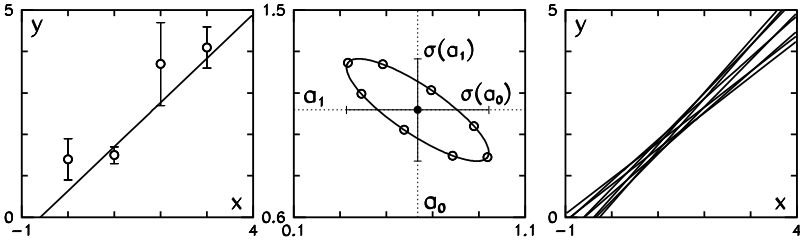


Fig. 5.10 Fitting a straight line to the data $x_i = \{0, 1, 2, 3\}$, $y_i = \{1.4, 1.5, 3.7, 4.1\}$ with the errors $\sigma_i = \{0.5, 0.2, 1.0, 0.5\}$ by using the least-squares method (example adapted from [19]). [*Left*] The function $f(x) = a_1 x + a_0$ minimizing χ^2 ($a_0 = 0.635849$, $a_1 = 1.06618$). [*Center*] The covariance ellipse centered at (a_0, a_1) with the errors $\pm\sigma(a_0) = \pm 0.307$ and $\pm\sigma(a_1) = \pm 0.222$. The *straight lines* corresponding to the points on the ellipse are equally probable. The *symbols* ∘ denote a few points corresponding to the bundle of *straight lines* in the [*Right*] panel: the *straight line* corresponding to the "true" parameters a_0 and a_1 lies within this bundle with the probability $1 - \mathrm{e}^{-1/2}$

are their variance and covariance. We see that the straight line runs through the "center-of-mass" of the data $(\overline{x}, \overline{y})$.

From (5.38) we obtain the variances of a_0 and a_1 (the diagonal elements of Σ^{-1}) that do not depend on the position of the points along the y-axis:

$$\sigma^2(a_0) = \left(\Sigma^{-1}\right)_{11} = \frac{S_{xx}}{S_{xx}S - S_x^2}, \qquad \sigma^2(a_1) = \left(\Sigma^{-1}\right)_{22} = \frac{S}{S_{xx}S - S_x^2}. \quad (5.39)$$

The off-diagonal elements of the covariance matrix Σ^{-1} for the parameters a_0 and a_1 are

$$\mathrm{cov}(a_0, a_1) = \left(\Sigma^{-1}\right)_{12} = \left(\Sigma^{-1}\right)_{21} = \frac{-S_x}{S_{xx}S - S_x^2},$$

thus the coefficient of linear correlation between a_0 and a_1 is

$$\rho = \frac{\mathrm{cov}(a_0, a_1)}{\sigma(a_0)\sigma(a_1)}.$$

The parameters a_0 and a_1, their uncertainties $\sigma(a_0) \equiv \sigma_0$ and $\sigma(a_1) \equiv \sigma_1$, and the coefficient ρ define the *covariance ellipse* centered at (a_0, a_1) with the semi-axes r_0 and r_1, rotated by α in the (a_0, a_1)-plane [19]. The parameters are

$$\begin{aligned} \tan 2\alpha &= 2\rho\sigma_0\sigma_1/\left(\sigma_0^2 - \sigma_1^2\right), \\ r_0^2 &= \sigma_0^2\sigma_1^2\left(1-\rho^2\right)/\left[\sigma_1^2\cos^2\alpha - \rho\sigma_0\sigma_1\sin 2\alpha + \sigma_0^2\sin^2\alpha\right], \\ r_1^2 &= \sigma_0^2\sigma_1^2\left(1-\rho^2\right)/\left[\sigma_1^2\sin^2\alpha + \rho\sigma_0\sigma_1\sin 2\alpha + \sigma_0^2\cos^2\alpha\right]. \end{aligned} \quad (5.40)$$

An example is shown in Fig. 5.10 (center). All points on the ellipse are equally probable and represent the "1σ" confidence interval for the parameters a_0 and a_1.

This interval corresponds to an infinite set of possible straight lines. For the indicated points on the ellipse they are shown in Fig. 5.10 (right).

How do the uncertainties of the model parameters, $\sigma^2(a_0)$ and $\sigma^2(a_1)$, change when the number of points n is increased? Assume that the points $\{x_i\}_{i=0}^{n-1}$ on some interval $[\alpha, \beta]$ are distributed uniformly, so $x_i = \alpha + i\Delta x$ with $\Delta x = (\beta - \alpha)/(n - 1)$, and that the measurement error is $\sigma_i = \sigma$. We compute $S = n/\sigma^2$, $S_x = n(\alpha + \beta)/(2\sigma^2)$ and $S_{xx} = n[(\alpha^2 + \alpha\beta + \beta^2)(2n - 1) - 3\alpha\beta]/[6(n - 1)\sigma^2]$. From (5.39) we read off the asymptotic behavior in the limit $n \to \infty$,

$$\sigma^2(a_0) \sim \frac{4(\alpha^2 + \alpha\beta + \beta^2)\sigma^2}{n(\alpha - \beta)^2} + \mathcal{O}\left(\frac{1}{n^2}\right), \qquad \sigma^2(a_1) \sim \frac{12\sigma^2}{n(\alpha - \beta)^2} + \mathcal{O}\left(\frac{1}{n^2}\right).$$

With increasing n, the regression coefficients a_0 and a_1 gain in precision similarly as the statistical averages, at the rate $\sigma(a_0), \sigma(a_1) \sim \mathcal{O}(n^{-1/2})$.

Unknown Errors If the errors σ_i of individual y_i are not known, the uncertainties of a_0 and a_1 can be estimated by using the procedure fully elaborated in [25]. We first set $\sigma_i = 1$ for $\forall i$ and compute a_0 and a_1 and their variances $\sigma^2(a_0)$ and $\sigma^2(a_1)$ by using (5.39). We use these a_0 and a_1 to compute the value of χ^2 from (5.35), and multiply both variances by $\chi^2/(n - 2)$. The estimates for the uncertainties of the parameters a_0 and a_1 are therefore rescaled as

$$\sigma(a_0) \to \sigma(a_0)\sqrt{\chi^2/(n-2)}, \qquad \sigma(a_1) \to \sigma(a_1)\sqrt{\chi^2/(n-2)}.$$

5.5.4 *Linear Regression (Fitting a Straight Line) with Errors in Both Coordinates*

In straight-line regression with the function $f(x) = a_1 x + a_0$, where the errors occur in both variables (y_i as well as x_i), we try to minimize

$$\chi^2 = \sum_{i=0}^{n-1} \frac{(y_i - a_1 x_i - a_0)^2}{a_1^2\sigma_{xi}^2 + \sigma_{yi}^2}.$$

To determine a_0 and a_1 we have to fulfill $\partial\chi^2/\partial a_0 = 0$ and $\partial\chi^2/\partial a_1 = 0$, but a_1 appears non-linearly, so the problem is not trivial; for a review, see [26]. From [27] we adopt a reliable algorithm to compute the parameters a_0 and a_1 and their uncer-

tainties, $\sigma(a_0)$ and $\sigma(a_1)$. It typically converges in $\approx$10 iterations.

Input: Values $(x_i, y_i)_{i=0}^{n-1}$, errors $(\sigma_{xi}, \sigma_{yi})_{i=0}^{n-1}$, correlations $|r| \le 1$ between the errors, initial approximation for a_1, tolerance ε
for $i = 0$ **to** $n-1$ **do**
  $w_{xi} = 1/\sigma_{xi}^2$;
  $w_{yi} = 1/\sigma_{yi}^2$;
end
while (|difference between consecutive $a_1| > \varepsilon$) **do**
  for $i = 0$ **to** $n-1$ **do**
    $W_i = w_{xi} w_{yi}/[w_{xi} + a_1^2 w_{yi} - 2a_1 r\sqrt{w_{xi} w_{yi}}]$;
  end
  $\overline{x} = \sum_i W_i x_i / \sum_i W_i$;
  $\overline{y} = \sum_i W_i y_i / \sum_i W_i$;
  for $i = 0$ **to** $n-1$ **do**
    $u_i = x_i - \overline{x}$;
    $v_i = y_i - \overline{y}$;
    $\beta_i = W_i[u_i/w_{yi} + a_1 v_i/w_{xi} - (a_1 u_i + v_i) r/\sqrt{w_{xi} w_{yi}}]$;
  end
  $a_1 = \sum_i W_i \beta_i v_i / \sum_i W_i \beta_i u_i$; // new estimate for a_1
end
$a_0 = \overline{y} - a_1 \overline{x}$;
for $i = 0$ **to** $n-1$ **do**
  $\xi_i = \overline{x} + \beta_i$;
end
$\overline{\xi} = \sum_i W_i \xi_i / \sum_i W_i$;
for $i = 0$ **to** $n-1$ **do**
  $\eta_i = \xi_i - \overline{\xi}$;
end
$\sigma^2(a_1) = 1/\sum_i W_i \eta_i^2$;
$\sigma^2(a_0) = 1/\sum_i W_i + \overline{\xi}^2 \sigma^2(a_1)$;
Output: a_0, a_1, $\sigma(a_0)$, $\sigma(a_1)$

5.5.5 *Fitting a Constant*

In zeroth-order regression a constant function is fitted to the values y_i. We minimize (5.35) with $f(x_i) = a$. From the condition $\mathrm{d}\chi^2/\mathrm{d}a = 0$ we get

$$a = \frac{1}{S} \sum_{i=0}^{n-1} \frac{y_i}{\sigma_i^2}, \quad S = \sum_{i=0}^{n-1} \frac{1}{\sigma_i^2}, \ \sigma(a) = \frac{1}{\sqrt{S}}, \tag{5.41}$$

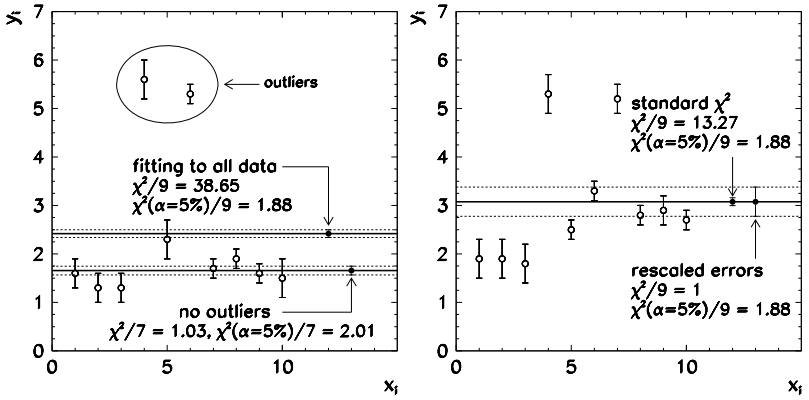


Fig. 5.11 Fitting a constant to the data $(x_i, y_i)_{i=0}^{9}$ (adapted from [19]). [*Left*] The weighted average in the presence of two outliers and without them. [*Right*] The weighted average of the data with an unexplained systematic error. The fit procedure yields the parameter a with a very small uncertainty inconsistent with the actual scatter of data. Rescaling of the errors according to (5.42) gives a more sensible result

which is precisely the weighted average (5.8). We use this method instead of the arithmetic mean whenever the measured values of the same quantity have different errors. For $n \to \infty$ the uncertainty of a scales as $\sigma(a) \sim \mathcal{O}(n^{-1/2})$.

After obtaining a, we may compute $\chi^2 = \sum_{i=0}^{n-1}(y_i - a)^2/\sigma_i^2$ to ascertain whether the premise of the normally distributed errors σ_i is justified or not. At the chosen risk level α (e.g. $\alpha = 5\ \%$) we determine χ_+^2 from the equation

$$\int_{\chi_+^2}^{\infty} \frac{\mathrm{d}P}{\mathrm{d}\chi^2}\left(\chi^2; n-1\right)\mathrm{d}\chi^2 = \alpha,$$

where $\mathrm{d}P/\mathrm{d}\chi^2$ is given in (5.26), and compare χ_+^2 to the χ^2 computed from the data. If $\chi^2 > \chi_+^2$, the premise is questionable; in the opposite case, we may fit the constant a and its uncertainty is considered as consistent with the data.

Example Numerous dangers lurk behind all this simplicity. Figure 5.11 (left) shows $n = 10$ data points (x_i, y_i) to which we fit a constant a. Following the procedure described above we get $a = 2.42 \pm 0.08$ and $\chi^2/(n-1) = 38.65$. At $\alpha = 5\ \%$ for $\nu = n - 1 = 9$ we read off $\chi_+^2(\alpha = 5\ \%)/9 = 1.88$ from Fig. 5.12. Since $\chi^2/(n-1) > \chi_+^2/(n-1)$, the assumption about the normal distribution of errors should be discarded. (In other words, with a probability much greater than α, the constant probability density is inconsistent with the data sample.) But if we omit the outliers y_3 and y_5, we get $a = 1.66 \pm 0.09$ and $\chi^2/7 = 1.03$, while $\chi_+^2(\alpha = 5\ \%)/7 = 2.01$. Now the fit is consistent with the data. Here we again run into the problems of robustness. More on this will be told in Sect. 5.5.7.

Another pitfall is illustrated by Fig. 5.11 (right). Applying the method from Sect. 5.3.1 we determine that some data points fall outside of the confidence in-

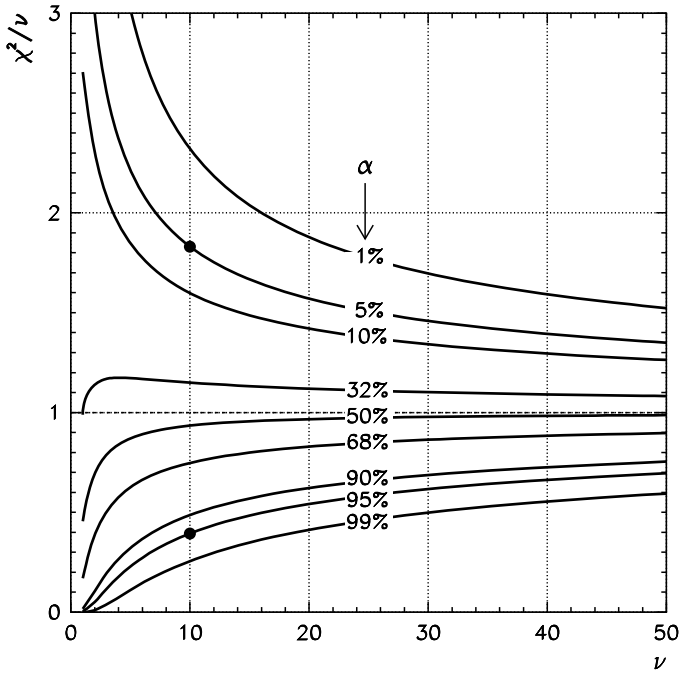


Fig. 5.12 The reduced value χ^2/ν used in the χ^2-test as a function of the number of degrees of freedom, at some typical risk levels α. The *symbols* • denote the points $(\nu, \chi^2_-/\nu) = (10, 0.394)$ and $(\nu, \chi^2_+/\nu) = (10, 1.83)$ from Example (5.28)

terval for the sample mean. By using the χ^2-test we obtain $\chi^2/9 = 13.27$ and again $\chi^2_+(\alpha = 5\ \%)/9 = 1.88$. But individual outliers should not be blamed for the large value of χ^2, as obviously the data have an underestimated systematic error. In such cases the measurement error should be *rescaled* [28]

$$\sigma_i' = \sigma_i \sqrt{\frac{\chi^2}{n-1}}, \quad i = 0, 1, \ldots, n-1. \tag{5.42}$$

We still form the weighted average for the computation of the fit parameter a by using (5.41), but its uncertainty becomes

$$\sigma'(a) = \sqrt{\frac{\chi^2}{n-1}} \left(\sum_{i=0}^{n-1} \frac{1}{\sigma_i^2} \right)^{-1/2}.$$

Thus we get a more sensible result corresponding to $\chi^2/(n-1) = 1$ (this example is shown in Fig. 5.11 (right)).

5.5.6 Generalized Linear Regression by Using SVD

In generalized linear regression

$$y(x) = \sum_{j=0}^{m-1} a_j \phi_j(x),$$

where ϕ_j are the basis functions, we attempt to use a small number of parameters a_j to fit the model function y to a large amount of data. Such problems are over-determined. Yet in many instances the data are not rich enough to allow for an unambiguous and physically sensible determination of the linear combination of the basis functions: often we can obtain several solutions of the problem that minimize

$$\chi^2 = \sum_{i=0}^{n-1} \left[\frac{y_i - \sum_{j=0}^{m-1} a_j \phi_j(x_i)}{\sigma_i} \right]^2$$

almost equally well. (From the strict mathematical point of view, the solution of the over-determined system by the least-squares methods is, of course, unique.)

We can obtain a much better handle over the meaningfulness of the parameters $\boldsymbol{a} = (a_0, a_1, \ldots, a_{m-1})^{\mathrm{T}}$ by the use of singular-value decomposition (SVD, see Sects. 3.3.2 and 3.3.3). Solving the least-squares problem by the SVD makes sense in the case of over-determined systems where we are trying to describe the measured data with a set of model parameters that is unnecessarily large. The singular values obtained by the SVD help us eliminate the superfluous combinations of the basis functions. We denote

$$A_{ij} = \frac{\phi_j(x_i)}{\sigma_i}, \qquad b_i = \frac{y_i}{\sigma_i},$$

and perform the "thin" SVD of the matrix $A = U \Lambda V^{\mathrm{T}} \in \mathbb{R}^{n \times m}$ (p. 123) where $n \geq m$. The matrix $U = (\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{m-1})$ has columns $\boldsymbol{u}_i$ of dimension n, the matrix $V = (\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m-1})$ has columns $\boldsymbol{v}_i$ of dimension m, and the diagonal matrix $\Lambda = \mathrm{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{m-1})$ contains the singular values λ_i. We obtain the vector of parameters $\boldsymbol{a}$ by summing

$$\boldsymbol{a} = \sum_{i=0}^{m-1} \frac{\boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{b}}{\lambda_i} \boldsymbol{v}_i.$$

The covariance matrix of the parameters $\boldsymbol{a}$ is $\mathrm{cov}(a_j, a_k) = \sum_{i=0}^{m-1} V_{ji} V_{ki} / \lambda_i^2$ and its diagonal elements represent the individual variances

$$\sigma^2(a_j) = \sum_{i=0}^{m-1} \frac{V_{ji}^2}{\lambda_i^2}. \tag{5.43}$$

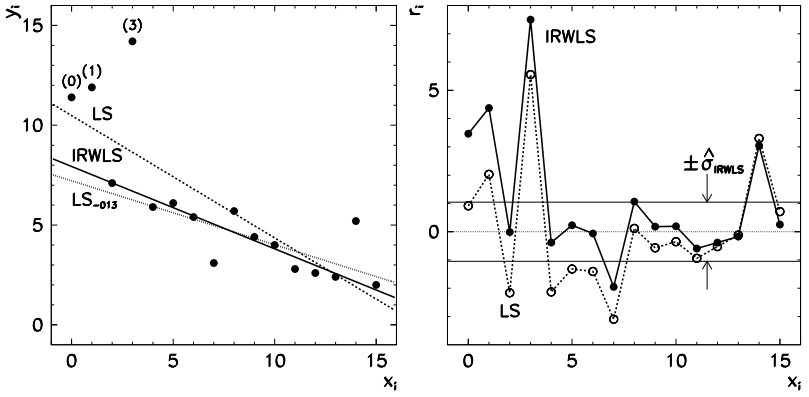


Fig. 5.13 Robust linear regression. (Example adapted from [17].) [*Left*] The regression straight line by the method of least squares (LS), by omitting three outliers (LS_{-013}), and by the iterative reweighting method (IRWLS). [*Right*] The residuals r_i and the estimate of dispersion by the IRWLS method

We should pay attention to all singular values λ_i for which the ratio $\lambda_i/\lambda_{\max}$ is smaller than $\approx n\varepsilon_{\mathrm{M}}$. Such values increase the error (5.43) and indicate that the inclusion of new model parameters is meaningless. Moreover, they do not contribute significantly to the minimization of χ^2, so we exclude them. We do this by setting $1/\lambda_i = 0$ (for a detailed explanation, see the comment to the `Fitsvd` algorithm in [25]). We may exclude also those singular values for which the ratio $\lambda_i/\lambda_{\max}$ is *larger* than $\approx n\varepsilon_{\mathrm{M}}$, until χ^2 starts to increase visibly.

5.5.7 Robust Methods for One-Dimensional Regression

Like all estimates of location and dispersion, regression methods are sensitive to outliers (Sect. 5.2.1). The straight line LS in Fig. 5.13 (left) corresponds to the standard regression on the data $\{(x_i, y_i)\}_{i=0}^{15}$ with unknown errors: here we minimize the sum of the squares of the residuals $r_i = y_i - (a_1 x_i + a_0)$, so the straight line LS fails to describe the bulk of the data because of the outliers at x_0, x_1, and x_3. If we remove these three outliers, we obtain the straight line denoted by LS_{-013}. Robust regression methods [29] yield a good description of the majority of the data without the need to remove individual outliers by hand.

The literature on robust regression techniques is abundant: for the identification of outliers (*regression diagnostics*) see [30]; for a review of methods see [31]. Numerous methods exist, all having some advantages and deficiencies; many of them are awkward to implement or entail high numerical costs. One of such methods is the regression in which not the sum $\sum_i r_i^2$ that is minimized, but the sum $\sum_i |r_i|$ (an L_1-type estimator; see e.g. [32] and [33]). Here we describe two procedures with proven good properties for everyday use.

IRWLS The *iterative reweighted least-squares method* (IRWLS) closely follows the logic of M-estimates of location (Sect. 5.2.2 and algorithm on p. 215). The iteration to the final values of the parameters a_0 and a_1 of the regression straight line typically converges in $\approx$30 steps. The determination of their uncertainties, the estimate for the dispersion, and other details are discussed in [17], p. 105. An example is shown in Fig. 5.13 (left, IRWLS line). Figure 5.13 (right) shows the residuals r_i in the LS and IRWLS methods.

Input: Values $(x_i, y_i)_{i=0}^{n-1}$, initial approximations for a_0 and a_1 from standard linear regression, relative precision ε
for $i = 0$ **to** $n-1$ **do**
  $r_i^{(0)} = y_i - (a_0 + a_1 x_i)$;
end
$\widehat{\sigma} = 1.4826 \cdot \text{median}(|r_i^{(0)}|, r_i^{(0)} \neq 0)$;
$k = 0$;
while ($\max_i |r_i^{(k+1)} - r_i^{(k)}| > \varepsilon\widehat{\sigma}$) **do**
  for $i = 0$ **to** $n-1$ **do**
    $w_i = W(r_i^{(k)}/\widehat{\sigma})$; // $W(t)$ given by (5.21)
  end
  Compute new estimates for a_0 and a_1 by solving the linear system

$$a_0 \sum_i w_i + a_1 \sum_i w_i x_i = \sum_i w_i y_i$$
$$a_0 \sum_i w_i x_i + a_1 \sum_i w_i x_i^2 = \sum_i w_i x_i y_i$$

  for $i = 0$ **to** $n-1$ **do**
    $r_i^{(k+1)} = y_i - (a_0 + a_1 x_i)$;
  end
  $k = k + 1$;
end
Output: a_0, a_1 by the IRWLS method

LMS The second method resembles standard linear regression, except that the median of the squares of the residuals $r_i = y_i - (a_1 x_i + a_0)$ is minimized, hence its name, *least median of squares* (LMS). We seek a_0 and a_1 such that

$$\text{median}(y_i - a_1 x_i - a_0)^2 = \text{min}. \tag{5.44}$$

The LMS method [34] is very robust and behaves well even in the rare circumstances in which IRWLS fails. An example is shown in Fig. 5.14 (left). We have a sample of $n_1 = 30$ data $y_i = ax_i + b + u_i$, where $a = 1$, $b = 2$, $x_i \sim U(1, 4)$, and $u_i \sim N(0, 0.2)$, and a set of $n_2 = 20$ points assumed to be outliers, $x_i \sim N(7, 0.5)$, $y_i \sim N(2, 0.5)$. We would like to fit a straight line to the data such that the result will be oblivious to the outlier portion of this compound set. Neither the LS nor the

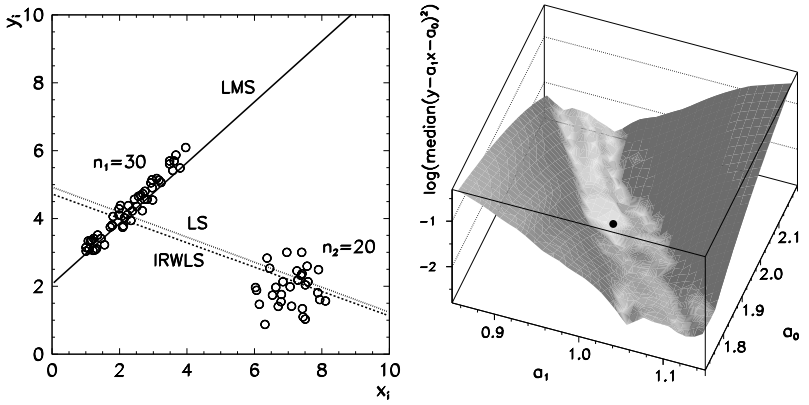


Fig. 5.14 Robust linear regression on the data with many outliers. [*Left*] Only the LMS method correctly describes the majority of the data. [*Right*] The function being minimized in the LMS method. The *symbol* • denotes the global minimum

IRWLS method yield meaningful results: both simply run the line through both data portions. In contrast, the LMS line fits the non-outlier group well.

The main nuisance of the LMS method is precisely the numerical minimization (5.44). The function we wish to minimize with respect to the parameters a_j has $\mathcal{O}(n^{m+1})$ local minima, where n is the number of data points (x_i, y_i) and m is the degree of the regression polynomial. In the example from the figure we have $n = n_1 + n_2 = 50$ and $m + 1 = 2$ (straight line), so there are $\approx$2500 local minima, among which the global minimum needs to be located, as shown in Fig. 5.14 (right). This is best accomplished—there are dangers since the function is continuous, but not continuously differentiable—by forming the function

$$M(a_1) = \min_{a_0} \left\{ \text{median}(y_i - a_1 x_i - a_0)^2 \right\},$$

where for each a_1 we determine a_0, and then minimize $M(a_1)$ with respect to a_1. This implementation becomes very costly at large n. Fast computations of the regression parameters by LMS methods are non-trivial: see [35, 36].

5.5.8 Non-linear Regression

In non-linear regression the fitting functions depend non-linearly on the parameters a_i. The minimization of (5.35) by (5.36) therefore requires a multi-dimensional minimum search. The well-tested tool (implemented in standard libraries) is the Levenberg–Marquardt method. Non-linear regression with strongly non-linear functions involving numerous regression parameters should be the last exit: if possible, convert the problem to linear regression. See [37–40].

5.6 Multiple Linear Regression

The remaining sections of this chapter, heavily indebted to the wonderfully clear exposition and carefully chosen examples of [41], introduce some of the most appealing methods of *multivariate analysis*, which is the simultaneous statistical analysis of a collection of variables. For example, we would like to understand the influence of several independent variables to a single dependent variable, or to study the mutual dependence of several variables. Such methods can be applied in innumerable areas of natural and social sciences.

5.6.1 The Basic Method

Section 5.5 was devoted to simple one-dimensional regression where each independent (input) quantity x_i corresponded to a single dependent (output) quantity y_i. An obvious generalization is multiple linear regression, by which we identify the influence of many input variables on a single output variable.

Assume that we have n measurements or other sources of a set of m independent variables x. We would like to ascertain whether a linear relation can be established between the values of x and the n dependent variables y. In other words, can the whole data set be described by the model

$$y_i = f(\boldsymbol{x}_i) + \Delta y_i = a_0 + \sum_{j=1}^{m} a_j X_{ij} + \Delta y_i, \quad i = 0, 1, \ldots, n-1 \tag{5.45}$$

(see Problem 5.12.1)? Here X_{ij} stands for the ith measurement of the jth independent variable, Δy_i are the unknown errors, and a_j are the regression parameters to be determined. We collect a single (ith) measurement of m independent variables into the vector $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})^{\mathrm{T}}$, so that all n measurements can be assembled in the matrix $X = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1})^{\mathrm{T}}$. (Note the index ranges $i = 0, 1, 2, \ldots$ and $j = 1, 2, 3, \ldots$) We compute the mean vector (of dimension m)

$$\overline{\boldsymbol{x}} = \frac{1}{n} \sum\nolimits_{i=0}^{n-1} \boldsymbol{x}_i, \tag{5.46}$$

and form the matrix of data from which their average has been subtracted,

$$\overline{X} = (\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}, \ldots, \overline{\boldsymbol{x}})^{\mathrm{T}}, \qquad X_{\mathrm{c}} = X - \overline{X}. \tag{5.47}$$

The matrices X, $\overline{X}$, and X_{c} have size $n \times m$. The dependent variables are handled similarly. We compute the mean of n measured values

$$\overline{y} = \frac{1}{n} \sum\nolimits_{i=0}^{n-1} y_i,$$

and use it to form the vector of dependent variables from which the mean has been subtracted,

$$\overline{\boldsymbol{y}} = (\overline{y}, \overline{y}, \ldots, \overline{y})^{\mathrm{T}}, \qquad \boldsymbol{y}_{\mathrm{c}} = \boldsymbol{y} - \overline{\boldsymbol{y}}.$$

The vectors $\boldsymbol{y}$, $\overline{\boldsymbol{y}}$, and $\boldsymbol{y}_{\mathrm{c}}$ have dimension n. Again, our basic requirement is to minimize the squares of the deviations (residuals) $y_i - f(\boldsymbol{x}_i)$ with respect to the parameters a_j. The measure of deviation is given in (5.35) which we generalize to more than one dimension and set $\sigma_i = 1$. For a more compact notation, we add a column of values 1 to the data matrix, yielding a $n \times (m+1)$ matrix, while the regression coefficients (including the zeroth one) are arranged in a vector of dimension $(m+1)$,

$$\Xi = (\mathbf{1}_n, X), \qquad \widehat{\boldsymbol{a}} = \left(\widehat{a}_0, \widehat{\boldsymbol{a}}_{(m)}^{\mathrm{T}}\right)^{\mathrm{T}}.$$

We obtain the vector of regression parameters $\widehat{\boldsymbol{a}}$ by solving the normal system (3.10),

$$\widehat{\boldsymbol{a}} = \left(\Xi^{\mathrm{T}}\Xi\right)^{-1}\Xi^{\mathrm{T}}\boldsymbol{y}.$$

The first component of the vector $\widehat{\boldsymbol{a}}$ (the zeroth regression parameter or the *intercept*) and the m *slopes* of the regression "straight lines" are then given by

$$\widehat{a}_0 = \overline{y} - \overline{\boldsymbol{x}}^{\mathrm{T}}\widehat{\boldsymbol{a}}_{(m)}, \quad \widehat{\boldsymbol{a}}_{(m)} = (a_1, a_2, \ldots, a_m)^{\mathrm{T}} = \left(X_{\mathrm{c}}^{\mathrm{T}}X_{\mathrm{c}}\right)^{-1}X_{\mathrm{c}}^{\mathrm{T}}\boldsymbol{y}_{\mathrm{c}} \tag{5.48}$$

(see Sect. 5.2 in [41]). Of course we would also like to know the uncertainties of the estimated parameters and the quality of the model description (5.45). The measure for the dispersion of the dependent variables is the *total sum of squares*

$$S_y = \boldsymbol{y}_{\mathrm{c}}^{\mathrm{T}}\boldsymbol{y}_{\mathrm{c}}.$$

The total dispersion has two contributions. The first contribution can be explained by linear regression in the independent variables x, and is equal to

$$S_{\mathrm{reg}} = \widehat{\boldsymbol{a}}^{\mathrm{T}}\left(\Xi^{\mathrm{T}}\Xi\right)\widehat{\boldsymbol{a}}.$$

The remaining portion of the dispersion that is not encompassed by the presumed linear model, is called the *residual sum of squares* and amounts to

$$S_{\mathrm{res}} = S_y - S_{\mathrm{reg}} = (\boldsymbol{y} - \Xi\widehat{\boldsymbol{a}})^{\mathrm{T}}(\boldsymbol{y} - \Xi\widehat{\boldsymbol{a}}).$$

Finally, we compute the *residual variance* $\widehat{\sigma}^2$ which also determines the variance of the individual regression parameters:

$$\widehat{\sigma}^2 = \frac{S_{\mathrm{res}}}{n-m-1}, \qquad \mathrm{var}(\widehat{\boldsymbol{a}}) = \widehat{\sigma}^2\left(\Xi^{\mathrm{T}}\Xi\right)^{-1}. \tag{5.49}$$

The fraction of the total dispersion in $\boldsymbol{y}$ that can be explained by linear regression in m independent variables x, is given by the correlation coefficient

$$R^2 = \frac{S_{\mathrm{reg}}}{S_y} = \frac{S_y - S_{\mathrm{res}}}{S_y} = 1 - \frac{S_{\mathrm{res}}}{S_y}.$$

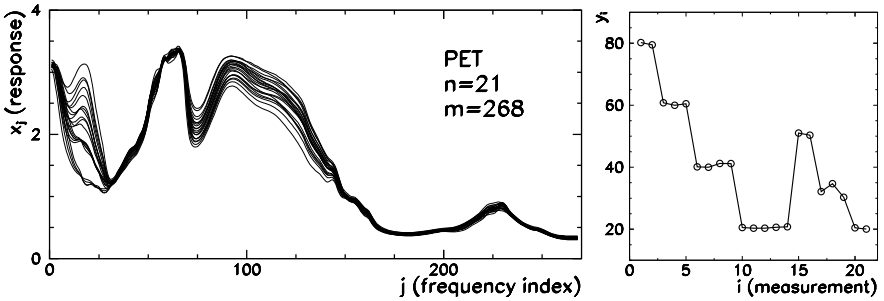


Fig. 5.15 Raman absorption spectrum of PET yarns as a typical experiment in which the number of measurements n is much smaller than the number of independent variables m in the individual measurement. (Example from [41] based on original data from [42].) [*Left*] The $n = 21$ measurements at $m = 268$ frequencies. [*Right*] Values of dependent variables (yarn density) in n measurements

The value R^2 lies between 0 and 1. Values close to 1 roughly indicate that the linear model (5.45) is an adequate description of data.

If we wish to judge the relevance of the individual parameters a_j, we must assume that some distribution law governs the errors Δy_i in the model (5.45). In most cases $n \gg 1$ and the errors are normally distributed, $\Delta y_i \sim N(0, \sigma^2)$. For each estimated regression parameter $\widehat{a}_j$ we form the statistic

$$t_j = \frac{\widehat{a}_j}{\widehat{\sigma}\sqrt{\xi_j}}, \quad \xi_j = \left[\left(\Xi^{\mathrm{T}}\Xi\right)^{-1}\right]_{jj},$$

where $\widehat{\sigma}$ is given by (5.49). A value $|t_j| > 2$ indicates that the corresponding parameter is relevant ($a_j \neq 0$). A value $|t_j| \leq 2$ (and even more so $|t_j| \approx 0$) hints at $a_j \approx 0$.

5.6.2 Principal-Component Multiple Regression

The computation of the coefficients $\boldsymbol{a}$ and their variances in multiple regression involves the matrix $X_{\mathrm{c}}^{\mathrm{T}} X_{\mathrm{c}}$, where X_{c} is the centered data matrix (5.47). If X_{c} is ill-conditioned or contains weakly linearly dependent columns, or when the number of measurements in smaller than the number of independent variables ($n < m$ or even $n \ll m$), $X_{\mathrm{c}}^{\mathrm{T}} X_{\mathrm{c}}$ becomes singular or nearly singular (see Fig. 5.15 and the following text). In such cases the regression parameters $\boldsymbol{a}$ cannot be uniquely determined or may have unphysical values and dispersions.

If the matrix $X_{\mathrm{c}}^{\mathrm{T}} X_{\mathrm{c}}$ is singular or nearly singular, we can use its pseudo-inverse (see Sect. 3.3.3). Assume that $X_{\mathrm{c}}^{\mathrm{T}} X_{\mathrm{c}}$ has known rank r ($1 \leq r \leq m$). It can be diagonalized as

$$X_{\mathrm{c}}^{\mathrm{T}} X_{\mathrm{c}} = V \Lambda V^{\mathrm{T}}, \quad \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r), \; V^{\mathrm{T}} V = I_r,$$

while the smallest $m - r$ eigenvalues are zero. The columns of the $m \times r$ matrix V are eigenvectors $\boldsymbol{v}_j$ corresponding to the individual eigenvalues λ_j. The pseudo-inverse of $X_c^T X_c$ is then

$$\left(X_c^T X_c\right)^+ = V \Lambda^{-1} V^T = \sum_{j=1}^{r} \frac{\boldsymbol{v}_j \boldsymbol{v}_j^T}{\lambda_j}.$$

Instead of the true inverse $(X_c^T X_c)^{-1}$ (which does not exist in this case) we use the pseudo-inverse in (5.48). Thus we obtain a unique vector of m regression coefficients by the method of *generalized* or *pseudo-inverse regression* (denoted by "pi" in the following):

$$\widehat{\boldsymbol{a}}_{(r)}^{\mathrm{pi}} = \left(X_c^T X_c\right)^+ X_c^T \boldsymbol{y}_c = \sum_{j=1}^{r} \frac{(\boldsymbol{v}_j \boldsymbol{v}_j^T) X_c^T \boldsymbol{y}_c}{\lambda_j}. \tag{5.50}$$

The values of dependent variables $\boldsymbol{y}$ (vector of dimension n) corresponding to the regression parameters (5.50) are

$$\widehat{\boldsymbol{y}}^{\mathrm{pi}} = \overline{\boldsymbol{y}} + X_c \widehat{\boldsymbol{a}}_{(r)}^{\mathrm{pi}}.$$

The estimate for the regression coefficients can also be obtained by decomposing the matrix X_c or the product $X_c^T X_c$ to principal components (see Sect. 5.7). The basic idea is to form the matrices $Z_r = X_c V$ that contain the first r principal components of X_c and correspond to those directions in r-dimensional space along which the data X_c have the largest variance, while the remaining $m - r$ components are neglected. As input variables in the regression we use the matrix Z_r, while the output variable is $\boldsymbol{y}_c$. The regression coefficients for the principal components Z_r (not for the variables X_c) are [41]

$$\boldsymbol{\zeta}_{(r)} = (\zeta_1, \zeta_2, \ldots, \zeta_r)^T = \left(Z_r^T Z_r\right)^{-1} Z_r^T \boldsymbol{y}_c = \Lambda^{-1} V^T X_c^T \boldsymbol{y}_c.$$

We compute the regression coefficients for the original independent variables (the data in the matrix X_c) by first forming the m-dimensional vectors

$$\widetilde{\boldsymbol{a}}_j = \zeta_j \boldsymbol{v}_j, \quad j = 1, 2, \ldots, r,$$

as $V \in \mathbb{R}^{m \times r}$. Then we compute the partial sums (here "pc" denotes principal components)

$$\boldsymbol{a}_{(\rho)}^{\mathrm{pc}} = \sum_{j=1}^{\rho} \widetilde{\boldsymbol{a}}_j = \sum_{j=1}^{\rho} \zeta_j \boldsymbol{v}_j, \quad \rho = 1, 2, \ldots, r. \tag{5.51}$$

The vector of regression coefficients $\boldsymbol{a}_{(\rho)}^{\mathrm{pc}}$ has m components (equal to the number of independent variables in a single measurement). The ρth partial sum gathers ρ

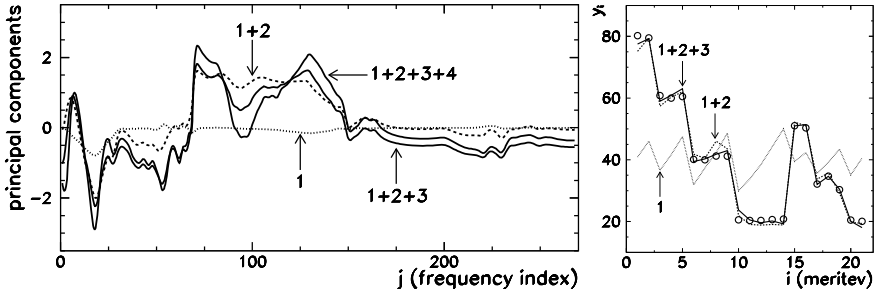


Fig. 5.16 Principal-component regression. [*Left*] Vectors of coefficients (5.51). The first four partial sums are shown. [*Right*] Values of dependent variables (*empty circles*). By including just three major principal components (three *curves* for 1, 2, and 3 components) the data can be described sufficiently well. See also caption to Fig. 5.15

principal components in it. The values of the dependent variables $\boldsymbol{y}$ corresponding to the regression parameters (5.50) are the same as in the pseudo-inverse method,

$$\widehat{\boldsymbol{y}}^{\mathrm{pc}} = \overline{\boldsymbol{y}} + X_{\mathrm{c}} V \widehat{\boldsymbol{\zeta}}_{(r)} = \overline{\boldsymbol{y}} + X_{\mathrm{c}} \boldsymbol{a}^{\mathrm{pc}}_{(r)} = \widehat{\boldsymbol{y}}^{\mathrm{pi}}.$$

The procedure outlined here is known as *principal-component regression* (PCR). Above all, it is applicable in the cases where the number of variables m is much larger than the number of measurements n. In the PCR method, it is precisely the restriction to a limited number of principal components that helps us avoid the problems due to the near-singularity of the data matrix.

Example Principal-component regression is a good tool to seek the connections between the infrared Raman spectrum of the polyethylene-terephthalate (PET) yarns and their density. Figure 5.15 (left) shows the $n = 21$ measurements of the spectrum at $m = 268$ frequencies, which are stored in the data matrix X (or X_{c} after centering). In each of the n measurements we also measure the density of the yarn, and store the results in the vector of dependent variables $\boldsymbol{y}$ (or $\boldsymbol{y}_{\mathrm{c}}$ after centering). The dependent variables are shown in Fig. 5.15 (right).

The final result of the regression is the coefficient vector (5.51) which we gradually augment by additional principal components. Figure 5.16 (left) shows the partial sums up to the fourth. Typically we include only a few principal components; further admixtures merely tend to increase the noise. Figure 5.16 (right) clearly shows how well the measured dependent variables can be described by taking just one, two, or three principal components.

5.7 Principal-Component Analysis

Principal-component analysis (PCA) is one of the most important tools to reduce the dimensionality of correlated multivariate data. The basic idea can be illus-

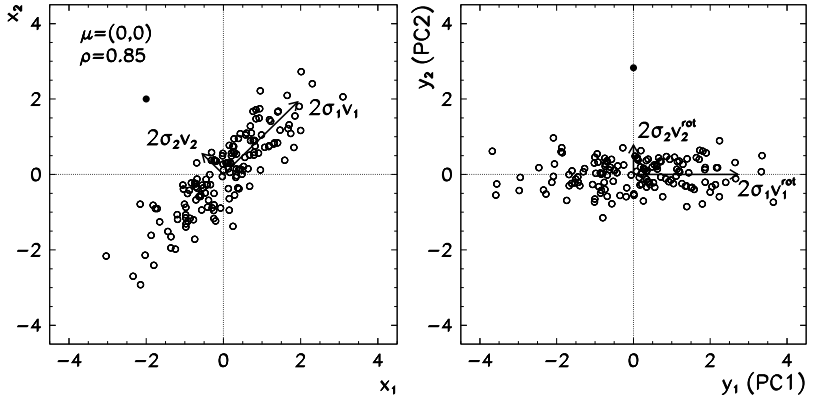


Fig. 5.17 [*Left*] Bivariate data with the rescaled eigenvectors of the covariance matrix $2\sigma_1 \boldsymbol{v}_1$ and $2\sigma_2 \boldsymbol{v}_2$ (the lengths of the vectors are twice the standard deviations along their directions). [*Right*] Transformed data. The *symbol* • represents the data point slightly detached from the central group, which cannot be identified from the analysis of the distribution of the variables x_1 or x_2 alone, yet it is easily caught as an outlier in the y_2 variable

trated by the example of a data set $\boldsymbol{x}_i = (x_{i1}, x_{i2})$ distributed according to the two-dimensional normal distribution $N(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)$ (see (5.60)) with the average $\boldsymbol{\mu} = (0, 0)$ and covariance matrix $\Sigma = ((1, \rho), (\rho, 1))$, where $\rho = 0.85$ is the correlation coefficient. The data points are shown in Fig. 5.17 (left). The eigenvectors of the covariance matrix Σ are $\boldsymbol{v}_1 = (1, 1)/\sqrt{2}$ in $\boldsymbol{v}_2 = (-1, 1)/\sqrt{2}$, and correspond to the eigenvalues $\lambda_1 = 1 + \rho = 1.85$ and $\lambda_2 = 1 - \rho = 0.15$.

The components x_{i1} and x_{i2} are strongly correlated, so it is obvious that in order to describe the data in Fig. 5.17 (left), a single dimension is almost sufficient: that along the eigenvector $\boldsymbol{v}_1$. It carries most of the dispersion while the variance along the vector $\boldsymbol{v}_2$ is relatively small. The basic idea of PCA is to determine the orthogonal linear transformation which projects the data $\boldsymbol{x}$ to a space with smaller dimension. For the example discussed above, we can use the transformation

$$\boldsymbol{y}_i = V^{\mathrm{T}} \boldsymbol{x}_i, \quad V = (\boldsymbol{v}_1, \boldsymbol{v}_2) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix},$$

to map all data into the vicinity of the y_1 axis, while just a minor part of their dispersion remains along the y_2 axis. Namely, the variances of the transformed variables y_1 and y_2 are

$$\begin{aligned} \mathrm{var}(y_{1,2}) &= \mathrm{var}\left(\frac{x_2 \pm x_1}{\sqrt{2}} \right) = \frac{1}{2} \mathrm{var}(x_2 \pm x_1) \\ &= \frac{1}{2} \big(\mathrm{var}(x_1) + \mathrm{var}(x_2) \pm 2\mathrm{cov}(x_1, x_2) \big) = \frac{1}{2} (1 + 1 \pm 2\rho) = \lambda_{1,2}. \end{aligned}$$

The eigenvalues of the covariance matrix, λ_1 and λ_2, are therefore equal to the variances of the transformed variables y_1 and y_2.

These two-dimensional thoughts can be generalized to the reduction of dimensionality of a set of n *correlated* multivariate data $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})^{\mathrm{T}}$, where $i = 0, 1, \ldots, n-1$. The leading example in this section are the data obtained in measuring radar reflections in the ionosphere (Problem 5.12.3, Fig. 5.28), where $n = 351$ measurements of $m = 33$ variables have been acquired. We wish to find such a linear combination of the data components $\boldsymbol{x}_i$,

$$y_{ji} = v_{j1}x_{i1} + v_{j2}x_{i2} + \cdots + v_{jm}x_{im} = \boldsymbol{v}_j^{\mathrm{T}}\boldsymbol{x}_i, \quad j = 1, 2, \ldots, r,\ r \le m,$$

that this transformation preserves the information about the dispersion of the original data as truthfully as possible, and that the variables y_{ji} become *uncorrelated*. We call the product $\boldsymbol{v}_j^{\mathrm{T}}\boldsymbol{x}$ the jth *principal component*, and the value of this product y_{ji} for a specific data component $\boldsymbol{x}_i$ the *principal-component score*. In summary, we are seeking a coefficient vector $\boldsymbol{v}_1$ that maximizes the variance

$$\frac{1}{n-1}\sum_{i=0}^{n-1}(y_{1i} - \overline{y}_1)^2,$$

then the vector $\boldsymbol{v}_2$ that maximizes $\sum_i (y_{2i} - \overline{y}_2)^2$ and simultaneously ensures that y_{1i} is uncorrelated to y_{2i}, then the vector $\boldsymbol{v}_3$ that maximizes $\sum_i (y_{3i} - \overline{y}_3)^2$ and ensures that y_{3i} is uncorrelated to both y_{2i} and y_{1i}, and so on. Due to this chain construction with intermediate requirements on the uncorrelatedness of the scores y_{ji}, PCA is also known as *decorrelation of variables*. This goal can be achieved by following either of the two paths: by diagonalizing the covariance matrix, or by the singular-value decomposition of the data matrix [43].

5.7.1 Principal Components by Diagonalizing the Covariance Matrix

To obtain the principal components from the covariance matrix, we first subtract the average (5.46) from each $\boldsymbol{x}_i$, and construct the $n \times m$ centered data matrix X_{c} according to (5.47). The estimate for the covariance matrix is

$$\widehat{\Sigma}_{xx} = \frac{1}{n-1}\sum_{i=0}^{n-1}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^{\mathrm{T}} = \frac{1}{n-1}X_{\mathrm{c}}^{\mathrm{T}}X_{\mathrm{c}}. \tag{5.52}$$

Then we compute the eigenvalues and eigenvectors of $\widehat{\Sigma}_{xx}$. This matrix is symmetric and its singular-value decomposition has the form

$$\widehat{\Sigma}_{xx} = V\Lambda V^{\mathrm{T}}, \quad \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m),\ V^{\mathrm{T}}V = I_m, \tag{5.53}$$

where λ_j are the eigenvalues of $\widehat{\Sigma}_{xx}$ ordered in decreasing magnitude, and the columns of V are its eigenvectors. The first r principal components of $\boldsymbol{x}$ are

$$y_j = \boldsymbol{v}_j^{\mathrm{T}}\boldsymbol{x}, \quad j = 1, 2, \ldots, r,\ r \le m.$$

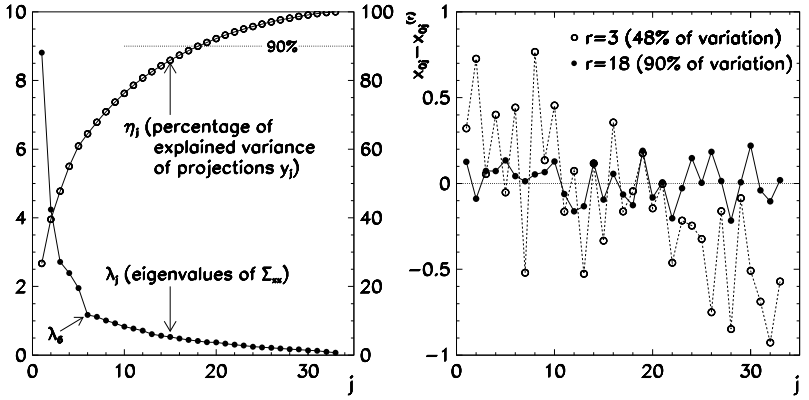


Fig. 5.18 Data from Problem 5.12.3 studied by PCA. [*Left*] Eigenvalues of the covariance matrix (*left y-axis*) and the percentage of the data variance (5.54) (*right y-axis*). The first eigenvalue λ_1 covers ≈27 % of the variance, while the sum up to including λ_{18} covers ≈90 %. [*Right*] Reconstruction of the data $\boldsymbol{x}_0$ ($i = 0$) by the sum of principal components up to ranks 3 and 18

The eigenvalues λ_j are the estimates for the variance of the jth principal component. What we wish to achieve by PCA is to use just a couple of the lowest principal components to account for nearly all variation of the original data.

The characteristic drop-off of the eigenvalues is shown in Fig. 5.18 (left, left y-axis). We define the percentage of the explained variance as

$$\eta_r = \frac{\sum_{j=1}^{r} \lambda_j}{\sum_{j=1}^{m} \lambda_j} = 1 - \frac{\sum_{j=r+1}^{m} \lambda_j}{\sum_{j=1}^{m} \lambda_j}. \tag{5.54}$$

With increasing rank r of the principal-component approximation, η_r also increases (Fig. 5.18 (left, right y-axis)). For a crude description of the data at least those principal components should be retained that correspond to the eigenvalues from the steepest portion of the λ_j curve. By this guideline, we may accept eigenvalues from λ_1 to λ_6. Typically, we take enough components that ≈90 % of the data variance is accounted for (up to including λ_{18} in the figure).

The rank-r approximation for each original data point $\boldsymbol{x}_i$ is

$$\boldsymbol{x}_i^{(r)} = \overline{\boldsymbol{x}} + \left(\sum_{j=1}^{r} \boldsymbol{v}_j \boldsymbol{v}_j^{\mathrm{T}} \right) (\boldsymbol{x}_i - \overline{\boldsymbol{x}}), \quad i = 0, 1, \ldots, n-1, \; r \le m.$$

With increasing rank r, $\boldsymbol{x}_i^{(r)}$ becomes an increasingly good approximation of $\boldsymbol{x}_i$; when $r = m$, we should, of course, retrieve $\boldsymbol{x}_i^{(m)} \equiv \boldsymbol{x}_i$. Figure 5.18 (right) shows the difference between the rank-r approximation and the actual data $\boldsymbol{x}_0$ for $r = 3$ and $r = 18$. Indeed, the approximation of a higher rank describes the data better (a smaller difference and a larger percentage of the described variance).

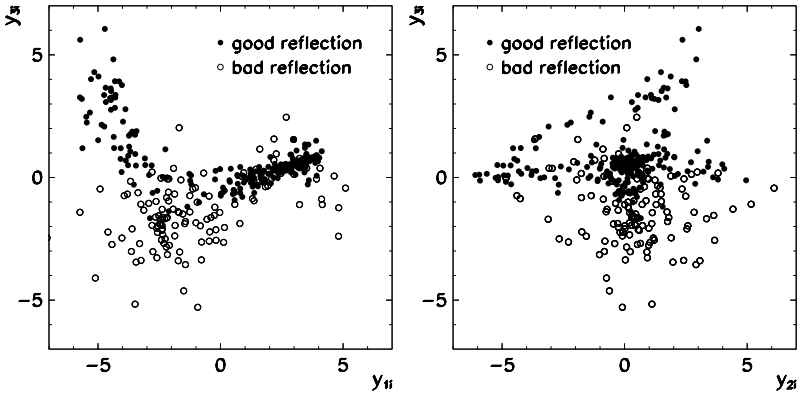


Fig. 5.19 The PCA scores for the data of Problem 5.12.3. [*Left*] Component 1 vs. component 3. [*Right*] Component 2 vs. component 3

PCA is not just a data reduction method. By evaluating the scores of the jth principal component for an individual (ith) data point,

$$y_{ji} = \boldsymbol{v}_j^{\mathrm{T}} \boldsymbol{x}_i, \quad i = 0, 1, \ldots, n-1, \tag{5.55}$$

we use the two-dimensional distribution of y_{ji} vs. $y_{j'i}$ ($j' \neq j$) to reveal peculiarities in the structure of the data, e.g. locate outliers (see Fig. 5.17) or identify clustering. Figure 5.19 shows the scores y_{3i} vs. y_{1i} and y_{3i} vs. y_{2i} for the data of Problem 5.12.3. In this case, two-dimensional images of the first few principal components help us clearly separate the good reflections from the bad ones.

5.7.2 Standardization of Data for PCA

When the estimated variances of the individual data elements $\boldsymbol{x}_i$ are comparable, PCA should be initialized with the data averaged by (5.46) and (5.47). But if the variances differ widely, or if the data elements are incomparable, the data should be standardized. This is a crucial step if quantities are given in different measurement units. Otherwise, the first few principal components will be totally dominated by the influence of the data components $\boldsymbol{x}_i$ with the largest numerical spread. The data should therefore always be centered and divided by the estimates of their standard deviations:

$$\boldsymbol{x}_i \longleftarrow \frac{\boldsymbol{x}_i - \overline{\boldsymbol{x}}}{\boldsymbol{s}_{xi}} \tag{5.56}$$

(the division of vectors is meant to be component-wise). The standardization of data means that the PCA procedure, *de facto*, is carried out with the correlation (and not the covariance) matrix. Further arguments in favor of the standardization or against it can be found in [43], Chap. 2.3.

5.7.3 Principal Components from the SVD of the Data Matrix

The second option to compute the principal components of a set of multivariate data leads through the singular-value decomposition (SVD) of the data matrix X_c. The SVD process has been described in Sect. 3.3.2. We use SVD to decompose the $n \times m$ matrix X_c as

$$X_c = U \Lambda V^T, \quad U \in \mathbb{R}^{n \times n}, \ \Lambda \in \mathbb{R}^{n \times m}, \ V \in \mathbb{R}^{m \times m}, \tag{5.57}$$

where the columns of V are the eigenvectors $\boldsymbol{v}_j$ of Σ_{xx}. In full SVD the matrix Λ has size $n \times m$ and the square $m \times m$ submatrix at its upper edge contains the *square roots* of the eigenvalues of the covariance matrix, multiplied by $(n-1)$. In other words, the eigenvalues λ_j from the decomposition (5.53) are equal to the values $\lambda_j^2/(n-1)$ from the decomposition (5.57). In addition, SVD rewards us with bonus principal-component scores: in scaled form they are hidden in the columns of U. We compute the scores (5.55) for the jth principal component as

$$y_{ji} = U_{ij} \sqrt{\lambda_j}, \quad i = 0, 1, \ldots, n-1.$$

PCA is usually carried out with the SVD of the centered data matrix X_c, not by diagonalizing the covariance matrix Σ_{xx}. The terms PCA and SVD are therefore often considered to be synonymous.

5.7.4 Improvements of PCA: Non-linearity, Robustness

PCA is a linear method that we can use to determine whether the data from $\mathbb{R}^m$, to a good approximation, actually "reside" in a smaller space $\mathbb{R}^r$, $r < m$. Yet the method fails, for example, in the simple case when the data in $\mathbb{R}^2$ is distributed approximately uniformly along the perimeter of the unit circle. Obviously such data has only one degree of freedom—the angle $\phi \in [0, 2\pi)$—while linear PCA will allow us to span these data on two practically arbitrary orthogonal directions (any two orthogonal unit vectors can describe the variance of such data). Linear PCA is also sensitive to outliers: even though the method is supposed to help us find the directions with the largest variation, it is unacceptable that a single outlier should radically alter the direction of any of the eigenvectors. Non-linear improvements to the PCA are described in [41] in Sects. 16.2 and 16.5. The initial reading on robust PCA can be found in Chap. 10 of [43] and Sect. 6.10 in [17]; see also [44].

5.8 Cluster Analysis ⋆

Cluster analysis is one of the basic methods of *unsupervised learning*. Its primary task is to classify sets of multivariate data into clusters, groups, or classes (all synonyms in common use). The number of classes is not necessarily known in advance. This is the key distinction with respect to discriminant analysis (discussed

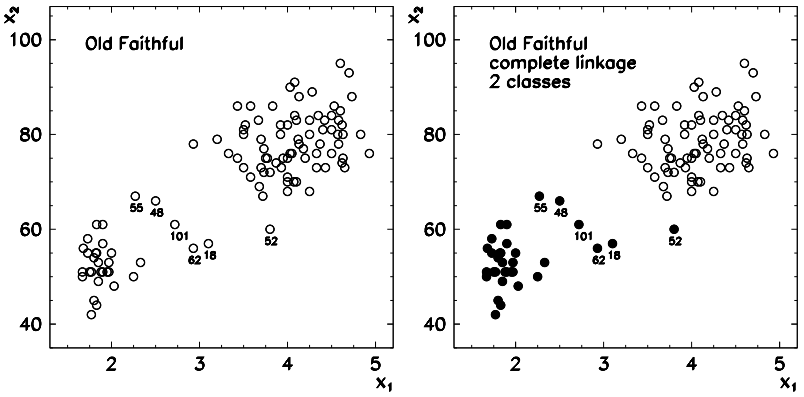


Fig. 5.20 Eruptions of the Old Faithful geyser (Yellowstone National Park). We have 107 measurements of the eruption duration (x_1) and the times between two consecutive eruptions (x_2), both in minutes. [*Left*] Unclassified data. [*Right*] Data classified in two clusters by using agglomerative clustering with complete linkage. See also Fig. 5.21

in Sect. 5.9) where we initially declare the number of classes and where previously processed data may be used to classify new data for which class membership is yet to be determined. A famous case is shown in Fig. 5.20 (left). We would like to know whether in the distribution of geyser eruptions in terms of their duration and pauses between the eruptions any natural ordering or classification can be established, reflecting an underlying physical mechanism [45].

A cluster can loosely be defined as a group of objects (points in the plane or in space) in which the objects are "close" to each other, while the objects within one group are "far away" from the objects in other groups. The way the distance between the objects and between the clusters is measured depends on the individual method. Often, the results can be in conflict with our subjective judgment: in Fig. 5.20 (left) we may recognize two or three clusters, depending on how we interpret the data 55, 48, 101, 62, 18, and 52.

Data clustering methods come in two broad varieties: hierarchical and non-hierarchical (or partitioning). Here we discuss the most typical representatives of each type. Further reading can be found in [46], in Chap. 11 of [47], in Chap. 12 of [41], and in specialized monographs [48–51].

5.8.1 Hierarchical Clustering

In hierarchical clustering the data are split in clusters of different sizes, among which some hierarchy is established. We initiate the process at the bottom of the hierarchy and interpret each data point as a cluster of its own. We locate the nearest two clusters, merge them, and repeat this until the last two clusters merge to a single cluster containing all data. This is the basic idea of *agglomerative* clustering. In the

second approach, known as *divisive* clustering, all data are understood as a single cluster which is gradually broken down into smaller clusters until we are left with as many clusters as there are data entries. Agglomerative clustering appears to have more adherents, and we discuss it in the following.

We represent each object (multivariate data item with m values) by the vector $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})^{\mathrm{T}}$, where $i = 0, 1, \ldots, n-1$. When the ranges of numerical values of the individual components differ widely, they should be standardized, $x_{ij} \leftarrow (x_{ij} - \overline{x}_j)/s_{x_j}$, where $\overline{x}_j = n^{-1} \sum_i x_{ij}$ and $s_{x_j} = (n-1)^{-1} \sum_i (x_{ij} - \overline{x}_j)^2$. To measure the distance between the objects we use

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left[\sum_{k=1}^{m} |x_{ik} - x_{jk}|^p \right]^{1/p},$$

in particular the cases $p = 2$ (Euclidean distance) and $p = 1$ (so-called Manhattan street distance). We use the n data entries to construct the symmetric $n \times n$ *proximity matrix* with the elements

$$D_{i,j} = \begin{cases} d(\boldsymbol{x}_i, \boldsymbol{x}_j); & i \neq j, \\ 0; & i = j, \end{cases} \tag{5.58}$$

where $i, j = 0, 1, \ldots, n-1$, and follow the algorithm [41]

Input: Multivariate data $\{\boldsymbol{x}_i\}_{i=0}^{n-1}$, each one is its own cluster
Compute the matrix $D^{(0)}$ by using (5.58).
for $s = 0$ **to** $n-1$ **do**
  Find the shortest distance $D_{I,J}$ in the matrix $D^{(s)}$.
  Merge clusters I and J into cluster IJ. // (*)
  Compute the distance $D_{IJ,K}$ between the new cluster IJ and all remaining clusters $K \neq IJ$ by using

$$D_{IJ,K} = \max\{D_{I,K}, D_{J,K}\} \quad \text{(complete linkage)}$$

  Construct a new $(n-s+1) \times (n-s+1)$ matrix $D^{(s+1)}$ by deleting the rows and columns I and J from $D^{(s)}$ and adding the row and column IJ with distances $D_{IJ,K}$. At the end of the loop $D^{(n-1)} = 0$.
end
Output: The list of clusters merged at step (*); the list of distances $D_{IJ,K}$ at the individual merge

Example Let us revisit the Old Faithful (Fig. 5.20). The final results of the algorithm above are the list of clusters that were merged into a larger cluster, and the list of distances $D_{IJ,K}$ at this merge. We illustrate this hierarchy graphically by a *dendrogram* (Fig. 5.21) in which the distances $D_{IJ,K}$ are represented by the different spacings between the sites where two clusters merge. A larger distance implies that the corresponding clusters are more disjunct. Vertical cuts through the branches of

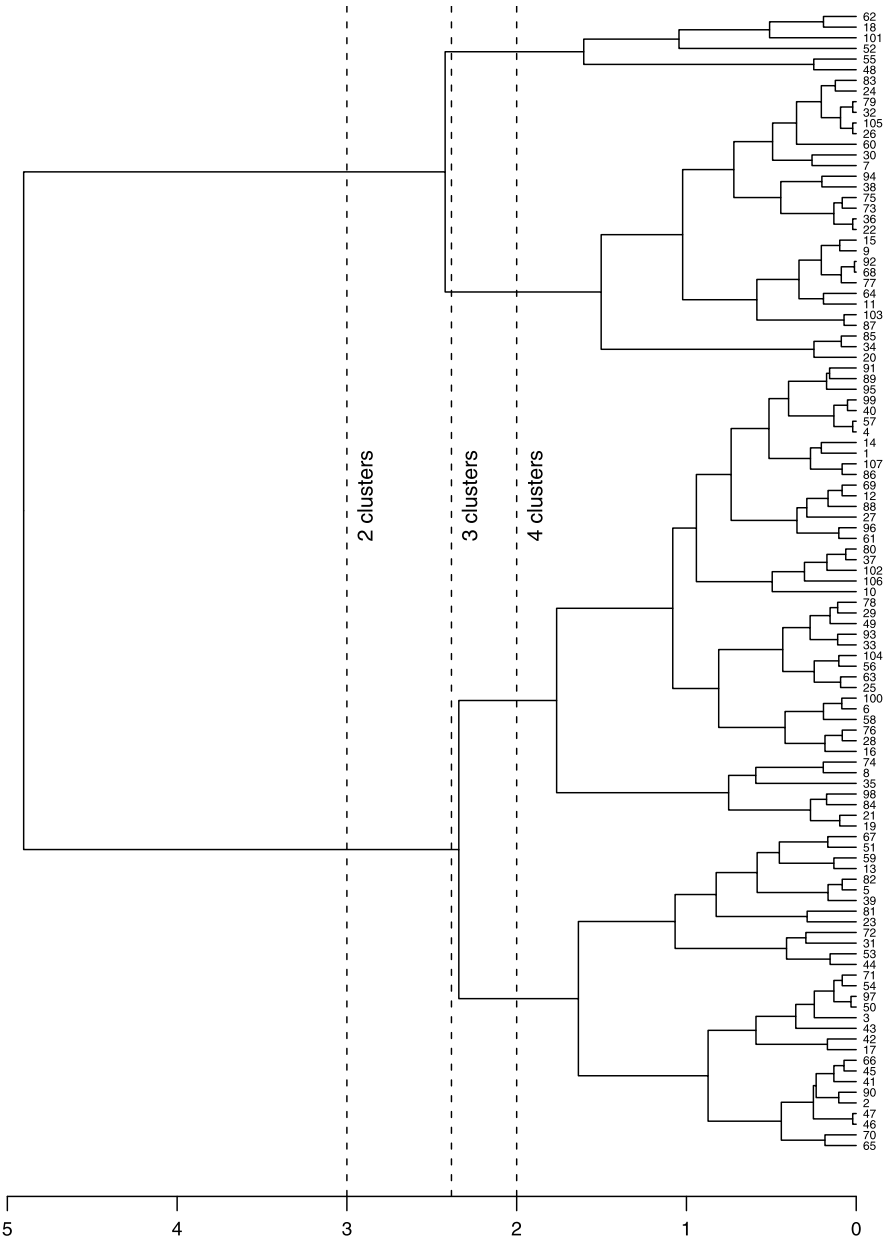


Fig. 5.21 Dendrogram for hierarchical clustering of the data from Fig. 5.20 by complete linkage. The intersections of vertical lines with the branches of the dendrogram define the number of classes (clusters). All elements to the right of the cut belong to the corresponding sub-cluster. The element 52 is misclassified in two-class clustering

the dendrogram reveal the number of clusters. According to the first possible cut (to the right of the value $D \approx 3$, two clusters) the elements from 62 to 20 belong to one cluster, while the elements from 91 to 65 belong to the other. According to the next possible cut (to the right of $D \approx 2.36$, three clusters) the elements from 62 to 48 fit into one, the elements from 83 to 20 into another, and those from 91 to 65 into the third cluster.

This algorithm has allowed us to perform clustering by *complete linkage* where we have determined the distance $D_{IJ,K}$ by finding the *maximum* value $\max\{D_{I,K}, D_{J,K}\}$. A typical outcome of such linkage is a large number of small, compact clusters. Another option could be to use *single linkage* where $D_{IJ,K}$ would be determined by finding the *minimum* values $\min\{D_{I,K}, D_{J,K}\}$. This tends to result in long chains of clusters made of just single data elements. The details on different types of linkage can be found in [49].

To compute and draw the dendrogram, we may resort to dedicated software. For example, in MATHEMATICA, we store the data x_{ij} in the matrix x and type

```
Needs["HierarchicalClustering`"]
x={{x_11,x_12,...,x_1m},...,{x_n1,x_n2,...,x_nm}};
DendrogramPlot[x,DistanceFunction->EuclideanDistance,
               LeafLabels->(#&)];
```

(Note that this command does not just draw the dendrogram but hides the complete clustering algorithm!) The same can be accomplished in the R environment [52, 53] which is a powerful suite of tools for statistical data analysis. In this environment the data should first be converted to the proximity matrix, then the clusters are formed, and finally the dendrogram is plotted:

```
require(graphics)
require(utils)
DATA <- matrix(scan("data.dat",0),ncol=2,byrow=TRUE)
hc <- hclust(dist(DATA),"complete")
dend <- as.dendrogram(hc)
plot(dend,horiz=TRUE)
```

If the input data already form relatively well delineated clusters, the final outcomes of hierarchical methods do not depend strongly on the specifics of linkage and the details of the algorithms. The main deficiency of the hierarchical methods is their unpredictability. Since the clustering proceeds by a stiff recipe which cannot be modified or adapted at any of the intermediate steps, the final distribution of the data into clusters can be arbitrarily bad. Moreover, hierarchical algorithms are numerically expensive: in naive implementations they require $\mathcal{O}(n^2)$ of memory and $\mathcal{O}(n^3)$ of processor time.

5.8.2 Partitioning Methods: k-Means

Many deficiencies of hierarchical methods can be avoided by using *partitioning methods*. Their main characteristic is that clustering is initiated with a predefined

number of clusters K, and that the classification into K clusters is not necessarily hierarchically related to a classification into another number of clusters: partitioning methods are therefore *non-hierarchical*. The number of clusters K is an input parameter that in many instances may be readily available. For example, in the bivariate data in Fig. 5.20 we will obviously attempt either $K = 2$ or $K = 3$; with multivariate data, the decision might be much harder.

A very popular approach is the method of *k-means*. In the standard implementation we randomly distribute the multivariate data set $\{\boldsymbol{x}_i\}_{i=0}^{n-1}$ over the chosen number of clusters K, and use the data within each group to determine their centers-of-mass (centroids). In the next step, each element is assigned to the centroid nearest to it, and we use this reordered configuration to compute the new centroids. We repeat this until the configuration no longer changes:

- Standardize the data as $x_{ij} \leftarrow (x_{ij} - \overline{x}_j)/s_{x_j}$, choose number of clusters K.
- Randomly distribute the data $\boldsymbol{x}_i$ into K clusters and compute the centroids (means) $\overline{\boldsymbol{x}}_k$ of each cluster $k = 1, 2, \ldots, K$.
- Compute the sum of squares of distances of each data point to its centroid,

$$\mathcal{D} = \sum_{k=1}^{K} \sum_{c(i)=k} \|\boldsymbol{x}_i - \overline{\boldsymbol{x}}_k\|_2^2, \tag{5.59}$$

  where $c(i)$ is the cluster containing the element $\boldsymbol{x}_i$.
- Locate the nearest centroid $\overline{\boldsymbol{x}}_k$ of the element $\boldsymbol{x}_i$ and reassign $\boldsymbol{x}_i$ to cluster k. After the reassignments, compute the new centroids $\overline{\boldsymbol{x}}_k$.
- Repeat from step (5.59) until no further readjustments of the elements are needed. Then $\mathcal{D}$ reaches its minimum. The final result is the list of elements within the clusters $c(i)$.

There is no need to randomize the initial assignments of the elements if we know better. For each presumed cluster the centroids can be computed in advance; for example, by looking at Fig. 5.20 and assuming $K = 2$ we might decide for $\overline{\boldsymbol{x}}_1 = (2, 50)$ and $\overline{\boldsymbol{x}}_2 = (4, 80)$, while with the assumption $K = 3$ we might start with $(2, 50)$, $(3, 60)$, and $(4, 80)$ (or corresponding standardized values).

Regardless of the initial configuration the algorithm converges very quickly: the numerical cost is just $\mathcal{O}(nK\nu)$, where ν is the number of iterations needed for convergence (typically $\nu \approx 3$). Because of its speed—and because it is so wonderfully simple to program—it is a popular tool to classify large data sets. It can also be used to roughly locate the cluster centroids, thereby providing the initial step for processing with other, more sophisticated algorithms. Figure 5.22 shows the initial configuration and the situation after the first iteration of the k-means algorithm for the geyser data from Fig. 5.20.

The method of k-means fails in classifying clusters with very different characters (e.g. with $K = 2$, in the case of one large and one small cluster, or of one compact and one elongated cluster, or of two approximately parallel elongated clusters as shown in Fig. 5.23). The method fails because it relies solely on the sum of distances (5.59) between the elements and the centroids, and is therefore insensitive to

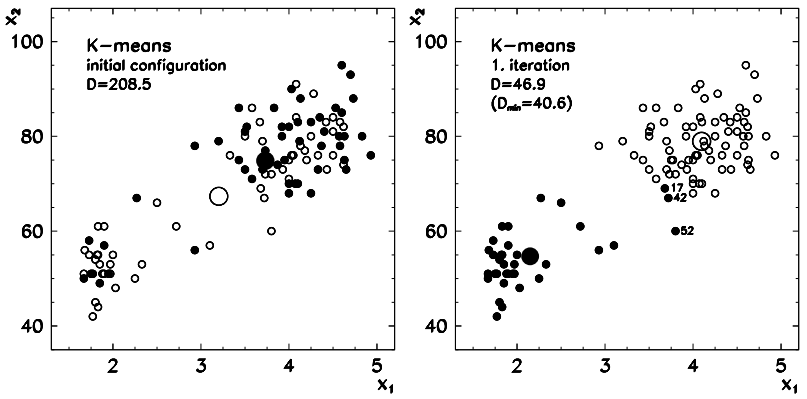


Fig. 5.22 Clustering bivariate data from Fig. 5.20 by using the method of k-means. [*Left*] At the beginning of the iteration the data are randomly distributed among both clusters. The centroids (*large symbols* • and ○) are therefore at completely wrong locations and the sum of distances $\mathcal{D} = 208.5$ (see (5.59)) is large. [*Right*] Situation after the first iteration. The data are already almost correctly classified, the centroids are at almost their final positions, and $\mathcal{D} = 46.9$ is near its final value 40.6. The iteration stops in the next step, in which the elements 17, 42, and 52 become rearranged

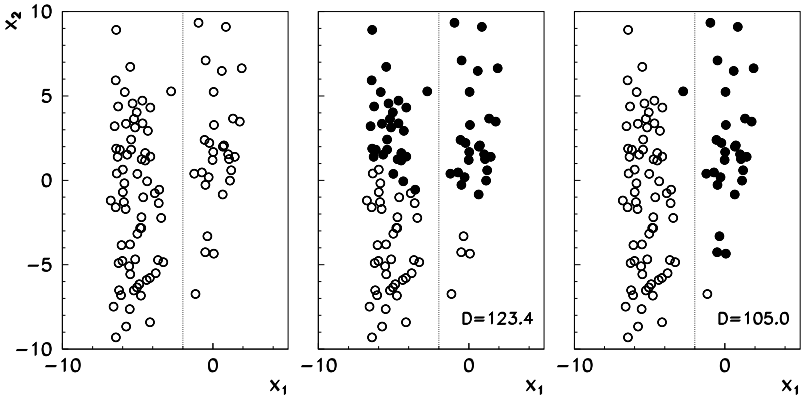


Fig. 5.23 The method of k-means sometimes fails. In such cases the algorithm typically locates just the local minimum of the sum of the squares of distances $\mathcal{D}$ (see (5.59)). [*Left*] Unclassified data. [*Center*] A wrong final configuration in a classification into two clusters (local minimum $\mathcal{D} = 123.4$). [*Right*] The correct final configuration (global minimum $\mathcal{D} = 105.0$)

the relative sizes or shapes of the clusters. One may therefore run the algorithm with several random initial configurations and ultimately accept the solution for which $\mathcal{D}$ reaches its global minimum (Fig. 5.23 (right)).

5.8.3 Gaussian Mixture Clustering and the EM Algorithm

The principle of the Gaussian mixture method is the search for a limited set of multivariate normal distributions that best reproduce the whole data set as a conglomerate of clusters (which may or may not overlap). In other words, we would like to frame the data $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})^{\mathrm{T}}$, where $i = 0, 1, \ldots, n-1$, into K ($k = 1, 2, \ldots, K$) multi-dimensional Gaussian distributions

$$N(\boldsymbol{x}; \boldsymbol{\mu}_k, \Sigma_k) = (2\pi)^{-m/2} (\det \Sigma_k)^{-1/2} \exp\left\{ -\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_k)^{\mathrm{T}} \Sigma_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k) \right\}. \quad (5.60)$$

Here $\boldsymbol{\mu}_k$ is the distribution mean (a vector of dimension m) and Σ_k the corresponding $m \times m$ covariance matrix. We determine the parameters $\boldsymbol{\mu}_k$ and Σ_k by using an iterative procedure called the EM (*expectation-maximization*) algorithm. With the EM algorithm we maximize the quantity

$$\mathcal{L} = \prod_{i=0}^{n-1} P(\boldsymbol{x}_i), \quad (5.61)$$

where $P(\boldsymbol{x}_i)$ is the probability that some point is found at $\boldsymbol{x}_i$. This probability is

$$P(\boldsymbol{x}_i) = \sum_{k=1}^{K} P(k) N(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \Sigma_k),$$

where $P(k)$ is the probability that a randomly chosen data point belongs to the cluster k. At the same time, $P(k)$ measures the fraction of all n data that belong to the cluster k. The corresponding probability for the kth component of the ith data point is then

$$P_{ik} = \frac{P(k) N(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \Sigma_k)}{\sum_k P(k) N(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \Sigma_k)} = \frac{P(k) N(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \Sigma_k)}{P(\boldsymbol{x}_i)}.$$

The computation of $\mathcal{L}$ and P_{ik} with the estimates $\widehat{\boldsymbol{\mu}}_k$, $\widehat{\Sigma}_k$, and $\widehat{P}(k)$, represents the *expectation step* (E) of the EM algorithm. We estimate the averages (centroids) of the K distributions, their covariance matrices, and probabilities $P(k)$, by

$$\widehat{\boldsymbol{\mu}}_k = \frac{\sum_i P_{ik} \boldsymbol{x}_i}{\sum_i P_{ik}}, \qquad \widehat{\Sigma}_k = \frac{\sum_i P_{ik} (\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)^{\mathrm{T}}}{\sum_i P_{ik}}, \qquad \widehat{P}(k) = \frac{1}{n} \sum_i P_{ik}.$$

These three estimates represent the second step (*maximization step*, M).

Example The algorithm is implemented iteratively. At the beginning we approximately determine the parameters $\widehat{\boldsymbol{\mu}}_k$, $\widehat{\Sigma}_k$, and $\widehat{P}(k)$. Figure 5.24 (left) shows

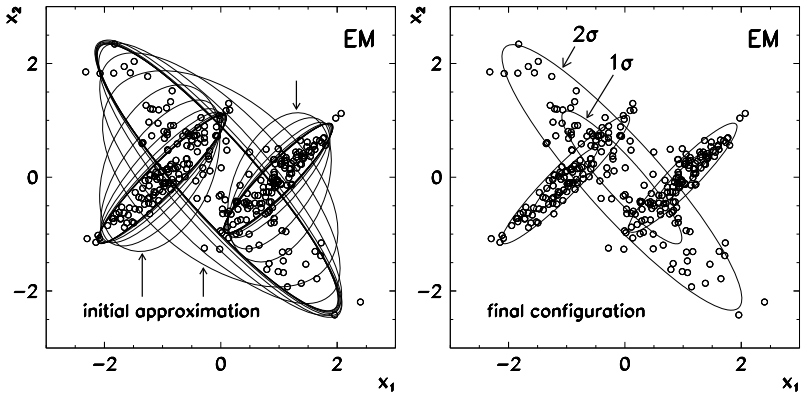


Fig. 5.24 Classification of $n = 300$ bivariate data into $K = 3$ clusters by the EM algorithm. [*Left*] Translation and rotation of the "2σ" covariance ellipses during the iteration. The semi-axes of the ellipses are $2r_0$ and $2r_1$ long (see (5.40)). [*Right*] The situation at the end of the iteration

$n = 300$ bivariate data ($m = 2$) which we would like to classify into three clusters ($K = 3$). We initialize the iteration with the estimates for the three centroids and three covariance matrices, for example,

$$\widehat{\boldsymbol{\mu}}_1 = (-1, 0), \qquad \widehat{\boldsymbol{\mu}}_2 = (0, 0), \qquad \widehat{\boldsymbol{\mu}}_3 = (1, 0), \qquad \widehat{\Sigma}_{1,2,3} = \begin{pmatrix} 0.25 & 0.00 \\ 0.00 & 0.05 \end{pmatrix}$$

and $\widehat{P}(1) = \widehat{P}(2) = \widehat{P}(3) = 1/3$, and use them to compute $\mathcal{L}$ and P_{ik} (step E). Then we compute the new estimates $\widehat{\boldsymbol{\mu}}_k$, $\widehat{\Sigma}_k$, and $\widehat{P}(k)$ (step M). We repeat this procedure until $\mathcal{L}$ stops increasing. (Multiple tries with different initial configurations may convince us that we have indeed reached the global, not a local, maximum.) During the iteration the covariance ellipses entangle the individual data clusters ever more closely, and after typically $\approx$10 rounds the procedure converges to the final Gaussian mixture that optimally fits the data and delineates K clusters. The final configuration for this case is shown in Fig. 5.24 (right).

For large n, many small probabilities are multiplied in (5.61); as a result, range underflows in finite-precision arithmetic may occur. The whole procedure should therefore be implemented solely by taking the logarithms of all quantities: instead of $\mathcal{L} = \prod_i P(\boldsymbol{x}_i)$ we should use $\log \mathcal{L} = \sum_i P(\boldsymbol{x}_i)$ and

$$\log N(\boldsymbol{x}; \widehat{\boldsymbol{\mu}}_k, \widehat{\Sigma}_k) = -\frac{1}{2}(\boldsymbol{x} - \widehat{\boldsymbol{\mu}}_k)^{\mathrm{T}} \widehat{\Sigma}_k^{-1} (\boldsymbol{x} - \widehat{\boldsymbol{\mu}}_k) - \frac{m}{2} \log 2\pi - \frac{1}{2} \log \det \widehat{\Sigma}_k .$$

Additional numerical details can be found in [25]; see also [44].

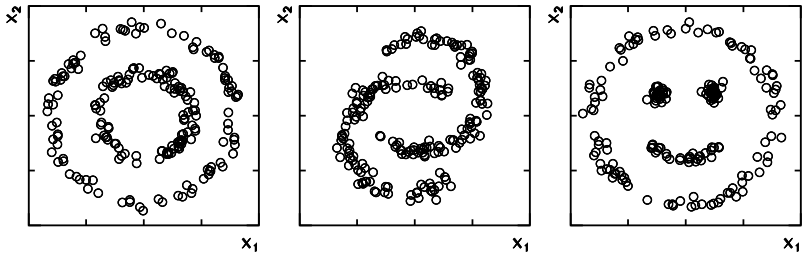


Fig. 5.25 Typical examples of bivariate data for which standard clustering methods fail. They can be handled effortlessly by spectral clustering

5.8.4 Spectral Methods

It is not hard to imagine a layout of bivariate data for which all approaches discussed so far (the hierarchical agglomerative method, the method of k-means, and the EM algorithm) will fail. Typical examples are shown in Fig. 5.25 hinting at problematic types of clusters like concentric regions or clusters that bite into one another. One of the most powerful modern methods of unsupervised learning from such data is *spectral clustering*.

The foundation of the method is, again, a kind of proximity matrix. But by applying a few simple transformations of this matrix the problematic data regions can be converted into more compact groups which are easier to cluster by simpler algorithms. A good introduction to these spectacularly effective methods is offered by the paper [54]. From [55] we adopt the algorithm for spectral clustering of multivariate data $\boldsymbol{x}_i$ $(i = 0, 1, \ldots, n-1)$ into K clusters:

- Use the data $\{\boldsymbol{x}_i\}_{i=0}^{n-1}$ to construct the *affinity matrix*

$$A_{ij} = \begin{cases} \exp[-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/(2\sigma^2)]; & i \neq j, \\ 0; & i = j. \end{cases}$$

  The way to determine the parameter σ is described below. The matrix A contains all information about the distances between the data. By using A, compute

$$L = D^{-1/2} A D^{-1/2}, \quad D = \mathrm{diag}(d_0, d_1, \ldots, d_{n-1}),\ d_i = \sum_{j=0}^{n-1} A_{ij}.$$

- Find the K eigenvectors $\boldsymbol{v}_k$ corresponding to the largest eigenvalues of L, and arrange them in the columns of the matrix $V = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_K) \in \mathbb{R}^{n \times K}$. Normalize the rows of V to unit length,

$$W_{ij} = V_{ij} \left[\sum_{k=1}^{K} V_{ik}^2 \right]^{-1/2}.$$

- Each row of the matrix $W \in \mathbb{R}^{n\times K}$ is a new point $\boldsymbol{w}_i$ (multivariate data entry) in space $\mathbb{R}^K$. By the method of k-means (Sect. 5.8.2) classify the points $\boldsymbol{w}_i$ into K clusters. Choose the parameter σ such that the points accumulate in compact, well separated groups (sometimes easily discernible). These groups are easy to handle by the k-means algorithm.
- Assign the original data points $\boldsymbol{x}_i$ to the cluster j precisely in the case that the ith row of the matrix W has been assigned to cluster j.

5.9 Linear Discriminant Analysis ⋆

Assume that an experiment gives us n measurements of m variables. Suppose we anticipate that the results of each of the n measurements will originate in one of the two possible classes R_r, $r \in \{1, 2\}$. A well-known example is the analysis of n microscopic samples of needle biopsies of potentially cancerous tissues, where each sample allows us to determine m properties of the tissue cells [56]. For example, one can monitor the shape, the symmetry, the size, and other cell properties, and arrange the ith measurement of all m variables in the vector

$$\boldsymbol{x}_{ri} = \left(x_1^{(r)}, x_2^{(r)}, \ldots, x_m^{(r)}\right)_i^{\mathrm{T}}, \quad i = 0, 1, \ldots, n-1, \ r \in \{1, 2\}. \tag{5.62}$$

If, by surgical methods, we realize that a microscopic sample corresponds to a benign tissue, we assign it to the class $r = 1$; if it is malignant, we assign it to the class $r = 2$.

Discriminant analysis allows us to solve two sequential problems. Firstly, we wish to devise a tool for classification into classes R_1 and R_2 on the basis of unambiguously identified n_1 benign and $n_2 = n - n_1$ malignant tissue samples. Secondly, we would like to apply this very classification tool to determine the class of a new, as yet unclassified observation (sample). This is one of the aspects distinguishing discriminant analysis from clustering analysis (Sect. 5.8): in the former the number of classes is known in advance, while in the latter both the number of classes (clusters) and the class assignments are unknown beforehand.

5.9.1 Binary Classification

Here we discuss the simplest case of *binary classification*, where only two classes are involved. We refer to the Bayes theorem of conditional probabilities. We assume that the prior probabilities

$$\mathcal{P}(\boldsymbol{\xi} \in R_r) = \mathcal{P}_r, \quad r = 1, 2,$$

are already known, i.e. we know the probabilities that a randomly chosen measurement $\boldsymbol{\xi} = \boldsymbol{x}$ belongs to either class R_1 or class R_2. If we do not know them, we can

estimate them from the population sample at hand: in the cancerous tissue example above, we can simply set $\mathcal{P}_1 = n_1/n$ and $\mathcal{P}_2 = n_2/n$. Suppose that the conditional probability density for class r is

$$p(\boldsymbol{\xi} = \boldsymbol{x} | \boldsymbol{\xi} \in R_r) = p_r(\boldsymbol{x}). \tag{5.63}$$

By using Bayes theorem $\mathcal{P}(A|B) = p(B|A)\mathcal{P}(A)/p(B)$, where $\mathcal{P}$ denote the probabilities and p the probability densities, we obtain the posterior probabilities

$$\widetilde{\mathcal{P}}_r(\boldsymbol{x}) = \mathcal{P}(\boldsymbol{\xi} \in R_r | \boldsymbol{\xi} = \boldsymbol{x}) = \frac{p_r(\boldsymbol{x})\mathcal{P}_r}{p_1(\boldsymbol{x})\mathcal{P}_1 + p_2(\boldsymbol{x})\mathcal{P}_2}, \tag{5.64}$$

which are the probabilities that the observed $\boldsymbol{x}$ belongs to the class R_1 or R_2. In binary classification we choose R_1 to be the class corresponding to the larger prior probability $\mathcal{P}_1$. This means that the observed $\boldsymbol{\xi}$ should be assigned to R_1 if $\widetilde{\mathcal{P}}_1(\boldsymbol{x})/\widetilde{\mathcal{P}}_2(\boldsymbol{x}) > 1$, or to R_2 otherwise. We can use (5.64) to assign $\boldsymbol{x}$ to R_1 if $p_1(\boldsymbol{x})/p_2(\boldsymbol{x}) > \mathcal{P}_2/\mathcal{P}_1$, or to R_2 otherwise. For practical use we prefer to define

$$L(\boldsymbol{x}) = \log \frac{p_1(\boldsymbol{x})\mathcal{P}_1}{p_2(\boldsymbol{x})\mathcal{P}_2} \tag{5.65}$$

and classify

$$L(\boldsymbol{x}) > 0 \iff \boldsymbol{x} \in R_1, \qquad L(\boldsymbol{x}) < 0 \iff \boldsymbol{x} \in R_2. \tag{5.66}$$

Usually we assume that the probability densities $p_1(\boldsymbol{x})$ and $p_2(\boldsymbol{x})$ correspond to the normal distributions, and that in general they have different means $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, yet equal covariance matrices $\Sigma_1 = \Sigma_2 = \Sigma_{xx}$, thus

$$p_r(\boldsymbol{x}) = (2\pi)^{-m/2} |\Sigma_{xx}|^{-1/2} \exp\left[-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_r)^{\mathrm{T}} \Sigma_{xx}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_r) \right].$$

(The procedure for $\Sigma_1 \neq \Sigma_2$ is described in [41], Sect. 8.3.7.) When this is plugged in (5.65) the function L can be written as $L(\boldsymbol{x}) = a_0 + \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}$, where

$$a_0 = -\frac{1}{2} \left\{ \boldsymbol{\mu}_1^{\mathrm{T}} \Sigma_{xx}^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^{\mathrm{T}} \Sigma_{xx}^{-1} \boldsymbol{\mu}_2 \right\} - \log \frac{\mathcal{P}_2}{\mathcal{P}_1}, \qquad \boldsymbol{a} = \Sigma_{xx}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2).$$

In practice we are only dealing with finite population samples, so we do not know the exact $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, and Σ_{xx}. But we can estimate them: we compute the vectors $\widehat{\boldsymbol{\mu}}_1$ in $\widehat{\boldsymbol{\mu}}_2$ (both of dimension m) by averaging over all n_1 and n_2 measurements,

$$\widehat{\boldsymbol{\mu}}_r = \overline{\boldsymbol{x}}_r = \frac{1}{n_r} \sum_{i=0}^{n_r - 1} \boldsymbol{x}_{ri}, \quad r = 1, 2$$

(see definition (5.62)). We estimate the $m \times m$ covariance matrix by

$$\widehat{\Sigma}_{xx} = \frac{1}{n_1 + n_2} \left[S_{xx}^{(1)} + S_{xx}^{(2)} \right],$$

where

$$S_{xx}^{(r)} = \sum_{i=0}^{n_r-1} (\boldsymbol{x}_{ri} - \overline{\boldsymbol{x}}_r)(\boldsymbol{x}_{ri} - \overline{\boldsymbol{x}}_r)^{\mathrm{T}}, \quad r = 1, 2.$$

The discriminant function is then

$$\widehat{L}(\boldsymbol{x}) = \widehat{a}_0 + \widehat{\boldsymbol{a}}^{\mathrm{T}}\boldsymbol{x}, \tag{5.67}$$

where

$$\widehat{a}_0 = \frac{1}{2}\left\{\overline{\boldsymbol{x}}_2^{\mathrm{T}}\widehat{\Sigma}_{xx}^{-1}\overline{\boldsymbol{x}}_2 - \overline{\boldsymbol{x}}_1^{\mathrm{T}}\widehat{\Sigma}_{xx}^{-1}\overline{\boldsymbol{x}}_1\right\} + \log\frac{n_1}{n_2}, \qquad \widehat{\boldsymbol{a}} = \widehat{\Sigma}_{xx}^{-1}(\overline{\boldsymbol{x}}_1 - \overline{\boldsymbol{x}}_2).$$

We have described the most basic procedure of binary classification with the discrimination function (5.67). This function is linear in the variables $\boldsymbol{x}$ and we have therefore performed a *linear discriminant analysis* (LDA). When some sample from a two-class population (n measurements of m variables belonging either to class R_1 or to class R_2) has been used to determine the $m + 1$ parameters a_0 and $\boldsymbol{a}$ of the function L, the criterion (5.66) allows us to classify all further measurements $\boldsymbol{x}$. We simply compute the value of $L(\boldsymbol{x})$ for each newly acquired $\boldsymbol{x}$ and check its sign: if it is negative, $\boldsymbol{x}$ belongs to R_1, otherwise to R_2.

The quality of the classification can be tested as follows. From the complete set of n measurements we omit a single measurement (one vector of dimension m) and estimate the parameters a_0 and $\boldsymbol{a}$ of L from the remaining $n - 1$ data entries. Then the omitted entry is classified according to this modified estimate for L. We repeat this procedure for all n measurements and obtain the values

n_{11}—data that belong to R_1 and were indeed assigned to R_1;
n_{12}—data that belong to R_2 but were assigned to R_1;
n_{21}—data that belong to R_1 but were assigned to R_2;
n_{22}—data that belong to R_2 and were indeed assigned to R_2.

Of course $n_{11} + n_{12} + n_{21} + n_{22} = n_1 + n_2 = n$. Ideally we would like to achieve $n_{11} = n_1$ and $n_{22} = n_2$, but in practice we almost invariably find $n_{12} \neq 0$ and/or $n_{21} \neq 0$. The measure for the inefficiency of binary classification (the *misclassification rate*) is the ratio $(n_{12} + n_{21})/n$. In the histogrammed measurements the classification inefficiency is apparent in the areas where the distribution for $L > 0$ leaks across the $L = 0$ boundary to the side with $L < 0$ and vice versa (see Fig. 5.26).

5.9.2 Logistic Discriminant Analysis

The usual linear discrimination analysis relies on a linear discriminant function (5.67) and is therefore strongly sensitive to outliers. Obviously, we may encounter problems of the same type by analyzing data that are not distributed normally or data from two classes with strongly differing covariance matrices. To a

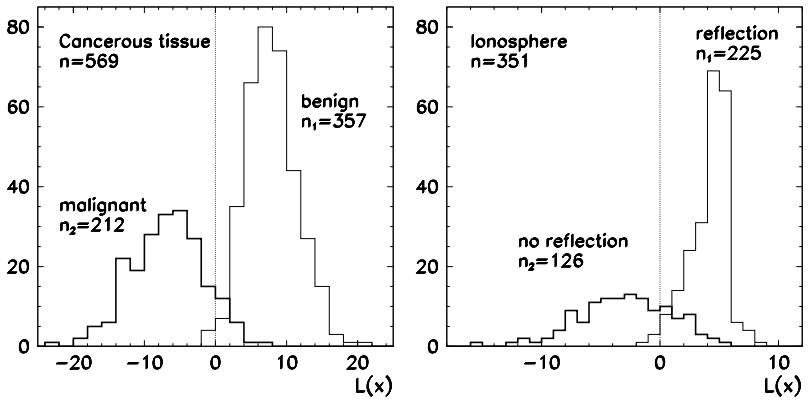


Fig. 5.26 Distribution of the measurements with respect to the linear discriminant function $L(\boldsymbol{x})$. [*Left*] Cancerous tissue (misclassification rate 4.2 %). [*Right*] Reflections of radar signals in the ionosphere (Problem 5.12.3; misclassification rate 13.7 %)

large extent, these obstacles can be overcome by *logistic discriminant analysis*. The assignment of an individual measurement $\boldsymbol{x}$ to one of the classes is recorded by keeping track of the dependent variable

$$y = \begin{cases} 1; & \boldsymbol{x} \in R_1, \\ 0; & \boldsymbol{x} \in R_2. \end{cases}$$

Then we collect the data into pairs $(\boldsymbol{x}, y)_i$ and finally compute the parameters $\widehat{a}_0$ and $\widehat{\boldsymbol{a}}$ directly by maximizing the conditional probability

$$\mathcal{L}(a_0, \boldsymbol{a}) = \prod_{i=0}^{n-1} \big[p_1(\boldsymbol{x}_i; a_0, \boldsymbol{a})\big]^{y_i} \big[1 - p_1(\boldsymbol{x}_i; a_0, \boldsymbol{a})\big]^{1-y_i},$$

where $p_1(\boldsymbol{x}) = p(\boldsymbol{\xi} = \boldsymbol{x} | \boldsymbol{\xi} \in R_1)$ (see (5.63)). Of course, the method has its deficiencies. Among others, it requires substantially larger data samples than linear discrimination for the same asymptotic efficiency. An efficient iterative scheme for the maximization of $\mathcal{L}$ with respect to a_0 and $\boldsymbol{a}$ and other details can be found in [41], Sect. 8.3.5.

5.9.3 Assignment to Multiple Classes

Multi-class discriminant analysis is a generalization of two-class analysis to multiple classes. Initial reading is offered by [41], Sect. 8.5, and [47], Chap. 12.

5.10 Canonical Correlation Analysis ⋆

In addition to the principal-component analysis (Sect. 5.7), one of the most widespread and generally useful methods of multivariate analysis is the *canonical correlation analysis* (CCA). The CCA method allows us to handle sets of data of the form

$$\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^{\mathrm{T}}, \qquad \boldsymbol{y}_i = (y_{i1}, y_{i2}, \ldots, y_{iq})^{\mathrm{T}}, \quad i = 0, 1, \ldots, n-1,$$

where $p \neq q$ in general. The basic premise is that the correlation between $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ is the most relevant carrier of information for this data set. By CCA we wish to reduce the dimensionality of the data by projecting $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ to a smaller set of canonical variables, among which we attempt to impose maximum correlation. From this viewpoint, the task of CCA is precisely opposite to the task of PCA, where data should become *decorrelated*.

An example are the $n = 3462$ multivariate data on measured properties of galaxies (Problem 5.12.4, adapted from [41] based on data [57–59]). Each of the n measurements represents a data point $\boldsymbol{x}_i$ ($0 \le i \le n-1$) with $p = 23$ values describing the brightnesses and luminosities of galaxies in various spectral ranges, while each $\boldsymbol{y}_i$ stores $q = 6$ variables describing other physical properties. We are interested in correlations between the individual components of these data.

From the data we estimate the means $\overline{\boldsymbol{x}} = (\sum_{i=0}^{n-1} \boldsymbol{x}_i)/n$ and $\overline{\boldsymbol{y}} = (\sum_{i=0}^{n-1} \boldsymbol{y}_i)/n$, and use them to estimate the covariance matrices

$$\Sigma_{xx} = \frac{1}{n-1} \sum_{i=0}^{n-1} (\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^{\mathrm{T}}, \qquad \Sigma_{yy} = \frac{1}{n-1} \sum_{i=0}^{n-1} (\boldsymbol{y}_i - \overline{\boldsymbol{y}})(\boldsymbol{y}_i - \overline{\boldsymbol{y}})^{\mathrm{T}},$$

and

$$\Sigma_{xy} = \Sigma_{yx}^{\mathrm{T}} = \frac{1}{n-1} \sum_{i=0}^{n-1} (\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{y}_i - \overline{\boldsymbol{y}})^{\mathrm{T}}.$$

For general $\boldsymbol{x}$ and $\boldsymbol{y}$ we construct the linear combinations

$$\begin{aligned} \xi_j &= a_{1j} x_1 + a_{2j} x_2 + \cdots + a_{pj} x_p = \boldsymbol{a}_j^{\mathrm{T}} \boldsymbol{x}, \\ \zeta_j &= b_{1j} y_1 + b_{2j} y_2 + \cdots + b_{qj} y_q = \boldsymbol{b}_j^{\mathrm{T}} \boldsymbol{y}, \end{aligned} \tag{5.68}$$

where $j = 1, 2, \ldots, r$ and $r \le \min(p, q)$. The coefficient vectors

$$\boldsymbol{a}_j = (a_{1j}, a_{2j}, \ldots, a_{pj})^{\mathrm{T}}, \qquad \boldsymbol{b}_j = (b_{1j}, b_{2j}, \ldots, b_{qj})^{\mathrm{T}},$$

are determined—this is the key step of CCA—such that the pairs (ξ_j, ζ_j) are arranged in decreasing degree of linear correlation

$$\rho_j = \mathrm{corr}(\xi_j, \zeta_j) = \frac{\boldsymbol{a}_j^{\mathrm{T}} \Sigma_{xy} \boldsymbol{b}_j}{\sqrt{\boldsymbol{a}_j^{\mathrm{T}} \Sigma_{xx} \boldsymbol{a}_j} \sqrt{\boldsymbol{b}_j^{\mathrm{T}} \Sigma_{yy} \boldsymbol{b}_j}}, \quad j = 1, 2, \ldots, r, \tag{5.69}$$

so that $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_r$. Moreover, ξ_j should be uncorrelated to all ξ_k with lower indices,

$$\operatorname{cov}(\xi_j, \xi_k) = \boldsymbol{a}_j^{\mathrm{T}} \Sigma_{xx} \boldsymbol{a}_k = 0, \quad k < j,$$

and analogously for all ζ_j,

$$\operatorname{cov}(\zeta_j, \zeta_k) = \boldsymbol{b}_j^{\mathrm{T}} \Sigma_{yy} \boldsymbol{b}_k = 0, \quad k < j.$$

In plain words, arbitrary correlations between the original variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are thus systematically relocated to an ordered sequence of decreasing correlations between the pairs (ξ_j, ζ_j) which are known as *canonical variates*, while the corresponding degrees of correlations between them (5.69) are the *canonical correlation coefficients*.

The coefficients vectors $\boldsymbol{a}_j$ and $\boldsymbol{b}_j$ are obtained after a short calculation in which the correlation $\operatorname{corr}(\xi, \zeta) = \boldsymbol{a}^{\mathrm{T}} \Sigma_{xy} \boldsymbol{b}$ is maximized. The full derivation can be found in [41], here we just quote the final result. We compute the products

$$\Omega_a = \Sigma_{xx}^{-1/2} \Sigma_{xy} \Sigma_{yy}^{-1} \Sigma_{yx} \Sigma_{xx}^{-1/2}, \qquad \Omega_b = \Sigma_{yy}^{-1/2} \Sigma_{yx} \Sigma_{xx}^{-1} \Sigma_{xy} \Sigma_{yy}^{-1/2},$$

and determine the eigenvectors and eigenvalues λ_j of the symmetric matrices $\Omega_a \in \mathbb{R}^{p \times p}$ and $\Omega_b \in \mathbb{R}^{q \times q}$. We order the eigenvalues in decreasing order and arrange the corresponding eigenvectors accordingly. We get

$$\boldsymbol{a}_j^{\mathrm{T}} = \sqrt{\lambda_j} \boldsymbol{u}_j^{\mathrm{T}} \Sigma_{xx}^{-1/2}, \qquad \boldsymbol{b}_j^{\mathrm{T}} = \boldsymbol{v}_j^{\mathrm{T}} \Sigma_{yy}^{-1/2}, \tag{5.70}$$

where $\boldsymbol{u}_j$ is the eigenvector of the matrix Ω_a corresponding to the eigenvalue λ_j, and $\boldsymbol{v}_j$ is the eigenvector of Ω_b (the first $r = \min(p, q)$ eigenvalues of Ω_a and Ω_b are identical). In the above expressions we need the "square roots" of the matrices Σ_{xx} and Σ_{yy}. Appendix A.7 teaches you how to do this.

The coefficient vectors (5.70) are all we need in the last analysis step. We use them to compute (5.68) for an arbitrary pair of data $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ $(i = 0, 1, \ldots, n-1)$. This gives us the *canonical variate scores*

$$\xi_{ij} = \boldsymbol{a}_j^{\mathrm{T}} \boldsymbol{x}_i, \qquad \zeta_{ij} = \boldsymbol{b}_j^{\mathrm{T}} \boldsymbol{y}_i, \quad j = 1, 2, \ldots, r, \tag{5.71}$$

where for each j we also compute the corresponding correlation coefficient (5.69). If some significant correlation is present in the original data, it should express itself most acutely in a two-dimensional rendering of the set of pairs (ξ_{i1}, ζ_{i1}). The next set of pairs (ξ_{i2}, ζ_{i2}) will most likely exhibit a smaller correlation, the third set (ξ_{i3}, ζ_{i3}) an even smaller one, and so on.

The images of canonical variate scores often reveal unexpected structures in the data or highlight outliers which would remain hidden in the original variables. A basic feel for the method is conveyed by Problem 5.12.4. Further reading on CCA can be found in Chap. 14 of [47] and Sect. 7.3 of [41].

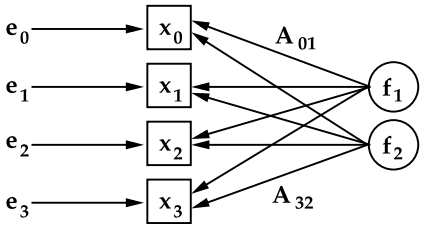


Fig. 5.27 A schematic representation of factor analysis. We construct a linear combination of a small number of factors (two in this example, $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$) and remainders $\boldsymbol{e}_i$ in order to represent a larger body of data $\boldsymbol{x}_i$. The weights are stored in the matrix A such that for each data entry we have $\boldsymbol{x}^{\mathrm{T}} = A\boldsymbol{f}^{\mathrm{T}} + \boldsymbol{e}^{\mathrm{T}}$, where $\boldsymbol{x}$, $\boldsymbol{f}$, and $\boldsymbol{e}$ are the corresponding rows of the matrices X, F, and E

5.11 Factor Analysis ⋆

Factor analysis is a multivariate statistical method in which we strive to explain a set of measured data by a much smaller number of variables known as *factors*. The factors do not necessarily correspond to physically measurable quantities and should therefore be considered as *latent* variables used only for a formal parameterization of the data. Classical factor analysis therefore belongs to the class of *latent variable models*; see Sects. 15.4 and 15.5 in [41]. In the context of time series we encounter one such model in Sect. 6.8. Numerous fields of application of factor analysis in natural sciences are discussed in [60].

Depending on how the data and the factors are connected, several types of factor analysis exist. By far the most common is a linear relation,

$$X = FA^{\mathrm{T}} + E. \tag{5.72}$$

The $n \times m$ matrix X contains n multivariate data $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})^{\mathrm{T}}$, one per row. The $n \times k$ matrix F contains the *factor scores* for the ith data. The number of factors k which, as a rule, is much smaller than the number of data components m, is chosen in advance. The $m \times k$ matrix A contains the *factor loadings* which determine the role of a given factor in the data $\boldsymbol{x}_i$ (see Fig. 5.27). The $n \times m$ matrix E contains the remainders which represent data errors.

In (5.72) only the data matrix X is known, while we are completely ignorant of all matrices at the right-hand side of the equation. A unique solution without additional assumptions does not exist. From many possible ways [60] we follow, as an illustration, the path of *principal-component factor analysis*. In this approach the factors are obtained by the additional requirement that the factors should describe the maximum variance of all measured data X.

The data should first be centered as in (5.47). The factors and the weights are then determined by fitting the matrix product FA^{T} to the data X_{c} in the least-squares sense. This means that for the chosen number of factors $k < m$ we must determine F and A^{T} such that the sum of squares of the elements of

$$E = X_{\mathrm{c}} - FA^{\mathrm{T}}$$

will be minimal. We know the solution of this problem from Sect. 3.3. The optimal approximation of the rank-r matrix $X \in \mathbb{R}^{n\times m}$ by the matrix $W \in \mathbb{R}^{n\times m}$ of a lower rank $k < r$ in the least-squares sense is given by the singular-value decomposition $X = U\Gamma V^{\mathrm{T}}$, such that

$$FA^{\mathrm{T}} = \gamma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^{\mathrm{T}} + \gamma_2 \boldsymbol{u}_2 \boldsymbol{v}_2^{\mathrm{T}} + \cdots + \gamma_k \boldsymbol{u}_k \boldsymbol{v}_k^{\mathrm{T}} = U_k \Gamma_k V_k^{\mathrm{T}}, \tag{5.73}$$

where $\boldsymbol{u}_i$ $(1 \le i \le k)$ are the first k columns of U, and $\boldsymbol{v}_i$ $(1 \le i \le k)$ are the first k columns of V. The vectors $\boldsymbol{u}_i$ (dimension n) and $\boldsymbol{v}_i$ (dimension m) are arranged in decreasing order of the corresponding singular values in the matrix $\Gamma_k = \mathrm{diag}(\gamma_1, \gamma_2, \ldots, \gamma_k)$. In this approach we neglect the measurement errors (remainders) $\boldsymbol{e}$, that is, we assume $\|E\| \ll \|X_{\mathrm{c}}\|$ in some matrix norm.

5.11.1 Determining the Factors and Weights from the Covariance Matrix

Since the singular values of X are closely related to the eigenvalues of the product $X^{\mathrm{T}}X$ (see Appendix A.6), we prefer to use the covariance matrix

$$\widehat{\Sigma}_{xx} = \frac{1}{n-1} X_{\mathrm{c}}^{\mathrm{T}} X_{\mathrm{c}} \tag{5.74}$$

instead of the centered data matrix X_{c}. At first sight, the decomposition (5.73) offers two solutions. The first possibility is

$$F = U_k, \qquad A^{\mathrm{T}} = \Gamma_k V_k^{\mathrm{T}}. \tag{5.75}$$

The singular values γ_i from the decomposition of the data matrix X_{c} are the square roots of the singular values λ_i from the decomposition of the covariance matrix $\widehat{\Sigma}_{xx} = U_k \Lambda_k V_k^{\mathrm{T}}$, thus we obtain

$$A = \left(\Gamma_k V_k^{\mathrm{T}}\right)^{\mathrm{T}} = V_k \Gamma_k = V_k \Lambda_k^{1/2} \tag{5.76}$$

and

$$F = U_k = X_{\mathrm{c}} V_k \Gamma_k^{-1} = X_{\mathrm{c}} A \Gamma_k^{-1} \Gamma_k^{-1} = X_{\mathrm{c}} A \Lambda_k^{-1}. \tag{5.77}$$

By following this procedure we obtain factors that are uncorrelated and have unit variance. The columns of the weight matrix A are directly comparable to each other. The matrix of correlations *between the data and the factors* is

$$C = S^{-1} A = S^{-1} V_k \Lambda_k^{1/2}, \tag{5.78}$$

where $S \in \mathbb{R}^{m\times m}$ is diagonal: its diagonal elements are the standard deviations of the data components,

$$S = \mathrm{diag}(\sqrt{\widehat{\Sigma}_{11}}, \sqrt{\widehat{\Sigma}_{22}}, \ldots, \sqrt{\widehat{\Sigma}_{mm}}).$$

The first method of determining the factors is then:

1. Compute the covariance matrix (5.74) and its singular-value decomposition $U_k \Lambda_k V_k^{\mathrm{T}}$.
2. Compute the weight matrix A by (5.76) and the factor matrix F by (5.77).
3. The data-factors correlation is then given by (5.78).

The second possibility that can be guessed from the decomposition (5.73) is

$$F = U_k \Gamma_k, \qquad A^{\mathrm{T}} = V_k^{\mathrm{T}}. \tag{5.79}$$

Again we compute the singular values of the covariance matrix instead of the singular values of the data matrix, and use $U_k = X_{\mathrm{c}} V_k \Gamma_k^{-1}$. We obtain

$$A = V_k, \qquad F = X_{\mathrm{c}} V_k \Gamma_k^{-1} \Gamma_k = X_{\mathrm{c}} V_k = X_{\mathrm{c}} A. \tag{5.80}$$

This option also gives us mutually uncorrelated factors, but they have different variances, and the columns of the weight matrix A are therefore not directly comparable. Nonetheless, the matrix of correlations between the data and the factors is still given by (5.78). We apply the following procedure.

1. Compute the covariance matrix (5.74) and its singular-value decomposition $U_k \Lambda_k V_k^{\mathrm{T}}$.
2. Compute the weight matrix A and the factor matrix F by (5.80).
3. The data-factors correlation is given by (5.78).

Example (Adapted from [60]) In some measurement we have acquired a set of $n = 10$ multivariate data with $m = 5$ components each, $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{i5})^{\mathrm{T}}$. We assume that the whole data set can be described by only two factors, which we would like to determine such that individual data components are well correlated to either one of the factors. We arrange the data in the matrix

$$X = \begin{pmatrix} 5.0 & 25.0 & 15.0 & 5.0 & 5.0 \\ 10.0 & 30.0 & 17.0 & 17.0 & 8.0 \\ 3.0 & 6.0 & 10.0 & 13.0 & 25.0 \\ 7.5 & 27.2 & 16.0 & 11.0 & 6.5 \\ 4.6 & 21.9 & 14.0 & 6.6 & 9.0 \\ 3.8 & 13.6 & 12.0 & 9.8 & 17.0 \\ 8.3 & 26.6 & 15.9 & 14.2 & 9.1 \\ 6.1 & 22.7 & 14.6 & 10.2 & 9.9 \\ 7.6 & 24.2 & 15.2 & 13.8 & 10.8 \\ 3.9 & 10.3 & 11.2 & 12.6 & 21.3 \end{pmatrix}. \tag{5.81}$$

From each row of X we subtract the mean vector

$$\boldsymbol{x} = \frac{1}{n} \sum_{i=0}^{n-1} \boldsymbol{x}_i = (5.98, 20.75, 14.09, 11.32, 12.16)^{\mathrm{T}},$$

and get the centered data matrix X_c by which we compute the covariance matrix

$$\widehat{\Sigma}_{xx} = \begin{pmatrix} 5.257 & 15.768 & 4.739 & 4.585 & -10.150 \\ 15.768 & 63.818 & 18.209 & 1.569 & -50.618 \\ 4.739 & 18.209 & 5.241 & 1.177 & -14.044 \\ 4.585 & 1.569 & 1.177 & 13.006 & 6.068 \\ -10.150 & -50.618 & -14.044 & 6.068 & 44.305 \end{pmatrix}.$$

The eigenvalues of $\widehat{\Sigma}_{xx}$ are $\lambda_1 = 114.039$, $\lambda_2 = 17.5673$, $\lambda_3 = 0.0216273$, and $\lambda_4 \approx \lambda_5 \approx 0$. We wish to describe the data by using just two factors ($k = 2$), so we keep only the two largest eigenvalues λ_1 and λ_2 which are stored in $\Lambda_2 = \mathrm{diag}(\lambda_1, \lambda_2)$. They correspond to the first two columns of V,

$$V_2 = \begin{pmatrix} -0.173 & 0.323 \\ -0.744 & 0.187 \\ -0.211 & 0.101 \\ 0.015 & 0.860 \\ 0.609 & 0.334 \end{pmatrix}.$$

We choose the first variant of determining the factor weights and factor scores. By using (5.76) and (5.77) we get

$$A = V_2 \Lambda_2^{1/2} = \begin{pmatrix} -1.851 & 1.353 \\ -7.949 & 0.783 \\ -2.249 & 0.424 \\ 0.157 & 3.603 \\ 6.506 & 1.401 \end{pmatrix}, \qquad F = X_c A \Lambda_2^{-1} = \begin{pmatrix} -0.716 & -1.731 \\ -0.997 & 1.625 \\ 1.892 & 0.383 \\ -0.835 & -0.066 \\ -0.243 & -1.277 \\ 0.849 & -0.463 \\ -0.652 & 0.830 \\ -0.278 & -0.301 \\ -0.363 & 0.705 \\ 1.342 & 0.296 \end{pmatrix}.$$

By using (5.78) and the matrix $S = \mathrm{diag}(2.293, 7.989, 2.289, 3.606, 6.656)$ we finally obtain the data-factor correlation matrix:

$$C = S^{-1} V_k \Lambda_k^{1/2} = \begin{pmatrix} \underline{-0.807} & 0.590 \\ \underline{-0.995} & 0.098 \\ \underline{-0.983} & 0.185 \\ 0.044 & \underline{0.999} \\ \underline{0.977} & 0.211 \end{pmatrix}. \tag{5.82}$$

The first column of (5.82) establishes a correlation of the first factor to the first, second, third, and fifth component of $\boldsymbol{x}_i$. The second column reveals a strong correlation between the second factor and the fourth data component.

The solution for the matrices F and A^{T} by using (5.73) is obviously not unique, as we may perform the transformations $F' = FT$ and $A' = AT^{-1}$ with any non-singular matrix T and still fulfill the equation. The solutions (5.75) and (5.79) presented here are just the most obvious ones: no transformation is applied to the factors. Still, even though the matrix (5.82) distinctly reveals the basic connections between the data and the factors, we would ideally wish to obtain a matrix with an even simpler structure,

$$C \approx \begin{pmatrix} \underline{-1} & 0 \\ \underline{-1} & 0 \\ \underline{-1} & 0 \\ 0 & \underline{1} \\ \underline{1} & 0 \end{pmatrix},$$

which would pronounce the correlation yet more clearly. This can be achieved by carefully chosen transformations of the factors; the most commonly used in practice are the so-called *varimax rotations*. Details can be found in [60].

5.11.2 Standardization of Data and Robust Factor Analysis

In Sect. 5.11.1 and in the Example on p. 267 we have used the centered data matrix X_{c}. Principal-component factor analysis is not suitable for data with exceedingly different variances, for example, for data with components in different measurement units. (This problem is common to all approaches exploiting the principal-component method; see Sect. 5.7.2.)

If the variances of the data components are very different, the data should be standardized: we should subtract the means and divide out the appropriate standard deviations as in (5.56). This turns the covariance matrix into the correlation matrix. To determine the factors and weights we still follow the procedure of Sect. 5.11.1, but slightly different matrices A, F, and C are obtained. For the data (5.81) from the Example on p. 267 we get

$$C = \begin{pmatrix} \underline{-0.917} & 0.398 \\ \underline{-0.992} & -0.124 \\ \underline{-0.999} & -0.035 \\ -0.178 & \underline{0.984} \\ \underline{0.907} & 0.421 \end{pmatrix},$$

which we may compare to the result (5.82).

The approaches to factor analysis outlined here are, again, very sensitive to outliers. A single runaway data component can strongly modify the covariance matrix and may result in completely different factor and weight matrices F and A, as well as the correlation matrix C. Robust factor analysis which is not hampered by these obstacles is discussed in the papers [61] and [62]; see also Sect. 15.4 in [41] and the often-quoted geo-chemical research work [63].

5.12 Problems

5.12.1 Multiple Regression

The lean body mass (the difference between the mass of the whole body and the mass of the fatty tissue) is an indicator of an individual's predisposition for certain types of disease. It can be measured by underwater weighting [64] but that procedure is cumbersome and expensive. In this Problem (adapted from [41]) we try to establish whether by much simpler measurements of the body mass, the circumference of certain body parts, and by knowing the person's age, multiple regression may be a tool to reach the comparable estimate of the lean body mass.

We collect n measurements of $m = 13$ independent variables x_1 (age), x_2 (body mass), x_3 (body height) and the circumferences x_4 (neck), x_5 (chest), x_6 (abdomen), x_7 (hip), x_8 (thigh), x_9 (knee), x_{10} (ankle), x_{11} (biceps), x_{12} (forearm), x_{13} (wrist). Assume that the dependent variable, the mass of the fatty tissue, can be described by the model

$$y = a_0 + \sum_{j=1}^{m} a_j x_j + \Delta y,$$

which applies for each of the n measurements (see (5.45)) and where Δy is the unknown error. The a_j are the regression parameters to be determined.

⊙ On the website [64] one can find the data for $n = 252$ measurements of $m = 13$ input variables x_j and one output variable y. Use the method of multiple regression described in Sect. 5.6.1 to determine the regression parameters a_j (and establish the corresponding independent variables) that are statistically significant for the determination of the lean body mass.

⊕ If there are fewer measurements than there are independent variables in each individual measurement ($n \ll m$), the data matrix $X_c^T X_c$ becomes non-invertible. In such cases we may apply the principal-component multiple regression (see, for example, the absorption spectrum for the PET yarns in Sect. 5.6.2). Try to reproduce Fig. 5.16. The data can be found at [65].

5.12.2 Nutritional Value of Food

We are used to specifying the nutritional value of individual food items by quoting their protein content (x_1), carbohydrates (x_2), fats (x_3) and saturated fats (x_4), as well as cholesterol (x_5), the total energy value (x_6), and the mass (x_7). Suppose we measure these $m = 7$ variables for each of the n food items, where $n \gg m$. Can we reduce the dimensionality of this set of data by using the principal-component analysis (PCA)?

⊙ The website [66] lists the data for $n = 691$ different food items. The values were measured for items with very different masses x_7, so let us first normalize all

data as

$$x_j \leftarrow x_j / x_7, \quad 1 \le j \le 6.$$

Since the measured values are expressed in different measurement units, you should also standardize the data by using (5.56), and construct the standardized data matrix X_c. Apply the PCA method from Sect. 5.7 in either of its two forms: by diagonalizing the covariance (or the correlation) matrix, or by singular-value decomposition of the data matrix.

Regardless of the way you compute the principal components, show how the percentage of the variance explained by inclusion of r principal-components changes with increasing r. For a few chosen $\boldsymbol{x}_i$, check how well r principal components reproduce the original data point. Plot the scores (5.55) for all n data at the same time, for a few of the leading principal components PC (in particular, PC1 vs. PC2, PC1 vs. PC3, and PC2 vs. PC3). Use these plots to determine whether some food items tend to form clusters and whether there are any outliers.

5.12.3 Discrimination of Radar Signals from Ionospheric Reflections

A system of sixteen antennas in Goose Bay (Labrador) has been used to measure the reflection of radar waves from free electrons in the ionosphere. (Example from [41] based on original data of [67].) Those reflected signals that hinted at a structure in the ionosphere, have been assigned to the class of "good" signals (class R_1). The remaining reflections were labeled as "bad" (class R_2). They have determined the character of $n = 351$ signals, each of which was represented by $m = 34$ independent variables $x_i = (\mathrm{Re}\, x_i, \mathrm{Im}\, x_i)$, $1 \le i \le 17$, corresponding to 17 pairs of complex values (the imaginary part of the first signal is always zero, so the actual number of independent variables is just $m = 33$).

⊙ Figure 5.28 makes it abundantly clear why a simple bivariate (for example, correlation) analysis does not suffice. By comparing the pairs of individual components of the signal alone, the classes cannot be cleanly separated, as "good" reflections are interspersed with the "bad" ones. Perform a linear discriminant analysis of the radar signals from these measurements by using the procedure described in Sect. 5.9. Try to reproduce Fig. 5.26 (right) and compute the misclassification rate.

⊕ Classify the same radar reflection data by the method of logistic discriminant analysis (Sect. 5.9.2). Compute the misclassification rate.

5.12.4 Canonical Correlation Analysis of Objects in the CDFS Area

Chandra Deep Field South (CDFS) is one of the most frequently studied areas in the sky. In a comprehensive study [57, 58] the astronomers have collected astromet-

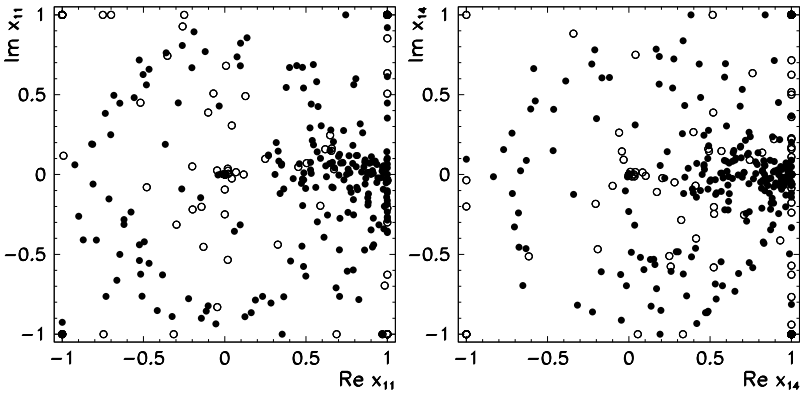


Fig. 5.28 Reflection of radar signals in the ionosphere. Shown are "good" (*symbols* •) and "bad" reflections (*symbols* ○). [*Left*] Variable x_{11}. [*Right*] Variable x_{14}

ric, photometric, and morphological data on 63501 astrophysical objects (for example, galaxies, stars, and quasars). The measurements of the spectro-photometric data have been performed in 17 wavelength bands spanning the range between 350 and 930 nm. The whole COMBO-17 (*Classifying Objects by Medium-Band Observations—A spectro-photometric* 17*-filter survey*) data set is accessible at [59].

This Problem deals only with the data which the researchers believe belongs to galaxies. We have $n = 3462$ multivariate data entries. From the complete set of variables in each measurement we select $p = 23$ variables of the data $\boldsymbol{x}_i$ and $q = 6$ variables of the data $\boldsymbol{y}_i$ (see Sect. 5.10). The first ten elements of $\boldsymbol{x}_i$ are the absolute magnitudes of individual galaxies in ten different spectral bands:

`UjMag,BjMag,VjMag,usMag,gsMag,rsMag,UbMag,BbMag,VbMag,S280Mag.`

The remaining elements are the observed brightnesses in thirteen spectral bands:

`W420F_E,W462F_E,W485F_D,W518F_E,W571F_S,W604F_E,W646F_D,`
`F_E,W753F_E,W815F_S,W856F_D,W914F_D,W914F_E.`

The elements of $\boldsymbol{y}_i$ are

`Rmag,ApD_Rmag,mu_max,MC_z,MC_z_ml,chi2red`

(total R-band magnitude, aperture difference of `Rmag`, central surface brightness in `Rmag`, mean shift in the distribution of the observed sources with respect to the red shift z, the peak of the red shift distribution, and the quality of the fit).

⊙ Use the canonical correlation analysis to compute the pairs of canonical variables (ξ_j, ζ_j) for $1 \le j \le 6$ with the coefficients (5.70) and their scores (5.71) for the data from the COMBO-17 database. Keep only those data from the database in which none of the 23 elements of $\boldsymbol{x}_i$ and none of the six elements of $\boldsymbol{y}_i$ are missing. For each j plot the pairs of scores and compute the corresponding canonical correlation coefficient.

References

1. J.E. Gentle, W. Härdle, Y. Mori (eds.), *Handbook of Computational Statistics. Concepts and Methods* (Springer, Berlin, 2004)
2. V. Barnett, T. Lewis, *Outliers in Statistical Data*, 3rd edn. (Wiley, New York, 1994)
3. R. Kandel, *Our Changing Climate* (McGraw-Hill, New York, 1991), p. 110
4. L. Davies, U. Gather, Robust statistics, in *Handbook of Computational Statistics. Concepts and Methods* (Springer, Berlin, 2004) pp. 655–695
5. Analytical Methods Committee, Robust statistics—how not to reject outliers, part 1: basic concepts. Analyst **114**, 1693 (1989)
6. Analytical Methods Committee, Robust statistics—how not to reject outliers, part 2: inter-laboratory trials. Analyst **114**, 1699 (1989)
7. V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: a survey. ACM Comput. Surv. **41**, 15 (2009)
8. A. Patcha, J.-M. Park, An overview of anomaly detection techniques: existing solutions and latest technological trends. Comput. Netw. **51**, 3448 (2007)
9. M. Agyemang, K. Barker, R. Alhajj, A comprehensive survey of numeric and symbolic outlier mining techniques. Intell. Data Anal. **10**, 521 (2006)
10. V.J. Hodge, J. Austin, A survey of outlier detection methodologies. Artif. Intell. Rev. **22**, 85 (2004)
11. L. Davies, U. Gather, The identification of multiple outliers. J. Am. Stat. Assoc. **88**, 782 (1993)
12. B. Iglewicz, J. Martinez, Outlier detection using robust measures of scale. J. Stat. Comput. Simul. **15**, 285 (1982)
13. F.E. Grubbs, Procedures for detecting outlying observations in samples. Technometrics **11**, 1 (1969)
14. W.J. Dixon, Ratios involving extreme values. Ann. Math. Stat. **22**, 68 (1951)
15. W.J. Dixon, Analysis of extreme values. Ann. Math. Stat. **21**, 488 (1950)
16. R.J. Beckman, R.D. Cook, Outlier.....s. Technometrics **25**, 119 (1983)
17. R.A. Maronna, R.D. Martin, V.J. Yohai, *Robust Statistics. Theory and Methods* (Wiley, Chichester, 2006)
18. M.R. Spiegel, *Schaum's Outline of Theory and Problems of Probability and Statistics* (McGraw-Hill, New York, 1975)
19. S. Brandt, *Data Analysis*, 3rd edn. (Springer, New York, 1999)
20. H.B. Mann, A. Wald, On the choice of the number of class intervals in the application of the chi square test. Ann. Math. Stat. **13**, 306 (1942)
21. W.C.M. Kallenberg, J. Oosterhoff, B.F. Schriever, The number of classes in chi-squared goodness-of-fit tests. J. Am. Stat. Assoc. **80**, 959 (1985), and references therein
22. W.C. Kallenberg, On moderate and large deviations in multinomial distributions. Ann. Stat. **13**, 1554 (1985)
23. M.A. Stephens, Use of the Kolmogorov–Smirnov, Cramer–Von Mises and related statistics without extensive tables. J. R. Stat. Soc. B **32**, 115 (1970)
24. A.F. Nikiforov, S.K. Suslov, V.B. Uvarov, *Classical Orthogonal Polynomials of a Discrete Variable*. Springer Series in Computational Physics (Springer, Berlin, 1991)
25. W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 3rd edn. (Cambridge University Press, Cambridge, 2007). See also the equivalent handbooks in Fortran, Pascal and C, as well as http://www.nr.com
26. C.A. Cantrell, Technical note: Review of methods for linear least-squares fitting of data and application to atmospheric chemistry problems. Atmos. Chem. Phys. **8**, 5477 (2008)
27. D. York et al., Unified equations for the slope, intercept, and standard errors of the best straight line. Am. J. Phys. **72**, 367 (2004)
28. K. Nakamura et al. (Particle Data Group), Review of particle physics. J. Phys. G **37**, 075021 (2010). See Sect. 5 of the Introduction

29. M.C. Ortiz, L.A. Sarabia, A. Herrero, Robust regression techniques. A useful alternative for the detection of outlier data in chemical analysis. Talanta **70**, 499 (2006)
30. J. Ferré, Regression diagnostics, in *Comprehensive Chemometrics: Chemical and Biochemical Data Analysis, Vol. 3*, ed. by S.D. Brown, R. Tauler, B. Walczak (2009), p. 33
31. P.J. Rousseeuw, A.M. Leroy, *Robust Regression and Outlier Detection* (Wiley, Hoboken, 2003)
32. I. Barrodale, F.D.K. Roberts, An improved algorithm for discrete l_1 linear approximation. SIAM J. Numer. Anal. **10**, 839 (1973)
33. S. Portnoy, R. Koenker, The Gaussian hare and the Laplacian tortoise: computability of squared-error versus absolute-error estimators. Stat. Sci. **12**, 279 (1997)
34. P.J. Rousseeuw, Least median of squares regression. J. Am. Stat. Assoc. **79**, 871 (1984)
35. T. Bernholt, Computing the least median of squares estimator in time $\mathcal{O}(n^d)$, in *Lecture Notes in Computer Science,* vol. 3480, ed. by O. Gervasi et al. (Springer, Berlin, 2005), p. 697
36. A. Stromberg, Computing the exact least median of squares estimate and stability diagnostics in multiple linear regression. SIAM J. Sci. Comput. **14**, 1289 (1993)
37. B.W. Rust, Fitting nature's basic functions, part I: polynomials and linear least squares. Comput. Sci. Eng. **Sep/Oct**, 84 (2001)
38. B.W. Rust, Fitting nature's basic functions, part II: estimating uncertainties and testing hypotheses, Comput. Sci. **Nov/Dec**, 60 (2001)
39. B.W. Rust, Fitting nature's basic functions, part III: exponentials, sinusoids, and nonlinear least squares, Comput. Sci. **Jul/Aug**, 72 (2002)
40. B.W. Rust, Fitting nature's basic functions, part IV: the variable projection algorithm, Comput. Sci. **Mar/Apr**, 74 (2003)
41. A.J. Izenman, *Modern Multivariate Statistical Techniques* (Springer, Berlin, 2008)
42. H. Swierenga, A.P. de Weijer, R.J. van Wijk, L.M.C. Buydens, Strategy for constructing robust multivariate calibration models. Chemom. Intell. Lab. Syst. **49**, 1 (1999)
43. I.T. Jolliffe, *Principal Component Analysis*, 2nd edn. (Springer, Berlin, 2002)
44. S. Roweis, Z. Ghahramani, A unifying review of linear Gaussian models. Neural Comput. **11**, 305 (1999)
45. A. Azzalini, A.W. Bowman, A look at some data on the Old Faithful geyser. J. R. Stat. Soc. C **39**, 357 (1990)
46. A.K. Jain, M.N. Murty, Data clustering: a review. ACM Comput. Surv. **31**, 264 (1999)
47. W. Härdle, L. Simar, *Applied Multivariate Statistical Analysis* (Springer, Berlin, 2007)
48. R. Xu, D.C. Wunsch II, *Clustering* (Wiley, Hoboken, 2009)
49. G. Gan, C. Ma, J. Wu, *Data Clustering. Theory, Algorithms, and Applications* (Philadelphia, SIAM, 2007)
50. J. Kogan, *Introduction to Clustering Large and High-Dimensional Data* (Cambridge University Press, Cambridge, 2007)
51. J. Valente de Oliveira, W. Pedrycz (eds.), *Advances in Fuzzy Clustering and Its Applications* (Wiley, Chichester, 2007)
52. The R Project for Statistical Computing. http://www.r-project.org/. Attention: the R reference manual has approximately 3000 pages!
53. J. Maindonald, J. Braun, *Data Analysis and Graphics Using R*, 2nd edn. (Cambridge University Press, Cambridge, 2006). A good introductory text for R, which is an open-source alternative to the S/S+ systems ("R *is to* S *what* OCTAVE *is to* MATLAB")
54. U. von Luxburg, A tutorial on spectral clustering. Technical Report No. Tr-149, Max-Planck-Institut für biologische Kybernetik, 2006
55. A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm. Adv. Neural Inf. Process. Syst. **14**, 849 (2001). See also Ref. [13] in this paper
56. O.L. Mangasarian, W.N. Street, W.H. Wolberg, Breast cancer diagnosis and prognosis via linear programming. Oper. Res. **43**, 570 (1995)
57. C. Wolf et al., A catalogue of the Chandra deep field south with multi-colour classification and photometric redshifts from COMBO-17. Astron. Astrophys. **421**, 913 (2004)

58. C. Wolf et al., Calibration update of the COMBO-17 CDFS catalogue. Astron. Astrophys. **492**, 933 (2008)
59. http://www.mpia.de/COMBO/combo_CDFSpublic.html. The data can be found at http://astrostatistics.psu.edu/datasets/COMBO17.html
60. R.A. Reyment, K.G. Jöreskog, L.F. Marcus, *Applied Factor Analysis in the Natural Sciences* (Cambridge University Press, Cambridge, 1993)
61. G. Pison, P.J. Rousseeuw, P. Filzmoser, C. Croux, Robust factor analysis. J. Multivar. Anal. **84**, 145 (2003)
62. P. Filzmoser, K. Hron, C. Reimann, R. Garrett, Robust factor analysis for compositional data. Comput. Geosci. **35**, 1854 (2009)
63. C. Reimann, P. Filzmoser, R.G. Garrett, Factor analysis applied to regional geochemical data: problems and possibilities. Appl. Geochem. **17**, 185 (2002)
64. http://lib.stat.cmu.edu/datasets/bodyfat, where all data is collected and the corresponding original literature is cited
65. http://astro.temple.edu/~alan/MMST/datasets.html
66. http://www.ntwrks.com/~mikev/chart1.html
67. V.G. Sigillito, S.P. Wing, L.V. Hutton, K.B. Baker, Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Tech. Dig. **10**, 262 (1989). The corresponding data file can be found at http://archive.ics.uci.edu/ml/datasets.html

Chapter 6
Modeling and Analysis of Time Series

A *time series* or *signal* $s(t) \in \Sigma$ should be understood as a sequential measurement of some quantity. The time variable and the corresponding signal may be discrete or continuous. The signals may be real, complex, or integer. In the analysis of time series we use mathematical tools to extract their basic characteristics and learn about the properties of their sources. The sources of signals are also called *processes* and are commonly identified with the signals that these processes generate. We divide the processes according to their statistical properties, as shown in Fig. 6.1. Fundamentally, we distinguish deterministic and random processes.

Deterministic Processes In deterministic processes, each value of the signal is precisely determined by a mathematical or physical law, a rule, or a table of values. If we understand the process dynamics well, such processes are known as *dynamical systems* [1, 2]. The state of a dynamical system in space Ξ can be uniquely described by the mapping of the system from state $x(t_0)$ at time t_0 to state $x(t)$ at time t, i.e.

$$x(t) = \phi\big(t, t_0, x(t_0)\big).$$

The observed signal s is the result of a function $F : \Xi \to \Sigma$ acting on the system evolving in time, $s(t) = F(x(t))$. Signals from deterministic processes are simply reproducible and we know how the signal behaved in the past and what it is going to look like in the future.

Random Processes In random processes the generated signal is not determined in advance by the process parameters; it comes about by chance. Before the measurement it is impossible to precisely foresee its value. The signal originating in a random process is known as a *random signal* and represents *one possible realization* of that process. In spite of the misleading nomenclature, the random signal is not necessarily random in time; just the choice of its realization is random (see Example on p. 283). Details are discussed in Sect. 6.2.

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9_6,

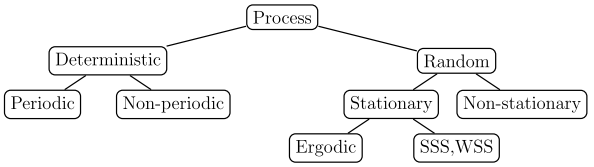


Fig. 6.1 Classification of processes. In deterministic processes the values of the signals are determined by rules or laws. In random processes the signals are formed based on chance. The SSS and WSS notation is defined on p. 282

6.1 Random Variables

A random or stochastic variable is a variable whose value is not determined in advance and which, at each measurement of an observable, assumes a value with a certain probability. The measurement of a quantity is known as the *realization of a random variable* or *drawing*. The values of a random variable are drawn randomly and independently. We denote random variables by Roman capitals, e.g. X, and the corresponding values with small letters, e.g. x. In this section we use the conventional notation of theories of measure [3, 4] and probability [5, 6].

6.1.1 Basic Definitions

A random variable X is characterized by its *sample space* Ω and its *probability measure* $P : A \to [0, 1]$ defining the probability that X is in some subset $A \subset \Omega$, which we write as $P(A) = \text{Prob}(X \in A)$. If the sample space is continuous, we may use the differential volume element $\mathrm{d}V(x)$ at $x \in \Omega$ to write the differential of the probability measure as

$$\mathrm{d}P(x) = p(x)\,\mathrm{d}V(x).$$

We use it to define the probability density p of the variable X. If Ω is countable, $P(x)$ represents the probability that the value x is drawn.

The most important operation on random variables is statistical averaging. We define the statistical average of a function (observable) f of a random variable X according to the type of the sample space Ω:

$$\text{discrete } \Omega: \quad \langle f \rangle = \sum_{x \in \Omega} f(x) P(x), \qquad \text{continuous } \Omega: \quad \langle f \rangle = \int_{\Omega} f(x)\,\mathrm{d}P(x).$$

In the literature we also find the notation $E[f] = \langle f \rangle$, helping us to define the variance or dispersion $\text{var}[f] = \langle (f - E[f])^2 \rangle$, as well as the standard deviation $\sigma[f] = (\text{var}[f])^{1/2}$.

An important quantity for a random variable X with the sample space Ω that is a subset of $\mathbb{R}$ or $\mathbb{Z}$, is the characteristic function

$$\phi_X(t) = \left\langle \mathrm{e}^{\mathrm{i}tX} \right\rangle, \quad \phi_X(0) = 1, \ \left|\phi_X(t)\right| < 1 \ \forall t \neq 0. \tag{6.1}$$

The characteristic function allows us to construct the statistical moments $\langle X^n \rangle$, $n = 0, 1, 2, \ldots$ since if ϕ_X is n-times continuously differentiable at the origin,

$$\lim_{t \to 0} \frac{\mathrm{d}^n}{\mathrm{d}t^n} \phi_X(t) = \mathrm{i}^n \langle X^n \rangle.$$

The characteristic function and the probability density are related by Fourier transformation,

$$p(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \phi_X(t) \mathrm{e}^{-\mathrm{i}tx} \, \mathrm{d}t$$

in the continuous case ($\Omega \subset \mathbb{R}$), or

$$P(x) = \frac{1}{N} \sum_{k=0}^{N-1} \phi_X(k) \mathrm{e}^{-\mathrm{i}(2\pi k/N)x}$$

in the discrete case ($\Omega \subset \mathbb{Z}_N$). The probability density and the characteristic function are equivalent descriptions of statistical properties of a random variable.

6.1.2 Generation of Random Numbers

In numerical calculations of statistical averages and in simulations involving random numbers we use random number generators (Appendix C). These are algorithms that *deterministically* produce numbers with desired probability distributions giving an impression of randomness. Such generators may therefore be seen as sources of realizations of random variables. For each generated sequence of numbers $\{x_i\}$ and arbitrary smooth observable f, a good random number generator should ensure *ergodicity*,

$$\big\langle f(x_t) \big\rangle_t = \lim_{T \to \infty} \big\langle f(x_t) \big\rangle_{t,T} = \langle f \rangle, \qquad \big\langle f(x_t) \big\rangle_{t,T} = \frac{1}{T} \sum_{t'=0}^{T-1} f\big(x_t'\big),$$

that is, the equality of statistical and time averages, as well as the property of *mixing*, for which we require that the quantity

$$\lim_{T \to \infty} \big\langle \big(f(x_t) - \langle f \rangle\big)\big(f(x_{t+j}) - \langle f \rangle\big) \big\rangle_{t,T} \tag{6.2}$$

vanishes when $j \to \infty$. (In Sect. 6.6 we shall see that (6.2) represents the auto-correlation of f at shift j.) Ergodicity and mixing are concepts from the theory of dynamical systems [7], where the latter implies the former, but not the other way around. Mixing is equivalent to the requirement that the drawn numbers are as independent as possible. For smooth enough f, the exponential convergence of the auto-correlation (6.2) ensures the statistically fastest possible convergence of the average $\langle f(x_t) \rangle_{t,T}$ to the limit $\langle f \rangle$, so $\langle f(x_t) \rangle_{t,T} - \langle f \rangle = \mathcal{O}(T^{-1/2})$. Fast mixing is a desirable (but not essential) property of random number generators.

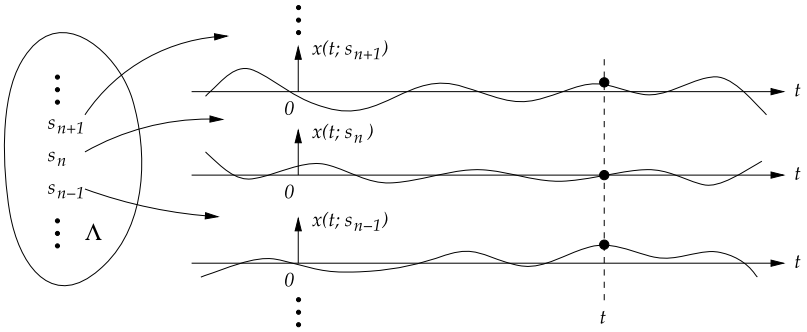


Fig. 6.2 A random process $X(t)$ and its three possible realizations

6.2 Random Processes

Random processes are omnipresent in physics. Quantum mechanics and statistical physics, for example, are founded on assumptions about the randomness of Nature at certain distance or energy scales [8]. In the following four sections we present the fundamentals of the theory of random processes and describe three typical representatives: random walks, Markov chains, and noise.

6.2.1 Basic Definitions

The random process is a generalization of the concept of random variables where, instead of a value, we randomly choose a time signal which we name the *realization of a random process* or the *sample path*. Random processes can be defined in two ways, illustrated in Fig. 6.2.

In the first approach we imagine a random variable S with values s distributed over the set Λ. We define an ensemble of functions $\{x(t; s) \in \Omega\}_{s\in\Lambda}$ with sample space Ω, where the time t runs over the set $\mathbb{T}$. The random process $X(t)$ represents a random function from the ensemble which we select by drawing the value s. At a given particular t we can understand $X(t)$ as a random variable $X(t) = x(t; S)$. In other words, if we make a vertical cut through Fig. 6.2 at a chosen t, the obtained values behave as a random variable. On the other hand, we can interpret the random process as drawing a set of random variables $\{X(t) \in \Omega\}_{t\in\mathbb{T}}$ which may be correlated.

If $\mathbb{T} = \mathbb{R}$, we are referring to continuous-time processes, while with $\mathbb{T} = \mathbb{N}$ or $\mathbb{Z}$ we are dealing with discrete-time processes. In analogy to random variables, random processes are also denoted by capital Roman letters, e.g. X, while we use small ones, like x, for their realizations. Sample spaces of random processes are commonly labeled by capital Greek letters, e.g. Ω. The statistical average over the realizations of the random process is denoted by $\langle\cdot\rangle$ or $E[\cdot]$.

Random processes are almost never defined by the probability distributions of the elements of their signals; rather, we use the time dependence of the statistical moments of the signals, for example, the average $\mu_X(t) = \langle X(t)\rangle$, the two-point autocorrelation $R_{XX}(t_1, t_2) = \langle X(t_1)X(t_2)\rangle$, and so on. A rare exception is the discrete-time real Gaussian random process. For n points of the signal at times $t_1, t_2, \ldots, t_n$ it can be described by the probability density

$$p_{X(t_1),X(t_2),\ldots,X(t_n)}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n \det(R)}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}} R^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right],$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$. The probability density $p_{X(t_1),X(t_2),\ldots,X(t_n)}(\boldsymbol{x})$ is defined by the vector of averages $\boldsymbol{\mu} = (\mu_X(t_1), \mu_X(t_2), \ldots, \mu_X(t_n))^{\mathrm{T}}$ and the autocorrelation matrix $R = [R_{XX}(t_i, t_j)]_{ij}$, which we will get to know in an instant. Most often we encounter Gaussian processes with the property $R_{XX}(t_1, t_2) = R_{XX}(t_1 - t_2)$.

Non-stationarity and Stationarity Random processes are distinguished by their statistical properties (we follow [9]). The process may be stationary or non-stationary. Roughly speaking, the statistics of stationary processes do not change with time; a more precise definition of stationarity of a random process X with sample space Ω requires us to use the joint probability measure for n points of the process at times $t_1, t_2, \ldots, t_n$,

$$P_{X(t_1),X(t_2),\ldots,X(t_n)}(A) = \mathrm{Prob}\big(X(t_1), X(t_2), \ldots, X(t_n) \in A\big),$$

where $A \subset \Omega^n$. A random process X is *stationary of order n* if this measure for arbitrary n points is invariant with respect to translations along the time axis,

$$P_{X(t_1),X(t_2),\ldots,X(t_n)} = P_{X(t_1+t),X(t_2+t),\ldots,X(t_n+t)} \quad \forall t.$$

Note that a process that is stationary of order $n+1$ is also stationary of order n, while the reverse is not necessarily true. In the following we assume that the sample space of the random process is $\Omega \subset \mathbb{R}$. In the first-order stationary process the measure $P_{X(t)}$ is constant, and therefore also all single-point averages are constant, like the moments: $\langle X(t)^m\rangle = \text{const.}$ for $m = 1, 2, \ldots$. For the second-order stationary process, the two-point measure $P_{X(t_1),X(t_2)}$ is translationally invariant. It follows that all two-point averages depend only on time *differences*. One such quantity is the two-point correlation of the observables f and g of the random process X:

$$\big\langle f\big(X(t_1)\big)g\big(X(t_2)\big)\big\rangle = \big\langle f\big(X(0)\big)g\big(X(t_2-t_1)\big)\big\rangle = \big\langle f\big(X(t_1-t_2)\big)g\big(X(0)\big)\big\rangle.$$

A very interesting average is the auto-correlation

$$R_{XX}(t_1, t_2) = \big\langle X(t_1)X(t_2)\big\rangle = R_{XX}(t_1 - t_2),$$

for which

$$\big|R_{XX}(\tau)\big| \le R_{XX}(0) = \big\langle X(t)^2\big\rangle, \quad R_{XX}(-\tau) = R_{XX}(\tau). \tag{6.3}$$

If the process is stationary of arbitrary order, it is said to be *strict-sense stationary* (SSS). In contrast, a random process X is *wide-sense stationary* (WSS) if the following holds: that the average is constant, $\langle X(t)\rangle = \text{const.}$; that the auto-correlation depends on time difference only, thus $R_{XX}(t_1, t_2) = R_{XX}(t_1 - t_2)$; and that the variance is bounded for all times, $\langle (X(t) - \langle X(t)\rangle)^2\rangle < \infty$. Every second-order stationary process in wide-sense stationary, while the reverse is not necessarily true.

Ergodicity Define the time average of the function f in the continuous case,

$$\big\langle f(t)\big\rangle_t = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} f\big(t'\big)\,\mathrm{d}t', \tag{6.4}$$

where $\mathbb{T} = \mathbb{R}$, or in the discrete case,

$$\big\langle f(t)\big\rangle_t = \lim_{T\to\infty} \frac{1}{2T+1} \sum_{t'=-T}^{T} f\big(t'\big), \tag{6.5}$$

where $\mathbb{T} = \mathbb{Z}$ (be alert to the notation $\langle\cdot\rangle_t$). Assume that almost every realization x of the process X fulfills the conditions that the time average is equal to the statistical average,

$$\big\langle x(t)\big\rangle_t = \big\langle X(t)\big\rangle,$$

and that the time auto-correlation is equal to the statistical one, i.e.

$$\big\langle x(t)x(t+\tau)\big\rangle_t = R_{XX}(\tau).$$

Then we say that the random process X is *ergodic*. We commonly refer to the first condition as *ergodicity on average*, while the second condition is known as *ergodicity in auto-correlation*. The first condition is fulfilled precisely when the average of the auto-covariance,

$$C_{XX}(\tau) = \big\langle \big(X(t) - \mu_X\big)\big(X(t+\tau) - \mu_X\big)\big\rangle, \quad \mu_X = \big\langle X(t)\big\rangle,$$

is equal to zero, so $\langle C_{XX}(t)\rangle_t = 0$. The sufficient condition for ergodicity on average is $\lim_{t\to\infty} C_{XX}(t) = 0$ [10].

In practice we are dealing only with *realizations* of random processes. To compute the statistical properties of processes from which these realizations originate, we commonly assume ergodicity and wide-sense stationarity: this allows us to use the signal to make inferences about the underlying process.

Assume that in a discrete-time random process $X(t)$, $t \in \mathbb{Z}$, we obtained a finite sample $\{x(t)\}_{t=0}^{N-1}$ from its infinitely long realization $x(t)$. If the process is ergodic, the average and the auto-correlation can be approximated by the sums

$$\big\langle X(t)\big\rangle \approx \frac{1}{T} \sum_{t'=0}^{N-1} x\big(t'\big), \qquad R_{XX}(\tau) \approx \frac{1}{N-\tau} \sum_{t'=0}^{N-1-\tau} x\big(t'\big)x\big(t'+\tau\big),$$

where the symmetry $R_{XX}(-\tau) = R_{XX}(\tau)$ has been exploited (see (6.3)).

Example Imagine a random process represented by the sum

$$X(t) = \sum_{i=1}^{n} w_i \cos(\omega_i t + Y_i), \tag{6.6}$$

where the weights w_i and the frequencies ω_i are constant, and the independent random variables Y_i are distributed normally on the interval $[0, 2\pi)$. We immediately realize that $\langle X(t)\rangle = 0$, while the auto-correlation is

$$R_{XX}(t_1, t_2) = \big\langle X(t_1)X(t_2)\big\rangle = \frac{1}{2}\sum_{i=1}^{n} w_i^2 \cos\big(\omega_i(t_1 - t_2)\big) = R_{XX}(t_1 - t_2).$$

The process X is therefore wide-sense stationary. It is easy to show that the time average of an individual *realization* is also zero, $\langle x(t)\rangle_t = 0$, while its time auto-correlation is equal to the statistical one, $\langle x(t)x(t+\tau)\rangle_t = R_{XX}(\tau)$. We conclude that the random process X is also ergodic.

Equation (6.6) also elucidates the initial definitions of random processes and their realizations (p. 277), claiming that "*the individual random signal is not necessarily random along the time axis; what is random is just the choice of its realization*". Namely, the dependence of the signal originating in the process (6.6) on the time variable t is known explicitly! It is by drawing the variable Y_i that randomness or unpredictability is built into the process.

6.3 Stable Distributions and Random Walks

6.3.1 Central Limit Theorem

Let $X_1, X_2, \ldots, X_n$ be real independent and identically distributed random variables with the probability density p_X, whose average $\mu_X = \langle X_i\rangle$ and variance $\sigma_X^2 = \langle (X_i - \mu_X)^2\rangle$ are bounded. Define the sum of random variables $Y_n = \sum_{i=1}^{n} X_i$ with the average $\langle Y_n\rangle = n\mu_X$ and variance $\sigma_{Y_n}^2 = n\sigma_X^2$. The probability density p_Y of the summed variable Y_n is given by the n-fold convolution of the densities of the variables X_i, i.e.

$$p_{Y_n} = \underbrace{p_X * p_X * \cdots * p_X}_{n}.$$

Let us define the rescaled variable $Z_n = (Y_n - n\mu_X)/(\sqrt{n}\sigma_X)$. Then, in the limit $n \to \infty$, the cumulative distribution function of the variable Z_n converges to the cumulative distribution function of the normal distribution $N(0, 1)$,

$$\lim_{n\to\infty} \mathrm{Prob}(Z_n < z) = \Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \mathrm{e}^{-\frac{1}{2}t^2}\, \mathrm{d}t.$$

In other words, the probability density p_{Y_n} in the limit $n \to \infty$ converges to the normal (Gaussian) probability density $N(\mu_{Y_n}, \sigma^2_{Y_n})$, the statement known as the *central limit theorem* (CLT). The speed of convergence to the normal distribution is the substance of the Berry–Esséen theorem [6]. If the third moment of $|X - \mu_X|$ is bounded, i.e. $\rho = \langle |X - \mu_X|^3 \rangle < \infty$, we have

$$\left|\text{Prob}(Z_n < z) - \Phi(z)\right| \leq \frac{C\rho}{\sqrt{n}\sigma_X^3},$$

where $C \geq 0.4097$ [11]. The central limit theorem and the mathematical form of the estimate remain valid even when we sum the variables X_i distributed according to different probability distributions, but only if the dispersion of the values is not excessive (Lindeberg criterion, see [6]).

6.3.2 Stable Distributions

The Gaussian distribution as the limit distribution in summation of independent random variables can be generalized by introducing the concept of *stable distributions* [12–14]. Assume that we have independent random variables X_1, X_2, and X_3 with the same distribution with respect to the sample space Ω. We say that this distribution is *stable* if for each pair of numbers a and b, a pair c and d exists such that the distribution of the linear combination $aX_1 + bX_2$ is the same as the distribution of $cX_3 + d$, i.e. if we have

$$\text{Prob}(aX_1 + bX_2 \in A) = \text{Prob}(cX_3 + d \in A) \quad \forall A \subset \Omega.$$

Such random numbers are also called stable; a superposition of stable random numbers is a linear function of the stable random number with the same distribution.

In general, stable distributions are most easily described by their characteristic functions. Among the many possible notations we follow that of [12]. We say that a random variable X has a stable distribution $p_{\text{stab}}(x; \alpha, \beta, \gamma, \delta)$ if the logarithm of its characteristic function (6.1) has the form

$$\log \phi_X(t) = \mathrm{i}\delta t - \gamma^\alpha |t|^\alpha \left[1 - \mathrm{i}\beta \Phi_\alpha(t)\right],$$

where

$$\Phi_\alpha(t) = \begin{cases} \text{sign}(t) \tan(\pi\alpha/2); & \alpha \neq 1, \\ -\frac{2}{\pi}\text{sign}(t) \log|t|; & \alpha = 1. \end{cases}$$

The parameter $\alpha \in (0, 2]$ is the *stability index* or the *characteristic exponent*, while the parameter $\beta \in [-1, 1]$ defines the *skewness* of the distribution. There is also the scaling parameter $\gamma > 0$ and the position parameter $\delta \in \mathbb{R}$. For $\alpha \in (1, 2]$ the average exists and is equal to $\langle X \rangle = \delta$. For general $\alpha \in (0, 2]$ the statistical moments $\langle |X|^p \rangle$ exist, where $p \in [0, \alpha)$.

For practical computations it is useful to replace the random variable X by another random variable Z,

$$X = \begin{cases} \gamma Z + \delta; & \alpha \neq 1, \\ \gamma(Z + \frac{2}{\pi}\beta \log \gamma) + \delta; & \alpha = 1, \end{cases}$$

because the characteristic function for Z is slightly simpler,

$$\log \phi_Z(t) = -|t|^\alpha \big[1 - \mathrm{i}\beta \Phi_\alpha(t)\big],$$

as it depends only on two parameters, α and β. The probability density p_Z of the variable Z is computed by inverse Fourier transformation of the characteristic function ϕ_Z. We obtain

$$p_Z(z; \alpha, \beta) = \frac{1}{\pi} \int_0^\infty \exp\big(-t^\alpha\big) \cos\big(zt - t^\alpha \beta \Phi_\alpha(t)\big)\, \mathrm{d}t.$$

We have $p_Z(-z; \alpha, \beta) = p_Z(z; \alpha, -\beta)$. The values of the functions p_Z and p_X can be computed by using integrators specially adapted to rapidly oscillating functions (see Appendix E.2; a modest programming support for stable distributions can also be found in [15]). With respect to α and β, the function p_Z with the argument z has the definition ranges

$$z \in \begin{cases} (-\infty, 0]; & \alpha < 1,\ \beta = -1, \\ [0, \infty); & \alpha < 1,\ \beta = 1, \\ \mathbb{R}; & \text{otherwise.} \end{cases}$$

The dependence of the stable distribution p_{stab} (p_X or p_Z with appropriate scaling) on the parameter α is shown in Fig. 6.3 (top left and right), while the dependence on β is shown in the same figure at bottom left and right.

A formulation as flexible as this allows us to generate all possible stable distributions. The most familiar ones are

$$\text{normal:} \quad \alpha = 2,\ \beta \in \mathbb{R}, \quad p_X(t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}t^2\right), \quad t \in \mathbb{R};$$

$$\text{Cauchy:} \quad \alpha = 1,\ \beta = 0, \quad p_X(t) = \frac{1}{\pi}\frac{1}{1+t^2}, \quad t \in \mathbb{R};$$

$$\text{Lévy:} \quad \alpha = \tfrac{1}{2},\ \beta = 1, \quad p_X(t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2t}\right) t^{-\frac{3}{2}}, \quad t \in \mathbb{R}_+.$$

A well-known property of stable distributions with $\alpha \in (0, 2)$ is the characteristic asymptotic behavior of their probability densities, known as *power* or *fat tails*. For

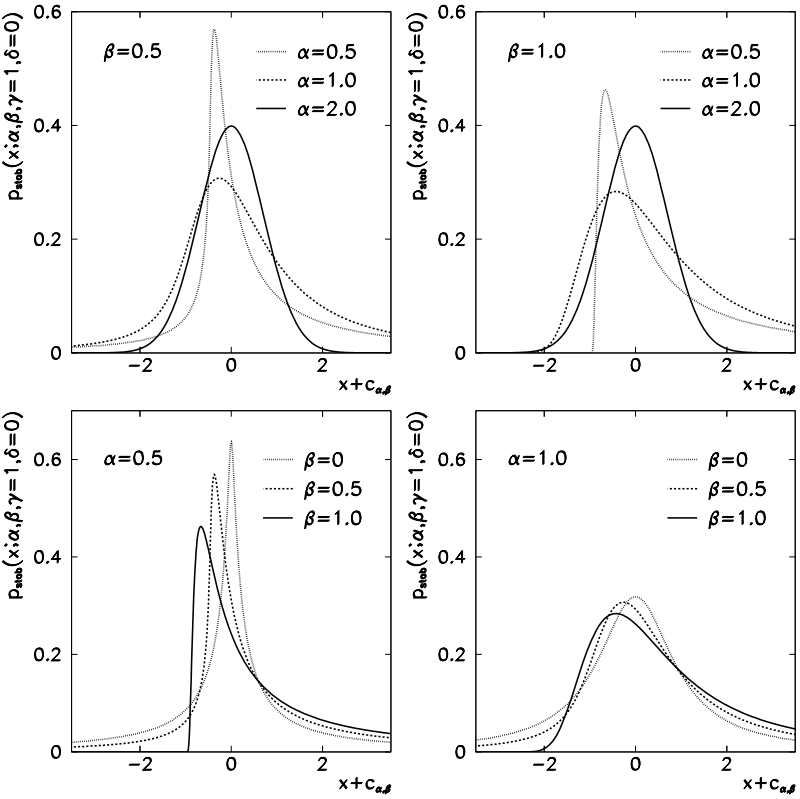


Fig. 6.3 Stable distributions $p_{\text{stab}}(x;\alpha,\beta,\gamma,\delta)$. [*Top left and right*] Dependence on α for $\beta = 0.5$ and 1.0. [*Bottom left and right*] Dependence on β for $\alpha = 0.5$ and 1.0. At $\alpha \neq 1$ the independent variable is shifted by $c_{\alpha,\beta} = \beta\tan(\pi\alpha/2)$

the cumulative probabilities the following holds:

$$\begin{aligned}
\beta \in (-1,1]: &\quad \int_x^\infty p_Z(z;\alpha,\beta)\,\mathrm{d}z \sim \frac{1}{2}c_\alpha(1+\beta)x^{-\alpha}, \quad x \to \infty, \\
\beta \in [-1,1): &\quad \int_{-\infty}^x p_Z(z;\alpha,\beta)\,\mathrm{d}z \sim \frac{1}{2}c_\alpha(1-\beta)(-x)^{-\alpha}, \quad x \to -\infty,
\end{aligned} \tag{6.7}$$

where $c_\alpha = 2\sin(\pi\alpha/2)\Gamma(\alpha)/\pi$. For $\beta \in (-1,1)$ such asymptotic behavior can be established in both limits, $x \to \pm\infty$. Recall that a probability density has the asymptotics $\mathcal{O}(|x|^{-\alpha-1})$ if the corresponding cumulative probability has the asymptotics $\mathcal{O}(|x|^{-\alpha})$.

6.3.3 Generalized Central Limit Theorem

By using our knowledge of the stable distributions (Sect. 6.3.2) we can write a generalized central limit theorem (or Lévy's limit theorem) elaborated in detail by [6, 14]. Here we attempt to convey only its essence.

Assume that we have a sequence of independent and identically distributed random numbers $\{X_i\}_{i\in\mathbb{N}}$ from which we form the partial sum $Y_n = \sum_{i=1}^{n} X_i$. Assume that their distribution has power tails, so that for $\alpha \in (0, 2]$ the limits

$$\lim_{x\to\pm\infty} |x|^{\alpha} \,\mathrm{Prob}(\pm X > x) = d_{\pm}$$

exist and $d = d_+ + d_- > 0$. Then real coefficients $a_n > 0$ and b_n exist such that the rescaled partial sum $Z_n = (Y_n - nb_n)/a_n$ in the limit $n \to \infty$ is stable and distributed according to $p_{\mathrm{stab}}(x; \alpha, \beta, 1, 0)$. The skewness of the stable distribution is $\beta = (d_+ - d_-)/(d_+ + d_-)$, and the coefficients a_n and b_n are [12, 16]

$$a_n = \begin{cases} (dn/c_\alpha)^{1/\alpha}; & \alpha \in (0, 2), \\ \sqrt{(dn\log n)/2}; & \alpha = 2, \end{cases}$$

$$b_n = \begin{cases} \langle X_i \rangle; & \alpha \in (1, 2], \\ \langle X_i \theta(|X_i| - a_n) \rangle; & \text{otherwise}, \end{cases}$$

where θ is the Heaviside (step) function. The constant c_α is defined in (6.7). The coefficient a_n for $\alpha < 2$ diverges as $\mathcal{O}(n^{1/\alpha})$ when n increases.

The generalized CLT is applicable to random-walk processes (discussed in the following), which are analogous to extending the partial sums of random numbers Y_n. From practical experience we know that the convergence to the stable distribution at $n \to \infty$ becomes ever slower and "fussier" when α decreases.

6.3.4 Discrete-Time Random Walks

Random walks are non-stationary random processes used in modeling of numerous physical and engineering phenomena. In this subsection, we present discrete-time random walks [6, 17, 18], while continuous-time random walks [17–20] are discussed in the next. Note that the meaning of α and β in the following is different from that in Sects. 6.3.2 and 6.3.3.

Imagine a discrete-time random process X that we observe as a sequence of random variables $\{X(t)\}_{t\in\mathbb{N}}$. The partial sums of this sequence are

$$Y(t) = Y(0) + \sum_{i=1}^{t} X(i) = Y(t-1) + X(t). \tag{6.8}$$

They represent a new discrete-time random process Y and a sequence of random variables $\{Y(t)\}_{n\in\mathbb{N}_0}$. The process Y is a *random walk*, a single step of which is the process $X(t)$. Let the sample space Ω of the processes X and Y be continuous. We are interested in the time evolution of the probability density $p_{Y(t)}$ of the random variable Y with a known initial density $p_{Y(0)}$.

If we assume that Y is a process in which the state of each point depends only on the state of the previous point, the time evolution of $p_{Y(t)}$ is given by

$$p_{Y(t)}(y) = \int_\Omega p\big(Y(t) = y \big| Y(t-1) = x\big) p_{Y(t-1)}(x)\,\mathrm{d}x,$$

where $p(Y(t) = y|Y(t-1) = x)$ is the conditional probability density that Y evolves from the value x at time $t-1$ to the value y at time t. We also assume that the process X is completely independent of its previous states, so that $p(X(t) = x|Y(t-1) = y-x) = p_{X(t)}(x)$ holds. It then follows that

$$p_{Y(t)}(y) = \int_\Omega p_{X(t)}(x) p_{Y(t-1)}(y-x)\,\mathrm{d}x = (p_{X(t)} * p_{Y(t-1)})(y),$$

where $*$ denotes the convolution. By using this formula, we can express $p_{Y(t)}$ as a convolution of the initial distribution $p_{Y(0)}$ with the distribution of the sum of the steps until t, p_X^{*t}:

$$p_{Y(t)} = p_{Y(0)} * p_X^{*t}, \quad p_X^{*t} = p_{X(1)} * p_{X(2)} * \cdots * p_{X(t)}. \tag{6.9}$$

The evolution of $p_{Y(t)}(y)$ is best computed in Fourier space, as it is given by the product of the Fourier transforms $\mathcal{F}$ of the probability densities,

$$\mathcal{F}[p_{Y(t)}] = \mathcal{F}[p_{Y(0)}] \prod_{i=1}^{t} \mathcal{F}[p_{X(i)}].$$

One frequently assumes that the value of the process Y at time zero is known and that $p_{Y(0)}(y) = \delta(y)$. This assumption is particularly useful when we are interested in the qualitative behavior of $p_{Y(t)}$ after long times.

Asymptotics The time asymptotics of the distributions $p_{Y(t)}$ is most easily established in the case of one-dimensional real random walks. Assume that all steps have the same distribution with the probability density $p_{X(t)} = p_X$, and that $Y(0) = 0$. The distribution that corresponds to the process Y is given by

$$p_{Y(t)} = \mathcal{F}^{-1}\big[\big(\mathcal{F}[p_X]\big)^t\big]$$

for all times t. The behavior of $p_{Y(t)}$ in the limit $t \to \infty$ is determined by the central limit theorem (Sect. 6.3.1) and its generalization (Sect. 6.3.3). These theorems tell us that for increasing t, $p_{Y(t)}$ converges to the limit distribution which is expressible by one of the stable distributions p_{stab}, such that

$$p_{Y(t)}(y) \sim L(t)\, p_{\mathrm{stab}}\big(L(t)y + t\mu(t)\big)$$

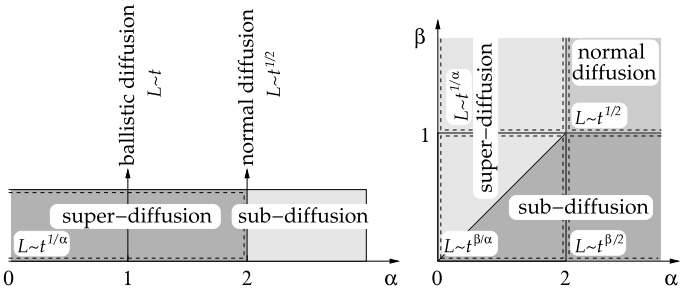


Fig. 6.4 The dependence of the characteristic spatial scale L on time t. [*Left*] Discrete-time random walk. [*Right*] Continuous-time random walk

for suitably chosen functions L and μ. The function L represents the effective width of the central part of the distribution $p_{Y(t)}$, where the majority of the probability is located, and is known as the *characteristic spatial scale*. The function μ has the role of the distribution average.

For the distribution of the steps p_X with a bounded variance ($\sigma_X^2 < \infty$) we also know, based on the CLT, that $p_{Y(t)}$ tends to the Gaussian with the width $L = \sigma_{Y(t)} \sim t^{1/2}$. This particular asymptotic dependence of the spatial scale on time is typical for *normal diffusion* and the same nomenclature pertains to the corresponding regime of the random walk (Fig. 6.4 (left)).

If the distribution p_X has the asymptotic behavior

$$p_X(x) \sim \frac{C_\pm}{|x|^{\alpha+1}}, \quad x \to \pm\infty,$$

where $C_\pm$ are constants, the distribution is said to have a *power* or *fat tail* (see Sect. 6.3.2). For $\alpha \in (0, 2)$ the second moment of the distribution no longer exists and $p_{Y(t)}$ at long times tends to the distribution with $L \sim t^{1/\alpha}$. Because in this case the characteristic scale changes faster than in normal diffusion, we are referring to *super-diffusion*. The dynamics of the process Y in this regime are known as *Lévy flights*. The diffusion with $\alpha = 1$ is said to be *ballistic*: particles propagate without constraints with given velocities, so $L \sim t$. Near $\alpha = 2$ we have $L(t) \sim (n \log n)^{1/2}$ and name the corresponding regime *log-normal diffusion*.

Properties of one-dimensional random walks described above are easily generalized to more dimensions. We observe the projection of a multi-dimensional walk $\hat{n}^{\mathrm{T}} Y(t)$ along the unit vector $\hat{n}$, and its probability density $p_{\hat{n}^{\mathrm{T}} Y(t)}$. For an individual $\hat{n}$ we apply the CLT or its generalization and determine the scale $L_{\hat{n}}$. In such a random walk, a particular direction $\hat{n}^*$ exists along which the scale is maximum or increases most rapidly with t: the characteristic scale of the distribution is then $L_{\hat{n}^*}$. Examples of two-dimensional random walks where the distributions in x and y directions are independent, are shown in Fig. 6.5.

If the densities $p_{X(t)}$ have power tails, the $p_{Y(t)}$ also has power tails. This applies regardless of the CLT or its generalization. Assume that in the limit $t \to \infty$ we have $p_{X(t)}(x) \sim C_{\pm,t} |x|^{-\alpha-1}$. During the process of forming the distribution $p_{Y(t)}$,

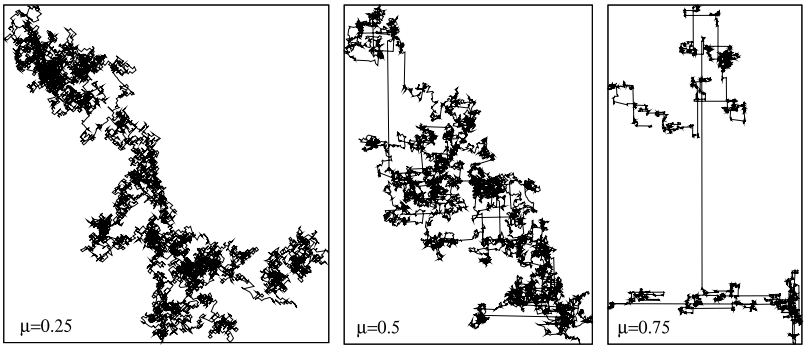


Fig. 6.5 Examples of random walks (x_t, y_t) with 10^4 steps, generated by the formulas $x_{t+1} = x_t + \text{sign}(X)|X|^{-\mu}$ and $y_{t+1} = y_t + \text{sign}(Y)|Y|^{-\mu}$, where X and Y are independent random variables distributed uniformly on the interval $[-1, 1]$

the amplitudes of the tails add up, so that $p_{Y(t)}(x) \sim (\sum_{i=1}^{t} C_{\pm,i})|x|^{-\alpha-1}$ in the limit $x \to \pm\infty$. This means that with increasing time, chances will increase that an observation of $Y(t)$ will yield an extreme event, since

$$\text{Prob}\big(|Y(t)| > y\big) \sim \sum_{i=1}^{t} \text{Prob}\big(|X(i)| > y\big), \quad y \to \infty.$$

The dispersion of the values generated in such processes can be estimated by robust methods (Sect. 5.2). For example, in sub-diffusive random walks we prefer to compute the MAD (5.9) instead of the standard deviation $\sigma_{Y(t)}$.

6.3.5 Continuous-Time Random Walks

In continuous-time random walks [17–20] the number of steps $N(t)$ performed until time t becomes a random variable. We reformulate the definition for the discrete-time random walk (6.8) as

$$Y(t) = Y(0) + \sum_{i=1}^{N(t)} X(i).$$

Obviously $Y(t)$ cannot be written in the iterative form $Y(t) = Y(t-1) + \cdots$ as in (6.8). The number of steps $N(t)$ has the probability distribution $P_{N(t)}$. Assume that $N(t)$ and $X(t)$ are independent processes, which is not always true, as in a given time it is not possible to perform arbitrarily many steps [21, 22]. If $X(i)$ at different times are independent and correspond to probability densities $p_{X(i)}$, the probability

density of the random variable $Y(t)$ is

$$p_{Y(t)}(y) = \sum_{n=0}^{\infty} P_{N(t)}(n)\big(p_{Y(0)} * p_X^{*n}\big)(y),$$

where p_X^{*n} is defined in (6.9).

We adopt the manner of interpretation of such a random walk and the choice of the distribution $P_{N(t)}$ from [19]. A random walk can be envisioned as a sequence of steps with randomly chosen lengths $X(i)$ and *waiting times* $T(i)$ between the steps. After N steps from the departure at the origin, the walk has brought us to the point $\mathcal{X}(N)$. Until then, time $\mathcal{T}(N)$ has elapsed, so that

$$\mathcal{X}(N) = \sum_{i=1}^{N} X(i), \qquad \mathcal{T}(N) = \sum_{i=1}^{N} T(i), \quad \mathcal{X}(0) = \mathcal{T}(0) = 0.$$

Within the allotted time, a certain point can be reached in various numbers of steps N. If the step lengths $X(i)$ and waiting times $T(i)$ are independent, the number of steps $N(t)$ until time t is determined by the process of drawing the waiting times. We introduce the probability that the ith step does not occur before time t,

$$P_{T(i)}(t) = \int_t^{\infty} p_{T(i)}(t')\,\mathrm{d}t',$$

where $p_{T(i)}$ is the probability density of the waiting times. The probability that n steps are performed in a time frame $[0, t]$ is then

$$P_{N(t)}(n) = \int_0^t p_T^{*n}(t')\,P_{T(n+1)}(t - t')\,\mathrm{d}t' = \big(p_T^{*n} * P_{T(n+1)}\big)(t),$$

where $p_T^{*n} = p_{T(1)} * p_{T(2)} * \cdots * p_{T(n)}$.

We compute the probability density $P_{N(t)}$ by using the Laplace transformation in the time variable and the Fourier transformation in the spatial variable. This allows us to work comfortably with function products in transform spaces instead of with convolutions. The procedure leads to the Montroll–Weiss equation (see [19] for details). This equation allows us to identify four regions of parameters defining the distributions p_X and p_T with different dependencies of the scale L on time t, which in turn determine the diffusion properties of the random walk. These regions are specified in the following and are shown in Fig. 6.4 (right). We assume that the distributions of steps and waiting times do not change during the walk, so that $p_{X(i)} = p_X$ and $p_{T(i)} = p_T$.

Normal Diffusion with the scale $L \sim t^{1/2}$ occurs when $\langle T\rangle < \infty$, $\sigma_X < \infty$.

Sub-Diffusion with the scale $L \sim t^{\beta/2}$ occurs with $\langle T\rangle = \infty$, $\sigma_X < \infty$, and the distribution of waiting times

$$p_T(t) \sim \frac{1}{t^{1+\beta}}, \quad \beta \in (0, 1).$$

Super-Diffusion with the scale $L \sim t^{1/\alpha}$ occurs for $\langle T \rangle < \infty$, $\sigma_X = \infty$, and the distribution of the steps

$$p_X(t) \sim \frac{1}{t^{1+\alpha}}, \quad \alpha \in (0, 2).$$

When $\langle T \rangle = \infty$ and $\sigma_X = \infty$, and

$$p_X(t) \sim \frac{1}{t^{1+\alpha}}, \qquad p_T(t) \sim \frac{1}{t^{1+\beta}}, \quad \alpha \in (0, 2),\ \beta \in (0, 1),$$

applies, the scale is $L \sim t^{\beta/\alpha}$. The walks are super-diffusive if $2\beta > \alpha$, and sub-diffusive otherwise. Processes with $\langle T \rangle = \infty$ are deeply non-Markovian: this means that the values of the process at some time depend on its complete history, not solely on the state just before that time. Further reading can be found in [18, 20].

6.4 Markov Chains ⋆

In seeking the solutions of the dynamics of complex systems one is often forced to resort to probabilistic description. The true dynamics is simplified to jumping between states with probabilities that, at each jump, depend only on the initial state (before the jump) and the final state (immediately afterwards). In this section we discuss one type of such random processes, the Markov chain [5, 23, 24].

6.4.1 Discrete-Time or Classical Markov Chains

Think of a random process X in the countable sample space Ω and discrete time $t = 0, 1, \ldots$. Define the conditional probability for a transition from the state x at time t to the state y at time $t + 1$,

$$\text{Prob}\big(X(t+1) = y \big| X(t) = x\big) = \big[P(t)\big]_{x,y}.$$

We arrange these probabilities in a *Markov* or *stochastic matrix* $P(t) = [P(t)]_{x,y}$, where $x, y \in \Omega$. Conservation of probability requires $\sum_{y\in\Omega}[P(t)]_{x,y} = 1$. A random process X is called a *Markov chain* with the initial probability distribution μ in Ω and the transition matrix $P(t)$, if the following conditions are fulfilled:

$$\text{Prob}\big(X(0) = x_0\big) = \mu(x_0)$$

and

$$\text{Prob}\big(X(t+1) = x_{t+1} \big| X(0) = x_0, \ldots, X(t) = x_t\big) = \big[P(t)\big]_{x_t, x_{t+1}}.$$

The second condition says that the probability for the present state to occur depends only on the state immediately preceding it. In other words, the probability to reach the point x_{t+1} at time $t+1$ depends on none of the previous points except x_t. We say that such a Markov chains describes a random process *without memory*. For a system *with memory* consisting of m steps we could imagine a larger sample space Ω^m and define the random process

$$Y(t) = \big(X(t), X(t-1), X(t-2), \ldots, X(t-m+1)\big) \in \Omega^m,$$

which again is a Markov chain. If the Markov matrix P does not depend on time, we are referring to a time-homogeneous Markov chain.

A Markov chain represents a description of the time evolution of a probability distribution in the sample space. If $\boldsymbol{p}(t) = \{p_x(t) = \mathrm{Prob}(X(t) = x)\}_{x\in\Omega}$ is the probability distribution at time t, the distribution at later times $t' > t$ is given by

$$\boldsymbol{p}^{\mathrm{T}}(t)P(t)P(t+1)\cdots P\big(t'\big) = \boldsymbol{p}^{\mathrm{T}}\big(t'\big).$$

Markov chains are linear discrete dynamical systems of time evolution of the probability distribution [25]. The probability for the transition from the state x_0 at time t_0 to the state x_1 at time t_1 is given by the matrix element (x_0, x_1) of the product of Markov matrices $P(t)$ with the times t on the interval $[t_0, t_1]$,

$$\mathrm{Prob}\big(X(t_1) = x_1 \big| X(t_0) = x_0\big) = \big[P(t_0)P(t_0+1)\cdots P(t_1)\big]_{x_0,x_1}, \tag{6.10}$$

which is one of the forms of the Chapman–Kolmogorov equation [26]. If the Markov chain is time-homogeneous ($P(t) = P$), the expression above simplifies to

$$\mathrm{Prob}\big(X(t_1) = x_1 \big| X(t_0) = x_0\big) = \big[P^{t_1-t_0}\big]_{x_0,x_1}.$$

This relation is often used to test whether the observed process is a Markov chain or not.

Detailed Balance In physically motivated Markov chains one frequently encounters the condition of *detailed balance* which states that in equilibrium the occupation probability of all states is equal, so

$$\pi_x \,\mathrm{Prob}\big(X(t+1) = y \big| X(t) = x\big) = \pi_y \,\mathrm{Prob}\big(X(t+1) = x \big| X(t) = y\big),$$

or

$$\pi_x \big[P(t)\big]_{x,y} = \pi_y \big[P(t)\big]_{y,x},$$

where $\boldsymbol{\pi} = \{\pi_x\}_{x\in\Omega}$ is a stationary distribution. If a distribution $\boldsymbol{\pi}$ exists for which the condition of detailed balance applies, the corresponding Markov chain is called *reversible*.

Reducibility If a non-zero probability exists that from any state in the chain we can arrive at any other state, we say that the states *communicate* and the corresponding chain is said to be *irreducible*. In the opposite case, we can form subsets of states with respect to which the chain is irreducible.

Periodicity, Reproducibility, Ergodicity In Markov chains it is important whether we are able to return to the original state from the sample space Ω and how. A state is *periodic* if we can return to it along the paths with the number of steps whose common multiples are different from 1. In the opposite case the state is non-periodic. A state is *reproducible* if we can return to it in finite time. If the Markov chain is irreducible on the whole sample space Ω and all states are non-periodic and reproducible, the chain is said to be *ergodic*.

Stationary Distributions If the Markov chain is time-homogeneous, the Perron–Frobenius theorem [27] guarantees that for each irreducible part of the chain at least one set of points (vector) $\boldsymbol{\pi} = \{\pi_x > 0\}_{x\in\Omega}$ exists such that

$$\boldsymbol{\pi}^{\mathrm{T}} P = \boldsymbol{\pi}^{\mathrm{T}},$$

with the normalization $\sum_x \pi_x = 1$. The vector $\boldsymbol{\pi}$ is a left eigenvector of the matrix P with the maximum possible eigenvalue of 1. We call such vectors *stationary* (or *invariant*, or *equilibrium*) distributions of the Markov chain. Stationary distributions play the role of probability distributions that are preserved within the dynamics of the Markov chain. If a Markov chain is ergodic, exactly one stationary distribution $\boldsymbol{\pi}$ exists to which any initial probability distribution $\boldsymbol{p}$ converges as

$$\boldsymbol{\pi}^{\mathrm{T}} - \boldsymbol{p}^{\mathrm{T}} P^t = \mathcal{O}\big(|\nu|^t\big),$$

where ν is the second largest eigenvalue of P. A stationary distribution $\boldsymbol{\pi}$ may also exist in time-inhomogeneous chains, where we require

$$\boldsymbol{\pi}^{\mathrm{T}} P(t) = \boldsymbol{\pi}^{\mathrm{T}} \quad \forall t,$$

but its existence is not guaranteed.

Entropy One way to think about a Markov chain is to establish a probabilistic description of a random motion of a particle or a more general system between the states in the space Ω. Assume that the Markov chain is time-homogeneous and that the transition probabilities are given by the matrix P. We are interested in the probability that, given some initial discrete probability distribution μ, the particle follows the trajectory $\{x_i\}_{i=0}^t$ beginning at the state x_0, from whence it jumps to x_1, to $x_2, \ldots$ and arrives at x_t after time t. This discrete probability distribution is

$$p(x_0, x_1, \ldots, x_t) = \mu(x_0) P_{x_0,x_1} P_{x_1,x_2} \cdots P_{x_{t-1},x_t}.$$

The entropy of such a random process until time t is defined as

$$H(t) = -\sum_{(x_0,x_1,\ldots,x_t)\in\Omega^{t+1}} p(x_0, x_1, \ldots, x_t) \log p(x_0, x_1, \ldots, x_t)$$

and tells us something about the richness (complexity) of the possible trajectories of the process in the sample space [28]. We expect physics-inspired Markov

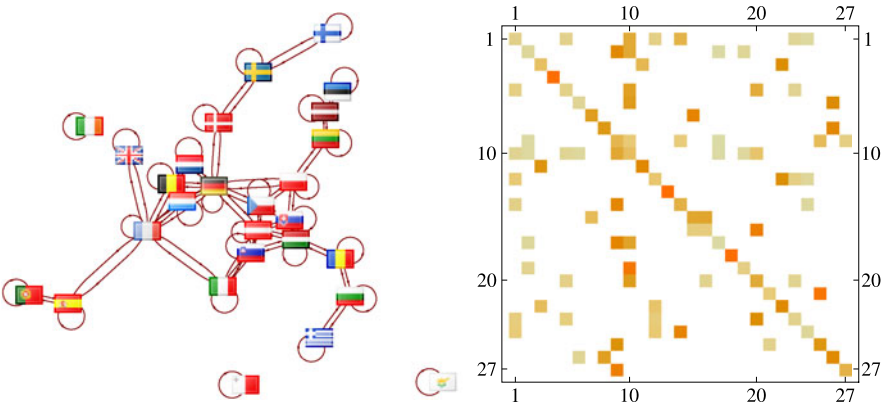


Fig. 6.6 Traveling within the EU as a discrete-time Markov chain. [*Left*] The graph of road and rail connections. [*Right*] The matrix elements of the Markov matrix P

chains to be ergodic, and in such chains the entropy increases linearly with time. Of course, this does not apply to periodic states where the complexity of the dynamics is strongly diminished. The rate of entropy growth in an ergodic chain with the Markov matrix P and stationary distribution $\boldsymbol{\pi}$ is given by

$$h = \lim_{t\to\infty} \frac{1}{t} H(t) = -\sum_{x,y\in\Omega} \pi_x P_{x,y} \log P_{x,y}.$$

If a Markov chain describes the dynamics of a sufficiently complex system, h is an approximation of the information entropy for that system, and is proportional to its statistical (Boltzmann) entropy [29, 30]. The connection between the rate of entropy growth and eigenvalues of Markov matrices is discussed in [31].

Example A tourist travels within the EU ($n = 27$ states). She decides about the entry into the next state upon looking at its relative size with respect to the sizes of all states accessible at that moment. The transitions between the states are a random process that can be described as a discrete-time Markov chain on the set of indices $\{i\}_{i=1}^{n}$ denoting individual states. The possibilities for the transitions are shown by the graph in Fig.6.6 (left) and are summarized by the function

$$c(i,j) = \begin{cases} 1; & \text{state } i \text{ connected to state } j \text{ by road or rail,} \\ 0; & \text{otherwise.} \end{cases}$$

Clearly, each state is connected to itself, hence $c(i,i) = 1$. The tourist departs the ith state heading for the jth state with the conditional probability

$$\text{Prob}(j|i) = \frac{c(i,j)S_j}{\sum_k S_k}, \quad c(k,i) \neq 0,$$

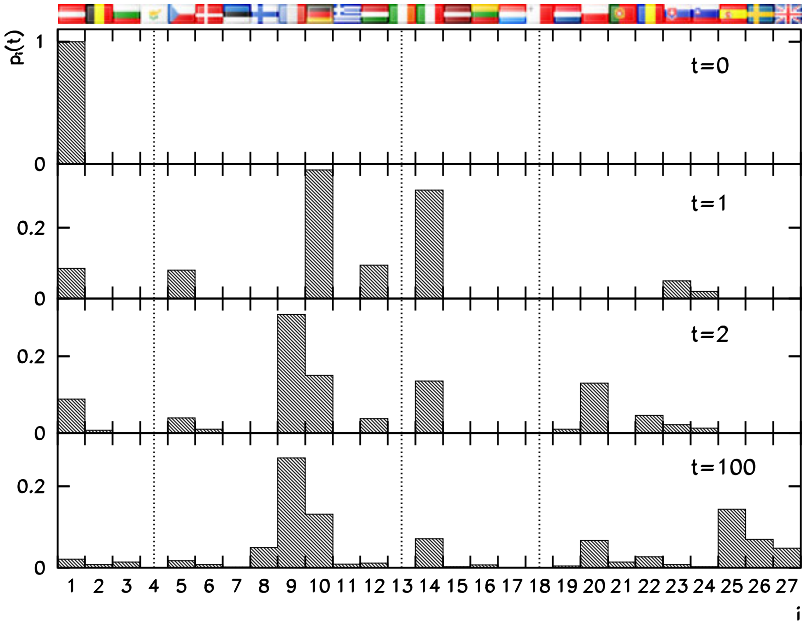


Fig. 6.7 The time evolution of the tourist's probability distribution for a trip started in Austria ($i = 1$). The distribution at $t = 100$ is an excellent approximation for the stationary distribution. Note that the shown bar sizes are *not* proportional to the sizes of the states! A good example is Finland ($i = 8$) that is reachable only through Sweden. The average probability of meeting the tourist in Finland is therefore only the seventh largest (4.9 %) although Finland is the fifth largest state. Three *dotted vertical lines* correspond to Cyprus, Malta, and Ireland, which the tourist never enters

where S_i is the size of the ith state. We arrange all conditional probabilities in the Markov matrix $P = [\text{Prob}(j|i)]_{i,j=1}^{n}$ shown in Fig. 6.6 (right).

The probability distribution of the tourist is represented by the array $\boldsymbol{p}(t) = \{p_i(t)\}_{i=1}^{n}$. This array can be treated as a vector of dimension n that evolves as

$$\boldsymbol{p}^{\mathrm{T}}(t) = \boldsymbol{p}^{\mathrm{T}}(0)\,P^t,$$

where $t \in \mathbb{N}_0$ counts the decisions on further travel ("time"). We are interested in the probability that the tourist will appear in the state i after time t if she started her travel in the state s, so $p_i(0) = \delta_{i,s}$. An example of a trip commenced in Austria is shown in Fig. 6.7. Gradually she wanders everywhere except Cyprus, Malta, and Ireland, which are not connected to the remaining states. All n states can thus be split in four connected subsets on which the Markov chain is irreducible. On each of these subsets we could devise a chain with reproducible states, but only on the largest subset of 24 states the chain is non-periodic and therefore also ergodic. Any initial position of the tourist ends up in a unique stationary distribution, the approximation of which is shown in Fig. 6.7 at $t = 100$.

6.4.2 Continuous-Time Markov Chains

It is not hard to generalize discrete-time Markov chains to continuous-time chains. Assume that a random process X with a countable sample space Ω is observed in continuous time $t \in \mathbb{R}$. In such a process, the probability for a transition from state x at time t to state y at a later time $t + \Delta t$ is

$$\mathrm{Prob}\big(X(t+\Delta t) = y \big| X(t) = x\big).$$

If the conditional probabilities satisfy the continuous analogue of (6.10), such a process represents a continuous-time Markov chain. In the limit $\Delta t \to 0$ we then have

$$\mathrm{Prob}\big(X(t+\Delta t) = y \big| X(t) = x\big) = \delta_{x,y} + \big[Q(t)\big]_{x,y}\Delta t + \ \mathcal{O}(\Delta t),$$

where $Q(t)$ is the *transition rate matrix*. Its matrix elements are $[Q(t)]_{x,y} \geq 0$ for $x \neq y$ and $[Q(t)]_{x,x} \leq 0$. From conservation of probability $\sum_{y\in\Omega} \mathrm{Prob}(X(t + \Delta t) = y|X(t) = x) = 1$ we obtain $\sum_y [Q(t)]_{x,y} = 0$ for each $x \in \Omega$, hence the decay rate of the individual state is

$$\big[Q(t)\big]_{x,x} = -\sum_{y\neq x}\big[Q(t)\big]_{x,y}.$$

Just like in the case of discrete chains (Sect. 6.4.1) the continuous-time Markov chains may be seen as a probabilistic description of the dynamics of some system in the set of states Ω. Let $\boldsymbol{p}(t) = \{p_x(t) = \mathrm{Prob}(X(t) = x)\}_{x\in\Omega}$ be the probability distribution of the system with respect to the states at time t. The time evolution of this distribution (as a vector) is then given by a system of ordinary differential equations

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{p}^{\mathrm{T}}(t) = \boldsymbol{p}^{\mathrm{T}}(t)Q(t). \tag{6.11}$$

The solution of this system is

$$\boldsymbol{p}^{\mathrm{T}}\big(t'\big) = \boldsymbol{p}^{\mathrm{T}}(t)\mathcal{P}\big(t,t'\big),$$

where the form of the matrix $\mathcal{P}$ depends on whether the Markov matrix Q is time-dependent or not. For a homogeneous chain ($Q(t) = Q$) we get

$$\mathcal{P}\big(t,t'\big) = \exp\big(\big(t'-t\big)Q\big).$$

In the case of a non-homogeneous chain (Q depends on time) the formal solution is

$$\mathcal{P}\big(t,t'\big) = \hat{\mathcal{T}}\exp\left(\int_t^{t'} Q(\tau)\,\mathrm{d}\tau\right), \tag{6.12}$$

where the time-ordering operator $\hat{\mathcal{T}}$ [32] creates the correct time ordering of the integration limits once the exponential function of matrices is power-expanded. Due to its complicated structure the solution (6.12) is generally useful at short times only; in other cases, we need to tackle the system (6.11) directly.

A continuous-time Markov chain is converted to a discrete-time chain by choosing the step size Δt and setting $P(t) = \mathcal{P}(t, t + \Delta t)$. If one assumes that the random process X at time t is in the state x, the probability that the process remains in this state until time $t + \Delta t$ is

$$\text{Prob}\big(X(r) = x \,\big|\, X(t) = x\big) = \exp\big(\Delta t [Q]_{x,x}\big) \quad \forall r \in (t, t + \Delta t]$$

in the case of a homogeneous chain, while it is

$$\text{Prob}\big(X(r) = x \,\big|\, X(t) = x\big) = \hat{\mathcal{T}} \exp\left(\int_0^{\Delta t} \big[Q(t + \tau)\big]_{x,x} \, \mathrm{d}\tau\right) \quad \forall r \in (t, t + \Delta t]$$

in the case of a non-homogeneous one. Reducibility, periodicity, reproducibility, and ergodicity of states for continuous-time Markov chains are introduced by complete analogy to discrete-time chains (Sect. 6.4.1).

Example Imagine an array of n sites with labels from 1 to n. Particles may jump between neighboring sites. We describe the dynamics of the particles by a continuous-time-homogeneous Markov chain, where the sample space are the sites on the chain, $\Omega = \{i\}_{i=1}^n$. The particle at site i can jump to the neighboring site to the left (label $i - 1$) or to the right (label $i + 1$) with the rate r, if this site exists. This rule can be condensed into the transition rate matrix

$$Q = \begin{pmatrix} -r & r & & & 0 \\ r & -2r & r & & \\ & \ddots & \ddots & \ddots & \\ & & r & -2r & r \\ 0 & & & r & -r \end{pmatrix}.$$

The distribution of the particles along the chain at time t is given by the array $\boldsymbol{p}(t) = \{p_i(t)\}_{i=1}^n$, which we interpret as a vector $(p_1(t), p_2(t), \ldots, p_n(t))^{\mathrm{T}}$. Its time evolution is given by the system of differential equations

$$\frac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{p}^{\mathrm{T}}(t) = \boldsymbol{p}^{\mathrm{T}}(t) Q.$$

At $t = 0$ all particles are at the chosen site s, thus $p_i(0) = \delta_{i,s}$. The time evolution of the distribution for $r = 0.1$ and $s = 11$ in the case of a chain of length $n = 21$ is shown in Fig. 6.8 (left). Early on, the distribution acquires a Gauss-like shape, but after long times it reaches the boundaries and flattens out. Ultimately we reach the limit $\lim_{t\to\infty} p_i(t) = 1/n$ representing the stationary distribution, $p_i = \pi_i$.

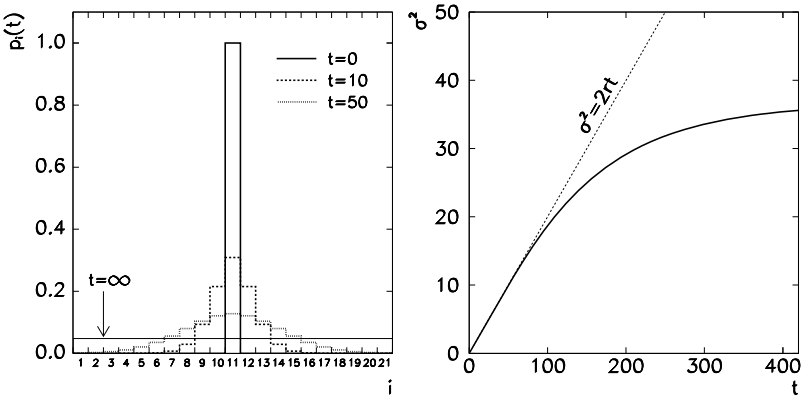


Fig. 6.8 Motion of particles between the states of a one-dimensional array of length $n = 21$ as a continuous-time Markov chain. [*Left*] Time evolution of the distribution $\boldsymbol{p}$ with particles initially at $s = 11$, with $r = 0.1$. After long times we obtain a constant distribution $p_i(t) = 1/21 \approx 0.04762$. [*Right*] Time evolution of the distribution variance

The average of an observable f with respect to the distribution $\boldsymbol{p}$ is $\langle f(i)\rangle_{i,t} = \sum_{i=1}^{n} f(i)\,p_i(t)$. The dynamics of the system is such that the distribution average does not change with time, $\langle i\rangle_{i,t} = s$. What does change is the variance,

$$\sigma^2(t) = \big\langle (i-s)^2 \big\rangle_{i,t},$$

shown in Fig. 6.8 (right). We see that $\sigma^2(t) \approx 2rt$ until the distribution spreads out to the boundaries of the chain which it is unable to cross. The direct proportionality of the variance and time indicates that the diffusion of particles in this system is normal, with the diffusion constant $D = r$.

6.5 Noise

Noise (for continuous or discrete time) is defined as a real random process Z which is wide-sense stationary and ergodic (see p. 282), and has zero average,

$$\big\langle Z(t) \big\rangle = 0,$$

a rapidly decreasing time auto-correlation R_{ZZ},

$$R_{ZZ}(t_1, t_2) = \big\langle Z(t_1) Z(t_2) \big\rangle = R_{ZZ}(t_1 - t_2),$$

and probability density

$$p_Z(x) = \big\langle \delta\big(x - Z(t)\big) \big\rangle.$$

Here $\langle\cdot\rangle$ denotes the statistical average over various realizations of the random process. Because noise is ergodic, the statistical average is equal to the time average. Noise as an abstract entity and, above all, in its individual realizations, is frequently used in stochastic simulations in physics, electric engineering, and acoustics.

6.5.1 Types of Noise

Noise is classified according to the functional forms of its probability density p_Z and auto-correlation R_{ZZ}. Auto-correlation is the most important property of noise defining its characteristics. It is usually presented in Fourier space by computing the *average power spectral density* (PSD). The power density S_Z is equal to the square of the Fourier transform of the signal's auto-correlation (see (4.1) and (4.18), as well as Sect. 4.2.7). For continuous-time noise we obtain (up to a factor)

$$S_Z(\omega) \propto \mathcal{F}[R_{ZZ}](\omega),$$

while for discrete-time noise

$$S_{Z,k} \propto \left(\mathcal{F}_N[R_{ZZ}]\right)_k.$$

Noises with a spectral density of the form

$$S_Z(\omega) \propto \omega^{-a}, \quad a > 0,$$

are said to possess *colors*, which are assigned according to the values of a. The most well-known noises are white ($a = 0$), pink ($a = 1$), and red ($a = 2$). Examples are shown in the left panels of Fig. 6.9. The corresponding distributions of the values of the noise are shown in the right panels.

In the strict sense, colored noises with parameters $a \geq 1$ are non-stationary random processes [33], since $S_Z \propto \omega^{-a}$ has a singularity at the origin. In practical calculations we require that S_Z differs from zero only for frequencies ω larger than some minimal frequency ω_0, e.g. $\omega_0 \approx \mathcal{O}(N^{-1})$, where N is the sample size.

White Noise The Fourier spectrum of *white noise* is "flat", $S_Z(\omega) = \text{const}$. The noise signal is therefore δ-auto-correlated in the statistical sense. For continuous-time noise this implies

$$R_{ZZ}(\tau) = \left\langle Z(t)^2 \right\rangle \delta(\tau),$$

while for discrete-time noise

$$R_{ZZ}(\tau) = \left\langle Z(t)^2 \right\rangle \delta_{\tau,0}.$$

White noise is strict-sense stationary and resembles the noise caused by thermal fluctuations of charge carriers at finite temperatures. We are commonly referring to Gaussian white noise since its probability density p_Z is the Gauss function.

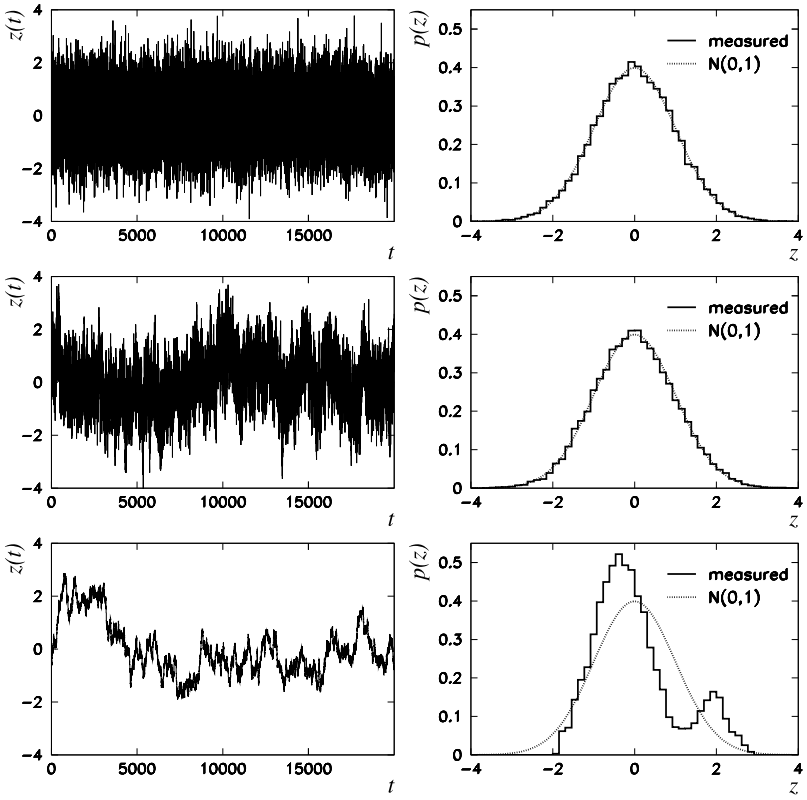


Fig. 6.9 Examples of realizations of various types of noise (*left panels*) and the corresponding distributions of values in these realizations (*right panels*). The color of the noise is defined by the parameter a in the power spectral density $S_Z(\omega) \propto \omega^{-a}$. [*Top*] White noise ($a = 0$). [*Center*] Pink noise ($a = 1$). [*Bottom*] Red (or Brownian) noise ($a = 2$). The *figures at the right* also show the normal distribution $N(0, 1)$

Pink Noise *Pink* or *flicker noise* has the power spectral density $S_Z(\omega) \propto \omega^{-1}$. It occurs, for example, in circuits with semiconductor elements.

Red (Brownian) Noise *Red* or *Brownian noise* has the power spectral density $S_Z(\omega) \propto \omega^{-2}$ and is typically encountered in random-walk processes. Realizations $z(t)$ of red noise can be obtained from white noise $w(t)$ by means of the Wiener process [34] defined by

$$z(t) = z(0) + \int_0^t w(\tau)\,\mathrm{d}\tau.$$

Once this equation is rewritten as $\mathrm{d}z(t)/\mathrm{d}t = w(t)$, the Fourier transformation can be applied to both sides, and we get

$$S_Z(\omega) = \frac{\langle w(t)^2 \rangle}{\omega^2}.$$

6.5.2 Generation of Noise

Generation of noise with appropriate statistical and spectral properties is of crucial importance in certain applications. A system with pronounced sensitivity to high-frequency content of signals will respond differently if white or red noise is introduced at its input. Two classes of noise generation methods exist: *online* and *offline*. In online mode we generate the signal values $\{z_i\}$ of noise Z sequentially, for example, while the simulation is running; the values should therefore be generated as quickly as possible. In offline mode, the complete realization of the noise $\{z_i\}_{i=0}^{N-1}$ with an arbitrary spectral density S_Z is generated in a single shot. Both methods are described in the following.

Online Mode White noise with a desired probability density p_Z can be generated directly by using a generator of random numbers that are distributed according to p_Z (see Appendix C).

A review of pink-noise generators can be found at [35]. The improved correlated generator [36] based on random weighted summation of n white-noise signals is particularly simple and effective. To generate a signal with the spectrum $S_Z(\omega) \propto \omega^{-1}$ we have to choose the appropriate weights $\boldsymbol{w} = \{w_i\}_{i=1}^n$ for the contributions of white noise, the bounding probabilities $\boldsymbol{p} = \{p_i\}_{i=1}^n$ for random summation of individual contributions, and the vector describing the state of the generator, $\boldsymbol{c} = \{c_i\}_{i=1}^n$. The components of this vector should be initialized with random numbers distributed according to $U(-1, 1)$ (uniformly on $[-1, 1]$). The algorithm to obtain the next value of the signal of pink noise is

Data: arrays of weights $\boldsymbol{w} = \{w_i\}_{i=1}^n$ and probabilities $\boldsymbol{p} = \{p_i\}_{i=1}^n$.
Input: state vector of the generator $\boldsymbol{c} = \{c_i\}_{i=1}^n$.
Draw a random x from $U(0, 1)$;
for $i = 1, 2, \ldots, n$ **do**
  if $x \le p_i$ **then**
    Draw a random y from $U(-1, 1)$;
    $c_i = w_i y$;
    Break the loop;
  end
end
$z = \sum_{i=1}^n c_i$;
Output: next value of the signal z and the state vector $\boldsymbol{c}$.

The uniform generator used to generate x and y in this algorithm may be very simple, as its properties do not influence significantly the quality of the generated pink noise. In [36] they recommend $n = 5$ and

$$\boldsymbol{w} = \{0.9506, 0.74235, 0.64925, 0.77175, 0.85015\},$$
$$\boldsymbol{p} = \{0.00198, 0.01478, 0.06378, 0.23378, 0.91578\}.$$

The generated signals have values on the interval $[-W, W]$, where $W = \sum_{i=1}^{n} w_i$. The values are distributed approximately according to $N(0, 1)$.

Red or Brownian noise can be generated by using a one-dimensional random walk from which the average of the last n steps is continuously subtracted. Imagine a random walk defined by the recurrence

$$a_{t+1} = a_t + \varepsilon w_t,$$

where w_t is a random number distributed according to $N(0, 1)$, and ε is the length of the step. The sequence of values of approximately red noise $\boldsymbol{z} = \{z_t\}_{t=0}^{\infty}$ is obtained by applying a linear filter on the sequence $\boldsymbol{a} = \{a_t\}_{t=0}^{\infty}$,

$$z_t = a_t - \frac{1}{m} \sum_{i=0}^{m-1} a_{t-i}.$$

The shape of the spectrum is controlled by the filter length m. Assume that we have computed the signal $\boldsymbol{z} = \{z_i\}_{i=0}^{N-1}$ of length N and its single-sided spectrum $\boldsymbol{S} = \{S_i\}_{i=0}^{N/2} = \mathrm{PSD}[\boldsymbol{z}]$. For $i > K = N/(2m)$ we obtain $\boldsymbol{S} \sim \mathcal{O}(i^{-2})$, while for $i < K$ the spectrum $\boldsymbol{S}$ does not change significantly. To generate a signal resembling red noise, we therefore need $N = \mathcal{O}(m)$. If we choose $\varepsilon = 2/\sqrt{m}$ and $K \gg 1$, the generated values are distributed according to $N(0, 1)$.

Offline Mode The simplest algorithm to generate a complete sample of noise with an arbitrary spectral density S_Z is based on the Fourier transformation. Here we adopt, with minor modifications, the algorithm from [37]. Generate the array $\boldsymbol{c} = \{c_i\}_{i=0}^{N-1}$ (with N odd) by using

$$c_0 = 0, \qquad c_{N-j}^{*} = c_j = \sqrt{S_Z\left(\frac{j\omega'}{N}\right)}(a_j + \mathrm{i}b_j), \quad j = 1, 2, \ldots, (N-1)/2,$$

where a_j and b_j are random numbers distributed according to $N(0, 1)$, and ω' is a characteristic frequency that we set according to the desired frequency range of the generated spectrum. By computing the inverse Fourier transform of $\boldsymbol{c}$,

$$z_k = \frac{1}{\|\boldsymbol{c}\|_2} \sum_{j=0}^{N-1} e^{\mathrm{i}2\pi jk/N} c_j,$$

we obtain the desired realization of noise. If S_Z is a bounded function of ω, the average probability density of the values z_k converges to $N(0,1)$ with increasing N. The same applies to colored noises with the parameter $a \in [0,1]$. In contrast, for a larger than 1 the probability density of the values in individual signals (and on average) deviates far from the normal distribution. Figure 6.9 shows the signals of white ($a = 0$), pink ($a = 1$), and red noise ($a = 2$), as well as the probability densities of values in these signals obtained by this method.

6.6 Time Correlation and Auto-Correlation

Correlations measure statistical dependence or similarity of quantities. The mathematical formulation of correlation varies according to the type of the quantities, but its meaning remains the same. Imagine two continuous-time random processes F and G. We define their time correlation as the statistical average of the product of the processes, one of which is shifted with respect to the other by τ [9]:

$$R_{FG}(\tau) = \big\langle F(t)^* G(t+\tau) \big\rangle.$$

The time shift τ is the parameter of the correlation. If the processes F and G are ergodic and their correlation is equal to the time correlation of their realizations f and g, such processes are known as *jointly ergodic*, and we have

$$R_{FG}(\tau) = R_{fg}(\tau) = \big\langle f(t)^* g(t+\tau) \big\rangle_t = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} f\big(t'\big)^* g\big(t'+\tau\big)\,\mathrm{d}t',$$

where the average $\langle\cdot\rangle_t$ is defined in (6.4). For discrete times the integral in the expression above should be rewritten as a sum, in accordance with the discrete definition of the time average, (6.5).

More formally, correlation can be understood as a binary operation that assigns the correlation function R_{fg} to the functions f and g. The precise form of the assignment depends on the definition domains X of the functions, e.g.

$$R_{fg}(\tau) = \int_X f(t)^* g(t+\tau)\,\mathrm{d}^n t, \quad X = \mathbb{R}^n, \mathbb{C}^n,$$

or

$$R_{fg}(\tau) = \sum_{t\in X} f(t)^* g(t+\tau), \quad X = \mathbb{Z}^n, \mathbb{Z}_N^n.$$

Correlation therefore represents an overlap integral in the domain of the shifted functions, where the shift is the argument of the correlation. The correlation of functions in the domain $\mathbb{Z}_N^2$, where $N = 20$, is illustrated by the Example below (see Fig. 6.10). An overview of the use of time correlations in theoretical physics is given by [38].

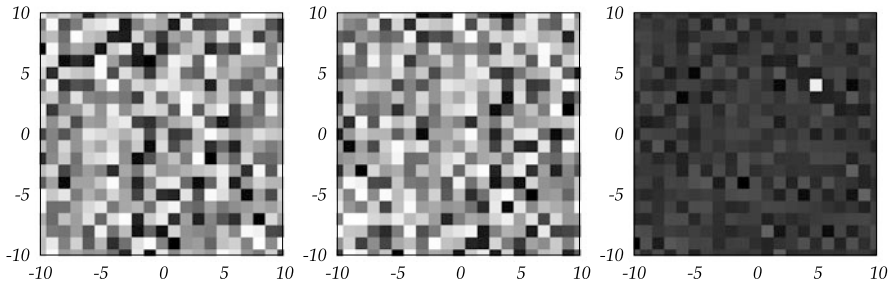


Fig. 6.10 The basic operational idea of an optical mouse. [*Left*] First acquired image: a two-dimensional sample (scalar field) F of size 20×20 with values $F_{i,j} \in [0, 1]$ and periodic boundary conditions. [*Center*] The shifted version of the image $\widetilde{F}_{i,j} = F_{i-5,j-4}$. [*Right*] The correlation of the original and shifted fields $C_{i,j} = (1/N^2) \sum_{\alpha,\beta} \widetilde{F}_{i+\alpha,j+\beta} F_{\alpha,\beta}$ indicates the shift of the mouse from the current position at $(0, 0)$ to $(5, 4)$

Basic Properties and Relation to Convolution Regardless of the character of the objects f and g (realizations of random processes, discrete or continuous functions), their correlation has some common properties. A correlation is *symmetric*,

$$R_{gf}(-\tau) = R_{fg}(\tau)^*, \tag{6.13}$$

and bounded in the sense

$$\big|R_{fg}(\tau)\big|^2 \le R_{ff}(0) R_{gg}(0), \qquad \big|R_{fg}(\tau)\big| \le \frac{1}{2}\big(R_{ff}(0) + R_{gg}(0)\big).$$

A correlation between different random processes or their realizations (functions) f and g is also called *cross-correlation*, while for $f = g$ we are referring to an *auto-correlation function* (ACF). Auto-correlation is the sum of auto-correlations of non-correlated parts: let $f = \sum_i s_i$ and $R_{s_i s_j} = 0$ for $i \neq j$. Then

$$R_{ff} = \sum_i R_{s_i s_i}.$$

The correlation R_{fg} can be expressed by the convolution, $* : (f, g) \mapsto f * g$, as

$$R_{fg}(\tau) = \big[f^*(-t) * g(t)\big](\tau).$$

When the existence of the correlation or of the convolution itself is questioned, we rely on the property that $f, g \in L^p(X)$ implies $R_{fg} \in L^p(X)$. In computations of correlations and convolutions where Fourier transformations are involved, we frequently assume that f and g are rapidly decreasing smooth functions [39]. For details see [40, 41].

Example An optical mouse is a standard external device for modern computers. By moving the mouse we communicate to a certain program our desire to move the graphical pointer. The mouse senses the motion by a small CCD camera capturing

the reflection of light from the surface illuminated by a light-emitting diode. (The image acquisition rate is usually about 1 kHz.) The program compares the subsequent images and uses correlations between them to determine the direction and length of the move. An example of such a computation is shown in Fig. 6.10. The more random the acquired samples are, the more precise the determination of the shift can be. This is why optical mice perform poorly on polished reflective surfaces or surfaces with fine periodic textures.

6.6.1 Sample Correlations of Signals

In practice we are not dealing with infinitely long signals but with their finite chunks or *samples*. We would like to use the sample to learn as much as possible about the whole signal, for example, by computing the approximation of its time average or a correlation between two signals. Let an infinitely long signal f be defined on the whole real or discrete time axis. Assume we only know its sample $\hat{f}$ in the time interval $[t_0, t_0 + T)$ of length T, thus

$$\hat{f}(t) = \begin{cases} f(t_0 + t); & t \in [0, T) \subset \mathbb{R} \text{ (continuous signal)}, \\ f(t_0 + t); & t \in [0, T - 1] \subset \mathbb{Z} \text{ (discrete signal)}. \end{cases}$$

For continuous time, a correlation of two such samples $\hat{f}$ and $\hat{g}$ is defined as

$$\mathcal{R}_{\hat{f}\hat{g}}(\tau) = w(\tau, T) \int_0^{T-|\tau|} \mathrm{d}t \begin{cases} \hat{f}(t)^* \hat{g}(t + \tau); & \tau \in [0, T), \\ \hat{f}(t + |\tau|)^* \hat{g}(t); & \tau \in (-T, 0), \end{cases}$$

while for discrete time we use

$$\mathcal{R}_{\hat{f}\hat{g}}(\tau) = w(\tau, T) \sum_{t=0}^{T-|\tau|-1} \begin{cases} \hat{f}(t)^* \hat{g}(t + \tau); & \tau \in [0, T - 1], \\ \hat{f}(t + |\tau|)^* \hat{g}(t); & \tau \in [-T + 1, 0]. \end{cases}$$

The weight w will be determined later; for the moment we require that it behaves asymptotically as $w(\tau, T) \sim T^{-1}$ when $T \to \infty$, so that we have the limit

$$\lim_{\substack{T \to \infty \\ t_0 \to -\infty}} \mathcal{R}_{\hat{f}\hat{g}} = R_{fg}.$$

Sample correlations defined in this manner have the symmetry $\mathcal{R}_{\hat{f}\hat{g}}(-\tau) = \mathcal{R}_{\hat{g}\hat{f}}(\tau)^*$, just as in (6.13), but they do not necessarily possess other previously enumerated properties of correlations. Which properties are shared among the infinite-signal and sample correlations strongly depends on the choice of the weight.

Choice of the Weight In choosing the weight to compute the sample correlation we are usually forced to make a compromise between mathematical correctness and technical usefulness. We exploit the following facts. If the signals f and g are realizations of jointly ergodic random processes F and G that are wide-sense stationary (WSS, p. 282), we have

$$\left\langle R_{\hat{f}\hat{g}}(\tau)\right\rangle = \big(T - |\tau|\big)w(\tau, T)R_{FG}(\tau). \tag{6.14}$$

If F is a discrete-time Gaussian random process and f is its realization, the variance of the auto-correlation is

$$\operatorname{var}\big[\mathcal{R}_{\hat{f}\hat{f}}(\tau)\big] \approx T w(\tau, T)^2 \sum_{t\in Z}\big\{R_{FF}(t)^2 + R_{FF}(t+\tau)R_{FF}(t-\tau)\big\}, \tag{6.15}$$

and this dependence on T and τ roughly applies also to other processes from the WSS class [42]. We conclude that it is sensible to use the weight

$$w(\tau, T) = \frac{1}{T - |\tau|},$$

since by (6.14) the sample correlation then yields the correct estimate of the correlation of random processes $\langle \mathcal{R}_{\hat{f}\hat{g}}(\tau)\rangle = R_{FG}(\tau)$. Such sample correlations are *unbiased*. Their main weakness is that for fixed T and increasing τ the amount of the signal used in the computation of the sample correlation decreases, causing the statistical error to behave as $\mathcal{O}((T - |\tau|)^{-1/2})$ according to (6.15). Therefore, for $\tau = \mathcal{O}(T)$ the sample correlation becomes completely unpredictable. Besides, the sample correlation does not necessarily fulfill the inequality

$$\big|\mathcal{R}_{\hat{f}\hat{f}}(\tau)\big| \le \big|\mathcal{R}_{\hat{f}\hat{f}}(0)\big|, \tag{6.16}$$

which is crucial in some applications. Therefore, one often prefers the weight

$$w(\tau, T) = \frac{1}{T},$$

which generates *biased* correlation estimates. In this case the sample correlation does not accurately estimate the correlation of the processes

$$\left\langle \mathcal{R}_{\hat{f}\hat{g}}(\tau)\right\rangle = \left(1 - \frac{|\tau|}{T}\right)R_{FG}(\tau),$$

but the statistical error is reduced: it becomes almost independent of the shift τ and is of the order $\mathcal{O}(T^{-1/2})$, and at the same time inequality (6.16) is valid.

6.6.2 Representation of Time Correlations

Assume that for signals f and g we computed the time averages $\mu = \langle f(t)\rangle_t$ and $\nu = \langle g(t)\rangle_t$. In terms of these averages and the remainders we can write

$$f(t) = \mu + x(t), \quad g(t) = \nu + y(t).$$

The time correlation of f and g is then

$$R_{x+\mu,y+\nu}(\tau) = \mu\nu + R_{xy}(\tau).$$

Therefore, the correlation with the product of the averages subtracted is

$$R^{(1)}_{fg}(\tau) = R_{fg}(\tau) - \mu\nu = R_{f-\mu,g-\nu}(\tau).$$

Alternatively, the averages are subtracted from the signal prior to the calculation of the correlation. We see that at shift $\tau = 0$, the correlation $R^{(1)}_{fg}$ is equal to the covariance with respect to the time average,

$$R^{(1)}_{fg}(0) = \big\langle \big(f(t) - \mu\big)\big(g(t) - \nu\big)\big\rangle_t.$$

In auto-correlation we have $f = g$ and the covariance is equal to the variance of the signal with respect to the time average, $R^{(1)}_{ff}(0) = \langle (f(t) - \mu)^2\rangle_t$. Because the variance of f is known, we divide this auto-correlation by the variance, and display

$$R^{(2)}_{ff}(\tau) = \frac{R^{(1)}_{ff}(\tau)}{R^{(1)}_{ff}(0)}.$$

6.6.3 Fast Computation of Discrete Sample Correlations

The correlation of finite sequences $a = \{a_i\}_{i=0}^{N-1}$ and $b = \{b_i\}_{i=0}^{N-1}$ with periodic boundary conditions $a_{i+N} = a_i$ and $b_{i+N} = b_i$ is a new periodic sequence $c = \{c_i\}_{i=0}^{N}$ with the terms $c_i = \sum_{j=0}^{N-1} a_j^* b_{j+i}$. This direct summation requires $\mathcal{O}(N^2)$ operations. A faster way of computing the correlation is possible by using the discrete Fourier transformation (DFT, see (4.11) and [43]). Namely, the Fourier transforms of the arrays a, b, and c are related by

$$\big(\mathcal{F}_N[c]\big)_i = N\big(\mathcal{F}_N[a]\big)_i^*\big(\mathcal{F}_N[b]\big)_i, \quad i = 0, 1, \ldots, N-1,$$

so that the inverse DFT gives us the correlation of periodic arrays

$$c = \mathcal{F}_N^{-1}[d], \quad d = N\Big(\big(\mathcal{F}_N[a]\big)_i^*\big(\mathcal{F}_N[b]\big)_i\Big)_{i=0}^{N-1}. \tag{6.17}$$

By using the fast DFT (FFT, Sect. 4.2.5) the correlation c according to the formula above can be computed in $\mathcal{O}(N \log_2 N)$ operations.

In computing the sample correlation of samples $\{a_i\}$ and $\{b_i\}$ we wish to effectively compute the sum of the form

$$c_i = \sum_{j=0}^{N-i-1} a_j^* b_{j+i}, \quad i = 0, 1, \ldots, N-1.$$

This is done by extending the arrays $\{a_i\}$ and $\{b_i\}$ to double their length and filling the attached part with zeros. Thus we obtain two new arrays,

$$\tilde{a} = \{a_0, a_1, \ldots, a_{N-1}, 0, \ldots, 0\}, \qquad \tilde{b} = \{b_0, b_1, \ldots, b_{N-1}, 0, \ldots, 0\},$$

for which we assume periodic boundary conditions $\tilde{a}_{i+2N} = \tilde{a}_i$ and $\tilde{b}_{i+2N} = \tilde{b}_i$. We rewrite the sample correlation as a correlation of periodic arrays

$$c_i = \sum_{j=0}^{2N-1} \tilde{a}_j^* \tilde{b}_{j+i}, \tag{6.18}$$

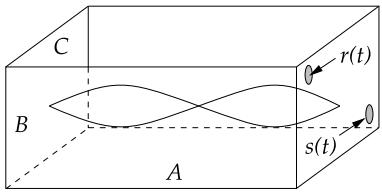
where only the first N terms of c are used in the computation of the sample correlation. A sum of the form (6.18) can be computed via (6.17) by using the FFT, except that the transforms appearing in it act on arrays of length $2N$. (The procedure described here closely resembles the multiplication of polynomials by using the FFT; see Sect. 4.2.6.)

In the case of two *non-periodic* signals whose correlation length is much smaller than the size of the samples picked from these signals, we may assume, without major loss of precision, that the samples are periodic. This allows us to again use (6.17) to compute the sample correlation.

Example We often perform measurements by perturbing the system by a real periodic signal

$$s(t) = S\cos(\omega t)$$

with known amplitude S and angular frequency ω, and observe the response of the system $r(t)$. An example of such analysis is the determination of the resonance spectrum of an acoustic resonator shown in the figure. On the resonator wall we use a loudspeaker to generate the perturbation $s(t)$ with the frequency $\nu = \omega/(2\pi)$, and measure the response $r(t)$ by a microphone. This spectroscopic method of testing objects or materials is known as the *continuous wave technique*. In all systems with a linear response to external perturbations the response can be written in the form



$$r(t) = R\cos(\omega t + \phi),$$

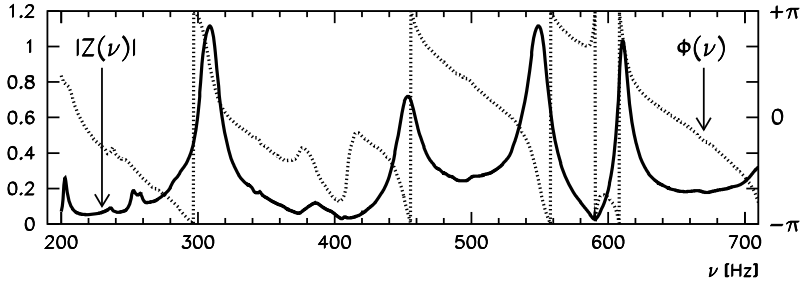


Fig. 6.11 The frequency dependence of the amplitude $|Z|$ and phase $\phi = \arg(Z)$ of the ratio between the complex response and complex perturbation during the excitation of a shoe-box acoustic resonator with sides $A = 56.7$ cm, $B = 38.5$ cm, and $C = 24$ cm. Near the resonances the phase changes rapidly and is approximately $\pm\pi/2$

where the amplitude of the response $R \propto S$ and the phase ϕ can be functions of ω. In complex notation, the response $\widetilde{r}(t) = R\mathrm{e}^{\mathrm{i}(\omega t+\phi)}$ is just the perturbation $\widetilde{s}(t) = S\mathrm{e}^{\mathrm{i}\omega t}$ multiplied by the factor $Z = (R/S)\mathrm{e}^{\mathrm{i}\phi}$, which we wish to determine. Because we generate the perturbation and know it exactly, Z can be computed by correlating the response and the complex perturbation,

$$\left\langle \widetilde{s}(t)^* r(t) \right\rangle_t = \frac{1}{2} S R \mathrm{e}^{\mathrm{i}\phi} = \frac{1}{2} Z S^2 .$$

An example of the determination of the amplitude $|Z|$ and the phase $\phi = \arg(Z)$ as functions of the frequency ν is shown in Fig. 6.11.

6.7 Auto-Regression Analysis of Discrete-Time Signals ⋆

In the field of digital signal analysis [10, 33] the *auto-regression model* (ARM) is a method that is also known as the *infinite impulse response filter* (IIR) or the *all-pole filter*, while in physical applications it goes under the name of *maximum-entropy model* (MEM). Here we discuss a very specific use of the theory, the prediction of signals and the estimate of their spectra.

Auto-regression modeling is closely related to linear prediction, and often in literature no distinction is made between them. A concise overview of linear prediction can be found in the classic paper [44], and a broader description of well-established methods and technicalities in [42]. General theory of linear prediction is presented in [45, 46], and the connection to linear models in [47, 48].

Preparing the Data for AR Analysis Let us discuss only discrete-time signals defined at times $t \in \mathbb{Z}$. If the signal $s(\tau)$ is continuous, we discretize it as $s_{\text{discrete}}(t) = s_{\text{continuous}}(\tau_0 + t\,\Delta t)$, where Δt is the time step and τ_0 the chosen origin. By discretizing the signal its spectrum is artificially constrained to the frequency

range $[-\nu_c, \nu_c]$, where $\nu_c = 1/(2\Delta t)$ is the Nyquist frequency (see (4.6)). From the signal we then try to remove all constant contributions or obvious dependencies, for example,

$$x(t) = s(t) - \text{power dependency of the signal.}$$

The signal x obtained in this way becomes approximately wide-sense stationary (WSS) and comparable to a realization of some ergodic random process, and its average is practically zero. Occasionally we attempt to use power transformations to modify the distribution of data so that it becomes more similar to one of the standardized distributions (for example, normal distribution) [49].

6.7.1 Auto-Regression (AR) Model

An auto-regression model for a discrete-time signal x is defined as the recurrence

$$x(t) = -\sum_{i=1}^{p} a_i x(t-i) + \varepsilon(t), \tag{6.19}$$

where $\varepsilon(t)$ is the *remainder* (or *deviation*, or *error*), while p is the *order* of the AR model. We wish to find the order p and coefficients $\{a_i\}_{i=1}^{p}$ such that the remainder ε is minimal and possibly similar to a realization of white noise.

Assume that x and ε are realizations of independent random processes X and E that are wide-sense stationary. Equation (6.19) then represents a relation between these processes. We multiply it by $X(t')^*$ from the left and right, and average over time. We obtain a recurrence relation for the auto-correlation

$$R_{XX}(t'-t) = -\sum_{i=1}^{p} a_i R_{XX}(t'-t-i) + \delta_{t,t'}\langle E(t)^2\rangle. \tag{6.20}$$

By introducing the vector of coefficients $\boldsymbol{a} = (a_1, a_2, \ldots, a_p)^{\mathrm{T}}$ and denoting $\rho = \langle E(t)^2\rangle$, the equation above can be rewritten in the Yule–Walker matrix form

$$R\begin{pmatrix} 1 \\ \boldsymbol{a} \end{pmatrix} = \begin{pmatrix} \rho \\ \mathbf{0} \end{pmatrix}, \tag{6.21}$$

where R is the auto-correlation matrix

$$R = \begin{pmatrix} R_{XX}(0) & R_{XX}(1)^* & R_{XX}(2)^* & \cdots & R_{XX}(p)^* \\ R_{XX}(1) & R_{XX}(0) & R_{XX}(1)^* & \cdots & R_{XX}(p-1)^* \\ \vdots & \vdots & \vdots & & \vdots \\ R_{XX}(p) & R_{XX}(p-1) & R_{XX}(p-2) & \cdots & R_{XX}(0) \end{pmatrix}.$$

The properties of auto-correlation (p. 305) ensure that R is positive definite.

The Yule–Walker system (6.21) relates the coefficients a_i of the AR model, the average square of the remainder, ρ, and the auto-correlation of the process, R_{XX}. If we know R_{XX}, we can determine a_i and ρ, or vice versa. The matrix R has a Toeplitz structure and the system can be effectively solved by the Levinson–Durbin method [50] in $\mathcal{O}(p^2)$ steps or one of the fast methods (see Sect. 3.2.4). With the solution we also obtain the coefficients a_i. Moreover, the solution at order p can be recursively used to seek the solution at order $p+1$. Levinson–Durbin recurrence lies at the heart of digital signal analysis [42, 46].

If the processes X and E are ergodic, statistical averages in (6.21) may be replaced by time averages, $R_{XX} = R_{xx} = \langle x(t')^* x(t)\rangle_t$ and $\langle E(t)^2\rangle = \langle \varepsilon(t)^2\rangle_t$. Because the model (6.19) is so general, and the constraints on the remainder ε so loose, the coefficients $\{a_i\}_{i=1}^p$ at chosen p can be calculated in several ways [42]. The most well-known are the method of least squares in its auto-correlation or covariance version, and the Burg's algorithm.

Determining the Parameters by the Auto-Correlation Method Let us select a sample of length T from the signal x (signal x different from zero on the interval $t \in [0, T-1]$ and zero outside). The remainder at time t is

$$\varepsilon(t) = x(t) + \sum_{i=1}^{p} a_i x(t-i). \tag{6.22}$$

Define the vector of remainders $\boldsymbol{\varepsilon}_{[\alpha,\beta]} = (\varepsilon(\alpha), \varepsilon(\alpha+1), \ldots, \varepsilon(\beta))^{\mathrm{T}}$ for an arbitrary time interval $[\alpha, \beta]$. Each equation of this type can be rewritten as

$$\boldsymbol{\varepsilon}_{[\alpha,\beta]} = X_{[\alpha,\beta]} \begin{pmatrix} 1 \\ \boldsymbol{a} \end{pmatrix},$$

where

$$X_{[\alpha,\beta]} = \begin{pmatrix} x(\alpha) & x(\alpha-1) & \cdots & x(\alpha-p) \\ x(\alpha+1) & x(\alpha) & \cdots & x(\alpha-p+1) \\ \vdots & \vdots & & \vdots \\ x(\beta-1) & x(\beta-2) & \cdots & x(\beta-p-1) \\ x(\beta) & x(\beta-1) & \cdots & x(\beta-p) \end{pmatrix}.$$

We determine the coefficients a_i by minimizing the sum of the squares of the remainders $\rho^{(\mathrm{a})}$ for the given sample, which can be written as

$$\rho^{(\mathrm{a})} = \sum_{t=0}^{T+p-1} \left|\varepsilon(t)\right|^2 = \left(1, \boldsymbol{a}^\dagger\right) R^{(\mathrm{a})} \begin{pmatrix} 1 \\ \boldsymbol{a} \end{pmatrix}, \quad R^{(\mathrm{a})} = X^\dagger_{[0,T+p-1]} X_{[0,T+p-1]}.$$

The sum also contains the values of the signal outside the interval $[0, T-1]$ according to the assumptions. The $(p+1) \times (p+1)$ matrix $R^{(\mathrm{a})}$ is Hermitian, with the

elements

$$R_{ij}^{(\mathrm{a})} = \sum_{t=0}^{T-(i-j)-1} x^*\big(t+(i-j)\big)x(t) \tag{6.23}$$

that depend only on the differences of indices i and j. The sum of the squares of the remainders is minimized by $\partial\rho^{(\mathrm{a})}/\partial\boldsymbol{a} = \mathbf{0}$, which can be written as

$$R^{(\mathrm{a})}\begin{pmatrix}1\\ \boldsymbol{a}\end{pmatrix} = \begin{pmatrix}\rho^{(\mathrm{a})}\\ \mathbf{0}\end{pmatrix}. \tag{6.24}$$

Note that $R^{(\mathrm{a})}$ is a Toeplitz matrix, so the system of equations is of the Yule–Walker type (6.21) and can be solved by the methods described on p. 312.

For short signal samples the biased auto-correlation estimate is not necessarily a good approximation of the auto-correlation of the process. In such cases we divide (6.24) on the left and on the right by T and replace the matrix elements of $R^{(\mathrm{a})}/T$ by the unbiased auto-correlation estimates.

Determining the Parameters by the Covariance Method Instead of minimizing the remainders, computed by (6.22), we can also opt to minimize the corresponding sum that involves only the values of the signal within the sample,

$$\rho^{(\mathrm{c})} = \sum_{t=p}^{T-1}\big|\varepsilon(t)\big|^2 = \big(1,\boldsymbol{a}^\dagger\big)R^{(\mathrm{c})}\begin{pmatrix}1\\ \boldsymbol{a}\end{pmatrix}, \quad R^{(\mathrm{c})} = X_{[p,T-1]}^\dagger X_{[p,T-1]}.$$

The equation for the minimum, $\partial\rho^{(\mathrm{c})}/\partial\boldsymbol{a} = \mathbf{0}$, can again be cast in the form (6.24),

$$R^{(\mathrm{c})}\begin{pmatrix}1\\ \boldsymbol{a}\end{pmatrix} = \begin{pmatrix}\rho^{(\mathrm{c})}\\ \mathbf{0}\end{pmatrix}, \qquad R_{ij}^{(\mathrm{c})} = \sum_{t=p}^{T-1} x^*(t-i)x(t-j). \tag{6.25}$$

The $(p+1)\times(p+1)$ matrix $R^{(\mathrm{c})}$ is still Hermitian, but its elements are not functions of differences of indices and it does not have a Toeplitz structure. The system (6.25) is therefore not of the Yule–Walker type, but it can still be solved in $\mathcal{O}(p^2)$ steps by using the CORVAR method described in [42]. A necessary (but not sufficient) condition for the non-singularity of $R^{(\mathrm{c})}$ is $p < T/2$. For details and other approaches to the minimization of remainders see [42, 44, 51].

Burg's Algorithm Burg's algorithm, known also as the maximum-entropy algorithm, can be used to compute the coefficients of a stable AR model, and is therefore primarily used in signal prediction (Sect. 6.7.2). If implemented correctly, it requires

$\mathcal{O}(3Tp) + \mathcal{O}(p^2)$ operations and $\mathcal{O}(3T + p)$ of memory [42, 45]. It can be found in the NUMERICAL RECIPES library [52] and in MATLAB [53].

Optimal Order There is no general rule to choose the order p at which the data in a signal sample would be described optimally. When p is increased, we are reducing the size of the remainder, which is measured by the average square $\langle \varepsilon^2 \rangle_t = \rho/(T + \mathcal{O}(1))$, or by the corresponding normalized quantity $\langle \varepsilon^2 \rangle_t / R_{XX}(0)$. In general, $\langle \varepsilon^2 \rangle_t$ initially decreases rapidly when p is increased, but at some p this decrease slows down. The value of p at which the transition between these two regimes occurs can be taken as optimal. For details see [44].

6.7.2 Use of AR Models

Auto-regression models are used in many ways, but the most well-known are *modeling* and *prediction*. In *modeling*, we use a known signal to compute the coefficients $\{a_i\}$ from which we infer the properties of the process that generated this signal. An example of modeling is the search for an approximation of the spectral density of a signal. We describe this in Sect. 6.7.3. In *linear prediction*, we determine the coefficients from the history of the signal $\{x(t-p), x(t-p+1), \ldots, x(t-1)\}$ and use them to predict its current value $x(t)$ and the forthcoming values $x(t+1), x(t+2), \ldots$ by using (6.19).

The Resonance Spectrum of the AR Model The dynamics of the auto-correlation R_{XX} defined by (6.20) can be calculated analytically. Let us define the characteristic polynomial of the AR model,

$$p_{\mathrm{AR}}(x) = x^p + \sum_{i=1}^{p} a_i x^{p-i} = \prod_{i=1}^{p} (x - z_i). \tag{6.26}$$

If we know the zeros $\{z_i\}_{i=1}^{p}$ of the polynomial p_{AR}, which we call the *resonances of the AR model*, the time evolution of the auto-correlation can be written as $R_{xx}(t) = \sum_{i=1}^{p} A_i z_i^t$, where A_i are constants. We determine the values of A_i from the initial conditions given by the derivatives $R_{XX}^{(i)}(0)$ with $i = 0, 1, \ldots, p-1$, or from p points of the auto-correlation at different times. If all zeros lie on the unit circle, we are referring to a *line spectral process* in which auto-correlations merely oscillate in time [46]. Similarly, the time evolution of the signal in the noiseless limit can be written as

$$x(t) = \sum_{i=1}^{p} B_i z_i^t,$$

where B_i are constants. This equation represents an auto-regressive approximation of the signal. When we use AR models, we often try to find a decomposition of precisely this kind, hoping that the signal-to-noise ratio is large.

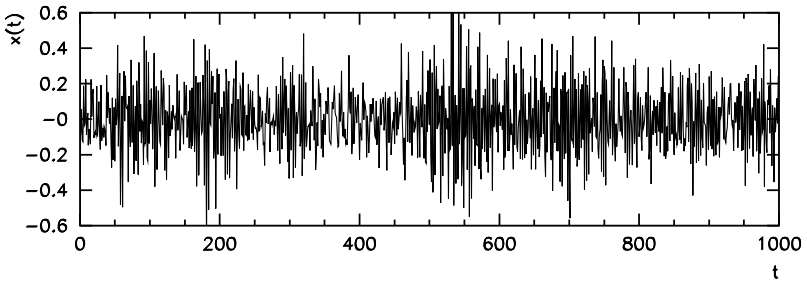


Fig. 6.12 The signal generated by the AR model (6.19) with the coefficients (6.27) and an admixture of Gaussian white noise ε with dispersion $\sigma_\varepsilon = 0.1$

If the coefficients a_i are real, the resonances z_i of the AR model occur in complex-conjugate pairs, if they reside away from the real axis in the complex plane. It happens that the zeros z_i fall outside of the unit circle (see Example below and Fig. 6.13 (right)). The signal that was generated by such an AR model then diverges as $\mathcal{O}((\max_i |z_i|)^t)$. Quite often, the zeros lying outside of the unit circle are just artifacts due to the presence of noise, which we strive to eliminate anyway, in particular when we wish to generate or predict long signal samples. In such cases we "fix" the problematic zeros by replacing z_i by $\hat{z}_i = z_i/|z_i|$ (which does not alter the phase) and computing the coefficients $\hat{a}_i$ of a slightly modified AR model with the characteristic polynomial

$$\prod_{i=1}^{p}(x - \hat{z}_i) = x^p + \sum_{i=1}^{p} \hat{a}_i x^{p-i}.$$

Example Assume that a process is described by the AR model with coefficients

$$\{a_i\}_{i=1}^{p} = \{0.1, 0.2, -0.3, 0.4, 0.5, 0.1, 0.2, -0.2, 0.5, -0.1\} \tag{6.27}$$

and that $\varepsilon(t)$ is white noise with $\sigma_\varepsilon = 0.1$. We use (6.19) to generate a signal sample $\{x(t)\}_{t=0}^{T-1}$ of length $T = 1000$, where the first $p = 10$ values are used as the initial condition of the iteration. A possible realization of the signal is shown in Fig. 6.12. The average square of the remainder in the model is shown in Fig. 6.13 (left). The spectrum of the resonances shown in Fig. 6.13 (right) reveals that this AR model is unstable, as three out of ten resonances lie outside of the unit circle.

As an exercise, let us study the generated signal by the AR model with the coefficients $\{\tilde{a}_i\}_{i=1}^{\tilde{p}}$ for different orders $\tilde{p}$ and by applying the covariance method. With increasing $\tilde{p}$ the average square of the remainder $\langle \varepsilon^2 \rangle_t$ decreases until it stabilizes at $\mathcal{O}(\sigma_\varepsilon^2)$. This occurs at $\tilde{p} = 10$, which is the order of the AR model used. For $\tilde{p} > p$, ε resembles the noise that was used in the generation of the signal. With

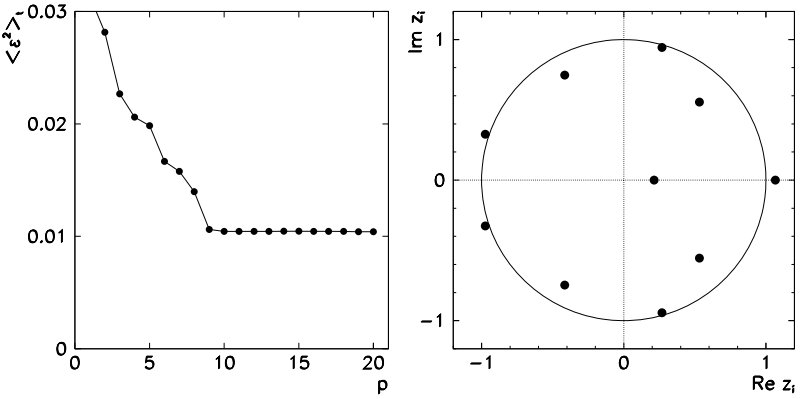


Fig. 6.13 [*Left*] The average square of the remainder in the AR model with respect to the sample values, as a function of order p. [*Right*] The resonance spectrum of the model used to generate the signal in Fig. 6.12. Three resonances lie outside of the unit circle, thus the model is unstable

increasing $\tilde{p}$ the coefficients $\tilde{a}_i$ approach a_i of the original model:

$\tilde{p}$	$\tilde{a}_1$	$\tilde{a}_2$	$\tilde{a}_3$	$\tilde{a}_4$	$\tilde{a}_5$	$\tilde{a}_6$	$\tilde{a}_7$	$\tilde{a}_8$	$\tilde{a}_9$	$\tilde{a}_{10}$
1	0.343									
2	0.453	0.321								
3	0.310	0.120	−0.442							
6	0.312	0.225	−0.443	0.374	0.288	0.400				
8	0.326	0.132	−0.472	0.179	0.462	0.374	0.095	−0.340		
10	0.095	0.202	−0.310	0.386	0.477	0.089	0.199	−0.202	0.470	−0.131

The approach of $\tilde{a}_i$ to the original coefficients a_i depends on the sample size and the noise intensity. Convergence is guaranteed only for infinite WSS signals. The coefficients $\tilde{a}_i$ for $i \geq \tilde{p} > p$ are not zero (for $\tilde{p} = 11$ we get $\{\tilde{a}_i\}_{i=1}^{10} = \{0.097, 0.194, -0.306, 0.383, 0.475, 0.080, 0.192, -0.197, 0.467, -0.133\}$, then $\tilde{a}_{11} = -0.018$) but they are random and on the order of $\mathcal{O}(\sigma_\varepsilon)$.

6.7.3 Estimate of the Fourier Spectrum

The power spectral density of a discrete-time signal $x(t)$, $t \in \mathbb{Z}$, is defined as

$$P_x(\omega) = \lim_{T\to\infty} \frac{1}{2T+1} \left| \sum_{t=-T}^{T} x(t) \mathrm{e}^{-\mathrm{i}\omega t} \right|^2, \quad -\pi \leq \omega < \pi.$$

(See [51, 54].) If x was obtained by discretizing a continuous signal with the step Δt, ω is related to frequency ν as $\omega = 2\pi\nu\Delta t$. The relation between the time auto-

correlation of the signal R_{xx} and the spectral density is

$$P_x(\omega) = \sum_{t=-\infty}^{\infty} R_{xx}(t)\mathrm{e}^{-\mathrm{i}\omega t}.$$

For samples of length T we only know the sample auto-correlation $\mathcal{R}_{\hat{x}\hat{x}}(t)$ and then the formula above can be rewritten as a *correlogram*

$$P_{\hat{x}}(\omega) = \sum_{t=-L}^{L} \mathcal{R}_{\hat{x}\hat{x}}(t)\mathrm{e}^{-\mathrm{i}\omega t} = \mathcal{R}_{\hat{x}\hat{x}}(0) + 2\sum_{t=1}^{L} \mathcal{R}_{\hat{x}\hat{x}}(t)\cos\omega t, \tag{6.28}$$

where the sample auto-correlation is used up to the maximum shift $L \leq T-1$. A correlogram is an approximation of the spectral density of the complete signal. If we assume that the signal is a realization of a WSS process, we can find the averages of the sample spectral densities for both defined approximations [42]:

$$\text{unbiased:} \quad \left\langle P_{\hat{x}}(\omega)\right\rangle = P_x(\omega) * D_L(\omega), \tag{6.29}$$

$$\text{biased:} \quad \left\langle P_{\hat{x}}(\omega)\right\rangle = P_x(\omega) * \frac{2}{L} D_L^2(\omega/2), \tag{6.30}$$

where $*$ denotes convolution and $D_L(\omega) = \mathrm{e}^{-\mathrm{i}2\pi\omega(L-1)}\sin(\pi\omega L)/\sin(\pi\omega)$ is the Dirichlet kernel. The sample spectral densities are biased approximations of the true spectral density $P_x(\omega)$, since the averages $\langle P_{\hat{x}}(\omega)\rangle$ are its convolutions with the kernel. With increasing L the kernel widens and it is sensible to use $L \ll T$ (at most $L \approx T/10$). For $L = T-1$ the correlogram is equal to the power spectral density of the discrete signal as computed by the DFT,

$$\tilde{P}_x(\omega) = \frac{1}{T}\left|\sum_{t=0}^{T-1} x(t)\mathrm{e}^{-\mathrm{i}\omega t}\right|^2, \tag{6.31}$$

but this is statistically unstable. The correlogram with the maximum shift of $L < T/2$ for a sample of size T can be computed at the points $\{\omega_k = 2\pi k/T\}_{k=0}^{T-1}$ by using the DFT. We arrange the values of the sample auto-correlation $\mathcal{R}_{\hat{x}\hat{x}}$ in an array of length T, $\mathcal{R} = \{\mathcal{R}_{\hat{x}\hat{x}}(0), \ldots, \mathcal{R}_{\hat{x}\hat{x}}(L), 0, \ldots, 0, \mathcal{R}_{\hat{x}\hat{x}}(-L), \ldots, \mathcal{R}_{\hat{x}\hat{x}}(-1)\}$. When we apply DFT to it, we obtain the spectral density

$$\left\{P_{\hat{x}}(\omega_k)\right\}_{k=0}^{T-1} = T\mathcal{F}_T[\mathcal{R}].$$

For the computation we should chose L such that $P_{\hat{x}}(\omega) \geq 0$, which is fulfilled in the case of auto-correlations, but does not necessarily apply to *sample* auto-correlations. A comparison of the sample spectral density (computed by DFT) and the correlogram is shown in Fig. 6.14. We see that the correlogram is less sensitive to the presence of noise. Details on estimates of spectral densities of signals are given in [42].

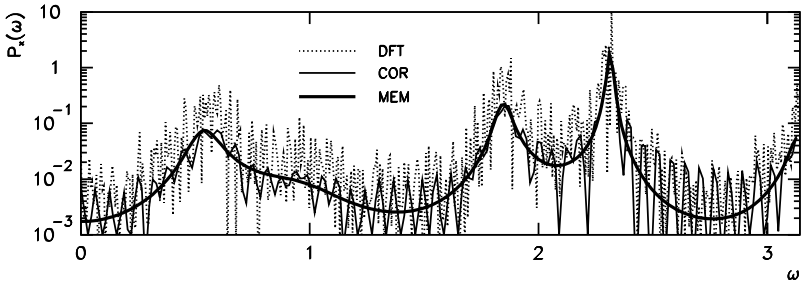


Fig. 6.14 Three approximations of the spectral density of the signal x from the Example on p. 315. The sample has a length of $T = 10^4$. Shown are the curves of the sample spectral density (DFT, (6.31)), the correlogram (COR, (6.28)) with $L = T/10$, and the maximum-entropy approximation at order $p = 10$ (MEM, (6.32))

Assume that we know the AR model of order p with the coefficients $\{a_i\}_{i=1}^p$ for a signal x. Let us look at this computation in the reverse direction: we use a linear transformation of the signal to generate the remainder $\varepsilon(t)$,

$$\varepsilon(t) = x(t) + \sum_{i=1}^{p} a_i x(t-i).$$

We apply the Fourier transformation $\mathcal{F}[f](\omega) = \sum_{t\in\mathbb{Z}} f(t)\mathrm{e}^{-\mathrm{i}\omega t}$ to both sides of the equation and get $\mathcal{F}[\varepsilon](\omega) = \mathcal{F}[x](\omega)\exp(-\mathrm{i}\omega p)\, p_{\mathrm{AR}}(\exp(\mathrm{i}\omega))$, where p_{AR} is the characteristic polynomial (6.26). The power spectral density of the signal is

$$P_x(\omega) = \frac{P_\varepsilon(\omega)}{|p_{\mathrm{AR}}(\exp(\mathrm{i}\omega))|^2}, \tag{6.32}$$

where $P_\varepsilon(\omega)$ is the power spectral density of the remainder ε. We almost invariably assume that the remainder is white noise with variance σ_ε^2, so that $P_\varepsilon(\omega) = \sigma_\varepsilon^2$. This assumption implies that this process has the maximum possible entropy among all processes with the same auto-correlation (the *maximum-entropy principle*). The estimate (6.32) is therefore known as the *maximum-entropy spectrum estimator* (MESE), while the method to find this approximation is called the *maximum-entropy method* (MEM) [46, 55].

Figure 6.14 makes it clear that the MEM estimate for orders $p \ll T$ is smoother than either the sample spectral density or the correlogram. The MEM estimate is also resilient to noise, until p is much smaller than T and below the limit at which the remainder starts to resemble noise. When p is increased further, aliasing effects sneak into the MEM estimate, recognizable as contributions oscillating with a period of $2\pi/p$ around the expected spectral density (Fig. 6.15).

We see from this figure that by increasing p we are revealing individual resonances in the signal. When all essential resonances have been accounted for, the remainder in the AR model resembles noise. If we know that the spectrum of the

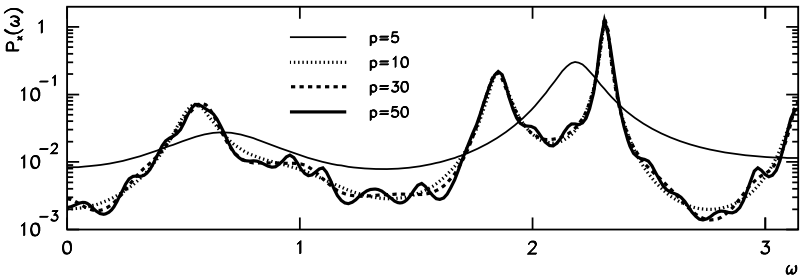


Fig. 6.15 Maximum-entropy spectrum approximations for different orders p of the AR models. The coefficients of the models were computed by the least-squares method for the data from Fig. 6.12. The curve of the exact spectral density (which corresponds to the model used to generate the sample) is almost identical to the curve at $p = 10$

signal should contain k prominent frequencies, it makes sense to choose the order $p = 2k$, since resonances occur in complex-conjugate pairs.

We recommend the use of the MEM spectral approximation when the signal can be described by an AR model of order p that is much smaller than the size of the sample. Moreover, p should be small enough that the methods we use to compute the coefficients remain stable. In most cases $p < 40$ suffices.

6.8 Independent Component Analysis ⋆

Independent component analysis (ICA) is a multivariate analysis technique from the class of latent-variable methods (another representative is the factor analysis discussed in Sect. 5.11). The classical problem that can be solved by ICA is the decomposition of an unknown mixture of signals to its independent components. For example, we measure the signals $x_1(t)$, $x_2(t)$, and $x_3(t)$ of three microphones recording the speech of two persons (sources) $s_1(t)$ and $s_2(t)$. We assume that the signals are linear combinations of the sources,

$$\begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{pmatrix} \begin{pmatrix} s_1(t) \\ s_2(t) \end{pmatrix},$$

where neither the sources s_i nor the matrix elements A_{ij} are known. A generalization of this example is the famous *cocktail-party problem*, in which we wish to reconstruct the signals of r speakers based on n measurements of signals from m microphones arranged in a room. The components of the vector representing the sources are the latent variables mentioned above.

A very similar problem is illustrated in Fig. 6.16 (see also Problem 6.9.4 and Fig. 6.22). The figure shows eight traces of an electro-cardiogram (ECG) of a pregnant woman recorded at various places on the thorax and the abdomen. In some

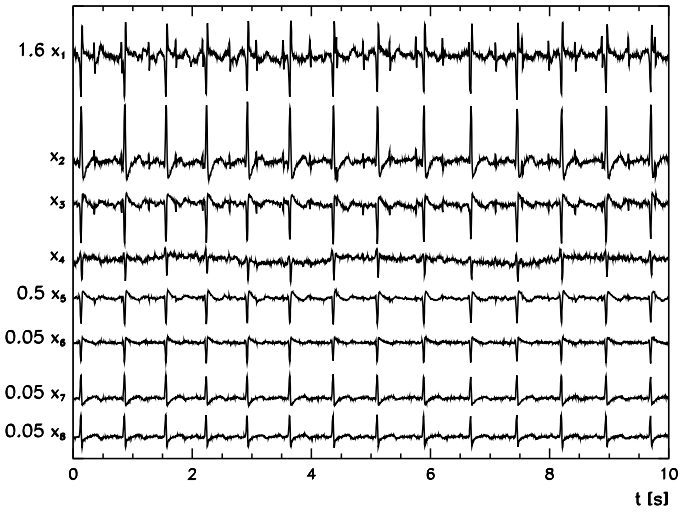


Fig. 6.16 Electro-cardiogram of a pregnant woman. The relatively weak and noisy signal of the child's heartbeat, visible in signals x_1, x_2, and x_3, mixes with the signal of the mother's heartbeat (Example adapted from [56] based on data from [57])

signals (x_3, x_2 and, above all, x_1) we can also see the child's heartbeat which has a higher frequency than the mother's heartbeat, but its amplitudes are smaller and very noisy. The task of ICA is to extract the original signal *sources* and to explain the mixing of these sources into the measured signals. The procedure should be done such that the computed sources are mutually as independent as possible.

Here we discuss only *static linear independent component analysis*, for which we assume that each vector of correlated measurements $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)^{\mathrm{T}}$ is generated by *linear* mixing of *independent sources* $\boldsymbol{s} = (s_1, s_2, \ldots, s_r)^{\mathrm{T}}$, thus

$$\boldsymbol{x} = A\boldsymbol{s}, \tag{6.33}$$

where $A \in \mathbb{R}^{m\times r}$ is the *mixing matrix* which does not depend on time. Usually the number of sources is smaller than the number of measured signals, $r \leq m$. We neglect measurement errors that, in principle, could be added to the right side of (6.33). We assume that noise is already contained in the sources $\boldsymbol{s}$: we are performing a *noiseless ICA*.

The independence of the sources is expressed by the statement that their joint probability density factorizes as $p(s_1, s_2, \ldots, s_r) = p_1(s_1)p_2(s_2)\cdots p_r(s_r)$ and therefore also $\langle f(s_i)g(s_j)\rangle = \langle f(s_i)\rangle\langle g(s_j)\rangle$ for arbitrary functions f and g. A pair of statistically independent variables $(s_i, s_{j\neq i})$ is uncorrelated and has the covariance $\mathrm{cov}(s_i, s_j) = 0$; the inverse is not necessarily true. This distinguishes ICA from the decorrelation of variables by PCA (Sect. 5.7), in which principal components of multivariate data are determined by maximizing their variances and minimizing their mutual correlations.

In the following we assume that the average of an individual vector of sources is zero, $\langle \boldsymbol{s} \rangle = \boldsymbol{0}$, and that the corresponding covariance matrix is diagonal, $\Sigma_{ss} = \mathrm{cov}(\boldsymbol{s}, \boldsymbol{s}) = \boldsymbol{s}\boldsymbol{s}^{\mathrm{T}} = I$. Other than that, the components s_i may have any probability distribution except Gaussian. The requirement for a non-Gaussian character of the distributions is essential, otherwise the ICA method allows for the determination of the independent components only up to an orthogonal transformation. At most one component of the signal is allowed to be Gaussian: for a detailed explanation see Sect. 15.3 in [56] and [58, 59].

6.8.1 *Estimate of the Separation Matrix and the FastICA Algorithm*

According to the model (6.33) with the matrix A of full rank, a *separation* or *unmixing matrix* W exists by which the sources $\boldsymbol{s}$ can be exactly reconstructed from the measured signals $\boldsymbol{x}$:

$$\boldsymbol{s} = W\boldsymbol{x}, \quad W = \left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_r)^{\mathrm{T}} \tag{6.34}$$

(see (3.10)). But in practice, neither the matrix A nor the vector of the sources $\boldsymbol{s}$ in the model (6.33) are known. In order to reconstruct $\boldsymbol{s}$, we therefore attempt to find an estimate of the separation matrix W. We resort to several additional requirements that should be fulfilled by the one-dimensional projections of the measured signals,

$$y_i = \boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x}, \quad i = 1, 2, \ldots, r.$$

If the vector $\boldsymbol{w}_i$ is equal to some row of the generalized inverse W of A (see (6.34)), the projection y_i is already one of the independent components, $y_i = s_i$. Since A (or W) is unknown, we try to find the vectors $\boldsymbol{w}_i$ such that the probability distribution of the projection y_i will be *as non-Gaussian as possible*. Seeking such vectors is the key part of the independent component analysis.

The deviation of the distribution of a continuous random variable y from the Gaussian distribution can be measured by the entropy

$$H(y) = -\int p(y) \log p(y)\, \mathrm{d}y,$$

where $p(y)$ is the probability density of y. Of all random variables with the same variance the Gaussian random variable has the maximum entropy [26]. The value of the entropy can therefore be used as a tool to gauge the similarity of some unknown distribution to the Gaussian distribution. Computationally it is more convenient to work with *negentropy*

$$I(y) = H(y_{\mathrm{Gauss}}) - H(y), \quad y_{\mathrm{Gauss}} \sim N(0, 1),$$

Table 6.1 Typical choices for the function G and its derivatives appearing in the approximation of negentropy (6.35) and in the FastICA algorithm to determine the independent components. For general use we recommend the functions (1) with the parameter $\alpha = 1$. For signals with pronounced outliers or super-Gaussian probability distributions, we recommend the functions (2)

	$G(y)$	$G'(y)$	$G''(y)$	Remark
(1)	$\frac{1}{\alpha}\log\cosh\alpha y$	$\tanh\alpha y$	$\alpha(1-\tanh^2\alpha y)$	$1 \le \alpha \le 2$
(2)	$-\mathrm{e}^{-y^2/2}$	$y\mathrm{e}^{-y^2/2}$	$(1-y^2)\mathrm{e}^{-y^2/2}$	

which is a non-negative quantity and is equal to zero only in the case that both y_{Gauss} and y are distributed normally with zero average and unit variance.

For an exact calculation of $I(y)$ we need the probability density $p(y)$, which we almost never know in practice. We therefore use approximations of the negentropy, most often [58]

$$I(y) \approx \left[\langle G(y)\rangle - \langle G(y_{\text{Gauss}})\rangle\right]^2, \tag{6.35}$$

where $G(y)$ is a non-quadratic function of y (two popular choices are given in Table 6.1). We compute the estimate y_i for a single independent component s_i by finding a vector $\boldsymbol{w}_i$ such that the projection $y_i = \boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x}$ will have maximum negentropy $I(y_i)$ (then its probability density will least resemble the Gaussian). We achieve the decomposition to independent components when we ultimately find all vectors $\boldsymbol{w}_i$ $(1 \le i \le r)$ corresponding to local maxima of $I(y_i)$. Since the independent components s_i (or the estimates y_i for them) are uncorrelated, they can be computed individually. Finally, the vectors $\boldsymbol{w}_i$ are arranged in the matrix W and we use (6.34) to compute the independent components $\boldsymbol{s}$.

6.8.2 The FastICA Algorithm

The independent components of the signals $\boldsymbol{x}_i = (x_1, x_2, \ldots, x_m)_i^{\mathrm{T}}$, which are measured at n consecutive times $(0 \le i \le n-1)$, can be determined by the FastICA algorithm described in the following. The measured signals are first arranged in the matrix

$$X = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1})^{\mathrm{T}} \in \mathbb{R}^{n\times m},$$

in which each of the n rows represents one m-variate data entry (for example, voltages on m microphones at time $t = i\Delta t$). We compute the mean $\overline{\boldsymbol{x}}$ by (5.46) and the covariance matrix Σ_{xx} by (5.52). We diagonalize Σ_{xx} as $\Sigma_{xx} = U\Lambda U^{\mathrm{T}}$ and thus obtain the orthogonal matrix U and the diagonal matrix Λ. The data are then transformed as

$$\boldsymbol{x}_i \longleftarrow \Lambda^{-1/2}U^{\mathrm{T}}(\boldsymbol{x}_i - \overline{\boldsymbol{x}}), \quad i = 0, 1, \ldots, n-1. \tag{6.36}$$

The transformation (6.36) is known as *data whitening* and is equivalent to a decomposition of the standardized data to their principal components (see Sects. 5.7.2

and 5.7.3). Whitening removes any trace of scale or correlation from the data. We then follow the algorithm [60]:

1. Choose the number of independent components r you wish to determine.
2. Randomly initialize the vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_r$ $(\boldsymbol{w}_k \in \mathbb{R}^m)$ normalized to $\|\boldsymbol{w}_k\|_2 = 1$, and arrange them in the matrix $W = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_r)^{\mathrm{T}} \in \mathbb{R}^{r \times m}$.
3. Perform a symmetric orthogonalization of W,

$$W \longleftarrow \left(W W^{\mathrm{T}}\right)^{-1/2} W.$$

   This step ensures that all previously found $\boldsymbol{w}_k$ are mutually orthogonal. The square root of the matrix is computed as described in Appendix A.7.
4. For each $k = 1, 2, \ldots, r$ compute new vectors

$$\boldsymbol{w}_k \longleftarrow \frac{1}{n}\left\{\sum_{i=0}^{n-1} \boldsymbol{x}_i G'\left(\boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}_i\right) - \boldsymbol{w}_k \sum_{i=0}^{n-1} G''\left(\boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}_i\right)\right\}, \tag{6.37}$$

   where the function pair $G'(y)$ and $G''(y)$ can be chosen from Table 6.1. This step is the crucial part of the algorithm ensuring that the iteration leads to the fixed point corresponding to the maximum of negentropy (6.35). Normalize the obtained vectors to unit length, $\boldsymbol{w}_k \longleftarrow \boldsymbol{w}_k / \|\boldsymbol{w}_k\|_2$.
5. Repeat items 3 and 4 until convergence is achieved in all components of the vectors $\boldsymbol{w}_k$. A good measure of convergence is the configuration in which the direction of the vector $\boldsymbol{w}_k$ in consecutive iterations (ν) and $(\nu - 1)$ no longer changes, i.e. when the absolute value of the scalar product $|\boldsymbol{w}_k^{(\nu)\mathrm{T}} \boldsymbol{w}_k^{(\nu-1)}|$ is close to unity. Typically a few times ten iterations are needed (see Fig. 6.17).
6. The independent components are the rows of the matrix $S = W X^{\mathrm{T}} \in \mathbb{R}^{r \times n}$.

The FastICA algorithm allows us to compute the matrix of independent components (sources) S in which the sources appear to be arranged arbitrarily. Namely, any permutation P of the source components $\boldsymbol{s}$ in (6.33) implies just a different mixing matrix, $\boldsymbol{x} = (AP^{-1})P\boldsymbol{s}$. From the viewpoint of the ICA method, the vectors $\boldsymbol{s}$ and $P\boldsymbol{s}$ are indistinguishable.

Moreover, the signals S determined by ICA are given up to a multiplicative constant: we may choose any constant c_j to divide the source s_j and multiply the jth row of the mixing matrix A without modifying the product $A\boldsymbol{s}$. Even vectors of opposite directions ($\boldsymbol{w}_k$ and $-\boldsymbol{w}_k$ as the rows of W), which frequently occur during the iterations of the FastICA algorithm, are therefore equivalent.

6.8.3 Stabilization of the FastICA Algorithm

The iteration step (6.37) follows from the Newton's method to search for the maximum of negentropy, so occasionally we may encounter convergence problems. In

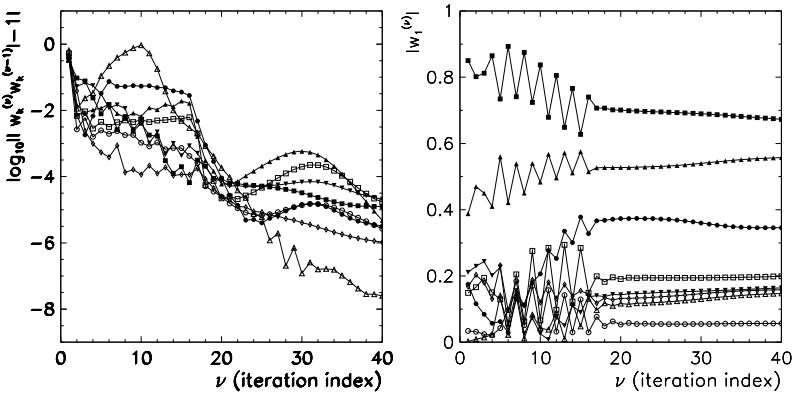


Fig. 6.17 Typical convergence in the FastICA algorithm applied to the relatively complex signals from Fig. 6.16. [*Left*] The difference of the scalar product $|\boldsymbol{w}_k^{(\nu)\mathrm{T}}\boldsymbol{w}_k^{(\nu-1)}|$ from 1 in subsequent iterations (ν) in the case of eight independent components ($k = 1, 2, \ldots, 8$). [*Right*] The convergence of eight components of the vector $\boldsymbol{w}_1$ in consecutive iterations (ν)

such cases the authors [60] recommend a stabilized version of the step (6.37), which is

$$\boldsymbol{w}_k \longleftarrow \boldsymbol{w}_k - \mu\left[\frac{1}{n}\sum_{i=0}^{n-1}\boldsymbol{x}_i G'\left(\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_i\right) - \beta\boldsymbol{w}_k\right]\left[\frac{1}{n}\sum_{i=0}^{n-1}G''\left(\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_i\right) - \beta\right]^{-1},$$

where

$$\beta = \frac{1}{n}\sum_{i=0}^{n-1}\left(\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_i\right)G''\left(\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_i\right),$$

and μ is a parameter controlling the stability of the iteration. The iteration can be stabilized by using the values of μ much smaller than one, for example, $\mu \approx 0.1$ or $\mu \approx 0.01$. Details of other possible improvements of the algorithm can be found in [60].

6.9 Problems

6.9.1 Logistic Map

In autonomous dynamical systems the time evolution of the points in phase space is defined by the map $\phi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, where the function $x(t) = \phi(t - t_0, x(t_0))$ describes the trajectory of the system and represents its time evolution from a known initial state $x(0)$. The paradigmatic example of a discrete system is the logistic map

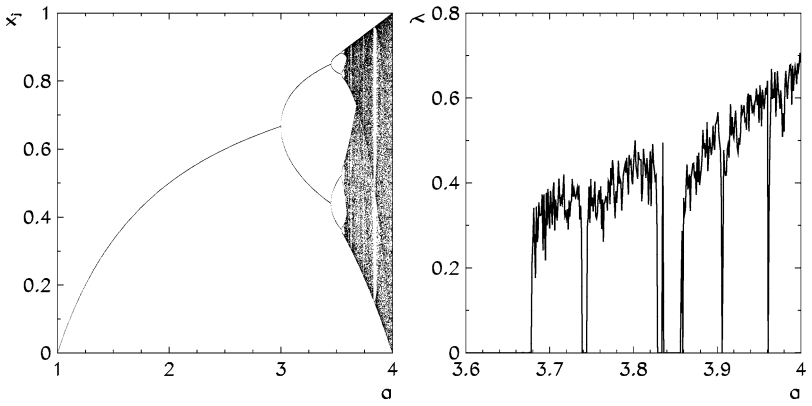


Fig. 6.18 Logistic map. [*Left*] Bifurcation diagram (dependence of the value from recurrence (6.38) on the parameter a after sufficient transition time). [*Right*] Lyapunov exponent in dependence of a

[25], for which one time step is given by the recurrence

$$x(t+1) = ax(t)\big(1 - x(t)\big), \quad a \in [1, 4], \tag{6.38}$$

and the phase space is $[0, 1]$. A periodic orbit with period T is a set of points with the property $x(t+T) = x(t)$. By observing the trajectories in phase space we can only find stable periodic orbits, as these attract the trajectories from their neighborhood. The occurrence of stable periodic orbits in the logistic map has a rich dependence on the parameter a, as demonstrated by the diagram in Fig. 6.18 (left). At given a we take an arbitrary initial point in phase space and observe the trajectory after a sufficiently long transition time. If one period is destroyed and another is formed when the value of a changes, we are referring to a *bifurcation* of the dynamics. The sequences of distances between the points at which bifurcations occur possess intriguing universal properties [61, 62].

Dynamical systems whose trajectories with *almost equal* initial points exponentially diverge with time, are called *chaotic*. The chaoticity of the system or its sensitivity to the precision of initial conditions can be measured by the Lyapunov exponent λ defined as

$$\lambda = \lim_{\substack{t\to\infty \\ \varepsilon\to 0}} \frac{1}{t} \log \frac{|\phi(t, x(0)+\varepsilon) - \phi(t, x(0))|}{|\varepsilon|}.$$

The Lyapunov exponent is closely related to entropy formation in the dynamical system. The chaoticity of the logistic map rapidly changes with the parameter a and the coefficient λ is equal to zero when stable periodic orbits occur, as shown in Fig. 6.18 (right). At $a = 4$ even the exact value is known: $\lambda = \log 2$.

⊙ For different values of a in the vicinity of 3.702, 3.74, 3.961, and 3.9778 compute $N = 1024$ values of the sequence $\{x(t)\}_{t=0}^{N-1}$ from the logistic map (6.38) with the chosen initial point $x(0) \in [0, 1]$. This sequence is your signal. Compare the

single-sided power spectral densities (PSD) of this signal computed by the Fourier transformation and by the method of maximum entropy. In the ranges of a corresponding to chaotic dynamics the spectrum has no prominent peaks and is practically continuous, which is one of the landmarks of chaos.

For individual values of a, compute the auto-correlations

$$R_{xx}(\tau) = \big\langle x(t)x(\tau+t)\big\rangle_t - \big(\big\langle x(t)\big\rangle_t\big)^2$$

of the sequences $\{x(t)\}_{i=0}^{N-1}$ and determine an approximation of the correlation length ξ as a function of a. Choose N large enough that $R_{xx}(\tau)/R_{xx}(0)$ will be precise to three digits. Assume that the auto-correlation has the form

$$R_{xx}(\tau) = A\mathrm{e}^{-|\tau|/\xi}.$$

It can be shown that

$$\xi = \log\left(\frac{|S_1^2 - S_2|}{S_1^2 + S_2}\right)^{-1}, \quad S_n = \sum_{t=0}^{\infty} R_{xx}(t)^n.$$

Systems in which all correlations between quantities converge to zero at long times, are said to *mix the space*, which is one of the key properties permitting a statistical analysis of dynamics.

6.9.2 Diffusion and Chaos in the Standard Map

In some Hamiltonian systems, continuous dynamics can be translated to discrete dynamics. This applies in particular to systems on which we act with a periodic external force in the form of short pulses. The point (p, q) of the trajectory of the system (position and momentum) at time $t+1$ is related to the point at time t when the system receives the pulse, by the equations

$$\begin{aligned} p(t+1) &= p(t) + F\big(q(t), p(t)\big), \\ q(t+1) &= q(t) + G\big(p(t+1)\big), \end{aligned}$$

where F is the force pulse and G represents the dynamics between the pulses. In such systems a very clear connection between diffusion and auto-correlation exists. The diffusion coefficient is defined as

$$D = \lim_{t\to\infty} \frac{1}{2t}\big\langle\big[q(t) - q(0)\big]^2\big\rangle,$$

where $\langle\cdot\rangle$ denotes the phase average (average over initial points $(q(0), p(0))$ distributed uniformly in the region of phase space which is invariant with respect to the dynamics). The time auto-correlation function G is defined as

$$R_{GG}(\tau) = \big\langle G\big(p(\tau)\big)G\big(p(0)\big)\big\rangle - \big\langle G(p(0)\big\rangle^2.$$

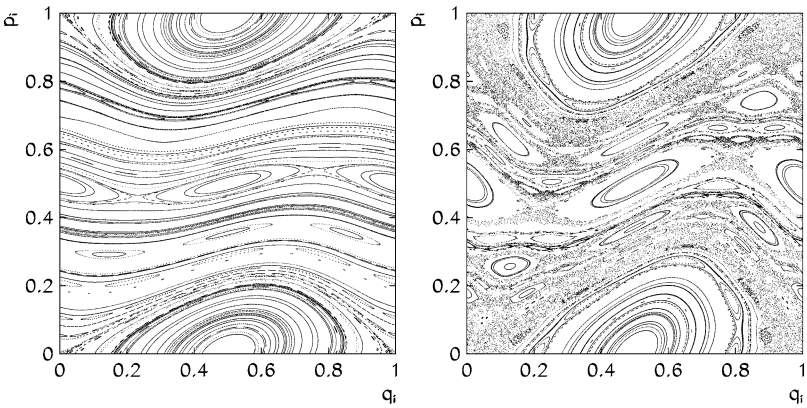


Fig. 6.19 Phase portraits of the standard map for different values of the parameter K. [*Left*] $K = 0.5$. [*Right*] $K = 1.0$

By assuming that this system sufficiently quickly mixes the phase space, a relation between the diffusion coefficient and auto-correlation can be derived:

$$D = R_{GG}(0) + 2\sum_{k=1}^{\infty} R_{GG}(k). \tag{6.39}$$

⊙ Analyze the dynamical system of the standard (Chirik's) map

$$p(t+1) = p(t) + \frac{K}{2\pi}\sin\bigl(2\pi q(t)\bigr) \mod 1,$$
$$q(t+1) = q(t) + p(t+1) \mod 1,$$

which is defined on the torus $(q(t), p(t)) \in [0, 1)^2$. This map played a major role in the development of the theory of classical and quantum chaos [63]. Examples of phase portraits obtained by propagating a number of uniformly distributed points in the case of $K = 0.5$ and $K = 1.0$ are shown in Fig. 6.19. By increasing K above ≈ 0.97 the chaotic region of the phase space, where the points are distributed uniformly, becomes ever larger, until diffusion occurs throughout.

Compute the time auto-correlation $R_{pp}(\tau) = \langle p(\tau + t)p(t)\rangle_t - \langle p(t)\rangle_t^2$ of the momentum samples $\{p(t)\}_{t=0}^{N-1}$ for the values $K = 1$, 2, 5, and 10, with the initial point $(q(0), p(0))$ in the chaotic region of the phase space. The expected absolute error of $R_{pp}(\tau)$ should be less than 10^{-3}. If the auto-correlation decays rapidly, show it in the rescaled form $R_{pp}(\tau)/R_{pp}(0)$ in logarithmic scale. If you observe an exponential fall-off of the auto-correlation,

$$\bigl|R_{pp}(\tau)\bigr| \sim \mathrm{e}^{-\tau/\xi},$$

estimate the correlation length ξ which represents the time after which the trajectories become statistically independent. To compute the auto-correlation use the FFT algorithm (Sect. 4.2.5).

⊕ Observe the dependence of the diffusion coefficient D on the parameter K from the interval $[2, 20]$. Compute D by summing the auto-correlations of momentum (i.e. by using (6.39) in which R_{GG} is replaced by R_{pp}). The initial points of the map should be located in the chaotic region of the phase space. An analytic approximation for the diffusion coefficient can be found in [63].

6.9.3 Phase Transitions in the Two-Dimensional Ising Model

Some properties of systems consisting of magnetic dipoles with local interactions can be nicely explained by the *Ising model*. This model assumes that the dipoles are arranged in the nodes of a two-dimensional mesh and that the spins s_i are either "up" ($s_i = 1$) or "down" ($s_i = -1$). The Hamiltonian (the energy) of such a system in the presence of an external magnetic field H can be written in the form

$$E = -J \sum_{\langle i,j \rangle} s_i s_j - H \sum_i s_i, \tag{6.40}$$

where the indices i and j represent the coordinates of the spins on the mesh (in two dimensions $i \in \mathbb{N}_0^2$). The first sum in (6.40) runs over neighboring points only. The parameter J determines how the spins interact. For $J > 0$ the adjacent dipoles tend to point in the same direction, while for $J < 0$ they tend in the opposite directions. In the following we set $J > 0$. The total magnetization of the system is

$$M = \sum_i s_i .$$

At different temperatures the system may reside in ferromagnetic or paramagnetic phase. The temperature T_c of the phase transition between these phases in the absence of external field ($H = 0$) is given by the equation $\sinh(2J/(k_B T_c)) = 1$, which has the solution $T_c \approx 2.269185\, J/k_B$.

⊙ At the book's website you can find the data for the thermodynamic equilibrium state of spins in the two-dimensional Ising model at various temperatures. A few examples are shown in Fig. 6.20. Compute the auto-correlation function of the orientation of spins on a $N \times N$ mesh,

$$R_{ss}(i, j) = \frac{1}{N^2} \sum_{k,l} s_{k+i,l+j} s_{k,l},$$

where we assume periodic boundary conditions. Average the auto-correlation over the neighborhood of the points with approximately the same radius r. Assume that the auto-correlation has the form

$$\left| R_{ss}(i, j) \right| \sim C \mathrm{e}^{-r/L}, \quad r = \sqrt{i^2 + j^2},$$

and estimate the correlation length L. Show L as a function of temperature: you should observe a strong increase in L near the critical temperature T_c.

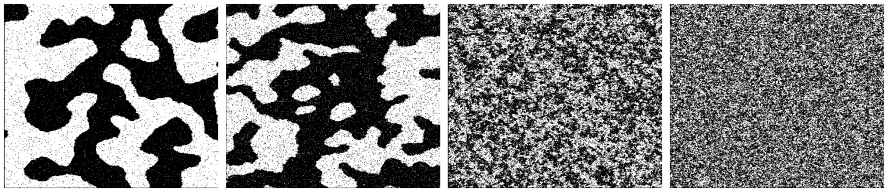


Fig. 6.20 A two-dimensional spin array of size 4096×4096 in the Ising model with $J = 1$ at equilibrium at various temperatures T. Black and white dots denote spins $s_i = 1$ and $s_i = -1$, respectively. Shown from left to right are images of the ferromagnetic phase at temperatures $T = 1.9$ and 2.1, followed by the paramagnetic phase at $T = 2.3$ and 2.4, all in units of J/k_B

6.9.4 Independent Component Analysis

Independent component analysis (ICA) is a method that can be applied to a set of measured signals $\boldsymbol{x}_j(t)$ in order to determine independent components (sources) $\boldsymbol{s}_j(t)$ that caused these signals, as well as the nature of the mixing of sources. We assume that the measured signals and their sources are linearly related,

$$\boldsymbol{x}(t) = A\boldsymbol{s}(t),$$

where A is the mixing matrix (see Sect. 6.8). At each instant t the sources $\boldsymbol{s}$ (for example, speakers in a room) and the measured signals $\boldsymbol{x}$ (for example, voltages on the microphones) have several components.

⊙ In the first part of the Problem we assume that the mixing matrix is known. Suppose that we have four sources:

$$\begin{aligned} s_{1i} &= \sin(10\pi i/n), \\ s_{2i} &= \exp\bigl[-10(i-n/2)^2/n^2\bigr] + 0.2\bigl[\mathcal{R}(0,1) - 0.5\bigr], \\ s_{3i} &= \sin^3(40\pi i/n)\exp(-1.5i/n), \\ s_{4i} &= \sin\bigl(100i^2/n^2\bigr), \end{aligned} \tag{6.41}$$

shown in Fig. 6.21 (left). The index i measures the time, $t = i\Delta t$ ($i = 0, 1, \ldots, n-1$) and $n = 1000$. We use $\mathcal{R}(0,1)$ to denote a uniformly distributed random number from the interval $[0, 1]$. The matrix A mixes the sources into the measured signals,

$$\begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{pmatrix} = \underbrace{\frac{1}{5}\begin{pmatrix} -1 & 2 & -1 & 1 \\ 2 & 1 & 3 & -2 \\ -2 & 2 & -1 & 1 \\ 1 & -2 & 3 & -3 \end{pmatrix}}_{A} \begin{pmatrix} s_1(t) \\ s_2(t) \\ s_3(t) \\ s_4(t) \end{pmatrix}, \tag{6.42}$$

which are shown in Fig. 6.21 (right). Use the FastICA algorithm described in Sect. 6.8.1 to determine the independent components (sources) $\boldsymbol{s}$ from the signals $\boldsymbol{x}$. Compare the computed sources to the original ones (6.41).

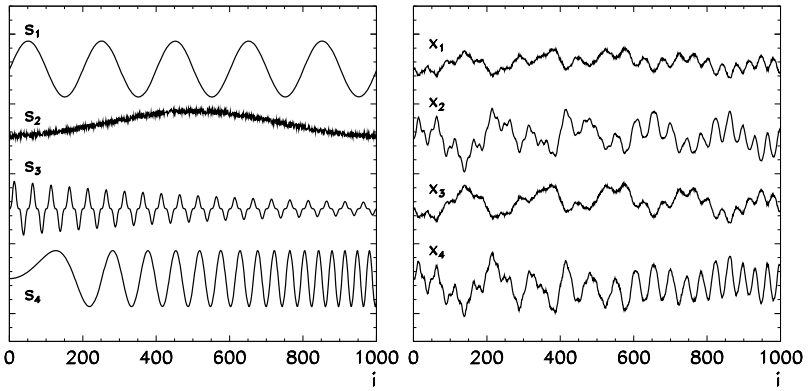


Fig. 6.21 [*Left*] The original signals (sources) from (6.41). [*Right*] The measured signals which are known mixtures of the sources (6.42)

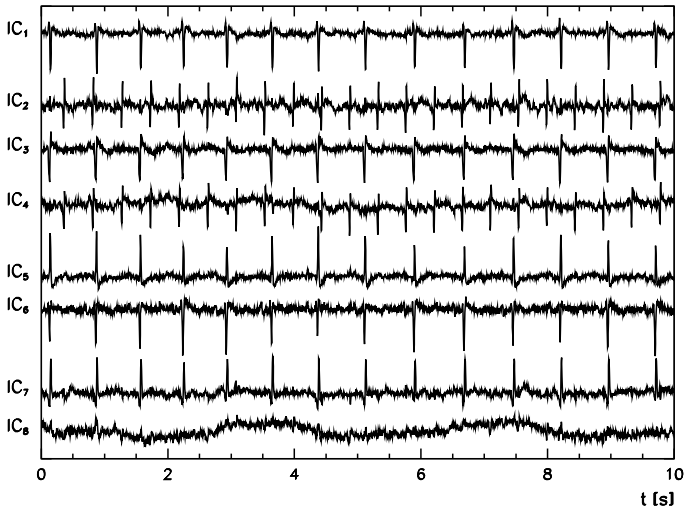


Fig. 6.22 Independent components (sources) of the signals shown in Fig. 6.16. The child's fast heartbeat is clearly identifiable in the independent components IC_2 and IC_4. Other components display the mother's heartbeat, except IC_8, which has probably been generated by breathing during the measurement

In the case described here the number of sources is equal to the number of measured signals, so the model $\boldsymbol{x} = A\boldsymbol{s}$ is exactly invertible, $\boldsymbol{s} = W\boldsymbol{x} = A^{-1}\boldsymbol{x}$. The computed and exact sources can therefore be accurately compared. Discuss the case with fewer sources than signals (invent your own mixing matrix).

⊕ Perform the independent component analysis of eight electro-cardiogram (ECG) traces of a pregnant woman shown in Fig. 6.16 [57]. There are 2500 voltage readouts with a sampling frequency of 500 Hz. The final result of the analysis are

the eight independent components which should closely resemble those of Fig. 6.22. Observe the convergence of the algorithm by monitoring the direction of the vectors $\boldsymbol{w}_k$ and the values of their components during the iteration (use Fig. 6.17 as an illustration). Use the stabilized version of the FastICA algorithm as described on p. 323. How do your conclusions change if raw data are first processed by the principal component analysis (PCA) and only the scores of a few leading principal components are retained for the analysis with the FastICA algorithm?

References

1. C. Robinson, *Dynamical Systems*, 2nd edn. (CRC Press, Boca Raton, 1999)
2. S.H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering* (Perseus Books, Reading, 1994)
3. P. Billingsley, *Probability and Measure*, 3rd edn. (Wiley-Interscience, New York, 1995)
4. R.M. Gray, *Probability, Random Processes and Ergodic Properties*, 2nd edn. (Springer, New York, 2009)
5. W. Feller, *An Introduction to Probability Theory and Its Applications, Vol. 1*, 3rd edn. (Wiley, New York, 1967)
6. W. Feller, *An Introduction to Probability Theory and Its Applications, Vol. 2*, 2nd edn. (Wiley, New York, 1971)
7. V.I. Arnold, A. Avez, *Ergodic Problems of Classical Mechanics* (Benjamin, New York, 1968)
8. L. Schiff, *Quantum Mechanics* (McGraw-Hill, New York, 1968)
9. P.Z. Peebles, *Probability, Random Variables and Random Signal Principles* (McGraw-Hill, New York, 1980)
10. J.D. Hamilton, *Time Series Analysis* (Princeton University Press, Princeton, 1994)
11. I.S. Tyurin, On the accuracy of the Gaussian approximation. Dokl. Math. **80**, 840 (2009)
12. J.P. Nolan, *Stable Distributions—Models for Heavy Tailed Data* (Birkhäuser, Boston, 2010)
13. S. Borak, W. Härdle, R. Weron, *Stable Distributions* (Humboldt University Berlin, Berlin, 2005)
14. A. Janicki, A. Weron, Can one see α-stable variables and processes? Stat. Sci. **9**, 109 (1994)
15. GSL (GNU Scientific Library). http://www.gnu.org/software/gsl
16. R. LePage, M. Woodroofe, J. Zinn, Convergence to a stable distribution via order statistics. Ann. Probab. **9**, 624 (1981)
17. B.D. Hughes, *Random Walks and Random Environments: Vol. 1: Random Walks* (Oxford University Press, New York, 1995)
18. M. Bazant, Random walks and diffusion. MIT OpenCourseWare, Course 18.366. http://ocw.mit.edu/courses/mathematics/
19. E.W. Montroll, G.H. Weiss, Random walks on lattices, II. J. Math. Phys. **6**, 167 (1965)
20. R. Metzler, J. Klafter, The random walk's guide to anomalous diffusion: a fractional dynamics approach. Phys. Rep. **339**, 1 (2000)
21. M.F. Shlesinger, B.J. West, J. Klafter, Lévy dynamics of enhanced diffusion: application to turbulence. Phys. Rev. Lett. **58**, 1100 (1987)
22. V. Tejedor, R. Metzler, Anomalous diffusion in correlated continuous time random walks. J. Phys. A, Math. Theor. **43**, 082002 (2010)
23. S. Meyn, R.L. Tweedie, *Markov Chains and Stochastic Stability* (Springer, Berlin, 1995)
24. E. Parzer, *Stochastic Processes* (Holden-Day, San Francisco, 1962)
25. E. Ott, *Chaos in Dynamical Systems*, 2nd edn. (Cambridge University Press, Cambridge, 2002)
26. A. Papoulis, *Probability, Random Variables and Stochastic Processes*, 3rd edn. (McGraw-Hill, New York, 1991)

27. C. Meyer, *Matrix Analysis and Applied Linear Algebra* (SIAM, Philadelphia, 2000)
28. T. Cover, J. Thomas, *Elements of Information Theory*, 2nd edn. (Wiley, New York, 2006)
29. E.T. Jaynes, Information theory and statistical mechanics. Phys. Rev. **106**, 620 (1957)
30. E.T. Jaynes, Information theory and statistical mechanics, II. Phys. Rev. **108**, 171 (1957)
31. M. Horvat, The ensemble of random Markov matrices. J. Stat. Mech. **2009**, P07005 (2009). For further reading see references therein
32. F. Schwabl, *Advanced Quantum Mechanics*, 4th edn. (Springer, Berlin, 2008)
33. A.V. Oppenheim, R.W. Schafer, *Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, 1975)
34. C.W. Gardiner, *Handbook of Stochastic Methods*, 2nd edn. (Springer, Berlin, 1997)
35. R. Whittle, DSP generation of pink (1/f) noise. http://www.firstpr.com.au/dsp/pink-noise/
36. L. Trammell, Pink noise generator; Improvements in the correlated pink noise generator evaluation; Hardware-friendly implementation of the correlated pink noise generator. http://home.earthlink.net/~ltrammell
37. P. Bourke, Generating noise with different power spectra laws. http://local.wasp.uwa.edu.au/~pbourke/fractals/noise
38. R. Zwanzig, Time-correlation functions and transport coefficients in statistical mechanics. Annu. Rev. Phys. Chem. **16**, 67 (1965)
39. M. Reed, B. Simon, *Methods of Modern Mathematical Physics, Vol. 1: Functional Analysis*, 2nd edn. (Academic Press, San Diego, 1980)
40. J.B. Conway, *A Course in Functional Analysis* (Springer, New York, 1985)
41. W. Rudin, *Functional Analysis* (McGraw-Hill, New York, 1973)
42. S.L. Marple Jr., *Digital Spectral Analysis* (Prentice-Hall, Englewood Cliffs, 1987)
43. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 2nd edn. (MIT Press, Cambridge, 2001)
44. J.M. Makhoul, Linear prediction: a tutorial review. Proc. IEEE **63**, 561 (1975)
45. S.V. Vaseghi, *Advanced Digital Signal Processing and Noise Reduction*, 4th edn. (Wiley, New York, 2008)
46. P. Vaidyanathan, *The Theory of Linear Prediction. Synthesis Lectures on Signal Processing* (Morgan & Claypool, San Rafael, 2008)
47. P.J. Brockwell, R.A. Davis, *Introduction to Time Series and Forecasting* (Springer, New York, 2002)
48. G.E.P. Box, G.M. Jenkins, G. Reinsel, *Time Series Analysis: Forecasting & Control*, 4th edn. (Wiley, New York, 2008)
49. G.E.P. Box, D.R. Cox, An analysis of transformations. J. R. Stat. Soc. B **26**, 211 (1964)
50. G.H. Golub, C.F. van Loan, *Matrix Computations*, 3rd edn. (The Johns Hopkins University Press, Baltimore, 1996)
51. S.M. Kay, S.L. Marple, Spectrum analysis—a modern perspective. Proc. IEEE **69**, 1380 (1981)
52. W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 3rd edn. (Cambridge University Press, Cambridge, 2007). See also the equivalent handbooks in Fortran, Pascal and C, as well as http://www.nr.com
53. Matlab, The MathWorks. http://www.mathworks.com
54. P. Stoica, R. Moses, *Spectral Analysis of Signals* (Prentice-Hall, Englewood Cliffs, 2005)
55. S. Haykin, *Adaptive Filter Theory*, 4th edn. (Prentice-Hall, Englewood Cliffs, 2001)
56. A.J. Izenman, *Modern Multivariate Statistical Techniques* (Springer, Berlin, 2008)
57. L. De Lathauwer, B. De Moor, J. Vandewalle, Fetal electrocardiogram extraction by blind source subspace separation. IEEE Trans. Biomed. Eng. **47**, 567 (2000). The signals are accessible at the website of KU Leuwen, http://www.esat.kuleuven.ac.be/~smc/daisy/daisydata.html
58. A. Hyvärinen, E. Oja, Independent component analysis: algorithms and applications. Neural Netw. **13**, 411 (2000)
59. A. Hyvärinen, Survey on independent component analysis. Neural Comput. Surv. **2**, 94 (1999)

60. A. Hyvärinen, Fast and robust fixed-point algorithms for independent component analysis. IEEE Trans. Neural Netw. **10**, 626 (1999)
61. M.J. Feigenbaum, Quantitative universality for a class of non-linear transformations. J. Stat. Phys. **19**, 25 (1978)
62. M.J. Feigenbaum, The universal metric properties of nonlinear transformations. J. Stat. Phys. **21**, 669 (1979)
63. J. Lichtenberg, M.A. Lieberman, *Regular and Stochastic Motion* (Springer, New York, 1983)

Chapter 7
Initial-Value Problems for ODE

Often we try to understand the evolution of a system from a known initial to an unknown later state by observing the temporal variation (dynamics) of quantities that we use to describe the system. There are endless examples. Physicists study oscillations of non-linear mechanical pendula and electric circuits, monitor the motion of charged particles in electro-magnetic fields, and predict orbits of satellites in the presence of other celestial bodies. Chemists investigate properties and reaction mechanisms of compounds. Biologists use models to describe the population dynamics of plant or animal species. Modern economics, among other, uses dynamical models to predict stock market exchange rates.

7.1 Evolution Equations

In simpler cases the dynamics of the system is determined by ordinary (not necessarily linear) differential equations (ODE) with time derivatives and prescribed initial conditions. (Problems with specified boundary conditions, which are harder than initial-value problems, are discussed in Chap. 8. Partial differential equations that are even more difficult to solve due to the appearance of derivatives in variables other than time, are treated in Chaps. 9, 10, and 11.) Though the dynamical (evolution) equation for a physical problem is known, it is often not analytically solvable and we have to resort to numerical methods. We first discuss first-order ordinary differential equations of the form

$$y' = f(x, y), \quad y(x_0) = y_0, \tag{7.1}$$

where f is a known function. In the language of dynamical analysis this equation defines a *non-autonomous system*. If the system does not depend on x,

$$y' = f(y),$$

it is called *autonomous*. The variable x most often means time.

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9_7, © Springer-Verlag Berlin Heidelberg 2012

Converting a Mth Order Equation to M First-Order Equations Problems with ordinary differential equations of higher order M can be rewritten as systems of M first-order differential equations by using auxiliary variables. We write the system as in (7.1), except that the solution y and the right-hand side of the equation f have more components and we treat them as vectors:

$$\boldsymbol{y}'(x) = \begin{pmatrix} y_1'(x) \\ y_2'(x) \\ \vdots \\ y_M'(x) \end{pmatrix} = \begin{pmatrix} f_1(x, y_1, y_2, \ldots, y_M) \\ f_2(x, y_1, y_2, \ldots, y_M) \\ \vdots \\ f_M(x, y_1, y_2, \ldots, y_M) \end{pmatrix} = \boldsymbol{f}(x, \boldsymbol{y}), \tag{7.2}$$

where $\boldsymbol{y} \in \mathbb{R}^M$ and $\{f_i\}_{i=1}^M$ are scalar functions, the components of $\boldsymbol{f} : \mathbb{R}^{M+1} \to \mathbb{R}^M$. We choose the new variables such that each intermediate variable is the derivative of the previous one. For a second-order equation $m\ddot{\boldsymbol{x}} = \boldsymbol{F}$ (Newton's law) this intermediate variable is the velocity $\boldsymbol{v}$, and the equation can be written as a system of two vector (six scalar) first-order equations

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{v} \\ m\dot{\boldsymbol{v}} &= \boldsymbol{F}(t, \boldsymbol{x}, \boldsymbol{v}) \end{aligned} \quad \Longleftrightarrow \quad \begin{pmatrix} y_1' \\ y_2' \\ y_3' \\ y_4' \\ y_5' \\ y_6' \end{pmatrix} = \begin{pmatrix} y_4 \\ y_5 \\ y_6 \\ F_1(x, y_1, y_2, y_3, y_4, y_5, y_6)/m \\ F_2(x, y_1, y_2, y_3, y_4, y_5, y_6)/m \\ F_3(x, y_1, y_2, y_3, y_4, y_5, y_6)/m \end{pmatrix}.$$

If sufficiently many derivatives of the functions f_i with respect to x and y_j exist, $\boldsymbol{f}$ can be Taylor-expanded around any point $(x_*, \boldsymbol{y}_*)$:

$$\boldsymbol{f}(x, \boldsymbol{y}) = \boldsymbol{f}(x_*, \boldsymbol{y}_*) + \left.\frac{\partial \boldsymbol{f}}{\partial x}\right|_* (x - x_*) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_* \cdot (\boldsymbol{y} - \boldsymbol{y}_*) + \cdots .$$

We shall see in the discussion of the stability of numerical methods that the local behavior of the solution strongly depends on the last term which contains the Jacobi matrix of the partial derivatives of the functions f_i with respect to the variables y_j,

$$J = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}} = \begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \cdots & \frac{\partial f_1}{\partial y_M} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \cdots & \frac{\partial f_2}{\partial y_M} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_M}{\partial y_1} & \frac{\partial f_M}{\partial y_2} & \cdots & \frac{\partial f_M}{\partial y_M} \end{pmatrix}. \tag{7.3}$$

The stability analysis (explained in Sect. 7.5 and Appendix F) is closely related to the solution of the system of differential equations $\boldsymbol{y}' = J\boldsymbol{y}$ and the search for (in general complex) eigenvalues of the matrix J.

When does (7.1) have a unique solution? Suppose we are seeking the solution in the closed region $\mathcal{D} \equiv \{(x, y) : x_0 - \alpha \le x \le x_0 + \alpha, y_0 - \beta \le y \le y_0 + \beta\}$. If f is

continuous in $\mathcal{D}$, bounded by $|f(x, y)| < B$, and if for each x, y_1, and y_2 from $\mathcal{D}$ we can find a constant Λ such that the Lipschitz inequality

$$\left|f(x, y_2) - f(x, y_1)\right| \leq \Lambda |y_2 - y_1| \tag{7.4}$$

is fulfilled, then on the interval centered at x_0 with width $\Delta x = \min\{\alpha, \beta/B, \gamma/\Lambda\}$, where $0 < \gamma < 1$, precisely one solution of (7.1) exists [1]. The parameter Λ in (7.4) plays an important role in the error estimates of numerical methods for differential equations. The generalization of the existence theorem to systems of equations can be found in [1].

Road-Signs To solve equations of the type (7.1), we use numerical methods in which the interval $[a, b]$ on the continuous x-axis is discretized by a mesh

$$x_0 = a < x_1 < x_2 < \cdots < x_N = b$$

of generally non-equidistant points, while the differential equation is transcribed as a difference scheme. The following sections are devoted to the analysis of precision, stability, convergence properties, and computational efficiency of individual difference schemes. We use $\boldsymbol{y}_n$ to denote the approximate values of the solution at the points x_n and $\boldsymbol{y}(x_n)$ to denote the exact values.

Initial-value problems for ordinary differential equations can be solved by *single-step methods* in which we need just the previous values $\boldsymbol{y}_n$ in order to obtain the next ones, $\boldsymbol{y}_{n+1}$, as well as *multi-step methods*, in which $\boldsymbol{y}_{n+1}$ is computed from several previous values $\boldsymbol{y}_n$, $\boldsymbol{y}_{n-1}$, ...

The methods also differ by how an individual time step is executed: in *explicit methods* the old values are used to compute the new ones by using a known explicit formula; in *implicit methods* (Sect. 7.9) the values in the next time step occur in two or more places at both sides of the equation and need to be computed at each step by solving a system of non-linear (and not necessarily algebraic) equations. In all methods we obtain discrete approximations for the values $\boldsymbol{y}(x_n)$ determined by the equation that specifies the derivatives $\boldsymbol{y}'$: these methods are therefore commonly known as *integrators*.

7.2 Explicit Euler's Methods

Euler's methods are the simplest representatives of difference schemes in which the approximate solution of the differential equation at the point x_n is used to compute the approximate solution at the next point, x_{n+1}. We restrict the discussion to the equidistant mesh,

$$h = (b - a)/N = x_{n+1} - x_n, \quad n = 0, 1, \ldots, N - 1.$$

Explicit (basic) Euler's method is just a reshuffled difference approximation of the first derivative: $y' \approx (y(x+h) - y(x))/h$. The approximate value of the solution $\boldsymbol{y}_{n+1}$ at the next mesh point is obtained from the previous one, $\boldsymbol{y}_n$, by using

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{y}'_n = \boldsymbol{y}_n + h\boldsymbol{f}(x_n, \boldsymbol{y}_n). \tag{7.5}$$

In other words, at each point x_n the curve of the exact solution is approximated by the tangent at that point, with the slope determined by the differential equation. We advance along this tangent to the next approximate solution value.

Local Discretization Error and the Method Order In one step from x_n to x_{n+1} we make a *discretization* or *truncation error*, which is defined as the difference between the numerical and exact solution in one step, assuming that these two solutions agree in all previous steps. (More on errors follows in Sect. 7.4.) By comparing the expression (7.5) for $\boldsymbol{y}_{n+1}$ with the Taylor expansion

$$\boldsymbol{y}(x_n + h) = \boldsymbol{y}(x_n) + h\boldsymbol{y}'(x_n) + \frac{1}{2}h^2\boldsymbol{y}''(\xi_n), \quad \xi_n \in [x_n, x_{n+1}],$$

we see that the local error of the Euler method is $\boldsymbol{y}(x_{n+1}) - \boldsymbol{y}_{n+1} \sim \mathcal{O}(h^2)$. A difference scheme with a local error of the order h^{p+1} is said to be a *method of order* p: the basic Euler method is first order in h.

In the *improved Euler method* the curve of the true solution is approximated by the tangent at the midpoint of the current interval $[x_n, x_{n+1}]$ and not at the initial point,

$$\begin{aligned} \boldsymbol{y}_{n+\frac{1}{2}} &= \boldsymbol{y}_n + \frac{1}{2}h\boldsymbol{y}'_n, \\ \boldsymbol{y}'_{n+\frac{1}{2}} &= \boldsymbol{f}\left(x_{n+\frac{1}{2}}, \boldsymbol{y}_{n+\frac{1}{2}}\right), \\ \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\boldsymbol{y}'_{n+\frac{1}{2}}. \end{aligned} \tag{7.6}$$

The local error of this method is $\mathcal{O}(h^3)$ (i.e. the method is second order) if the third derivative of the solution is bounded in the vicinity of x_n, but the method may be unstable in certain cases.

There is also the *symmetrized Euler method*, which follows from the symmetric difference approximation of the derivative, $y' \approx (y(x+h) - y(x-h))/2h$. We use the difference scheme

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_{n-1} + 2h\boldsymbol{y}'_n, \tag{7.7}$$

so we have to known *two* previous solution values at each step: we are dealing with a two-step method forcing us to use uniform mesh spacings. The initial condition $\boldsymbol{y}_0 = \boldsymbol{y}(x_0)$ is known, while the approximation of the solution $\boldsymbol{y}_1$ can be, for example, computed by the basic Euler method. The symmetrized Euler method is also second order.

7.3 Explicit Methods of the Runge–Kutta Type

Methods of the Runge–Kutta (RK) type are the best known representatives of single-step methods. The solution is advanced in a sequence of short steps h like in the Euler methods, but the right-hand side of the differential equation (7.1) is evaluated at particular intermediate points within each time step; by a suitable combination of these auxiliary quantities we wish the method to attain higher precision. The solution value at the next point of a step is computed as

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{F}(h, x_n, \boldsymbol{y}_n; \boldsymbol{f}), \tag{7.8}$$

where

$$\boldsymbol{F}(h, x_n, \boldsymbol{y}_n; \boldsymbol{f}) = \sum_{i=1}^{m} b_i \boldsymbol{k}_i.$$

We express the quantities $\boldsymbol{k}_i$ by the right sides of (7.2),

$$\boldsymbol{k}_i = \boldsymbol{f}\left(x_n + c_i h, \boldsymbol{y}_n + h\sum_{j=1}^{m} a_{ij}\boldsymbol{k}_j\right), \quad i = 1, 2, \ldots, m. \tag{7.9}$$

The procedure (7.8) is known as a *m-stage* Runge–Kutta (RK) method. We try to determine the parameters a_{ij}, b_i, and c_i such that the order of the method p is as high as possible. We Taylor-expand the numerical solution $\boldsymbol{y}_1$ and the exact solution $\boldsymbol{y}(x+h)$ around x and compare the coefficients in both expansions [2]. This results in a system of non-linear equations for the parameters that in general does not have a unique solution. Each solution corresponds to a different RK method. If we set $c_1 = 0$ and $a_{ij} = 0$ for $j \geq i$ (the sum in (7.9) then runs only over $1 \leq j \leq i-1$), at each step $x_n \to x_{n+1}$ all $\boldsymbol{k}_i$ can be computed: the method is *explicit* since in the computation of $\boldsymbol{k}_i$ only the previous values $\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_{i-1}$ are used. In other cases the method is *implicit* and the system (7.9) needs to be solved for $\boldsymbol{k}_i$ at each step: such methods are discussed in Sect. 7.9.

In some cases the system for the parameters is easily solvable and leads to m-stage formulas requiring one evaluation of $\boldsymbol{f}(x, \boldsymbol{y})$ per stage. The only solution for the coefficients in the case $p = m = 1$ is $b_1 = 1$, $b_i = 0$ $(i > 1)$, $c_i = 0$, $a_{ij} = 0$, and gives the basic Euler method (7.5). One of the solutions for $p = m = 4$ corresponds to the popular Runge–Kutta method of fourth order (RK4):

$$\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{f}(x_n, \boldsymbol{y}_n),\\
\boldsymbol{k}_2 &= \boldsymbol{f}\left(x_n + \frac{1}{2}h, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_1\right),\\
\boldsymbol{k}_3 &= \boldsymbol{f}\left(x_n + \frac{1}{2}h, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_2\right),\\
\boldsymbol{k}_4 &= \boldsymbol{f}(x_n + h, \boldsymbol{y}_n + h\boldsymbol{k}_3),\\
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{h}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) + \mathcal{O}\big(h^5\big).
\end{aligned} \tag{7.10}$$

We arrange the coefficients a_{ij}, b_i, and c_i of any RK method in a *Butcher tableau*. A general explicit scheme is shown at the left, the RK4 method at the right:

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_m & a_{m1} & a_{m2} & \cdots & a_{mm-1} & \\ \hline & b_1 & b_2 & \cdots & b_{m-1} & b_m \end{array} \qquad \begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array}$$

The order p cannot be attained with less than p stages. The explicit RK4 method is of highest order at which we still have $p = m$, and this is one of the reasons for its popularity. For RK methods of orders $p > 4$ we need $m > p$ stages, and each additional stage implies another evaluation of $\boldsymbol{f}(x, \boldsymbol{y})$, a consideration of prime importance if the functions $\boldsymbol{f}$ are computationally expensive. The world record in local precision is held by the 10th order RK methods requiring 18 [3] or 17 stages [4].

Dense Output for the RK4 Method Expressions (7.10) allow us to obtain the solution at the mesh points x_n. But often the point $x^\star$ at which the solution is sought is not known in advance, as it may implicitly depend on the computed solution and may be determined by some equation $\boldsymbol{g}(x^\star, \boldsymbol{y}(x^\star)) = \boldsymbol{0}$, for example, when seeking the Poincaré sections of the solution. The results at intermediate points $x^\star \in [x_n, x_{n+1}]$ can be obtained by using *dense output* [2] which requires further $(m^\star - m)$ stages to be added to the scheme. The approximate solution at an arbitrary point on the interval $[x_n, x_{n+1}]$ is given by

$$\boldsymbol{y}_{n+\theta} = \boldsymbol{y}_n + h \sum_{i=1}^{m^\star} b_i(\theta) \boldsymbol{k}_i, \quad 0 \le \theta \le 1. \tag{7.11}$$

The polynomials b_i are determined by requiring $\boldsymbol{y}_{n+\theta} - \boldsymbol{y}(x_n + \theta h) = \mathcal{O}(h^{p^\star+1})$. The dense output for the RK4 method can be formulated already with $m^\star = m$ (i.e. without computing additional $\boldsymbol{k}_i$). We use

$$b_1(\theta) = \theta - \frac{3\theta^2}{2} + \frac{2\theta^3}{3}, \qquad b_2(\theta) = b_3(\theta) = \theta^2 - \frac{2\theta^3}{3}, \qquad b_4(\theta) = -\frac{\theta^2}{2} + \frac{2\theta^3}{3}.$$

A price has to paid for this convenient interpolation: while the basic method has order $p = 4$, the solution given by the dense output has only order $p^\star = 3$.

7.4 Errors of Explicit Methods

When an explicit method is implemented in a program, two kinds of error occur: *discretization* (also known as *truncation*) errors and *round-off* errors. The errors can be

local (occurring in one step) or *global* (accumulated throughout the integration interval). In the following we also introduce the concepts of *consistency*, *convergence*, and *stability* of explicit methods.

7.4.1 Discretization and Round-Off Errors

The *discretization error* originates in the nature of the method and it appears because the sum (7.8) or the Taylor expansion of the true solution to which we compare it, is finite. This error occurs regardless of the arithmetic precision of the implementation. It depends on the step size h, on the order of the method, and on the function $\boldsymbol{f}$ itself. The local discretization error τ_{n+1} in the nth step of an explicit method is defined as

$$\tau_{n+1} = \left\| \boldsymbol{y}(x_{n+1}) - \boldsymbol{y}(x_n) - h\boldsymbol{F}\big(h, x_n, \boldsymbol{y}(x_n); \boldsymbol{f}\big) \right\|. \tag{7.12}$$

Assuming that the numerical and exact solution match perfectly until the nth step, we have $\boldsymbol{y}(x_n) = \boldsymbol{y}_n$ and $\boldsymbol{y}(x_n) + h\boldsymbol{F}(h, x_n, \boldsymbol{y}(x_n); \boldsymbol{f}) = \boldsymbol{y}_{n+1}$, and then

$$\tau_{n+1} = \left\| \boldsymbol{y}_{n+1} - \boldsymbol{y}(x_{n+1}) \right\|.$$

In general, for an explicit method of order p we have

$$\tau_{n+1} = \Psi\big(\boldsymbol{y}(x_n)\big)h^{p+1} + \mathcal{O}\big(h^{p+2}\big). \tag{7.13}$$

The first term at the right side of (7.12) is called the *principal local discretization error* in which the function Ψ depends on the elementary differentials up to order $p+1$, evaluated at $\boldsymbol{y}(x_n)$ [5].

Round-off errors occur because instead of the exact solution $\boldsymbol{y}_n$ of the *difference scheme* we compute a numerically approximate value $\boldsymbol{Y}_n$, with the difference $\rho_n \equiv \|\boldsymbol{Y}_n - \boldsymbol{y}_n\|$. These errors arise due to the finite precision of floating-point arithmetic and depend on the type of arithmetic operations performed and their quantity. The round-off errors increase with the number of steps, and this is certainly one reason *not to* decrease the step size h excessively.

The equation for τ_n and the analogous expression for ρ_n define the contributions to the local discretization and round-off errors at the level of the difference scheme. But the success of an integrator is preferably judged by the *global error* $E_N \equiv \|\boldsymbol{Y}_N - \boldsymbol{y}(x_N)\|$ that measures the cumulative difference between the true value and its computed approximation at the last integration point ($n = N$). The error that accumulates until some intermediate point x_n, $E_n \equiv \|\boldsymbol{Y}_n - \boldsymbol{y}(x_n)\|$, can also be understood as global up to that point. The global error *is not* equal to the sum of the local errors between $x_0 = a$ and $x_N = b$. It results from a propagation of local errors along the whole integration interval.

Most often, we can determine only an upper limit for the global error. In general, the order of the global error is one less than the order of the local error. If the local error satisfies

$$\tau_{n+1} \leq Ch^{p+1}$$

(see (7.13)) and $\boldsymbol{F}$ is Lipschitz-continuous, $\|\boldsymbol{F}(h, x, \boldsymbol{y}; \boldsymbol{f}) - \boldsymbol{F}(h, x, \boldsymbol{z}; \boldsymbol{f})\| \leq \Lambda \|\boldsymbol{y} - \boldsymbol{z}\|$, the global discretization error can be estimated as

$$\|E_N\| \leq Ch^p \frac{1}{\Lambda}\left[\mathrm{e}^{\Lambda(b-a)} - 1\right] + \mathrm{e}^{\Lambda(b-a)} \left\|\boldsymbol{y}_0 - \boldsymbol{y}(x_0)\right\|.$$

(The last term at the right vanishes if the initial condition is numerically exact.) In general the step h along $[a, b]$ can be non-uniform. In that case h in the above equation stands for $h = \max_n\{h_n\}$.

Example Useful global error estimates exist for simpler difference schemes. The local error can be propagated to the global error either by resorting to known or assumed properties of the exact solution, or by exploiting the properties of the difference scheme (see [6] and [2], Chap. II). The global errors of the scalar basic and symmetrized Euler methods with equidistant steps h ((7.5) and (7.7)) can be estimated as

$$\text{Eq. (7.5):} \quad |E_N| \leq \mathrm{e}^{\Lambda(b-a)}\left[|\rho_0| + \frac{1}{\Lambda}\left(\frac{hM_2}{2} + \frac{\rho}{h}\right)\right], \tag{7.14}$$

$$\text{Eq. (7.7):} \quad |E_N| \leq \mathrm{e}^{\Lambda(b-a)}\left[\max\left\{|\rho_0|, |\rho_1|\right\} + \frac{1}{\Lambda}\left(\frac{h^2M_3}{3} + \frac{\rho}{h}\right)\right], \tag{7.15}$$

where ρ_0 and ρ_1 are the round-off errors at $x_0 = a$ and x_1, while the largest round-off error along the whole integration interval is $\rho \equiv \max_{1\leq n\leq N} |\rho_n|$. The Lipschitz constant Λ is equal to the upper limit for $|\partial f/\partial y|$. The estimates for the upper limits of $M_2 = \max|y''|$ and $M_3 = \max|y'''|$ (both on $a \leq x \leq b$) are harder to obtain, as at least something needs to be known about the true solution $y(x)$! Still, (7.14) and (7.15) clearly reveal the interplay of the discretization and round-off errors. An optimal h exists at which the global error is minimal (Fig. 7.1 for the basic Euler method.)

7.4.2 Consistency, Convergence, Stability

The differential equation and the difference scheme for it are *consistent* when

$$\lim_{h\to 0} \boldsymbol{F}(h, x_n, \boldsymbol{y}_n; \boldsymbol{f}) = \boldsymbol{f}(x_n, \boldsymbol{y}_n)$$

(the difference scheme approximates the differential equation arbitrarily well in the limit $h \to 0$). A difference scheme is *convergent* if the approximate solution converges to the exact solution when the step size h is decreased,

$$\lim_{h\to 0} \left\|\boldsymbol{y}_n - \boldsymbol{y}(x_n)\right\| = 0, \quad nh = x_n - x_0.$$

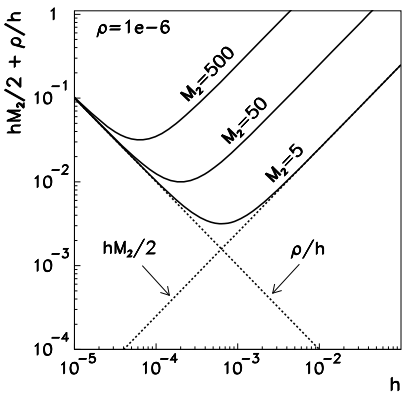


Fig. 7.1 The h-dependent part of the total global error (7.14) for the basic Euler method on an equidistant mesh for $\rho = 10^{-6}$ and for three values $M_2 = 5$, 50, and 500. Separately shown for $M_2 = 5$ are the linearly increasing discretization component of the error and the inversely proportional round-off part

A difference scheme is *stable* if numerical errors do not accumulate, i.e. when for numerical solutions $\boldsymbol{y}_n$ and $\widetilde{\boldsymbol{y}}_n$, with initial conditions $\boldsymbol{y}_0$ and $\widetilde{\boldsymbol{y}}_0$, we have

$$\|\boldsymbol{y}_n - \widetilde{\boldsymbol{y}}_n\| \leq C\|\boldsymbol{y}_0 - \widetilde{\boldsymbol{y}}_0\|,$$

where C does not depend on h. More on stability will follow in Sect. 7.5. For convergence of a difference scheme, both consistency and stability are needed:

$$\text{consistency} + \text{stability} \Longleftrightarrow \text{convergence}.$$

All explicit RK methods are convergent within their stability regions [7]. The relation between the discretization error of the scheme, its stability, and the convergence of its solution to that of the differential equation also applies to difference schemes for partial differential equations (Sects. 9.3 and 9.5).

7.4.3 Richardson Extrapolation

Global error estimates usually provide crude, over-estimated upper bounds. This motivates us to control the step size h in explicit RK methods carefully, and use the error estimates to adjust that size accordingly. The local error in an explicit scheme of order p can be estimated if we assume that the factor of the h^{p+1} term in the remainder of the Taylor series does not change too quickly on the current interval of length h (or is constant, to order h^{p+1}). We rely on the fact that the principal discretization error in (7.12) is much larger than the remainder, which is not always the case (see Sect. 7.9). This is the idea behind *Richardson extrapolation* which can be used to improve the accuracy of the solution by one order at each step of the difference scheme.

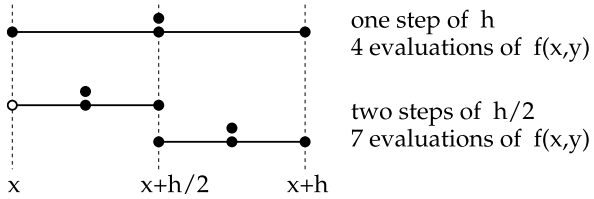


Fig. 7.2 Estimation of the local discretization error for the RK4 method by halving the step size h. The *symbols* • denote the evaluations of the function $\boldsymbol{f}(x, \boldsymbol{y})$ (there is no need to repeat the evaluation at the point ∘)

We use the solution $\boldsymbol{y}_{n-1}$ to compute its next value $\boldsymbol{y}_n$, once in a single step of h, then in two steps of $h/2$ each (Fig. 7.2). If the method is of order p, we have

$$1 \cdot h: \quad \boldsymbol{y}_n = \boldsymbol{y}(x_n) + Ch^{p+1} + \mathcal{O}\big(h^{p+2}\big),$$
$$2 \cdot \frac{1}{2}h: \quad \widetilde{\boldsymbol{y}}_n = \boldsymbol{y}(x_n) + 2C\left(\frac{1}{2}h\right)^{p+1} + \mathcal{O}\big(h^{p+2}\big).$$

We eliminate C from these equations and obtain

$$\boldsymbol{y}(x_n) = \widetilde{\boldsymbol{y}}_n + \frac{\widetilde{\boldsymbol{y}}_n - \boldsymbol{y}_n}{2^p - 1} + \mathcal{O}\big(h^{p+2}\big), \tag{7.16}$$

which is an improved approximation of the true solution $\boldsymbol{y}(x_n)$.

The norm $\|\widetilde{\boldsymbol{y}}_n - \boldsymbol{y}_n\|$ is a good measure of the local discretization error that can be built into the computer program for any chosen method in order to assess whether the chosen step size h is too small or too large with respect to the prescribed precision. If the error in a given step is above the desired value, we decrease h and repeat the procedure.

7.4.4 Embedded Methods

To estimate the error with a single halving of the step, 11 evaluations of the function $\boldsymbol{f}(x, \boldsymbol{y})$ are needed (Fig. 7.2), which may be too costly numerically. This served as a motivation for *embedded* Runge–Kutta methods [2] in which the local error can be estimated differently. They are m-stage RK methods which contain one linear combination of the intermediate quantities $\boldsymbol{k}_i$ to attain an approximation of order p,

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h(b_1\boldsymbol{k}_1 + b_2\boldsymbol{k}_2 + \cdots + b_m\boldsymbol{k}_m) + \mathcal{O}\big(h^{p+1}\big),$$

and another combination yielding an approximation of order $\widehat{p} \neq p$,

$$\widehat{\boldsymbol{y}}_{n+1} = \boldsymbol{y}_n + h(\widehat{b}_1\boldsymbol{k}_1 + \widehat{b}_2\boldsymbol{k}_2 + \cdots + \widehat{b}_m\boldsymbol{k}_m) + \mathcal{O}\big(h^{\widehat{p}+1}\big).$$

An embedded method is denoted by $p(\widehat{p})$; usually $\widehat{p} = p \pm 1$. The coefficients b_i and $\widehat{b}_i$ are determined as to minimize the leading error constant of either the low-

order or the high-order solution approximation. This consideration dictates whether the estimate $\boldsymbol{y}_{n+1}$ or $\widehat{\boldsymbol{y}}_{n+1}$ should be taken as the initial value in the next step of the integration. Among the most practical is the Dormand–Prince 5(4) method, in which the initial value for the next step is $\boldsymbol{y}_{n+1}$, while the local error estimate is

$$\boldsymbol{\Delta} = \boldsymbol{y}_{n+1} - \widehat{\boldsymbol{y}}_{n+1} = \sum_{i=1}^{m} (b_i - \widehat{b}_i)\boldsymbol{k}_i .$$

By decreasing h we can bring the value $\|\boldsymbol{\Delta}\|$ below the prescribed tolerance and thus control the magnitude of the local error. The coefficients of the Dormand–Prince 5(4) method are listed in this table:

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
y_{n+1}	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
$\widehat{y}_{n+1}$	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

An abundance of similar methods with such handy error estimates can be found in the literature, for example, the formerly popular fourth-order Fehlberg method with Cash–Karp parameters and six stages [8]. Until recently it was implemented in [9] where it has now been replaced by the Dormand–Prince 5(4) method [10]. The order 8(5,3) method [11] is also widely used.

Dense Output for the Dormand–Prince 5(4) Method Dense output of the form (7.11) for the Dormand–Prince 5(4) method, for which no additional evaluations of the function f are needed [2], is obtained by

$$\begin{aligned}
b_1(\theta) &= \theta^2(3-2\theta)b_1 + \theta(\theta-1)^2 \\
&\quad - 5\theta^2(\theta-1)^2(2558722523 - 31403016\,\theta)/11282082432, \\
b_2(\theta) &= 0, \\
b_3(\theta) &= \theta^2(3-2\theta)b_3 + 100\,\theta^2(\theta-1)^2(882725551 - 15701508\,\theta)/32700410799, \\
b_4(\theta) &= \theta^2(3-2\theta)b_4 - 25\,\theta^2(\theta-1)^2(443332067 - 31403016\,\theta)/1880347072, \\
b_5(\theta) &= \theta^2(3-2\theta)b_5 \\
&\quad + 32805\,\theta^2(\theta-1)^2(23143187 - 3489224\,\theta)/199316789632, \\
b_6(\theta) &= \theta^2(3-2\theta)b_6 - 55\,\theta^2(\theta-1)^2(29972135 - 7076736\,\theta)/822651844,
\end{aligned}$$

$$b_7(\theta) = \theta^2(\theta-1) + 10\,\theta^2(\theta-1)^2(7414447 - 829305\,\theta)/29380423.$$

The expressions for the individual $b_i(\theta)$ follow from an interpolation of function values by Hermite polynomials, which yields a fourth-order dense output. We have written the coefficients carefully in terms of fractions: the strict precision requirements of Problem 7.14.6 teach us why.

7.4.5 Automatic Step-Size Control

The spacings between the points $h = x_{n+1} - x_n$ used in difference schemes for ordinary differential equations need not be uniform and can be made smaller or larger according to the required local precision. To ensure their efficiency, difference schemes should be implemented by applying *adaptive step-size control*: the algorithm should constantly "sense" when the right-hand side (7.2) dictates a wild change in the solution or when it is tracing a humble functional dependence, and adjust the step size accordingly. The usual criterion for a change in the step size is the local discretization error.

In extrapolation or embedded RK methods, two solution estimates are available at each step, $\boldsymbol{y}_{n+1}$ (of order p) and $\widehat{\boldsymbol{y}}_{n+1}$ (of order $\widehat{p}$). We would like to constrain the local error $\boldsymbol{y}_{n+1} - \widehat{\boldsymbol{y}}_{n+1}$ by components, as

$$|y_{n+1,i} - \widehat{y}_{n+1,i}| \le S_i, \quad S_i = A_i + R_i \max\bigl\{|y_{n,i}|, |y_{n+1,i}|\bigr\},$$

where A_i are the prescribed absolute local errors of the ith solution component, and R_i are the relative local errors. As a joint measure of the computed error one may take

$$E = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(\frac{y_{n+1,i} - \widehat{y}_{n+1,i}}{S_i}\right)^2},$$

where M is the number of solution components. A well-established advice [2] for the choice of the optimal step size is

$$h_{\mathrm{opt}} = h(1/E)^{1/(q+1)}, \quad q = \min(p, \widehat{p}),$$

but more polished programs include another "safety factor", which ensures that the error estimate in the next step is acceptable and that h neither increases nor decreases too quickly. For this safety factor V, various authors propose empirically determined values

$$V = 0.8, 0.9, (0.25)^{1/(q+1)} \text{ or } (0.38)^{1/(q+1)},$$

where V has a lower bound of $V_{\min} \approx 0.2$ and an upper bound of $V_{\max} \approx 5$. For the new value of the step size we ultimately take

$$h_{\mathrm{new}} = h \cdot \min\bigl\{V_{\max}, \max\bigl\{V_{\min}, V \cdot (1/E)^{1/(q+1)}\bigr\}\bigr\}.$$

If at some step $E \leq 1$, we accept the computed step size and continue the solution $\boldsymbol{y}_n$ with the new initial value $\boldsymbol{y}_{n+1}$ and step h_{new}. If $E > 1$, we reject the step size and try again with the initial value $\boldsymbol{y}_n$ and step h_{new}. If a step has been rejected, it is recommendable to set $V_{\max} = 1$. Details on the choice of the initial step size and step size control can be found in [2] as well as in [9, 12].

7.5 Stability of One-Step Methods

Stability of explicit methods (for example, from the Runge–Kutta family) is judged by their *absolute* or *asymptotic stability* which is defined in Appendix F. The stability of a differential equation is not equivalent to the stability of the corresponding difference scheme, although they are closely related.

Stability of the Differential Equation The essence of stability can be elucidated with the one-dimensional linear initial-value problem $y' = \lambda y$, $y(x_0) = y_0$, $x \geq x_0$, $\lambda \in \mathbb{C}$. The analytic solution is $y(x) = y_0 \mathrm{e}^{\lambda(x-x_0)}$. The problem has a stable fixed point $y = 0$ if λ lies in the left complex half-plane ($\mathrm{Re}\,\lambda < 0$). A homogeneous system of linear differential equations is just a generalization of the above problem to

$$\boldsymbol{y}' = A\boldsymbol{y}, \quad \boldsymbol{y}(x_0) = \boldsymbol{y}_0, \ x \geq x_0, \tag{7.17}$$

where A is a matrix with eigenvalues λ_i that all lie in the left complex half-plane. Namely, when A is diagonalized, we decouple the system (7.17) to independent one-dimensional problems of the form $y' = \lambda_i y$, where $\lambda_i \in \mathbb{C}$ are the eigenvalues of the Jacobi matrix $J = A$. This has a physical background: the local behavior of some component of the solution depends on the character of the corresponding eigenvalue (increasing solutions for positive real eigenvalues, decreasing for negative real eigenvalues, and oscillatory for imaginary eigenvalues).

The stability of the most general problem

$$\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y}), \quad \boldsymbol{y}(x_0) = \boldsymbol{y}_0, \ x \geq x_0, \tag{7.18}$$

depends on the eigenvalues λ_i of the Jacobi matrix J with the elements $J_{ij} = \partial \boldsymbol{f}_i / \partial \boldsymbol{y}_j$ that we compute *locally*, at current x and $\boldsymbol{y}$. We use them to compute

$$r_{\max} = \max_i \left\{ \mathrm{Re}\,\lambda_i(J) \right\}.$$

If $r_{\max} < 0$, the system of equations is locally stable, while for $r_{\max} > 0$ it is locally unstable. See also Example on p. 669 in Appendix F.

Stability of the Numerical Method for a Differential Equation Consider (7.17) again. The individual components of the exact solution $\boldsymbol{y}$ tend to zero for $x \to \infty$ if the corresponding eigenvalues of A satisfy $\mathrm{Re}\,\lambda_i < 0$. For a numerical method

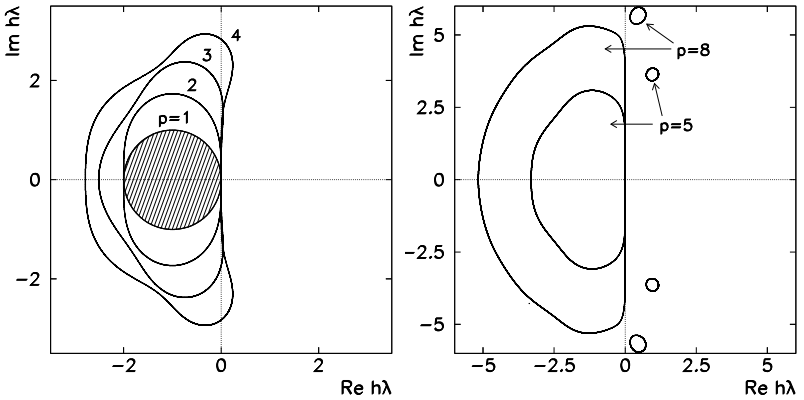


Fig. 7.3 [*Left*] Regions of stability of explicit pth order, p-stage methods $1 \le p \le 4$. The basic Euler method corresponds to the interior of the circle denoted by $p = 1$, while the interior of the shape 4 corresponds to the RK4 method. [*Right*] Regions of stability of Dormand–Prince methods of orders five and eight (note the change of scales). The regions become larger with increasing order p, but they remain bounded for all explicit schemes

this is not necessarily true. In a method of the form (7.8) the subsequent numerical solutions are computed as

$$\boldsymbol{y}_{n+1} = S(h\lambda)\boldsymbol{y}_n,$$

where S is the *growth factor* or *stability function* [13]. We define the region of absolute stability of such a method as the set of complex numbers $h\lambda$ (h real positive number, $\lambda \in \mathbb{C}$), for which $\lim_{n\to\infty} \|\boldsymbol{y}_n\| = 0$ (fixed point at origin is stable). The method is stable with values $h\lambda$ for which $|S(h\lambda)| \le 1$.

Example With the basic Euler method (7.5) for the scalar problem $y' = \lambda y = f(y)$ we obtain $y_{n+1} = y_n + hf(y_n) = (1 + h\lambda)y_n$ or

$$S(h\lambda) = \frac{\mathrm{d}y_{n+1}}{\mathrm{d}y_n} = 1 + h\lambda.$$

The region with $|S| = |1 + h\lambda| \le 1$ defines a shifted circle in the complex plane $(\mathrm{Re}\, h\lambda, \mathrm{Im}\, h\lambda)$ (shaded area $p = 1$ in Fig. 7.3). The basic Euler method remains stable only with a step size h for which $h\lambda$ lies within that area.

Similar calculations can be done for other methods. For an arbitrary scalar explicit method of order p with $q > p$ stages, the stability function becomes

$$S = \frac{\mathrm{d}y_{n+1}}{\mathrm{d}y_n} = 1 + h\lambda + \frac{(h\lambda)^2}{2!} + \cdots + \frac{(h\lambda)^p}{p!} + \sum_{i=p+1}^{q} \gamma_i (h\lambda)^i,$$

where the coefficients γ_i depend on the given scheme, and the region of stability is defined by $|S| \le 1$. Figure 7.3 shows the region of stability in the complex plane $(\mathrm{Re}\, h\lambda, \mathrm{Im}\, h\lambda)$ for explicit Runge–Kutta methods of order p.

Stability in vector problems is related to the spectral radius of the Jacobi matrix. If we use the basic Euler method to solve the system (7.17), we obtain $\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + hA\boldsymbol{y}_n = (I + hA)\boldsymbol{y}_n$. In this case the spectral radius $\rho(I + hA)$ should not exceed unity. In solving (7.18) by the general method (7.8), the spectral radius $\rho(I + h[\partial \boldsymbol{F}(h, x_n, \boldsymbol{y}_n; \boldsymbol{f})/\partial \boldsymbol{y}_n])$ should be less than 1. The procedures to compute the Jacobi matrix during the integration are described in [14].

7.6 Extrapolation Methods ⋆

Extrapolation methods, first suggested by Bulirsch and Stoer [15], exploit a generalization of Richardson extrapolation to a sequence of crumblings of the integration interval. By using some pth order method, e.g. the basic Euler method, we compute the solution on an interval $[x_n, x_n + H]$ with ever shorter steps $h_i = H/n_i$ $(i = 1, 2, \ldots, k)$, where n_i are positive integers

$$n_1 < n_2 < \cdots < n_k.$$

For example, we compute the solution at $x_n + H$ in two steps of $H/2$, four steps of $H/4$, six steps of $H/6$, and so on. We obtain the quantities

$$\boldsymbol{T}_{i,1} = \boldsymbol{y}_{h_i}(x_0 + H), \quad i = 1, 2, \ldots, k.$$

The global error of the basic method of order p is of order p (p. 341) and has the asymptotic expansion $\boldsymbol{y}(x) - \boldsymbol{y}_h(x) = \boldsymbol{e}_p(x)h^p + \boldsymbol{e}_{p+1}(x)h^{p+1} + \cdots + \boldsymbol{e}_N(x)h^N + \mathcal{O}(h^{N+1})$. In the extrapolation method we eliminate the higher terms in this expansion by constructing an interpolation polynomial $\boldsymbol{P}(h) = \widehat{\boldsymbol{y}} - \boldsymbol{e}_p h^p - \boldsymbol{e}_{p+1}h^{p+1} - \cdots - \boldsymbol{e}_{p+k-2}h^{p+k-2}$, for which we require

$$\boldsymbol{P}(h_i) = \boldsymbol{T}_{i,1}, \quad i = j, j-1, \ldots, j-k+1. \tag{7.19}$$

We then "extrapolate to $h \to 0$" by using as the final result

$$\boldsymbol{P}(0) = \widehat{\boldsymbol{y}} = \boldsymbol{T}_{j,k}.$$

Each value $\boldsymbol{T}_{j,k}$ is an approximation of $\boldsymbol{y}(x_n + H)$ of order $p + k - 1$. Equations (7.19) constitute a system of k equations for k unknowns $\widehat{\boldsymbol{y}}, \boldsymbol{e}_p, \ldots, \boldsymbol{e}_{p+k-2}$ (which can even be solved analytically for $p = 1$). We generate the consecutive elements by

$$\boldsymbol{T}_{j,k+1} = \boldsymbol{T}_{j,k} + \frac{\boldsymbol{T}_{j,k} - \boldsymbol{T}_{j-1,k}}{n_j/n_{j-k} - 1}. \tag{7.20}$$

For example, with $n_1 = 1$ and $n_2 = 2$ we obtain $\boldsymbol{T}_{22} = \boldsymbol{T}_{21} + (\boldsymbol{T}_{21} - \boldsymbol{T}_{11})/(2^1 - 1)$, which is precisely Richardson's formula (7.16) for $p = 1$. By repeating the procedure for $j = 1, 2, \ldots$ and $k = 1, 2, \ldots, j$ we obtain a whole fan of estimates for the

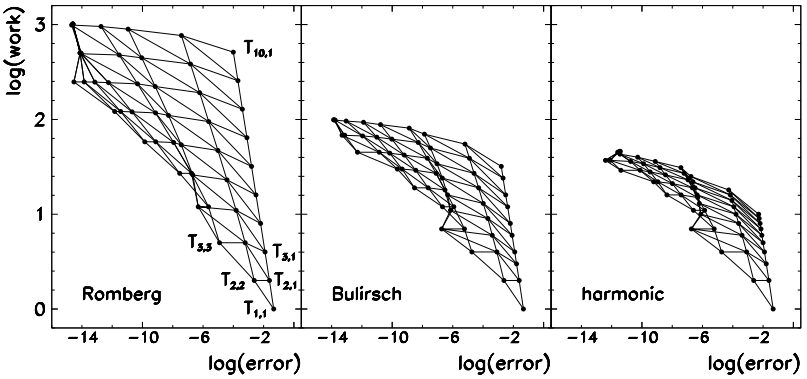


Fig. 7.4 Solving the differential equation $y' = (-y \sin x + 2 \tan x)y$ with the initial condition $y(\pi/6) = 2/\sqrt{3}$ by extrapolation methods on an interval of length $H = 0.2$. The basic method is the Euler explicit scheme (7.5). Shown is the numerical cost ("work") versus the solution error for three different sequences $\{n_i\}$. Some lines connecting the points $T_{j,k}$ overlap due to round-off errors appearing in the recurrence (7.20)

final value (each $\boldsymbol{T}_{j,k}$ is labeled by the corresponding order of the result):

$$
\begin{array}{lllll}
j \downarrow, k \rightarrow & \boldsymbol{T}_{1,1}(p) & & & \\
& \boldsymbol{T}_{2,1}(p) & \boldsymbol{T}_{2,2}(p+1) & & \\
& \boldsymbol{T}_{3,1}(p) & \boldsymbol{T}_{3,2}(p+1) & \boldsymbol{T}_{3,3}(p+2) & \\
& \dots & \dots & \dots & \dots
\end{array}
$$

Various sequences of factors n_i by which the integration interval is fragmented, are available, e.g. the Romberg $1, 2, 4, 8, 16, 32, \dots$ or the optimized Bulirsch sequence $1, 2, 3, 4, 6, 8, 12, 16, 24, 32, \dots$ (powers 2^k and $1.5 \cdot 2^k$); the latter is more economical in higher orders of the method. Recently the harmonic sequence $1, 2, 3, 4, 5, 6, \dots$ has gained popularity (see Fig. 7.4).

At low precisions the extrapolation integrators are just as efficient as Runge–Kutta methods. But they truly prosper in the regime of precision computations: by sequential construction of the elements $\boldsymbol{T}_{j,k}$ we can, in principle, attain arbitrarily high orders, which means that they can also be arbitrarily faster than fixed-order methods. However, extrapolation methods are not suited for solving equations with singularities on the integration interval.

Extrapolation methods also permit the control of step size H. Moreover, they allow us to change the *order* p of the basic scheme along the integration interval. A simultaneous choice of the optimal step size H and of the order p is cumbersome, but it can be done [2]. Adaptive extrapolation methods may "jump" in very long steps, which obviously calls for a reliable interpolation of the solution. If the basic method is the explicit or implicit Euler scheme, one can derive closed formulas for dense output in analogy to (7.11); see [16]. As an impression, Fig. 7.5 shows the comparison of one-step integrators: a few equidistant explicit methods of the

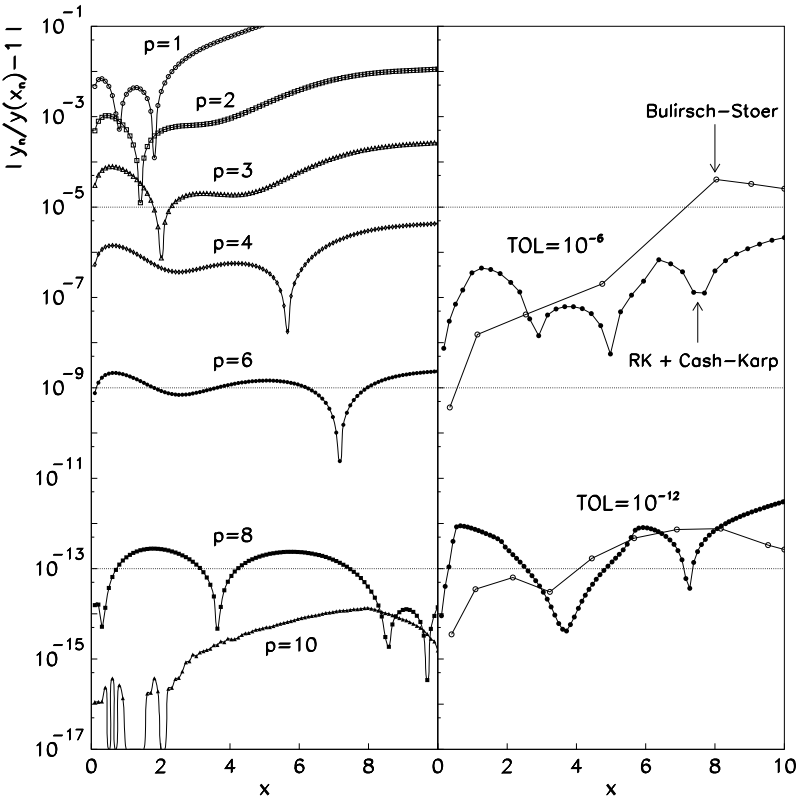


Fig. 7.5 Relative global error of integrators for $y'' = \mathrm{e}^x \sin x - y - y' = f(x, y, y')$ with initial conditions $y(0) = 1$, $y'(0) = 0$. [*Left*] The error of the basic Euler ($p = 1$), improved Euler ($p = 2$), Kutta ($p = 3$), Runge–Kutta ($p = 4$), Butcher ($p = 6$), and Hairer explicit methods with $p = 8$ and $p = 10$, without step size control on a mesh of 100 points. The errors can be further reduced at all orders if we allow for more evaluations of the function f. [*Right*] The error of the adaptive RK4 (Cash–Karp) method and the Bulirsch–Stoer extrapolation method for two tolerances. Here comparable errors are attained with far fewer evaluations of f and at fewer mesh points

RK type, the adaptive embedded RK4 method (Cash–Karp), and the Bulirsch–Stoer extrapolation method.

7.7 Multi-Step Methods ⋆

This section is devoted to the basics of multi-step methods, in particular the predictor–corrector (PC) methods and the backward differentiation methods for equations of the type $\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y})$. Together with symplectic integrators (Sect. 7.12) they play an important role in the integration of orbits in astronomy.

One characteristic of a multi-step method is that in order to initialize it, solutions $\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots$ at $x_0, x_1, \ldots$ need to be known already. They can be computed by one of the single-step methods, but its order should at least match the order of the multi-step method they feed. We integrate (7.2) on $[x_n, x_{n+1}]$,

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \int_{x_n}^{x_{n+1}} \boldsymbol{f}\big(\xi, \boldsymbol{y}(\xi)\big)\,\mathrm{d}\xi, \tag{7.21}$$

and approximate $\boldsymbol{f}$ by the Newton interpolation polynomial through the points at which we already know the solution values, up to including x_n. We get [2]

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h \sum_{j=0}^{k-1} \left[(-1)^j \int_0^1 \binom{-s}{j} \mathrm{d}s \right] \nabla^j \boldsymbol{f}_n. \tag{7.22}$$

We have used the usual backward differences $\nabla^{j+1} \boldsymbol{f}_n = \nabla^j \boldsymbol{f}_n - \nabla^j \boldsymbol{f}_{n-1}$, where $\nabla^0 \boldsymbol{f}_n = \boldsymbol{f}_n$ and $\boldsymbol{f}_n = \boldsymbol{f}(x_n, \boldsymbol{y}_n)$. It follows that

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\left[1 + \frac{1}{2}\nabla + \frac{5}{12}\nabla^2 + \frac{3}{8}\nabla^3 + \frac{251}{720}\nabla^4 + \frac{95}{288}\nabla^5 + \cdots\right]\boldsymbol{f}_n.$$

If we express the backward differences by function values, we obtain explicit Adams methods of various orders. The order depends on the number of points spanned by the interpolation polynomial. From the third order upwards we have

$$\begin{aligned}
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{h}{12}(23\boldsymbol{f}_n - 16\boldsymbol{f}_{n-1} + 5\boldsymbol{f}_{n-2}) + \mathcal{O}\big(h^4\big),\\
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{h}{24}(55\boldsymbol{f}_n - 59\boldsymbol{f}_{n-1} + 37\boldsymbol{f}_{n-2} - 9\boldsymbol{f}_{n-3}) + \mathcal{O}\big(h^5\big),\\
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{h}{720}(1901\boldsymbol{f}_n - 2774\boldsymbol{f}_{n-1} + 2616\boldsymbol{f}_{n-2}\\
&\quad - 1274\boldsymbol{f}_{n-3} + 251\boldsymbol{f}_{n-4}) + \mathcal{O}\big(h^6\big),\\
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{h}{1440}(4277\boldsymbol{f}_n - 7923\boldsymbol{f}_{n-1} + 9982\boldsymbol{f}_{n-2}\\
&\quad - 7298\boldsymbol{f}_{n-3} + 2877\boldsymbol{f}_{n-4} - 475\boldsymbol{f}_{n-5}) + \mathcal{O}\big(h^7\big).
\end{aligned} \tag{7.23}$$

If the interpolation polynomial $\boldsymbol{f}$ also includes the point x_{n+1} at which the solution $\boldsymbol{y}_{n+1}$ is not yet known, we obtain

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\left[1 - \frac{1}{2}\nabla - \frac{1}{12}\nabla^2 - \frac{1}{24}\nabla^3 - \frac{19}{720}\nabla^4 - \frac{3}{160}\nabla^5 + \cdots\right]\boldsymbol{f}_{n+1}.$$

This leads to a class of implicit Adams methods, for example,

$$\begin{aligned}
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{h}{12}(5\boldsymbol{f}_{n+1} + 8\boldsymbol{f}_n - \boldsymbol{f}_{n-1}) + \mathcal{O}(h^4),\\
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{h}{24}(9\boldsymbol{f}_{n+1} + 19\boldsymbol{f}_n - 5\boldsymbol{f}_{n-1} + \boldsymbol{f}_{n-2}) + \mathcal{O}(h^5),\\
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{h}{720}\big(251\boldsymbol{f}_{n+1} + 646\boldsymbol{f}_n - 264\boldsymbol{f}_{n-1}\\
&\qquad + 106\boldsymbol{f}_{n-2} - 19\boldsymbol{f}_{n-3}\big) + \mathcal{O}(h^6),\\
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{h}{1440}\big(475\boldsymbol{f}_{n+1} + 1427\boldsymbol{f}_n - 798\boldsymbol{f}_{n-1}\\
&\qquad + 482\boldsymbol{f}_{n-2} - 173\boldsymbol{f}_{n-3} + 27\boldsymbol{f}_{n-4}\big) + \mathcal{O}(h^7).
\end{aligned} \tag{7.24}$$

The equations are implicit since the unknown solution $\boldsymbol{y}_{n+1}$ occurs on the left as well as on the right, in $\boldsymbol{f}_{n+1} = \boldsymbol{f}(x_{n+1}, \boldsymbol{y}_{n+1})$. We solve them iteratively.

7.7.1 Predictor–Corrector Methods

Explicit and implicit Adams methods can be merged into efficient procedures of various orders, known as *predictor–corrector methods* due to the characteristic sequence of approximations. In predicting $\boldsymbol{y}_{n+1}$ by one of the explicit formulas, e.g. (7.23), we rely on the extrapolation of the Newton polynomial to the point x_{n+1} which lies outside of the interpolation interval. We denote this prediction by

$$\mathrm{P}: \quad \boldsymbol{y}_{n+1}^{(\mathrm{P})},$$

and call it the *predictor* (P). Correspondingly, the predicted derivative

$$\mathrm{E}_1: \quad \boldsymbol{y}_{n+1}^{\prime(\mathrm{P})} = \boldsymbol{f}\big(x_{n+1}, \boldsymbol{y}_{n+1}^{(\mathrm{P})}\big) \tag{7.25}$$

is also questionable. We denote this step by E_1 (*evaluation*). In the next step we use one of the implicit methods, e.g. (7.24), which we solve by iteration. Especially for small h the iteration converges rapidly, so that instead of the unknown functions $\boldsymbol{f}_{n+1}$ we take the extrapolated derivatives (7.25). In this step we obtain the improved approximation for the solution, the *corrector* (C),

$$\mathrm{C}: \quad \boldsymbol{y}_{n+1}^{(\mathrm{C})}.$$

Finally, we improve the derivative,

$$\mathrm{E}_2: \quad \boldsymbol{y}_{n+1}^{\prime(\mathrm{C})} = \boldsymbol{f}\big(x_{n+1}, \boldsymbol{y}_{n+1}^{(\mathrm{C})}\big).$$

The usual operation cycle in one complete step from x_n to x_{n+1} is then $\mathrm{PE}_1\mathrm{CE}_2$ or $\mathrm{PE}_1(\mathrm{CE}_2)^m$ if the corrector step is executed in m iterations in the implicit part.

Local Error Estimate From the Taylor expansions of $\boldsymbol{y}_{n+1}$ around x_n and $\boldsymbol{y}_n$ around x_{n+1} simple local error estimates can be derived [17]. For the fourth-order pair of the Adams predictor and corrector (see (7.23) and (7.24)) we obtain

$$\boldsymbol{y}(x_n) - \boldsymbol{y}_n^{(\mathrm{P})} \approx \frac{251}{270}\big(\boldsymbol{y}_n^{(\mathrm{C})} - \boldsymbol{y}_n^{(\mathrm{P})}\big), \qquad \boldsymbol{y}(x_n) - \boldsymbol{y}_n^{(\mathrm{C})} \approx \frac{19}{270}\big(\boldsymbol{y}_n^{(\mathrm{P})} - \boldsymbol{y}_n^{(\mathrm{C})}\big),$$

if we assume that the fifth derivative of $\boldsymbol{f}$ is constant on $[x_n, x_{n+1}]$. This gives us the popular Adams–Bashforth–Moulton explicit method

$$\begin{aligned}
\boldsymbol{y}_{n+1}^{(\mathrm{P})} &= \boldsymbol{y}_n + \frac{h}{24}(55\boldsymbol{f}_n - 59\boldsymbol{f}_{n-1} + 37\boldsymbol{f}_{n-2} - 9\boldsymbol{f}_{n-3}),\\
\boldsymbol{y}_{n+1}^{*} &= \boldsymbol{y}_{n+1}^{(\mathrm{P})} + \frac{251}{270}\big(\boldsymbol{y}_n^{(\mathrm{C})} - \boldsymbol{y}_n^{(\mathrm{P})}\big),\\
\boldsymbol{y}_{n+1}^{(\mathrm{C})} &= \boldsymbol{y}_n + \frac{h}{24}\big(9\boldsymbol{f}_{n+1}^{*} + 19\boldsymbol{f}_n - 5\boldsymbol{f}_{n-1} + \boldsymbol{f}_{n-2}\big),\\
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_{n+1}^{(\mathrm{C})} + \frac{19}{270}\big(\boldsymbol{y}_{n+1}^{(\mathrm{P})} - \boldsymbol{y}_{n+1}^{(\mathrm{C})}\big),
\end{aligned}$$

where we have denoted $\boldsymbol{f}_{n+1}^{*} = \boldsymbol{f}(x_{n+1}, \boldsymbol{y}_{n+1}^{*})$.

Such simple error estimates that depend only on the function values in individual steps, are one of the charms of predictor–corrector methods. Similar formulas can be derived in the case when x_{n-1} and $\boldsymbol{y}_{n-1}$ are used in (7.21) instead of x_n and $\boldsymbol{y}_n$, which leads to explicit Nyström predictors and implicit Milne–Simpson correctors [2].

Predictor–corrector methods perform best in tracing smooth solutions, when precision requirements are relatively severe, and when the evaluation of the functions $\boldsymbol{f}$ is numerically expensive. In one predictor or corrector step the function $\boldsymbol{f}$ needs to be evaluated only once!

7.7.2 Stability of Multi-Step Methods

For simplicity we discuss here only scalar equations. A general k-step method has the form of a difference equation

$$\sum\nolimits_{i=0}^{k} \alpha_i y_{n+i} = h \sum\nolimits_{i=0}^{k} \beta_i f_{n+i}. \tag{7.26}$$

Stability is related to the generating polynomials for the coefficients of the method,

$$\rho(\zeta) = \sum\nolimits_{i=0}^{k} \alpha_i \zeta^i, \qquad \sigma(\zeta) = \sum\nolimits_{i=0}^{k} \beta_i \zeta^i.$$

When the differential equation $y' = 0$ is being solved and the solution of the corresponding difference equation (7.26) remains bounded, the multi-step method is said

to be *zero-stable*. This is equivalent to the requirement that all zeros of ρ lie within the unit circle or on it; in the latter case the zeros should be simple. Zero stability therefore concerns only the coefficients α_i, not β_i.

Example Explicit and implicit k-step Adams methods have the generating polynomial $\rho(\zeta) = \zeta^k - \zeta^{k-1} = \zeta^{k-1}(\zeta - 1)$ with a simple zero $\zeta = 1$ and a zero $\zeta = 0$ of multiplicity $(k-1)$, so they are zero-stable. The explicit Nyström and Milne–Simpson methods are also zero-stable, since for them $\rho(\zeta) = \zeta^k - \zeta^{k-2} = \zeta^{k-2}(\zeta - 1)(\zeta + 1)$. Nevertheless, the simple zero $\zeta = -1$ may cause problems with certain classes of differential equations [2].

For stiff differential problems of the form $y' = f(y)$ (Sect. 7.10) stability cannot be defined only in terms of the coefficients α_i, but β_i become relevant as well. In such problems we study the stability of a difference scheme for the problem $y' = \lambda y$ that can be understood as a linearization of $y' = f(y)$ in the vicinity of a fixed point. If the equation $y' = \lambda y$ is solved by (7.26), we get

$$\sum_{i=0}^{k} (\alpha_i - h\lambda\beta_i) y_{n+i} = 0.$$

In this case, stability depends on the roots of the equation $\rho(\zeta) - \mu\sigma(\zeta) = 0$, where $\mu = h\lambda$. The stability region of the method (7.26) is the set of complex values μ for which all zeros of the polynomial $\rho - \mu\sigma$ lie within the unit circle or on it, and in the latter case the zeros should be simple.

Example For the explicit Adams method (see (7.23), $k = 4$) we obtain

$$\rho(\zeta) - \mu\sigma(\zeta) = \zeta^4 - \zeta^3 - \mu\left(\frac{55}{24}\zeta^3 - \frac{59}{24}\zeta^2 + \frac{37}{24}\zeta - \frac{9}{24}\right) = 0,$$

from which we compute μ as a function of ζ. Let us write $\zeta = \mathrm{e}^{\mathrm{i}\phi}$ so that ζ goes around the whole unit circle for $0 \le \phi \le 2\pi$. When it does that, the variable μ outlines the edge of the stability region in the complex plane $(\mathrm{Re}\, h\lambda, \mathrm{Im}\, h\lambda)$ denoted by the index 4 in Fig. 7.6 (left). The figure also shows the stability regions of explicit methods of orders 1, 2, and 3. Figure 7.6 (right) shows the stability regions of implicit Adams methods.

Example Note that a higher order of the local error of the scheme does not necessarily imply better stability. As an example [2] we solve $y' = y$, $y(0) = 1$ (analytic solution $y(x) = \mathrm{e}^x$) by using a third-order difference method

$$y_{n+2} + 4y_{n+1} - 5y_n = h\big[4f(x_{n+1}, y_{n+1}) + 2f(x_n, y_n)\big] + \mathcal{O}\big(h^4\big),$$

which is in the form (7.26). The corresponding characteristic equation is

$$p(\zeta) - h\sigma(\zeta) = \zeta^2 + 4(1-h)\zeta - (5 + 2h) = 0,$$

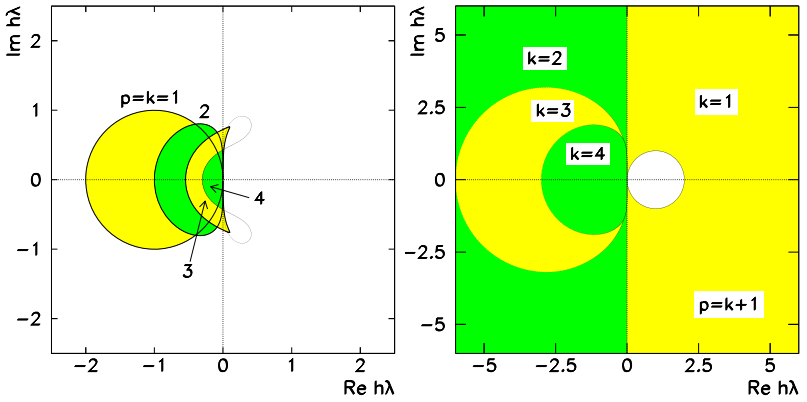


Fig. 7.6 Regions of linear stability of k-step explicit [*Left*] and implicit [*Right*] Adams methods. (Note the different axis ranges.) The order of the explicit and implicit methods is $p = k$ and $p = k + 1$, respectively. With increasing order the regions of stability (of both explicit and implicit methods) shrink

with the solutions $\zeta_1 = 1 + h + \mathcal{O}(h^2)$ and $\zeta_2 = -5 + \mathcal{O}(h)$. With the values at the first two points, $y_0 = 1$ and $y_1 = \mathrm{e}^h$, we obtain the solution $y_n = (1 + h)^n + (-5)^n$, which is obviously unstable at $n \to \infty$. As an exercise, plot the numerical solution computed with steps $h = 0.1$, 0.05, and 0.025!

7.7.3 Backward Differentiation Methods

Adams, Nyström, and Milne–Simpson methods are all based on the *integration of the right side* of the differential equation, as in (7.21). Instead, we could *differentiate the left side* of the equation and this is what we do in *backward differentiation formulas* (BDF), again by spanning an interpolation polynomial through a couple of known points. The most useful are implicit formulas generated by the series

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j \boldsymbol{y}_{n+1} = h \boldsymbol{f}_{n+1}$$

(compare to (7.22)). As an example we give the third- and fourth-order difference schemes:

$$\begin{aligned} \frac{11}{6}\boldsymbol{y}_{n+1} - 3\boldsymbol{y}_n + \frac{3}{2}\boldsymbol{y}_{n-1} - \frac{1}{3}\boldsymbol{y}_{n-2} &= h\boldsymbol{f}_{n+1} + \mathcal{O}\big(h^4\big), \\ \frac{25}{12}\boldsymbol{y}_{n+1} - 4\boldsymbol{y}_n + 3\boldsymbol{y}_{n-1} - \frac{4}{3}\boldsymbol{y}_{n-2} + \frac{1}{4}\boldsymbol{y}_{n-3} &= h\boldsymbol{f}_{n+1} + \mathcal{O}\big(h^5\big), \end{aligned} \tag{7.27}$$

while the formulas for orders $p = 1$, 2, 5, and 6 are listed in [2]. Backward differentiation formulas of orders $p > 6$ are not stable, while the methods of order

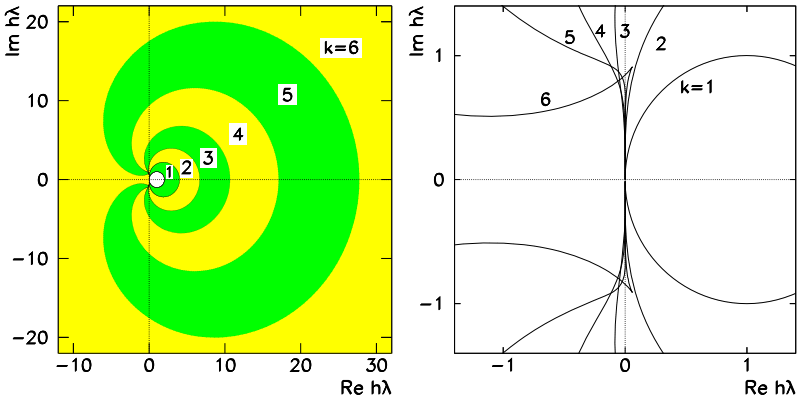


Fig. 7.7 [*Left*] Stability regions of implicit backward differentiation methods of order $p = k$. With increasing order the regions shrink: the stability region for the method of order p is the *exterior* of the geometric shape denoted by p. [*Right*] Zoom-in of the left panel near the origin. Only the first- and second-order methods are stable throughout the left complex half-plane (Sect. 7.9)

$1 \le p \le 6$ boast large, outwardly unbounded regions of linear stability (see Fig. 7.7 and compare the axis range to that in Fig. 7.6). The integrators of this subsection can also be implemented with a variable step size [2].

7.8 Conservative Second-Order Equations

Second-order vector differential equations of the form $\boldsymbol{y}'' = \boldsymbol{f}(x, \boldsymbol{y}, \boldsymbol{y}')$ are ubiquitous in physics: they embody every single instance of Newton's law $m\ddot{\boldsymbol{x}} = \boldsymbol{F}(t, \boldsymbol{x}, \dot{\boldsymbol{x}})$. It is interesting enough if only t and $\boldsymbol{x}$ occur at the right, or even only $\boldsymbol{x}$, as in

$$m\ddot{x} = F(x),$$

where m is constant and F is an arbitrary integrable function. Such equations appear when forces are independent of the velocities, as in undamped oscillators (where $F(x) = -kx$) or in the motion of bodies in the classical gravitational field. If m has units of mass and x units of length, the two terms of the first integral of this equation,

$$\frac{1}{2}m\dot{x}^2 - \int F(x)\,\mathrm{d}x = \text{const},$$

represent the kinetic energy $T(\dot{x})$ and the potential energy $U(x)$ (up to an additive constant) with the conservation law $T + U = E_0$ characteristic for conservative systems. It follows that $\dot{x} = \pm[2(E_0 - U(x))/m]^{1/2}$, and by integration we imme-

diately obtain the dependence of the spatial coordinate on time,

$$t = t_0 + \int_{x_0}^{x} \frac{\mathrm{d}\xi}{\dot{x}(\xi)} = t_0 \pm \int_{x_0}^{x} \frac{\mathrm{d}\xi}{\sqrt{\frac{2}{m}(E_0 - U(\xi))}}.$$

The integral can be computed analytically if U is a polynomial of at most fourth degree (and even then we are dealing with elliptic integrals). For other cases of such conservative equations we use numerical integration.

7.8.1 Runge–Kutta–Nyström Methods

Equations of the form $\boldsymbol{y}'' = \boldsymbol{f}(x, \boldsymbol{y}, \boldsymbol{y}')$ can be rewritten as systems of two first-order equations

$$\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{y}' \end{pmatrix}' = \begin{pmatrix} \boldsymbol{y}' \\ \boldsymbol{f}(x, \boldsymbol{y}, \boldsymbol{y}') \end{pmatrix},$$

with initial conditions $\boldsymbol{y}(x_0) = \boldsymbol{y}_0$, $\boldsymbol{y}'(x_0) = \boldsymbol{y}'_0$. Such systems can be solved by using any method from previous sections. But a substantial simplification can be achieved for equations without first derivatives $\boldsymbol{y}'' = \boldsymbol{f}(x, \boldsymbol{y})$ (Newton's law with velocity-independent forces). A special class of Runge–Kutta schemes exists for them [2], the Runge–Kutta–Nyström methods (RKN):

$$\boldsymbol{k}'_i = \boldsymbol{f}\left(x_n + c_i h, \boldsymbol{y}_n + c_i h \boldsymbol{y}'_n + h^2 \sum_{j=1}^{m} \overline{a}_{ij} \boldsymbol{k}'_j\right),$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{y}'_n + h^2 \sum_{i=1}^{m} \overline{b}_i \boldsymbol{k}'_i,$$

$$\boldsymbol{y}'_{n+1} = \boldsymbol{y}'_n + h \sum_{i=1}^{m} b_i \boldsymbol{k}'_i,$$

where m is the number of stages. We determine the parameters $\overline{a}_{ij}$, b_i, $\overline{b}_i$, and c_i such that the order of the method is as high as possible, analogously to the methods of the core RK family. The very useful RKN method of the fifth order,

c_i	0				$\overline{a}_{ij}$
	$\frac{1}{5}$	$\frac{1}{50}$			
	$\frac{2}{3}$	$-\frac{1}{27}$	$\frac{7}{27}$		
	1	$\frac{3}{10}$	$-\frac{2}{35}$	$\frac{9}{35}$	
	$\overline{b}_i$	$\frac{14}{336}$	$\frac{100}{366}$	$\frac{54}{336}$	0
	$\overline{b}_i$	$\frac{14}{336}$	$\frac{125}{366}$	$\frac{162}{336}$	$\frac{35}{336}$

requires only *four* evaluations of the function $\boldsymbol{f}$, while in the standard RK methods at least six are needed for the same order. The RKN methods also allow for local error control by adjusting the step size: this can be done either by extrapolation or within an embedded scheme (Sect. 7.4.4). The RKN method of order seven can be found in [18], and the methods of orders from 8 to 11 in [19].

7.8.2 Multi-Step Methods

Multi-step methods of Sect. 7.7 can also be generalized to second-order conservative equations. We integrate the equation $\boldsymbol{y}'' = \boldsymbol{f}(x, \boldsymbol{y})$ twice [2] and obtain

$$\begin{aligned}&\boldsymbol{y}(x+h) - 2\boldsymbol{y}(x) + \boldsymbol{y}(x-h)\\ &\quad = h^2 \int_0^1 (1-s)\big[\boldsymbol{f}\big(x+sh, \boldsymbol{y}(x+sh)\big) + \boldsymbol{f}\big(x-sh, \boldsymbol{y}(x-sh)\big)\big]\,\mathrm{d}s.\end{aligned}$$

Then we replace the function $\boldsymbol{f}$ by an interpolation polynomial. If $(x_n, \boldsymbol{y}_n)$ is the last point touched by the polynomial, we obtain the Störmer family of explicit methods

$$\boldsymbol{y}_{n+1} - 2\boldsymbol{y}_n + \boldsymbol{y}_{n-1} = h^2\left[1 + \frac{1}{12}\nabla^2 + \frac{1}{12}\nabla^3 + \frac{19}{240}\nabla^4 + \frac{3}{40}\nabla^5 + \cdots\right]\boldsymbol{f}_n,$$

while if also the point $(x_{n+1}, \boldsymbol{y}_{n+1})$ is included, we obtain the Cowell family

$$\boldsymbol{y}_{n+1} - 2\boldsymbol{y}_n + \boldsymbol{y}_{n-1} = h^2\left[1 - \nabla + \frac{1}{12}\nabla^2 - \frac{1}{240}\nabla^4 - \frac{1}{240}\nabla^5 + \cdots\right]\boldsymbol{f}_{n+1}.$$

Note that the third backward difference is absent in this expression, which results in a very useful fifth-order implicit Numerov method

$$\boldsymbol{y}_{n+1} - 2\boldsymbol{y}_n + \boldsymbol{y}_{n-1} = \frac{h^2}{12}\big[\boldsymbol{f}_{n+1} + 10\boldsymbol{f}_n + \boldsymbol{f}_{n-1}\big] + \mathcal{O}\big(h^6\big).$$

The stability criteria for the methods mentioned above are described in [2].

7.9 Implicit Single-Step Methods

For stiff problems discussed in Sect. 7.10 we need difference schemes where scalar equations $y' = \lambda y$ ($\lambda \in \mathbb{C}$) would enjoy stability (bounded numerical solutions) at least in the whole left complex half-plane, i.e. for $\mathrm{Re}(h\lambda) \le 0$ with $h > 0$. Such methods are called A-stable (Fig. 7.8).

The A-stability criterion is satisfied by the methods of the Runge–Kutta type. Implicit methods are given by (7.8) and (7.9) in which $a_{ij} \neq 0$ for $j \ge i$. Let us

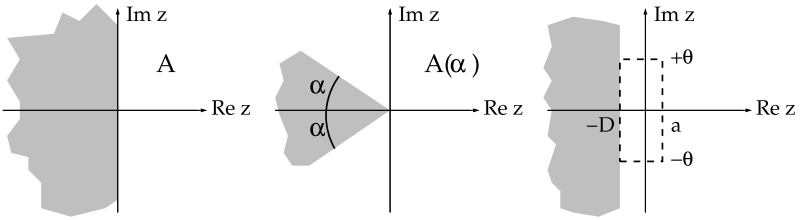


Fig. 7.8 Definitions of linear stability regions (scalar problem $y' = \lambda y$) for implicit methods, where $z = h\lambda$. [*Left*] A-stability for single-step methods. [*Center*] $A(\alpha)$-stability for single-step methods. [*Right*] "Stiff stability" for multi-step methods

replace the quantities $\boldsymbol{k}_i$ by the argument $\boldsymbol{g}_i$ such that $\boldsymbol{k}_i = \boldsymbol{f}(x_i, \boldsymbol{g}_i)$ [20, 21]! The implicit Runge–Kutta methods can then be written in the form

$$\boldsymbol{g}_i = \boldsymbol{y}_n + h \sum_{j=1}^{m} a_{ij} \boldsymbol{f}(x_n + c_j h, \boldsymbol{g}_j), \tag{7.28}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h \sum_{j=1}^{m} b_j \boldsymbol{f}(x_n + c_j h, \boldsymbol{g}_j), \tag{7.29}$$

where m is the number of stages and $1 \le i \le m$. Equation (7.28) immediately discloses the price we have to pay for stability: at each step we need to solve a system of m (in general, non-linear, vector) equations for the quantities $\boldsymbol{g}_i$.

The simplest one-step method is the *implicit Euler method* of the first order. As its foundation we take the explicit Euler formula (7.5), but now we evaluate the right side of the equation at the next point, thus

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h \boldsymbol{f}(x_{n+1}, \boldsymbol{y}_{n+1}). \tag{7.30}$$

With the implicit method for the equation $y' = \lambda y$ we got $y_{n+1} = (1 + h\lambda) y_n$; with the implicit one we obtain $y_{n+1} = (1 - h\lambda)^{-1} y_n$. Now the region of absolute linear stability has become the *exterior* of the circle in the complex plane in Fig. 7.9 (left) and is outwardly *unbounded*: the implicit Euler method is therefore more than A-stable, since its stability region encompasses the whole left and almost the whole right complex half-plane. The second-order *implicit midpoint method*

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h \boldsymbol{f}\left(\frac{x_n + x_{n+1}}{2}, \frac{\boldsymbol{y}_n + \boldsymbol{y}_{n+1}}{2} \right) \tag{7.31}$$

and the *implicit trapezoidal method*

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{h}{2} \big(\boldsymbol{f}(x_n, \boldsymbol{y}_n) + \boldsymbol{f}(x_{n+1}, \boldsymbol{y}_{n+1}) \big)$$

are also A-stable: their stability region is precisely the half-plane $\mathrm{Re}\, h\lambda < 0$.

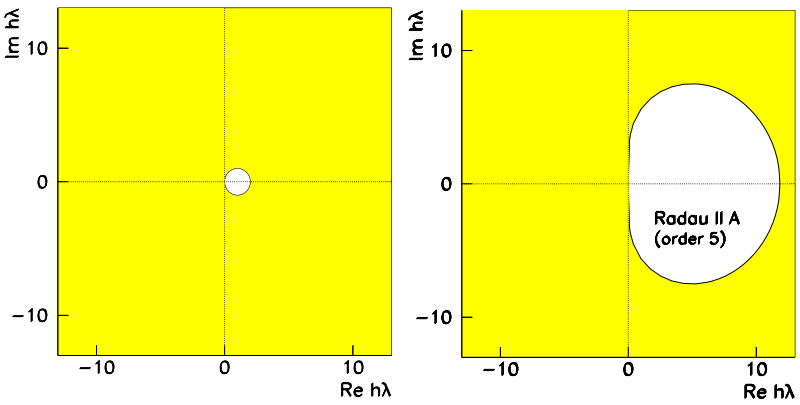


Fig. 7.9 Linear stability regions of implicit RK methods. [*Left*] Implicit Euler method corresponds to the whole exterior of the circle centered at $h\lambda = 1$ with radius 1 (compare to Fig. 7.3). [*Right*] The shaded exterior of the oval is the stability region of the fifth-order "Radau 5" method from the Radau II A class [22]. The stability region of the implicit midpoint, trapezoidal, and the sixth-order "Gauss 6" method is the whole half-plane $\mathrm{Re}\, h\lambda < 0$. All these methods are A-stable

Radau 5 and Gauss 6 Methods The orders of the three implicit schemes mentioned above are too low for the integration of stiff problems. For serious use we recommend the implicit fifth-order Radau 5 method [22] from the Radau II A family (for details see [20]). Its coefficients are given by the Butcher tableau

$$\begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array}$$

This method is also A-stable (Fig. 7.9 (right)). We complete the collection by the sixth-order Gauss 6 method from the Gauss family (details in [20]). Apart from A-stability this method possesses other important properties relevant for geometric integration discussed in Sect. 7.12. Its coefficients are given in the tableau

$$\begin{array}{c|ccc} \frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\ \frac{1}{2}+\frac{\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}+\frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \end{array}$$

Extensive stability regions are one of the main attractions of implicit methods. Yet all implicit methods are not suitable for stiff problems, nor for geometric integration. The coefficients of other popular schemes are listed in [20]. For non-linear

equations the theory of linear A-stability needs to be generalized to B-stability (see Appendix F).

7.9.1 Solution by Newton's Iteration

Implicit Runge–Kutta methods are more demanding than the explicit ones, as in general they call for solution of systems of non-linear equations. The main numerical obstacle is the system (7.28) that can be treated by Newton's iteration [20]. We introduce

$$\boldsymbol{z}_i = \boldsymbol{g}_i - \boldsymbol{y}_n,$$

which alleviates the influence of round-off errors. Then (7.28) becomes

$$\boldsymbol{z}_i = h \sum_{j=1}^{m} a_{ij} \boldsymbol{f}(x_n + c_j h, \boldsymbol{y}_n + \boldsymbol{z}_j).$$

The Jacobi matrix used in the iteration is approximated by

$$J = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(x_n + c_j h, \boldsymbol{y}_n + \boldsymbol{z}_j) \approx \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(x_n, \boldsymbol{y}_n),$$

and assemble the solution in the kth step in the vector

$$\boldsymbol{Z}^{(k)} = \left(\boldsymbol{z}_1^{(k)\mathrm{T}}, \boldsymbol{z}_2^{(k)\mathrm{T}}, \ldots, \boldsymbol{z}_m^{(k)\mathrm{T}}\right)^{\mathrm{T}}.$$

Newton's iteration has the form

$$\begin{aligned} (I - hA \otimes J)\Delta \boldsymbol{Z}^{(k)} &= -\boldsymbol{Z}^{(k)} + h(A \otimes I)\boldsymbol{F}\left(\boldsymbol{Z}^{(k)}\right), \\ \boldsymbol{Z}^{(k+1)} &= \boldsymbol{Z}^{(k)} + \Delta \boldsymbol{Z}^{(k)}, \end{aligned} \tag{7.32}$$

where

$$I - hA \otimes J = \begin{pmatrix} I - ha_{11}J & \cdots & -ha_{1m}J \\ \vdots & & \vdots \\ -ha_{m1}J & \cdots & I - ha_{mm}J \end{pmatrix},$$

$$A \otimes I = \begin{pmatrix} a_{11}I & \cdots & a_{1m}I \\ \vdots & & \vdots \\ a_{m1}I & \cdots & a_{mm}I \end{pmatrix},$$

and where we have denoted

$$\boldsymbol{F}\left(\boldsymbol{Z}^{(k)}\right) = \left(\boldsymbol{f}^{\mathrm{T}}\left(x_n + c_1 h, \boldsymbol{y}_n + \boldsymbol{z}_1^{(k)}\right), \ldots, \boldsymbol{f}^{\mathrm{T}}\left(x_n + c_m h, \boldsymbol{y}_n + \boldsymbol{z}_m^{(k)}\right)\right)^{\mathrm{T}}.$$

In each iteration the function $\boldsymbol{f}$ needs to be computed m times, and a linear system with a $Mm \times Mm$ matrix solved, where M is the number of equations in the system. The simplest, but also the worst, initial approximation for Newton's iteration is $\boldsymbol{Z}^{(0)} = \boldsymbol{0}$. We control the convergence by monitoring the ratio

$$\zeta^{(k)} = \left\| \Delta \boldsymbol{Z}^{(k)} \right\| / \left\| \Delta \boldsymbol{Z}^{(k-1)} \right\|.$$

With the prescribed local error TOL we terminate the iteration when

$$\frac{\zeta^{(k)}}{1 - \zeta^{(k)}} \left\| \Delta \boldsymbol{Z}^{(k)} \right\| \le \kappa \cdot \mathrm{TOL},$$

and accept the values $\boldsymbol{Z}^{(k+1)}$ as the final outcome. Experience shows that sensible values κ are approximately between 0.01 and 0.1. Numerous further instructions on effective solution of the system (7.32) can be found in [20].

When all $\boldsymbol{z}_i$ have been obtained by Newton's iteration, we need to compute the next solution $\boldsymbol{y}_{n+1}$ by using (7.29). At first sight it appears as if $\boldsymbol{f}$ needs to be evaluated yet m more times, but this can be avoided. Namely, (7.29) can be rewritten as

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \sum_{i=1}^{m} d_i \boldsymbol{z}_i,$$

where $(d_1, d_2, \ldots, d_m) = (b_1, b_2, \ldots, b_m) A^{-1}$. For example, for the Radau 5 method (p. 360) we have simply $d_1 = d_2 = 0$ and $d_3 = 1$.

7.9.2 Rosenbrock Linearization

If an autonomous differential equation $\boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{y})$ is non-linear, even implicit schemes may become unstable, even if they are stable with respect to the linear problem $\boldsymbol{y}' = A\boldsymbol{y}$. Moreover, implicit equations can generally be solved only iteratively, so convergence problems may appear. With *Rosenbrock methods* we try to circumvent these problems by *linearization*. First we linearize the auxiliary quantities $\boldsymbol{k}_i$ in the m-stage implicit RK method,

$$\boldsymbol{k}_i = h\boldsymbol{f}\Bigg(\underbrace{\boldsymbol{y}_n + \sum_{j=1}^{i-1} a_{ij}\boldsymbol{k}_j}_{\boldsymbol{g}_i} + a_{ii}\boldsymbol{k}_i\Bigg) \approx h\boldsymbol{f}(\boldsymbol{g}_i) + ha_{ii} \underbrace{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(\boldsymbol{g}_i)}_{J_y(\boldsymbol{g}_i)} \boldsymbol{k}_i,$$

where $1 \le i \le m$. The computing cost at the right can be reduced if the true Jacobi matrix $J_y(\boldsymbol{g}_i)$ is replaced by the approximate one, $J_y^{(n)} = J_y(\boldsymbol{y}_n)$, which needs to be computed only once at each step. On the other hand, the ansatz can be enriched by

adding a few linear combinations of $J_y(\boldsymbol{y}_n)\boldsymbol{k}_i$ at the right. With this small additional expense a m-stage Rosenbrock method takes the form

$$\left[I - h\gamma_{ii} J_y^{(n)}\right]\boldsymbol{k}_i = h\boldsymbol{f}\left(\boldsymbol{y}_n + \sum_{j=1}^{i-1} \alpha_{ij}\boldsymbol{k}_j\right) + hJ_y^{(n)} \sum_{j=1}^{i-1} \gamma_{ij}\boldsymbol{k}_j, \tag{7.33}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \sum_{j=1}^{s} b_j \boldsymbol{k}_j, \tag{7.34}$$

where $i = 1, 2, \ldots, m$. Again the coefficients α_{ij}, γ_{ij}, and c_i are tuned such that the maximum order of the method is attained. Just like in the non-linearized implicit RK methods, here too at each stage of the nth step we have to solve a system of m equations for the unknowns $\boldsymbol{k}_i$, with the matrix $I - h\gamma_{ii} J_y^{(n)} = I - h\gamma_{ii}[\partial \boldsymbol{f}/\partial \boldsymbol{y}](\boldsymbol{y}_n)$, except that now the system is linear.

Non-autonomous Equations Rosenbrock methods were devised for autonomous problems $\boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{y})$. Like in other approaches they can be generalized to non-autonomous problems $\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y})$ by adding the equation $x' = 1$ to the system. In practical implementations (NUMERICAL RECIPES, NAG) this has already been taken care of. We can explicitly solve (7.33) for x and thus obtain a system of linear equations for the quantities $\boldsymbol{k}_i$:

$$\boldsymbol{k}_i = h\boldsymbol{f}\left(x_n + \alpha_i h, \boldsymbol{y}_n + \sum_{j=1}^{i-1} \alpha_{ij}\boldsymbol{k}_j\right) + hJ_y^{(n)} \sum_{j=1}^{i} \gamma_{ij}\boldsymbol{k}_j + h^2\gamma_i \boldsymbol{J}_x^{(n)}.$$

The concluding part of each time step—(7.34)—remains the same. Above we have used the abbreviations

$$\alpha_i = \sum_{j=1}^{i-1} \alpha_{ij}, \qquad \gamma_i = \sum_{j=1}^{i} \gamma_{ij}, \qquad \boldsymbol{J}_x^{(n)} = \frac{\partial \boldsymbol{f}}{\partial x}(x_n, \boldsymbol{y}_n), \qquad J_y^{(n)} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(x_n, \boldsymbol{y}_n).$$

Implicit Differential Equations Implicit equations of the form $M\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y})$ with constant non-singular matrices M can be rewritten in the equivalent form $\boldsymbol{y}' = M^{-1}\boldsymbol{f}(x, \boldsymbol{y})$, resulting in

$$M\boldsymbol{k}_i = h\boldsymbol{f}\left(x_n + \alpha_i h, \boldsymbol{y}_n + \sum_{j=1}^{i-1} \alpha_{ij}\boldsymbol{k}_j\right) + hJ_y^{(n)} \sum_{j=1}^{i} \gamma_{ij}\boldsymbol{k}_j + h^2\gamma_i \boldsymbol{J}_x^{(n)}.$$

By simple transformations among the variables and by exploiting the banded structure of the matrices M and J_y, the numerical efficiency of Rosenbrock methods can be strongly enhanced [20]. In similar ways, problems with non-constant matrices $M(x)$ can be harnessed [23].

7.10 Stiff Problems

We are referring to stiff problems with ordinary differential equations when two very different scales are involved in the solution, for example, two characteristic times. As an illustration, think of a robotic arm that moves with angular velocities of $\approx$1 Hz and we strike the arm by a small hammer, exciting oscillations with a frequency of $\approx$100 Hz. We wish to simulate the movement of this arm by numerical integration of the equations of motion. A variable step-size integrator will be forced to reduce the step size in order to faithfully trace the physically totally irrelevant high-frequency component that dies out almost immediately; at the same time, it will apply almost imperceptible corrections to the slowly changing low-frequency solution.

Stiffness for *linear systems* can be quantified by a criterion for the eigenvalues λ_i of the Jacobi matrix corresponding to the model problem $\boldsymbol{y}' = A\boldsymbol{y}$ from the definition (7.17). A problem is stiff when

$$\max_{1\le i\le n} |\operatorname{Re}\lambda_i| \gg \min_{1\le i\le n} |\operatorname{Re}\lambda_i|. \tag{7.35}$$

The ratio R between the left and right side of this inequality is mostly a good measure of stiffness. If some eigenvalue is zero, we obtain an infinite R: in such cases the problem is not stiff if other eigenvalues are small. The definition of stiffness by (7.35) is inappropriate for *non-linear systems*. We therefore define stiffness from a practical computational standpoint. A problem is stiff on an interval if the integrator is forced to use a step size h which is incommensurately small with respect to the slow change of the solution on that interval.

Example The famous Robertson chemical kinetics problem involves compounds A, B, and C linked by a chain of reactions with very different reaction rates,

$$\begin{aligned} A &\longrightarrow B && \text{proceeds slowly } (0.04),\\ B+B &\longrightarrow B+C && \text{very fast } (3\cdot 10^7),\\ B+C &\longrightarrow A+C && \text{fast } (10^4). \end{aligned}$$

The processes are described by a system of differential equations

$$\begin{aligned} A\colon\quad & y_1' = -0.04y_1 + 10^4 y_2 y_3, && y_1(0)=1,\\ B\colon\quad & y_2' = 0.04y_1 - 10^4 y_2 y_3 - 3\cdot 10^7 y_2^2, && y_2(0)=0,\\ C\colon\quad & y_3' = 3\cdot 10^7 y_2^2, && y_3(0)=0. \end{aligned} \tag{7.36}$$

It turns out that the solution y_2 achieves its maximum $y_2(x)\approx 0.0000365$ at $x\approx 0.0045$, while y_3 still rapidly increases there, and $y_1\approx 1$. The Jacobi matrix of partial derivatives (7.3) for the system (7.36) around that point is

$$\begin{pmatrix} -0.04 & 10^4 y_3 & 10^4 y_2\\ 0.04 & -10^4 y_3 - 6\cdot 10^7 y_2 & -10^4 y_2\\ 0 & 6\cdot 10^7 y_2 & 0 \end{pmatrix} \approx \begin{pmatrix} -0.04 & 1.434 & 0.365\\ 0.04 & -2191 & -0.365\\ 0 & 2189 & 0 \end{pmatrix},$$

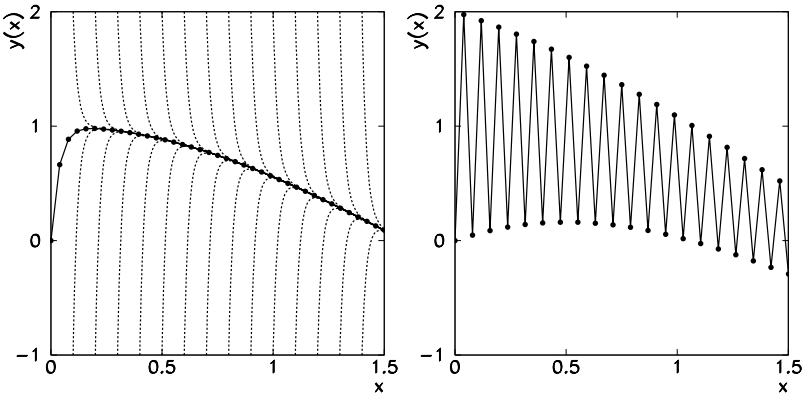


Fig. 7.10 Solving the stiff differential equation $y' = -50(y - \cos x)$ with the initial condition $y(0) = 0$. [*Left*] Families of solutions with initial approximations along $y = 2$ and $y = -1$ (*dotted curves*) and the solution by the implicit Euler method with step $h = 0.0395$. [*Right*] Solution by the explicit Euler method with step $h = 0.0395$

with eigenvalues $\lambda_1 \approx 0$, $\lambda_2 \approx -0.405$, $\lambda_3 \approx -2190.3$. Explicit methods fail here. Why? We find the answer in Fig. 7.3 (right) for the Dormand–Prince 5(4) method. Its stability region for $\mathrm{Re}\, h\lambda < 0$ reaches to $\mathrm{Re}\, h\lambda \approx -3.3$. Stability is guaranteed for $-3.3 \le -2190h$ or $h \le 0.0015$. The problem (7.36) is therefore stiff because of the eigenvalue λ_3 which is much larger than the rest. An explicit integrator will have to use a step this small in order to trace the solution that is very humble on almost the whole integration interval.

Figure 7.10 shows the numerical solution of the stiff problem [24]

$$y' = -50(y - \cos x), \quad y(0) = 0,$$

by using the explicit and implicit Euler method. The solution is the smooth curve shown in the center of the left panel. After transitory increases or decreases all other solutions with different initial conditions converge to this solution. Such *transients* are typical for stiff problems. The implicit method nicely follows the solution; the explicit method (right panel) with the same step size hesitatingly dances around it. As a rule, stiff problems should therefore be integrated by implicit schemes; implicitness provides the stabilization.

Example (7.36) is not very terrifying. For numerical modeling of the concentration of long-lived compounds in the Earth's atmosphere the integrators must handle intervals on the order of centuries, while the time scales of some chemical processes are up to 17 orders of magnitude smaller. The integrator must therefore reliably detect stiffness, and the choice of the appropriate step size becomes critical. Precise instructions can be found in [20], pp. 123–127.

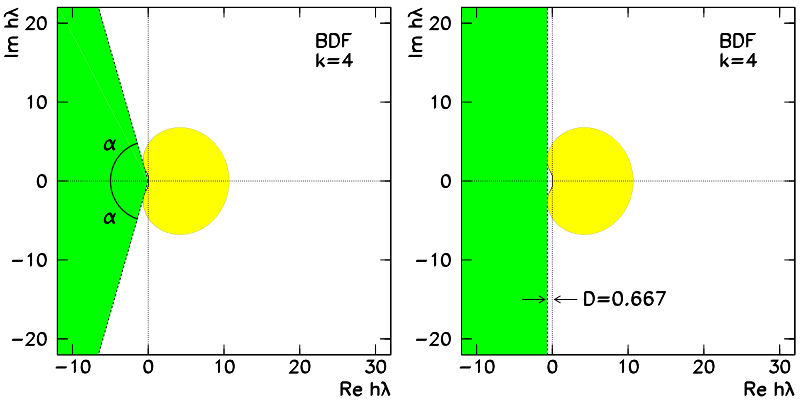


Fig. 7.11 Stability of the fourth-order BDF method (see (7.27)). [*Left*] $A(\alpha)$-stability with angle $\alpha = 73.35°$. [*Right*] Stiff stability with $D = 0.667$. See also Fig. 7.7

7.11 Implicit Multi-Step Methods ⋆

So far we have encountered only single-step implicit methods. Stiff problems can also be efficiently solved by using multi-step methods, whose stability was discussed in Sect. 7.7. Stability regions of multi-step methods are large, but any A-stable implicit multi-step method can have at most order $p = 2$ [25]. This constraint, known as the *Dahlquist barrier*, is clearly shown by Fig. 7.6 (right) for implicit Adams methods and Fig. 7.7 (right) for BDF methods.

The low order of A-stable implicit multi-step methods diminishes their numerical precision. Yet not all stiff problems require stability on the whole half-plane $\mathrm{Re}\, h\lambda < 0$. Moreover, solutions of stiff equations with eigenvalues near the imaginary axis strongly oscillate and the step size needs to be reduced anyway in order to reproduce the high-frequency solution components. We therefore relax the requirements of strict A-stability and settle for $A(\alpha)$-*stable methods* which are stable at least on the slices defined by $|\arg(-z)| \le \alpha$ with $\alpha \le \pi/2$ (Fig. 7.8 (center)). A "completely" A-stable method is therefore $A(\pi/2)$-stable. We also define "stiff stability" for which we require the stability region $\mathrm{Re}\, z < -D$ for some $D > 0$ and a "sufficient precision" of the method within the rectangle $-D \le \mathrm{Re}\, z \le a$, $-\theta \le \mathrm{Im}\, z \le \theta$ for some $a > 0$ and $\theta \sim \pi/5$ (Fig. 7.8 (right)).

BDF methods of orders $3 \le p \le 6$ (Fig. 7.7) correspond to $\alpha = 86.03°$, $D = 0.083$ $(k = 3)$, $\alpha = 73.35°$, $D = 0.667$ $(k = 4)$, $\alpha = 51.84°$, $D = 2.327$ $(k = 5)$, $\alpha = 17.84°$, $D = 6.075$ $(k = 6)$. Figure 7.11 shows the region of $A(\alpha)$-stability and stiff stability for the fourth-order method (7.27). $A(\alpha)$-stable multi-step methods of higher orders with $\alpha \lesssim \pi/2$ do exist, but they possess large leading error constants and they are only of limited use. In order to cross the Dahlquist barrier more general multi-step methods can be devised: see [20].

7.12 Geometric Integration ⋆

In the following we use the variable pair $(t, \boldsymbol{y})$ instead of $(x, \boldsymbol{y})$, in the spirit of dynamical analysis pervading this section. We are interested in the solutions of equations $\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y})$ where $\dot{}$ denotes the time derivative, especially in the context of autonomous Hamiltonian systems [26]. Such systems are described by the Hamiltonians $H(\boldsymbol{p}, \boldsymbol{q})$, where

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}) = J^{-1}\nabla H(\boldsymbol{y}), \quad \boldsymbol{y} = \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{q} \end{pmatrix}, \; J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \tag{7.37}$$

$\boldsymbol{p} \in \mathbb{R}^d$, $\boldsymbol{q} \in \mathbb{R}^d$ and $\nabla = (\nabla_p, \nabla_q)^{\mathrm{T}} = (\partial_{p_1}, \partial_{p_2}, \ldots, \partial_{p_d}, \partial_{q_1}, \partial_{q_2}, \ldots, \partial_{q_d})^{\mathrm{T}}$. The dynamical equations for the canonical variables $\boldsymbol{p}$ and $\boldsymbol{q}$ are

$$\dot{\boldsymbol{p}} = -\nabla_q H(\boldsymbol{p}, \boldsymbol{q}), \qquad \dot{\boldsymbol{q}} = \nabla_p H(\boldsymbol{p}, \boldsymbol{q}) \tag{7.38}$$

(see also Appendix G). How successful are the methods of previous sections in such problems? We provide the answer from several viewpoints. All single-step methods discussed so far can be understood as mappings of the solution from "time" nh to "time" $(n+1)h$,

$$\boldsymbol{y}_{n+1} = \phi_h(\boldsymbol{y}_n). \tag{7.39}$$

Does the numerical solution preserve the invariants of the continuous problem, for example, the Hamiltonian $H(\boldsymbol{p}, \boldsymbol{q})$? Does numerical integration of the Hamiltonian system (the solution of the initial-value problem) preserve the symplectic structure of the phase space? We may also ask whether the integrator is symmetric and reversible. A numerical method that satisfies at least one of these requirements is known as a *geometric integrator*.

7.12.1 Preservation of Invariants

A non-constant function $I(\boldsymbol{y})$ is called the *first integral* (or *invariant*, or *constant of motion*) of (7.37) if

$$\nabla I(\boldsymbol{y}) \cdot \boldsymbol{f}(\boldsymbol{y}) = 0 \quad \forall \boldsymbol{y}. \tag{7.40}$$

This means that at any point of the phase space (along any solution $\boldsymbol{y}$) the gradient $\nabla I(\boldsymbol{y})$ is orthogonal to the vector field $\boldsymbol{f}(\boldsymbol{y})$. A nice scalar example is the mathematical pendulum with the Hamiltonian $H(p, q) = \frac{1}{2}p^2 - \cos q$. The equations of motion (Newton's law) are $\dot{p} = -\sin q$ and $\dot{q} = p$, which can be written in the form (7.37) with $\boldsymbol{f}(\boldsymbol{y}) = (-\sin q, p)^{\mathrm{T}}$. Obviously $\nabla H(\boldsymbol{y}) \cdot \boldsymbol{f}(\boldsymbol{y}) = (p, \sin q)(-\sin q, p)^{\mathrm{T}} = 0$. A further example are the three invariants of the classical Kepler problem, discussed in Problem 7.14.13.

Let us test some integrators on an even simpler case of the one-dimensional harmonic oscillator (linear pendulum) with the spring constant k^2, described by the Hamiltonian

$$H(p,q) = \frac{1}{2}\left(p^2 + k^2 q^2\right), \tag{7.41}$$

with the analytic solution

$$\begin{pmatrix} \widetilde{p}(h) \\ \widetilde{q}(h) \end{pmatrix} = \begin{pmatrix} \cos h & -k^2 \sin h \\ \sin h & \cos h \end{pmatrix} \begin{pmatrix} p(0) \\ q(0) \end{pmatrix} \tag{7.42}$$

at time $t = h$. The basic explicit Euler method (7.5) is first order, so it also approximates the solution (7.42) only to first order. This implies

$$\begin{pmatrix} \widetilde{p}(h) \\ \widetilde{q}(h) \end{pmatrix}_{\text{Euler}} = \begin{pmatrix} 1 & -k^2 h \\ h & 1 \end{pmatrix} \begin{pmatrix} p(0) \\ q(0) \end{pmatrix}, \tag{7.43}$$

which can be seen if one explicit Euler step is done for (7.38). Instead of the correct value of the energy (7.41) we get

$$\frac{1}{2}\left(\widetilde{p}^2 + k^2 \widetilde{q}^2\right) = \frac{1}{2}\left(1 + k^2 h^2\right)\left(p^2 + k^2 q^2\right),$$

which is unbounded when the number of steps goes to infinity, regardless of the step size h. The RK4 method does not fare much better, as it results in damping

$$\frac{1}{2}\left(\widetilde{p}^2 + k^2 \widetilde{q}^2\right) = \frac{1}{2}\left(1 - k^6 h^6/72\right)\left(p^2 + k^2 q^2\right).$$

We also obtain damping with the implicit Euler method. Figure 7.12 (left) shows the solution (p, q) by the explicit Euler (7.5), implicit Euler (7.30), and implicit midpoint method (7.31) in the form for an autonomous Hamiltonian system:

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h J^{-1} \nabla H\left(\frac{1}{2}(\boldsymbol{y}_n + \boldsymbol{y}_{n+1})\right). \tag{7.44}$$

In general, explicit and implicit RK methods preserve only linear invariants. As an example [20], consider the conservation of mass in the process (7.36): from the system of equations we see $\dot{y}_1 + \dot{y}_2 + \dot{y}_3 = 0$, so $I(\boldsymbol{y}) = y_1 + y_2 + y_3$ is a linear invariant of the system. If the coefficients of an m-stage RK method (see (7.8) and Sect. 7.9) satisfy

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0, \quad i, j = 1, 2, \ldots, m, \tag{7.45}$$

the method also preserves quadratic invariants of the form

$$Q(\boldsymbol{y}) = \boldsymbol{y}^{\mathrm{T}} C \boldsymbol{y}, \tag{7.46}$$

where C is a symmetric square matrix. An example of such an invariant is the kinetic energy of a N-body system, $T(\boldsymbol{p}) = \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{p}_i / m_i$. By definition (7.40), $Q(\boldsymbol{y})$

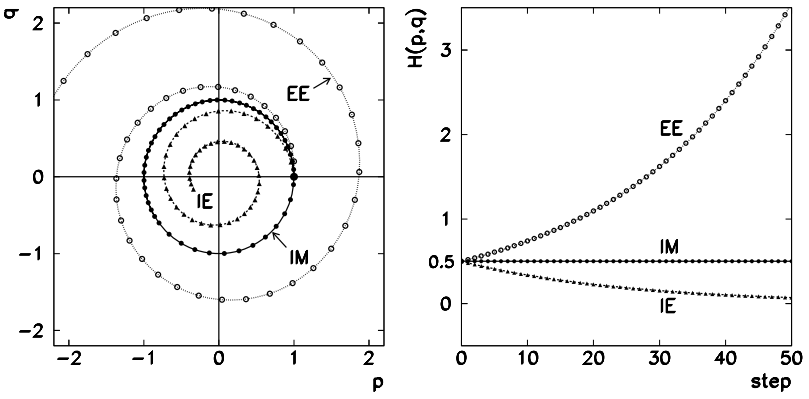


Fig. 7.12 [*Left*] The solution of the harmonic oscillator problem $\dot{p} = -k^2 q$, $\dot{q} = p$ with initial condition $(p, q) = (1, 0)$ (*large symbol* •) and $k = 1$ by explicit Euler's (EE), implicit Euler's (IE), and implicit midpoint method (IM) in 50 steps of $h = 0.2$. [*Right*] The values of $H(p, q) = \frac{1}{2}(p^2 + k^2 q^2)$ at current p and q for these three methods

is the invariant of the system (7.37) when $\boldsymbol{y}^{\mathrm{T}} C \boldsymbol{f}(\boldsymbol{y}) = 0$ for all $\boldsymbol{y}$. The second-order implicit midpoint method (7.44), the two-stage scheme

$$\boldsymbol{k}_1 = \boldsymbol{f}\left(\boldsymbol{y}_n + h\left[\frac{1}{4}\boldsymbol{k}_1 + \left(\frac{1}{4} - \frac{\sqrt{3}}{6}\right)\boldsymbol{k}_2\right]\right),$$

$$\boldsymbol{k}_2 = \boldsymbol{f}\left(\boldsymbol{y}_n + h\left[\left(\frac{1}{4} + \frac{\sqrt{3}}{6}\right)\boldsymbol{k}_1 + \frac{1}{4}\boldsymbol{k}_2\right]\right),$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{h}{2}(\boldsymbol{k}_1 + \boldsymbol{k}_2)$$

of fourth-order, as well as the sixth-order, three-stage Gauss 6 method (defined on p. 361) all satisfy the condition (7.45) and therefore preserve quadratic invariants of the form (7.46).

The energy of the linear harmonic oscillator $H(p, q)$ is a quadratic invariant in the variables p and q. Figure 7.12 (right) shows the dependence of $H(p, q)$ on the number of steps in the explicit Euler, implicit Euler, and the implicit midpoint method. Only the latter preserves the energy.

Partitioned RK Methods When the equations of motion of dynamical systems can be written in separated form

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{z}), \qquad \dot{\boldsymbol{z}} = \boldsymbol{g}(\boldsymbol{y}, \boldsymbol{z}),$$

special methods are available for their solution that do not belong to the classical Runge–Kutta family. They are known as *partitioned Runge–Kutta methods* [20] and

have the form

$$\boldsymbol{k}_i = \boldsymbol{f}\left(\boldsymbol{y}_n + h\sum_{j=1}^{m} a_{ij}\boldsymbol{k}_j, \boldsymbol{z}_n + h\sum_{j=1}^{m}\widehat{a}_{ij}\boldsymbol{l}_j\right),$$

$$\boldsymbol{l}_i = \boldsymbol{g}\left(\boldsymbol{y}_n + h\sum_{j=1}^{m} a_{ij}\boldsymbol{k}_j, \boldsymbol{z}_n + h\sum_{j=1}^{m}\widehat{a}_{ij}\boldsymbol{l}_j\right),$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\sum_{i=1}^{m} b_i\boldsymbol{k}_i, \qquad \boldsymbol{z}_{n+1} = \boldsymbol{z}_n + h\sum_{i=1}^{m}\widehat{b}_i\boldsymbol{l}_i.$$

In one step $n \to n+1$, two different classical RK methods are merged: we seek the solutions $\boldsymbol{y}$ by a method with coefficients a_{ij} and b_i, and the solutions $\boldsymbol{z}$ by another method with coefficients $\widehat{a}_{ij}$ and $\widehat{b}_i$ (or vice versa). The partitioned RK method of the lowest order joins two first-order Euler methods: the implicit one in $\boldsymbol{y}$ and the explicit one in $\boldsymbol{z}$ (or vice versa):

$$\begin{aligned}\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\boldsymbol{f}(\boldsymbol{y}_n, \boldsymbol{z}_{n+1}),\\ \boldsymbol{z}_{n+1} &= \boldsymbol{z}_n + h\boldsymbol{g}(\boldsymbol{y}_n, \boldsymbol{z}_{n+1}),\end{aligned} \quad \text{or} \quad \begin{aligned}\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\boldsymbol{f}(\boldsymbol{y}_{n+1}, \boldsymbol{z}_n),\\ \boldsymbol{z}_{n+1} &= \boldsymbol{z}_n + h\boldsymbol{g}(\boldsymbol{y}_{n+1}, \boldsymbol{z}_n).\end{aligned}$$

Another relevant partitioned method is the Störmer–Verlet scheme useful in solving the important class of problems $\ddot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q})$ which can be expressed as

$$\dot{\boldsymbol{p}} = \boldsymbol{f}(\boldsymbol{q}), \qquad \dot{\boldsymbol{q}} = \boldsymbol{p}.$$

This scheme is second order and is *explicit*:

$$\begin{aligned}\boldsymbol{p}_{n+1/2} &= \boldsymbol{p}_n + \frac{h}{2}\boldsymbol{f}(\boldsymbol{q}_n),\\ \boldsymbol{q}_{n+1} &= \boldsymbol{q}_n + h\boldsymbol{p}_{n+1/2},\\ \boldsymbol{p}_{n+1} &= \boldsymbol{p}_{n+1/2} + \frac{h}{2}\boldsymbol{f}(\boldsymbol{q}_{n+1}).\end{aligned} \tag{7.47}$$

By interchanging the roles of $\boldsymbol{p}$ and $\boldsymbol{q}$ as in the Euler pair, the adjoint Störmer–Verlet method can be derived (see next subsection).

In general the partitioned RK methods do not preserve quadratic invariants of the form $Q(\boldsymbol{y}) = \boldsymbol{y}^{\mathrm{T}} C\boldsymbol{y}$. If the coefficients of the partitioned method satisfy

$$\begin{aligned} b_i\widehat{a}_{ij} + \widehat{b}_j a_{ji} - b_i\widehat{b}_j = 0, \quad i, j = 1, 2, \ldots, m,\\ b_i - \widehat{b}_i = 0, \quad i = 1, 2, \ldots, m,\end{aligned}$$

the methods preserve the quadratic invariants

$$Q(\boldsymbol{y}, z) = \boldsymbol{y}^{\mathrm{T}} D\boldsymbol{z} \tag{7.48}$$

(or $Q(\boldsymbol{p},\boldsymbol{q}) = \boldsymbol{p}^{\mathrm{T}} D\boldsymbol{q}$), where D is a constant matrix. An example of such an invariant are the components of angular momentum of an N-body system, $\boldsymbol{L} = \sum_{i=1}^{N} \boldsymbol{q}_i \times \boldsymbol{p}_i$. The partitioned Euler method and the Störmer–Verlet method preserve the invariants of the form (7.48).

7.12.2 Preservation of the Symplectic Structure

In Hamiltonian systems the continuous mapping of $(\boldsymbol{p},\boldsymbol{q})$ at time $t = 0$ to $(\widetilde{\boldsymbol{p}},\widetilde{\boldsymbol{q}})$ at time $t = h$ along the solution preserves the symplectic structure, or the phase space volume,

$$\sum \mathrm{d}\widetilde{\boldsymbol{p}} \wedge \mathrm{d}\widetilde{\boldsymbol{q}} = \sum \mathrm{d}\boldsymbol{p} \wedge \mathrm{d}\boldsymbol{q}.$$

The numerical method (7.39) preserves this structure if it satisfies

$$\big[\phi_h(\boldsymbol{p},\boldsymbol{q})\big]^{\mathrm{T}} J \phi_h(\boldsymbol{p},\boldsymbol{q}) = J,$$

which also implies $\det \phi_h(\boldsymbol{p},\boldsymbol{q}) = 1$ or the preservation of volume in the phase space. Partitioned Euler's methods from the previous subsection are symplectic. (Check this as an exercise.) For Hamiltonian systems they have the form

$$\begin{aligned} \boldsymbol{p}_{n+1} &= \boldsymbol{p}_n - h\nabla_q H(\boldsymbol{p}_{n+1},\boldsymbol{q}_n), \\ \boldsymbol{q}_{n+1} &= \boldsymbol{q}_n + h\nabla_p H(\boldsymbol{p}_{n+1},\boldsymbol{q}_n), \end{aligned} \quad \text{and} \quad \begin{aligned} \boldsymbol{p}_{n+1} &= \boldsymbol{p}_n - h\nabla_q H(\boldsymbol{p}_n,\boldsymbol{q}_{n+1}), \\ \boldsymbol{q}_{n+1} &= \boldsymbol{q}_n + h\nabla_p H(\boldsymbol{p}_n,\boldsymbol{q}_{n+1}). \end{aligned} \tag{7.49}$$

The Störmer–Verlet schemes,

$$\begin{aligned} \boldsymbol{p}_{n+1/2} &= \boldsymbol{p}_n - \frac{h}{2}\nabla_q H(\boldsymbol{p}_{n+1/2},\boldsymbol{q}_n), \\ \boldsymbol{q}_{n+1} &= \boldsymbol{q}_n + \frac{h}{2}\big(\nabla_p H(\boldsymbol{p}_{n+1/2},\boldsymbol{q}_n) + \nabla_p H(\boldsymbol{p}_{n+1/2},\boldsymbol{q}_{n+1})\big), \\ \boldsymbol{p}_{n+1} &= \boldsymbol{p}_{n+1/2} - \frac{h}{2}\nabla_q H(\boldsymbol{p}_{n+1/2},\boldsymbol{q}_{n+1}), \end{aligned} \tag{7.50}$$

or

$$\begin{aligned} \boldsymbol{q}_{n+1/2} &= \boldsymbol{q}_n + \frac{h}{2}\nabla_q H(\boldsymbol{p}_n,\boldsymbol{q}_{n+1/2}), \\ \boldsymbol{p}_{n+1} &= \boldsymbol{p}_n - \frac{h}{2}\big(\nabla_p H(\boldsymbol{p}_n,\boldsymbol{q}_{n+1/2}) + \nabla_p H(\boldsymbol{p}_{n+1},\boldsymbol{q}_{n+1/2})\big), \\ \boldsymbol{q}_{n+1} &= \boldsymbol{q}_{n+1/2} + \frac{h}{2}\nabla_q H(\boldsymbol{p}_{n+1},\boldsymbol{q}_{n+1/2}), \end{aligned} \tag{7.51}$$

are also symplectic, and so is the implicit midpoint method (7.44). Figure 7.13 shows the effect of three low-order integrators on the size and shape of Arnold's cat [27].

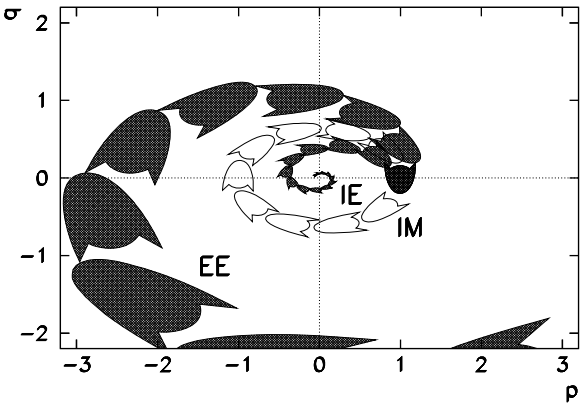


Fig. 7.13 Evolution of the Arnold cat in the (p, q) plane for the harmonic oscillator problem with $k = 1.6$. The explicit Euler method (EE) enlarges the area while the implicit (IE) reduces it. The implicit midpoint method (IM) preserves the area but not its shape

The symplectic Euler schemes (7.49) and the Störmer–Verlet schemes (7.50) and (7.51) for general Hamiltonians $H(\boldsymbol{p}, \boldsymbol{q})$ are implicit. For separable Hamiltonians, $H(\boldsymbol{p}, \boldsymbol{q}) = T(\boldsymbol{p}) + V(\boldsymbol{q})$, all these methods become explicit. For partitioned RK methods the condition (7.45) also implies symplecticity (not only preservation of quadratic invariants). Classical explicit RK methods are not symplectic.

7.12.3 Reversibility and Symmetry

Conservative Hamiltonian systems are reversible. If at some point of the flow corresponding to the equation $\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y})$ we change the direction of velocity, the form of the trajectory does not change; we just reverse the motion. An invertible linear transformation ρ helps us define a more general notion of ρ-reversibility. A differential equation $\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y})$ and the vector field $\boldsymbol{f}(\boldsymbol{y})$ are ρ-reversible if

$$\rho \boldsymbol{f}(\boldsymbol{y}) = -\boldsymbol{f}(\rho \boldsymbol{y}) \quad \forall \boldsymbol{y}. \tag{7.52}$$

An analogous concept can be defined for single-step numerical methods (7.39). The method ϕ_h is symmetric (with respect to time reversal) if $\phi_h \circ \phi_{-h} = 1$. If the method ϕ_h, applied to a ρ-reversible differential equation, satisfies

$$\rho \circ \phi_h = \phi_{-h} \circ \rho, \tag{7.53}$$

then ϕ_h is a ρ-reversible mapping precisely when ϕ_h is symmetric. (Symmetry and ρ-reversibility of a discrete map are equivalent properties.) All explicit and implicit methods of the RK type satisfy (7.53) if (7.52) is valid. Partitioned RK methods satisfy the condition (7.53) if ρ can be written in the form $\rho(\boldsymbol{u}, \boldsymbol{v}) = (\rho_1(\boldsymbol{u}), \rho_2(\boldsymbol{v}))$, where ρ_1 and ρ_2 are invertible mappings. The symplectic Euler methods (7.49) are not symmetric (or ρ-reversible). The Störmer–Verlet methods (7.50) and (7.51) are symmetric and therefore also ρ-reversible [28].

7.12.4 Modified Hamiltonians and Equations of Motion

Unfortunately it is difficult to find an integration method for general (also non-integrable) Hamiltonian systems that would preserve the symmetry properties of the system and its invariants, and at the same time fulfill the symplectic condition [29, 30]. We therefore often opt for methods satisfying at least one of these requirements: for the integration of astronomical orbits we might prefer to ensure the periodicity and preservation of energy, while in the study of charged-particle motion in storage rings, where radiation losses can be compensated for, the behavior of the volume in phase space (spatial and momentum dispersion) may be more relevant. In the following we discuss only explicit methods for which we insist on symplecticity but sacrifice exact energy conservation. A surprise is in store.

In the basic Euler method a small modification of the matrix in (7.43),

$$\begin{pmatrix} \widetilde{p}(h) \\ \widetilde{q}(h) \end{pmatrix}_{\text{symp. Euler}} = \begin{pmatrix} 1-h^2k^2 & -hk^2 \\ h & 1 \end{pmatrix} \begin{pmatrix} p(0) \\ q(0) \end{pmatrix}, \tag{7.54}$$

renders the mapping exactly symplectic ($\det \phi_h = 1$). The modified mapping does not preserve the energy, since after many repetitions of (7.54) we obtain $\widetilde{p}^2 + k^2\widetilde{q}^2 = p^2 + k^2q^2 + \mathcal{O}(h^2)$: the value of the energy is only first-order accurate. On the example of the linear harmonic oscillator this can be explained by the conserved quantity (constant of motion)

$$\widetilde{H}(p,q) = \frac{1}{2}\big(p^2 + k^2q^2\big) + \frac{h}{2}k^2pq = \text{const.} \tag{7.55}$$

In other words, after many repetitions of (7.54) with initial conditions $(p,q) = (1,0)$ and some small h, the points in phase space lie on the ellipse $p^2 + k^2q^2 + hk^2pq = 1$, which *always* differs from the analytic solution $p^2 + k^2q^2 = 1$ at most to order h. The error in the energy due to the local discretization error does not increase. The scheme (7.54) therefore preserves the symplectic *and* Hamiltonian structure, but the system evolves in accordance with a modified (or "perturbed") Hamiltonian (7.55). These observations are illustrated in Fig. 7.14.

This interesting realization is not confined to linear systems: an example is already the non-linearized pendulum with the Hamiltonian $H(p,q) = \frac{1}{2}p^2 - \cos q$. The true, "unperturbed" Hamiltonian equations of motion are $\dot{p} = -\sin q$, $\dot{q} = p$. By using the symplectic Euler method (explicit in q and implicit in p) with step h we are in fact solving the equations

$$\begin{pmatrix} \dot{p} \\ \dot{q} \end{pmatrix} = \begin{pmatrix} -\sin q \\ p \end{pmatrix} + \frac{h}{2}\begin{pmatrix} p\cos q \\ -\sin q \end{pmatrix} + \frac{h^2}{12}\begin{pmatrix} (p^2 - 2\cos q)\sin q \\ 2p\cos q \end{pmatrix} + \cdots,$$

which correspond to the modified Hamiltonian

$$\widetilde{H}(p,q) = \frac{1}{2}p^2 - \cos q - \frac{h}{2}p\sin q + \frac{h^2}{12}\big(p^2 - \cos q\big)\cos q + \cdots.$$

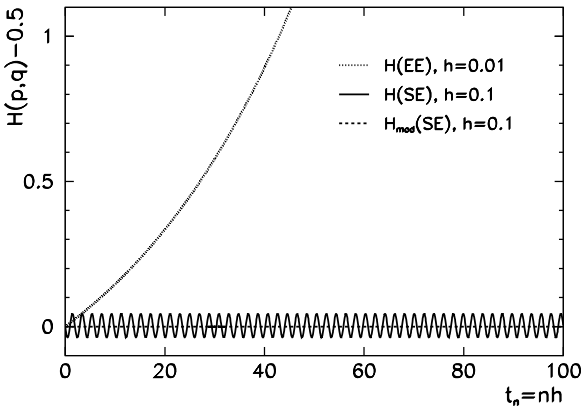


Fig. 7.14 Time evolution of the Hamiltonian for the linear harmonic oscillator with $k = 1.6$. Shown is the energy (7.41) for the explicit Euler method (EE) with step $h = 0.01$ and the energies (7.41) and (7.55) for the symplectic Euler method (SE) with step $h = 0.1$. This method *exactly* preserves the invariant (7.55)

The analysis of modified Hamiltonians and the corresponding equations of motion applies to a more general class of symplectic integrators. The sequence of mappings

$$q' = q + hc\left(\frac{\partial T}{\partial p}\right)_p, \qquad p' = p - hd\left(\frac{\partial V}{\partial q}\right)_{q=q'}, \tag{7.56}$$

each of which is symplectic by itself, faithfully describes the time evolution of the system in accordance with a modified Hamiltonian $\widetilde{H} = H + hH_1 + h^2H_2 + \cdots$. Higher-order symplectic integrators can be constructed by repeated applications of the tandem (7.56), where the parameters c and d are different at each step. An integrator of order p can be assigned the modified Hamiltonian

$$\widetilde{H}_p = H + h^p H_p + \mathcal{O}\big(h^{p+1}\big).$$

By reducing h we can therefore come (almost) arbitrarily close to exact energy conservation. However, we note that for non-linear systems the convergence in the term $h^p H_p$ has not yet been strictly proven.

The mathematical background and the construction of explicit high-order symplectic integrators are given in Appendix G. An excellent reference on geometric integration is offered by the monograph [28] and the review article [31]. Geometric integration is a crucial tool for the study of charged-particle trajectories in accelerators; a comprehensive discussion in this context is [32]. For partial differential equations (PDE), geometric methods are less well developed. The PDE solution approaches based on difference methods that ignore geometric aspects are discussed in Chaps. 9 and 10. An introduction to conservative discretizations of Hamiltonian PDEs can be found in [33].

7.13 Lie-Series Integration ⋆

Lie-series integrators represent a class of explicit methods to solve ordinary differential equations. They are conceptually simple, but difficult to apply to general

cases, as the equations of motion need to be differentiated explicitly. However, when the equations are given in terms of algebraic functions that can be manipulated efficiently, and when high precision is required, Lie-series integrators reveal their true power. Their typical area of use is dynamical astronomy [34–36]. Note that Lie-series integrators in their basic forms are *not* symplectic and their energy and angular momentum errors are unbounded [37].

Here we discuss autonomous dynamical systems $\phi : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ and denote the independent and dependent variables by $t \in \mathbb{R}$ and $x \in \mathbb{R}^n$, respectively. A trajectory starting at $x(0)$ is given by $x(t) = \phi(t, x(0))$ and is determined by the system of ODEs

$$\dot{x}(t) = F\big(x(t)\big).$$

7.13.1 Taylor Expansion of the Trajectory

By assuming that the trajectories are smooth functions, they can be Taylor-expanded as

$$x(t + \Delta t) = \sum_{k=0}^{n} \frac{x^{(k)}(t)}{k!} (\Delta t)^k + R_n(t),$$

where $(\cdot)^{(k)}$ denotes the kth time derivative, and the remainder is

$$R_n(t) = \frac{1}{n!} \int_t^{t+\Delta t} x^{(n+1)}(\tau)(\tau - t)^n \, \mathrm{d}\tau.$$

Defining the rescaled derivatives $F_k(x(t)) = x^{(k)}(t)/k!$ the Lie-series integrator of order n with time step $\Delta t > 0$ can be written explicitly as

$$x(t + \Delta t) = x(t) + \sum_{k=1}^{n} F_k\big(x(t)\big)(\Delta t)^k + R_n(t).$$

The remainder $|R_n(t)|$ represents the local error of the integrator.

Note that for given equations of motion, a Lie-series integrator of order p can be constructed if F is continuously differentiable up to $p + 1$ times. The applicability of the integrator depends on whether the rescaled derivatives F_k can be efficiently generated from the right-hand side of the equations of motion, $F(x) = F_1(x)$. They can be obtained by using the recurrence relation

$$F_k(x) = \frac{1}{k} F'_{k-1}(x) F(x). \tag{7.57}$$

For algebraic F_i this recurrence can be exploited efficiently to very high orders by using programs for symbolic computations.

Note that the local error has a very simple form, thus the global error can be well controlled. By the mean-value theorem, a $\xi \in [t, t+\Delta t]$ exists such that $R_n(t) = F_{n+1}(\xi)(\Delta t)^{n+1}$; if $F_{n+1}(x(t)) \neq 0$ and $\Delta t \ll 1$, the remainder becomes $R_n(t) \approx F_{n+1}(x(t))(\Delta t)^{n+1}$. Hence $R_n(t)$ scales as $\mathcal{O}((\Delta t)^{n+1})$, and for a given $x(t)$, it can be estimated by finding a convergence radius ρ of the series defined as $|F_k(x(t))| \asymp \rho^{-k}$ when $k \to \infty$. A fast convergence of the series is achieved if $\Delta t \ll \rho$. In this limit, $R_n(t)$ for large orders n can be approximated as

$$\big|R_n(t)\big| \asymp \sum_{k=n+1}^{\infty} \left(\frac{\Delta t}{\rho}\right)^k = \left(\frac{\Delta t}{\rho}\right)^{n+1} \left(1 - \frac{\Delta t}{\rho}\right)^{-1}.$$

Example The Lorenz system (see [38, 39] and Problem 7.14.7) is a dynamical system with a three-dimensional phase space: the trajectory of the system is described by the vector $x(t) = (x_1(t), x_2(t), x_3(t))^{\mathrm{T}} \in \mathbb{R}^3$, and is determined by the system of equations (7.62); in the notation of this section,

$$\begin{aligned} \dot{x}_1 &= \sigma(x_2 - x_1) = F_1(x_1, x_2, x_3), \\ \dot{x}_2 &= x_1(r - x_3) - x_2 = F_2(x_1, x_2, x_3), \\ \dot{x}_3 &= x_1 x_2 - b x_3 = F_3(x_1, x_2, x_3). \end{aligned} \tag{7.58}$$

Here we use $\sigma = 10$, $r = 28$, and $b = 8/3$, for which the system is chaotic and possesses a finite strange attractor.

We would like to calculate the trajectory of the system over long times as precisely as possible by Lie-series integration. In order to minimize the round-off errors, we work in floating-point arithmetic with precision $\varepsilon_{\mathrm{M}} = 10^{-1024}$ by using the GMP library [40]. The Lie integrator of order p has the form

$$x_i(t + \Delta t) = x_i(t) + \sum_{k=1}^{p} F_{i,k}\big(x(t)\big)(\Delta t)^k,$$

where $F_{i,k}$ denotes the rescaled derivative $x_i^{(k)}(t)/k!$. Any such derivative can be expressed as a polynomial of variables x_i in the form

$$F_{i,k}(x) = \sum_{\alpha \in I_{i,k}} A_{\alpha}^{i,k} x_1^{\alpha_1} x_2^{\alpha_2} x_3^{\alpha_3}, \quad \alpha = \{\alpha_1, \alpha_2, \alpha_3\},$$

where $I_{i,k}$ is the set of possible powers corresponding to $F_{i,k}(x)$. The explicit form of $F_{i,k}(x)$ can be obtained systematically by the recurrence (7.57), a task most swiftly performed by MATHEMATICA in exact arithmetic. (Note that since x and F are vectors, F' is a matrix.) The size of $I_{i,k}$ (the number of terms in $F_{i,k}$) for all i increases sub-exponentially with increasing k, as shown in Fig. 7.15 (left).

Because the attractor is finite and the dynamics is ergodic, a single trajectory visits the complete phase space on long time scales. Each trajectory experiences the same local errors, only at different times. Consequently, the maximum local error at

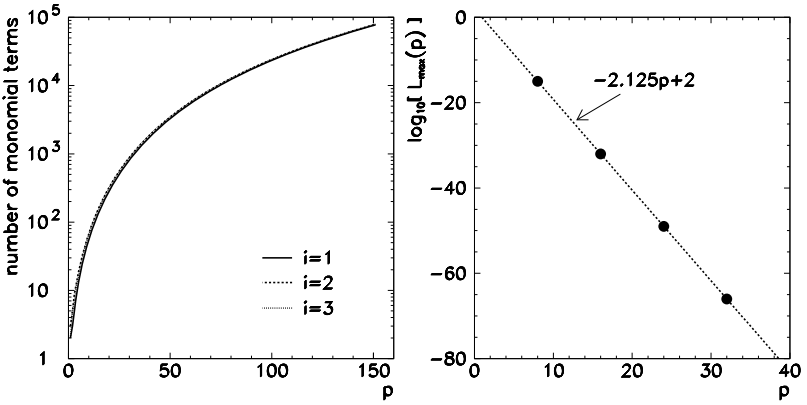


Fig. 7.15 [*Left*] The number of monomial terms in $F_{i,k}$ up to $k = p$ in the Taylor series expansion, as a function of order p. [*Right*] The maximal local error $L_{\max}$ as a function of order p at time step $\Delta t = 10^{-3}$ and precision $\varepsilon_{\mathrm{M}} = 10^{-1024}$ (*symbols* •)

given order p and time step Δt is $L_{\max}(p, \Delta t) = \max_{k\in\mathbb{N}} |R_p(k\Delta t)|$ and is equal for all coordinates. In finite floating-point arithmetic with precision ε_{M}, the maximum local error scales with ε_{M} and Δt as

$$L_{\max}(p, \Delta t) = \mathcal{O}\left(\varepsilon_{\mathrm{M}} \max_i \sum_{k=1}^{p} (\Delta t)^k S_{i,k} P_{i,k} \Lambda^{P_{i,k}}\right) + \mathcal{O}\big((\Delta t)^{p+1}\big),$$

where $S_{i,k} = \sum_{\alpha\in I_{i,k}} \alpha$ is the sum of powers in $I_{i,k}$, $P_{i,k} = \max I_{i,k}$ is the maximum power in $I_{i,k}$, and Λ is the typical length scale of the attractor.

The first term of $L_{\max}$ corresponds to round-off errors, while the second term appears because the integrator has a finite order. In our numerical calculation the arithmetic precision is so large that round-off errors have a negligible effect on the dynamics, hence the second term dominates. From Fig. 7.15 (right) we see that $L_{\max}$ depends exponentially on p like $L_{\max}(p, \Delta t) \asymp (\Delta t/\rho_{\mathrm{eff}})^p$, where ρ_{eff} is the effective Taylor series convergence radius for the attractor.

The deviation of the numerical trajectory $\hat{x}(t)$ obtained by the Lie-series integrator with step Δt, from the exact one, $x(t)$, which both start at the same point, $x(0) = \hat{x}(0)$, is represented by the global error $G_n = \|\hat{x}(n\Delta t) - x(n\Delta t)\|_2$. In the limit $G_n \ll 1$ the upper bound of the global error can be estimated as

$$G_n \le G_{\max} \sim L_{\max}(p, \Delta t) \frac{\exp(\lambda_{\max} n\Delta t) - 1}{\exp(\lambda_{\max}\Delta t) - 1}, \quad n \to \infty. \tag{7.59}$$

Here we have assumed that $G_{\max}$ grows *maximally* with n *on average*, i.e. it increases by a factor $\exp(\lambda_{\max}\Delta t)$ in one iteration, where $\lambda_{\max}$ is the maximum Lyapunov exponent. Consequently, $G_{\max}$ grows exponentially as $\mathcal{O}(\exp(\lambda_{\max} n\Delta t))$. In the limit of small perturbations, $\varepsilon \to 0$, the maximum Lyapunov exponent $\lambda_{\max}$ can

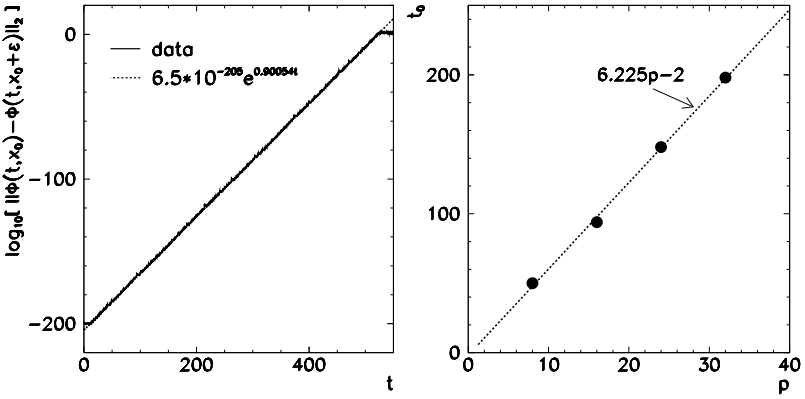


Fig. 7.16 [*Left*] The divergence of trajectories starting at neighboring points x_0 and $x_0 + \varepsilon$ for $x_0 = (1, 1, 1)^{\mathrm{T}}$ and $\varepsilon = (10^{-200}, 0, 0)^{\mathrm{T}}$. [*Right*] The time t_0 until which the global error remains below a given threshold value ε, as a function of order p, for $\varepsilon = 0.1$ and a trajectory starting at $x(0) = (1, 1, 1)^{\mathrm{T}}$ (*symbols* •)

be determined by using the asymptotic relation

$$\left\|\phi(t, x + \varepsilon) - \phi(t, x)\right\|_2 \sim \|\varepsilon\|_2 \exp(\lambda_{\max} t), \qquad t \to \infty.$$

In practice, this is approximately valid for times $t = n\Delta t$ and perturbations such that $\lambda_{\max} t \ll -\log \|\varepsilon\|_2$. From the results presented in Fig. 7.16 (left) we find $\lambda_{\max} = 0.90054 \pm 10^{-5}$, which agrees well with the value found in [41].

The upper bound of G_n given in (7.59) is quite loose, because the increase of the error is usually not maximal as assumed. A tighter bound can be found empirically by calculating the envelope of G_n along $n \in \mathbb{N}$, which is modeled by

$$G_{\max} \approx L_{\max} \exp(\mu n \Delta t).$$

Here $\mu \leq \lambda_{\max}$ is the effective Lyapunov exponent that can be computed from the threshold time $t_0(\varepsilon)$ until which the global error remains below a specified threshold value ε. The threshold time as a function of the order of the integrator is shown in Fig. 7.16 (right), and $\mu = \log(\varepsilon / L_{\max})/t_0(\varepsilon)$. By applying this formula to the data in the figure, we find $\mu \approx 0.79$: the global error increases significantly slower than is maximally possible on average.

For $\mu t \gg 1$ and $\mu \Delta t \ll 1$, these results can be condensed in an empirical formula $\log_{10} G_{\max} \approx -2.125p + 0.343t + \log_{10} L_{\max}(p = 0)$, which allows us to estimate the resources needed to compute the trajectory until $t = 10^3$ to an accuracy of one digit. By setting $G_{\max} \approx 1$ at $t = 10^3$, we find the minimal order of the integrator $p = 160$, while $G_{\max}$ at this p and $t = 0$ gives the upper bound on the precision of the derivatives, which is $\varepsilon_{\mathrm{der}} \approx 10^{-340}$. In order to compute the derivatives to precision $\varepsilon_{\mathrm{der}}$, we obviously need to work with arithmetic much finer than that, i.e. $\varepsilon_{\mathrm{M}} \ll \varepsilon_{\mathrm{der}}$.

7.14 Problems

7.14.1 Time Dependence of Filament Temperature

An example of a scalar first-order initial-value problem is the time dependence of the temperature of a filament (e.g. in a light bulb) [42]. At time zero we expose the filament with resistance R to a current pulse from a charged capacitor with capacitance C. The differential equation for the temperature is

$$mc_\mathrm{p}\frac{\mathrm{d}T}{\mathrm{d}t} = RI_0^2 \exp\left(-\frac{2t}{RC}\right) - \sigma S T^4.$$

By introducing dimensionless variables we obtain an equation with a single parameter a,

$$\frac{\mathrm{d}y}{\mathrm{d}x} = a\mathrm{e}^{-2x} - y^4. \tag{7.60}$$

We neglect the radiation received by the filament from the environment, so the temperature will ultimately approach zero. Moreover, we assume that the current pulse is strong enough that the initial internal energy of the filament may be ignored as well. The initial condition for (7.60) is therefore $y(0) = 0$.

⊙ Solve this problem by using several Euler methods discussed in Sect. 7.2, as well as by select methods of Sects. 7.3 and 7.4. Determine the required step size for the solution to remain stable. Choose the method (and the corresponding step sizes) to compute the families of solutions for $a = 2(4)18$! What would be your method of choice for a precise determination of the maximum temperature and the times at which these maxima occur? If the radiation from the environment is not neglected, (7.60) involves two parameters:

$$\frac{\mathrm{d}y}{\mathrm{d}x} = a\mathrm{e}^{-2x} - b\left(y^4 - 1\right).$$

Find the families of solutions of these equations for $a = 5$ and $b = 0(0.5)2$! Explain the physical meaning of parameters a and b.

7.14.2 Oblique Projectile Motion with Drag Force and Wind

In this problem (adapted from [43]) we would like to compute the trajectory of an object launched under an angle with a specific initial velocity, and experiencing a quadratic drag force and the presence of wind. The equations of motion are

$$\dot{x} = v\cos\theta,$$
$$\dot{z} = v\sin\theta,$$

$$\dot{\theta} = -\frac{g}{v}\cos\theta,$$
$$\dot{v} = -\frac{F_\mathrm{d}}{m} - g\sin\theta,$$

where the drag force is $F_\mathrm{d} = \frac{1}{2}c\rho S((\dot{x} - w(t))^2 + \dot{z}^2)$.

$\odot$ Solve this problem by using methods for the solution of systems of ordinary differential equations. We shoot the cannon-ball with mass $m = 15$ kg and velocity $v_0 = 50$ m/s from the origin $(x, z) = (0, 0)$ at an angle θ_0 with respect to the positive x-axis. Change the initial angle θ_0 in steps of 5°. Other parameters are $c = 0.2$ (drag coefficient), $\rho = 1.29$ kg/m^3 (density of air), $S = 0.25$ m^2 (projectile cross-sectional area). The winds are blowing in the x-direction: no wind ($w(t) = 0$); constant-velocity headwind ($w(t) = -10$ m/s); non-uniform tailwind (blowing with $w(t) = +10$ m/s every odd second and with $w(t) = 0$ every even second); and gusty wind (Gaussian distributed $w(t)$ with zero mean and 10 m/s standard deviation). Think of another aspect of the problem and discuss it accordingly.

$\bigoplus$ Solve the problem in the case that the winds blow in arbitrary directions in the (x, y) plane.

7.14.3 *Influence of Fossil Fuels on Atmospheric* $\mathbf{CO_2}$ *Content*

The study of the influence of fossil fuel burning on the atmospheric CO_2 content is an important geochemical problem. The fraction of CO_2 in the present atmosphere is $\approx 3.5 \times 10^{-4}$, but a small change in this fraction may have strong repercussions on the global climate. A relatively simple model [44] can be used to simulate the interaction of carbon compounds in the atmosphere, shallow waters, and deep oceans. The main variables in the model are the partial pressure of CO_2 in the atmosphere, p, the concentrations of carbonates (dissolved C) in shallow and deep oceans, σ_s and σ_d, and the alkalinities in shallow and deep oceans, α_s and α_d. The auxiliary variables in shallow oceans, where chemical processes between the gaseous CO_2 and dissolved carbonates occur, are the concentrations of hydrogen carbonates and carbonates, h_s and c_s, and the partial pressure of CO_2 in water, p_s. The burning of fossil fuels from the beginning of the industrial age is the source of CO_2 entering as the function f. The corresponding system of differential equations is

$$\frac{\mathrm{d}p}{\mathrm{d}t} = \frac{p_\mathrm{s} - p}{d} + \frac{f(t)}{\mu_1},$$
$$\frac{\mathrm{d}\sigma_\mathrm{s}}{\mathrm{d}t} = \frac{1}{v_\mathrm{s}}\left[w(\sigma_\mathrm{d} - \sigma_\mathrm{s}) - k_1 - \mu_2\frac{p_\mathrm{s} - p}{d}\right],$$
$$\frac{\mathrm{d}\sigma_\mathrm{d}}{\mathrm{d}t} = \frac{1}{v_\mathrm{d}}\left[k_1 - w(\sigma_\mathrm{d} - \sigma_\mathrm{s})\right],$$
$$\frac{\mathrm{d}\alpha_\mathrm{s}}{\mathrm{d}t} = \frac{1}{v_\mathrm{s}}\left[w(\alpha_\mathrm{d} - \alpha_\mathrm{s}) - k_2\right],$$

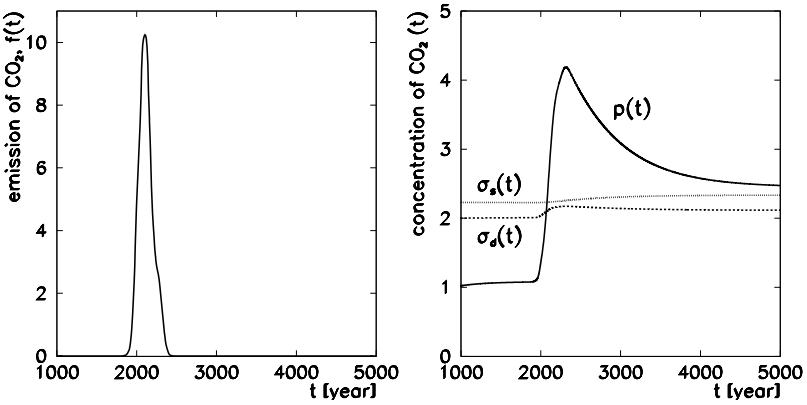


Fig. 7.17 [*Left*] The function f describing the emissions of CO_2 into the atmosphere from year 1000 to 5000. [*Right*] Concentrations of CO_2 in the atmosphere (p), shallow (σ_s), and deep ocean (σ_d) as functions of time (solution by the adaptive RK4 method)

$$\frac{d\alpha_d}{dt} = \frac{1}{v_d}\big[k_2 - w(\alpha_d - \alpha_s)\big].$$

It is complemented by the shallow-ocean equilibrium equations:

$$h_s = \frac{\sigma_s - (\sigma_s^2 - k_3\alpha_s(2\sigma_s - \alpha_s))^{1/2}}{k_3}, \qquad c_s = \frac{\alpha_s - h_s}{2}, \qquad p_s = k_4\frac{h_s^2}{c_s}.$$

The numerical constants are $d = 8.64$, $\mu_1 = 4.95 \times 10^2$, $\mu_2 = 4.95 \times 10^{-2}$, $v_s = 0.12$, $v_d = 1.23$, $w = 0.001$, $k_1 = 2.19 \times 10^{-4}$, $k_2 = 6.12 \times 10^{-5}$, $k_3 = 0.997148$, $k_4 = 6.79 \times 10^{-2}$.

⊙ Solve the system of equations with the initial conditions $p = 1.00$, $\sigma_s = 2.01$, $\sigma_d = 2.23$, $\alpha_s = 2.20$ and $\alpha_d = 2.26$ in year $t = 1000$. The emission of CO_2 between the years 1000 and 5000 (Fig. 7.17 (left)) is approximated by the function

$$f(t) = \sum_{i=1}^{3} c_i \exp\left(-\frac{(t-t_i)^2}{s_i^2}\right), \qquad \begin{array}{l} c_1 = 2.0, \quad t_1 = 1988, \, s_1 = 21, \\ c_2 = 10.5, \, t_2 = 2100, \, s_2 = 96, \\ c_3 = 2.7, \quad t_3 = 2265, \, s_3 = 57. \end{array}$$

Determine the carbon concentrations in the atmosphere, shallow, and deep oceans from year 1000 to 5000 by select integrators and reproduce Fig. 7.17 (right). Which method is the most appropriate? Compare the initial and final concentrations. When does the atmospheric concentration of CO_2 reach its maximum? The chemical processes have quite different reaction rates, so the problem outlined above is mildly *stiff*: stiffness was introduced in Sect. 7.10.

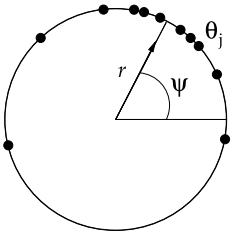
7.14.4 Synchronization of Globally Coupled Oscillators

Kuramoto's model [45, 46] describes a large set of weakly (but non-linearly) coupled oscillators whose natural frequencies have a certain probability distribution. Due to the non-linearity of the coupling, a fraction of the oscillators may oscillate in phase ("collective synchronization") in specific conditions. This effect can be observed in networks of pacemaker cells in the brain and heart, in synchronous flashing of firefly swarms, with crickets chirping in unison, or even in electronic circuits based on Josephson junctions.

In this problem we discuss a finite number ($N \gg 1$) of coupled oscillators. Their dynamics is governed by the system of non-linear differential equations

$$\frac{\mathrm{d}\theta_i}{\mathrm{d}t} = \omega_i + \frac{K}{N}\sum_{j=1}^{N} \sin(\theta_j - \theta_i), \quad i = 1, 2, \ldots, N, \tag{7.61}$$

where θ_i is the phase of the ith oscillator, ω_i is its natural (proper, non-coupled) frequency, and K is the coupling constant. Let the natural frequencies ω_i be distributed according to some probability density $g(\omega)$ which is even and monotonously decreasing on either side of the mean value Ω, so $g(\Omega + \omega) = g(\Omega - \omega)$. Rotational symmetry allows us to set $\Omega = 0$ and redefine $\theta_i \to \theta_i + \Omega t$ for each t (we change to a system rotating uniformly with the frequency Ω): the equations of motion do not change, only the position of the peak of $g(\omega)$ shifts and we have $g(\omega) = g(-\omega)$.



The behavior of the set of oscillators is described by the complex order parameter that can be computed from the dynamical equations at any time t and measures the collective "rhythm" of the whole population of oscillators. We define it as

$$r(t)\mathrm{e}^{\mathrm{i}\psi(t)} = \frac{1}{N}\sum_j \mathrm{e}^{\mathrm{i}\theta_j(t)}.$$

The modulus of the parameter, $r(t)$, measures the level of coherence, while its phase, $\psi(t)$, measures the average phase (see figure). If K is smaller than the critical value $K_* = 2/\pi g(0)$, the oscillators oscillate incoherently (each with a different frequency and mutually uncorrelated in phase). After long times $r(t)$ oscillates out, with fluctuations on the order of $1/\sqrt{N}$, as anticipated for θ_j distributed uniformly on the interval $[0, 2\pi]$. When K is increased above K_*, an ever larger fraction of oscillators oscillate coherently: their phases tend to congregate around the average phase ψ, while the modulus of the order parameter asymptotically approaches a non-zero value $\lim_{t\to\infty} r(t) = r_\infty$. The fraction of the oscillators oscillating coherently after long times is

$$\int_{-Kr_\infty}^{+Kr_\infty} g(\omega)\,\mathrm{d}\omega.$$

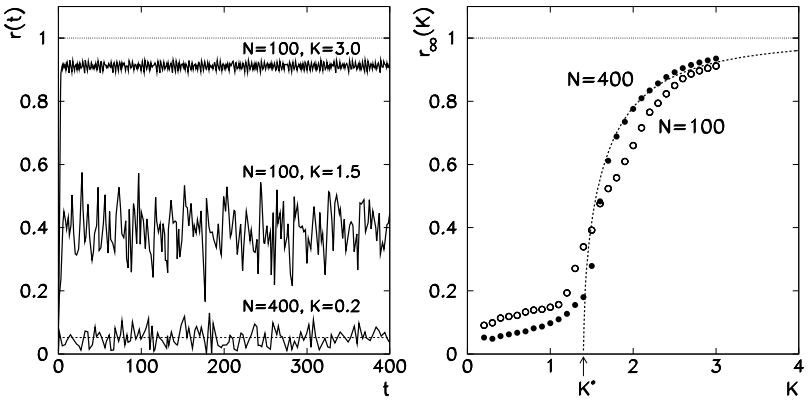


Fig. 7.18 [*Left*] The modulus of the order parameter as a function of t. The *dotted line* at the bottom is the average of $r(t)$ in $[0, 400]$, which is $0.052 \approx 1/\sqrt{400}$. [*Right*] The value of r after long times as a function of K for $N = 100$ and $N = 400$

For certain distributions $g(\omega)$, both K_* and r_∞ can be calculated analytically [47]. For the Lorentz distribution of natural frequencies,

$$g(\omega) = \frac{\Delta}{\pi} \frac{1}{\omega^2 + \Delta^2},$$

we obtain $K_* = 2\Delta$ and the dependence $r_\infty = [1 - (K_*/K)^2]^{1/2}$ at $K > K_*$.

⊙ Discuss the Kuramoto set of oscillators with Lorentz or Gauss distributions of natural frequencies. Map the chosen continuous distribution to a sufficiently large discrete set of oscillators (at least $N \approx 100$), and solve the system (7.61). At time zero the phases θ_j should be distributed uniformly on $[0, 2\pi]$. Use the solution $\theta_j(t)$ to compute the complex order parameter and monitor the behavior of $r(t)$ (Fig. 7.18 (left)) for the chosen value $K < K_*$ and for some value $K > K_*$. The critical value K_* is between 1.0 and 2.0. Compute $r(t)$ at long times (r_∞) for different values of the coupling parameter K in the vicinity of K_*. You ought to notice a phase transition with an approximately square-root dependence (Fig. 7.18 (right)). Draw the time dependence of the phase $\psi(t)$. Increase the number of oscillators to a reasonable limit (in terms of CPU time) and compare the results to the previous ones. Do not run the integrator with too strict precision tolerances.

7.14.5 Excitation of Muscle Fibers

Non-linear evolution equations with highly interesting solutions appear in modeling of muscle fiber excitations by external voltage pulses [48]. The relevant quantities are the cell membrane potential $V(t)$ that measures the current deviation of the potential difference on two sides of the membrane from a reference value (for example,

in the unexcited state), and the set of variables $w_i(t)$ describing the fraction of the ion channel that, at time t and specific potential difference, are in the conducting, non-conducting, or some other state. Already the relatively simple Morris–Lecar model [49] with two ion channels (Ca^{2+} and K^+) yields an extremely colorful physical picture. The dynamics of the voltage perturbation along the fiber is described by the coupled equations

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = I - \overline{g}_{\mathrm{Ca}} m_\infty(V)(V - V_{\mathrm{Ca}}) - \overline{g}_{\mathrm{K}} w (V - V_{\mathrm{K}}) - \overline{g}_{\mathrm{L}}(V - V_{\mathrm{L}}) + s(t),$$
$$\frac{\mathrm{d}w}{\mathrm{d}t} = \phi \frac{w_\infty(V) - w}{\tau_w(V)},$$

where C is the membrane capacitance per unit surface, $\overline{g}_{\mathrm{Ca}}$, $\overline{g}_{\mathrm{K}}$, and $\overline{g}_{\mathrm{L}}$ are the specific conductivities for the ion channels Ca^{2+}, K^+, and leakage, I is the static component of the surface current density through the fiber, and $s(t)$ is the external perturbation. The quantity w is the fraction of the open channels K^+. We have denoted

$$m_\infty(V) = \frac{1}{2}\left[1 + \tanh\left(\frac{V - V_1}{V_2}\right)\right],$$
$$w_\infty(V) = \frac{1}{2}\left[1 + \tanh\left(\frac{V - V_3}{V_4}\right)\right],$$
$$\tau_w(V) = \left[1 + \cosh\left(\frac{V - V_3}{2V_4}\right)\right]^{-1},$$

where $m_\infty(V)$ and $w_\infty(V)$ are the fractions of open channels Ca^{2+} and K^+ in the stationary state ($t \to \infty$). In the model, the characteristic time τ_w for the K^+ channel to open depends on the current value of V. (For the Ca^{2+} channel we assume instantaneous activation by V, otherwise we would have two equations, $\mathrm{d}w_1/\mathrm{d}t = \cdots$, $\mathrm{d}w_2/\mathrm{d}t = \cdots$ and two characteristic times $\tau_{w1} = \cdots$, $\tau_{w2} = \cdots$.)

⊙ Solve the Morris–Lecar system by two or three integration methods with various step sizes. Reproduce Fig. 7.19 (left) with the initial conditions listed in the first four rows of Table 7.1. Use the parameters $V_1 = -1.2$, $V_2 = 18.0$, $V_3 = 2.0$, $V_4 = 30.0$, $\overline{g}_{\mathrm{Ca}} = 4.4$, $\overline{g}_{\mathrm{K}} = 8.0$, $\overline{g}_{\mathrm{L}} = 2.0$, $V_{\mathrm{Ca}} = 120.0$, $V_{\mathrm{K}} = -84.0$, $V_{\mathrm{L}} = -60.0$, $\phi = 0.04$, and $C = 20.0$. (On purpose, we have not converted the equations to dimensionless form, as all quantities have a clear physical meaning. For units of all quantities see [49].)

Find the maximum and minimum values of the potential, $V_{\max}$ and $V_{\min}$, at large times and determine the stationary-state curves (*nullclines*) that connect all points with the properties $\mathrm{d}V/\mathrm{d}t = 0$ or $\mathrm{d}w/\mathrm{d}t = 0$ in the phase diagram (V versus w), see Fig. 7.19 (right). The intersection of these curves is the asymptotic stationary state, if such a state exists.

Observe the time dependence of V on a wider interval $[0, 800]$ when you act on the system by an external pulse $s(t)$ of the form

$$s(t) = 30H(t - 100)H(105 - t) + 30H(t - 470)H(475 - t),$$

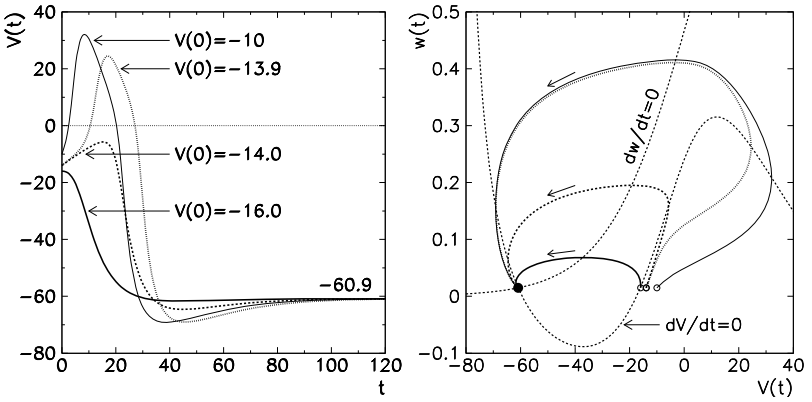


Fig. 7.19 Solutions of the Morris–Lecar model. [*Left*] The membrane potential V as a function of time for different initial conditions. [*Right*] Phase diagram ($w(t)$ versus $V(t)$) for the same initial conditions (*symbols* ∘) as in the left figure. Also shown are the stationary-state curves connecting the points with $\mathrm{d}V/\mathrm{d}t = 0$ and $\mathrm{d}w/\mathrm{d}t = 0$. They intersect at $(V, w) \approx (-60.9, 0.015)$ corresponding to the stationary state (*symbol* •)

Table 7.1 Initial conditions for V and w, the current density I, the integration interval length T, and the list of tasks for solving the Morris–Lecar model. The curves of the stationary state (s.s.) and the shape of the pulse $s(t)$ are defined in the text

$V(0)$	$w(0)$	I	T	Observe
−16.0	0.014915	0	150	$V(t)$, $w(t)$, $V(w)$, s.s. curves
−14.0	0.014915	0	150	
−13.9	0.014915	0	150	
−10.0	0.014915	0	150	
−10.0	0.014915	0(1)300	800	$V_{\max}(I)$, $V_{\min}(I)$
−26.59	0.129	90	800	$V(t)$ with double pulse $s(t)$

where $H(t)$ is the Heaviside (step) function. In an actual experiment this truly represents a 5 ms long step pulse with the peak current density 30 $\mu\mathrm{A/cm}^2$ at time 100 ms and another such pulse at time 470 ms.

7.14.6 Restricted Three-Body Problem (Arenstorf Orbits)

A basic problem in astronomy is to find the trajectory of a light object (for example, a satellite) in the presence of two much heavier bodies whose motion is not influenced by the light object. The heavy bodies with a mass ratio $\mu : (1 - \mu)$ circle in the (x, y) plane with frequency 1 around their common center of gravity, which is at the origin: the ratio of their orbital radii is then $(1 - \mu) : \mu$. In general the

light object may move outside the plane defined by the orbits of heavy bodies. The dimensionless equations of motion for the light body are

$$\ddot{x} = x + 2\dot{y} - \frac{(1-\mu)(x+\mu)}{r^3} - \frac{\mu(x-1+\mu)}{s^3},$$
$$\ddot{y} = y - 2\dot{x} - \frac{(1-\mu)y}{r^3} - \frac{\mu y}{s^3},$$
$$\ddot{z} = -\frac{(1-\mu)z}{r^3} - \frac{\mu z}{s^3},$$

where

$$r = \sqrt{(x+\mu)^2 + y^2 + z^2}, \qquad s = \sqrt{(x-1+\mu)^2 + y^2 + z^2}.$$

A detailed analysis of planar solutions of the special case $\mu = 0.5$ (frequently used as a benchmark test for symplectic methods) can be found in [50].

⊙ Solve the restricted three-body problem corresponding to the Earth and Moon as the heavy objects, hence $\mu = M_{\text{Moon}}/M_{\text{Earth}} = 0.012277471$. Consider the planar case ($z = 0$) with the initial conditions (example parameters from [28])

$$\big(x(0), y(0)\big) = (0.994, 0),$$
$$\big(\dot{x}(0), \dot{y}(0)\big) = (0, -2.00158510637908252240537862 24),$$

for which the solution is periodic with the period

$$T = 17.06521656015796255889172062 49.$$

Use the Euler explicit method with step $h = T/24000$ and the RK4 method with $h = T/6000$. Plot the dependence of x, $\dot{x}$, y, and $\dot{y}$ on time, as well as the phase diagrams $(x, \dot{x})$ and $(y, \dot{y})$, as in Fig. 7.20 (left). Can you obtain a periodic solution by reducing the step size in either the Euler or RK4 method? (Pretend that the solution is periodic if the deviation of $x(10T)$, $\dot{x}(10T)$, $y(10T)$, and $\dot{y}(10T)$ from the initial conditions after ten returns does not exceed 1 %.)

⊕ Use the Dormand–Prince method 5(4) with adaptive step size. Select a few tolerances on the local error and determine the number of steps needed for one cycle in the phase diagram at this precision (Fig. 7.20 (right)). Enrich the problem by allowing non-planar orbits of the third body ($z \neq 0$).

The problem outlined here is an example of motion of satellites along the "horseshoe" orbits in the gravitational field of the Sun–Earth system. Recently two nearby Earth asteroids have been discovered, 3753 Cruithne [51] and 2002 AA29 [52], whose orbits have similar properties. Even before that, the restricted three-body problem was analyzed theoretically for the Sun–Jupiter system [53]. You can find a rich set of initial conditions for it in [54] or [55].

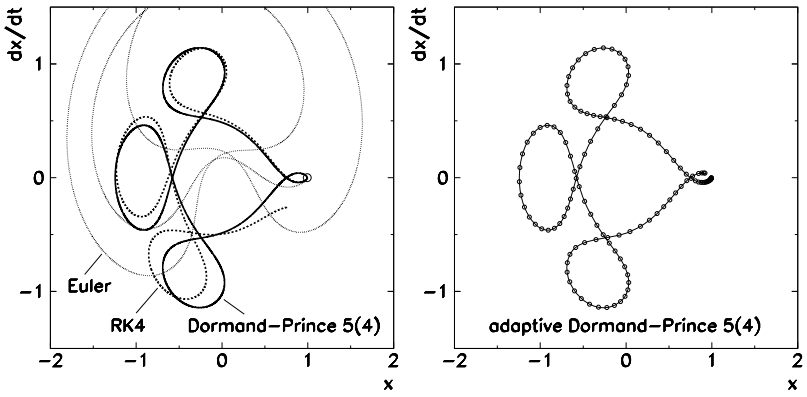


Fig. 7.20 Orbits of the light object in the restricted three-body problem. [*Left*] Solution by the explicit Euler method (7.5) with step $h = T/24000$ and by the RK4 method (7.10) and Dormand–Prince 5(4) method (p. 345) with steps $h = T/6000$. [*Right*] Solution by the Dormand–Prince 5(4) method with adaptive step size. At 0.001 tolerance on the local error a mere 134 steps are needed

7.14.7 Lorenz System

A classical problem of atmospheric physics is the convection of a fluid with kinematic viscosity ν, volume expansion coefficient β, and heat diffusion coefficient D, in a layer of thickness H in which a constant temperature difference $\Delta T_0 = T(0) - T(H) > 0$ persists between the top and bottom. Simpler cases were treated by Lord Rayleigh already in 1916, but in certain regimes of Prandtl ($\sigma = \nu/\beta$) and Rayleigh ($R = g\beta H^3 \Delta T_0/D\nu$) numbers the convection dynamics (velocity of motion, heat transfer) becomes very complex [56].

The Lorenz system [38]

$$\begin{aligned} \dot{X} &= -\sigma X + \sigma Y, \\ \dot{Y} &= -XZ + rX - Y, \\ \dot{Z} &= XY - bZ, \end{aligned} \tag{7.62}$$

offers a simplified picture of such convection. The variable X represents the velocity of the convection current. The variable Y is proportional to the temperature difference between the ascending and descending convection currents (warmer fluid goes up, colder goes down), while Z is the deviation of the vertical temperature profile from the linear height dependence (a positive value implies strong gradients in the vicinity of the top or bottom edge). The derivatives in (7.62) are with respect to dimensionless time $\tau = \pi^2(1 + a^2)\kappa t/H^2$, where a is a parameter. The coefficient $r = R/R_c$ is the Rayleigh number in units of the critical value $R_c = \pi^4(1 + a^2)^3/a^2$ (with a minimum at $27\pi^4/4$), while $b = 4/(1 + a^2)$.

⊙ Solve the Lorenz system with parameters $\sigma = 10$, $b = 8/3$, $r = 28$, and initial conditions $X(0) = 0$, $Y(0) = 1$, $Z(0) = 0$, which are slightly "off" the

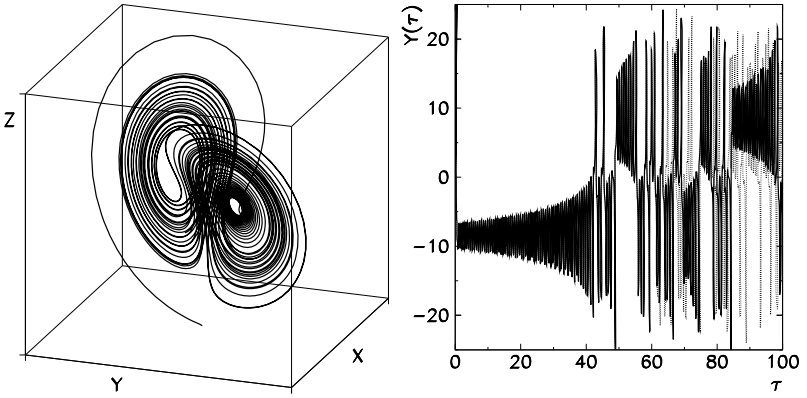


Fig. 7.21 The Lorenz problem. [*Left*] The solution on the time interval [0, 50] in the phase space of variables X, Y, and Z. [*Right*] Dependence of the solution $Y(\tau)$ on the initial conditions. Shown are two solutions with the initial condition $Y(0) = 1$ (*dotted curve*) and with $Y(0) = 1 + 10^{-12}$ (*full curve*)

convection-free state at $(0, 0, 0)$. Use the RK4 method. Plot the three-dimensional phase space (Fig. 7.21 (left)) and some typical two-dimensional slices through this space in all three planes. Quasi-stationary convection states correspond to the points $(6\sqrt{2}, 6\sqrt{2}, 27)$ and $(-6\sqrt{2}, -6\sqrt{2}, 27)$. Solve the system with slightly perturbed initial conditions. For example, take $Y(0) = 1 + \varepsilon$ instead of $Y(0) = 1$, where $|\varepsilon| \ll 1$ and observe the spread of the solutions as in Fig. 7.21 (right).

7.14.8 Sine Pendulum

In Newton's law $m\ddot{x} = F(x)$ for a non-linearized (sine) pendulum the restoring force is $F(x) = -\sin x$. Since F does not depend on the velocity, the system $m\dot{x} = p$, $\dot{p} = F(x)$, can be solved by a special trapezoidal-like scheme [42]

$$x(t+h) = x(t) + hu\left(t + \frac{1}{2}h\right),$$

$$u\left(t + \frac{1}{2}h\right) = u\left(t - \frac{1}{2}h\right) + h\frac{F(t, x(t))}{m}.$$

The auxiliary variable u best approximates the velocity $\dot{x}$ at the midpoints of $[t, t+h]$ and we assign it to those points.

⊙ Solve the differential equation

$$\frac{\mathrm{d}^2 x}{\mathrm{d}t^2} + \sin x = 0, \quad x(0) = 1,\ \dot{x}(0) = 0,$$

by the scheme given above and by the RK4 method, and compare the results. Find the step size h to achieve six-digit precision. Investigate the periodic stability of the methods: let the computation proceed over a large number (100, 1000, 10000) of oscillations and observe the change in the oscillation amplitudes. Compute the energy

$$E = V + T = 1 - \cos x + \frac{1}{2}\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^2$$

and see whether it remains constant. The analytic solution can be expressed in terms of the Jacobi elliptic functions sn() and cn() implemented in the GSL library in the `gsl_sf_elljac_e()` routine. The period of oscillation of the analytic solution is $4K(\sin\frac{1}{2}x(0))$, where $K(u)$ is the complete elliptic integral of the first kind: $K(\sin 0.5) \approx 1.83703$.

⊕ Study the resonance curve of the driven damped pendulum

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + \beta\frac{\mathrm{d}x}{\mathrm{d}t} + \sin x = A\sin\omega_0 t,$$

where β is the damping coefficient, while A and ω_0 are the driving amplitude and frequency. When ω_0 is increased or decreased, the amplitudes at the same ω_0 differ: you can clearly observe hysteresis effects [57].

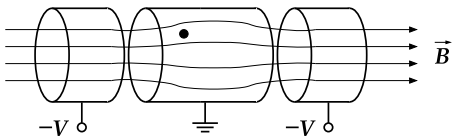
7.14.9 Charged Particles in Electric and Magnetic Fields

Motion of charged particles in complicated configurations of electric and magnetic fields abounds in experimental particle physics, for example, in electro-magnetic traps used to study the properties of individual particles or plasma, Wien filters with crossed electric and magnetic fields acting as velocity selectors in beam-lines, and in magnetic spectrometers. In all these cases the particle with mass m and charge e experiences the Lorentz force

$$\boldsymbol{F} = \frac{\mathrm{d}(\gamma m\boldsymbol{v})}{\mathrm{d}t} = e(\boldsymbol{E} + \boldsymbol{v}\times\boldsymbol{B}). \tag{7.63}$$

In practical situations the hardest problem is the precise knowledge of the vector fields $\boldsymbol{E}$ and $\boldsymbol{B}$ at all places; the numerical integration of the equations of motion is the easier task.

⊙ Numerically integrate the equations of motion of a positively charged particle in the Penning trap shown in the figure. Longitudinal confinement is provided by the electric potential,



while the axially symmetric magnetic field confines the particles in the transverse directions. The system (7.63) can be solved as a system of six first-order equations,

$$\begin{aligned} \dot{\boldsymbol{r}} &= \boldsymbol{v}, & \boldsymbol{r}(0) &= \boldsymbol{r}_0, \\ \dot{\boldsymbol{v}} &= \kappa(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}), & \boldsymbol{v}(0) &= \boldsymbol{v}_0, \end{aligned}$$

where $\kappa = e/\gamma m$. (Note that the Lorentz factor γ depends on the magnitude of the velocity!) The electric and magnetic fields have the form

$$\begin{aligned} E_z(\pm z) &= \pm E_0 \exp\bigl(-5(z \pm 3)^2\bigr), \\ B_z(\pm z) &= \quad B_0\bigl(0.5 + 0.2\exp\bigl(-z^2\bigr)\bigr), \end{aligned}$$

$E_0 = B_0 = 1$, and we neglect the transverse components of the magnetic field.

⊕ Study the precession of the average spin vector of a polarized beam of spin-1/2 particles in non-constant magnetic fields. The dynamics of the spatial part of the spin four-vector is determined by the Thomas equation

$$\frac{\mathrm{d}\boldsymbol{S}}{\mathrm{d}t} = \kappa \boldsymbol{S} \times \left[\frac{g}{2}\boldsymbol{B}^{\parallel} + \left(1 + \frac{g-2}{2}\gamma\right)\boldsymbol{B}^{\perp}\right], \quad \boldsymbol{S}(0) = \boldsymbol{S}_0,$$

where g is the gyro-magnetic ratio, and we split the magnetic field to its components parallel and perpendicular to the particle velocity vector, $\boldsymbol{B}^{\parallel} \equiv (\hat{\boldsymbol{v}} \cdot \boldsymbol{B})\hat{\boldsymbol{v}}$ and $\boldsymbol{B}^{\perp} \equiv \boldsymbol{B} - \boldsymbol{B}^{\parallel}$. Consider this vector equation as three new coupled scalar equations in the system, and run the integrator again. Observe the spin precession of protons with various momenta and incident angles in magnetic systems like a dipole magnet, a system of two quadrupole magnets rotated by 90°, a sextupole magnet, or some other field configuration!

7.14.10 Chaotic Scattering

Classical planar scattering of a point particle with mass M on a potential $V(\boldsymbol{r})$ is described by the four-dimensional system of equations of motion

$$M\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = -\nabla V(\boldsymbol{r}), \qquad \frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \boldsymbol{v},$$

where $\boldsymbol{r} = (x, y)$ and $\boldsymbol{v} = (v_x, v_y)$. Because the energy $E = \frac{1}{2}M\boldsymbol{v}^2 + V(\boldsymbol{r})$ is conserved, the phase space is only three-dimensional. Therefore the state of the particle is uniquely defined by the variables x, y, and the angle θ between the vector $\boldsymbol{v}$ and the positive x-axis, since the magnitude of the velocity can be calculated from the energy,

$$|\boldsymbol{v}| = \sqrt{2\bigl(E - V(\boldsymbol{r})\bigr)/M}.$$

⊙ Study the planar scattering of a particle of mass M on the potential

$$V(\boldsymbol{r}) = x^2 y^2 \exp\left[-\left(x^2 + y^2\right)\right],$$

which has four maxima at $(x, y) = (1, \pm 1)$ and $(-1, \pm 1)$ with values $E_\mathrm{m} = 1/\mathrm{e}^2$ [58]. Let the particle with mass $M = 1$ and energy E impinge towards the central part of the potential from large distances with impact parameter b (parallel to the x-axis and at constant distance b from it).

Plot some typical particle trajectories in the (x, y) plane as a function of the parameter b (Fig. 7.22 (top right)). Compute the scattering angles ϕ for a large set of values of b, $-3 \le b \le 3$, at $E/E_\mathrm{m} = 1.626$ (regular scattering) and at $E/E_\mathrm{m} = 0.260$ (chaotic scattering). Look closer at the obtained results in the ever narrower regions of b, e.g. for $-0.6 < b < -0.1$, $-0.400 < b < -0.270$, and $-0.3920 < b < -0.3770$ (Fig. 7.22 (bottom left and right)). Compute the time delay (time spent by the particle in the central part of the potential) as a function of b. By using a constant step-size integrator the delay is simply proportional to the number of steps taken.

7.14.11 Hydrogen Burning in the pp I Chain

Fusion of hydrogen nuclei occurs in stars with masses less than $\approx 1.3 M_\odot$ in three branches of the pp chain. The first stage (pp I branch) involves the processes [59]

$$\begin{aligned}
{}^1\mathrm{H} + {}^1\mathrm{H} &\longrightarrow {}^2\mathrm{H} + \mathrm{e}^+ + \nu_\mathrm{e} && (\lambda_\mathrm{pp}),\\
{}^1\mathrm{H} + {}^1\mathrm{H} + \mathrm{e}^- &\longrightarrow {}^2\mathrm{H} + \nu_\mathrm{e} && (\lambda'_\mathrm{pp}),\\
{}^2\mathrm{H} + {}^1\mathrm{H} &\longrightarrow {}^3\mathrm{He} + \gamma && (\lambda_\mathrm{pd}),\\
{}^3\mathrm{He} + {}^3\mathrm{He} &\longrightarrow {}^4\mathrm{He} + {}^1\mathrm{H} + {}^1\mathrm{H} && (\lambda_{33}),
\end{aligned}$$

where λ_i are the reaction rates. The pp I branch contributes 85 % to the solar luminosity, but the process rates in it are very different due to a large span of reaction cross-sections and Coulomb barriers [60, 61]. We express λ_i in units of reactions per unit time per $(\mathrm{mol/cm}^3)^{N-1}$, where N is the number of interacting particles excluding photons. At temperatures in the solar interior ($\approx 15 \times 10^6$ K) they are $\lambda_\mathrm{pp} = 8.2 \times 10^{-20}$, $\lambda'_\mathrm{pp} = 2.9 \times 10^{-24}$, $\lambda_\mathrm{pd} = 1.3 \times 10^{-2}$, $\lambda_{33} = 2.3 \times 10^{-10}$.

For the listed processes $\lambda'_\mathrm{pp} \ll \lambda_\mathrm{pp} \ll \lambda_{33} \ll \lambda_\mathrm{pd}$. The first two processes are by far the slowest ones, as they are governed by the weak interaction. Here we

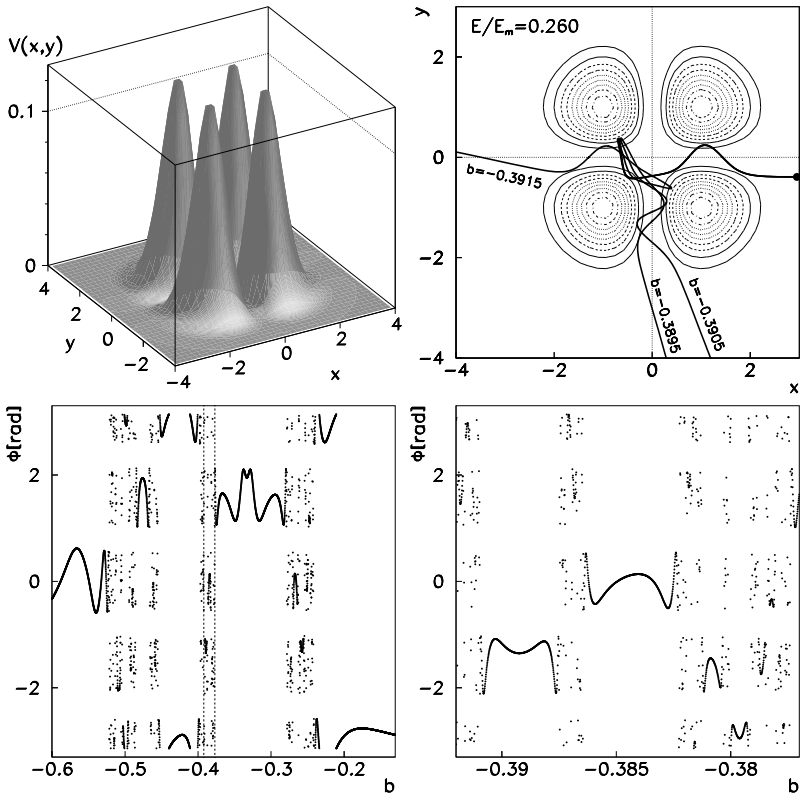


Fig. 7.22 Chaotic scattering on the potential $V(x, y) = x^2y^2 \exp[-(x^2 + y^2)]$. [*Top left*] The potential $V(x, y)$. [*Top right*] Dependence of the solution on initial conditions $x(0) = 3$, $y(0) = b$, $\dot{x}(0) = 0$, and $\dot{y}(0) = -\sqrt{2.0(E - V(x(0), y(0)))/M}$ with parameters $M = 1$, $E = 0.260E_{\mathrm{m}}$. Shown are the solutions at three different impact parameters $b = -0.3915$, 0.3905, and 0.3895. [*Bottom left*] Dependence of the scattering angle ϕ on b with $-0.60 \le b \le -0.13$. [*Bottom right*] The zoom-in in the region $-0.392 \le b \le -0.377$ (denoted by *dashed vertical lines* in the *figure at left*). We are witnessing classical chaos: we observe a large sensitivity of the system to the initial conditions, and self-similarity (equal or similar structures appear at different scales)

approximate $\lambda'_{\mathrm{pp}} = 0$. The equations for the pp I branch then become

$$\begin{aligned}
\frac{\mathrm{d}n_{\mathrm{p}}}{\mathrm{d}t} &= -\lambda_{\mathrm{pp}} n_{\mathrm{p}}^2 - \lambda_{\mathrm{pd}} n_{\mathrm{p}} n_{\mathrm{d}} + \lambda_{33} n_3^2, \\
\frac{\mathrm{d}n_{\mathrm{d}}}{\mathrm{d}t} &= \lambda_{\mathrm{pp}} \frac{n_{\mathrm{p}}^2}{2} - \lambda_{\mathrm{pd}} n_{\mathrm{p}} n_{\mathrm{d}}, \\
\frac{\mathrm{d}n_3}{\mathrm{d}t} &= \lambda_{\mathrm{pd}} n_{\mathrm{p}} n_{\mathrm{d}} - \lambda_{33} n_3^2, \\
\frac{\mathrm{d}n_4}{\mathrm{d}t} &= \lambda_{33} \frac{n_3^2}{2},
\end{aligned} \tag{7.64}$$

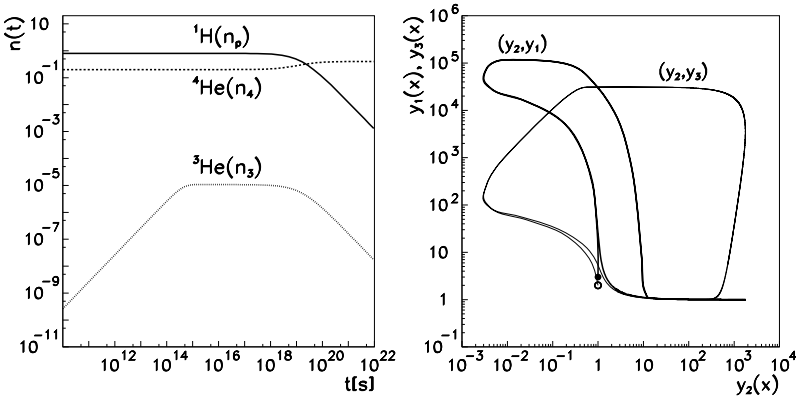


Fig. 7.23 Examples of stiff differential equations. [*Left*] Hydrogen burning in the pp I chain (see (7.64)). The solution for n_d lies below the shown area. [*Right*] Limit cycles of the Oregonator (see (7.65)). The *symbols* ○ and ● denote the initial conditions

where we have denoted the isotope concentrations by $n_\mathrm{p} = n(^1\mathrm{H})$, $n_\mathrm{d} = n(^2\mathrm{H})$, $n_3 = n(^3\mathrm{He})$, and $n_4 = n(^4\mathrm{He})$.

⊙ Solve the system (7.64) on the time interval $10^{10}\ \mathrm{s} \le t \le 10^{22}\ \mathrm{s}$. The initial condition for n_p can be computed from the estimate $(\lambda_{\mathrm{pp}} n_\mathrm{p})^{-1} \approx 10^{10}$ years which is valid at the conditions in the solar interior, and we set $n_\mathrm{d} = n_3 = n_4 = 0$. At first you may restrict the computation to large times, when ^{2}H and ^{3}He are in equilibrium ($\mathrm{d}n_\mathrm{d}/\mathrm{d}t = \mathrm{d}n_3/\mathrm{d}t = 0$). In that case the stiff system of four equations reduces to a non-stiff system of two equations. To consider all times except the shortest, assume that ^{2}H is in equilibrium: then the equation for ^{3}He can be solved analytically by assuming that the concentrations of ^{1}H and ^{4}He do not change substantially, and by using the solution for n_3 in the equations for n_p and n_4. For finding the solution at arbitrary times, use an integrator tailored to stiff systems (see Fig. 7.23 (left)).

⊕ Augment the basic system of equations for the pp I branch with contributions of heavier isotopes, and compare the results. Use the reactions given in [59–62], which list the reaction rates also for temperatures that are different from those in the solar interior.

7.14.12 Oregonator

Oregonator is a domesticated name for the chemical reaction of $\mathrm{HBrO_2}$, $\mathrm{Br^-}$, and Ce(IV). The dynamics is described by a stiff set of equations [63]

$$\begin{aligned} y_1' &= \kappa_1\big(y_2 + y_1(1 - \alpha y_1 - y_2)\big), \\ y_2' &= \kappa_2\big(y_3 - y_2(1 + y_1)\big), \\ y_3' &= \kappa_3(y_1 - y_3), \end{aligned} \tag{7.65}$$

where $\kappa_1 = 1/\kappa_2 = 77.27$, $\kappa_3 = 0.161$, and $\alpha = 8.375 \times 10^{-6}$. As expected from a stiff system, the solutions change over many orders of magnitude (Fig. 7.23 (left and right)).

⊙ Solve the system (7.65) with initial conditions $y_1(0) = 3$, $y_2(0) = 1$, $y_3(0) = 2$. Use an explicit integrator of your own choice and the implicit fifth-order integrator Radau 5 described on p. 360. If possible, resort to adaptive step size control in both cases. Plot the solutions $y_1(x)$, $y_2(x)$, and $y_3(x)$ for $0 \le x \le 360$. The solutions on this interval form interesting three-dimensional limit cycles. Display them by plotting pairs of coordinates (y_2, y_1) and (y_2, y_3) as shown in Fig. 7.23 (right). What are the required step sizes in the computation with the explicit and implicit schemes, respectively?

7.14.13 Kepler's Problem

The solution of the Kepler problem is one of the very first homework exercises of any astronomer. We treat the planar motion of two gravitationally attracting bodies by placing one body at the origin and describing the other body by spatial coordinates (q_1, q_2) and momenta (p_1, p_2). The dimensionless equations of motion (Newton's law) are

$$\ddot{q}_1 = -\frac{q_1}{(q_1^2 + q_2^2)^{3/2}}, \qquad \ddot{q}_2 = -\frac{q_2}{(q_1^2 + q_2^2)^{3/2}}.$$

These equations describe the time evolution of a Hamiltonian system with three conserved quantities. The first one is the Hamiltonian (the total energy)

$$H(p_1, p_2, q_1, q_2) = \frac{1}{2}\left(p_1^2 + p_2^2\right) - \frac{1}{\sqrt{q_1^2 + q_2^2}}, \quad p_i = \dot{q}_i.$$

The other quadratic invariant is the angular momentum $L(p_1, p_2, q_1, q_2) = q_1 p_2 - q_2 p_1$.

⊙ Solve Kepler's problem with eccentricity $e = 0.6$ and initial conditions

$$q_1(0) = 1 - e, \qquad \dot{q}_1(0) = 0, \qquad q_2(0) = 0, \qquad \dot{q}_2(0) = \sqrt{\frac{1+e}{1-e}}.$$

The orbital period is 2π, and the exact values of the energy and angular momentum are $H_0 = -1/2$ and $L_0 = \sqrt{1 - e^2}$. First use the explicit Euler method (7.5) with a step size h that still ensures stability, say $h = 0.0005$, then use the same h in the implicit midpoint method (7.31). Observe the solution over at least 10 periods. Repeat the calculation by using the Störmer–Verlet method (7.47) with 10/100 times larger h (and, accordingly, 10/100 times fewer steps).

Plot the orbits ($q_2(t)$ versus $q_1(t)$), the energy, and the angular momentum. Check whether the integrators conserve the third invariant of Kepler's problem, the

Laplace–Runge–Lenz vector $\boldsymbol{A} = \boldsymbol{p} \times \boldsymbol{L} - \boldsymbol{r}/r$, written by components as

$$\begin{pmatrix} A_1 \\ A_2 \\ 0 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ q_1 p_2 - q_2 p_1 \end{pmatrix} - \frac{1}{\sqrt{q_1^2 + q_2^2}} \begin{pmatrix} q_1 \\ q_2 \\ 0 \end{pmatrix}.$$

$\bigoplus$ Solve the problem by using fourth- and sixth-order symplectic integrators with coefficients given in Appendix G.

7.14.14 Northern Lights

When charged particles emitted from the Sun (solar wind) encounter the Earth's magnetic field, they cause Northern lights. Assuming that the magnetic field is axially symmetric along the z-axis, the particle trajectories $(x(s), y(s), z(s))$ are determined by the equations (quoted in [2] based on work of [64]):

$$\begin{aligned} x'' &= \frac{1}{r^5}\big(3yzz' - \big(3z^2 - r^2\big)y'\big), \\ y'' &= \frac{1}{r^5}\big(\big(3z^2 - r^2\big)x' - 3xzz'\big), \\ z'' &= \frac{1}{r^5}\big(3xzy' - 3yzx'\big), \end{aligned}$$

where $r^2 = x^2 + y^2 + z^2$ and $'$ denotes the derivative with respect to the parameter s. In polar coordinates, $x = R\cos\phi$ and $y = R\sin\phi$, we rewrite the system as

$$R'' = \left(\frac{2\gamma}{R} + \frac{R}{r^3}\right)\left(\frac{2\gamma}{R^2} + \frac{3R^2}{r^5} - \frac{1}{r^3}\right), \tag{7.66}$$

$$z'' = \left(\frac{2\gamma}{R} + \frac{R}{r^3}\right)\frac{3Rz}{r^5}, \tag{7.67}$$

$$\phi' = \left(\frac{2\gamma}{R} + \frac{R}{r^3}\right)\frac{1}{R}, \tag{7.68}$$

where $r^2 = R^2 + z^2$, and the parameter γ is the integration constant when the equation for ϕ'' is integrated once.

$\bigodot$ Solve (7.66) and (7.67) on the interval $0 \le s \le 0.3$ by one of the explicit Störmer methods for equations $\boldsymbol{y}'' = \boldsymbol{f}(x, \boldsymbol{y})$ from Sect. 7.8, e.g.

$$\begin{aligned} &\boldsymbol{y}_{n+1} - 2\boldsymbol{y}_n + \boldsymbol{y}_{n-1} = h^2 \boldsymbol{f}_n, \\ &\boldsymbol{y}_{n+1} - 2\boldsymbol{y}_n + \boldsymbol{y}_{n-1} = h^2\left(\frac{13}{12}\boldsymbol{f}_n - \frac{1}{6}\boldsymbol{f}_{n-1} + \frac{1}{12}\boldsymbol{f}_{n-2}\right), \end{aligned}$$

$$\boldsymbol{y}_{n+1} - 2\boldsymbol{y}_n + \boldsymbol{y}_{n-1} = h^2\left(\frac{7}{6}\boldsymbol{f}_n - \frac{5}{12}\boldsymbol{f}_{n-1} + \frac{1}{3}\boldsymbol{f}_{n-2} - \frac{1}{12}\boldsymbol{f}_{n-3}\right),$$

and determine ϕ by integrating (7.68). The initial conditions are $R_0 = 0.257453$, $R_0' = \sqrt{Q_0}\cos u$, $z_0 = 0.314687$, and $z_0' = \sqrt{Q_0}\sin u$, where

$$r_0 = \sqrt{R_0^2 + z_0^2}, \qquad Q_0 = 1 - \left(2\gamma/R_0 + R_0/r_0^3\right)^2, \quad \gamma = -0.5,\ u = 5\pi/4.$$

7.14.15 Galactic Dynamics

Galaxies are assemblies of $N \gg 1$ stars in mutual gravitational attraction. One way of studying galactic dynamics is to compute the orbit of a single star in the gravitational potential generated by the remaining $N - 1$ stars. Assume that this potential is

$$V(x, y, z) = A\log\left(C + \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2}\right),$$

and that it rotates around the galactic axis with uniform velocity, while its functional form does not change with time [65]. The Lagrangian for the system rotating with constant angular velocity Ω is

$$L = \frac{1}{2}\left((\dot{x} - \Omega y)^2 + (\dot{y} + \Omega x)^2 + \dot{z}^2\right) - V(x, y, z).$$

With the coordinates $q_1 = x$, $q_2 = y$, $q_3 = z$, and

$$p_1 = \frac{\partial L}{\partial \dot{x}} = \dot{x} - \Omega y, \qquad p_2 = \frac{\partial L}{\partial \dot{y}} = \dot{y} + \Omega x, \qquad p_3 = \frac{\partial L}{\partial \dot{z}} = \dot{z},$$

we obtain the Hamiltonian $H = p_1\dot{q}_1 + p_2\dot{q}_2 + p_3\dot{q}_3 - L$ or

$$H = \frac{1}{2}\left(p_1^2 + p_2^2 + p_3^2\right) + \Omega(p_1 q_2 - p_2 q_1) + A\log\left(C + \frac{q_1^2}{a^2} + \frac{q_2^2}{b^2} + \frac{q_3^2}{c^2}\right).$$

⊙ From this Hamiltonian, derive the equations of motion, $\dot{q}_i = \partial H/\partial p_i$ and $\dot{p}_i = -\partial H/\partial q_i$. Solve them by using the initial values (taken from [2]):

$$q_1(0) = 2.5, \qquad q_2(0) = 0, \qquad q_3(0) = 0, \qquad p_1(0) = 0, \qquad p_3(0) = 0,$$

while $p_2(0)$ is the larger of the solutions of the equation $H = 2$. Use the parameters $A = C = 1$, $\Omega = 0.25$, $a = 1.25$, $b = 1$, and $c = 0.75$. Compute the solution on the interval $0 \le t \le 1000000$ by the RK4 method (7.10) with steps $h = 0.1$ and 0.025. Plot the orbits (connect the coordinates q_1, q_2, and q_3 as in Fig. 7.24 (top)). Plot the Poincaré sections of the solution (coordinates q_1 and q_3) with the half-plane $q_1 > 0$, $q_2 = 0$, $\dot{q}_2 > 0$ (Fig. 7.24 (bottom)). Plot the time evolution of the energy H.

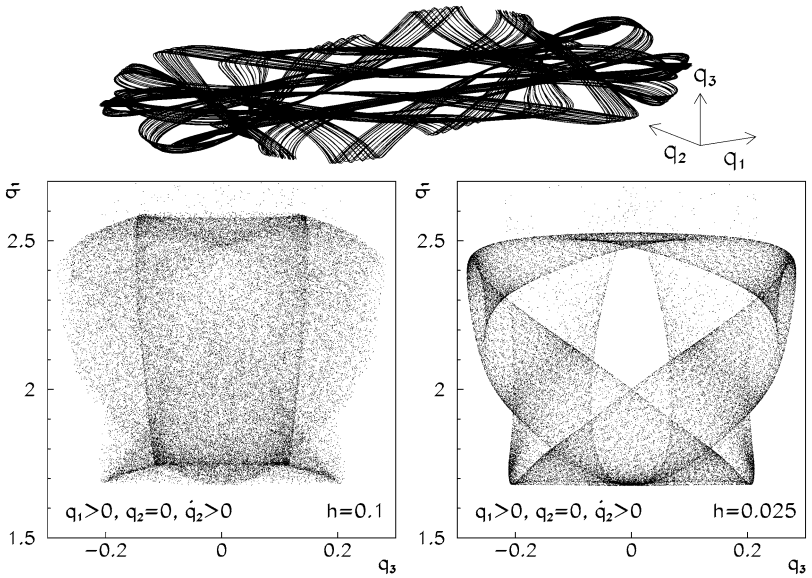


Fig. 7.24 Galactic dynamics. [*Top*] Trajectory until $t = 10000$. [*Bottom*] Poincaré sections of the trajectory for $0 \le t \le 1000000$ (coordinates q_1 and q_3) with the half-plane $q_1 > 0$, $q_2 = 0$, $\dot{q}_2 > 0$. [*Bottom left*] RK4 method, $h = 0.1$ (46230 intersections). [*Bottom right*] RK4 method, $h = 0.025$ (47022 intersections)

⊕ Solve the problem by using the fifth-order implicit method Radau 5 and the sixth-order Gauss method defined in Sect. 7.9. Use the step sizes $h = 0.1$, 0.2, and 0.4. Again, plot the time evolution of the energy H.

References

1. E.A. Coddington, N. Levinson, *Theory of Ordinary Differential Equations* (Krieger, Malabar, 1984)
2. E. Hairer, S.P. Nørsett, G. Wanner, *Solving Ordinary Differential Equations I; Nonstiff Problems*. Springer Series in Computational Mathematics, vol. 8 (Springer, Berlin, 2000)
3. A.R. Curtis, High-order explicit Runge Kutta formulae, their uses and limitations. J. Inst. Math. Appl. **16**, 35 (1975)
4. E. Hairer, A Runge–Kutta method of order 10. J. Inst. Math. Appl. **21**, 47 (1978)
5. J.C. Butcher, *Numerical Methods for Ordinary Differential Equations*, 2nd edn. (Wiley, Chichester, 2008)
6. E. Isaacson, H.B. Keller, *Analysis of Numerical Methods* (Wiley, New York, 1966)
7. P. Henrici, *Discrete Variable Methods in Ordinary Differential Equations* (Wiley, New York, 1962)
8. J.R. Cash, A.H. Karp, A variable order Runge–Kutta method for initial value problems with rapidly varying right-hand sides. ACM Trans. Math. Softw. **16**, 201 (1990)
9. W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 3rd edn. (Cambridge University Press, Cambridge, 2007). See also the equivalent handbooks in Fortran, Pascal and C, as well as http://www.nr.com

10. J.R. Dormand, P.J. Prince, A family of embedded Runge–Kutta formulae. J. Comput. Appl. Math. **6**, 19 (1980)
11. P.J. Prince, J.R. Dormand, High order embedded Runge–Kutta formulae. J. Comput. Appl. Math. **7**, 67 (1981)
12. Numerical Algorithms Group, http://www.nag.co.uk
13. J.H.E. Cartwright, O. Piro, The dynamics of Runge–Kutta methods. Int. J. Bifurc. Chaos **2**, 427 (1992)
14. J.D. Day, Run time estimation of the spectral radius of Jacobians. J. Comput. Appl. Math. **11**, 315 (1984)
15. R. Bulirsch, J. Stoer, Fehlerabschätzungen und Extrapolation mit rationalen Funktionen bei Verfahren vom Richardson-Typus. Numer. Math. **6**, 413 (1964)
16. E. Hairer, A. Ostermann, Dense output for extrapolation methods. Numer. Math. **58**, 419 (1990)
17. J.D. Hoffman, *Numerical Methods for Engineers and Scientists*, 2nd edn. (Marcel Dekker, New York, 2001)
18. J.R. Dormand, P.J. Prince, New Runge–Kutta algorithms for numerical simulation in dynamical astronomy. Celest. Mech. **18**, 223 (1978)
19. S. Filippi, J. Gräf, New Runge–Kutta–Nyström formula-pairs of order 8(7), 9(8), 10(9) and 11(10) for differential equations of the form $y'' = f(t, y)$. J. Comput. Appl. Math. **14**, 361 (1986)
20. E. Hairer, G. Wanner, *Solving Ordinary Differential Equations II; Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics, vol. 14 (Springer, Berlin, 2004)
21. J.C. Butcher, Implicit Runge–Kutta processes. Math. Comput. **18**, 50 (1964)
22. B.L. Ehle, High order A-stable methods for the numerical solution of systems of D.E.'s. BIT Numer. Math. **8**, 276 (1968)
23. L.F. Shampine, M.W. Reichelt, The Matlab ODE suite. SIAM J. Sci. Comput. **18**, 1 (1997)
24. C.F. Curtiss, J.O. Hirschfelder, Integration of stiff equations. Proc. Natl. Acad. Sci. **38**, 235 (1952)
25. G. Dahlquist, A special stability problem for linear multistep methods. BIT Numer. Math. **3**, 27 (1963)
26. H. Goldstein, *Classical Mechanics*, 2nd edn. (Addison-Wesley, Reading, 1981)
27. V.I. Arnol'd, *Mathematische Methoden der klassischen Mechanik* (VEB Deutscher Verlag der Wissenschaften, Berlin, 1988)
28. E. Hairer, C. Lubich, G. Wanner, *Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations*, 2nd edn. (Springer, Berlin, 2006)
29. Z. Ge, J.E. Marsden, Lie–Poisson integrators and Lie–Poisson Hamilton–Jacobi theory. Phys. Lett. A **133**, 134 (1988)
30. J.M. Sanz-Serna, Runge–Kutta schemes for Hamiltonian systems. BIT Numer. Math. **28**, 877 (1988)
31. R.I. McLachlan, G.R.W. Quispel, Geometric integrators for ODEs. J. Phys. A, Math. Gen. **39**, 5251 (2006)
32. É. Forest, Geometric integration for particle accelerators. J. Phys. A, Math. Gen. **39**, 5321 (2006)
33. T. Bridges, S. Reich, Numerical methods for Hamiltonian PDEs. J. Phys. A, Math. Gen. **39**, 5287 (2006)
34. A. Hanslmeier, R. Dvorak, Numerical integration with Lie series. Astron. Astrophys. **132**, 203 (1984)
35. M. Delva, Integration of the elliptic restricted three-body problem with Lie series. Celest. Mech. **34**, 145 (1984)
36. H. Lichtenegger, The dynamics of bodies with variable masses. Celest. Mech. **34**, 357 (1984)
37. S. Eggl, R. Dvorak, *An Introduction to Common Numerical Integration Codes Used in Dynamical Astronomy*. Lecture Notes in Physics, vol. 790 (Springer, Berlin, 2010), p. 431
38. E.N. Lorenz, Deterministic nonperiodic flow. J. Atmos. Sci. **20**, 130 (1963)

39. A. Trevisan, F. Pancotti, Periodic orbits, Lyapunov vectors, and singular vectors in the Lorenz System. J. Atmos. Sci. **55**, 390 (1998)
40. GNU Multi Precision (GMP), Free Library for Arbitrary Precision Arithmetic. http://gmplib.org
41. J.C. Sprott, *Chaos and Time-Series Analysis* (Oxford University Press, Oxford, 2003). See also http://sprott.physics.wisc.edu/chaostsa
42. I. Kuščer, A. Kodre, H. Neunzert, *Mathematik in Physik und Technik* (Springer, Berlin, 1993)
43. C. Moler, *Numerical Computing with MATLAB* (SIAM, Philadelphia, 2008)
44. J.C.G. Walker, *Numerical Adventures with Geochemical Cycles* (Oxford University Press, Oxford, 1991)
45. Y. Kuramoto, *Chemical Oscillations, Waves, and Turbulence* (Springer, Berlin, 1984)
46. S.H. Strogatz, From Kuramoto to Crawford: exploring the onset of synchronization in populations of coupled oscillators. Physica D **143**, 1 (2000)
47. E. Ott, *Chaos in Dynamical Systems*, 2nd edn. (Cambridge University Press, Cambridge, 2002)
48. G.B. Ermentrout, J. Rinzel, Analysis of neural excitability and oscillations, in *Methods in Neuronal Modelling: From Synapses to Networks"*, ed. by C. Koch, I. Segev (MIT Press, Cambridge, 1989)
49. C. Morris, H. Lecar, Voltage oscillations in the barnacle giant muscle fiber. Biophys. J. **35**, 193 (1981)
50. S.A. Chin, C.R. Chen, Forward symplectic integrators for solving gravitational few-body problems. Celest. Mech. Dyn. Astron. **91**, 301 (2005)
51. P.A. Wiegert, K.A. Innanen, An asteroidal companion to the Earth. Nature **387**, 685 (1997)
52. M. Connors, P. Chodas, S. Mikkola, Discovery of an asteroid and quasi-satellite in an Earth-like horseshoe orbit. Meteorit. Planet. Sci. **37**, 1435 (2002)
53. D.B. Taylor, Horseshoe periodic orbits in the restricted problem of three bodies for a Sun–Jupiter mass ratio. Astron. Astrophys. **103**, 288 (1981)
54. P.W. Sharp, Comparisons of integrators on a diverse collection of restricted three-body test problems. IMA J. Numer. Anal. **24**, 557 (2004)
55. E. Barrabés, S. Mikkola, Families of periodic horseshoe orbits in the restricted three-body problem. Astron. Astrophys. **432**, 1115 (2005)
56. B. Saltzman, Finite amplitude free convection as an initial value problem, I. J. Atmos. Sci. **19**, 329 (1962)
57. L.D. Landau, E.M. Lifshitz, *Course of Theoretical Physics, Vol. 1: Mechanics*, 3rd edn. (Butterworth-Heinemann, Oxford, 2003)
58. S. Bleher, E. Ott, C. Grebogi, Routes to chaotic scattering. Phys. Rev. Lett. **63**, 919 (1990)
59. M. Stix, *The Sun. An Introduction*, 2nd edn. (Springer, Berlin, 2002)
60. G.R. Caughlan, W.A. Fowler, Thermonuclear reaction rates V. At. Data Nucl. Data Tables **40**, 283 (1988). The data collection is also accessible at the website http://www.phy.ornl.gov/astrophysics/data/cf88/
61. C. Angulo et al., A compilation of charged-particle induced thermonuclear reaction rates. Nucl. Phys. A **656**, 3 (1999). See also the website of the NACRE Collaboration. http://pntpm.ulb.ac.be/nacre.htm
62. E.L. Schatzman, F. Praderie, *The Stars* (Springer, Berlin, 1993)
63. J.R. Field, R.M. Noyes, Oscillations in chemical systems, IV: limit cycle behavior in a model of a real chemical reaction. J. Chem. Phys. **60**, 1877 (1974)
64. C. Störmer, Sur les trajectoires des corpuscules électrisés. Arch. Sci. Phys. Nat., Genève **24**, 5–18 (1907). 113–158, 221–247
65. J. Binney, S. Tremaine, *Galactic Dynamics* (Princeton University Press, Princeton, 1987)

Chapter 8
Boundary-Value Problems for ODE

In this chapter we discuss methods for the solution of problems with ordinary differential equations, where we require the solution to satisfy the equation within the definition domain, and boundary conditions at its edges. We seek, for example, the function y that solves the equation

$$y'' = f\left(x, y, y'\right)$$

for $x \in R = [a, b]$, with mixed boundary conditions

$$\alpha_0 y(a) - \alpha_1 y'(a) = \alpha, \quad |\alpha_0| + |\alpha_1| \neq 0,$$
$$\beta_0 y(b) + \beta_1 y'(b) = \beta, \quad |\beta_0| + |\beta_1| \neq 0,$$

where the coefficients satisfy $\alpha_0\alpha_1 \geq 0$, $\beta_0\beta_1 \geq 0$, and $|\alpha_0| + |\beta_0| \neq 0$. When does the corresponding solution exist at all? If f is continuous and continuously differentiable on R, and if it satisfies the Lipschitz condition (7.4) on R for both y and y', and if for some $M > 0$ we have $\partial f/\partial y > 0$ and $|\partial f/\partial y'| \leq M$, and, finally, if $\alpha_0\alpha_1 \geq 0$ and $\beta_0\beta_1 \geq 0$, such a boundary-value problem has a unique solution [1]. In the simplified case when f is linear in y and y',

$$-y'' + p(x)y' + q(x)y = r(x),$$

where p, q, and r are continuous on R, for each α and β a unique solution exists precisely when $q(x) > 0$ for each $x \in R$. Sometimes we impose additional conditions that the solution should fulfill within the domain R. The corresponding existence theorems can be found in [2, 3].

Uniqueness of solutions of non-linear problems is much harder to fathom than in linear problems: a seemingly simple problem $y'' = -\delta e^y$ with conditions $y(0) = y(1) = 0$ (Gelfand–Bratu equation of diffusion-reaction kinetics in a layer) has two solutions for $0 < \delta < \delta_c \approx 3.51$ while the solution does not exist for $\delta > \delta_c$. For the spherical diffusion-reaction problem $y'' + (2/x)y' = \phi^2 y + \exp[\gamma\beta(1 - y)/(1 + \beta(1 - \gamma))]$ with conditions $y'(0) = 0$, $y(1) = 1$ at least 15 solutions are known to exist. How do we numerically compute all of them?

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9_8,

8.1 Difference Methods for Scalar Boundary-Value Problems

In developing difference schemes for boundary-value problems we utilize our knowledge from the methods used for initial-value problems (Chap. 7). Our cornerstone will be the linear problem with Dirichlet boundary conditions,

$$Ly = -y'' + p(x)y' + q(x)y = r(x), \quad y(a) = \alpha,\ y(b) = \beta, \tag{8.1}$$

where L is a linear second-order differential operator and where p, q, and r are continuous on $[a, b]$. This is a classical two-point boundary-value problem in which the boundary conditions are expressed at two points ($x = a$ and $x = b$). More general multi-point problems with "boundary" conditions in the form $\sum_{j=1}^{N_\mathrm{b}} \alpha_j y(\xi_j) = \alpha$ at $a = \xi_1 < \xi_2 < \cdots < \xi_{N_\mathrm{b}} = b$, are discussed in [2, 3]. To the interval $[a, b]$ we assign an equidistant mesh $a = x_0, x_1, \ldots, x_N = b$, where

$$x_j = a + jh, \quad j = 0, 1, \ldots, N,\ h = (b-a)/N. \tag{8.2}$$

Next, we discretize the derivatives and function values in the operator L, and denote $y(x_j) = y_j$. For example, the first derivative can be discretized by the non-symmetric (one-sided) differences with the discretization error $\mathcal{O}(h)$,

$$y'_j = \frac{y_{j+1} - y_j}{h} - \frac{h}{2}y''(\xi), \tag{8.3}$$

$$y'_j = \frac{y_j - y_{j-1}}{h} + \frac{h}{2}y''(\xi), \tag{8.4}$$

where ξ is some point positioned between the extreme points appearing in the formula. The order of the error can be improved by using symmetric (central) or non-symmetric differences that include more mesh points, for example:

$$y'_j = \frac{y_{j+1} - y_{j-1}}{2h} - \frac{h^2}{6}y'''(\xi), \tag{8.5}$$

$$y'_j = \frac{-y_{j+2} + 8y_{j+1} - 8y_{j-1} + y_{j-2}}{12h} + \frac{h^4}{30}y^{(5)}(\xi), \tag{8.6}$$

$$y'_j = \frac{-y_{j+2} + 4y_{j+1} - 3y_j}{2h} + \frac{h^2}{3}y'''(\xi), \tag{8.7}$$

$$y'_j = \frac{3y_j - 4y_{j-1} + y_{j-2}}{2h} + \frac{h^2}{3}y'''(\xi). \tag{8.8}$$

Expressions (8.5) and (8.6) are symmetric, but at the boundary points x_0 and x_N they require us to access the points x_{-1} and x_{N+1} (or even x_{-2} and x_{N+2}) beyond the mesh, and write additional difference equations for them. Expressions (8.7) and (8.8) are non-symmetric, but at the boundaries they can be applied directly. One-sided differences may have more natural physical backgrounds: we prefer to differentiate in the "*upwind*" direction from which the information is arriving, for

example, facing the incoming wave. We shall encounter similar choices in the discretization of hyperbolic partial differential equations in Chap. 9.

The second derivative may be approximated by the central difference

$$y_j'' = \frac{y_{j+1} - 2y_j + y_{j-1}}{h^2} - \frac{h^2}{12} y^{(4)}(\xi).$$

We use u_j to denote the approximate solution at x_j. By using the central difference for the first derivative we obtain the difference scheme

$$L_h u_j = -\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} + p(x_j)\frac{u_{j+1} - u_{j-1}}{2h} + q(x_j)u_j = r(x_j), \quad (8.9)$$

where L_h is a second-order difference operator. The scheme is applicable for $j = 1, 2, \ldots, N-1$, and the Dirichlet boundary conditions fix the remaining two values $u_0 = \alpha$ and $u_N = \beta$. We collect the solution components $u_1, u_2, \ldots, u_{N-1}$ in the vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_{N-1})^{\mathrm{T}}$, and rewrite the difference equation in the matrix form $A\boldsymbol{u} = \boldsymbol{r}$, where

$$A = \begin{pmatrix} b_1 & c_1 & 0 & & \\ a_2 & b_2 & c_2 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & a_{N-2} & b_{N-2} & c_{N-2} \\ & & 0 & a_{N-1} & b_{N-1} \end{pmatrix}, \qquad \boldsymbol{r} = \begin{pmatrix} \frac{1}{2}h^2 r(x_1) - a_1\alpha \\ \frac{1}{2}h^2 r(x_2) \\ \vdots \\ \frac{1}{2}h^2 r(x_{N-2}) \\ \frac{1}{2}h^2 r(x_{N-1}) - c_{N-1}\beta \end{pmatrix}.$$

The matrix A is tridiagonal, with the matrix elements

$$a_j = -\frac{1}{2}\left[1 + \frac{h}{2}p(x_j)\right], \qquad b_j = 1 + \frac{h^2}{2}q(x_j), \qquad c_j = -\frac{1}{2}\left[1 - \frac{h}{2}p(x_j)\right].$$

In the case of mixed boundary conditions, $\alpha_0 y(a) - \alpha_1 y'(a) = \alpha$, we can take the one-sided difference (8.7) and use it to approximate the boundary condition as

$$\alpha_0 u_0 - \alpha_1 \frac{-u_2 + 4u_1 - 3u_0}{2h} = \alpha.$$

This gives us the expression

$$u_0 = \frac{\alpha_1}{2h\alpha_0 + 3\alpha_1}(4u_1 - u_2) + \frac{2h\alpha}{2h\alpha_0 + 3\alpha_1},$$

which we insert in (8.9) at $j = 1$. When the coefficients in front of u_1 and u_2 are collected and the remainder is put to the right side of the equation, we again find a matrix equation for $\boldsymbol{u}$ with a tridiagonal matrix A, in which only the first row is modified:

$$b_1 \to b_1 + \frac{4\alpha_1 a_1}{2h\alpha_0 + 3\alpha_1},$$

$$c_1 \to c_1 - \frac{\alpha_1 a_1}{2h\alpha_0 + 3\alpha_1},$$
$$\frac{1}{2}h^2 r(x_1) - a_1\alpha \to \frac{1}{2}h^2 r(x_1) - \frac{2ha_1\alpha}{2h\alpha_0 + 3\alpha_1},$$

where the denominators should satisfy $2h\alpha_0 + 3\alpha_1 \neq 0$.

8.1.1 Consistency, Stability, and Convergence

The order of the numerical error of difference schemes for boundary-value problems is defined by analogy to initial-value problems (Sect. 7.4). The discretization error at x_j is $\tau_j = L_h y(x_j) - Ly(x_j)$. For the scheme (8.9) this means

$$\tau_j = -\left[\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} - y''(x_j)\right] + p(x_j)\left[\frac{u_{j+1} - u_{j-1}}{2h} - y'(x_j)\right]$$
$$= -\frac{h^2}{12}\left[y^{(4)}(\xi) - 2p(x_j)y'''(\eta)\right] = \mathcal{O}\left(h^2\right), \tag{8.10}$$

where ξ and η are points from the interval $[x_{j-1}, x_{j+1}]$. The scheme (8.9) is *consistent* with the differential equation (8.1): for all functions y with a continuous second derivative y'' we observe $\tau_j \to 0$ when $h \to 0$. For functions y with a continuous fourth derivative on $[a, b]$ the difference operator L_h approximates the differential operator L to second order.

The notion of stability, however, needs to be introduced in a different manner than for initial-value problems. We say that the difference operator L_h is *stable* for some small enough h if a positive constant Λ can be found such that

$$|u_j| \leq \Lambda\left[\max\left\{|u_0|, |u_N|\right\} + \max_{1\leq j\leq N-1} |L_h u_j|\right]. \tag{8.11}$$

If on the interval $[a, b]$ the functions p and q can be bounded by $|p(x)| \leq P_+$ and $0 < Q_- \leq q(x) \leq Q_+$, it can be shown that the scheme (8.9) is stable if we set $h \leq 2/P_+$. Then (8.11) holds with the constant $\Lambda = \max\{1, 1/Q_-\}$ [1]. If, in addition, y is four times continuously differentiable on $[a, b]$, the stability theorem also provides a useful error estimate ($0 \leq j \leq N$):

$$\left|u_j - y(x_j)\right| \leq \Lambda\frac{h^2}{12}\left[\max_{a\leq x\leq b}\left|y^{(4)}(x)\right| + 2P_+ \max_{a\leq x\leq b}\left|y'''(x)\right|\right]. \tag{8.12}$$

As in initial-value problems, we define convergence of difference schemes for boundary-value problems: a stable difference scheme that is consistent with the differential equation to order h^p, is convergent at the same order, thus

$$h \to 0 \quad \Rightarrow \quad \max_{0\leq j\leq N}\left|u_j - y(x_j)\right| \to 0.$$

Increasing the Order by Extrapolation There is a lovely way to use the second-order difference scheme (8.9) and obtain the result which has fourth-order precision. We first solve the boundary-value problem on a mesh with spacings $h = (b-a)/N$, yielding the approximate solution $\boldsymbol{u}^{(h)}$. Then we use the scheme again, but on a mesh with spacings half as large, $h/2 = (b-a)/(2N)$, and get the values $\boldsymbol{u}^{(h/2)}$. The improved approximation for the values $y(x)$ on the mesh with spacings h is then

$$u_j^{(h/2,h)} = \frac{4}{3} u_{2j}^{(h/2)} - \frac{1}{3} u_j^{(h)}, \quad j = 0, 1, \ldots, N. \tag{8.13}$$

The approximation (8.13) is of order four,

$$\left\| \boldsymbol{u}^{(h/2,h)} - \boldsymbol{y} \right\| = \mathcal{O}\left(h^4\right).$$

By further halvings even higher orders can be attained (see Sect. 8.2).

8.1.2 Non-linear Scalar Boundary-Value Problems

Solving non-linear boundary-value problems is more demanding than solving linear ones. The discretization of the differential equation itself is usually easy: the main work to be done is the solution of the system of non-linear equations that follows from this discretization. We discuss boundary-value problems of the form

$$Ly = -y'' + f\left(x, y, y'\right) = 0 \tag{8.14}$$

on the interval $[a, b]$ with boundary conditions $y(a) = \alpha$ and $y(b) = \beta$. Assume that f is (once) continuously differentiable with respect to y and y', and that

$$0 < Q_- \le \frac{\partial f}{\partial y} \le Q_+, \quad \left| \frac{\partial f}{\partial y'} \right| \le P_+,$$

for some positive constants Q_-, Q_+, and P_+. The difference approximation of the boundary-value problem (8.14) is

$$L_h u_j = -\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} + f\left(x_j, u_j, \frac{u_{j+1} - u_{j-1}}{2h}\right) = 0, \tag{8.15}$$

and it applies for $j = 1, 2, \ldots, N-1$, while the boundary conditions again dictate $u_0 = \alpha$ and $u_N = \beta$. With the above assumptions for $\partial f/\partial y$ and $\partial f/\partial y'$, and by choosing $h \le 2/P_+$ the difference scheme (8.15) is stable, is consistent with the differential equation to order $\mathcal{O}(h^2)$, and has the error estimate (8.12) as for the linear scalar problem.

Equation (8.15) in matrix form represents a system of non-linear equations. We solve it by explicit iteration [1] or by using Newton's method in which at each step

a matrix equation needs to be solved. The explicit iteration is

$$u_j^{(n+1)} = \frac{1}{1+\omega}\left[\frac{1}{2}\left(u_{j+1}^{(n)} + u_{j-1}^{(n)}\right) + \omega u_j^{(n)} - \frac{h^2}{2} f\left(x_j, u_j^{(n)}, \frac{u_{j+1}^{(n)} - u_{j-1}^{(n)}}{2h}\right)\right],$$

where $j = 1, 2, \ldots, N-1$, and the boundary values are $u_0 = \alpha$ and $u_N = \beta$. We start the iteration with some initial approximation $\boldsymbol{u}^{(0)}$, and the speed of convergence is determined by the parameter ω. By choosing $\omega \geq \frac{1}{2}h^2 Q_+$ the iteration converges for any initial approximation [1].

In order to use the Newton method, we rewrite (8.15) as $\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{0}$, where $\boldsymbol{F} = (F_1(\boldsymbol{u}), F_2(\boldsymbol{u}), \ldots, F_{N-1}(\boldsymbol{u}))^{\mathrm{T}}$ with $F_j(\boldsymbol{u}) = \frac{1}{2}h^2(L_h u_j)$. For the system of equations $\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{0}$, Newton's method is an iteration to the fixed point

$$\boldsymbol{u}^{(n+1)} = \boldsymbol{G}\left(\boldsymbol{u}^{(n)}\right),$$

where

$$\boldsymbol{G}(\boldsymbol{u}) = \boldsymbol{u} - \left[J(\boldsymbol{u})\right]^{-1} \boldsymbol{F}(\boldsymbol{u})$$

is the iteration function. Here $J(\boldsymbol{u})$ is the Jacobi matrix which is tridiagonal,

$$J(\boldsymbol{u}) = \frac{\partial \boldsymbol{F}(\boldsymbol{u})}{\partial \boldsymbol{u}} = \begin{pmatrix} B_1(\boldsymbol{u}) & C_1(\boldsymbol{u}) & & & \\ A_2(\boldsymbol{u}) & B_2(\boldsymbol{u}) & C_2(\boldsymbol{u}) & & \\ & \ddots & \ddots & \ddots & \\ & A_{N-2}(\boldsymbol{u}) & B_{N-2}(\boldsymbol{u}) & C_{N-2}(\boldsymbol{u}) & \\ & & A_{N-1}(\boldsymbol{u}) & B_{N-1}(\boldsymbol{u}) & \end{pmatrix},$$

with the matrix elements

$$A_j(\boldsymbol{u}) = -\frac{1}{2}\left[1 + \frac{h}{2}\frac{\partial f}{\partial y'}\left(x_j, u_j, \frac{u_{j+1} - u_{j-1}}{2h}\right)\right],$$

$$B_j(\boldsymbol{u}) = 1 + \frac{h^2}{2}\frac{\partial f}{\partial y}\left(x_j, u_j, \frac{u_{j+1} - u_{j-1}}{2h}\right),$$

$$C_j(\boldsymbol{u}) = -\frac{1}{2}\left[1 - \frac{h}{2}\frac{\partial f}{\partial y'}\left(x_j, u_j, \frac{u_{j+1} - u_{j-1}}{2h}\right)\right].$$

(The boundary conditions $u_0 = \alpha$ and $u_N = \beta$ also enter through the quantities $B_1(\boldsymbol{u})$, $C_1(\boldsymbol{u})$, $A_{N-1}(\boldsymbol{u})$, $B_{N-1}(\boldsymbol{u})$, $F_1(\boldsymbol{u})$, and $F_{N-1}(\boldsymbol{u})$.) In each step of the Newton iteration we are solving a system of linear equations

$$J\left(\boldsymbol{u}^{(n)}\right)\Delta\boldsymbol{u}^{(n)} = -\boldsymbol{F}\left(\boldsymbol{u}^{(n)}\right), \tag{8.16}$$

and the solution $\Delta\boldsymbol{u}^{(n)}$ of this system gives us the next approximation of the solution

$$\boldsymbol{u}^{(n+1)} = \boldsymbol{u}^{(n)} + \Delta\boldsymbol{u}^{(n)}, \quad n = 0, 1, 2, \ldots. \tag{8.17}$$

We repeat the iteration until the difference between subsequent approximations (as measured in some norm) drops below the specified tolerance.

Especially in the cases where the boundary-value problem is expected to have multiple solutions, one should choose the initial approximation $\boldsymbol{u}^{(0)}$ carefully. We try to exploit a known symmetry property of the equation or boundary conditions, or perhaps the asymptotic behavior of a similar equation. The initial approximation should at least satisfy the boundary conditions. It also makes sense to precede the Newton method by a few cycles of the explicit iteration.

Example A problem of the form (8.14) is the Gelfand–Bratu equation

$$y'' = -\delta \mathrm{e}^{y}, \quad 0 < x < 1,\ 0 < \delta < 3.51, \tag{8.18}$$

with boundary conditions $y(0) = y(1) = 0$. To solve it, we use Newton's method and the approximation (8.15), where $f(x, y, y') = -\delta \mathrm{e}^{y}$. The function f depends only on y, so the off-diagonal elements of the Jacobi matrix are trivial:

$$A_j(\boldsymbol{u}) = -\frac{1}{2}, \qquad B_j(\boldsymbol{u}) = 1 + \frac{h^2}{2}\left(-\delta \mathrm{e}^{u_j}\right), \qquad C_j(\boldsymbol{u}) = -\frac{1}{2}.$$

The components of the right side of (8.16) for $j = 1, 2, \ldots, N-1$ are

$$F_j(\boldsymbol{u}) = \frac{1}{2}h^2(L_h u_j) = \frac{h^2}{2}\left[-\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} - \delta \mathrm{e}^{u_j}\right].$$

The boundary conditions, which do not change in (8.16–8.17), are $u_0 = u_N = 0$. We start the iteration with the initial approximation $y(x) = \mu x(1-x)$,

$$u_j = \mu x_j (1 - x_j), \quad j = 0, 1, \ldots, N,$$

where μ is a positive real constant. For $0 < \delta < 3.51$ the Bratu–Gelfand problem has two solutions (Problem 8.9.1) and by using different values of μ Newton's iteration may bounce between these two solutions, or it may not converge at all (Fig. 8.1 (left)). The dependence of the norm of the last update in the iteration and the difference between the numerical and analytic solution on the length of the subintervals is shown in Fig. 8.1 (right) in the case of $\delta = 1$.

When we solve non-linear boundary-value problems by simple iteration, Newton's method, or some other iterative method, we always obtain the numerical solution after a series of steps. During the iteration, the solution changes throughout the domain and *relaxes* to the true solution (the sequence of curves 1, 2, 3, 4 in Fig. 8.4 (left)); only the boundary conditions remain the same, e.g. as in the Dirichlet case $y(a) = \alpha$ and $y(b) = \beta$ in that figure. Section 8.3 on *shooting methods* will show us how to obtain the solution in an altogether different manner.

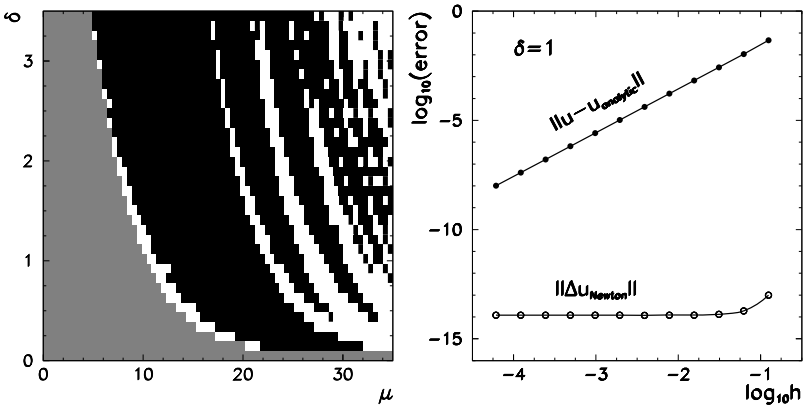


Fig. 8.1 Convergence of the Newton method for the Gelfand–Bratu problem. [*Left*] The iteration for different values of μ in the initial approximation for $y(x)$ and δ converges to the first (*gray areas*) or second solution (*black areas*). In *white areas* the iteration does not converge or the solution type cannot be ascertained within the specified tolerance. [*Right*] The norm of the last solution update $\Delta \boldsymbol{u}$ and the differences between the numerical and analytic solution as a function of subinterval length h, for $\delta = 1$

8.2 Difference Methods for Systems of Boundary-Value Problems

A large class of boundary-value problems can be written in the form of a system of M (not necessarily linear) first-order differential equations

$$\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y}), \quad a < x < b, \tag{8.19}$$

where $\boldsymbol{y} \in \mathbb{R}^M$ and $\{f_i\}_{i=1}^M$ are scalar functions (components of $\boldsymbol{f} : \mathbb{R}^{M+1} \to \mathbb{R}^M$) just as in (7.2). The general form of the boundary condition is

$$\boldsymbol{g}\big(\boldsymbol{y}(a), \boldsymbol{y}(b)\big) = \boldsymbol{0}.$$

As in the scalar boundary-value problems we discretize the interval $[a, b]$ on a mesh with points $a = x_0, x_1, \ldots, x_N = b$. We collect the approximate solutions in the $M(N+1)$-dimensional vector $\boldsymbol{u} = (\boldsymbol{u}_0^{\mathrm{T}}, \boldsymbol{u}_1^{\mathrm{T}}, \ldots, \boldsymbol{u}_N^{\mathrm{T}})^{\mathrm{T}}$, in which each component $\boldsymbol{u}_j$ ($j = 0, 1, \ldots, N$) is a M-dimensional vector.

There are many ways [1] to discretize the derivatives and the arguments of the function $\boldsymbol{f}$. The difference approximation of (8.19) with the trapezoidal formula for $\boldsymbol{f}$ is

$$L_h \boldsymbol{u}_j = \frac{\boldsymbol{u}_{j+1} - \boldsymbol{u}_j}{h} - \frac{1}{2}\big[\boldsymbol{f}(x_{j+1}, \boldsymbol{u}_{j+1}) + \boldsymbol{f}(x_j, \boldsymbol{u}_j)\big] = \boldsymbol{0},$$

where $j = 0, 1, \ldots, N-1$, and the boundary condition has the form

$$\boldsymbol{g}(\boldsymbol{u}_0, \boldsymbol{u}_N) = \boldsymbol{0}.$$

We write the discretized equation and boundary condition as a system of non-linear equations $\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{0}$, where $\boldsymbol{F} = (\boldsymbol{F}_0^{\mathrm{T}}(\boldsymbol{u}), \boldsymbol{F}_1^{\mathrm{T}}(\boldsymbol{u}), \ldots, \boldsymbol{F}_{N-1}^{\mathrm{T}}(\boldsymbol{u}), \boldsymbol{g}^{\mathrm{T}}(\boldsymbol{u}_0, \boldsymbol{u}_N))^{\mathrm{T}}$. The first N components of $\boldsymbol{F}$ are given by $\boldsymbol{F}_j(\boldsymbol{u}) = -h(L_h \boldsymbol{u}_j)$. The system can again be solved by Newton's iteration. At each step of (8.16) we solve

$$J\big(\boldsymbol{u}^{(n)}\big)\Delta\boldsymbol{u}^{(n)} = \begin{pmatrix} S_0 & R_0 & & & & & \\ & S_1 & R_1 & & & & \\ & & S_2 & R_2 & & & \\ & & & \ddots & \ddots & & \\ & & & & S_{N-1} & R_{N-1} \\ B_a & & & & & B_b \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{u}_0^{(n)} \\ \Delta\boldsymbol{u}_1^{(n)} \\ \Delta\boldsymbol{u}_2^{(n)} \\ \vdots \\ \Delta\boldsymbol{u}_{N-1}^{(n)} \\ \Delta\boldsymbol{u}_N^{(n)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{q}_0^{(n)} \\ \boldsymbol{q}_1^{(n)} \\ \boldsymbol{q}_2^{(n)} \\ \vdots \\ \boldsymbol{q}_{N-1}^{(n)} \\ \boldsymbol{\beta}^{(n)} \end{pmatrix}$$

for $\Delta\boldsymbol{u}^{(n)}$ and use it to compute the next approximation $\boldsymbol{u}^{(n+1)}$ by (8.17). The Jacobi matrix has dimensions $M(N+1) \times M(N+1)$ and has a characteristic block structure. It consists of $M \times M$ matrices determined by the difference scheme and the boundary condition:

$$S_j = I + \frac{h}{2}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\big(x_j, \boldsymbol{u}_j^{(n)}\big), \tag{8.20}$$

$$R_j = -I + \frac{h}{2}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\big(x_{j+1}, \boldsymbol{u}_{j+1}^{(n)}\big), \tag{8.21}$$

$$B_a = \frac{\partial \boldsymbol{g}(\boldsymbol{u}, \boldsymbol{v})}{\partial \boldsymbol{u}}, \qquad B_b = \frac{\partial \boldsymbol{g}(\boldsymbol{u}, \boldsymbol{v})}{\partial \boldsymbol{v}} \quad \text{with } \boldsymbol{u} = \boldsymbol{u}_0^{(n)},\ \boldsymbol{v} = \boldsymbol{u}_N^{(n)}, \tag{8.22}$$

while the right side of the equation is given by

$$\boldsymbol{q}_j^{(n)} = \big(\boldsymbol{u}_{j+1}^{(n)} - \boldsymbol{u}_j^{(n)}\big) - \frac{h}{2}\big[\boldsymbol{f}\big(x_{j+1}, \boldsymbol{u}_{j+1}^{(n)}\big) + \boldsymbol{f}\big(x_j, \boldsymbol{u}_j^{(n)}\big)\big], \tag{8.23}$$

$$\boldsymbol{\beta}^{(n)} = -\boldsymbol{g}\big(\boldsymbol{u}_0^{(n)}, \boldsymbol{u}_N^{(n)}\big).$$

(As an exercise, repeat the derivation above with a trapezoidal scheme in which $\boldsymbol{y}'$ and $\boldsymbol{f}$ are computed at x_{j-1} and x_j instead of x_j and x_{j+1}.)

Example The second-order Gelfand–Bratu problem (see (8.18)) can be solved by the above method when it is written as a system of $M = 2$ first-order equations:

$$\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y}), \quad \boldsymbol{y} = \begin{pmatrix} u_0 \\ u_1 \end{pmatrix}, \ \boldsymbol{f}(x, \boldsymbol{y}) = \begin{pmatrix} u_1 \\ -\delta \mathrm{e}^{u_0} \end{pmatrix}.$$

We use $u_{0,j}$ and $u_{1,j}$ to denote the values of the first and second component, respectively, of the numerical solution $\boldsymbol{u}_j$ at x_j. The $M \times M = 2 \times 2$ matrices (8.20)

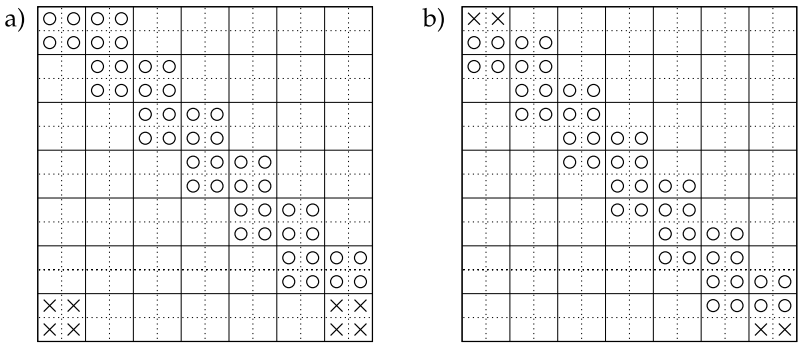


Fig. 8.2 The structure of the Jacobi matrix for systems of non-linear boundary-value problems for $M = 2$, $N = 6$. The *symbols* $\circ$ denote the elements originating in the difference scheme, while the *symbols* $\times$ denote the elements fixed by the boundary conditions. Matrices of the form (**a**) occur with unseparated boundary conditions $\boldsymbol{g}(\boldsymbol{u}_0, \boldsymbol{u}_N) = \boldsymbol{0}$. For improved numerical efficiency, one could attempt to separate the boundary conditions as $\boldsymbol{g}_1(\boldsymbol{y}(a)) = \boldsymbol{0}$ and $\boldsymbol{g}_2(\boldsymbol{y}(b)) = \boldsymbol{0}$, leading to the banded structure (**b**)

and (8.21) appearing in the Jacobi matrix are

$$S_j = \begin{pmatrix} 1 & \frac{h}{2} \\ -\frac{h}{2}\delta \exp(u_{0,j}^{(n)}) & 1 \end{pmatrix}, \qquad R_j = \begin{pmatrix} -1 & \frac{h}{2} \\ -\frac{h}{2}\delta \exp(u_{0,j+1}^{(n)}) & -1 \end{pmatrix}.$$

The right side of the Newton system is given by (8.23),

$$\boldsymbol{q}_j^{(n)} = \begin{pmatrix} u_{0,j+1} - u_{0,j} - \frac{h}{2}[u_{1,j+1} + u_{1,j}] \\ u_{1,j+1} - u_{1,j} + \frac{h}{2}[\delta \exp(u_{0,j+1}^{(n)}) + \delta \exp(u_{0,j}^{(n)})] \end{pmatrix}.$$

Equation (8.22) tells us that the boundary condition $\boldsymbol{g}(\boldsymbol{u}_0, \boldsymbol{u}_N) = \boldsymbol{0}$ is

$$B_a = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \qquad B_b = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \qquad \boldsymbol{\beta}^{(n)} = -\boldsymbol{g}\big(\boldsymbol{u}_0^{(n)}, \boldsymbol{u}_N^{(n)}\big) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

We choose an initial approximation $\boldsymbol{u}^{(0)}$ and run the iteration (8.16)–(8.17).

The Jacobi matrix has a deficiency which implies a higher numerical cost of solving the system of equations: the matrix B_a corresponding to the boundary condition at $x = a$ appears in the bottom left corner (Fig. 8.2(a)). This form occurs because the boundary conditions are not separated, $\boldsymbol{g}(\boldsymbol{u}_0, \boldsymbol{u}_N) = \boldsymbol{0}$ (or $B_a \boldsymbol{y}(a) + B_b \boldsymbol{y}(b) = \boldsymbol{\beta}$ in the linear case). Problems with unseparated conditions can be rephrased as problems with separated conditions $\boldsymbol{g}_1(\boldsymbol{y}(a)) = \boldsymbol{0}$ and $\boldsymbol{g}_2(\boldsymbol{y}(b)) = \boldsymbol{0}$ (or $B_1 \boldsymbol{y}(a) = \boldsymbol{\beta}_1$ and $B_2 \boldsymbol{y}(b) = \boldsymbol{\beta}_2$ in the linear case), yielding a nicer form of the Jacobi matrix (Fig. 8.2(b)). The price for this is the increase in the number of differential equations. Details on the separation of boundary conditions can be found in [2, 3].

8.2.1 Linear Systems

We made an exception and treated the more general class of non-linear systems first. Linear boundary-value problems with M first-order equations

$$\boldsymbol{y}' = A(x)\boldsymbol{y} + \boldsymbol{q}(x), \quad a < x < b, \tag{8.24}$$

and boundary conditions

$$B_a \boldsymbol{y}(a) + B_b \boldsymbol{y}(b) = \boldsymbol{\beta},$$

are just their special cases. Two simple difference schemes to solve linear systems of the form (8.24) [2, 3] are the trapezoidal scheme

$$\boldsymbol{u}_{j+1} - \boldsymbol{u}_j = \frac{h}{2}\big[A(x_{j+1})\boldsymbol{u}_{j+1} + A(x_j)\boldsymbol{u}_j\big] + \frac{h}{2}[\boldsymbol{q}_{j+1} + \boldsymbol{q}_j]$$

and the midpoint scheme

$$\boldsymbol{u}_{j+1} - \boldsymbol{u}_j = \frac{h}{2}A(x_{j+1/2})(\boldsymbol{u}_{j+1} + \boldsymbol{u}_j) + h\boldsymbol{q}_{j+1/2},$$

applicable at $j = 1, 2, \ldots, N-1$, while the boundary conditions in both cases are $B_a\boldsymbol{u}_0 + B_b\boldsymbol{u}_N = \boldsymbol{\beta}$. When the schemes are written in matrix form, we obtain a matrix with the same structure as in the Jacobi matrix for the non-linear case,

$$\begin{pmatrix} S_0 & R_0 & & & & \\ & S_1 & R_1 & & & \\ & & S_2 & R_2 & & \\ & & & \ddots & \ddots & \\ & & & & S_{N-1} & R_{N-1} \\ B_a & & & & 0 & B_b \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_{N-1} \\ \boldsymbol{u}_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{q}_0 \\ \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \\ \vdots \\ \boldsymbol{q}_{N-1} \\ \boldsymbol{\beta} \end{pmatrix}, \tag{8.25}$$

where the $M \times M$ matrices R_j and S_j and vectors $\boldsymbol{q}_j$ (dimension M) are listed in Table 8.1. The solution vector $\boldsymbol{u}$ and the vector $(\boldsymbol{q}^{\mathrm{T}}, \boldsymbol{\beta}^{\mathrm{T}})^{\mathrm{T}}$ have dimensions $M \times (N+1)$. Thus, the solutions of linear boundary-value problems can be found by solving systems of linear equations: there is no need for iterative methods.

8.2.2 Schemes of Higher Orders

In order to improve the convergence for systems of boundary-value problems we try to increase the order of the difference scheme. For systems of the form

$$\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y}),$$

Table 8.1 The elements of the matrix A and the vector at the right of (8.25) for the trapezoidal and midpoint scheme for the solution of the linear system $\boldsymbol{y}' = A\boldsymbol{y} + \boldsymbol{q}$. The $M \times M$ matrices B_a and B_b are given by the boundary condition $B_a\boldsymbol{y}(a) + B_b\boldsymbol{y}(b) = \boldsymbol{\beta}$

Trapezoidal scheme	Midpoint scheme
$S_j = I + \frac{h}{2}A(x_j)$	$S_j = I + \frac{h}{2}A(x_{j+1/2})$
$R_j = -I + \frac{h}{2}A(x_{j+1})$	$R_j = -I + \frac{h}{2}A(x_{j+1/2})$
$\boldsymbol{q}_j = -\frac{h}{2}[\boldsymbol{q}(x_{j+1}) + \boldsymbol{q}(x_j)]$	$\boldsymbol{q}_j = -h\boldsymbol{q}(x_{j+1/2})$

where $\boldsymbol{f}$ is easy to differentiate, a higher-order scheme can be found by replacing $\boldsymbol{f}$ on each subinterval $[x_j, x_{j+1}]$ by its Hermite interpolant of order $p = 2k$, and integrating the system $\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y})$ on this subinterval. We get a scheme of order p, for which the function $\boldsymbol{f}$ and its derivatives up to order $k-1$ at x_j and x_{j+1} are needed (for details see [2, 3]). For example, at $k = 2$ we obtain

$$\frac{\boldsymbol{u}_{j+1} - \boldsymbol{u}_j}{h} = \frac{1}{2}\big[\boldsymbol{f}(x_j, \boldsymbol{u}_j) + \boldsymbol{f}(x_{j+1}, \boldsymbol{u}_{j+1})\big] + \frac{h}{12}\big[\boldsymbol{f}'(x_j, \boldsymbol{u}_j) - \boldsymbol{f}'(x_{j+1}, \boldsymbol{u}_{j+1})\big],$$

which is of fourth-order and requires the computation of the total derivative

$$\boldsymbol{f}'(x_j, \boldsymbol{u}_j) = \left[\frac{\partial \boldsymbol{f}}{\partial x} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\boldsymbol{f}\right]_{(x_j, \boldsymbol{u}_j)}.$$

Higher orders can also be attained by extrapolating the mesh spacing h, in analogy to the scalar case (8.13). The approximation of a higher order is obtained by solving the boundary-value problem on a sequence of ever finer meshes (ever smaller spacings h) and constructing linear combinations of solutions from the individual meshes such that the discretization errors cancel out to an extent as large as possible. For example, we solve the boundary-value problem on two meshes with spacings h and $h/2$, and form the combination of the solutions

$$\boldsymbol{u}_j^{(h/2,h)} = \frac{4}{3}\boldsymbol{u}_{2j}^{(h/2)} - \frac{1}{3}\boldsymbol{u}_j^{(h)}. \tag{8.26}$$

The finer mesh can also have a spacing of $h/3$, and we form

$$\boldsymbol{u}_j^{(h/3,h)} = \frac{9}{8}\boldsymbol{u}_{3j}^{(h/3)} - \frac{1}{8}\boldsymbol{u}_j^{(h)}. \tag{8.27}$$

Either combination is fourth-order, $\|\boldsymbol{u}^{(h/2,h)} - \boldsymbol{y}\| \sim \|\boldsymbol{u}^{(h/3,h)} - \boldsymbol{y}\| = \mathcal{O}(h^4)$. Moreover, the approximations $\boldsymbol{u}^{(h/2,h)}$ and $\boldsymbol{u}^{(h/3,h)}$ can be merged in

$$\boldsymbol{u}_j^{(h/3,h/2)} = \frac{9}{5}\boldsymbol{u}_j^{(h/3,h)} - \frac{4}{5}\boldsymbol{u}_j^{(h/2,h)}, \tag{8.28}$$

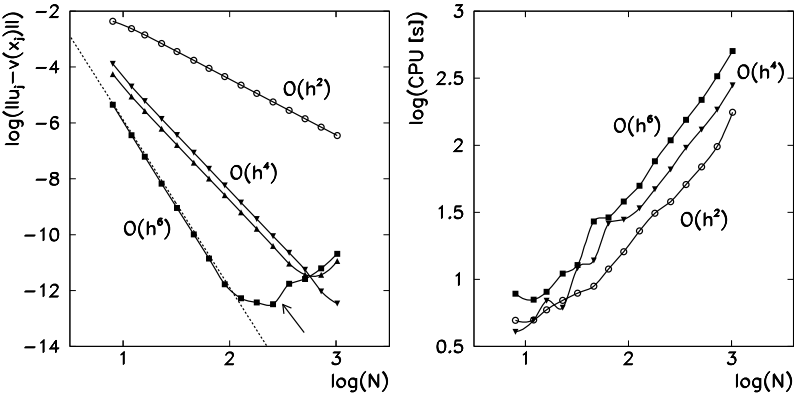


Fig. 8.3 [*Left*] Convergence of the numerical approximation to the exact solution of $y'' - \alpha(2x-1)y' - 2\alpha y = 0$ with boundary conditions $y(0) = y(1) = 1$ when the mesh spacing is decreased and higher-order methods are used. The basic method (8.9) is of order $\mathcal{O}(h^2)$, the approximations (8.26) and (8.27) are $\mathcal{O}(h^4)$, and (8.28) is $\mathcal{O}(h^6)$. We should not be oblivious to the possibility of round-off errors at large N (region to the right of the *arrow*)! [*Right*] Numerical cost of the approximations

resulting in a *sixth-order* approximation, $\|\boldsymbol{u}^{(h/3,h/2)} - \boldsymbol{y}\| = \mathcal{O}(h^6)$. Of course, formulas (8.26), (8.27), and (8.28) are also applicable to scalar problems.

A gradual fragmentation of the interval in order to increase the numerical precision (Fig. 8.3) obviously requires additional computation time. But it also allows us to obtain reliable local error estimates which aid us in determining the convergence criteria or in choosing the mesh size in regions where the numerical error is expected to increase. Details can be found in [2, 3].

8.3 Shooting Methods

The essence of shooting methods can be illustrated by the scalar linear problem

$$y'' + p(x)y' + q(x)y = r(x), \quad a \le x \le b,$$

with Dirichlet boundary conditions

$$y(a) = \alpha, \qquad y(b) = \beta.$$

Assume that this problem has a unique solution $y(x)$, which we try to find by integrating the *initial-value problem* with some values $y(a)$ and $y'(a)$ towards the right boundary of the interval $[a, b]$. The value $y(a) = \alpha$ is given by the boundary condition, while the slope $y'(a)$ is unknown. Let us guess this slope, the "shooting angle", $y'(a) = s_1$, and compute the solution of the differential equation up to $x = b$. We obtain the solution $y_1(x)$ (for example, curve 1 in Fig. 8.4 (right)) which in general

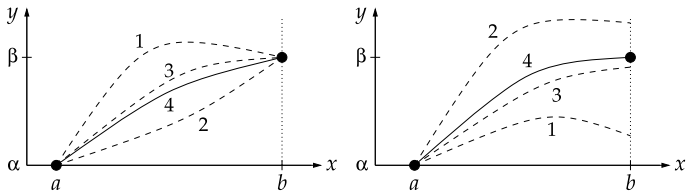


Fig. 8.4 [*Left*] The principle of a relaxation difference method. The numerical solution with an initial approximation satisfying the boundary conditions converges to the exact solution after some number of steps, maintaining $y(a) = \alpha$ and $y(b) = \beta$ at all times. [*Right*] The basic idea of the shooting method. We integrate the differential equation with the initial value $y(a) = \alpha$ and slope $y'(a)$, and keep on changing this slope until the second boundary condition $y(b) = \beta$ is satisfied to some precision

differs from the exact solution $y(x)$, since it does not satisfy the second boundary condition, $y_1(b) \neq y(b) = \beta$. Let us guess another angle, say, $y'(a) = s_2$, which corresponds to the solution $y_2(x)$ that in general also differs from the exact one, as $y_2(b) \neq y(b)$ and also $y_2(b) \neq y_1(b)$. Since the problem is *linear*, the true solution is the linear combination

$$y(x) = (1 - \theta)y_1(x) + \theta y_2(x),$$

where

$$\theta = \frac{\beta - y_1(b)}{y_2(b) - y_1(b)}.$$

(Mathematicians call this a *convex combination of* y_1 *and* y_2.) A linear boundary-value problem can thus be solved in just two steps. Figure 8.4 (right) shows two approximate solutions "1" and "2" from which the final one ("4") is computed.

8.3.1 Second-Order Linear Equations

Let us approach the solution of linear boundary-value problems by shooting methods in more generally. We follow [1] and discuss equations of the form

$$Ly = -y'' + p(x)y' + q(x)y = r(x), \quad a \le x \le b, \tag{8.29}$$

with the boundary conditions

$$a_0 y(a) - a_1 y'(a) = \alpha, \quad a_0 a_1 \ge 0, \ |a_0| + |a_1| \neq 0,$$
$$b_0 y(b) + b_1 y'(b) = \beta, \quad b_0 b_1 \ge 0, \ |b_0| + |b_1| \neq 0,$$

as well as $|a_0| + |b_0| \neq 0$. In addition to the continuity of the functions p, q, and r on $[a, b]$ we require that the corresponding *homogeneous problem* $Ly = 0$ with boundary conditions $a_0 y(a) - a_1 y'(a) = 0$ and $b_0 y(b) + b_1 y'(b) = 0$ has only a

trivial solution, $y(x) = 0$. Then the problem (8.29) has a unique solution [1]. Define the auxiliary function $y^{(1)}$ that solves the initial-value problem

$$Ly^{(1)} = r(x), \quad y^{(1)}(a) = -\alpha c_1, \; y^{(1)\prime}(a) = -\alpha c_0,$$

and the function $y^{(2)}$ that solves the corresponding homogeneous problem

$$Ly^{(2)} = 0, \quad y^{(2)}(a) = a_1, \; y^{(2)\prime}(a) = a_0,$$

where c_0 and c_1 are *any such constants* that $a_1 c_0 - a_0 c_1 = 1$. The function

$$y(x) = y(x; s) = y^{(1)}(x) + s y^{(2)}(x) \tag{8.30}$$

obviously satisfies the left boundary condition $a_0 y(a) - a_1 y'(a) = \alpha(a_1 c_0 - a_0 c_1) = \alpha$ and also solves (8.29) if some s can be found—a sort of generalized "shooting angle"—such that the boundary condition at the right,

$$\phi(s) = b_0 y(b; s) + b_1 y'(b; s) - \beta = 0,$$

is also satisfied. We insert (8.30) at $x = b$ in this expression and realize that the equation for $\phi(s)$ is linear in the parameter s, with the solution

$$s = \frac{\beta - b_0 y^{(1)}(b) - b_1 y^{(1)\prime}(b)}{b_0 y^{(2)}(b) + b_1 y^{(2)\prime}(b)}. \tag{8.31}$$

(This expression has a meaning if the denominator differs from zero. If it were zero, $y^{(2)}$ would be a non-trivial solution of the homogeneous problem $Ly = 0$, which is excluded by assumption.) Numerically we solve the initial-value problems $Ly^{(1)} = r(x)$ and $Ly^{(2)} = 0$ as two decoupled systems of two first-order differential equations,

$$\begin{pmatrix} y^{(1)} \\ v^{(1)} \end{pmatrix}' = \begin{pmatrix} v^{(1)} \\ pv^{(1)} + qy^{(1)} - r \end{pmatrix}, \quad \begin{matrix} y^{(1)}(a) = -\alpha c_1, \\ v^{(1)}(a) = -\alpha c_0, \end{matrix} \tag{8.32}$$

$$\begin{pmatrix} y^{(2)} \\ v^{(2)} \end{pmatrix}' = \begin{pmatrix} v^{(2)} \\ pv^{(2)} + qy^{(2)} \end{pmatrix}, \quad \begin{matrix} y^{(2)}(a) = a_1, \\ v^{(2)}(a) = a_0. \end{matrix} \tag{8.33}$$

These systems can be solved by any method for the solution of initial-value problems on the chosen mesh, e.g. (8.2). We denote the numerical solutions of (8.32) and (8.33) at x_j (mesh functions) by $\widetilde{y}_j^{(1)}$, $\widetilde{v}_j^{(1)}$, $\widetilde{y}_j^{(2)}$ in $\widetilde{v}_j^{(2)}$. At $x = a$ we apply the initial conditions

$$\widetilde{y}_0^{(1)} = -\alpha c_1, \qquad \widetilde{v}_0^{(1)} = -\alpha c_0, \qquad \widetilde{y}_0^{(2)} = a_1, \qquad \widetilde{v}_0^{(2)} = a_0,$$

and obtain the solution and its derivative at arbitrary x_j by the combinations

$$\widetilde{y}_j = \widetilde{y}_j^{(1)} + s\widetilde{y}_j^{(2)}, \tag{8.34}$$

$$\widetilde{v}_j = \widetilde{v}_j^{(1)} + s\widetilde{v}_j^{(2)}, \tag{8.35}$$

where

$$s = \frac{\beta - b_0\widetilde{y}_N^{(1)} - b_1\widetilde{v}_N^{(1)}}{b_0\widetilde{y}_N^{(2)} + b_1\widetilde{v}_N^{(2)}}. \tag{8.36}$$

Note that the approximate solution of the boundary-value problem is obtained only after both initial-value problems have been solved on the whole domain: in constructing (8.34) and (8.35) we need the parameter (8.36) that can be computed only when the mesh functions at $j = N$ become known.

For linear problems of the form (8.29) with smooth functions p, q, and r the procedure described above should work well: if a stable integration method with an error of order $\mathcal{O}(h^p)$ is used to solve the initial-value problem, the errors of the solution $\widetilde{y}$ and its derivative $\widetilde{v}$ are also of order $\mathcal{O}(h^p)$, thus

$$\left|\widetilde{y}_j - y(x_j)\right| = \mathcal{O}\left(h^p\right), \qquad \left|\widetilde{v}_j - y'(x_j)\right| = \mathcal{O}\left(h^p\right).$$

Still, this "naive" shooting method may lead to large numerical errors: why they occur and how we try to avoid them is discussed in Sect. 8.3.5.

8.3.2 Systems of Linear Second-Order Equations

Let us generalize this approach to systems of M linear second-order equations

$$L\boldsymbol{y} = -\boldsymbol{y}'' + P(x)\boldsymbol{y}' + Q(x)\boldsymbol{y} = \boldsymbol{r}(x), \quad a \le x \le b, \tag{8.37}$$

where P and Q are $M \times M$ matrices whose elements are continuous in x on $[a, b]$, while $\boldsymbol{y}$ and $\boldsymbol{r}$ are M-dimensional vectors. Let the boundary conditions be

$$\begin{aligned} A_0\boldsymbol{y}(a) - A_1\boldsymbol{y}'(a) &= \boldsymbol{\alpha}, \quad \det A_0 \neq 0, \\ B_0\boldsymbol{y}(b) + B_1\boldsymbol{y}'(b) &= \boldsymbol{\beta}, \end{aligned}$$

where A_0, A_1, B_0, and B_1 are constant matrices, and $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are constant vectors. We follow the steps from the scalar case (Sect. 8.3.1) and introduce a M-component vector function $\boldsymbol{y}^{(0)}$ that solves the initial-value problem

$$L\boldsymbol{y}^{(0)} = \boldsymbol{r}(x), \qquad A_0\boldsymbol{y}^{(0)}(a) - A_1\boldsymbol{y}^{(0)\prime}(a) = \boldsymbol{\alpha}, \;\; \boldsymbol{y}^{(0)\prime}(a) = \boldsymbol{0}. \tag{8.38}$$

Define the M-component vector functions $\boldsymbol{y}^{(m)}(x)$, $m = 1, 2, \ldots, M$, that solve the system of M homogeneous initial-value problems

$$L\boldsymbol{y}^{(m)} = \boldsymbol{0}, \qquad A_0\boldsymbol{y}^{(m)}(a) - A_1\boldsymbol{y}^{(m)\prime}(a) = \boldsymbol{0}, \;\; \boldsymbol{y}^{(m)\prime}(a) = \boldsymbol{I}^{(m)}, \tag{8.39}$$

where $\boldsymbol{I}^{(m)}$ is the unit vector with the value 1 at the mth position. We arrange the solutions of the system (8.39) in a $M \times M$ matrix,

$$Y(x) = \big(\boldsymbol{y}^{(1)}(x), \boldsymbol{y}^{(2)}(x), \ldots, \boldsymbol{y}^{(M)}(x)\big).$$

The solution of the original boundary-value problem (8.37) is then

$$\boldsymbol{y}(x) = \boldsymbol{y}(x;\boldsymbol{s}) = \boldsymbol{y}^{(0)}(x) + Y(x)\boldsymbol{s} = \boldsymbol{y}^{(0)}(x) + \sum_{m=1}^{M} s_m \boldsymbol{y}^{(m)}(x),$$

where the vector $\boldsymbol{s} = (s_1, s_2, \ldots, s_M)^{\mathrm{T}}$ is the root of the equation

$$\boldsymbol{\phi}(\boldsymbol{s}) = B_0 \boldsymbol{y}(b;\boldsymbol{s}) + B_1 \boldsymbol{y}'(b;\boldsymbol{s}) - \boldsymbol{\beta} = \boldsymbol{0}.$$

This equation is linear in the parameter $\boldsymbol{s}$ that can be computed by solving

$$\big[B_0 \boldsymbol{Y}(b) + B_1 \boldsymbol{Y}'(b)\big]\boldsymbol{s} = \boldsymbol{\beta} - B_0 \boldsymbol{y}^{(0)}(b) - B_1 \boldsymbol{y}^{(0)\prime}(b)$$

(compare this to (8.31) and the corresponding commentary).

We compute the solution of (8.37) analogously to the scalar case. On the uniform mesh (8.2) we integrate the initial-value problems (8.38) and (8.39), and obtain the mesh functions $\widetilde{y}_j^{(0)}$, $\widetilde{v}_j^{(0)}$, $\widetilde{y}_j^{(m)}$, and $\widetilde{v}_j^{(m)}$, $m = 1, 2, \ldots, M$, at x_j. The approximations for the function $\boldsymbol{y}(x)$ and its derivative $\boldsymbol{y}'(x)$ are

$$\widetilde{\boldsymbol{y}}_j(\boldsymbol{s}) = \widetilde{\boldsymbol{y}}_j^{(0)} + \sum_{m=1}^{M} s_m \widetilde{\boldsymbol{y}}_j^{(m)}, \tag{8.40}$$

$$\widetilde{\boldsymbol{v}}_j(\boldsymbol{s}) = \widetilde{\boldsymbol{v}}_j^{(0)} + \sum_{m=1}^{M} s_m \widetilde{\boldsymbol{v}}_j^{(m)}, \tag{8.41}$$

where $\boldsymbol{s}$ is the root of the equation

$$\boldsymbol{\phi}(\boldsymbol{s}) = B_0 \widetilde{\boldsymbol{y}}_N(\boldsymbol{s}) + B_1 \widetilde{\boldsymbol{v}}_N(\boldsymbol{s}) - \boldsymbol{\beta} = \boldsymbol{0}.$$

We should stress again that one needs to solve the (vector) initial-value problem (8.38) *and* the system of M (vector) initial-value problems (8.39): this gives us the (vector) parameter $\boldsymbol{s}$ that we use to construct the final solution of the boundary-value problem (8.37) by using (8.40) and (8.41). If the initial-value problems are integrated by a method with the error of order $\mathcal{O}(h^p)$, the error of the solution (or its derivative) of the boundary-value problem will be

$$\big\|\widetilde{\boldsymbol{y}}_j(\boldsymbol{s}) - \boldsymbol{y}(x_j)\big\| = \mathcal{O}\big(h^p\big), \qquad \big\|\widetilde{\boldsymbol{v}}_j(\boldsymbol{s}) - \boldsymbol{y}'(x_j)\big\| = \mathcal{O}\big(h^p\big).$$

8.3.3 Non-linear Second-Order Equations

Solving non-linear scalar boundary-value problems of the form

$$y'' = f\big(x, y, y'\big), \quad a \le x \le b,$$

with the boundary conditions

$$\begin{aligned} a_0 y(a) - a_1 y'(a) &= \alpha, \quad a_0, a_1 \ge 0, \\ b_0 y(b) + b_1 y'(b) &= \beta, \quad b_0, b_1 \ge 0,\ a_0 + b_0 > 0, \end{aligned}$$

allows us to apply the tools we learned in studying linear problems. With minor modifications, this subsection also closely follows [1]. Assume that the function $f(x, u, v)$ is continuous in x on the interval $[a, b]$ and is uniformly Lipschitz in u and v. If we further assume that

$$\frac{\partial f}{\partial u} > 0, \quad \left|\frac{\partial f}{\partial v}\right| \le C,\ C > 0,$$

such a boundary-value problem has a unique solution. As before, we seek its solution by solving the corresponding initial-value problem

$$\begin{aligned} y'' &= f\big(x, y, y'\big), \quad a \le x \le b, \\ y(a) &= a_1 s - c_1 \alpha, \\ y'(a) &= a_0 s - c_0 \alpha, \end{aligned}$$

where c_0 and c_1 are constants such that $a_1 c_0 - a_0 c_1 = 1$. The solution $u = u(x; s)$ of this initial-value problem is the solution of the original boundary-value problem precisely when the parameter s is the root of the equation

$$\phi(s) = b_0 y(b; s) + b_1 y'(b; s) - \beta = 0.$$

Let us denote $y = u$ and $y' = v$, and transform the initial-value problem to the system of two first-order equations,

$$\begin{pmatrix} u \\ v \end{pmatrix}' = \begin{pmatrix} v \\ f(x, u, v) \end{pmatrix}, \quad \begin{aligned} u(a) &= a_1 s - c_1 \alpha, \\ v(a) &= a_0 s - c_0 \alpha, \end{aligned} \tag{8.42}$$

which can be numerically solved by some method discussed in Chap. 7. The solutions of the system at $x = b$ are $u(b; s)$ and $v(b; s)$, which we use to form

$$\phi(s) = b_0 u(b; s) + b_1 v(b; s) - \beta \tag{8.43}$$

and check whether at the present parameter s we attain $\phi(s) = 0$. We repeat this procedure until this requirement is met to specified precision, or until the parameter s settles to its final value $s^\star$. For the computation of $s^\star$, Newton's method can be used, as described in the following.

Solving by Newton's Method We choose an initial approximation $s^{(0)}$ and iterate

$$s^{(\nu+1)} = s^{(\nu)} - \frac{\phi(s^{(\nu)})}{\phi'(s^{(\nu)})}, \quad \nu = 0, 1, 2, \ldots, \tag{8.44}$$

where $'$ denotes the derivative with respect to s. The function $\phi(s)$ from (8.43) is already known; we obtain the function $\phi'(s)$ by first defining the functions

$$\xi(x) = \frac{\partial u(x;s)}{\partial s}, \qquad \eta(x) = \frac{\partial v(x;s)}{\partial s},$$

where $u = y$ and $v = y'$ are the solutions of (8.42). We differentiate the system (8.42) with respect to s and get an additional system of first-order equations

$$\begin{pmatrix} \xi \\ \eta \end{pmatrix}' = \begin{pmatrix} \eta \\ p(x;s)\eta + q(x;s)\xi \end{pmatrix}, \quad \begin{matrix} \xi(a) = a_1 \\ \eta(a) = a_0 \end{matrix}, \tag{8.45}$$

where

$$p(x;s) = \frac{\partial f(x, u(x;s), v(x;s))}{\partial v}, \qquad q(x;s) = \frac{\partial f(x, u(x;s), v(x;s))}{\partial u}.$$

We use the solutions of (8.45) at $x = b$ to construct

$$\phi'(s) = b_0\xi(b;s) + b_1\eta(b;s).$$

The initial-value problems (8.45) and (8.42) represent a system of four first-order differential equations which we solve for each s simultaneously on the same mesh: at the current s we use the solutions $u(b;s)$ and $v(b;s)$, as well as the corresponding value $\phi(s)$ at the extreme point of the interval, to compute the new value of $\xi(b;s)$ and $\eta(b;s)$, as well as the corresponding values $\phi'(s)$, and finally use $\phi(s)$ and $\phi'(s)$ in the iteration (8.44) until convergence is achieved. The additional time needed to solve (8.45) is compensated by the rapid convergence of the Newton method.

8.3.4 Systems of Non-linear Equations

Boundary-value problems for systems of non-linear differential equations are solved by first transforming them to an equivalent system of M (in general non-linear) first-order equations

$$\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y}), \quad a < x < b, \tag{8.46}$$

where $\boldsymbol{y} \in \mathbb{R}^M$, $\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^M \to \mathbb{R}^M$. We discuss boundary conditions of the form

$$A\boldsymbol{y}(a) + B\boldsymbol{y}(b) = \boldsymbol{\alpha}. \tag{8.47}$$

Here A and B are constant $M \times M$ matrices and $\boldsymbol{\alpha}$ is a constant M-dimensional vector. To the boundary-value problem (8.46)–(8.47) we assign the initial-value problem

$$\boldsymbol{u}' = \boldsymbol{f}(x, \boldsymbol{u}), \quad \boldsymbol{u}(a) = \boldsymbol{s}. \tag{8.48}$$

If $\boldsymbol{f}$ is uniformly Lipschitz, and the Jacobi matrix $\partial \boldsymbol{f}/\partial \boldsymbol{u}$ exists, a unique solution $\boldsymbol{u}(x; \boldsymbol{s})$ of this initial-value problem exists that smoothly depends on the components of $\boldsymbol{s}$. We seek a vector of parameters $\boldsymbol{s}^\star$ for which the solution of the initial-value problem $\boldsymbol{u}(x; \boldsymbol{s})$ is also the solution of the boundary-value problem (8.46) and (8.47). This occurs precisely when for $\boldsymbol{s} = \boldsymbol{s}^\star$ we have

$$\boldsymbol{\phi}(\boldsymbol{s}) = A\boldsymbol{s} + B\boldsymbol{u}(b; \boldsymbol{s}) - \boldsymbol{\alpha} = \boldsymbol{0}. \tag{8.49}$$

The initial-value problem is best solved on the usual uniform mesh (8.2) by using some stable method for the solution of initial-value problems. At the given $\boldsymbol{s}$, each point x_j carries the numerical solution $\widetilde{\boldsymbol{u}}_j$. If a method with the error of order $\mathcal{O}(h^p)$ is used, then $\|\widetilde{\boldsymbol{u}}_j(\boldsymbol{s}) - \boldsymbol{u}(x_j; \boldsymbol{s})\| \le \mathcal{O}(h^p)$. The solution $\boldsymbol{s}^\star$ can be found by simple iteration or by using Newton's method.

Solution by Simple Iteration We are seeking the fixed point of the mapping

$$\boldsymbol{s} = \boldsymbol{G}(\boldsymbol{s}) = \boldsymbol{s} - (A + B)^{-1}\big[A\boldsymbol{s} + B\widetilde{\boldsymbol{u}}_N(\boldsymbol{s}) - \boldsymbol{\alpha}\big],$$

where we have assumed that the matrix $Q = A + B$ is non-singular. (For more general boundary conditions (8.47) this is not necessarily true: then a different Q is needed to achieve convergence [1].) With an arbitrary initial approximation $\boldsymbol{s}^{(0)}$ we iterate

$$\boldsymbol{s}^{(\nu+1)} = \boldsymbol{G}\big(\boldsymbol{s}^{(\nu)}\big), \quad \nu = 0, 1, \ldots.$$

The convergence in the parameters $\boldsymbol{s}$ is mirrored in the convergence of the solutions at the individual points x_j ($j = 0, 1, \ldots, N$), for which

$$\big\|\widetilde{\boldsymbol{u}}_j\big(\boldsymbol{s}^{(\nu)}\big) - \boldsymbol{u}\big(x_j; \boldsymbol{s}^\star\big)\big\| \le \mathcal{O}\big(h^p\big) + \omega^\nu \big\|\boldsymbol{s}^\star - \boldsymbol{s}^{(0)}\big\|, \quad \nu = 0, 1, \ldots,$$

where $\boldsymbol{u}(x; \boldsymbol{s}^\star) \equiv \boldsymbol{y}(x)$ is the desired solution of the boundary-value problem (8.46) and (8.47). The parameter ω ($0 < \omega < 1$) satisfies the inequality

$$\int_a^b \mathcal{J}(x)\,\mathrm{d}x \le \log\left(1 + \frac{\omega}{P}\right),$$

where $P = \|(A + B)^{-1}B\|_\infty$ and $\mathcal{J}$ is a function that satisfies $\mathcal{J}(x) \ge \|\partial \boldsymbol{f}(x, \boldsymbol{u})/\partial \boldsymbol{u}\|_\infty$.

Solving by Newton's Method If the parameter ω in the simple iteration is close to one, the convergence is slow. A much faster approach is again offered by the Newton method, which we start with the initial approximation $\boldsymbol{s}^{(0)}$ and iterate

$$\boldsymbol{s}^{(\nu+1)} = \boldsymbol{s}^{(\nu)} + \Delta\boldsymbol{s}^{(\nu)}, \quad \nu = 0, 1, 2, \ldots, \tag{8.50}$$

where $\Delta\boldsymbol{s}^{(\nu)}$ is the solution of the system of linear equations

$$\frac{\partial\boldsymbol{\phi}(\boldsymbol{s}^{(\nu)})}{\partial\boldsymbol{s}}\Delta\boldsymbol{s}^{(\nu)} = -\boldsymbol{\phi}\big(\boldsymbol{s}^{(\nu)}\big). \tag{8.51}$$

We decompose the Jacobi matrix $\partial\boldsymbol{\phi}(\boldsymbol{s})/\partial\boldsymbol{s}$, which is non-singular for all $\boldsymbol{s}$ [4], as

$$\frac{\partial\boldsymbol{\phi}(\boldsymbol{s})}{\partial\boldsymbol{s}} = A + BW(b;\boldsymbol{s}), \tag{8.52}$$

where the matrix $W(x;\boldsymbol{s}) = \partial\boldsymbol{u}(x;\boldsymbol{s})/\partial\boldsymbol{s}$ is the solution of the system

$$W' = \frac{\partial\boldsymbol{f}(x,\boldsymbol{u}(x;\boldsymbol{s}))}{\partial\boldsymbol{u}}W, \quad W(0) = I. \tag{8.53}$$

For Newton's method we repeat the following procedure (details are in [1]): in the νth iteration step (i.e. *at current values* $\boldsymbol{s}^{(\nu)}$) on the mesh (8.2) we numerically solve the non-linear initial-value problem (8.48) with $\boldsymbol{s} = \boldsymbol{s}^{(\nu)}$, yielding the M components of the solution $\widetilde{\boldsymbol{u}}(\boldsymbol{s}^{(\nu)})$. Simultaneously, and on the same mesh, we solve M systems of linear differential equations (8.53), where we use the currently available $\widetilde{\boldsymbol{u}}(\boldsymbol{s}^{(\nu)})$ to compute the matrix $\partial\boldsymbol{f}(x,\boldsymbol{u})/\partial\boldsymbol{u}$. This gives us the approximate solution vector $\widetilde{\boldsymbol{u}}(\boldsymbol{s}^{(\nu)})$ and the corresponding $M \times M$ matrix $W(x;\boldsymbol{s}^{(\nu)})$. We use them to compute the new vector $\boldsymbol{\phi}(\boldsymbol{s})$ according to (8.49) and the new matrix $\partial\boldsymbol{\phi}(\boldsymbol{s})/\partial\boldsymbol{s}$ according to (8.52). These are used to solve the matrix system for $\Delta\boldsymbol{s}^{(\nu)}$ (see (8.51)) and, finally, we use (8.50) to compute the next approximate vector of the shooting parameters $\boldsymbol{s}^{(\nu+1)}$. We repeat the whole procedure until convergence to the final values $\boldsymbol{s}^{\star}$ is achieved.

8.3.5 Multiple (Parallel) Shooting

The basic version of scalar shooting from the beginning of Sect. 8.3 may be plagued by round-off errors. Such errors are generated, for example, when the combination (8.34) is formed and the contributions $\widetilde{y}_j^{(1)}$ and $s\widetilde{y}_j^{(2)}$ are almost equal in magnitude and oppositely signed, or when the denominator of (8.36) is very small. Such errors, due to which the solution of the corresponding initial-value problem blows up, are difficult to localize on an arbitrary mesh (8.2) and are almost unpreventable. We may face the same problems with systems of equations. Sometimes the most naive solution helps: increase arithmetic precision. If this does not work, one may opt for multiple shooting.

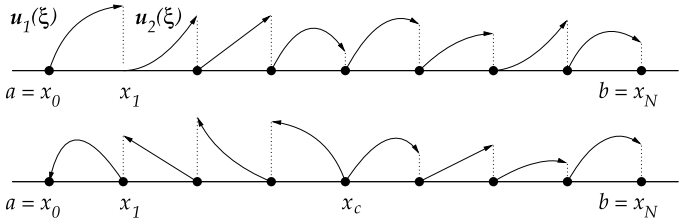


Fig. 8.5 Multiple (parallel) shooting. [*Top*] Shooting from $x = a$ towards $x = b$. We use the solutions $\boldsymbol{u}_j(\xi)$ from all subintervals $[x_{j-1}, x_j]$ to assemble the final solution on the whole interval $[a, b]$. [*Bottom*] Shooting from a critical point x_c (with a singularity, a known intermediate value, or another prescribed intermediate "boundary" condition) towards $x = a$ and $x = b$. We do not discuss such cases here; for further reading see [2, 3]

Here we describe multiple shooting for systems of non-linear equations (8.46) with boundary conditions (8.47), as approached by [2, 3]. We rescale the problem with the variable $x \in [a, b]$ on the mesh (8.2) by introducing a new variable ξ that takes the values $\xi \in [0, 1]$ on each subinterval $[x_{j-1}, x_j]$,

$$\xi = \frac{x - x_j}{x_j - x_{j-1}} = \frac{x - x_j}{h}$$

(see Fig. 8.5). On these subintervals one solves decoupled initial-value problems, where initial conditions are imposed *for each subinterval* such that finally the main boundary conditions at $x = a$ and $x = b$ are fulfilled. (If in the interior of $[a, b]$ further requirements need to be met, a non-uniform mesh may be utilized, so that certain conditions can be realized precisely at the required critical points. For details see [2, 3].) We also transform the vectors $\boldsymbol{y}$ and $\boldsymbol{f}$,

$$\boldsymbol{y}_j(\xi) = \boldsymbol{y}(x_{j-1} + \xi h),$$
$$\boldsymbol{f}_j(\xi, \boldsymbol{z}) = h\boldsymbol{f}(x_{j-1} + \xi h, \boldsymbol{z}).$$

Thus, *on each* $(j th)$ *subinterval* we are solving the system

$$\frac{\mathrm{d}\boldsymbol{y}_j}{\mathrm{d}\xi} = \boldsymbol{f}_j\big(\xi, \boldsymbol{y}_j(\xi)\big), \quad 0 < \xi < 1, \ j = 1, 2, \ldots, N,$$

while the boundary conditions become

$$A\boldsymbol{y}_1(0) + B\boldsymbol{y}_N(1) = \boldsymbol{\alpha}.$$

Moreover, we need to glue together continuously the solutions at each internal point of the mesh: we do this by requiring $\boldsymbol{y}_j(0) - \boldsymbol{y}_{j-1}(1) = \boldsymbol{0}$ (be warned again: the argument of these vector functions is the new variable ξ). The system of equations, the boundary conditions, and the local continuity conditions are condensed in the

equations

$$\frac{\mathrm{d}\boldsymbol{Y}}{\mathrm{d}\xi} = \boldsymbol{F}(\xi, \boldsymbol{Y}), \qquad \widetilde{A}\boldsymbol{Y}(0) + \widetilde{B}\boldsymbol{Y}(1) = \widetilde{\alpha}, \tag{8.54}$$

where the vector

$$\boldsymbol{Y}(\xi) = \left(\boldsymbol{y}_1^{\mathrm{T}}(\xi), \boldsymbol{y}_2^{\mathrm{T}}(\xi), \ldots, \boldsymbol{y}_N^{\mathrm{T}}(\xi)\right)^{\mathrm{T}} \in \mathbb{R}^{MN}$$

contains the complete solution, and the N vector functions $\boldsymbol{f}_i : [0, 1] \times \mathbb{R}^M \to \mathbb{R}^M$ can be understood as a vector function $\boldsymbol{F} : [0, 1] \times \mathbb{R}^{MN} \to \mathbb{R}^{MN}$,

$$\boldsymbol{F}(\xi, \boldsymbol{Y}) = \left(\boldsymbol{f}_1^{\mathrm{T}}(\xi, \boldsymbol{y}_1), \boldsymbol{f}_2^{\mathrm{T}}(\xi, \boldsymbol{y}_2), \ldots, \boldsymbol{f}_N^{\mathrm{T}}(\xi, \boldsymbol{y}_N)\right)^{\mathrm{T}}.$$

The vector $\widehat{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}^{\mathrm{T}}, \boldsymbol{0}^{\mathrm{T}}, \ldots, \boldsymbol{0}^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{MN}$ contains the boundary and continuity conditions. The $MN \times MN$ matrices $\widetilde{A}$ and $\widetilde{B}$ have the block structure

$$\widetilde{A} = \begin{pmatrix} A & & & & \\ & I & & & \\ & & I & & \\ & & & \ddots & \\ & & & & I \end{pmatrix}, \qquad \widetilde{B} = \begin{pmatrix} 0 & & & & B \\ -I & 0 & & & \\ & -I & 0 & & \\ & & \ddots & \ddots & \\ & & & -I & 0 \end{pmatrix}.$$

The matrices A and B from the underlying boundary-value problem, the identity I and the zero matrix 0 have size $M \times M$.

The system (8.54) corresponds to the vector initial-value problem consisting of MN first-order equations and the initial condition:

$$\frac{\mathrm{d}\boldsymbol{U}}{\mathrm{d}\xi} = \boldsymbol{F}(\xi, \boldsymbol{U}), \quad 0 < \xi \le 1, \ \boldsymbol{U}(0) = \boldsymbol{s}, \tag{8.55}$$

where the solution vector $\boldsymbol{U}$ has the same structure as the vector $\boldsymbol{Y}$. We solve this problem on the interval $\xi \in [0, 1]$ by simple shooting. With respect to individual subintervals $[x_{j-1}, x_j]$, the system of equations (8.55) is decoupled and completely equivalent to N vector systems of dimension M,

$$\frac{\mathrm{d}\boldsymbol{u}_j(\xi)}{\mathrm{d}\xi} = \boldsymbol{f}_j(\xi, \boldsymbol{u}_j), \quad \boldsymbol{u}_j(0) = \boldsymbol{s}_j, \ j = 1, 2, \ldots, N,$$

which can be solved independently (the vectors $\boldsymbol{s}_j$ of length M are the elements of the vector $\boldsymbol{s} = (\boldsymbol{s}_1^{\mathrm{T}}, \boldsymbol{s}_2^{\mathrm{T}}, \ldots, \boldsymbol{s}_N^{\mathrm{T}})^{\mathrm{T}}$). As it has been explained in the previous two subsections, we must solve, in addition to the system (8.55), the matrix initial-value problem

$$\frac{\mathrm{d}W}{\mathrm{d}\xi} = \frac{\partial \boldsymbol{F}(\xi, \boldsymbol{U}(\xi; \boldsymbol{s}))}{\partial \boldsymbol{Y}} W, \quad W(0) = I. \tag{8.56}$$

The matrix W is block-diagonal, $W = \mathrm{diag}\{W_1(\xi, \boldsymbol{s}_1), W_2(\xi, \boldsymbol{s}_2), \ldots, W_N(\xi, \boldsymbol{s}_N)\}$, so this system is decoupled as well, and can be written in the form

$$\frac{\mathrm{d}W_j}{\mathrm{d}\xi} = \frac{\partial \boldsymbol{f}_j(\xi, \boldsymbol{u}_j(\xi; \boldsymbol{s}_j))}{\partial \boldsymbol{u}_j} W_j, \quad W_j(0) = I, \ j = 1, 2, \ldots, N.$$

Its solution can be again found by Newton's iteration with an appropriate initial approximation of the shooting parameters $\boldsymbol{s}^{(0)}$. With the current approximation $\boldsymbol{s} = \boldsymbol{s}^{(\nu)}$ we solve the vector system (8.55), and use its solution to compute the matrix $\partial \boldsymbol{F}/\partial \boldsymbol{Y}$ at the right-hand side of (8.56). We use the resulting vector $\boldsymbol{U}(1; \boldsymbol{s}^{(\nu)})$ and matrix $W(1; \boldsymbol{s}^{(\nu)})$ (at the maximum value $\xi = 1$) to compute

$$\boldsymbol{\phi}\big(\boldsymbol{s}^{(\nu)}\big) = \widetilde{A}\boldsymbol{s}^{(\nu)} + \widetilde{B}\boldsymbol{U}\big(1; \boldsymbol{s}^{(\nu)}\big) - \widetilde{\boldsymbol{\alpha}}$$

and solve the matrix system for the update to the vector of shooting parameters,

$$\big[\widetilde{A} + \widetilde{B}W\big(1; \boldsymbol{s}^{(\nu)}\big)\big]\Delta \boldsymbol{s}^{(\nu)} = -\boldsymbol{\phi}\big(\boldsymbol{s}^{(\nu)}\big).$$

Finally, we compute the new shooting parameters,

$$\boldsymbol{s}^{(\nu+1)} = \boldsymbol{s}^{(\nu)} + \Delta \boldsymbol{s}^{(\nu)},$$

and repeat the procedure until it converges.

Advantage of Multiple Shooting We had to wait until the end to learn why the strenuous solving of the system (8.48) on narrower subintervals with $\xi \in [0, 1]$ is preferable to solving the system (8.55) on the original interval with $x \in [a, b]$. If the systems are solved by using a stable method with the error of order $\mathcal{O}(h^p)$, the errors at the points on the subintervals can be estimated by

$$\big\|\boldsymbol{u}(x_j) - \boldsymbol{u}_j\big\| \le h^p M_1 \exp\big(\Lambda_1 |x_0 - x_j|\big),$$
$$\big\|\boldsymbol{U}(\xi_j) - \boldsymbol{U}_j\big\| \le h^p M_2 \exp\big(\Lambda_2 |\xi_0 - \xi_j|\big),$$

where the constants $M_{1,2}$ and $\Lambda_{1,2}$ can be bounded by the values of the functions $\boldsymbol{f}$ and $\boldsymbol{F}$, and their derivatives [5, 6]. Because the original mesh (8.2) is split into N subintervals, we have $\Lambda_2 \approx \Lambda_1(b-a)/N$, so the upper bound for the error in the case of multiple shooting decreases exponentially or becomes proportional to $[\exp(\Lambda_1|b-a|)]^{1/N}$ instead of to $[\exp(\Lambda_1|b-a|)]$.

8.4 Asymptotic Discretization Schemes ⋆

In discussions of ordinary and partial differential equations we are often concerned about the limits in which some physical mechanism expressed by these equations dominates over other mechanisms. The limit (asymptotic) equation is an approximation of the complete physical picture, but it is usually much simpler than the full

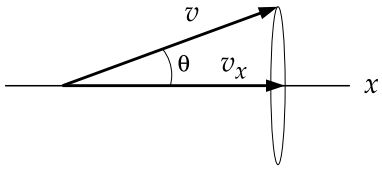


Fig. 8.6 Particle transport. All particles travel with equal magnitudes of velocity v in directions given by the cosines $\mu = \cos\theta = v_x/v$. Each parameter μ defines a cone of directions: even though the transport equation is one-dimensional, the density function in fact describes a three-dimensional region of an infinite layer

equation, and easier to use. We should be particularly alert in problems that allow one part of the domain to be treated asymptotically, while the rest requires a non-asymptotic approach. In such cases the continuous equation has to be discretized such that the solutions of the discrete equation converge to the true solution in both regimes when the mesh spacing is reduced.

Difference schemes with correct convergence properties in the asymptotic regime are called *asymptotic-preserving discretization schemes*. Inadequate discretization may yield an inefficient scheme, which actually leads to the correct asymptotic solution, but only by using mesh spacings that are much smaller than the spatial scale characteristic of the asymptotic solution.

Following closely [7, 8], we illustrate the idea of asymptotic discretization by the transport equation for particles in matter with absorption (absorption cross-section σ_a, absorptions per unit length) and scattering (scattering cross-section σ_s, scatterings per unit length). This equation originates in the stationary limit of the corresponding partial differential equation, but in its continuous and discrete form it is very instructive and possesses all the properties of the boundary-value problem, so we discuss it in this chapter. The equation has the form

$$\mu\frac{\partial(vN(x,\mu))}{\partial x} + (\sigma_\mathrm{a}+\sigma_\mathrm{s})vN(x,\mu) = \frac{\sigma_\mathrm{s}}{2}\int_{-1}^{1} vN\big(x,\mu'\big)\,\mathrm{d}\mu' + Q(x,\mu). \quad (8.57)$$

Its solution is the particle density function $N(x,\mu)$ depending on the coordinate (variable $x \in [0,1]$) and angle ($\mu = \cos\theta = v_x/v \in [-1,1]$), see Fig. 8.6. The number of particles on the interval $[x, x+\mathrm{d}x]$ traveling in the directions $[\mu, \mu+\mathrm{d}\mu]$ is $N(x,\mu)\,\mathrm{d}x\,\mathrm{d}\mu$. The particle absorption rate on these intervals is $\sigma_\mathrm{a} vN(x,\mu)\,\mathrm{d}x\,\mathrm{d}\mu$, while the scattering rate is $\sigma_\mathrm{s} vN(x,\mu)\,\mathrm{d}x\,\mathrm{d}\mu$. The average distance between particle interactions (the mean free path) is $\lambda = 1/\sigma_\mathrm{t}$, where $\sigma_\mathrm{t} = \sigma_\mathrm{a} + \sigma_\mathrm{s}$ is the total cross-section. The generation of new particles is described by the term $Q(x,\mu)$: the number of particles per unit time created at $[x, x+\mathrm{d}x]$ with initial directions $[\mu, \mu+\mathrm{d}\mu]$ is $Q(x,\mu)\,\mathrm{d}x\,\mathrm{d}\mu$. The solution of (8.57) is uniquely determined by the boundary conditions at $x = 0$ for $\mu > 0$ and at $x = 1$ for $\mu < 0$.

Equation (8.57) can be rewritten in a more compact form

$$\mu\frac{\partial\psi}{\partial x} + \sigma_\mathrm{t}\psi = (\sigma_\mathrm{t}-\sigma_\mathrm{a})\phi + Q,$$

where $\psi = vN$ is the *angular flux*, while its average over all directions,

$$\phi = \frac{1}{2}\int_{-1}^{1} \psi(x, \mu')\, \mathrm{d}\mu', \tag{8.58}$$

is the *scalar flux*. The key physics consideration follows now. We use a scaling parameter ε to extract the asymptotic behavior from the equation. Assume that we are studying a diffusion problem in a one-dimensional layer with thickness much larger than λ, in which scattering processes dominate (the absorption is weak). Suppose that the angular flux, the absorption and scattering cross-sections, and the source term Q are continuous and that they change insignificantly on distances comparable to the mean free path. This asymptotic diffusion limit is described by the equation

$$\mu\frac{\partial\psi}{\partial x} + \frac{\sigma_\mathrm{t}}{\varepsilon}\psi = \left(\frac{\sigma_\mathrm{t}}{\varepsilon} - \varepsilon\sigma_\mathrm{a}\right)\phi + \varepsilon Q \tag{8.59}$$

in the limit $\varepsilon \to 0$, since then $\varepsilon\sigma_\mathrm{a} \to 0$, $\sigma_\mathrm{t}/\varepsilon \to \infty$ (or $\lambda/\varepsilon \to 0$) and $\varepsilon Q \to 0$. We obtain the *continuous* version of the transport equation valid in the asymptotic regime by expanding the solution in terms of the scaling parameter:

$$\psi = \sum_{n=0}^{\infty} \varepsilon^n \psi^{(n)}.$$

When this is inserted in (8.59) and the coefficients of the same powers of ε on both sides of the equation are matched, we get, to first three orders:

$$\begin{aligned}
\psi^{(0)} &= \phi^{(0)}, \\
\psi^{(1)} &= \phi^{(1)} - \frac{\mu}{\sigma_\mathrm{t}}\frac{\partial\phi^{(0)}}{\partial x}, \\
\psi^{(2)} &= \phi^{(2)} - \frac{\sigma_\mathrm{a}}{\sigma_\mathrm{t}}\phi^{(0)} + \frac{\mu}{\sigma_\mathrm{t}}\frac{\partial}{\partial x}\left[\phi^{(1)} - \frac{\mu}{\sigma_\mathrm{t}}\frac{\partial\phi^{(0)}}{\partial x}\right] + \frac{Q}{\sigma_\mathrm{t}}.
\end{aligned} \tag{8.60}$$

By integrating (8.60) over the directions μ, we obtain the diffusion equation

$$-\frac{\partial}{\partial x}\left[\frac{1}{3\sigma_\mathrm{t}}\frac{\partial\phi^{(0)}}{\partial x}\right] + \sigma_\mathrm{a}\phi^{(0)} = Q. \tag{8.61}$$

The asymptotic behavior of the transport equation with the assumptions specified above is therefore captured adequately by a simpler equation of the diffusion type, with a characteristic length $L = 1/\sqrt{3\sigma_\mathrm{t}\sigma_\mathrm{a}}$ that does not depend on ε.

8.4.1 Discretization

In the asymptotic limit, the solution of the full transport equation satisfies the diffusion equation. Does the analogous conclusion follow in the discrete case? The basic

message of this section is the warning that the discrete solution of the transport equation satisfies the discrete diffusion equation only in a discretization that preserves the physical content of such an asymptotic limit. In the following we show one asymptotic and one non-asymptotic discretization of the transport equation on a uniform mesh with cells (subintervals) $[x_{j-1/2}, x_{j+1/2}]$ of length $h = x_{j+1/2} - x_{j-1/2}$.

We discretize (8.59) in coordinates (x_j, $j = 1, 2, \ldots, N$, such that $x_{1/2} = 0$ and $x_{N+1/2} = 1$) and angles (μ_m, $m = 1, 2, \ldots, M$). For clarity we assume constant σ_t, σ_a, and Q. We obtain

$$\mu_m(\psi_{m,j+\frac{1}{2}} - \psi_{m,j-\frac{1}{2}}) + \frac{h\sigma_\mathrm{t}}{\varepsilon}\psi_{m,j} = h\left(\frac{\sigma_\mathrm{t}}{\varepsilon} - \varepsilon\sigma_\mathrm{a}\right)\phi_j + h\varepsilon Q.$$

The discrete solutions $\psi_{m,j\pm1/2}$ (computed at the cell edges) and $\phi_{m,j}$ (averaged over one cell) are functions of two indices. In a computer implementation we arrange the solution in a $M \times (N+1)$-dimensional array:

$$\psi = (\psi_{1,1/2}, \psi_{1,3/2}, \ldots, \psi_{1,N+1/2}, \ldots, \psi_{M,1/2}, \psi_{M,3/2}, \ldots, \psi_{M,N+1/2})^\mathrm{T}.$$

Let us assume Dirichlet boundary conditions. They are expressed as

$$\psi_{m,1/2} = f_m \quad (\mu_m > 0), \qquad \psi_{m,N+1/2} = g_m \quad (\mu_m < 0).$$

"Upwind" Discretization The discrete angular flux has $N+1$ spatial components. The discrete transport equation connects N pairs of cell boundaries. The one missing equation that relates the angular flux at the cell edge to the angular flux within the cell, will determine the asymptotic or non-asymptotic character of the whole scheme. We first attempt an "*upwind*" (U) discretization

$$\psi_{m,j} = \begin{cases} \psi_{m,j+1/2}; & \mu_m > 0, \\ \psi_{m,j-1/2}; & \mu_m < 0, \end{cases}$$

and apply the same asymptotic analysis in terms of powers of ε as in the continuous case. We find that the asymptotic solution satisfies the difference equation

$$\frac{1}{4h}\left[\phi_j^{(0)} - \phi_{j-1}^{(0)}\right] + \frac{1}{4h}\left[\phi_j^{(0)} - \phi_{j+1}^{(0)}\right] = 0, \tag{8.62}$$

which is the discrete form of a "bare" diffusion equation $\partial^2\phi^{(0)}/\partial x^2 = 0$ from which scattering and absorption cross-sections have vanished! Therefore, the upwind discretization asymptotically generates a difference scheme which does not reflect the corresponding limit in the continuous equation.

"Diamond" Discretization We arrive at a quite different asymptotic equation by the so-called *diamond* (D) discretization in the nomenclature of [7],

$$\psi_{m,j} = \frac{1}{2}(\psi_{m,j+1/2} + \psi_{m,j-1/2}).$$

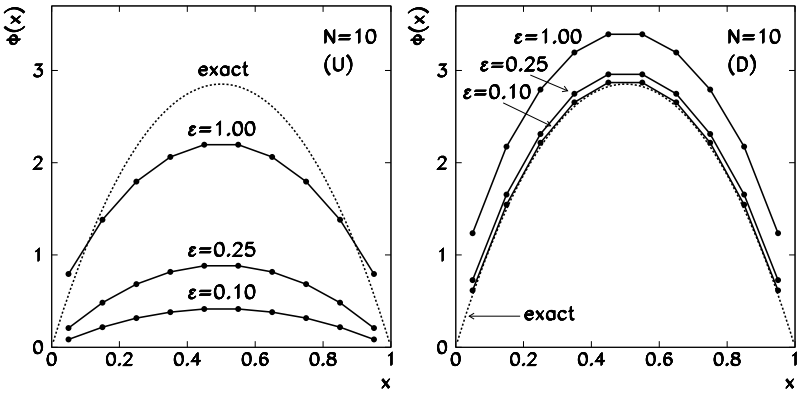


Fig. 8.7 The scalar flux from the solution of the discretized transport equation on a mesh of $N = 10$ cells for various scaling parameters ε. [*Left*] The upwind scheme has no particle sources Q present in (8.62), so the solution converges to zero when $\varepsilon \to 0$. [*Right*] The diamond scheme (8.63) has the correct asymptotic behavior

By applying the analysis in orders of the scaling parameter ε we realize that the diamond solution in the asymptotic diffusion limit satisfies the equation

$$-\frac{1}{3\sigma_t}\frac{1}{h^2}\left[\phi^{(0)}_{j+3/2} - 2\phi^{(0)}_{j+1/2} + \phi^{(0)}_{j-1/2}\right] + \frac{\sigma_a}{4}\left[\phi^{(0)}_{j+3/2} + 2\phi^{(0)}_{j+1/2} + \phi^{(0)}_{j-1/2}\right] = \frac{1}{2}(Q_{j+1} + Q_j), \tag{8.63}$$

which is a valid discretization of (8.61) and involves both cross-sections and the source term. The diamond discretization thus yields a difference scheme that preserves the character of the original equation in the asymptotic limit.

Example (Adapted from [10]) We check how the upwind and diamond schemes work in practice by computing the numerical solution on a mesh with $N = 10$ cells and parameters $Q = 1\ \mathrm{cm}^{-1}\mathrm{s}^{-1}$, $\sigma_t = 10\ \mathrm{cm}^{-1}$, $\sigma_a = 0.1\ \mathrm{cm}^{-1}$, and $\varepsilon = 0.1$. We need to solve a system of linear equations with a $M(N+1) \times M(N+1)$ matrix, so excessive M and N may quickly exhaust our memory resources! The scalar flux ϕ_j, which we compute from the angular fluxes $\psi_{m,j}$ by integrating over angles in analogy to (8.58), is therefore best evaluated by Gauss quadrature

$$\phi_j = \sum_{m=1}^{M} w_m \psi_{m,j},$$

which is exceptionally precise even with few points ($M = 4$, 8, or 16). Here w_m are the quadrature weights and μ_m the corresponding nodes (see Table 25.4 in [9] and Appendix E). Figure 8.7 (left) shows the computed scalar flux for various parameters ε by using the upwind scheme. In the asymptotic limit $\varepsilon \to 0$ the solution converges

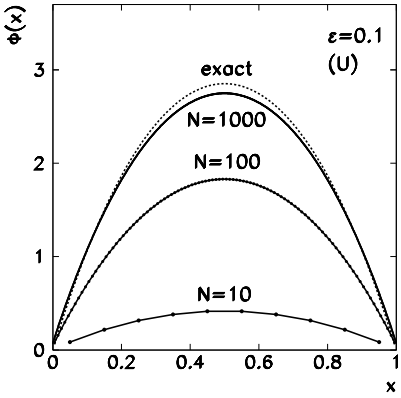


Fig. 8.8 Convergence of the scalar flux in the non-asymptotic difference scheme in upwind discretization with $N = 10$, 100, and 1000 cells, to the exact solution. The values of the parameters are $Q = 1\ \mathrm{cm^{-1}s^{-1}}$, $\sigma_\mathrm{t} = 10\ \mathrm{cm^{-1}}$, $\sigma_\mathrm{a} = 0.1\ \mathrm{cm^{-1}}$, and $\varepsilon = 0.1$. For a precise solution we need an extremely fine mesh: for $N = 1000$ the cell length is equal to $1/100$ of the mean free path

to zero, as (8.62) does not contain the source term Q. Figure 8.7 (right) shows the flux computed by using the diamond scheme. In the limit $\varepsilon \to 0$ the numerical solution converges to the exact solution.

Figure 8.8 tells us that the upwind scheme does in fact converge to the exact solution, but only with a very large number of cells (a very fine spatial mesh). For a precise solution one needs to employ a mesh in which the cell length is several orders of magnitude smaller than the mean free path [8].

8.5 Collocation Methods ⋆

In Sect. 8.1 we discussed the solution of scalar boundary-value problems by difference methods. We obtained the solution by approximating the derivatives in the differential equations and boundary conditions, and solving the resulting difference equations. In collocation methods we expand the (as yet unknown) solution y as

$$y(x) \approx u(x) = \sum\nolimits_n a_n \phi_n(x), \quad a \le x \le b,$$

and satisfy the requirements of the problem with appropriate coefficients a_n. The basis functions ϕ_n may be trigonometric functions, cubic B-splines, or polynomials. In the following we discuss the basic properties of collocation methods for three classes of scalar boundary-value problems: linear second-order problems by using B-spline collocation in Sect. 8.5.1, and linear and non-linear problems of higher orders by using Legendre polynomials in Sects. 8.5.2 and 8.5.3, trailing closely the presentation in [2, 3]. For greater clarity we use a uniform mesh, $x_{j+1} - x_j = h_j = h$, throughout this section.

8.5.1 Scalar Linear Second-Order Boundary-Value Problems

The basic idea of the collocation method for solving scalar boundary-value problems can be demonstrated by the linear problem of the form

$$y'' + p(x)y' + q(x)y = r(x), \quad 0 \le x \le 1,$$

with boundary conditions

$$y(0) = \alpha, \qquad y(1) = \beta.$$

The lowest degree of the polynomials for a continuous and continuously differentiable interpolation of the solution over all subintervals, is three. We can use piecewise continuous cubic polynomials known as cubic *B-splines*. The basis B-spline B_j is defined as

$$B_j(x) = \begin{cases} 0; & x \le x_{j-2}, \\ \frac{(x-x_{j-2})^3}{6\Delta x^3}; & x_{j-2} \le x \le x_{j-1}, \\ \frac{1}{6} + \frac{(x-x_{j-1})}{2\Delta x} + \frac{(x-x_{j-1})^2}{2\Delta x^2} - \frac{(x-x_{j-1})^3}{2\Delta x^3}; & x_{j-1} \le x \le x_j, \\ \frac{1}{6} - \frac{(x-x_{j+1})}{2\Delta x} + \frac{(x-x_{j+1})^2}{2\Delta x^2} + \frac{(x-x_{j+1})^3}{2\Delta x^3}; & x_j \le x \le x_{j+1}, \\ -\frac{(x-x_{j+2})^3}{6\Delta x^3}; & x_{j+1} \le x \le x_{j+2}, \\ 0; & x_{j+2} \le x. \end{cases}$$

The function B_j is composed by "gluing" together cubic functions between x_{j-2} and x_{j-1}, between x_{j-1} and x_j, between x_j and x_{j+1}, and between x_{j+1} and x_{j+2}. The support of the spline B_j is the interval $[x_{j-2}, x_{j+2}]$, on which B_j is twice continuously differentiable. The resulting spline has values $B_j(x_j) = 2/3$ and $B_j(x_{j\pm1}) = 1/6$ at the central points, the first derivatives are $B_j'(x_j) = 0$ and $B_j'(x_{j\pm1}) = \mp 1/(2h)$, and the second derivatives are $B_j''(x_j) = -2/h^2$ and $B_j''(x_{j\pm1}) = 1/h^2$. At the remaining points $x_{j\pm k}$, $k \ge 2$, the values $B_j(x_j)$ are zero. Figure 8.9 shows the spline B_5 and its nearest neighbors B_4 and B_6 on a uniform mesh on $x \in [0, 1]$.

We write the solution of the boundary-value problem as

$$y(x) \approx u(x) = \sum_{j=-1}^{N+1} a_j B_j(x). \tag{8.64}$$

The leftmost spline B_{-1} and the rightmost spline B_{N+1} are included in the sum because both have non-zero contributions on $[a, b]$: the former at $a = x_0$, the latter at $b = x_N$. The values of u and its first and second derivative at x_j are thus determined

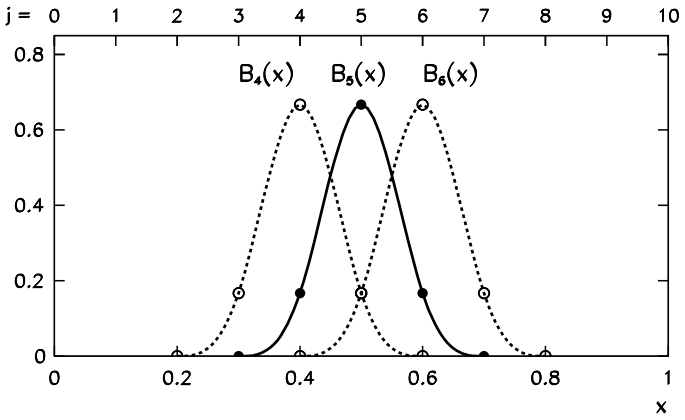


Fig. 8.9 The cubic spline B_5 and its nearest neighbors B_4 and B_6 on a uniform mesh $x \in [0, 1]$ with $(N + 1) = 11$ points (including the boundaries)

by the linear combinations of three adjacent coefficients a_j:

$$u(x_j) = \frac{1}{6}(a_{j+1} + 4a_j + a_{j-1}),$$
$$u'(x_j) = \frac{1}{2h}(a_{j+1} - a_{j-1}),$$
$$u''(x_j) = \frac{1}{h^2}(a_{j+1} - 2a_j + a_{j-1}).$$

The key requirement of collocation is imposed at this stage. The remainder

$$y''(x) + p(x)y' + q(x)y - r(x)$$

should be *zero at the collocation points*. Here collocation and mesh points are the same, but in the following subsections we also discuss "true" collocation where collocation points can be chosen differently. We denote the function values at x_j by $p(x_j) = p_j$, $q(x_j) = q_j$, and $r(x_j) = r_j$. When we account for boundary conditions, we obtain a system of equations for the coefficients a_j,

$$\begin{pmatrix} g_0 & h_0 & & & & \\ d_1 & g_1 & h_1 & & & \\ & d_2 & g_2 & h_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & d_{N-1} & g_{N-1} & h_{N-1} \\ & & & & d_N & g_N \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{N-1} \\ a_N \end{pmatrix} = \begin{pmatrix} R_0 \\ R_1 \\ R_2 \\ \vdots \\ R_{N-1} \\ R_N \end{pmatrix}. \tag{8.65}$$

The matrix elements and the right side of the equation are given by

$$g_j = \begin{cases} -6 + 2hp_0; & j = 0, \\ 4(-3 + h^2 q_j); & j = 1, 2, \ldots, N-1, \\ -6 - 2hp_N; & j = N, \end{cases}$$

$$d_j = \begin{cases} 6 - 3hp_j + h^2 q_j; & j = 1, 2, \ldots, N-1, \\ -hp_N; & j = N, \end{cases}$$

$$h_j = \begin{cases} hp_0; & j = 0, \\ 6 + 3hp_j + h^2 q_j; & j = 1, 2, \ldots, N-1, \end{cases}$$

$$R_j = \begin{cases} h^2 r_0 - \alpha(6 - 3hp_0 + h^2 q_0); & j = 0, \\ 6h^2 r_j; & j = 1, 2, \ldots, N-1, \\ h^2 r_N - \beta(6 + 3hp_N + h^2 q_N); & j = N. \end{cases}$$

We use this method (and compare it to the difference method) in Problem 8.9.5.

8.5.2 Scalar Linear Boundary-Value Problems of Higher Orders

We discuss linear scalar boundary-value problems of the form

$$Ly = y^{(M)} - \sum_{m=1}^{M} c_m(x) y^{(m-1)} = q(x), \quad a < x < b, \tag{8.66}$$

where M is the order of the highest derivative. As the basic mesh we take (8.2) but pay attention to the indices: the subintervals $[x_j, x_{j+1}]$ are labeled by j, while j and the additional subscript k define the *collocation points* within these subintervals,

$$x_{jk} = x_j + h\rho_k, \quad 0 \le j \le N-1, \ 1 \le k \le K. \tag{8.67}$$

The points on the subintervals are uniquely defined by the *canonical parameters*

$$0 \le \rho_1 \le \rho_2 \le \cdots \le \rho_K \le 1$$

that depend on the method. We usually choose them such that at some order K we obtain a quadrature formula of maximum possible precision on the subinterval. In Gauss collocation, ρ_k are the zeros of Legendre polynomials, where $\rho_1 > 0$ and $\rho_K < 1$, so none of the collocation points coincide with the mesh points. In Radau collocation we have $\rho_1 > 0$ and $\rho_K = 1$, so the last collocation point on the subinterval always coincides with a mesh point (except at $x = a$). In Lobatto collocation we have $\rho_1 = 0$ and $\rho_K = 1$. Figure 8.10 shows the basic uniform mesh x_j with the collocation points of the Gauss scheme of orders two and three. The canonical points for collocations of orders from $K = 2$ to $K = 5$ are listed in Table 8.2.

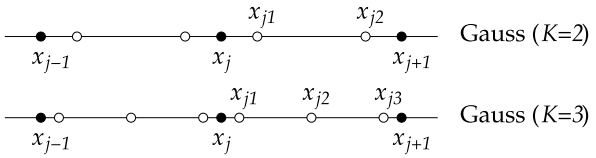


Fig. 8.10 Collocation points $x_{jk} = x_j + h\rho_k$ (*symbols* ∘) for Gauss collocation of order two with canonical points $\rho_{1,2} = 1/2 \mp \sqrt{3}/6$, and for Gauss collocation of order three with canonical points $\rho_{1,3} = 1/2 \mp \sqrt{15}/10$, and $\rho_2 = 1/2$ embedded in the basic uniform mesh x_j (*symbols* •). In other collocations (Radau or Lobatto) some collocation points coincide with the mesh points

Table 8.2 Canonical parameters ρ_k of Gauss collocation points from order $K = 2$ to order $K = 5$, which are zeros of the Legendre polynomials $P_K(2x - 1)$ rescaled to the interval [0, 1]. The collocation points are given by (8.67)

ρ_k	$K = 2$	$K = 3$	$K = 4$	$K = 5$
ρ_1	$\frac{1}{2} - \frac{\sqrt{3}}{6}$	$\frac{1}{2} - \frac{\sqrt{15}}{10}$	$\frac{1}{2} - \frac{\sqrt{35(15+2\sqrt{30})}}{70}$	$\frac{1}{2} - \frac{\sqrt{7(35+2\sqrt{70})}}{42}$
ρ_2	$\frac{1}{2} + \frac{\sqrt{3}}{6}$	$\frac{1}{2}$	$\frac{1}{2} - \frac{\sqrt{35(15-2\sqrt{30})}}{70}$	$\frac{1}{2} - \frac{\sqrt{7(35-2\sqrt{70})}}{42}$
ρ_3		$\frac{1}{2} + \frac{\sqrt{15}}{10}$	$\frac{1}{2} + \frac{\sqrt{35(15-2\sqrt{30})}}{70}$	$\frac{1}{2}$
ρ_4			$\frac{1}{2} + \frac{\sqrt{35(15+2\sqrt{30})}}{70}$	$\frac{1}{2} + \frac{\sqrt{7(35-2\sqrt{70})}}{42}$
ρ_5				$\frac{1}{2} + \frac{\sqrt{7(35+2\sqrt{70})}}{42}$

In the collocation method we require the numerical solution u to satisfy the differential equation *at the collocation points*, so that *for each subinterval* indexed by $0 \le j \le N - 1$ and *for all collocation points* defined by the canonical parameters ρ_k $(1 \le k \le K)$, we have

$$u^{(M)}(x_{jk}) - \sum_{m=1}^{M} c_m(x_{jk}) u^{(m-1)}(x_{jk}) - q(x_{jk}) = 0. \tag{8.68}$$

The simple method with cubic splines from the preceding subsection corresponds to a kind of "degenerate" collocation, since the mesh and collocation points coincide. In the following we show the structure of "true" collocation [2, 3] that is valid regardless of the type of the chosen basis functions. Let the degree of the polynomials on individual subintervals be $K + M$ for some $K \ge M$. On the subinterval $[x_j, x_{j+1}]$ we Taylor-expand the polynomial u around x_j,

$$u(x) = \sum_{n=1}^{K+M} u^{(n-1)}(x_j) \frac{(x - x_j)^{n-1}}{(n-1)!}$$

$$= \sum_{m=1}^{M} u_{jm} \frac{(x-x_j)^{m-1}}{(m-1)!} + h^M \sum_{k=1}^{K} z_{jk} \psi_k\left(\frac{x-x_j}{h}\right), \tag{8.69}$$

where we have defined

$$z_{jk} = h^{k-1} u^{(M+k-1)}(x_j), \qquad \psi_k(t) = \frac{t^{M+k-1}}{(M+k-1)!}, \quad 0 \le t \le 1,$$

and where we have collected the components of the solution and its $(M-1)$ derivatives *at the jth mesh point* into the vector $\boldsymbol{u}_j$ with components u_{jm}:

$$\boldsymbol{u}_j = (u_{j1}, u_{j2}, \ldots, u_{jM})^{\mathrm{T}} = \left(u(x_j), u'(x_j), \ldots, u^{(M-1)}(x_j)\right)^{\mathrm{T}}. \tag{8.70}$$

When the collocation solution is used in (8.66), we obtain

$$Lu(x) = h^M \sum_{k=1}^{K} z_{jk} L\left[\psi_k\left(\frac{x-x_j}{h}\right)\right] - \sum_{m=1}^{M} c_m(x) \sum_{n=m}^{M} u_{jn} \frac{(x-x_j)^{n-m}}{(n-m)!}.$$

We further define $\boldsymbol{z}_j = (z_{j1}, z_{j2}, \ldots, z_{jK})^{\mathrm{T}}$ and $\boldsymbol{q}_j = (q(x_{j1}), q(x_{j2}), \ldots, q(x_{jK}))^{\mathrm{T}}$. The collocation requirement (8.68) results in a linear system

$$V_j \boldsymbol{u}_j + W_j \boldsymbol{z}_j = \boldsymbol{q}_j, \quad 0 \le j \le N-1, \tag{8.71}$$

where the elements of the $K \times M$ matrix V_j and of the $K \times K$ matrix W_j are

$$(V_j)_{km} = -\sum_{l=1}^{m} c_l(x_{jk}) \frac{(h\rho_k)^{m-l}}{(m-l)!}, \quad 1 \le k \le K,\ 1 \le m \le M,$$

$$(W_j)_{km} = \psi_m^{(M)}(\rho_k) - \sum_{l=1}^{M} c_l(x_{jk}) h^{M-l+1} \psi_m^{(l-1)}(\rho_k), \quad 1 \le k \le K,\ 1 \le m \le K.$$

Moreover, we require that the interpolation polynomial u and its $(M-1)$ derivatives are continuous at all internal boundary points of the subintervals. This introduces an additional system of equations

$$\boldsymbol{u}_{j+1} = C\boldsymbol{u}_j + D\boldsymbol{z}_j, \quad 0 \le j \le N-1, \tag{8.72}$$

with an upper-triangular $M \times M$ matrix C and a $M \times K$ matrix D, with the elements

$$C_{km} = \frac{h^{m-k}}{(m-k)!}, \quad k \le m,$$

$$D_{km} = h^{M-k+1} \psi_m^{(k-1)}(1), \quad 1 \le k \le M,\ 1 \le m \le K.$$

Finally, we consider the boundary conditions in the usual form $B_a \boldsymbol{u}_0 + B_b \boldsymbol{u}_N = \boldsymbol{\beta}$. The last step is the elimination of $\boldsymbol{z}_j$ from (8.71) and (8.72). We obtain

$$\boldsymbol{u}_{j+1} = \Gamma_j \boldsymbol{u}_j + \boldsymbol{r}_j, \quad 0 \le j \le N-1,$$

where $\Gamma_j = C - DW_j^{-1}V_j$ and $\boldsymbol{r}_j = DW_j^{-1}\boldsymbol{q}_j$. In summary, the problem (8.66) has boiled down to solving the matrix system of the form

$$\begin{pmatrix} -\Gamma_0 & I & & & & & \\ & -\Gamma_1 & I & & & & \\ & & -\Gamma_2 & I & & & \\ & & & \ddots & \ddots & & \\ & & & & -\Gamma_{N-1} & I & \\ B_a & & & & & & B_b \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_{N-1} \\ \boldsymbol{u}_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_{N-1} \\ \boldsymbol{\beta} \end{pmatrix}. \tag{8.73}$$

Note that each vector $\boldsymbol{u}_j$ has dimension M and contains the solution $u_j = u(x_j)$ at the mesh point x_j as well as its $(M-1)$ derivatives at this point (see definition (8.70)). The size of the matrix in (8.73) is $M(N+1) \times M(N+1)$. In the practical implementation we can compute in advance the quantities that do not depend on the choice of the mesh, like $\psi_m^{(l)}(\rho_k)$ and $\psi_m^{(l)}(1)$. We can also compute the vector of local variables $\boldsymbol{z}_j = W_j^{-1}(\boldsymbol{q}_j - V_j\boldsymbol{u}_j)$ that was eliminated in the derivation, and when we insert the values z_{jk} into (8.69), we obtain the complete collocation solution (not just at the mesh points). The method described here is illustrated by a linear fourth-order problem in the following Example (adapted from [2, 3]), and in Problem 8.9.4.

Example We use the collocation method on the interval $[a, b] = [0, 1]$ to solve the linear second-order boundary-value problem

$$y'' = -\frac{1}{x}y' + \left(\frac{8}{8-x^2}\right)^2, \quad 0 < x < 1, \tag{8.74}$$

with the boundary conditions $y'(0) = y(1) = 0$, and the analytic solution

$$y(x) = 2\log\left(\frac{7}{8-x^2}\right).$$

We rewrite the equation as

$$Ly = y'' + \frac{1}{x}y' = \left(\frac{8}{8-x^2}\right)^2,$$

whence, by comparing it to (8.66) we infer $M = 2$ (a second-order problem) and

$$c_1(x) = 0, \qquad c_2(x) = -\frac{1}{x}, \qquad q(x) = \left(\frac{8}{8-x^2}\right)^2.$$

We solve the problem on the uniform mesh $a = x_0, x_1, \ldots, x_N = b$ and opt for Gauss collocation with four canonical points (column $K = 4$ of Table 8.2) on each subinterval $[x_j, x_{j+1}]$. The collocation points are $x_{jk} = x_j + h\rho_k$, where $0 \le j \le$

$N-1$ and $1 \le k \le K$. We put the solution at each point x_j in the vector

$$\boldsymbol{u}_j = \big(u(x_j), u'(x_j)\big)^{\mathrm{T}}. \tag{8.75}$$

In computing the matrices V_j from (8.71) we use $c_1(x_{jk}) = 0$ and $c_2(x_{jk}) = -1/x_{jk}$, while to compute the matrices W_j we also need the derivatives of $\psi_k(t)$,

$$\psi_k(\rho_k) = \frac{\rho_k^{M+k-1}}{(M+k-1)!} = \frac{\rho_k^{k+1}}{(k+1)!}, \qquad \psi_k'(\rho_k) = \frac{\rho_k^{k}}{k!}, \qquad \psi_k''(\rho_k) = \frac{\rho_k^{k-1}}{(k-1)!}.$$

The constant matrices C and D appearing in (8.72) are

$$C = \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix}, \qquad D = \begin{pmatrix} h^2\psi_1(1) & h^2\psi_2(1) & h^2\psi_3(1) & h^2\psi_4(1) \\ h\psi_1'(1) & h\psi_2'(1) & h\psi_3'(1) & h\psi_4'(1) \end{pmatrix}.$$

We write the boundary conditions $y'(0) = y(1) = 0$ in the form $B_a\boldsymbol{u}_0 + B_b\boldsymbol{u}_N = \boldsymbol{\beta}$, whence we read off

$$B_a = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \qquad B_b = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \qquad \boldsymbol{\beta} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

From the right-hand side of the differential equation, described by the function q, we compute (at each collocation point) the vector

$$\boldsymbol{q}_j = \big(q(x_{j1}), q(x_{j2}), q(x_{j3}), q(x_{j4})\big)^{\mathrm{T}}.$$

Finally, we compute the matrices $\Gamma_j = C - DW_j^{-1}V_j$ and vectors $\boldsymbol{r}_j = DW_j^{-1}\boldsymbol{q}_j$, and solve the system (8.73) which gives us the values of the solution and its derivative (see (8.75)) at each mesh point x_j. Figure 8.11 (left) shows the absolute error of the numerical solution, and Fig. 8.11 (right) the error of the derivative for $N = 8$ (nine points $x_0, x_1, \ldots, x_8$ or eight subintervals).

8.5.3 *Scalar Non-linear Boundary-Value Problems of Higher Orders*

In the preceding subsection we learned how to deal with linear scalar problems of higher orders. Now let us discuss *non-linear* problems of the form

$$Ny = y^{(M)} - f\big(x, y, y', \ldots, y^{(M-1)}\big) = y^{(M)} - f(x, \boldsymbol{y}) = 0, \quad a < x < b,$$

where the vector

$$\boldsymbol{y}(x) = \big(y(x), y'(x), y''(x), \ldots, y^{(M-1)}\big)^{\mathrm{T}}$$

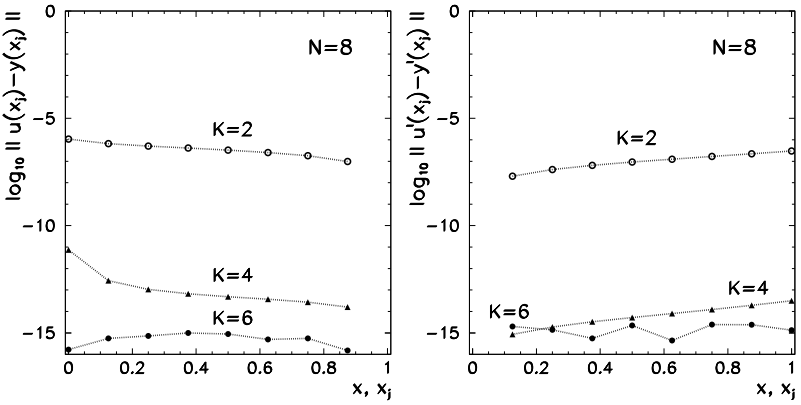


Fig. 8.11 Numerical errors in solving the problem (8.74) on a mesh with nine points ($N = 8$) by Gauss collocation of orders $K = 2$, $K = 4$, and $K = 6$ (two, four, and six canonical points on each subinterval). [*Left*] Error of the solution. Due to the boundary condition $y(1) = 0$ the logarithm of the error at x_N is undefined. [*Right*] Error of the derivative. Due to the condition $y'(0) = 0$ the logarithm of the error at x_0 is undefined

again contains the values of the solution and its $(M - 1)$ derivatives at x (or the corresponding mesh or collocation points). The function f is non-linear in the arguments $\boldsymbol{y}$, and the boundary condition is

$$\boldsymbol{g}\big(\boldsymbol{y}(a), \boldsymbol{y}(b)\big) = \boldsymbol{0}.$$

One possible way towards the solution of such problems leads through *linearization*. The basic idea is to linearize the non-linear problem and use some iterative method (like Newton's) to solve the corresponding linear boundary-value problems. If linearization is performed correctly, we may hope that the sequence of solutions of the linearized problem will converge to the solution of the non-linear one. The basic form of the method can be obtained from the expansion around the exact solution,

$$\begin{aligned} y^{(M)}(x) &= f\big(x, \boldsymbol{y}(x)\big) \\ &\approx f\big(x, \widetilde{\boldsymbol{y}}(x)\big) + \sum_{m=1}^{M} \frac{\partial f}{\partial y_m}\big(x, \widetilde{\boldsymbol{y}}(x)\big)\big(y^{(m-1)}(x) - \widetilde{y}^{(m-1)}(x)\big), \end{aligned}$$

where the ordering of the components of the vector $\widetilde{\boldsymbol{y}}$ is the same as in $\boldsymbol{y}$ (function, first derivative, second derivative, and so on). The difference between the current and the final numerical solution can be measured by the function

$$z(x) = y(x) - \widetilde{y}(x),$$

and, by analogy to the vectors $\boldsymbol{y}$ and $\widetilde{\boldsymbol{y}}$, a similar relation applies to the whole vector $\boldsymbol{z}$, e.g. $z^{(M)}(x) = y^{(M)}(x) - \widetilde{y}^{(M)}(x)$. The expansion results in a *linear* differen-

tial equation

$$z^{(M)} - \sum_{m=1}^{M} \underbrace{\frac{\partial f(x, \widetilde{\boldsymbol{y}}(x))}{\partial y_m}}_{c_m(x,\widetilde{\boldsymbol{y}})} z^{(m-1)}(x) = -\left[\widetilde{y}^{(M)} - f(x, \widetilde{\boldsymbol{y}})\right] \tag{8.76}$$

for the correction of the solution, z. We solve (8.76) iteratively by some initial approximation $\widetilde{\boldsymbol{y}}$. The expression on its right-hand side plays the role of the source term $q(x)$ (compare (8.76) and (8.66)) and we compute it in each iteration step by using the current solution $\widetilde{\boldsymbol{y}}$. The coefficients $c_m(x, \widetilde{\boldsymbol{y}})$ are also computed from the values of the current solution and its derivatives. We write the boundary condition $\boldsymbol{g}(\boldsymbol{y}(a), \boldsymbol{y}(b)) = \boldsymbol{0}$ in linearized form

$$B_a \boldsymbol{y}(a) + B_b \boldsymbol{y}(b) = \boldsymbol{0},$$

where

$$B_a = \frac{\partial \boldsymbol{g}(\boldsymbol{u}, \boldsymbol{v})}{\partial \boldsymbol{u}}, \qquad B_b = \frac{\partial \boldsymbol{g}(\boldsymbol{u}, \boldsymbol{v})}{\partial \boldsymbol{v}}, \quad \text{at } \boldsymbol{u} = \boldsymbol{y}(a),\ \boldsymbol{v} = \boldsymbol{y}(b).$$

In terms of the vector $\boldsymbol{z}$ this means

$$B_a z(a) + B_b z(b) = -\boldsymbol{g}\big(\widetilde{\boldsymbol{y}}(a), \widetilde{\boldsymbol{y}}(b)\big). \tag{8.77}$$

Solving non-linear scalar boundary-value problems of higher orders thus essentially translates to the use of Newton's method: the iteration is set off by some initial approximation for $\widetilde{\boldsymbol{y}}$, which is used to solve the linear differential equation (8.76) with the boundary condition (8.77) by collocation. Then we update

$$\widetilde{\boldsymbol{y}} \leftarrow \widetilde{\boldsymbol{y}} + z,$$

and repeat the procedure until $\|z\|$ drops below the desired precision. If the functions f and $\boldsymbol{g}$ have continuous second partial derivatives (plus a few additional mild assumptions), the method is quadratically convergent [2, 3].

8.5.4 Systems of Boundary-Value Problems

Collocation methods are also widely used for systems of non-linear boundary-value problems of higher orders, but their discussion is beyond the scope of this book. Among the best-known collocation tools are COLSYS [11, 12] and TWPBVP [13]. More recent is the (non-collocation) code MIRKDC [14], which is at the heart of the `bvp4c` routines built into the MATLAB environment. The latest development is the very fast BVP_SOLVER package [15] which is an enhanced version of the `bvp4c` package.

8.6 Weighted-Residual Methods ⋆

Weighted-residual methods are potent tools for the solution of boundary-value problems with ordinary and partial differential equations, and are at the core of numerous versions of the finite element method explained in Sect. 10.6. Their basic characteristic is the transformation of a boundary-value problem to a variational one. We restrict the discussion to boundary-value problems of the form

$$Lv = -\frac{\mathrm{d}}{\mathrm{d}x}\left(p(x)\frac{\mathrm{d}v}{\mathrm{d}x}\right) + q(x)v = f(x) \tag{8.78}$$

on $a < x < b$ with boundary conditions $x(a) = x(b) = 0$, where $p(x) > 0$ and $q(x) \geq 0$. Assume that the function p is continuously differentiable on $[a, b]$, while q and f are continuous on $[a, b]$. We multiply the differential equation (8.78) by some function w in the scalar-product sense (A.2), and obtain $\langle w, Lv - f\rangle = 0$. The solution of this problem is also the solution of the boundary-value problem (8.78) for all functions w for which this scalar product exists. By using

$$\langle w, Lv - f\rangle = 0 \quad \forall w \in L^2(a, b) \tag{8.79}$$

we have thus turned the boundary-value problems in its variational form: we find the solution v of the boundary-value problem (8.78) precisely when (8.79) applies for *any* square-integrable function w. The name of the method comes from the manner in which the residual $Lv - f$ (measuring the deviation from the exact fulfillment of the differential equation) is weighted by the function w in the integral (8.79).

In a concrete method we can only find a numerical solution u which is an approximation of the true solution v. We write u as a linear combination of some simpler functions,

$$v(x) \approx u(x) = \sum_{j=1}^{J} c_j \phi_j(x), \tag{8.80}$$

and call it the *trial function*. The functions ϕ_j are the basis functions of a finite-dimensional space of trial functions T^J. We usually think of $\phi_j \in T^J \subset L^2(a, b)$. The *weight* (or *test*) function is also set up as the sum

$$w(x) \approx \widetilde{w}(x) = \sum_{j=1}^{J} d_j \psi_j(x), \tag{8.81}$$

where, in general, ψ_j are different from ϕ_j and span the space of weight functions, W^J. These functions are frequently taken from $\psi_j \in W^J \subset L^2(a, b)$, but not necessarily so: for example, the Dirac delta-"functions" may also act as weight functions.

The primary goal is to determine the unknown coefficients c_j with chosen ϕ_j and ψ_j such that u is a sufficiently good approximation of v. In their basic outline the weighted-residual methods are no different from, say, Fourier or collocation methods. What distinguishes all these methods from one another is the manner in

which the residual is defined and its magnitude quantified. In weighted-residual methods we attempt to fulfill the equation $\langle \widetilde{w}, Lv - f \rangle = 0$ for each function $\widetilde{w}$: the residual $Lv - f$ should be orthogonal to all $\widetilde{w}$ from the space of weight functions. This means

$$\sum_j d_j \langle \psi_j, Lv - f \rangle = 0.$$

Since this condition should be valid for *any* choice of the coefficients d_j, we must also have

$$\langle \psi_j, Lv - f \rangle = 0 \quad \forall j.$$

Weighted-residual methods are classified according to the choice of function spaces for the trial functions ϕ_j and weight functions ψ_j. In the Galerkin method we take ϕ_j and ψ_j from the same space and $\phi_j = \psi_j$. If we choose ψ_j to be delta-"functions" at the collocation points z_j, so that $\langle \psi_j, f \rangle = f(z_j)$, the residual satisfies the condition $\langle \psi_j, Lu_c - f \rangle = 0$, where u_c is the collocation solution (see Sect. 8.5). When ϕ_j and ψ_j are taken from closely related but *different* function spaces (for example, from two spaces of polynomial splines of different orders), we are referring to Petrov–Galerkin methods.

Galerkin Method In the following we discuss the Galerkin method which is the most common. We can achieve a more symmetric and numerically practicable form of the variational formulation if we integrate (8.79) by parts and thereby eliminate the terms involving second derivatives. (This turns out to be helpful in implementations of the two- and three-dimensional finite element method where it is difficult to construct continuously differentiable function approximations. This is why it makes sense to stick to very simple basis functions.) We get

$$\int_a^b w\big[-(pv')' + qv - f\big]\,\mathrm{d}x = -wpv'|_a^b + \int_a^b \big[w'pv' + wqv - wf\big]\,\mathrm{d}x.$$

If w fulfills the same boundary conditions as v, the integrated part is zero, and the Galerkin condition assumes a nice symmetric form:

$$A(w, v) - \langle w, f \rangle = 0,$$

where

$$A(w, v) = \int_a^b \big[w'pv' + wqv\big]\,\mathrm{d}x. \tag{8.82}$$

In boundary-value problems describing mechanical systems, the bilinear form $A(u, v)$ represents the internal or *strain energy*. The functions v and w must satisfy the condition [16]

$$\int_a^b \big[(z')^2 + z^2\big]\,\mathrm{d}x < \infty, \quad z(a) = z(b) = 0,\ z \in \{v, w\},$$

and we are solving the variational problem $A(w, v) = \langle w, f \rangle$ for $\forall w$. In the actual implementation we replace v by u, and w by $\widetilde{w}$. According to Galerkin, we span u and $\widetilde{w}$ in the same function space ($u, \widetilde{w} \in T^J$) with the basis functions ϕ_j. We therefore compute the approximation u by solving the variational problem

$$A(\widetilde{w}, u) = \langle \widetilde{w}, f \rangle \quad \forall \widetilde{w} \in T^J. \tag{8.83}$$

When we insert the expansions (8.80) and (8.81) for u and $\widetilde{w}$, we obtain a system of linear equations for the expansion coefficients c_k of the approximate solution,

$$\sum_{k=1}^{J} c_k A(\phi_j, \phi_k) = \langle \phi_j, f \rangle, \quad j = 1, 2, \ldots, J.$$

How difficult it is to compute the integrals (8.82) depends on the choice of the basis functions ϕ_j (and, of course, on the functions p and q in the differential equation). If the basis functions are polynomials, it is sensible to choose low degrees. The fruits of the discussion in this section are collected in Sect. 10.6 describing the finite element method in one and two dimensions. We will encounter weighted-residual methods again in numerical approaches to partial differential equations in Chap. 11.

8.7 Boundary-Value Problems with Eigenvalues

In a boundary-value problems with eigenvalues (in short, *eigenvalue problems*) we seek not only the function satisfying the differential equation and boundary conditions, but also the scalars appearing in the formulation of the problem. To a physicist, the most relevant are the one-dimensional Sturm–Liouville problems

$$-\frac{\mathrm{d}}{\mathrm{d}x}\left(p(x)\frac{\mathrm{d}y}{\mathrm{d}x}\right) + q(x)y = \lambda w(x)y, \quad a < x < b, \tag{8.84}$$

where we wish to compute the functions y and the scalars λ. The coefficient functions p, q, and w are known real functions. We assume that p and w are strictly positive on (a, b). Moreover, we assume that p, q, and w are defined on the closed interval $[a, b]$ and are at least piecewise continuous on it. Here we discuss only regular Sturm–Liouville problems, where a and b are finite, and the boundary conditions at both endpoints can be expressed as

$$\begin{aligned} a_1 y(a) - a_2 p(a) y'(a) &= 0, \\ b_1 y(b) - b_2 p(b) y'(b) &= 0. \end{aligned} \tag{8.85}$$

The value of the parameter λ for which we are able to find a non-trivial solution of (8.84) with boundary conditions (8.85) is called the *eigenvalue* and the corresponding solution y is the eigenfunction.

Transformation to the Liouville Normal Form By using the transformation

$$\xi(x) = \int_a^x \sqrt{\frac{w(s)}{p(s)}}\,\mathrm{d}s, \qquad Y(\xi) = y(x)\big[w(x)p(x)\big]^{1/4},$$

(8.84) can be turned into its *Liouville normal form*

$$-\frac{\mathrm{d}^2 Y}{\mathrm{d}\xi^2} + Q(\xi)Y = \lambda Y, \quad \xi \in [0, B], \qquad B = \int_a^b \sqrt{\frac{w(s)}{p(s)}}\,\mathrm{d}s, \tag{8.86}$$

where

$$Q(\xi) = \frac{q(x(\xi))}{w(x(\xi))} + \frac{1}{m}\frac{\mathrm{d}^2 m}{\mathrm{d}\xi^2}, \qquad m(\xi) = \big[w\big(x(\xi)\big)p\big(x(\xi)\big)\big]^{1/4}.$$

The boundary conditions (8.85) change accordingly. They become

$$\alpha_1 Y(0) - \alpha_2 Y'(0) = 0,$$
$$\beta_1 Y(B) - \beta_2 Y'(B) = 0.$$

The constants $\alpha_{1,2}$ and $\beta_{1,2}$ can be expressed by the constants $a_{1,2}$ and $b_{1,2}$ listed in [17]. In (8.86) we recognize the Schrödinger equation for the bound states of a quantum particle in the potential $Q(\xi)$. The parameter λ determines the value of the eigenenergy, and the solution y is the corresponding eigenfunction. Any regular Sturm–Liouville problem can be turned into an equation of this type by using the transformation described above.

Two Useful Theorems The eigenvalues of regular Sturm–Liouville problems are real, and the eigenfunctions belonging to the individual eigenvalues are orthogonal with respect to the scalar product (A.1). The eigenvalues λ_k are distinct (there are no pairs of linearly independent eigenfunctions with equal eigenvalues) and can be arranged in an increasing sequence $\lambda_0 < \lambda_1 < \lambda_2 < \cdots$. The corresponding eigenfunctions y_k have precisely k zeros on the open interval (a, b). The functions y_k form a complete orthogonal system on (a, b). The following theorems, given without proof, are useful in practical work for the book-keeping of solution zeros.

The *Sturm comparison theorem* relates the positions of solution zeros of *different* boundary-value problems. Assume that the function y_1 on the interval (a, b) is a non-trivial solution of the equation $(p_1(x)y')' + q_1(x)y = 0$, and y_2 a non-trivial solution of $(p_2(x)y')' + q_2(x)y = 0$, where $0 < p_2 \le p_1$ and $q_1 \le q_2$ for $x \in (a, b)$. Then between any two zeros of y_1 there is at least one zero of y_2, except when y_2 is just a constant multiple of y_1.

The consequence is the theorem on the *interlacing of zeros* which relates the positions of zeros of functions that solve the *same* differential equation. If y_1 and y_2 are linearly independent solutions of $(p(x)y')' + q(x)y = 0$ on (a, b), and $p > 0$ on (a, b), then between any two zeros of one function there is precisely one zero of the other function (see example in Fig. 8.12).

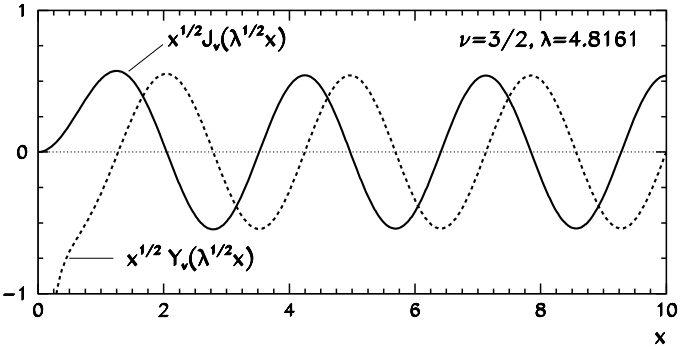


Fig. 8.12 Bessel's equation in Liouville form $-y''(x) + (\nu^2 - 1/4)x^{-2}y(x) = \lambda y(x)$ for $x \in (0, \infty)$ has two linearly independent solutions: $\sqrt{x}J_\nu(\sqrt{\lambda}x)$, regular everywhere, and $\sqrt{x}Y_\nu(\sqrt{\lambda}x)$, which has a singularity at the origin. The twofold character of the solutions is of key significance in the description of physical phenomena involving these functions. Shown are the solutions with $\nu = 1.5$ and the eigenvalue $\lambda = 4.8161$ that is determined by the boundary conditions. The zeros of one eigenfunction interlace with the zeros of the other. See also Fig. 1.13

The dependence of the eigenvalues on the endpoints, the boundary conditions, and the coefficients of the function w is discussed in [18]. The asymptotic expansions of the eigenvalues and eigenfunctions for $\lambda \to \infty$ are given in [17].

8.7.1 Difference Methods

The building blocks of the difference method for solving eigenvalue problems of the form (8.86) with Dirichlet boundary conditions $y(a) = 0$ and $y(b) = 0$ can be taken from the difference scheme (8.9) for problems (8.1) that do not involve eigenvalues. As usual we discretize all quantities on the uniform mesh (8.2). We denote the approximate value of the exact solution $y(x_j)$ by u_j, and the value $q(x_j)$ by q_j. By using the central difference for the second derivative we obtain a system of $(N-1)$ equations,

$$-\frac{u_{j-1} - 2u_j + u_{j+1}}{h^2} + q_j u_j = \Lambda u_j, \quad j = 1, 2, \ldots, N-1, \tag{8.87}$$

where Λ is an approximation of the exact eigenvalue λ. The boundary conditions translate to $u_0 = 0$ and $u_N = 0$. It is clear that at given N, solving this system will give us only $(N-1)$ approximate eigenvalues $\{\Lambda_k\}_{k=1}^{N-1}$, while the continuous problem has infinitely many. The difference scheme can then be packaged as

$$A\boldsymbol{u} = \Lambda \boldsymbol{u}, \tag{8.88}$$

where $\boldsymbol{u} = (u_1, u_2, \ldots, u_{N-1})^{\mathrm{T}}$ is the vector of solution components, except for the trivial ones at the endpoints, and the matrix is

$$A = \frac{1}{h^2} D + Q,$$

where

$$D = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}, \qquad Q = \begin{pmatrix} q_1 & & & & \\ & q_2 & & & \\ & & \ddots & & \\ & & & q_{N-2} & \\ & & & & q_{N-1} \end{pmatrix}.$$

The error of the method is $\mathcal{O}(k^4 h^2)$, more precisely,

$$|\lambda_k - \Lambda_k| \le C k^4 h^2, \quad k = 1, 2, \ldots, N-1,$$

where C depends only on the function q. The $\mathcal{O}(h^2)$ dependence should not come as a surprise once we recall the discretization error estimate (8.10). But in the context of eigenvalues, the $\mathcal{O}(k^4)$ dependence is much more relevant: we anticipate large errors of approximate eigenvalues with high indices k.

More general boundary conditions of the form (8.85) represent only a minor complication. Let us take just $p(a) = p(b) = 1$, and define $\alpha = a_2/a_1 \neq 0$ and $\beta = b_2/b_1 \neq 0$. The boundary conditions then access points beyond the mesh,

$$\frac{u_1 - u_{-1}}{2h} \approx \frac{u_0}{\alpha}, \qquad \frac{u_{N+1} - u_{N-1}}{2h} \approx \frac{u_N}{\beta}.$$

Equation (8.87) can also be considered at $x = a$ ($j = 0$) and $x = b$ ($j = N$), so u_{-1} and u_{N+1} can be eliminated. This results in a $(N+1) \times (N+1)$ matrix system of the form $A'\boldsymbol{u} = \Lambda M \boldsymbol{u}$, where M is a positive diagonal matrix.

Increasing the Order of Error By using a simple correction due to Numerov and Cowell, the discretization error can be improved by two orders: the scheme

$$\begin{aligned} &-\frac{u_{j-1} - 2u_j + u_{j+1}}{h^2} + (q_j - \Lambda) u_j \\ &\quad = -\frac{1}{12}\big[(q_{j-1} - \Lambda) u_{j-1} - 2(q_j - \Lambda) u_j + (q_{j+1} - \Lambda) u_{j+1}\big] \end{aligned}$$

with boundary conditions $u_0 = 0$ and $u_N = 0$ leads to the matrix equation

$$A\boldsymbol{u} = \Lambda M \boldsymbol{u}, \tag{8.89}$$

where

$$A = \frac{1}{h^2} D + MQ, \quad M = I - \frac{1}{12} D.$$

In general A, is not symmetric, but D and M commute, while Q is diagonal, so we can rewrite the generalized matrix eigenvalue problem (8.89) in the form $M^{-1}A\boldsymbol{u} = \Lambda\boldsymbol{u}$, where $M^{-1}A$ is symmetric (but sparse). The order of the Numerov–Cowell method is $\mathcal{O}(k^6h^4)$.

The trick (8.13) that we used to extrapolate the solutions to an ever finer mesh can be applied to boundary-problems with eigenvalues as well. We first compute the eigenvalues on a mesh with spacing h, then do it again on a mesh with spacing $h/2$. The improved estimate for the kth eigenvalue is then

$$\Lambda_k^{(h/2,h)} = \frac{4}{3}\Lambda_k^{(h/2)} - \frac{1}{3}\Lambda_k^{(h)}.$$

If the "standard" difference scheme of order $\mathcal{O}(k^4h^2)$ is used in the extrapolation, the order of the eigenvalue error becomes $\mathcal{O}(k^6h^4)$, and if the Numerov–Cowell scheme of order $\mathcal{O}(k^6h^4)$ is used, the resulting error of the eigenvalues is $\mathcal{O}(k^8h^6)$. We realize that an improved spatial discretization implies an even larger error in the determination of high-lying eigenvalues. Simple difference schemes without further improvements are therefore suitable only for the computation of low-lying eigenvalues (λ_k at small k).

A very efficient way to overcome this obstacle is the correction [19] that reduces the error of the standard difference scheme to $\mathcal{O}(k^0h^2)$ (the error becomes independent of k), and the error of the Numerov–Cowell scheme to $\mathcal{O}(k^2h^4)$. The improved approximation for λ_k is given by

$$\widetilde{\Lambda}_k = \Lambda_k + \delta\Lambda_k, \tag{8.90}$$

where, for either method, the correction $\delta\Lambda_k$ is a simple function of the argument that depends only on the type of the boundary conditions:

$$\delta\Lambda_k = \begin{cases} \Delta(k); & y(a) = y(b) = 0, \\ \Delta(k-\frac{1}{2}); & y(a) = y'(b) = 0, \\ \Delta(k-1); & y'(a) = y'(b) = 0. \end{cases}$$

In the case of the "standard" scheme (8.87) we use

$$\Delta(k) = \frac{\pi^2}{(b-a)^2}\left[k^2 - \frac{4N^2}{\pi^2}\sin^2\left(\frac{k\pi}{2N}\right)\right],$$

while for the Numerov–Cowell scheme (8.89) we use

$$\Delta(k) = \frac{\pi^2}{(b-a)^2}\left[k^2 - \frac{4N^2}{\pi^2}\sin^2\left(\frac{k\pi}{2N}\right)\left(1 - \frac{1}{3}\sin^2\left(\frac{k\pi}{2N}\right)\right)^{-1}\right].$$

For Sturm–Liouville problems in the more general form (8.84) we can write a difference scheme

$$-\frac{1}{h}\left(p_{j+1/2}\frac{u_{j+1}-u_j}{h} - p_{j-1/2}\frac{u_j-u_{j-1}}{h}\right) + q_j u_j = \Lambda w_j u_j,$$

which is of order $\mathcal{O}(k^4h^2)$ if the functions p''', q'', and w'' are continuous. For this scheme a correction of the form (8.90) is not available.

8.7.2 Shooting Methods with Prüfer Transformation

Boundary-value problems with eigenvalues can also be solved by shooting (Sect. 8.3): we need to solve initial-value problems by integration from $x = a$ to $x = b$ (or "backwards" from $x = b$ to $x = a$, or "two-way" from $x = a$ to $x = c \in [a, b]$ from the left, and from $x = b$ to $x = c$ from the right). The role of the shooting parameter, unknown in advance, is played by the value of λ. If the boundary conditions are satisfied at the integration endpoint for some λ, that λ is the eigenvalue, and the corresponding solution is the eigenfunction. For example, when shooting from $x = a$ to $x = b$, we start with the values that satisfy the boundary conditions at $x = a$,

$$\big(pu'\big)(a) = a_1, \quad u(a) = a_2.$$

We solve the initial-value problem and denote the solution at $x = b$ by $u_{\mathrm{L}}(x, \lambda)$. A meaningful function that measures the deviation from the exact fulfillment of the boundary conditions at $x = b$, is

$$D(\lambda) = b_1 u_{\mathrm{L}}(b, \lambda) - b_2\big(pu'_{\mathrm{L}}\big)(b, \lambda).$$

The sought eigenvalues λ are the zeros of $D(\lambda)$, which can be found by using the methods from Chap. 2. An analogous procedure can obviously be derived for "backward" shooting from $x = b$ to $x = a$.

Exponentially growing components of the solution in simple one-way shooting may cause instabilities that one can again try to fend off by shooting in two directions (Fig. 8.5 (bottom)). We shoot from $x = a$ with the initial conditions $(pu')(a) = a_1$ and $u(a) = a_2$, yielding the "left part" of the solution, $u_{\mathrm{L}}(x, \lambda)$. By shooting backwards from $x = b$ with the initial conditions $(pu')(b) = b_1$ and $u(b) = b_2$, we obtain the "right part" of the solution, $u_{\mathrm{R}}(x, \lambda)$. A suitable quantity that measures the difference between the solution parts at the point c, where $a < c < b$, is the determinant of Wronski,

$$D(\lambda) = \begin{vmatrix} (pu'_{\mathrm{L}})(c, \lambda) & (pu'_{\mathrm{R}})(c, \lambda) \\ (u_{\mathrm{L}})(c, \lambda) & (u_{\mathrm{R}})(c, \lambda) \end{vmatrix}. \tag{8.91}$$

This expression vanishes precisely when the solutions $u_{\mathrm{L}}(x, \lambda)$ and $u_{\mathrm{R}}(x, \lambda)$ are smoothly joined at $x = c$. Then u_{L} and u_{R}, taken together, represent the eigenfunction across the whole interval, and λ is the corresponding eigenvalue.

In problems where eigenvalues are found in clusters, the functions $D(\lambda)$ defined as in (8.91) may exhibit wild oscillations around zero, with large changes in the amplitudes and distances between the nodes. Such behavior can be alleviated by

Prüfer transformation [19] that is used to transform the functions in the Sturm–Liouville problem to polar coordinates in the (pu', u) plane. The basic (unscaled) form of the transformation is

$$\begin{aligned} pu' &= r\cos\theta, \\ u &= r\sin\theta. \end{aligned}$$

The functions r and θ satisfy the differential equations

$$\begin{aligned} \frac{r'}{r} &= \left[\frac{1}{p} - (\lambda w - q)\right]\sin\theta\cos\theta, \\ \theta' &= \frac{1}{p}\cos^2\theta + (\lambda w - q)\sin^2\theta. \end{aligned} \tag{8.92}$$

In new variables, the boundary conditions (8.85) become

$$\begin{aligned} \theta(a) &= \alpha, \quad \tan\alpha = a_2/a_1, \\ \theta(b) &= \beta, \quad \tan\beta = b_2/b_1. \end{aligned}$$

The Prüfer transformation now reveals its two great charms. If we wish to determine just the eigenvalues (and not the eigenfunctions), we need to solve just one first-order equation (8.92). Besides, their determination is very direct: for a regular Sturm–Liouville problem, where the coefficient functions p, q, and w are piecewise continuous with $p, q > 0$, and where the parameters α and β are normalized such that

$$\alpha \in [0, \pi), \quad \beta \in (0, \pi],$$

the eigenvalue λ_k is the value of λ that solves (8.92) and satisfies

$$\theta(a, \lambda) = \alpha, \qquad \theta(b, \lambda) = \beta + k\pi.$$

In solving the eigenvalue problem with Dirichlet boundary conditions $y(a) = 0$ and $y(b) = 0$, we have to fulfill $\theta(a) = 0$ and $\theta(b) = \pi + k\pi$ (both determined up to an integer multiple of π). Thus, when shooting from $x = a$ to $x = b$, the eigenvalues λ_k are the roots of the equation $D(\lambda) = \theta(b, \lambda) - (k+1)\pi = 0$.

Example (Rescaled Prüfer Transformation; Adapted from [19]) There are pitfalls in seeking the eigenvalues by Prüfer-transformed shooting. The dotted curve in Fig. 8.13 (left) shows $D(\lambda)$ for solving the Mathieu equation

$$-y'' + (2\rho\cos 2x)y = \lambda y, \quad y(-\pi/2) = y(\pi/2) = 0, \ \rho = 15, \tag{8.93}$$

by shooting from $a = -\pi/2$ to $b = \pi/2$. Note the staircase-like character of $D(\lambda)$, where the abscissas of the intercepts of the curve with the ordinates $k\pi$ determine the eigenvalues λ_k. In places where eigenvalues form clusters (e.g. the pairs $\lambda_{0,1}$ or

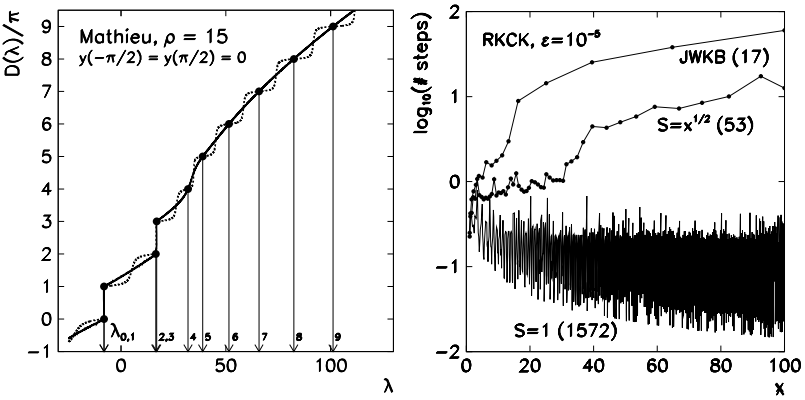


Fig. 8.13 [*Left*] The function $D(\lambda)/\pi$ in solving (8.93) with $\rho = 15$ and boundary conditions $y(-\pi/2) = y(\pi/2) = 0$ by Prüfer-transformed shooting from $x = -\pi/2$ to $x = \pi/2$. The *dotted curve* shows the unscaled version, $S(x, \lambda) = 1$, while the *full curve* shows the rescaled one, $S(x, \lambda) = \sqrt{\max\{1, \lambda - q(x)\}}$. Both curves go through the points (λ_k, k). [*Right*] The step size in integration of the Prüfer equation (8.94) for the initial-value problem $u'' + xu = 0$, $u(1) = 0$, by the adaptive RK4 method (tolerance 10^{-5}): without scaling ($S = 1$), with scaling ($S = \sqrt{x}$), by solving the modified equation $\theta' = \sqrt{x}(1 + 5\sin^2\theta/(16x^3))$. The *numbers in brackets* denote the number of integration steps for integration up to $x = 100$ with precision $\varepsilon = 10^{-5}$

$\lambda_{2,3}$), the problem of finding the intercept is poorly conditioned, because the curve there is almost vertical: no matter how hard we try, when locating the intercept, we tumble into one of the two eigenvalues—it is next to impossible to locate both, in particular if ρ is large. For higher eigenvalues, the problem is poorly conditioned because the curve there is almost horizontal.

One can resort to two tricks: rescaling the Prüfer transformation and changing the way we shoot. The rescaled Prüfer transformation has the form

$$pu' = \sqrt{S} r \cos\theta,$$
$$u = \frac{1}{\sqrt{S}} r \sin\theta,$$

where the function $S(x, \lambda)$ should be such that the staircase behavior of $D(\lambda)$ in a broad range of eigenvalues will become smoother. The differential equations for r and θ are

$$\begin{aligned} \frac{r'}{r} &= \left[\frac{S}{p} - \frac{\lambda w - q}{S}\right] \sin\theta\cos\theta - \frac{1}{2}\frac{S'}{S}\cos 2\theta, \\ \theta' &= \frac{S}{p}\cos^2\theta + \frac{\lambda w - q}{S}\sin^2\theta + \frac{S'}{S}\sin\theta\cos\theta, \end{aligned} \tag{8.94}$$

and the boundary conditions become

$$\theta(a) = \alpha, \quad \tan\alpha = S(a)a_2/a_1,$$

$$\theta(b) = \beta, \quad \tan\beta = S(b)b_2/b_1.$$

In "two-way" shooting with any parameter λ, we obtain the "left" solution $\theta_L(x, \lambda)$ and the "right" solution $\theta_R(x, \lambda)$ that both solve (8.94) in their respective domains, and satisfy

$$\theta_L(a) = \alpha, \qquad \theta_R(b) = \beta.$$

The departure from the boundary condition is best measured by the *Prüfer miss-distance function* $D(\lambda)$,

$$D(\lambda) = \theta_L(c, \lambda) - \theta_R(c, \lambda).$$

The eigenvalues λ_k are then uniquely determined by the equation

$$D(\lambda_k) = k\pi, \quad k = 0, 1, 2, \ldots.$$

The full curve in Fig. 8.13 (left) shows the function $D(\lambda)/\pi$ for the Prüfer shooting solution of the Mathieu equation (8.93) with the scaling function

$$S(x, \lambda) = \begin{cases} 1; & \lambda - 2\rho\cos 2x \le 1, \\ \sqrt{\lambda - 2\rho\cos 2x}; & \lambda - 2\rho\cos 2x > 1. \end{cases}$$

Above $\lambda \approx 20$ the scaling function has smoothened the jumpy $D(\lambda)$ to a humble curve, along which it is much easier to frame the intercepts (λ_k, k) and locate the eigenvalues λ_k. Two-way shooting requires additional care, as the continuity of the solution at the joining point $c \in [a, b]$ has to be ensured. Special attention in rescaling the solution is also called for if the function $S(x)$ has a discontinuity at $x = c$.

The choice of an efficient function $S(x, \lambda)$ and the optimal joining point c sometimes requires a stroke of luck, as illustrated in Fig. 8.13 (right). The figure shows the step size in solving the Prüfer equation (8.94) for the initial-value problem $u'' + xu = 0$ by adaptive RK4 integration without scaling, by using the scaling function $S = \sqrt{x}$, and by calculating with the modified equation (JWKB) $\theta' = \sqrt{x}(1 + 5\sin^2\theta/(16x^3))$—see [19], where you can find further instructions on the choice of the joining point, the scaling function, and on ways of controlling the eigenvalue errors.

8.7.3 Pruess Method

The Pruess method is based on the approximation of the coefficient functions of the *differential equation* with simpler, piecewise continuous functions. We approximate the continuous problem (8.84) and (8.85) by the regular problem

$$-\frac{\mathrm{d}}{\mathrm{d}x}\left(P(x)\frac{\mathrm{d}Y}{\mathrm{d}x}\right) + Q(x)Y = \Lambda W(x)Y, \quad a < x < b, \tag{8.95}$$

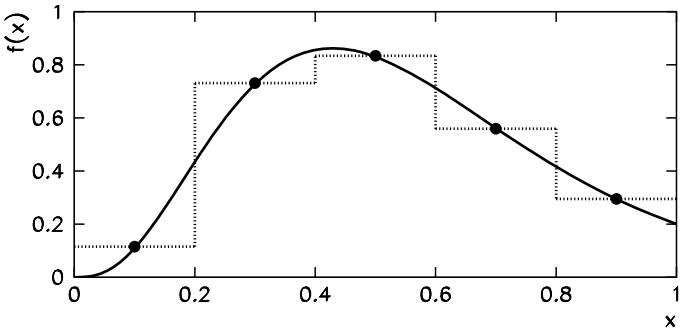


Fig. 8.14 Piecewise constant approximation of the function f, approximating the values $f(x)$ on subintervals (x_{j-1}, x_j) by the midpoint values $f((x_{j-1} + x_j)/2)$

with boundary conditions

$$a_1 Y(a) - a_2 P(a) Y'(a) = 0,$$
$$b_1 Y(b) - b_2 P(b) Y'(b) = 0,$$

where P, Q, and W are approximations of p, q, and w [20]. One could, for example, choose them to be piecewise polynomial functions of degree m that are continuous at the joining points of the subintervals (x_{j-1}, x_j). Because we are approximating the equation rather than its solution, the resulting boundary-value problem preserves the set of eigenvalues (spectrum) of the original problem, which can also be infinite. Certainly we should not expect this property from difference methods with finite-dimensional matrices.

The convergence of the approximate eigenvalues Λ_k to the exact values λ_k of course depends on the polynomial degree m. If P, Q, and W are polynomials that interpolate p, q, and w *on each subinterval* at the Gauss collocation points (see Fig. 8.10 and Table 8.2), we have $|\lambda_k - \Lambda_k| \le C(k)h^{2m+2}$. The simplest approximating functions are constant functions, $m = 0$, and they are used in most practical implementations of the Pruess approximation. We approximate the values $f(x)$ on the subintervals (x_{j-1}, x_j) by the values at the central points $f((x_{j-1} + x_j)/2)$, resulting in a *piecewise continuous midpoint approximation*. Figure 8.14 shows an example on a uniform mesh. For $m = 0$ the piecewise constant approximation is equivalent to Gauss interpolation, and we have

$$|\lambda_k - \Lambda_k| \le Ck^3h^2.$$

In [19] a more restrictive estimate $|\lambda_k - \Lambda_k| \le Ckh^2 \max\{1, |\lambda_k|\}$ can be found that holds for large k. For Sturm–Liouville problems in normal form (8.86), we also have the estimate $|\lambda_k - \Lambda_k| \le Ch^2\sqrt{\max\{1, \lambda_k\}}$.

Assume that we have approximated the functions p, q, and w on intervals (x_{j-1}, x_j) by the constant values $P = p_j$, $Q = q_j$, and $W = w_j$. The solution

of (8.95) on the interval $[x_{j-1}, x_j]$ then has the general form [19]

$$Y(x) = f_j F_j(x) + g_j G_j(x),$$

where F_j and G_j are independent solutions of $-Y'' = k_j Y$, and where

$$k_j = \frac{\Lambda w_j - q_j}{p_j}.$$

The explicit solution is

$$Y(x) = Y(x_{j-1})\Psi_j(x - x_{j-1}) + \big(PY'\big)(x_{j-1})\Phi_j(x - x_{j-1})/p_j, \tag{8.96}$$

where

$$\Psi_j(s) = \begin{cases} \cos\omega_j s; & k_j > 0, \\ 1; & k_j = 0, \\ \cosh\omega_j s; & k_j < 0, \end{cases} \qquad \Phi_j(s) = \begin{cases} \sin(\omega_j s)/\omega_j; & k_j > 0, \\ s; & k_j = 0, \\ \sinh(\omega_j s)/\omega_j; & k_j < 0, \end{cases}$$

and $\omega_j = \sqrt{|k_j|}$. By using (8.96) and its derivative it is easy to show (check it as an exercise) that the solution on one interval can be joined by the solution on the adjacent interval by the matrices

$$\begin{pmatrix} PY' \\ Y \end{pmatrix}_j = T_j \begin{pmatrix} PY' \\ Y \end{pmatrix}_{j-1}, \quad T_j = \begin{cases} T_j^{(+)}; & k_j > 0, \\ T_j^{(0)}; & k_j = 0, \\ T_j^{(-)}; & k_j < 0, \end{cases} \tag{8.97}$$

where

$$T_j^{(+)} = \begin{pmatrix} \cos(\omega_j h_j) & -p_j\omega_j \sin(\omega_j h_j) \\ \sin(\omega_j h_j)/(p_j\omega_j) & \cos(\omega_j h_j) \end{pmatrix},$$

$$T_j^{(0)} = \begin{pmatrix} 1 & 0 \\ h_j/p_j & 1 \end{pmatrix},$$

$$T_j^{(-)} = \begin{pmatrix} \cosh(\omega_j h_j) & p_j\omega_j \sinh(\omega_j h_j) \\ \sinh(\omega_j h_j)/(p_j\omega_j) & \cosh(\omega_j h_j) \end{pmatrix}.$$

We have denoted $h_j = x_j - x_{j-1}$ and have armed ourselves—an exception in this book—for the optional non-uniform mesh. Each inverse matrix T^{-1} can be obtained by simply replacing h by $(-h)$ in T. A good implementation of the Pruess approximation utilizes two-way shooting, in which one uses the mesh (8.2) to shoot from $x = a$ and $x = b$ towards the joining point $c \in [a, b]$. We let the joining point coincide with one of the mesh points, so that $c = x_s$, $0 < s < N$. The sketch of the

algorithm [19] is

Input: Index s of point $c = x_s$, initial values $(PY')_0$ and Y_0 from the boundary condition at $x_0 = a$, and initial values $(PY')_N$ and Y_N from the boundary condition at $x_N = b$, tolerance ε
repeat
  Choose (or change) the value λ;
  for $j = 1$ **step** 1 **to** s **do**
    Compute $(PY', Y)_j$ from $(PY', Y)_{j-1}$ by (8.97)
  end
  $(PY')_{\mathrm{L}}(c) = (PY')_s$;
  $Y_{\mathrm{L}}(c) = Y_s$;
  for $j = N$ **step** -1 **to** $s + 1$ **do**
    Compute $(PY', Y)_{j-1}$ from $(PY', Y)_j$ by (8.97)
    and take into account that $T_j^{-1}(h_j) = T_j(-h_j)$
  end
  $(PY')_{\mathrm{R}}(c) = (PY')_s$;
  $Y_{\mathrm{R}}(c) = Y_s$;
  Form the Wronski determinant $D(\lambda)$ by (8.91);
until $D(\lambda) \le \varepsilon$;
Output: Eigenvalue λ

In commercial codes (see Appendix I) this algorithm is of course enhanced by Prüfer transformation and a prudent choice of the scaling function, by a generator of non-uniform meshes, and numerical error control. Further details can be found in [19].

8.7.4 Singular Sturm–Liouville Problems

So far we have only discussed regular Sturm–Liouville problems of the form (8.84). A problem becomes singular if one or both endpoints become singular. The endpoint a is singular if in its neighborhood any of the coefficient functions $1/p$, q, or w is non-integrable, so if

$$\int_a \left[\frac{1}{p(x)} + |q(x)| + w(x) \right] \mathrm{d}x = +\infty.$$

(An endpoint may also be at infinity.) A well-known example of a singular eigenvalue problem is the radial part of the Schrödinger equation for the hydrogen atom,

$$-\mathcal{R}''(r) + \left[-\frac{1}{r} + \frac{l(l+1)}{r^2} \right] \mathcal{R}(r) = \lambda \mathcal{R}(r), \quad r \in (0, \infty),$$

with eigenvalues $\lambda_k = -1/(2k+4)^2$ for $l = 1$. Naively we might wish to solve this task as if it were a regular problem: we establish regular boundary conditions

$\mathcal{R}(\varepsilon) = \mathcal{R}(R) = 0$ for some small ε and large R, and solve the problem with any previously described method on the interval $[\varepsilon, R]$. If we choose $\varepsilon = 0.0001$ and $R = 1000$, the relative error of the first few eigenvalues with respect to the exact values is $\approx 10^{-12}$. But for higher k, we will be forced to increase R strongly in order to keep the same relative error. For example, with $R = 1000$, already λ_{17} is very imprecise, while λ_{18} even becomes positive. How the differential equation, the boundary conditions, and the expected spectrum drive our choice of ε and R, is beyond our scope.

Indeed, atomic physicists approach the Schrödinger equation with Coulomb potential differently, but universal packages for boundary-value problems (Appendix I, p. 689) do not recognize the physical background of the equations and the most natural solution options. We have taken the above example from [19], which offers excellent reading on singular Sturm–Liouville problems.

8.7.5 Eigenvalue-Dependent Boundary Conditions

Now and then, one encounters Sturm–Liouville problems in which the eigenvalues appear as parameters in the boundary conditions, for example, as in

$$-\big(py'\big)' + q(x)y = \lambda w(x)y, \quad x \in [a, b],$$

with boundary conditions

$$\begin{aligned} \alpha_1 y(a) - \alpha_2\big(py'\big)(a) &= \lambda\big[\alpha_1' y(a) - \alpha_2'\big(py'\big)(a)\big], \\ \beta_1 y(b) + \beta_2\big(py'\big)(b) &= 0, \end{aligned}$$

or

$$\begin{aligned} \alpha_1(\lambda)y(a) - \alpha_2(\lambda)p(a)y'(a) &= 0, \\ \beta_1(\lambda)y(b) + \beta_2(\lambda)p(b)y'(b) &= 0. \end{aligned}$$

The standard difference schemes like (8.88) or (8.89) are inappropriate for such problems, since the corresponding matrix equations become too complicated. In general we obtain equations of the type $K(\Lambda)\boldsymbol{u} = \Lambda\boldsymbol{u}$ or $K(\Lambda)\boldsymbol{u} = \Lambda M(\Lambda)\boldsymbol{u}$. Problems of this type are therefore best attacked by shooting.

Boundary-value problems containing eigenvalues in the boundary conditions can be transformed such that various approaches or approximations can be used to determine the asymptotic behavior of eigenvalues and to compute the eigenfunctions (see [21–25]). So far methods have been developed only for very specific analytic forms of the boundary conditions. Here we mention just a few of the most physically interesting ones, for boundary-value problems in Sturm–Liouville normal form

$$-y'' + qy = \lambda y, \quad x \in [0, 1].$$

The paper [26] discusses transformations between the problems in which both boundary conditions are of "constant" (C) form,

$$\frac{y'}{y}(0) = \alpha, \qquad \frac{y'}{y}(1) = \beta,$$

and problems in which the eigenvalue appears in an "affine" (A),

$$\frac{y'}{y}(1) = \alpha\lambda + \beta,$$

or "bilinear" (B) manner,

$$\frac{y'}{y}(1) = \frac{\alpha\lambda + \beta}{\gamma\lambda + \delta}.$$

Between the forms A, B, and C simple explicit mappings B → A and A → C (thus also B → C) exist, and their spectra are equivalent. Therefore, with appropriate transformations, the boundary-value problem can be brought into the form C, which can also be solved by non-shooting methods. Physically relevant are also problems with the boundary conditions in the form

$$\frac{y'}{y}(0) = \cot\phi, \qquad \frac{y'}{y}(1) = \alpha\lambda + \beta - \sum_n \frac{\gamma_n}{\lambda - \delta_n} = f(\lambda), \tag{8.98}$$

where $\phi \in [0, \pi)$. A transformation exists for such problems that generates a boundary-value problem with different functions $q(x) \to \hat{q}(x)$ and $f(\lambda) \to \hat{f}(\lambda)$, such that $\hat{f}(\lambda)$ either contains less terms than $f(\lambda)$, or these terms contain fewer singularities [27, 28]. By repeated use of this transformation, the problem can be converted to the standard Sturm–Liouville problem whose eigenvalues are almost identical to those of the problem with the condition (8.98). Similar transformations [29] exist for boundary conditions in the form

$$\frac{y'}{y}(1) = \alpha\lambda^2 + \beta\lambda + \gamma.$$

8.8 Isospectral Problems ⋆

As a curiosity in boundary-value problems, we mention the field of inverse problems. In the famous paper [30] Mark Kac asked whether one can "*hear the shape of a drum*". In other words, can we uniquely determine the *shape* of a drum membrane if we know (measure) its eigenfrequencies?

We solve the wave equation $\nabla^2 v = v_{tt}$ on the domain (drum) R by the ansatz $v(x,t) = \phi(x,y)T(t)$, whence $\nabla^2\phi/\phi = T_{tt}/T = \text{const} = \lambda$. The displacement of

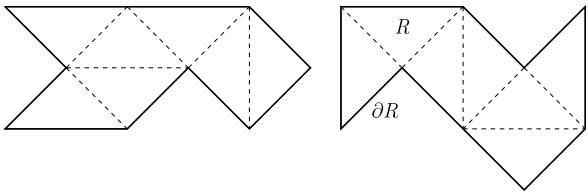


Fig. 8.15 Two geometrically different drum membranes (isospectral domains) with identical discrete sets of eigenfrequencies

the drum membrane from equilibrium is $v(x,t) = \phi(x,y)\sin\sqrt{\lambda}t$, where $\phi(x,y)$ is the solution of the two-dimensional eigenvalue problem

$$\begin{aligned} -\nabla^2\phi &= \lambda\phi \quad \text{on membrane } R, \\ \phi &= 0 \qquad\quad \text{on membrane boundary } \partial R, \end{aligned}$$

and where the eigenvalues λ_i are the squares of the drum's eigenfrequencies.

We are therefore asking whether the geometry of the domain R is uniquely determined by the discrete set of eigenvalues (the spectrum). The interesting answer is: it is not. One can show that the observed eigenvalues do determine some properties of R, like its surface area, circumference, and connectivity [30], but geometrically different domains can be found with precisely the same spectrum, including possible degeneracies [31, 32]. We call such domains isospectral. Figure 8.15 shows the example of two membranes with equal spectra. More on isospectral problems and in general on inverse (ill-posed) problems can be found in [33–36]. Inverse Sturm–Liouville problems, in which the eigenvalue is contained in the boundary conditions, are discussed in [37].

8.9 Problems

8.9.1 Gelfand–Bratu Equation

One of the problems encountered in chemical kinetics involving diffusion with exothermic reactions is the Gelfand–Bratu boundary-value problem

$$y'' = -\delta \mathrm{e}^y, \quad 0 < x < 1,$$

with boundary conditions $y(0) = y(1) = 0$. For $0 < \delta < \delta_c \approx 3.51$ this problem has two solutions, while for $\delta > \delta_c$ there are no real solutions. In the physically relevant domain the two analytic solutions are given by

$$y(x) = -2\ln\left[\frac{\cosh\xi(1-2x)}{\cosh\xi}\right],$$

where ξ is one of the two solutions of $\cosh\xi = \sqrt{8/\delta}\xi$.

⊙ Solve the problem by using difference methods described in Sect. 8.1. Start with $\delta = 1$: in this case the two analytic solutions are defined by the parameters

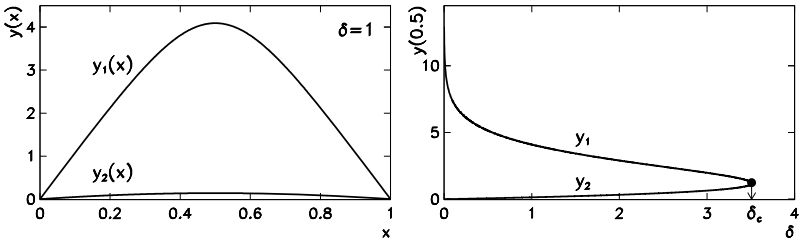


Fig. 8.16 Gelfand–Bratu equation. [*Left*] Two possible analytic solutions at $\delta = 1$. [*Right*] The dependence of the value $y(0.5)$ on the parameter δ (bifurcation diagram)

$\xi_1 \approx 0.379291149762689$ and $\xi_2 \approx 2.734675693030527$ (Fig. 8.16 (left)). Use the explicit iterative method with an appropriate convergence parameter ω. The equation tells us that the solution should be concave on $[0, 1]$, so choose the initial approximation accordingly, e.g. $y(x) = \mu x(1 - x)$, where $\mu = 1$ or $\mu = 16$. Change the parameter δ and note the value of the smaller and larger of the two solutions at $x = 0.5$. By increasing δ you should observe two curves that ultimately meet at $\delta = \delta_c$ (bifurcation diagram, Fig. 8.16 (right)).

⊕ Solve the problem by using the Newton method (8.16) and (8.17).

8.9.2 Measles Epidemic

In a simple epidemiological model [2, 3] of a measles epidemic spreading in a population of size P we assume that any person in the population belongs to one of four groups: at time t there are $S(t)$ susceptible persons; $F(t)$ infectives; $L(t)$ latent persons (infected but with clinical signs as yet unpronounced); and $I(t)$ immunes. So at any time $t \in [0, 1]$ we have

$$S(t) + F(t) + L(t) + I(t) = P.$$

We use dimensionless quantities $y_1 = S/P$, $y_2 = L/P$, $y_3 = F/P$. The dynamics of the infection is described by a system of non-linear differential equations,

$$\begin{aligned}
\dot{y}_1 &= \mu - \beta(t) y_1 y_3 = f_1(t, \boldsymbol{y}), \\
\dot{y}_2 &= \beta(t) y_1 y_3 - y_2/\lambda = f_2(t, \boldsymbol{y}), \\
\dot{y}_3 &= y_2/\lambda - y_3/\eta = f_3(t, \boldsymbol{y}),
\end{aligned}$$

where $\boldsymbol{y} = (y_1, y_2, y_3)^{\mathrm{T}}$ and $\beta(t) = \beta_0(1 + \cos 2\pi t)$. The spreading infection is periodic: the boundary condition is

$$\boldsymbol{y}(1) = \boldsymbol{y}(0).$$

⊙ This non-linear problem has the form (8.19) and you can solve it on $t \in [0, 1]$ by using Newton's method described in Sect. 8.2. Use $y_1(t) = 0.1$, $y_2(t) = y_3(t) = 0.005(1 - \cos 2\pi t)$ as the initial approximation for the iteration. Choose a reasonably large mesh size N, as the division to N intervals implies a $3(N+1) \times 3(N+1)$ Jacobi matrix. Use the parameters $\mu = 0.02$, $\lambda = 0.0279$, $\eta = 0.01$, and $\beta_0 = 1575$.

8.9.3 Diffusion-Reaction Kinetics in a Catalytic Pellet

An important problem in chemical engineering is the computation of diffusion and reaction kinetics in a small porous sphere (a pellet). Assume that within the pellet with radius R a substance (platinum) is distributed in order to catalyze the dehydrogenation of cyclohexane [38]. The kinetics in spherical geometry in the region $0 \le r \le R$ is given by the boundary-value problem

$$D\left[\frac{1}{r^2}\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}c}{\mathrm{d}r}\right)\right] = k\mathcal{R}(c), \quad \left.\frac{\mathrm{d}c}{\mathrm{d}r}\right|_{r=0} = 0,\ c|_{r=R} = c_0,$$

where c is the concentration of cyclohexane, D is the diffusion constant, k is the reaction rate (all at given temperature), while the function $\mathcal{R}(c)$ defines the dependence on concentration. By substituting $r/R \to r$ and $c(r)/c(R) \to c(r)$ we write the problem in dimensionless form

$$\frac{\mathrm{d}^2c}{\mathrm{d}r^2} = \Phi^2\frac{\mathcal{R}(c)}{c_0} - \frac{2}{r}\frac{\mathrm{d}c}{\mathrm{d}r} = f\left(r, c, c'\right), \quad \left.\frac{\mathrm{d}c}{\mathrm{d}r}\right|_{r=0} = 0,\ c|_{r=1} = 1,$$

where $\Phi^2 = kR^2/D$.

⊙ If $\mathcal{R}(c) = c$ the equation is linear. At temperature 700 K the realistic parameter values are $k = 4/\mathrm{s}^{-1}$ and $D = 0.05\ \mathrm{cm}^2/\mathrm{s}$, so that for $R = 2.5$ mm we obtain $\Phi \approx 2.236$. The analytic solution is

$$c(r) = \frac{\sinh \Phi r}{r \sinh \Phi}$$

(see Fig. 8.17 (left)). Solve the problem by using Newton's method from Sect. 8.3.3 (even though the problem is linear). You are solving the initial-value problems

$$\begin{pmatrix} u \\ v \end{pmatrix}' = \begin{pmatrix} v \\ \Phi^2 u - \frac{2}{r}v \end{pmatrix}, \qquad \begin{pmatrix} \xi \\ \eta \end{pmatrix}' = \begin{pmatrix} \eta \\ \Phi^2 \xi - \frac{2}{r}\eta \end{pmatrix},$$

with boundary conditions $u(0) = s$, $v(0) = 0$, $\xi(0) = 1$, and $\eta(0) = 0$. Newton's iteration (8.44) involves functions $\phi(s) = u(1; s) - 1$ and $\phi'(s) = \xi(1; s)$.

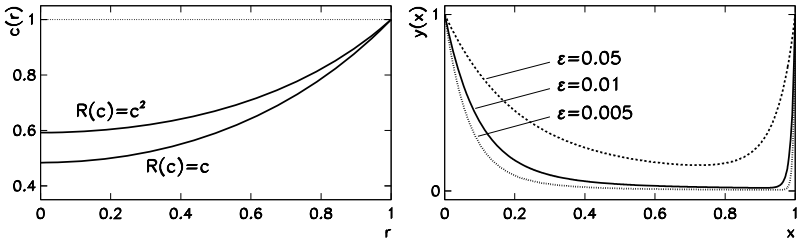


Fig. 8.17 Solutions of Problems 8.9.3 and 8.9.5. [*Left*] Concentration of cyclohexane in the catalytic pellet as a function of radius, for two different functions $\mathcal{R}(c)$ determining the kinetics rate. [*Right*] The boundary-layer problem (see (8.99)) for three values of the parameter ε

We are also interested in the average reaction rate within the pellet and the average rate computed at the surface:

$$E = \left[\int_0^R \mathcal{R}\big(c(r)\big) r^2 \,\mathrm{d}r\right]\left[\int_0^R \mathcal{R}(c_0) r^2 \,\mathrm{d}r\right]^{-1}.$$

The quantity E is a measure for the efficiency of the catalytic process: if mass transfer does not restrict the reaction rate, the average rate is the same in the interior of the pellet and at the surface, and then $E = 1$; if diffusion is significant, the mass transfer reduces the reaction rate within the pellet, indicated by $E < 1$. We integrate the differential equation by parts and obtain

$$D\int_0^R \left[\frac{1}{r^2}\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}c}{\mathrm{d}r}\right)\right] r^2 \,\mathrm{d}r = k\int_0^R \mathcal{R}(c) r^2 \,\mathrm{d}r = DR^2 \frac{\mathrm{d}c}{\mathrm{d}r}\bigg|_{r=R},$$

so that

$$E = \frac{3R^2 D \frac{\mathrm{d}c}{\mathrm{d}r}\big|_{r=R}}{k\mathcal{R}(c_0)R^3}.$$

In dimensionless quantities you are therefore seeking $E = (3/\Phi^2)(\mathrm{d}c/\mathrm{d}r)|_{r=1}$, while the value from the analytic solution is

$$E = \frac{3}{\Phi}\left[\frac{1}{\tanh\Phi} - \frac{1}{\Phi}\right] \approx 0.7727.$$

$\bigoplus$ Solve the pellet problem with $\mathcal{R}(c) = c^2$ by using Newton's method for non-linear problems described in Sect. 8.3.3 With the same boundary conditions as before, you are now solving the coupled initial-value problems

$$\begin{pmatrix} u \\ v \end{pmatrix}' = \begin{pmatrix} v \\ \Phi^2 u^2 - \frac{2}{r}v \end{pmatrix}, \qquad \begin{pmatrix} \xi \\ \eta \end{pmatrix}' = \begin{pmatrix} \eta \\ 2\Phi^2 u\xi - \frac{2}{r}\eta \end{pmatrix}.$$

8.9.4 Deflection of a Beam with Inhomogeneous Elastic Modulus

Under certain assumptions, small vertical deflections of a beam that is simply supported at both endpoints, can be described by the fourth-order boundary-value problem [2, 3]

$$\left(x^3 u''\right)'' = 1,$$
$$u(1) = u''(1) = u(2) = u''(2) = 0,$$

on the interval $x \in [1, 2]$. The analytic solution is

$$u(x) = \frac{10\log 2 - 3}{4}(1 - x) + \frac{1}{2}\left[\frac{1}{x} + (3 + x)\log x - x\right].$$

⊙ Solve this problem by using the collocation method for scalar boundary-value problems of higher orders, as described in Sect. 8.5.2. Use the uniform mesh (8.2) and Gauss collocation of orders $K = 4$ (see Table 8.2) and $K = 6$ (compute the corresponding canonical parameters ρ_k on your own). How crude a mesh (what N) can you afford that the error of the numerical solution is still acceptable?

⊕ Transform this problem to a set of first-order equations and solve it by shooting or by using the difference method.

8.9.5 A Boundary-Layer Problem

Here we discuss the boundary-value problem (adapted from [39]):

$$\varepsilon y'' - x^2 y' - y = 0, \quad 0 < x < 1, \tag{8.99}$$

with boundary conditions $y(0) = y(1) = 1$ and parameter $\varepsilon = 0.01$. An analytic solution exists that involves hypergeometric functions, but it cannot be written in a simple closed form. The main feature of this problem is the rapid change of the solution in the vicinity of the right endpoint $x = 1$ (see Fig. 8.17 (right)). Such a region is known as the *boundary layer* and it requires particular care in order to avoid the potential instabilities in the numerical solution.

⊙ Solve the problem by using the usual difference method described in Sect. 8.1 (see (8.9)), where you can also find the stability criterion for the size of the mesh spacing h. You may also blindly divide the interval $[0, 1]$ to $N = 10$, 20, 50, or 100 subintervals, and observe the corresponding solutions.

Use the same method to study the boundary-layer behavior in the problem

$$\varepsilon y'' - y' + 1 = 0, \quad 0 < x < 1, \tag{8.100}$$

with boundary conditions $y(0) = 1$ and $y(1) = 3$. The analytic solution is $y(x) = 1 + x + (\mathrm{e}^{x/\varepsilon} - 1)/(\mathrm{e}^{1/\varepsilon} - 1)$. Solve the same problem by keeping the central difference for the second derivative, but using the one-sided formula (8.4) for the first

derivative. Write the difference equations in matrix form and solve the resulting equation for $N = 10$, 20, and 40. Discuss the solutions with $\varepsilon = 0.1$ and 0.01. Compare all solutions. How does the error change when N increases? Which method is preferable if ε is reduced still further?

⊕ Solve the problems (8.99) and (8.100) by using the collocation method with cubic B-splines, ending in (8.65). How do the solutions depend on the number of points (or the number of basis functions) on the interval?

Recall that the key step of the collocation method is the requirement that the remainder $\mathcal{R}(x) = y'' + p(x)y' + q(x)y - r(x)$ is zero at the collocation points. There is another way of finding the zero of the remainder: we construct the *least-square error function*

$$E = \int_a^b \big[\mathcal{R}(x)\big]^2 \,\mathrm{d}x,$$

which, of course, depends on the coefficients a_j that enter the expression through the expansion (8.64). We find the zero of $\mathcal{R}$ when we minimize E with respect to a_j, i.e. by requiring $\partial E/\partial a_j = 0$. We obtain the matrix system

$$\sum_{j=0}^{N} \alpha_{ij} a_j = \beta_i,$$

where

$$\alpha_{ij} = \int_a^b \chi_i(x)\chi_j(x)\,\mathrm{d}x, \qquad \beta_i = \int_a^b \chi_i(x) r(x)\,\mathrm{d}x,$$

and $\chi_i(x) = \phi_i''(x) + p(x)\phi_i(x)'' + q(x)\phi_i(x)$. This method is more stable since the error function E in some sense averages the integrand over the interval, while standard collocation "checks" only the agreement at the collocation points. The price we pay for this is an increase in the matrix bandwidth.

8.9.6 Small Oscillations of an Inhomogeneous String

Small transverse deflections $U(x,t)$ of a string with linear mass density ρ are described by the equation

$$\rho(x)\frac{\partial^2 U}{\partial t^2}(x,t) = \frac{\partial}{\partial x}\left(T(x)\frac{\partial U}{\partial x}(x,t)\right) + F(x,t),$$

where T is the longitudinal internal tension and F is the external force per unit length. At the endpoints of $[0, L]$ the string is fixed, so $U(0,t) = U(L,t) = 0$. We are interested in the oscillations of an inhomogeneous string, which we seek by the ansatz $U(x,t) = u(x)\exp(\mathrm{i}\omega t)$. The eigenmodes are given by the equation

$$-\omega^2 \rho(x) u(x) = \frac{\partial}{\partial x}\left(T(x)\frac{\partial u}{\partial x}(x)\right). \tag{8.101}$$

⊙ Determine a few lowest eigenfunctions and eigenvalues of the string with the density

$$\rho(x) = \begin{cases} \rho_0; & x \in [0, L/2], \\ \rho_1; & x \in (L/2, L]. \end{cases}$$

Solve the problem by using the difference method: divide the interval $[0, L]$ to N subintervals with endpoints $x_j = jh$, $h = L/N$, and use this mesh to discretize the string deflections $\boldsymbol{u} = (u_1, u_2, \ldots, u_{N-1})^{\mathrm{T}}$, the first and second derivative in (8.101), as well as ρ and T. You obtain a system of difference equations

$$-(h\omega)^2 \rho_j u_j = T_j(u_{j+1} - 2u_j + u_{j-1}) + \frac{1}{4}(T_{j+1} - T_{j-1})(u_{j+1} - u_{j-1}),$$

valid for $1 \le j \le N-1$. Initially assume constant string tension, $T = \text{const}$. By including the boundary conditions $u_0 = u_N = 0$, the system can be written in matrix form $A\boldsymbol{u} = (h\omega)^2 \boldsymbol{\rho u}$, where A is symmetric, or

$$\rho^{-1/2} A \rho^{-1/2} \boldsymbol{v} = (h\omega)^2 \boldsymbol{v},$$

where $\boldsymbol{v} = \rho^{-1/2}\boldsymbol{u}$ and ρ is the diagonal density matrix $\rho = \mathrm{diag}\{\rho_j\}_{j=1}^{N-1}$. Use methods from Sect. 3.4 to solve the resulting matrix eigenvalue problem.

Solve this problem by shooting: choose an energy $E = \omega^2$ and solve (8.101) as an initial-value problem by using an integrator of your choice (for example, RK4), starting at $x = 0$ with initial conditions $u(0) = 0$ and $u'(0) = 1$ towards $x = L$. Change E until the boundary condition $u(L) = 0$ is satisfied to desired precision. This gives you just one solution, but further eigenmodes can be found in the same manner by realizing that the sequence of eigenvalues follows the number of nodes of u on the interval.

Discuss the solutions obtained by the difference and shooting methods, and compare both to the analytic solution (two modes corresponding to the segments with ρ_0 and ρ_1 joined smoothly at the junction). Plot the computed eigenvalues as a function of the ratio ρ_1/ρ_0.

⊕ Use the difference method to study the oscillation of the string under constant tension, $T = \text{const}$, but with a random density distribution,

$$\rho_j = a w_j \left[\frac{1}{N} \sum_{n=1}^{N-1} w_n \right]^{-1}, \quad w_n = \chi_n + 1,$$

for $a = 0$, 1, 10, and 100. Here χ_n is a random variable uniformly distributed on $[0, 1]$. Analyze the form of the eigenmodes for different N and plot the most appealing ones. At chosen a and N, compute the center-of-mass of the eigenmode, $M(\boldsymbol{u})$, and its dispersion, $S(\boldsymbol{u})$,

$$M(\boldsymbol{u}) = \sum_{j=1}^{N-1} j u_j, \qquad S(\boldsymbol{u}) = \sqrt{\sum_{j=1}^{N-1} j^2 u_j^2 - M(\mathbf{u})^2}.$$

8.9.7 One-Dimensional Schrödinger Equation

In this Problem we are solving the stationary Schrödinger equation

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2\psi}{\mathrm{d}x^2} + V(x)\psi = E\psi$$

in an infinite potential well of width a, with the potential

$$V = \begin{cases} 0; & |x| < a/2, \\ \infty; & |x| \geq a/2, \end{cases}$$

and in a finite well ($V(|x| \geq a/2) = V_0$). The analytic solutions for both cases can be found in any textbook on quantum mechanics. Here we wish to compute the eigenfunctions ψ and eigenenergies E numerically.

⊙ First use the difference method to solve the infinite well problem. Divide the interval $[-a/2, a/2]$ with N points ($x_j = -a/2 + j(a/N)$) and discretize the differential equation accordingly. In dimensionless form you get

$$\frac{\psi_{i-1} - 2\psi_i + \psi_{i+1}}{h^2} + E\psi_i = 0$$

or $\psi_{i-1} - (2-\lambda)\psi_i + \psi_{i+1} = 0$, where $\lambda = Eh^2 = k^2h^2$. You should also discretize the boundary conditions at $x = -a/2$ and $x = a/2$ which, in general (and for the finite well) are of the mixed type,

$$c_1\psi_0 + c_2\frac{\psi_1 - \psi_{-1}}{2h} = 0,$$

$$d_1\psi_N + d_2\frac{\psi_{N+1} - \psi_{N-1}}{2h} = 0,$$

while for the infinite well we clearly have $\psi_0 = \psi_N = 0$. Both cases lead to tridiagonal systems of equations

$$A\boldsymbol{\psi} = \lambda\boldsymbol{\psi}$$

for the eigenvectors $\boldsymbol{\psi} = (\psi_0, \psi_1, \ldots, \psi_N)^{\mathrm{T}}$ and eigenvalues λ, which can be solved by using methods from Sect. 3.4. Determine a few lowest eigenfunctions and eigenvalues! What plays a larger role in the error: the approximation of the second derivative by the central difference or the graininess of the interval (the finite dimension of the matrix being diagonalized)?

Solve the problem by shooting: start with a "cosine" ($\psi(0) = 1$, $\psi'(0) = 0$) or "sine" ($\psi(0) = 0$, $\psi'(0) = 1$) initial condition at the origin, select a value of E, and solve the initial-value problem by integrating the differential equation to $x = a/2$. Check the validity of the boundary condition $\psi(a/2) = 0$. Change E (and/or ψ') until the boundary condition is fulfilled to required precision. This procedure gives you both the odd and even eigenfunctions with the corresponding eigenvalues.

⊕ Use shooting to solve the finite well problem. Note: the spectrum is finite! Only the boundary conditions change; the condition at $x = a/2$ is given by the requirement that the wave-function is smooth at the boundary, and has the form

$$c_1\psi(a/2) + c_2\psi'(a/2) = 0.$$

8.9.8 A Fourth-Order Eigenvalue Problem

Difference and shooting methods are also suitable for the solution of higher-order eigenvalue problems: think of them as Sturm–Liouville problems but involving higher derivatives. An example of such a problem is

$$\frac{\mathrm{d}^4 y}{\mathrm{d}x^4} - \big[\lambda q(x) - r(x)\big]y(x) = 0, \quad x \in [a, b], \tag{8.102}$$

with boundary conditions

$$y(a) = y(b) = y'(a) = y'(b) = 0.$$

We approximate the differential equation by the difference scheme on a discrete mesh (8.2). As usual we use u_j to denote the approximation of $y(x_j)$, as well as $q_j = q(x_j)$ and $r_j = r(x_j)$. We approximate the fourth derivative at x_j, $j = 2, 3, \ldots, N-2$, by using (25.3.28) from [9],

$$\left.\frac{\mathrm{d}^4 y}{\mathrm{d}x^4}\right|_{x=x_j} \approx \frac{u_{j-2} - 4u_{j-1} + 6u_j - 4u_{j+1} + u_{j+2}}{h^4}.$$

We write the boundary conditions as $u(a) = u_0 = 0$, $u(b) = u_N = 0$, and

$$u'(a) \approx \frac{u_1 - u_{-1}}{2h} = 0, \qquad u'(b) \approx \frac{u_{N+1} - u_{N-1}}{2h} = 0.$$

At the left endpoint ($x = x_0 = a$) we use

$$\left.\frac{\mathrm{d}^4 y}{\mathrm{d}x^4}\right|_{x=x_1} \approx \frac{-4u_0 + 7u_1 - 4u_2 + u_3 - 2hu'(a)}{h^4},$$

and analogously at $x = x_N = b$. By using the difference equations at $j = 1$ and $j = N-1$ and the boundary conditions for the first derivative, we eliminate u_{-1} and u_{N+1}, ending up with a system of equations for the solution vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_{N-1})^{\mathrm{T}}$ which has the form of a generalized matrix eigenvalue problem,

$$\big(A + h^4 R\big)\boldsymbol{u} = \Lambda h^4 Q\boldsymbol{u}. \tag{8.103}$$

Here $R = \text{diag}(r_1, r_2, \ldots, r_{N-1})$ and $Q = \text{diag}(q_1, q_2, \ldots, q_{N-1})$, while A is pentadiagonal, with the structure

$$\begin{aligned} A_{1,1} = A_{N-1,N-1} &= 7, \\ A_{i,i} &= 6, \quad i = 2, 3, \ldots, N-2, \\ A_{i,i\pm1} &= -4, \quad i = 1, 2, \ldots, N-2 \text{ or } i = 2, 3, \ldots, N-1, \\ A_{i,i\pm2} &= 1, \quad i = 1, 2, \ldots, N-3 \text{ or } i = 3, 4, \ldots, N-1. \end{aligned}$$

$\odot$ Solve the problem (8.102) by using the difference method described above: solve (8.103) on $[a, b] = [1, \mathrm{e}]$, with the coefficient functions $r(x) = 0$ and $q(x) = 1/x^4$. Find some of the lowest eigenvalues. The exact values of the first five are $\lambda_1 = 531.836459064$, $\lambda_2 = 3919.16273370$, $\lambda_3 = 14865.4293298$, $\lambda_4 = 40373.3097672$, $\lambda_5 = 89795.9545147$ (precise to 12 decimal places). The method is second order, but it can be turned into a fourth-order method if a seven-point difference is used for the fourth derivative. When the boundary conditions are accounted for, we obtain a heptadiagonal matrix given in [40].

$\oplus$ Solve this problem by shooting, and compare the results to those obtained in the first part. Use the paper [41] for further reference.

References

1. H.B. Keller, *Numerical Methods for Two-Point Boundary-Value Problems* (Blaisdell, Waltham, 1968)
2. U.M. Ascher, R.M.M. Mattheij, R.D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations* (SIAM, Philadelphia, 1995)
3. U. Ascher, R.D. Russell, Reformulation of boundary value problems into "standard form". SIAM Rev. **23**, 238 (1981)
4. E. Isaacson, H.B. Keller, *Analysis of Numerical Methods* (Wiley, New York, 1966)
5. P. Henrici, *Discrete Variable Methods in Ordinary Differential Equations* (Wiley, New York, 1962)
6. P. Henrici, *Error Propagation for Difference Methods* (Wiley, New York, 1963)
7. E.W. Larsen, J.E. Morel, W.F. Miller, Asymptotic solutions of numerical transport problems in optically thick, diffusive regimes. J. Comput. Phys. **69**, 283 (1987)
8. E.W. Larsen, J.E. Morel, Asymptotic solutions of numerical transport problems in optically thick, diffusive regimes, II. J. Comput. Phys. **83**, 212 (1989)
9. M. Abramowitz, I.A. Stegun, *Handbook of Mathematical Functions*, 10th edn. (Dover, Mineola, 1972)
10. D. Knoll, J. Morel, L. Margolin, M. Shashkov, Physically motivated discretization methods. Los Alamos Sci. **29**, 188 (2005)
11. U. Ascher, J. Christiansen, R.D. Russell, Collocation software for boundary-value ODEs. ACM Trans. Math. Softw. **7**, 209 (1981)
12. G. Bader, U. Ascher, A new basis implementation for a mixed order boundary value ODE solver. SIAM J. Sci. Stat. Comput. **8**, 483 (1987)
13. J.R. Cash, M.H. Wright, A deferred correction method for nonlinear two-point boundary value problems: implementation and numerical evaluation. SIAM J. Sci. Stat. Comput. **12**, 971 (1991)
14. W.H. Enright, P.H. Muir, Runge–Kutta software with defect control for boundary value ODEs. SIAM J. Sci. Comput. **17**, 479 (1996)

15. L.F. Shampine, P.H. Muir, H. Xu, A user-friendly Fortran BVP solver. J. Numer. Anal. Ind. Appl. Math. **1**, 201 (2006)
16. J.E. Flaherty, Finite element analysis. CSCI, MATH 6860 Lecture Notes, Rensselaer Polytechnic Institute, Troy, 2000
17. C.T. Fulton, S.A. Pruess, Eigenvalue and eigenfunction asymptotics for regular Sturm–Liouville problems. J. Math. Anal. Appl. **188**, 297 (1994)
18. Q. Kong, A. Zettl, Eigenvalues of regular Sturm–Liouville problems. J. Differ. Equ. **131**, 1 (1996)
19. J.D. Pryce, *Numerical Solution of Sturm–Liouville Problems* (Clarendon, Oxford, 1993)
20. S. Pruess, Estimating the eigenvalues of Sturm–Liouville problems by approximating the differential equation. SIAM J. Numer. Anal. **10**, 55 (1973)
21. J. Walter, Regular eigenvalue problems with eigenvalue parameter in the boundary condition. Math. Z. **133**, 301 (1973)
22. C.T. Fulton, Two-point boundary value problems with eigenparameter contained in the boundary conditions. Proc. R. Soc. Edinb. **77A**, 293 (1977)
23. C.T. Fulton, Singular eigenvalue problems with eigenvalue parameter contained in the boundary conditions. Proc. R. Soc. Edinb. **87A**, 1 (1980)
24. D. Hinton, An expansion theorem for an eigenvalue problem with eigenvalue parameter in the boundary condition. Q. J. Math. Oxf. **2**, 33 (1979)
25. D. Hinton, Eigenfunction expansions for a singular eigenvalue problem with eigenparameter in the boundary condition. SIAM J. Math. Anal. **12**, 572 (1981)
26. P.A. Binding, P.J. Browne, B.A. Watson, Transformations between Sturm–Liouville problems with eigenvalue dependent and independent boundary conditions. Bull. Lond. Math. Soc. **33**, 749 (2001)
27. P.A. Binding, P.J. Browne, B.A. Watson, Sturm–Liouville problems with boundary conditions rationally dependent on the eigenparameter, I. Proc. Edinb. Math. Soc. **45**, 631 (2002)
28. P.A. Binding, P.J. Browne, B.A. Watson, Sturm–Liouville problems with boundary conditions rationally dependent on the eigenparameter, II. J. Comput. Appl. Math. **148**, 147 (2002) (inverse problem)
29. W.J. Code, P.J. Browne, Sturm–Liouville problems with boundary conditions depending quadratically on the eigenparameter. J. Math. Anal. Appl. **309**, 729 (2005)
30. M. Kac, Can one hear the shape of a drum? Am. Math. Mon. **73**(4), 1 (1966)
31. C. Gordon, D. Webb, S. Wolpert, One cannot hear the shape of the drum. Bull. Am. Math. Soc. **27**(1), 134 (1992)
32. S.J. Chapman, Drums that sound the same. Am. Math. Mon. **102**(2), 124 (1995)
33. A. Kirsch, *An Introduction to the Mathematical Theory of Inverse Problems* (Springer, New York, 1996)
34. A.N. Tikhonov, V. Arsenin, *Solutions of Ill-Posed Problems* (Wiley, New York, 1977)
35. A.N. Tikhonov, A.S. Leonov, A.G. Yagola, *Nonlinear Ill-Posed Problems, Vols. I and II* (Chapman and Hall, London, 1998)
36. C.R. Vogel, *Computational Methods for Inverse Problems* (SIAM, Philadelphia, 2002)
37. C.M. McCarthy, W. Rundell, Eigenparameter dependent inverse Sturm–Liouville problems. Numer. Funct. Anal. Optim. **24**, 85 (2003)
38. M.E. Davis, *Numerical Methods and Modeling for Chemical Engineers* (Wiley, New York, 1984)
39. M.H. Holmes, *Introduction to Numerical Methods in Differential Equations* (Springer, New York, 2007)
40. G. Vanden Berghe, M. Van Daele, H. De Meyer, A five-diagonal finite difference method based on mixed-type interpolation for computing eigenvalues of fourth-order two-point boundary-value problems. J. Comput. Appl. Math. **41**, 359 (1992)
41. D.J. Jones, Use of a shooting method to compute eigenvalues of fourth-order two-point boundary value problems. J. Comput. Appl. Math. **47**, 395 (1993)

Chapter 9
Difference Methods for One-Dimensional PDE

Solving partial differential equations (PDE) is so crucial to mathematical physics that we devote three chapters to it. The solution methods largely depend on the type of PDE. The type of a general system of linear first-order equations

$$A\boldsymbol{v}_x + B\boldsymbol{v}_y = \boldsymbol{c},$$

where $\boldsymbol{v} = (v_1, v_2, \ldots, v_M)^{\mathrm{T}}$ is the solution vector, $\boldsymbol{c} = (c_1, c_2, \ldots, c_M)^{\mathrm{T}}$ is the vector of inhomogeneous terms, and A and B are $M \times M$ matrices, is determined by the zeros of the characteristic polynomial $\rho(\lambda) = \det(A - \lambda B)$. The system is *hyperbolic* if $\rho(\lambda)$ has precisely M distinct zeros, or if it has M real zeros and the system $(A - \lambda B)\boldsymbol{u} = \boldsymbol{0}$ possesses precisely M linearly independent solutions. The system is *parabolic* if $\rho(\lambda)$ has M real zeros but $(A - \lambda B)\boldsymbol{u} = \boldsymbol{0}$ does not have M linearly independent solutions. The system without real zeros of $\rho(\lambda)$ is *elliptic*. (The classification of systems with mixed real and complex zeros of $\rho(\lambda)$ is more involved.) For second-order linear PDE of the general form

$$Lv = av_{xx} + bv_{xy} + cv_{yy} + dv_x + ev_y + fv = Q, \tag{9.1}$$

where we seek the scalar solution $v(x, y)$ on some domain on the (x, y)-plane at given initial and boundary conditions, the classification is simpler: it is determined already by the discriminant $\mathcal{D} = b^2 - 4ac$.

Hyperbolic Equations PDE of the hyperbolic type ($\mathcal{D} > 0$) describe time-dependent conservative processes that do not evolve towards a stationary state, for example, the propagation of waves. We will learn the basic numerical approaches on the case of the first-order advection equation

$$v_t \pm cv_x = 0$$

and the second-order wave equation

$$v_{tt} = c^2 v_{xx}.$$

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9_9, © Springer-Verlag Berlin Heidelberg 2012

We will be interested in the dissipation of the numerical solution (diminishing magnitude of the solution or its Fourier components) and its dispersion (different propagation velocity of Fourier components with different wave vectors). Among other, we use high-order non-linear hyperbolic equations to model atmospheric phenomena and solve complex aerodynamic problems.

Parabolic Equations The equations of the parabolic type ($\mathcal{D} = 0$) describe time-dependent dissipative processes that evolve to the stationary final state. We will study the fundamentals on the case of the diffusion equation

$$v_t = D\nabla^2 v$$

in one, two, and three spatial dimensions, which we use to model the diffusion of substances or heat in matter. We shall pay some attention to peculiarities that occur in the presence of non-linearities.

Elliptic Equations The equations of the elliptic type ($\mathcal{D} < 0$) describe systems in equilibrium or stationary states, and their solutions (in the context of variational calculus) usually maximize or minimize the integrals representing the energy of the system. Among the best known is the Poisson equation

$$\nabla^2 v = \rho,$$

which lends itself for the study of stationary heat currents or slow curl-free currents of inviscid and incompressible fluids. Of course, we encounter it time and again in the studies of electrostatics and gravitation.

The classification of PDE is not merely academic. Frequently we meet equations that have different types in different parts of the definition domain. An example is the Euler–Tricomi equation

$$v_{xx} = x v_{yy},$$

which is relevant for the study of motion of bodies in matter with velocities that approach the speed of sound in that matter. This equation is hyperbolic at $x > 0$, parabolic at $x = 0$, and elliptic at $x < 0$. For mixed-type equations, the numerical methods often need to be tailored uniquely. Further problems appear in non-linear PDE, as in the Burgers equation used to describe the propagation of shock waves,

$$v_t + v v_x = D v_{xx},$$

where the dominant regime (hyperbolic or parabolic) is determined by the parameter D. On the other hand, the numerical method itself may leave its footprint in the character of the solution: this danger lurks in difference schemes for hyperbolic equations, where parabolic components often steal their way into the solution (see Sect. 9.8).

In this chapter we discuss *difference methods* for one-dimensional PDE (one space coordinate). Retracing the footsteps of [1], we learn the basics by studying

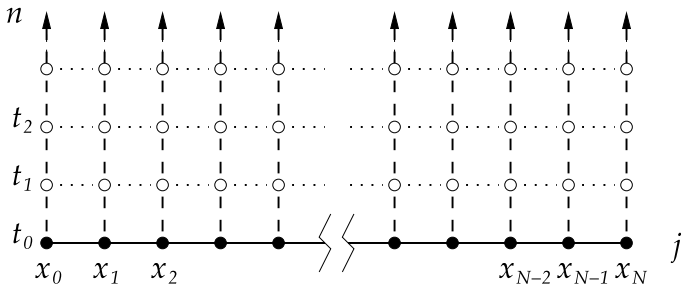


Fig. 9.1 The uniform mesh for the solution of initial-boundary-value problems for evolution (parabolic and hyperbolic) PDE in one temporal and one spatial dimension on the interval $[x_0, x_N]$. The discretization Δt and $\Delta x = (x_N - x_0)/N$ determines the mesh points $(x_j, t_n) = (j\Delta x, n\Delta t)$, and at each of them the true solution v_j^n and its approximation u_j^n. We arrange the solutions in the vector $\boldsymbol{u}^n = (u_0^n, u_1^n, \ldots, u_N^n)^{\mathrm{T}}$

the parabolic problem $v_t - Dv_{xx} = Q$ in Sects. 9.1–9.5. As for ordinary differential equations, solving PDE by difference schemes requires us to grasp three basic concepts: the *consistency* of the difference scheme with the differential equation, the *stability* of the scheme, and the *convergence* of the numerical solution to the true solution. Later we discuss hyperbolic and elliptic PDE.

9.1 Discretization of the Differential Equation

Here we set up difference methods for parabolic PDE involving two independent variables t and x that most often represent the time and space coordinates. Our prototype problem is the linear diffusion equation

$$v_t - Dv_{xx} = Q. \tag{9.2}$$

We shall distinguish pure initial-value problems, e.g. on $x \in \mathbb{R}$ with the initial condition $v(x, 0) = f(x)$, and initial-boundary-value problems, e.g. on $x \in [0, 1]$ with the same initial condition and additional boundary conditions. An example of the former is the time dependence of the temperature of a thin, infinitely long wire; an example of the latter is a thin rod of finite length with both ends held at constant temperature.

Initial-boundary-value problems are more relevant. We discretize each of the continuous variables separately (see Fig. 9.1) by forming a two-dimensional mesh along which we trace the time evolution of the initial condition. We also have to discretize the initial and boundary conditions (see Sect. 9.2). A unique solution of (9.2) exists for pure Dirichlet or mixed Dirichlet–Neumann boundary conditions. For pure Neumann conditions the solution is determined only up to an additive constant: if $v(x, y)$ solves the equation, so does $v(x, y) + C$.

On this mesh we construct various finite differences, for example

$$\left.\frac{\partial v}{\partial x}\right|_{(j\Delta x,n\Delta t)} = \frac{v_{j+1}^n - v_j^n}{\Delta x} + \mathcal{O}(\Delta x),$$

$$\left.\frac{\partial v}{\partial x}\right|_{(j\Delta x,n\Delta t)} = \frac{v_j^n - v_{j-1}^n}{\Delta x} + \mathcal{O}(\Delta x),$$

$$\left.\frac{\partial v}{\partial x}\right|_{(j\Delta x,n\Delta t)} = \frac{v_{j+1}^n - v_{j-1}^n}{2\Delta x} + \mathcal{O}\big(\Delta x^2\big), \tag{9.3}$$

$$\left.\frac{\partial v}{\partial t}\right|_{(j\Delta x,n\Delta t)} = \frac{v_j^{n+1} - v_j^n}{\Delta t} + \mathcal{O}(\Delta t), \tag{9.4}$$

$$\left.\frac{\partial v}{\partial t}\right|_{(j\Delta x,n\Delta t)} = \frac{v_j^{n+1} - v_j^{n-1}}{2\Delta t} + \mathcal{O}\big(\Delta t^2\big), \tag{9.5}$$

$$\left.\frac{\partial^2 v}{\partial x^2}\right|_{(j\Delta x,n\Delta t)} = \frac{v_{j+1}^n - 2v_j^n + v_{j-1}^n}{\Delta x^2} + \mathcal{O}\big(\Delta x^2\big), \tag{9.6}$$

$$\left.\frac{\partial^2 v}{\partial t^2}\right|_{(j\Delta x,n\Delta t)} = \frac{v_j^{n+1} - 2v_j^n + v_j^{n-1}}{\Delta t^2} + \mathcal{O}\big(\Delta t^2\big). \tag{9.7}$$

(Note that expressions (9.4)–(9.7) tell us how well the individual parts of the difference schemes approximate the exact partial derivatives in the differential equation, but this has only an indirect relation to how well the solution u of the difference scheme approximates the true solution v of the differential equation.) In writing down the differences we use the shorthand notation

$$\Delta_+^{(x)} u_j = u_{j+1}^n - u_j^n, \tag{9.8}$$

$$\Delta_-^{(x)} u_j = u_j^n - u_{j-1}^n, \tag{9.9}$$

$$\Delta_0^{(x)} u_j = u_{j+1}^n - u_{j-1}^n, \tag{9.10}$$

$$\Delta_2^{(x)} u_j = u_{j+1}^n - 2u_j + u_{j-1}^n, \tag{9.11}$$

occasionally also

$$\Delta_4^{(x)} u_j = -\frac{1}{12}u_{j+2}^n + \frac{4}{3}u_{j+1}^n - \frac{5}{2}u_j^n + \frac{4}{3}u_{j-1}^n - \frac{1}{12}u_{j-2}^n. \tag{9.12}$$

We can therefore discretize the one-dimensional linear diffusion equation (9.2) as

$$u_j^{n+1} = u_j^n + r\big(u_{j+1}^n - 2u_j^n + u_{j-1}^n\big) + \Delta t q_j^n, \tag{9.13}$$

where we have denoted $r = D\Delta t/\Delta x^2$ and $q_j^n = Q(j\Delta x, n\Delta t)$. This gives us an explicit method of order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$ which advances one step in time by using

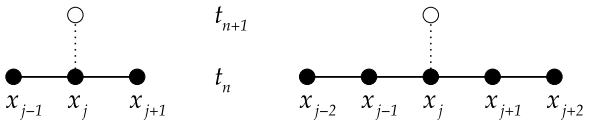


Fig. 9.2 [*Left*] The FTCS scheme with the spatial difference $\Delta_2^{(x)}$ defined by (9.11). [*Right*] The FTCS scheme with the difference $\Delta_4^{(x)}$ (see (9.12)) for which two additional (numerical) boundary conditions at the points x_{-1} and x_{N+1} are needed

the symmetric spatial difference $\Delta_2^{(x)}$, thus we name it *Forward-Time, Centered-Space* (FTCS, Fig. 9.2 (left)). The scheme is stable for $0 < r \leq 1/2$ (we will say more on stability in Sect. 9.5). This method is just one of the possible explicit *two-level* difference methods of the form $\boldsymbol{u}^{n+1} = F\boldsymbol{u}^n + \Delta t \boldsymbol{q}^n$.

9.2 Discretization of Initial and Boundary Conditions

A consistent discretization of the initial and boundary conditions is essential for a correct numerical solution of PDE. If either of these conditions are discretized to an order lower than the order of the difference scheme, the solution also has a lower order. (In general, the order of the numerical solution does not exceed the lowest order of any discrete approximation in the formulation of the problem.) The importance of boundary conditions for the solution in the interior of the definition domain is illustrated in the following and in Fig. 9.3.

In the scheme for the one-dimensional diffusion equation, the easiest boundary conditions to implement are Dirichlet: imagine a rod with a prescribed temperature dependence at its endpoints ($x_0 = 0$ and $x_N = 1$), for example, due to heating and cooling, independent of the inhomogeneous term Q. We discretize the initial condition $v(x, 0) = f(x)$ as

$$u_j^0 = f(j\Delta x), \quad j = 0, 1, \ldots, N, \tag{9.14}$$

and the boundary conditions $v(0, t) = a(t)$ and $v(1, t) = b(t)$ as

$$u_0^{n+1} = a\big((n+1)\Delta t\big) = a^{n+1}, \quad n = 0, 1, \ldots, N, \tag{9.15}$$

$$u_N^{n+1} = b\big((n+1)\Delta t\big) = b^{n+1}, \quad n = 0, 1, \ldots, N. \tag{9.16}$$

Homogeneous Dirichlet conditions ($u_0^n = u_N^n = 0$) are a special case: they describe a rod whose ends are in contact with a heat reservoir at zero temperature.

For one complete time step in the FTCS scheme both (9.15) and (9.16) are essential. Namely, we use (9.13) at $n = 0$ to compute the values u_j^1 for $j = 1, 2, \ldots, N-1$ from the initial condition (9.14), but not for $j = 0$ and $j = N$, since in this case the second difference at the right of (9.13) would reach for indices outside of the definition domain ($j = -1$ and $j = N + 1$, Fig. 9.2 (left)). In one time step we therefore

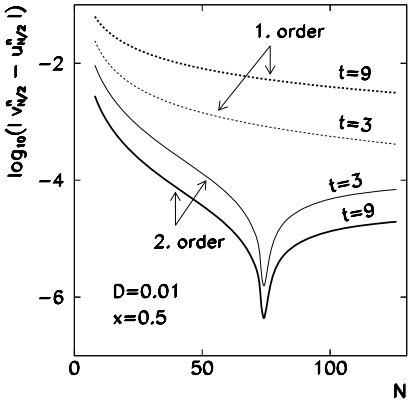


Fig. 9.3 The error of the numerical solution of $v_t = Dv_{xx}$ by the FTCS scheme on $x \in [0, 1]$ for $t > 0$ with the initial condition $v(x, 0) = \sin^2 \pi x$ and boundary conditions $v_x(0, t) = v_x(1, t) = 0$, and $D = 0.01$. Shown is the error at $x = 0.5$ in dependence of the discretization ($\Delta x = 1/N$) for two different treatments of the boundary conditions: to first order (9.17) or to second order (9.18), at two different times $t = n\Delta t$

map $N+1$ values in the space coordinate to the next $N+1$ values: the scheme (9.13) contains $N-1$ equations, and the missing equations are (9.15) and (9.16).

We face a different situation with Neumann boundary conditions which fix the derivative at the endpoints, like $v_x(0, t) = c(t)$, $v_x(1, t) = d(t)$. Such conditions prescribe the heat current density from the rod or into it. (The homogeneous condition $c(t) = 0$ means that the rod is thermally isolated at the left end.) At $x = x_0 = 0$ we get, to first order,

$$\frac{u_1^{n+1} - u_0^{n+1}}{\Delta x} = c^{n+1}, \tag{9.17}$$

thus $u_0^{n+1} = u_1^{n+1} - \Delta x c^{n+1}$ (and analogously at $x = x_N = 1$). Since the FTCS scheme is second-order in the space coordinate, such a crude approximation may spoil the complete numerical solution (see Fig. 9.3). It is preferable to use the symmetric difference

$$\frac{u_1^{n+1} - u_{-1}^{n+1}}{2\Delta x} = c^{n+1},$$

but this formula reaches to a *ghost point* $x_{-1} = x_0 - \Delta x$ outside of the mesh. This additional point implies that we are dealing with $N+2$ unknowns, while having only $N+1$ equations at our disposal. But we can write the FTCS scheme at $(0, n+1)$ by including the symmetric difference in the form $u_{-1}^n = u_1^n - 2\Delta x c^n$, and thereby eliminate the point x_{-1}. At $x = 0$ we get

$$\begin{aligned} u_0^{n+1} &= u_0^n + r\left(u_1^n - 2u_0^n + u_{-1}^n\right) \\ &= u_0^n + 2r\left(u_1^n - u_0^n\right) - 2r\Delta x c^n \end{aligned} \tag{9.18}$$

(and similarly at $x = 1$). Equation (9.13) and two boundary conditions therefore again constitute $N+1$ equations for $N+1$ variables, and all discrete approximations remain consistently of second order in the space coordinate. The advantage of the symmetric difference becomes clear from the example in Fig. 9.3.

9.3 Consistency ⋆

We define the consistency of the differential equation and the corresponding difference scheme in analogy to the ordinary differential equations (Chap. 7). The equation $Lv = Q$ and the scheme $L_j^n u_j^n = q_j^n$ are *point-wise consistent* if for each function $\phi(x,t)$ (even for the true solution v) at the point (x,t) we have

$$(L\phi - Q)_j^n - \left[L_j^n \phi(j\Delta x, n\Delta t) - q_j^n\right] \to 0,$$

when $\Delta x \to 0$, $\Delta t \to 0$, and $(j\Delta x, (n+1)\Delta t) \to (x,t)$. Instead of the point-wise consistency, we sometimes define a more strict *consistency in norm*: the equation that is first-order in time and the explicit two-level scheme

$$\boldsymbol{u}^{n+1} = F\boldsymbol{u}^n + \Delta t \boldsymbol{q}^n \tag{9.19}$$

are consistent when the true solution of the equation v satisfies

$$\boldsymbol{v}^{n+1} = F\boldsymbol{v}^n + \Delta t \boldsymbol{q}^n + \Delta t \boldsymbol{\tau}^n,$$

where the discretization error $\|\boldsymbol{\tau}^n\| \to 0$ when $\Delta x \to 0$, $\Delta t \to 0$ (the vector $\boldsymbol{u}^n$ is defined in the caption to Fig. 9.1, and $\boldsymbol{v}^n$ has the components $v(j\Delta x, n\Delta t)$). We say that the difference scheme is of order (p,q) when $\|\boldsymbol{\tau}^n\| = \mathcal{O}(\Delta x^p) + \mathcal{O}(\Delta t^q)$.

Up to which order in time and space coordinate is the FTCS scheme consistent with the diffusion equation? We Taylor-expand the expressions

$$\begin{aligned} v_j^{n+1} &= v_j^n + \Delta t (v_t)_j^n + \frac{\Delta t^2}{2}(v_{tt})_j^n + \mathcal{O}\big(\Delta t^3\big), \\ v_{j\pm1}^n &= v_j^n \pm \Delta x (v_x)_j^n + \frac{\Delta x^2}{2}(v_{xx})_j^n \pm \frac{\Delta x^3}{6}(v_{xxx})_j^n \\ &\quad + \frac{\Delta x^4}{24}(v_{xxxx})_j^n \pm \frac{\Delta x^5}{120}(v_{xxxxx})_j^n + \mathcal{O}\big(\Delta x^6\big), \end{aligned}$$

and compute the difference between the differential equation and its difference approximation (9.13). We have $v_t = Dv_{xx}$ and $v_{tt} = D(v_{xx})_t = D^2 v_{xxxx}$, whence

$$\begin{aligned} &[v_t - Dv_{xx}]_j^n - \left[\frac{\Delta_+^{(t)}}{\Delta t} - D\frac{\Delta_2^{(x)}}{\Delta x^2}\right] v_j^n \\ &\quad = \frac{D}{2}\left[\frac{\Delta x^2}{6} - D\Delta t\right](v_{xxxx})_j^n + \mathcal{O}\big(\Delta t^2\big) + \mathcal{O}\big(\Delta x^4\big). \end{aligned}$$

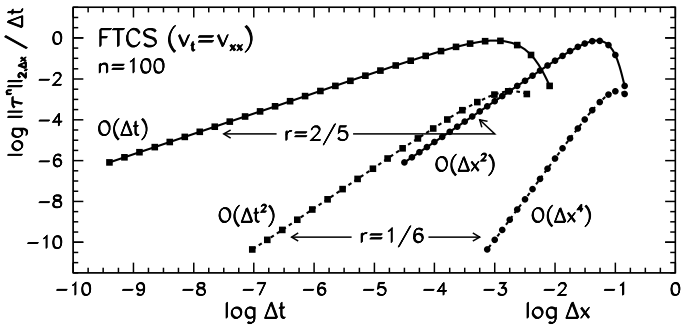


Fig. 9.4 The order of consistency (and convergence) in the $\|\cdot\|_{2,\Delta x}$ norm, of the FTCS scheme for the diffusion equation $v_t = Dv_{xx}$ on [0, 1] with the initial condition $v(x, 0) = \sin \pi x$ and Dirichlet boundary conditions $v(0, t) = v(1, t) = 0$. Shown are two pairs of curves for $r = 2/5$ (asymptotic order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$) and $r = 1/6$ (asymptotic order $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^4)$). The figure for $\|\cdot\|_\infty$ closely resembles this one

We see that with a general $r = D\Delta t/\Delta x^2$, the equation and its difference approximation are point-wise consistent to order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$, while in the special case $r = 1/6$ they are consistent even to $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^4)$ (see Fig. 9.4). The local discretization error in the FTCS scheme can therefore be strongly reduced if we fix $\Delta t = \Delta x^2/(6D)$ once the spatial discretization Δx has been decided for. Alas, with such short steps Δt, the solution evolves very slowly: it takes $6N^2$ steps for it to diffuse from one end of the interval to another.

Consistency in norm requires us to distinguish between initial-value and initial-boundary-value problems. In the "sup"-norm (Appendix A) the initial-value problem for the diffusion equation on $x \in \mathbb{R}$ for $t > 0$ and the FTCS scheme are consistent to order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$, if v_{tt} and v_{xxxx} are uniformly bounded on $\mathbb{R} \times [0, t_0]$ for some $t_0 > t$. If we can restrict $(\|\boldsymbol{v}^n_{tt}\|_{2,\Delta x})^2 < A < \infty$ and $(\|\boldsymbol{v}^n_{xxxx}\|_{2,\Delta x})^2 < B < \infty$, they are also consistent in the $l_{2,\Delta x}$ norm [1].

In initial-boundary-value problems, the order of consistency depends on the discretization of the boundary conditions. For homogeneous Dirichlet conditions, the FTCS scheme on [0, 1] is point-wise consistent to order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$ as well as in the "sup" and $l_{2,\Delta x}$ norms (Fig. 9.4). Things are different if we have, say, a homogeneous Dirichlet condition $u_N^{n+1} = 0$ at $x = 1$, and a homogeneous Neumann condition at $x = 1$. If the latter is discretized to second order (as in (9.18) with $c^n = 0$), the scheme and the equation are consistent only to order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x)$ in "sup" and $l_{2,\Delta x}$ norms: one order less in spite of the effort! Worse still, discretizing it to first order (as in (9.17) with $c^n = 0$), we fail to attain even first-order consistency, since the local discretization error becomes [1]

$$\tau_1^n = -\frac{D}{2}(v_{xx})_1^n + \mathcal{O}(\Delta t) + \mathcal{O}(\Delta x).$$

We can avoid this problem by shifting the mesh for $\Delta x/2$ (*offset grid*): we discretize the spatial coordinate as

$$x_j = (j-1)\Delta x + \Delta x/2, \quad j = 0, 1, \ldots, N,$$

hence $x_0 = -\Delta x/2$ and $x_1 = \Delta x/2$. This gives us a more natural mesh for the homogeneous Neumann boundary condition $u_0^{n+1} = u_1^{n+1}$ at $x = 0$, and makes the scheme consistent with the equation to order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x)$.

9.4 Implicit Schemes

Similarly as in ordinary differential equations (Chap. 7), the stability properties of difference schemes for PDE improve when we resort to implicit methods. Instead of computing the second difference in the space coordinate and the source term at time $n\Delta t$ (as in FTCS), we compute it at time $(n+1)\Delta t$, yielding the implicit two-level *Backward-Time, Centered-Space* (BTCS) scheme of order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$,

$$u_j^{n+1} = u_j^n + r\big(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\big) + \Delta t q_j^{n+1}, \tag{9.20}$$

which is absolutely stable for any r. Similarly, by using the averages of the BTCS and FCTS expressions at the right-hand side of the equation,

$$u_j^{n+1} = u_j^n + \frac{r}{2}\Delta_2^{(x)}\big(u_j^{n+1} + u_j^n\big) + \frac{\Delta t}{2}\big(q_j^{n+1} + q_j^n\big), \tag{9.21}$$

we have formed the Crank–Nicolson method of order $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$, which is absolutely stable. The notion of "absolute stability" should be taken with a grain of salt: see Sect. 9.5, then the discussion in Appendix H, (H.4).

What is the most sensible way to implement the *temporal* discretization of the differential equation? We should be particularly cautious in the study of reaction-diffusion processes described by the equations of the form

$$v_t = Dv_{xx} + bv,$$

especially in cases where the characteristic time scale of the diffusion part of the process, $\tau_\mathrm{d} = L^2/D$, and the scale of the reaction part, $\tau_\mathrm{r} = |1/b|$, are very different. (Here L is the distance on which the solution gradient changes substantially.) Three obvious ways to perform an implicit discretization of this equation are

$$\frac{u_j^\star - u_j^n}{\Delta t} = bu_j^\star, \qquad \frac{u_j^{n+1} - u_j^\star}{\Delta t} - D\Delta_2^{(x)}u_j^{n+1} = 0,$$

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} - D\Delta_2^{(x)}u_j^{n+1} = bu_j^{n+1},$$

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} - D\Delta_2^{(x)}\frac{1}{2}\big[u_j^{n+1} + u_j^n\big] = b\frac{1}{2}\big[u_j^{n+1} + u_j^n\big].$$

Here we do not discuss the nuances of these schemes, but the reader should be aware that there are important distinctions between their solutions [2].

9.5 Stability and Convergence ⋆

The consistency of the difference scheme $L_j^n u_j^n = q_j^n$ with the differential equation $Lv = Q$ by itself does not guarantee that the solution of the difference equation converges to the solution of the differential equation in the sense

$$\left\| \boldsymbol{u}^{n+1} - \boldsymbol{v}^{n+1} \right\| = \mathcal{O}\big(\Delta x^p\big) + \mathcal{O}\big(\Delta t^q\big).$$

For convergence, the scheme also needs to be *stable*. As in the initial-value problems for ordinary differential equations, this means that small errors in the initial condition map to small errors in the solution.

The concepts of consistency, stability, and convergence are linked by the Lax theorem [1]. It tells us that the two-level scheme (9.19) which is consistent with the differential equation to order $\mathcal{O}(\Delta t^q) + \mathcal{O}(\Delta x^p)$ (in some norm) and is stable in this norm, is also convergent in the same order. We should understand this as a warning. We can envision many discretizations and apparently promising difference schemes for any given PDE, but only some of them are stable.

The difference scheme (9.19) is stable in the norm $\|\cdot\|$ when such constants $\Delta x_0 > 0$, $\Delta t_0 > 0$, $\Lambda \geq 0$, and $\beta \geq 0$ exist that

$$\left\| \boldsymbol{u}^{n+1} \right\| \leq \Lambda \mathrm{e}^{\beta t} \left\| \boldsymbol{u}^0 \right\| \tag{9.22}$$

for $0 < \Delta x \leq \Delta x_0$, $0 < \Delta t \leq \Delta t_0$, and $0 \leq (n+1)\Delta t = t$. (The inhomogeneous term $\Delta t \boldsymbol{q}^n$ contributes only to the discretization error $\Delta t \boldsymbol{\tau}^n$, so the definition used here applies both to homogeneous and inhomogeneous schemes. For the remainder of this section we set $\boldsymbol{q} = 0$.) Therefore, the solutions are allowed to grow, but not faster than exponentially. Let us stress the message of (9.22): a stable solution may grow with time, but not with the number of time steps.

9.5.1 Initial-Value Problems

For the stability analysis of difference schemes for initial-value problems with PDE we rely on the discrete Fourier transformation. (By using this tool we already wade into spectral methods for PDE discussed in Chap. 11.) Following [1] we merge the explicit scheme (FTCS), the implicit scheme (BTCS), and the Crank–Nicolson (CN) implicit scheme for the diffusion equation to a single expression

$$\begin{aligned} &-aru_{j-1}^{n+1} + (1+2ar)u_j^{n+1} - aru_{j+1}^{n+1} \\ &\quad = (1-a)ru_{j-1}^n + \big[1 - 2(1-a)r\big]u_j^n + (1-a)ru_{j+1}^n. \end{aligned} \tag{9.23}$$

The value $a = 0$ corresponds to FTCS, the value $a = 1$ to BTCS, while $a = 1/2$ gives the Crank–Nicolson scheme. By using the discrete Fourier transformation $\widetilde{u}^n(\xi) = \sum_j \mathrm{e}^{-\mathrm{i}j\xi} u_j^n$ we get

$$\begin{aligned} -ar\mathrm{e}^{-\mathrm{i}\xi}\widetilde{u}^{n+1}(\xi) + (1+2ar)\widetilde{u}^{n+1}(\xi) - ar\mathrm{e}^{\mathrm{i}\xi}\widetilde{u}^{n+1}(\xi) \\ = (1-a)r\mathrm{e}^{-\mathrm{i}\xi}\widetilde{u}^n(\xi)\big[1-2(1-a)r\big]\widetilde{u}^n(\xi)(1-a)r\mathrm{e}^{\mathrm{i}\xi}\widetilde{u}^n(\xi) \end{aligned}$$

or

$$\left(1 + 4ar\sin^2\frac{\xi}{2}\right)\widetilde{u}^{n+1}(\xi) = \left(1 - 4(1-a)r\sin^2\frac{\xi}{2}\right)\widetilde{u}^n(\xi).$$

We write this as $\widetilde{u}^{n+1}(\xi) = \rho(\xi)\widetilde{u}^n(\xi)$, whence we read off the ratio between the transform at time $(n+1)\Delta t$ and the transform at $n\Delta t$. The ratio

$$\rho(\xi) = \frac{1 - 4(1-a)r\sin^2\frac{\xi}{2}}{1 + 4ar\sin^2\frac{\xi}{2}} \tag{9.24}$$

is called the *symbol* of the difference scheme. We can repeat this step for all subsequent times, hence $\widetilde{u}^{n+1}(\xi) = [\rho(\xi)]^{n+1}\widetilde{u}^0(\xi)$. Obviously the scheme is stable by definition (9.22) if its symbol is bounded at least as

$$\big|\rho(\xi)\big| \le 1 + C\Delta t \le \mathrm{e}^{C\Delta t} \tag{9.25}$$

(Neumann criterion). The one single numerical value of the symbol elegantly "warns" us that during the time evolution one of the Fourier components is spinning out of control. For the FTCS, BTCS, and CN schemes we establish the stability criterion by seeking the maxima of $\partial\rho(\xi)/\partial\xi = 0$. For $a \ge 1/2$ the scheme (9.23) is unconditionally stable, while for $a < 1/2$ it is conditionally stable, with the condition

$$r = D\frac{\Delta t}{\Delta x^2} \le \frac{1}{2(1-2a)}. \tag{9.26}$$

The stability regions of the FTCS and Crank–Nicolson schemes in dependence of the Fourier parameter ξ for different values of r are shown in Fig. 9.5. We will shed a different light on the limiting case of $r = 1/2$ for the FTCS scheme in the discussion of dispersion and dissipation (Sect. 9.10).

Does the FTCS method remain stable if a convection term is added to the diffusion equation, $v_t + cv_x = Dv_{xx}$ with the parameter $c < 0$? The corresponding scheme is

$$u_j^{n+1} = u_j^n + r\Delta_2^{(x)}u_j^n - \frac{R}{2}\Delta_0^{(x)}u_j^n,$$

where $r = D\Delta t/\Delta x^2$ and $R = c\Delta t/\Delta x$. If $r^2 \ge R^2/4$, the parabolic character of the equation is more pronounced; in the opposite case, its hyperbolic nature prevails.

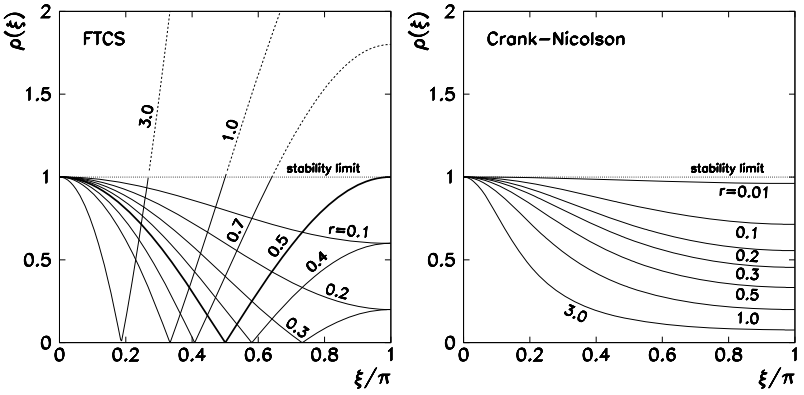


Fig. 9.5 [*Left*] The absolute value of the symbol $\rho(\xi)$ of the conditionally stable FTCS scheme and [*Right*] of the absolutely stable Crank–Nicolson scheme for the diffusion equation, as a function of the Fourier parameter ξ at different $r = D\Delta t/\Delta x^2$. The stability limit and the symbols in the unstable region are denoted by *dotted lines*

The symbol of the scheme $\rho(\xi) = (1-2r) + 2r\cos\xi - \mathrm{i}R\sin\xi$ is complex, with the absolute value

$$\left|\rho(\xi)\right|^2 = (1-2r)^2 + R^2 + 4r(1-2r)\cos\xi + \left(4r^2 - R^2\right)\cos^2\xi.$$

A short calculation [1] shows that we may again choose $\Lambda = 1$, $\beta = C = 0$, and thus bound $|\rho(\xi)| \le 1$ and achieve conditional stability, with the condition

$$R^2/2 \le r \le 1/2 \tag{9.27}$$

(check it). What about the diffusion equation in the form $v_t = Dv_{xx} + bv$ with the parameter $b > 0$? The FTCS scheme with all terms included is

$$u_j^{n+1} = u_j^n + r\Delta_2^{(x)}u_j^n + b\Delta t u_j^n,$$

and has the symbol

$$\rho(\xi) = \left(1 - 4r\sin^2\frac{\xi}{2}\right) + b\Delta t.$$

Now we must choose $\beta = C \neq 0$ and allow the solution to grow with time, although slower than exponentially. Whenever terms with zeroth derivatives are added, one may expect similar behavior. The stability criterion with $\beta \neq 0$ only ensures that the numerical solution converges to the analytic solution when $\Delta t \to 0$ and $\Delta x \to 0$. It does *not* guarantee that this occurs for *any* Δt and Δx.

9.5.2 Initial-Boundary-Value Problems

The stability of schemes for initial-boundary-value problems can be approached by using two tools: operator norms and Fourier transformation. Any two-level scheme for the homogeneous equation ($\boldsymbol{q} = 0$) can be written in the form (9.19). This also implies to implicit schemes $F_1\boldsymbol{u}^{n+1} = F_2\boldsymbol{u}^n$ with invertible matrices F_1, since then $\boldsymbol{u}^{n+1} = (F_1^{-1}F_2)\boldsymbol{u}^n = F\boldsymbol{u}^n$. The criterion (9.22) implies that we must have $\|F^{n+1}\| \leq \Lambda e^{\beta t}$ for stability. The spectral radius of an arbitrary matrix A is bounded as $\rho(A) \leq \|A\|$ (Appendix A.4), so we must ensure that $\rho(F) \leq 1 + C\Delta t$, in analogy to the initial-value problem (9.25).

We insert an illustration that serves as an example for the Problems in Sect. 9.13. Let us write the FTCS scheme by using a spatial discretization at $N+1$ points for the homogeneous equation $v_t = Dv_{xx}$ with the initial condition $v(x,0) = f(x)$ (9.14) and Dirichlet boundary conditions $v(0,t) = 0$ (9.15) and $v(1,t) = 0$ (9.16). In matrix form the scheme becomes $\boldsymbol{u}^{n+1} = F\boldsymbol{u}^n$ where $\boldsymbol{u} = (u_1, u_2, \ldots, u_{N-1})^{\mathrm{T}}$ is the solution vector and F is a symmetric tridiagonal matrix of the form $T(a,b,c)$ (A.6) with $a = r$, $b = 1-2r$, $c = r$. (Check it! Why this equation does not include values at $j = 0$ and $j = N$?) The eigenvalues of F are $\lambda_j = 1 - 2r + 2r\cos(j\pi/N) = 1 - 4r\sin^2(j\pi/2N)$. We will attain stability if the spectral radius (the maximum eigenvalue) can be bounded as

$$\max\left|1 - 4r\sin^2\frac{j\pi}{2N}\right| \leq 1,$$

which translates to

$$0 \leq r \leq \left[2\sin^2\frac{(N-1)\pi}{2N}\right]^{-1}. \tag{9.28}$$

(Convince yourself that to compute the spectral radius you need the eigenvalue at $j = N-1$, not $j = 1$!) What happens if we replace the Dirichlet boundary condition at $x = 0$ by a Neumann condition $v_x(0,t) = 0$ in the form (9.17)? Only the extreme upper left matrix element changes from $1-2r$ to $1-r$. The matrix of the scheme then becomes $I - rT_{N_1D}$, where T_{N_1D} has the form (A.9). The eigenvalues of the matrix $I - rT_{N_1D}$ are $\lambda_j = 1 - r[2 - 2\cos((2j-1)\pi/(2N-1))] = 1 - 4r\sin^2((2j-1)\pi/2(2N-1))$, and we have stability under the condition

$$0 \leq r \leq \left[2\sin^2\frac{(2N-3)\pi}{2(2N-1)}\right]^{-1}.$$

With an ever finer spatial discretization (increasing N) the upper limit of r for stability in both cases approaches the value of $1/2$ *from above*. In both cases we may therefore simply require $0 \leq r \leq 1/2$. Nevertheless, these bounds should not lull the reader into the mistaken belief that this is the only stability criterion that ever appears in relation to the diffusion equation.

In the second approach we apply the Fourier transformation and assign to the approximate solution at each mesh point a *discrete Fourier mode*

$$u_j^n = \xi^n \mathrm{e}^{\mathrm{i}jp\pi\Delta x}, \tag{9.29}$$

where the superscript n in ξ^n indeed means the nth power. We seek the solution of the difference equations in the form

$$u^n(x) = \sum_p c_p^n \mathrm{e}^{\mathrm{i}p\pi x}.$$

Let us derive the criterion for the stability of the FTCS scheme for the equation $v_t = Dv_{xx}$. We insert the terms (9.29) in (9.23) with $a = 0$, yielding

$$\xi^{n+1}\mathrm{e}^{\mathrm{i}pj\pi\Delta x} = \xi^n \mathrm{e}^{\mathrm{i}pj\pi\Delta x}\left(r\mathrm{e}^{-\mathrm{i}p\pi\Delta x} + (1-2r) + r\mathrm{e}^{\mathrm{i}p\pi\Delta x}\right)$$

or $\xi = 1 - 4r\sin^2(p\pi\Delta x/2)$. We have stability if no Fourier mode (in magnitude) exceeds unity, i.e.

$$|\xi| = \left|1 - 4r\sin^2\frac{p\pi\Delta x}{2}\right| \le 1.$$

The ξ is the *amplification factor* and the corresponding inequality is the *discrete* Neumann criterion. (As an exercise, compute ξ for the general scheme (9.23).) We have derived a requirement that is the same as (9.28) if we set $\Delta x = 1/N$, even though we have not mentioned the boundary conditions. Namely, the stability of the scheme for an initial-value problem is just a necessary condition for the stability of the scheme for an initial-boundary-value problem.

There is yet a third way to approach stability that treats the boundary conditions properly and relies on Gershgorin's theorem (see e.g. [1]). This path can be chosen in the cases when the Fourier approach is difficult, for example, when dealing with mixed boundary conditions

$$v(0,t) - g_0 v_x(0,t) = f_0,$$
$$v(1,t) + g_1 v_x(1,t) = f_1,$$

with constant f_j and $g_j \ge 0$. We just give the final result: the *necessary* condition for stability is

$$r \le \min\left\{\frac{1}{2+\Delta x/g_0}, \frac{1}{2+\Delta x/g_1}\right\} < \frac{1}{2}.$$

A consistent inclusion of boundary conditions therefore narrows the range of r for which the scheme for an initial-boundary-value problem is stable.

9.6 Energy Estimates and Theorems on Maxima ⋆

When certain partial differential equations are solved numerically, use can be made of two tools that help us determine the properties of the solution before it is known in detail: energy estimates and theorems on maxima. They are also applicable to non-linear problems, for which analytic solutions or analytic representations of their properties do not exist.

9.6.1 Energy Estimates

The basic example is the Dirichlet problem for the diffusion equation $v_t = v_{xx}$ on $(x,t) \in [0,1] \times [0,\infty)$ with the initial condition $v(x,0) = f(x)$ and boundary conditions $v(0,t) = v(1,t) = 0$. Assume that v, v_t, and v_{xx} on $x \in [0,1]$ for $t \geq 0$ are continuous in x and t. We define a scalar function, the "energy"

$$E(t) = \int_0^1 v^2(x,t)\,\mathrm{d}x,$$

the time evolution of which tells us something about the current solution. The time derivative is

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\int_0^1 v^2(x,t)\,\mathrm{d}x = \int_0^1 \frac{\partial}{\partial t} v^2(x,t)\,\mathrm{d}x = 2\int_0^1 v(x,t)\underbrace{v_t(x,t)}_{v_{xx}(x,t)}\,\mathrm{d}x$$

$$= 2\big[v(x,t)v_x(x,t)\big]_0^1 - 2\int_0^1 \big[v_x(x,t)\big]^2\,\mathrm{d}x = -2\int_0^1 \big[v_x(x,t)\big]^2\,\mathrm{d}x \leq 0.$$

Since v is smooth, one is allowed to exchange the order of integration and differentiation in the first step. In the second step we used integration by parts to swap the spatial derivatives; the integrated portion vanishes due to boundary conditions. Hence, $E(t)$ does not increase with time,

$$E(t) \leq E(0) \quad \Longrightarrow \quad \int_0^1 v^2(x,t)\,\mathrm{d}x \leq \int_0^1 f^2(x)\,\mathrm{d}x, \tag{9.30}$$

i.e. the norm of the solution, measured by the integral $E(t)$, is bounded from above by the norm of the initial data f. This leads to a kind of *stability estimate*: small perturbations of the initial condition lead to small perturbations of the solution: for two different solutions v_1 and v_2 with initial conditions f_1 and f_2 we have

$$\int_0^1 (v_1 - v_2)^2(x,t)\,\mathrm{d}x \leq \int_0^1 (f_1 - f_2)^2(x,t)\,\mathrm{d}x.$$

Charmingly, this criterion is also applicable to non-linear problems, since it does not depend on the representation of the solution. (Check that an estimate of the form (9.30) can also be obtained for the non-linear problem $v_t = v_{xx} - v^3$.)

Difference Schemes Energy estimates can just as well be formulated for solutions of difference schemes which are used to approximate the differential equation. In one space dimension, the discrete "energy" at time $t_n = n\Delta t$ is defined as

$$E^n = \Delta x \sum_{j=0}^{N} \left(u_j^n\right)^2, \tag{9.31}$$

where u_j^n is the numerical solution at $x_j = x_0 + j\Delta x$ at time t_n. (Of course, if we have homogeneous Dirichlet boundary conditions, the outermost terms $u_0 = u_N = 0$ do not contribute.) Further steps depend on the form of the difference scheme. For the explicit scheme (9.13) with $r = \Delta t/\Delta x^2$ we get [3]

$$(1-r)\left(E^{n+1} - E^n\right) \le 0.$$

With the stability condition $r \le 1/2$ (see (9.26) at $\alpha = 0$) this means

$$E^{n+1} \le E^n.$$

In other words, for two different solutions $\boldsymbol{v}^n$ and $\boldsymbol{w}^n$ of the difference scheme with the initial conditions $\boldsymbol{v}^0$ and $\boldsymbol{w}^0$, the following estimate holds true:

$$\Delta x \sum_{j=0}^{N} \left(v_j^n - w_j^n\right)^2 \le \Delta x \sum_{j=0}^{N} \left(v_j^0 - w_j^0\right)^2 \quad \forall n \ge 0.$$

In the sense of Appendix F, the difference scheme (9.13) is a stable discrete dynamical system. The error in the solution is bounded from above by the error in the initial conditions.

9.6.2 Theorems on Maxima

Theorems on maxima are another tool used to become familiar with the continuum (PDE) problem and its difference representation (difference scheme) without knowing the exact solutions. These theorems are non-trivial, but require only the basic knowledge of mathematical analysis; we adopt them from [3].

Consider the one-dimensional diffusion equation $v_t = v_{xx}$ and its non-linear generalization $v_t = (D(v)v_x)_x$ (Problem 9.13.1). Define the initial-boundary-value problems on the rectangle $R = \{(x,t) : x \in [0,1], t \in [0,T]\}$ with the boundary encompassing three sides of R (the one at $t = T$ is missing):

$$\begin{aligned} \partial R = &\left\{(x,t) : x = 0,\ 0 \le t \le T\right\} \cup \left\{(x,t) : t = 0,\ 0 \le x \le 1\right\} \\ &\cup \left\{(x,t) : x = 1,\ 0 \le t \le T\right\}. \end{aligned} \tag{9.32}$$

Then the following theorem holds. Assume that v is continuous on R and solves the problem $v_t = v_{xx}$ with the initial condition $v(x,0) = f(x)$ and boundary conditions $v(0,t) = v_{\mathrm{L}}(t)$ and $v(1,t) = v_{\mathrm{R}}(t)$. Further assume that v_t, v_x, and v_{xx} are continuous on $(0,1) \times (0,T]$. Then

$$\inf_{(x,t)\in\partial R} \big\{ f(x), v_{\mathrm{L}}(t), v_{\mathrm{R}}(t) \big\} \le v(x,t) \le \sup_{(x,t)\in\partial R} \big\{ f(x), v_{\mathrm{L}}(t), v_{\mathrm{R}}(t) \big\} \tag{9.33}$$

for all $(x,t) \in R$. The theorem carries a physical content: the solution cannot reach the maximum *in the interior* of the definition domain; it can only reach it at the beginning (at time $t = 0$) or on one of the space boundaries ($x = 0$ or $x = 1$). An example of the former case is a thin rod at constant temperature that we put in contact with a heat reservoir at lower temperature at its ends: the temperature maximum occurs at $t = 0$, at all later times the temperature in the rod decreases. An example of the latter case is the rod at temperature zero that we start heating at one end to a higher temperature: the temperature maximum is always attained only at this boundary.

The theorem for the linear problem $v_t = v_{xx}$ has another interesting consequence for stability in the sense of sensitivity to initial and boundary conditions. It can be shown that the perturbation in the solution is bounded by the perturbations in the initial or boundary conditions. If the initial conditions are perturbed as $f(x) \to \widetilde{f}(x)$, $v_{\mathrm{L}}(t) \to \widetilde{v}_{\mathrm{L}}(t)$, and $v_{\mathrm{R}}(t) \to \widetilde{v}_{\mathrm{R}}(t)$, we have the estimate

$$\sup_{(x,t)\in R} \big| v(x,t) - \widetilde{v}(x,t) \big| \le \sup_{(x,t)\in\partial R} \big\{ \big|\Delta f(x)\big|, \big|\Delta v_{\mathrm{L}}(t)\big|, \big|\Delta v_{\mathrm{R}}(t)\big| \big\},$$

where $\Delta f(x) = f(x) - \widetilde{f}(x)$, $\Delta v_{\mathrm{L}}(t) = v_{\mathrm{L}}(t) - \widetilde{v}_{\mathrm{L}}(t)$, and $\Delta v_{\mathrm{R}}(t) = v_{\mathrm{R}}(t) - \widetilde{v}_{\mathrm{R}}(t)$. The linearity of the equation is crucial for the connection to stability: if $v(x,t)$ and $\widetilde{v}(x,t)$ are its solutions, then so is $v(x,t) - \widetilde{v}(x,t)$.

Difference Schemes Similar theorems can be derived for difference schemes. The domain R and its partial boundary ∂R (definition (9.32)) should of course be understood in the discrete sense, i.e. at the corresponding points (x_j, t_n). For the approximate solution u_j^n generated by the FTCS scheme (9.13), the following holds:

$$\min_{(x_j,t_n)\in\partial R} \big\{ f(x_j), u_{\mathrm{L}}(t_n), u_{\mathrm{R}}(t_n) \big\} \le u_j^n \le \max_{(x_j,t_n)\in\partial R} \big\{ f(x_j), u_{\mathrm{L}}(t_n), u_{\mathrm{R}}(t_n) \big\} \tag{9.34}$$

for any $(x_j, t_n) \in R$, as long as the stability requirement $r = \Delta t/\Delta x^2 \le 1/2$ is met. For the implicit scheme (9.20) this estimate applies for any $\Delta x > 0$, $\Delta t > 0$.

Non-linear PDE Even for non-linear equations, theorems on maxima tell us something about the nature of their solutions. For the solution of (9.52), discussed in Problem 9.13.1, the result in the continuous case is precisely the same as for the linear equation: (9.33) applies although we cannot derive the additional criterion for stability to perturbations in the initial condition as we have done with the linear equation. On the other hand, the solution of the explicit difference scheme,

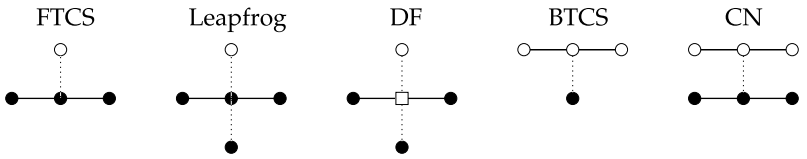


Fig. 9.6 Two-level (FTCS, BTCS, CN) and three-level (Leapfrog, DF) difference schemes for the diffusion equation $v_t - Dv_{xx} = Q$. (Compare to Fig. 9.2.) The properties of the depicted schemes are listed in Table 9.1. In the DF scheme the value of u at $(j\Delta x, n\Delta t)$ does not appear (*empty square*)

e.g. (9.53)–(9.54), can be bounded by the maximum and minimum value precisely by using (9.34), as long as the function $D(v)$ is smooth and strictly positive, $0 < D_{\min} \le D(v) \le D_{\max}$, and as long as $D_{\max}\Delta t/\Delta x^2 \le 1/2$ applies [3].

9.7 Higher-Order Schemes

One might imagine that the order of the FTCS scheme for (9.2) in the time variable could be improved by using the symmetric difference (9.5) instead of (9.4). Indeed, this gives us the explicit *leapfrog* method of order $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$,

$$u_j^{n+1} = u_j^{n-1} + 2r\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) + \Delta t q_j^n. \tag{9.35}$$

This is a three-level scheme (Fig. 9.6): we compute the values at time $(n+1)\Delta t$ from those at times $n\Delta t$ and $(n-1)\Delta t$. But unfortunately this scheme is unstable regardless of r (Sect. 9.5).

The leapfrog method can be stabilized by replacing u_j^n by its time average $\frac{1}{2}(u_j^{n+1} + u_j^{n-1})$. This gives us the explicit three-level Dufort–Frankel (DF) scheme

$$u_j^{n+1} = \frac{2r}{1+2r}\left(u_{j+1}^n + u_{j-1}^n\right) + \frac{1-2r}{1+2r}u_j^{n-1} + \frac{1}{1+2r}\Delta t q_j^n, \tag{9.36}$$

which is unconditionally stable, but only conditionally consistent (and thus only conditionally convergent). Convergence is ensured if r is constant [1].

On the other hand, we can increase the order of the FTCS scheme in the space variable by replacing the three-point difference by a five-point formula,

$$u_j^{n+1} = u_j^n + r\left(-\frac{1}{12}u_{j+2}^n + \frac{4}{3}u_{j+1}^n - \frac{5}{2}u_j^n + \frac{4}{3}u_{j-1}^n - \frac{1}{12}u_{j-2}^n\right) + \Delta t q_j^n. \tag{9.37}$$

The resulting scheme is of order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^4)$ and remains stable, but additional boundary conditions must be specified at the points x_{-1} and x_{N+1} that lie outside of the definition domain (Fig. 9.2 (right)). In this case we prescribe *numerical boundary conditions*. We solve the diffusion equation with homogeneous Dirichlet condition to this order if we use $u_{-1} = u_{N+1} = 0$ (solution vanishes at the

Table 9.1 Select explicit (E) and implicit (I) difference schemes for the solution of the one-dimensional diffusion equation $v_t - Dv_{xx} = Q$, where $r = D\Delta t/\Delta x^2$

Scheme	Type	Order of error	Stability
FTCS (9.13)	E	$\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$	$0 < r \leq 1/2$
Leapfrog (9.35)	E	$\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$	unstable
Dufort–Frankel (9.36)	E	$\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$	stable
FTCS5 (9.37)	E	$\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^4)$	$0 < r \leq 3/8$
BTCS (9.20)	I	$\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$	$\forall r > 0$
Crank–Nicolson (9.21)	I	$\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$	$\forall r > 0$

ghost points) or $u^n_{-1} - 2u^n_0 + u^n_1 = u^n_{N+1} - 2u^n_N + u^n_{N-1} = 0$ (the second derivatives at the endpoints of the rod are zero).

We have described just a few representative explicit schemes for the solution of the one-dimensional diffusion equation. By Taylor expansions and the method of undetermined coefficients it is possible to construct many other schemes [1]. As in ordinary differential equations (Chap. 7), an alternative is offered by the implicit schemes (Table 9.1).

It is difficult to formulate a general advice when to use a high-order scheme. Now and then high-order schemes are a good choice, but if the numerical cost allows it, a finer discretization might be preferable to increasing the order. Table 9.1 summarizes the basic properties of the methods for the solution of (9.2).

9.8 Hyperbolic Equations

In this section we discuss hyperbolic equations of first order,

$$av_x + bv_y = c,$$

where a, b, and c may in general be functions of x, y, and v (but not v_x or v_y), and of second order,

$$av_{xx} + bv_{xy} + cv_{yy} + d = 0,$$

where a, b, c, and d may be functions of x, y, v, v_x, and v_y (but not v_{xx}, v_{xy}, or v_{yy}). (The reader will reassign $x \to t$ and $y \to x$ if needed.) Note that the character of the equation may change on the domain: for example, the equation

$$yv_{xx} + xv_{xy} + yv_{yy} = F(x, y, v, v_x, v_y),$$

is hyperbolic for $|x| > 2|y|$, parabolic for $|x| = 2|y|$, and elliptic for $|x| < 2|y|$.

Characteristics A familiar property of the mentioned quasi-linear hyperbolic equations are the *characteristic curves* or *characteristics*: they represent directions defined at each point of the domain, along which the solution is independent of the partial derivatives in other directions. The method of solving hyperbolic equations

by characteristics excels in the propagation of discontinuities: the sequence of characteristics is a natural mesh for them. But this method becomes cumbersome when the equations become more complex [4]. In the following we therefore seek the solutions by difference methods. We shall return to the problem of discontinuities in Sect. 9.12.

Properties of Solutions In this section and later in Sect. 9.11 we show that the scalar hyperbolic equation

$$v_t + cv_x = 0 \tag{9.38}$$

with a constant value of c is an appropriate model equation for the study of one-dimensional hyperbolic problems. Namely, the related matrix problem

$$\boldsymbol{v}_t = A\boldsymbol{v}_x$$

can be decoupled to a set of equations of the form (9.38) by diagonalizing A. Its analytic solution $v(x,t) = f(x-ct)$ is known: for an arbitrary function f the equation describes the quantity v (for example, a compression in the wave) "carried" by the flow *without change in shape* (no dissipation) along the x-axis with velocity c. In parabolic PDE (diffusion equation) dissipation is contained already in the equation, and that beneficially influences the stability of the corresponding difference schemes. For hyperbolic equations we also desire numerical solutions that maintain an undamped propagation of waves and remain stable. In addition, *dispersion* can be induced by a carelessly designed difference scheme: this occurs when different Fourier components of the solution propagate with different velocities, causing the solution to decay or become distorted.

9.8.1 Explicit Schemes

We start with a naive approximation of the initial-value problem for (9.38). At lowest order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x)$ we discretize the equation by approximating the spatial derivative by $v_x \approx (\Delta_+^{(x)} u_j)/\Delta x$ (*Forward-Time, Forward-Space*, FTFS) or by $v_x \approx (\Delta_-^{(x)} u_j)/\Delta x$ (*Forward-Time, Backward-Space*, FTBS), resulting in

$$u_j^{n+1} = u_j^n - R\big(u_{j+1}^n - u_j^n\big), \tag{9.39}$$

$$\text{or} \quad u_j^{n+1} = u_j^n - R\big(u_j^n - u_{j-1}^n\big), \tag{9.40}$$

where $R = c\Delta t/\Delta x$. The FTFS scheme for $c < 0$ is conditionally stable, with the condition $|R| \le 1$ (and unstable for $c > 0$), while the FTBS scheme is conditionally stable for $c > 0$, with the condition $0 \le R \le 1$ (and unstable for $c < 0$, see also Sect. 9.5). This symmetry between the direction of spatial differencing and stability is related to the behavior of the solution near the characteristics: in the FTFS scheme with $c < 0$ (or FTBS with $c > 0$) the solution for $|R| \le 1$ (or $0 \le R \le 1$) moves

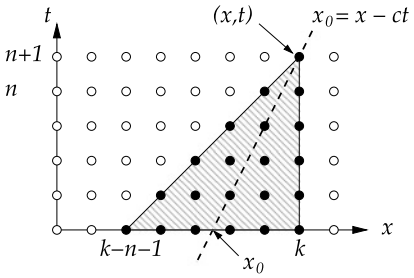


Fig. 9.7 Analytic and numerical domain of dependence of the point (x, t) for the hyperbolic equation $v_t + cv_x = 0$ with the initial condition $v(x, 0) = f(x)$. The analytic domain is the point x_0 (where the characteristic intersects the $t = 0$ axis) and is contained in the numerical domain of the FTBS scheme (9.40), which is $[(k - n - 1)\Delta x, k\Delta x]$

along the characteristics towards the stable solution; in the opposite cases it moves away from them.

Courant–Friedrichs–Lewy Criterion We have discussed the stability of difference schemes for parabolic PDE in Sect. 9.5. The stability conditions pertaining to schemes for hyperbolic PDE are not as easy to formulate. When parabolic schemes, either explicit or implicit, are written in matrix form, we obtain symmetric matrices that allow us to derive necessary *and* sufficient conditions for stability in terms of spectral radii. The corresponding matrices for hyperbolic PDE are often asymmetric, and we can obtain only necessary conditions. In the cases where we are unable to find necessary and sufficient conditions by means of Neumann (discrete Fourier) analysis, we can rely on the Courant–Friedrichs–Lewy (CFL) criterion which serves as a good orientation [1].

The CFL criterion relates the *analytic and numerical domains of dependence*. The analytic domain of dependence of the point (x, t) for the equation $v_t + cv_x = 0$ with the initial condition $v(x, 0) = f(x)$ is the set of points on which the solution of the equation at (x, t) depends. This set includes a single point $x_0 = x - ct$: the solution at (x, t) depends exclusively on the value of the function f at x_0. The numerical domain of dependence is the set of points on the x-axis from which the difference scheme transports the initial condition to the final value at $(x, t) = (k\Delta x, (n + 1)\Delta t)$. Figure 9.7 shows the numerical domain of dependence for the FTBS scheme (9.40): it is the interval $[(k - n - 1)\Delta x, k\Delta x]$ at $t = 0$. All past time steps are contained in the triangle defined by this interval and the apex at (x, t).

The partial differential equation and the *explicit* difference scheme for it satisfy the CFL criterion when the analytic domain for this equation is contained in the numerical domain of the corresponding scheme. But fulfilling the CFL requirement is just a *necessary* condition for stability. If we use the FTBS scheme for $v_t + cv_x = 0$, the analytic domain is $x_0 = x - ct = k\Delta x - c(n + 1)\Delta t = [k - R(n + 1)]\Delta x$, and it is contained in the numerical domain precisely when

$$(k - n - 1)\Delta x \le \big[k - R(n + 1)\big]\Delta x \le k\Delta x.$$

Table 9.2 Consistency and stability properties of the explicit (E) and implicit (I) difference schemes for solving the one-dimensional hyperbolic problem $v_t + cv_x = 0$ with the parameter $R = c\Delta t/\Delta x$

Scheme	Type	Order of error	Stability
FTFS (9.39)	E	$\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x)$	$-1 \le R \le 0$
FTBS (9.40)	E	$\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x)$	$0 \le R \le 1$
FTCS (9.41)	E	$\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$	unstable
Lax–Wendroff (9.42)	E	$\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$	$\lvert R\rvert \le 1$
Lax–Friedrichs (9.43)	E	$\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2/\Delta t)$	$\lvert R\rvert \le 1$
BTFS (9.44)	I	$\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x)$	$R \le 0$
BTBS (9.45)	I	$\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x)$	$R \ge 0$
BTCS (9.46)	I	$\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$	$\forall R$
Lax–Wendroff (9.47)	I	$\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$	$\forall R$
Crank–Nicolson (9.48)	I	$\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$	$\forall R$

The inequalities hold true if $0 \le R \le 1$. In a similar way the necessary conditions for stability of other schemes for hyperbolic PDE can be derived. The results are summarized in Table 9.2. Implicit schemes are absolutely stable and the CFL criterion is not useful for them.

Increasing the Order As in the leapfrog method for the diffusion equation, we can use the centered difference $\Delta_0^{(x)}$ (FTCS) instead of the one-side differences $\Delta_+^{(x)}$ or $\Delta_-^{(x)}$ to gain one order in the space coordinate,

$$u_j^{n+1} = u_j^n - \frac{R}{2}\big(u_{j+1}^n - u_{j-1}^n\big). \tag{9.41}$$

But recall that the stability condition of the scheme for $v_t + cv_x = Dv_{xx}$ was (9.27), while here we have $r = 0$: the scheme (9.41) is therefore unstable.

How do we attain a higher order and yet maintain stability? Since $v_t = -cv_x$ and $v_{tt} = (-cv_x)_t = -(cv_t)_x = c^2 v_{xx}$, we resort to the Taylor expansion

$$\begin{aligned} v_j^{n+1} &= v_j^n + (v_t)_j^n \Delta t + (v_{tt})_j^n \frac{\Delta t^2}{2} + \mathcal{O}\big(\Delta t^3\big) \\ &= v_j^n + (-cv_x)_j^n \Delta t + \big(c^2 v_{xx}\big)_j^n \frac{\Delta t^2}{2} + \mathcal{O}\big(\Delta t^3\big). \end{aligned}$$

The approximations $v_x \approx (\Delta_0^{(x)} u_j^n)/\Delta x$ and $v_{xx} \approx (\Delta_2^{(x)} u_j^n)/\Delta x^2$ give us the *linear Lax–Wendroff scheme*

$$u_j^{n+1} = u_j^n - \frac{R}{2}\Delta_0^{(x)} u_j^n + \frac{R^2}{2}\Delta_2^{(x)} u_j^n, \tag{9.42}$$

with the error of order $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$. (Beware that in this book and in literature there are other schemes that carry the same name.) From the symbol $|\rho(\xi)|^2 =$

$1 - 4R^2 \sin^2(\xi/2) + 4R^4 \sin^4(\xi/2)$ we infer the stability criterion $|R| \leq 1$. We have sinned a bit. If we read the scheme backwards, the equation

$$v_t + cv_x = \frac{c^2 \Delta t}{2} v_{xx}$$

emerges: a diffusive (dissipative) term has appeared at the right-hand side of the equation that damps the solution for all $t > 0$. The consequences will be discussed in Sect. 9.10. Let us also mention the Lax–Friedrichs scheme,

$$u_j^{n+1} = \frac{1}{2}\left(u_{j+1}^n + u_{j-1}^n\right) - \frac{R}{2}\Delta_0^{(x)} u_j^n, \tag{9.43}$$

which is conditionally stable ($|R| \leq 1$) and consistent to $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2/\Delta t)$. Both schemes will become more familiar in Problem 9.13.2 and will be useful for the solution of PDE that can be expressed in conservative form (Sect. 9.12).

9.8.2 Implicit Schemes

If hyperbolic PDE in some specific regimes (e.g. aerodynamic computations involving high Reynolds numbers) were solved by explicit schemes, prohibitively short time steps would be required to ensure stability. Implicit schemes have much better stability properties and become a viable option in such cases. Here we only mention the simplest ones [1]. It is easy to derive the implicit versions of the FTFS (9.39), FTBS (9.40), FTCS (9.41), and Lax–Wendroff schemes (9.42): all one has to do is to evaluate the spatial derivative at time $(n + 1)$ instead of n. The corresponding quartet of implicit schemes is

$$(1 - R)u_j^{n+1} + Ru_{j+1}^{n+1} = u_j^n, \tag{9.44}$$

$$-Ru_{j-1}^{n+1} + (1 + R)u_j^{n+1} = u_j^n, \tag{9.45}$$

$$-\frac{R}{2}u_{j-1}^{n+1} + u_j^{n+1} + \frac{R}{2}u_{j+1}^{n+1} = u_j^n, \tag{9.46}$$

$$\left(-\frac{R^2}{2} - \frac{R}{2}\right)u_{j-1}^{n+1} + \left(1 + R^2\right)u_j^{n+1} + \left(-\frac{R^2}{2} + \frac{R}{2}\right)u_{j+1}^{n+1} = u_j^n. \tag{9.47}$$

If we approximate the term cv_x by the central difference $\Delta_0^{(x)}$ averaged between $n\Delta t$ and $(n + 1)\Delta t$, we obtain the Crank–Nicolson scheme,

$$-\frac{R}{4}u_{j-1}^{n+1} + u_j^{n+1} + \frac{R}{4}u_{j+1}^{n+1} = \frac{R}{4}u_{j-1}^n + u_j^n - \frac{R}{4}u_{j+1}^n, \tag{9.48}$$

with the symbol $\rho(\xi) = (1 - (\mathrm{i}R/2)\sin\xi)/(1 + (\mathrm{i}R/2)\sin\xi)$, for which obviously $|\rho(\xi)| \equiv 1$. The BTCS, implicit Lax–Wendroff, and Crank–Nicolson schemes are

stable regardless of the sign of c, which makes us comfortable in the cases where c is a function of independent variables (when c changes throughout the definition domain). The main properties of the explicit and implicit schemes for the one-dimensional hyperbolic equation $v_t + cv_x = 0$ are summarized in Table 9.2.

9.8.3 Wave Equation

The one-dimensional second-order wave equation

$$v_{tt} = c^2 v_{xx}, \quad x \in [0, 1], c > 0, \tag{9.49}$$

with the initial conditions $v(x, 0) = f(x)$ and $v_t(x, 0) = g(x)$ is important enough to merit its own subsection. It is easy to derive a simple explicit difference scheme for it: we approximate the time derivative at the left as (9.7) and the spatial derivative at the right as (9.6), and obtain

$$u_j^{n+1} = R^2\big(u_{j+1}^n + u_{j-1}^n\big) + 2\big(1 - R^2\big)u_j^n - u_j^{n-1}, \tag{9.50}$$

where $R = c\Delta t/\Delta x$. The scheme (9.50) is stable for $R \leq 1$ and has three levels: we need the solutions at $(n-1)\Delta t$ and $n\Delta t$ to compute the solution at $(n+1)\Delta t$. At $t = 0$ the solution is given by the initial condition, $u_j^0 = f(j\Delta x) = f_j$. But the solution in the first time step $t = \Delta t$—the so-called *initialization scheme*—depends on the discretization of the remaining initial and boundary conditions. For Dirichlet boundary conditions $v(0, t) = a(t)$ and $v(1, t) = b(t)$ we use the approximation $v_t(x, 0) = g(j\Delta x) = g_j \approx (u_j^1 - u_j^0)/\Delta t$ at $j = 0, 1, \ldots, N$ to express

$$u_j^1 = u_j^0 + \Delta t g(j\Delta x) = f_j + \Delta t g_j.$$

One can show that with such initialization the scheme (9.50) is convergent only to order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$ [1]. The symmetric approximation for the time derivative $g_j \approx (u_j^1 - u_j^{-1})/(2\Delta t)$ offers a better initialization

$$u_j^1 = \frac{R^2}{2}(f_{j+1} + f_{j-1}) + \big(1 - R^2\big)f_j + \Delta t g_j$$

(derive this as an exercise). This improvement in the initialization makes the scheme (9.50) convergent at order $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$, as expected from it.

For the wave equation, we could also use the three-level implicit scheme

$$\frac{1}{\Delta t^2}\Delta_2^{(t)} u_j^n = \frac{1}{\Delta x^2}\left[\frac{1}{4}\Delta_2^{(x)} u_j^{n+1} + \frac{1}{2}\Delta_2^{(x)} u_j^n + \frac{1}{4}\Delta_2^{(x)} u_j^{n-1}\right], \tag{9.51}$$

which is of order $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$ and is absolutely stable for all $\Delta t/\Delta x > 0$. It requires us to solve a tridiagonal system of equations at each time step.

In Sect. 9.11 and Problem 9.13.4 we show yet another attractive option, namely how we can use the substitutions $v_1 = cv_x$ and $v_2 = v_t$ to translate the solution of (9.49) to the solution of the system of two first-order hyperbolic equations

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}_t = \begin{pmatrix} 0 & c \\ c & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}_x .$$

Diagonalizing this system reveals the eigenvalues $\pm c$ corresponding to the decoupled scalar equations $(V_1)_t + c(V_1)_x = 0$ and $(V_2)_t - c(V_2)_x = 0$.

9.9 Non-linear Equations and Equations of Mixed Type ⋆

So far we have studied the solution methods for PDE and the criteria for consistency, stability, and convergence of difference schemes only in the case of linear equations. But real life abounds in non-linear equations involving time and space derivatives of higher orders. Here we illustrate some typical approaches to such problems.

Non-linear Diffusion Equation Let us discuss the non-linear equation

$$v_t = \big(D(v)v_x\big)_x, \quad D(v) = \frac{1 + av^2}{1 + bv^2},\ a > b > 0, \tag{9.52}$$

which is used to describe diffusion in porous substances or heat transfer in matter with a temperature-dependent coefficient of thermal conductivity. (This equation is the topic of Problem 9.13.1. An even more interesting case, the two-dimensional growth of biofilms on substrates, is treated in Problem 10.9.2.) We denote $\mathcal{V} = D(v)v_x$ and approximate the time derivative to first order, $v_t(x,t) \approx (u_j^{n+1} - u_j^n)/\Delta t$, and the spatial derivative $\mathcal{V}_x$ by the symmetric difference

$$\mathcal{V}_x(x,t) \approx \big(\mathcal{V}^n_{j+1/2} - \mathcal{V}^n_{j-1/2}\big)/\Delta x,$$

where we use the average of the function $D(v)$ in both terms of the numerator and the one-sided difference for the time derivative. We obtain

$$\mathcal{V}^n_{j+1/2} \approx \frac{1}{2}\big[D\big(u^n_{j+1}\big) + D\big(u^n_j\big)\big]\frac{u^n_{j+1} - u^n_j}{\Delta x}, \tag{9.53}$$

$$\mathcal{V}^n_{j-1/2} \approx \frac{1}{2}\big[D\big(u^n_j\big) + D\big(u^n_{j-1}\big)\big]\frac{u^n_j - u^n_{j-1}}{\Delta x}, \tag{9.54}$$

which ultimately gives us the scheme

$$\frac{u^{n+1}_j - u^n_j}{\Delta t} = \frac{\mathcal{V}^n_{j+1/2} - \mathcal{V}^n_{j-1/2}}{\Delta x}.$$

The discretization itself was easy, but when we wish to establish whether this scheme is stable, we must make the key—and risky—step. If we assume that the

solution v is smooth, $D(v)$ near some point $(x_\star, t_\star)$ may be replaced by the constant value $D_\star = D(u(x_\star, t_\star))$. The solution near that point is then determined by the linear diffusion equation $v_t = D_\star v_{xx}$ or the FTCS scheme (9.13), which is stable if $D_\star \Delta t/\Delta x^2 \le 1/2$ (see (9.26)). The function $D(v)$ is bounded, $D(v) \le a/b$, so the linearized scheme is conditionally stable, with the condition $\Delta t \le (b/a)(\Delta x^2/2)$.

Burgers Equation Linearization by local "freezing" of the numerical solution works surprisingly well even in problems with stronger non-linearities. But the choice of the appropriate difference scheme is always followed by the difficult question of its stability. Further problems may appear in equations of mixed types, where several properties of the solution interlace. A typical example is the Burgers equation

$$v_t + vv_x = Dv_{xx}, \quad D = \text{const},$$

that is used in models of gas dynamics, acoustic phenomena like shock waves, and turbulence. The equation occupies and important place in the hierarchy of the approximations of the Navier–Stokes equation, in particular because it is one of the few non-linear PDE with known analytic solutions (at various boundary conditions). It is therefore a benchmark problem for numerical methods for non-linear PDE. The Burgers equation is of mixed parabolic-hyperbolic type. For small values of D (negligible diffusive term Dv_{xx}) the equation is strongly hyperbolic and the solutions tend to evolve to propagating discontinuities that are hard to resolve numerically. For large D (relatively small non-linear advection term vv_x) its parabolic character is more pronounced.

The non-linearities in the Burgers equation are seemingly easy to handle by an explicit scheme: we attempt

$$u_j^{n+1} = u_j^n - \frac{R}{2} u_j^n \Delta_0^{(x)} u_j^n + r\Delta_2^{(x)} u_j^n, \tag{9.55}$$

where $R = \Delta t/\Delta x$ and $r = D\Delta t/\Delta x^2$. But for small D the scheme is unstable even if we choose $r \le 1/2$ as instructed by (9.26). Stability is restored if we resort to the implicit scheme (BTCS), which we obtain by evaluating the last two terms on the right-hand side of (9.55) at time $(n+1)\Delta t$ instead of at $n\Delta t$,

$$u_j^{n+1} = u_j^n - \frac{R}{2} u_j^{n+1} \Delta_0^{(x)} u_j^{n+1} + r\Delta_2^{(x)} u_j^{n+1}, \tag{9.56}$$

but this gives birth to a new problem: at each time step we need to solve a system of non-linear equations! The most direct approach, also frequently used in practice, is solving (9.56) by Newton's method. Let us assume Dirichlet boundary conditions $u_0 = u_N = 0$ so that the complete solution fits into the vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_{N-1})^{\mathrm{T}} = (u_1^{n+1}, u_2^{n+1}, \ldots, u_{N-1}^{n+1})^{\mathrm{T}}$. We write the system (9.56) in the form $\boldsymbol{f}(\boldsymbol{u})$, where $\boldsymbol{f} : \mathbb{R}^{N-1} \to \mathbb{R}^{N-1}$, with the components

$$f_j(\boldsymbol{u}) = u_j + \frac{R}{2} u_j (u_{j+1} - u_{j-1}) - r(u_{j+1} - 2u_j + u_{j-1}) - u_j^n,$$

where $j = 1, 2, \ldots, N-1$. The Jacobi matrix $[J(\boldsymbol{u})]_{jk} = \partial f_j / \partial u_k$ is tridiagonal,

$$J(\boldsymbol{u}) = \begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & c_3 & \\ & & \ddots & \ddots & \ddots \\ & & & a_{N-1} & b_{N-1} \end{pmatrix},$$

with the matrix elements

$$a_j = -\frac{R}{2} u_j - r, \qquad b_j = 1 + \frac{R}{2}(u_{j+1} - u_{j-1}) + 2r, \qquad c_j = \frac{R}{2} u_j - r.$$

Starting with an appropriate initial approximation (e.g. $\boldsymbol{u}^{n+1} = \boldsymbol{u}^n$) we iterate

$$J(\boldsymbol{u}) \Delta \boldsymbol{u} = -\boldsymbol{f}(\boldsymbol{u}),$$
$$\boldsymbol{u} = \boldsymbol{u} + \Delta \boldsymbol{u},$$

until $\Delta \boldsymbol{u}$ in some norm, e.g. (A.4), drops below the specified tolerance. When the iteration has converged, we have only accomplished a single time step and mapped the solution $\boldsymbol{u}^n$ to $\boldsymbol{u}^{n+1}$.

If solving the non-linear system appears to be too big of a nuisance, we may again try some linearization. We can do this by *lagging* one part of the non-linear term, for example, the zeroth derivative (function value),

$$u_j^{n+1} = u_j^n - \frac{R}{2} \underline{u_j^n} \Delta_0^{(x)} u_j^{n+1} + r \Delta_2^{(x)} u_j^{n+1}. \tag{9.57}$$

This leads to a system of *linear* equations of the form $F \boldsymbol{u}^{n+1} = \boldsymbol{u}^n$ for the solution $\boldsymbol{u}^{n+1}$, with a tridiagonal coefficient matrix F.

The non-linearity in the scheme can also be circumvented by expanding the factors in the non-linear term to first order,

$$u_j^{n+1} = u_j^n + \delta_j. \tag{9.58}$$

When we insert this in (9.56) and neglect all terms quadratic in δ, we again obtain a system of *linear* equations for the correction $\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots, \delta_{N-1})^{\mathrm{T}}$,

$$\left(\frac{R}{2} u_j^n - r \right) \delta_{j+1} + \left(1 + 2r + \frac{R}{2} \Delta_0^{(x)} u_j^n \right) \delta_j + \left(-\frac{R}{2} u_j^n - r \right) \delta_{j-1}$$
$$= -\frac{R}{2} u_j^n \Delta_0^{(x)} u_j^n + r \Delta_2^{(x)} u_j^n. \tag{9.59}$$

We solve this tridiagonal system for $\boldsymbol{\delta}$ by using the current solution $\boldsymbol{u}^n$, which appears at both sides of the equation, and compute $\boldsymbol{u}^{n+1}$ by (9.58).

The numerical solution of non-linear PDE is so full of pitfalls that it is nearly impossible to formulate a set of common instructions. In most cases, the stability criteria developed for linear problems are not applicable; we have only limited control over dispersion and dissipation (Sect. 9.10); without approximations, we are forced to solve systems of non-linear equations. With almost every new equation we learn from the start. Problem 9.13.5 helps us realize how the simple methods described above work (or fail) in practice in the case of the Burgers equation. Problem 9.13.7 is devoted to the related Korteweg–de Vries equation, and Problem 9.13.9 to the cubic Schrödinger equation.

9.10 Dispersion and Dissipation ⋆

Analytic solutions of hyperbolic and parabolic one-dimensional PDE can be written as the sum of the terms $v(x,t) = \widehat{v}\exp[\mathrm{i}(\omega t + kx)]$. Such functions satisfy the chosen PDE only if a particular analytic connection—a dispersion relation—exists between the frequency of the wave ω and the wave vector $k = 2\pi/\lambda$.

Example (Rephrased from [1] whose definitions and notation we adopt in this section) In the advective equation, $v_t \pm cv_x = 0$, the dispersion relation holds only if $\omega = \mp kc$, with the solution components $v(x,t) = \widehat{v}\exp[\mathrm{i}(kx - \omega t)]$: such a wave propagates with the velocity $\mp c = \omega/k$ that does not depend on the frequency, and with a constant amplitude.

The equation $v_t \pm cv_{xxx} = 0$ corresponds to the dispersion relation $\omega = k^3 c$ and the solution is $v(x,t) = \widehat{v}\exp[\mathrm{i}(kx + k^3 ct)]$: waves travel without dissipation with velocities $\pm k^2 c$ in directions opposite to those of the first-order equation, but we witness dispersion: components with different wave vectors propagate with different velocities.

The dispersion relation for the diffusion equation $v_t = Dv_{xx}$ is $\omega = \mathrm{i}Dk^2$, and the solution components are $v(x,t) = \widehat{v}\exp(-Dk^2 t)\exp(\mathrm{i}kx)$: such "waves" do not propagate and their amplitude diminishes with time.

What about dispersion and dissipation in the *difference schemes* for PDE? In the usual discretization $x = j\Delta x$ and $t = n\Delta t$, the solution components are

$$u_j^n = \widehat{u}\mathrm{e}^{\mathrm{i}(kj\Delta x + \omega n\Delta t)}. \tag{9.60}$$

The component with the highest frequency in the solution corresponds to the term $\exp[2\pi\mathrm{i}(N/2)j\Delta x]$, thus the interval $0 \le k\Delta x \le \pi$ contains the information on all components in the Fourier representation of the solution. We separate the real and imaginary parts of the dispersion relation $\omega = \omega(k)$,

$$\omega = \alpha(k) + \mathrm{i}\beta(k),$$

thus $\mathrm{e}^{\mathrm{i}\omega t} = \mathrm{e}^{\mathrm{i}\alpha t}\mathrm{e}^{-\beta t}$, and

$$u_j^n = \widehat{u}\left(\mathrm{e}^{-\beta\Delta t}\right)^n \mathrm{e}^{\mathrm{i}k[j\Delta x - (-\alpha/k)n\Delta t]}$$

(k is the wave vector, while j is the space index). The discrete dispersion relation is intimately related to the symbol of the difference scheme, $\mathrm{e}^{\mathrm{i}\omega\Delta t} = \rho(k\Delta x)$ (Sect. 9.5). The real part,

$$\alpha\Delta t = \arctan\left[\frac{\operatorname{Im}\rho(k\Delta x)}{\operatorname{Re}\rho(k\Delta x)}\right],$$

determines the propagation and dispersion, while the imaginary part,

$$\beta\Delta t = -\log\left|\rho(k\Delta x)\right|,$$

determines the dissipation of the solution. This immediately allows us to classify the solutions of a given difference scheme:

$$\begin{array}{ll}
\alpha = 0 \text{ for all } k & \Longrightarrow \text{ solution does not propagate,} \\
\alpha \neq 0 \text{ for some } k & \Longrightarrow \text{ solution propagates with velocity } -\alpha/k, \\
\mathrm{d}^2\alpha/\mathrm{d}k^2 \neq 0 & \Longrightarrow \text{ scheme is dispersive,} \\
\beta > 0 \text{ for some } k & \Longrightarrow \text{ scheme is dissipative,} \\
\beta < 0 \text{ for some } k & \Longrightarrow \text{ solution diverges (scheme unstable),} \\
\beta = 0 \text{ for all } k & \Longrightarrow \text{ scheme is non-dissipative.}
\end{array}$$

Example Let us inspect the dispersion and dissipation properties of the FTCS scheme (9.13) for the diffusion equation. The symbol of the scheme is given by (9.24) where $a = 0$ and $\xi = k\Delta x$, thus $\exp(\mathrm{i}\omega\Delta t) = 1 - 4r\sin^2(k\Delta x/2)$, and hence

$$\alpha = 0, \qquad \omega = \mathrm{i}\beta = -\frac{\mathrm{i}}{\Delta t}\ln\left|1 - 4r\sin^2\frac{k\Delta x}{2}\right|.$$

The solution u_j^n therefore does not propagate ($\alpha = 0$). We now see (9.26) in a different light from the viewpoint of the parameter β: if $r > 1/2$, then $\beta < 0$, so the factor $(\exp(-\beta\Delta t))^n$ diverges and the scheme is unstable. But if $r \leq 1/2$, we have $\beta \geq 0$, and $(\exp(-\beta\Delta t))^n$ reduces the amplitude of the solution. All Fourier components fade out, only the one with $\beta = 0$ survives, and this one determines the asymptotics. The dissipation of the solution of the difference scheme follows the physical dissipation dictated by the diffusion equation.

Example We repeat the exercise for the FTFS scheme (9.39) for the hyperbolic advective equation $v_t + cv_x = 0$ with $c < 0$ and $R = c\Delta t/\Delta x$. When we insert the Fourier expansion (9.60) in the scheme, we obtain the dispersion relation $\exp(\mathrm{i}\omega\Delta t) = 1 + R - R\cos k\Delta x - \mathrm{i}R\sin k\Delta x$, whence we read off

$$\alpha = -\frac{1}{\Delta t}\arctan\left[\frac{R\sin k\Delta x}{1 + 2R\sin^2\frac{k\Delta x}{2}}\right], \tag{9.61}$$

$$\beta = -\frac{1}{2\Delta t}\log\left[(1+R)^2 - 2R(1+R)\cos k\Delta x + R^2\right]. \tag{9.62}$$

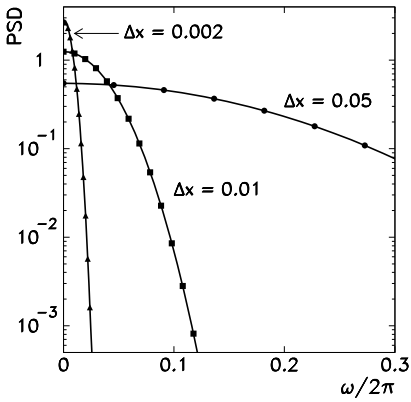


Fig. 9.8 Power spectral density of the function $v_j^0 = \sin^{40} \pi j \Delta x$ (as an initial condition for the equation $v_t + c v_x = 0$) with spatial discretizations $\Delta x = 0.05$, $\Delta x = 0.01$, and $\Delta x = 0.002$. Only the lowest Fourier components are relevant at small Δx

Since α is non-linear in k, we have $\mathrm{d}^2\alpha/\mathrm{d}k^2 \neq 0$ and the scheme (9.39) is therefore always dispersive. If $\beta < 0$, the scheme is unstable. If $\beta > 0$, the scheme is stable for $|R| \leq 1$ and then all solution components with $k \neq 0$ decay, while the $k = 0$ component neither grows nor decays. The scheme is dissipative, except for $R = -1$, when $\beta = 0$ for all k. It is impossible to completely remove dissipation from (9.39), but it can be limited by choosing a sufficiently small Δx. Namely, by reducing Δx we shift the solution components to lower frequencies, where dissipation is less pronounced (see Fig. 9.8 and [1]).

From the dispersion relations (9.61) and (9.62) we see that high-frequency components of the numerical solution (small λ or $k\Delta x$ near π) propagate with velocity $-\alpha/k = 0$ if $R > -1/2$, or with velocity $-\alpha/k = c/R$ if $R < -1/2$. Low-frequency components (large λ or $k\Delta x \ll 1$) are more interesting: if $|R| < 1/2$, they propagate slower than the components of the analytic solution, and faster if $1/2 < |R| < 1$. But sometimes there is a relief [1]: the components with the most "wrong" velocities are also the most strongly damped. If the underlying problem (differential equation) or the corresponding difference scheme contain dissipation, it often hides the effects of dispersion. Figure 9.9 (left) shows the error of the propagation velocity $c - (-\alpha/k)$ in dependence of R for the FTFS scheme. The error is smallest near the stability limit ($R = -1$).

A similar analysis can be performed for the Crank–Nicolson scheme. Its symbol is $|\rho(k\Delta x)| = 1$, so it is non-dissipative, but is has strong dispersion: as shown in Fig. 9.9 (right), the phase error can be reduced somewhat by decreasing Δx, but even this works only for low-frequency components. This means that solutions with broad Fourier spectra are hard to represent on the mesh. In general, Δx must be sufficiently small for high-frequency components to become irrelevant for the solution. See also Problem 9.13.2.

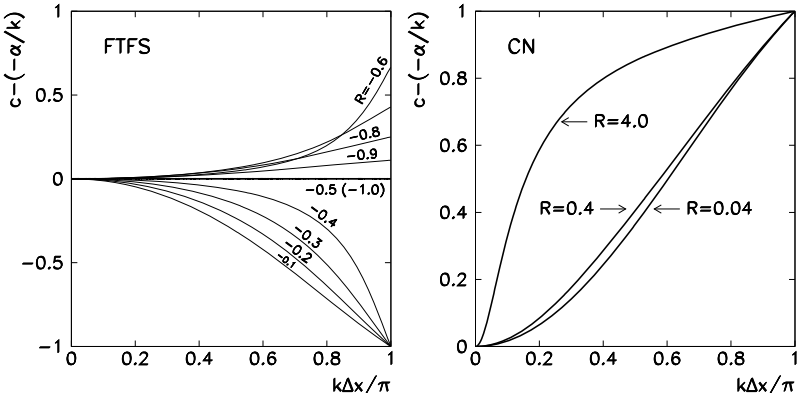


Fig. 9.9 [*Left*] Error of propagation velocity $c-(-\alpha/k)$ as a function of R in the FTFS scheme. The error is smallest near the stability limit ($R=-1$). [*Right*] Error of propagation velocity in the Crank–Nicolson scheme. By further decrease of R the curves become barely distinguishable from the curve corresponding to $R=0.04$

9.11 Systems of Hyperbolic and Parabolic PDE ⋆

In this section we discuss difference schemes for some classes of systems of one-dimensional hyperbolic and parabolic PDE. We adopt the definitions of consistency, stability, and convergence with obvious generalizations from our treatment of the scalar cases (Sects. 9.3 and 9.5). By the Lax theorem, consistent stable schemes are also convergent. Stability is again probed by the discrete Fourier transformation (p. 476): the role of the symbol of the scheme is now taken by the *amplification matrix* $G(\xi)$ measuring the growth of the solution's Fourier components in a given time step, $\widetilde{\boldsymbol{u}}^{n+1}(\xi)=G(\xi)\widetilde{\boldsymbol{u}}^{n}=G^{n+1}(\xi)\widetilde{\boldsymbol{u}}^{0}$. In the following we give the stability criteria of the basic schemes without proof.

A general system of one-dimensional linear hyperbolic PDE with constant coefficients can be written as

$$\boldsymbol{v}_t = A\boldsymbol{v}_x, \quad \boldsymbol{v}(x,0)=\boldsymbol{v}_0(x),\ x\in\mathbb{R},\ t>0, \tag{9.63}$$

where $\boldsymbol{v}=(v_1,v_2,\ldots,v_M)^{\mathrm{T}}$ is the vector of unknowns and A is a $M\times M$ matrix. Let us assume that A is diagonalizable, so that a matrix S exists such that the similarity transformation $D=SAS^{-1}$ diagonalizes A, hence $S\boldsymbol{v}_t=SA\boldsymbol{v}_x=SAS^{-1}S\boldsymbol{v}_x=DS\boldsymbol{v}_x$, and S^{-1} is the matrix containing the eigenvectors of A. The new vector $\boldsymbol{V}=S\boldsymbol{v}$ can be used to translate the original system of coupled equations for the *primitive variables* $\boldsymbol{v}$ to the diagonal system $\boldsymbol{V}_t=D\boldsymbol{V}_x$, i.e.

$$(V_m)_t-\lambda_m(V_m)_x=0, \quad m=1,2,\ldots,M, \tag{9.64}$$

for the *characteristic variables* $\boldsymbol{V}$, where λ_m are the eigenvalues of A. The individual solutions of the decoupled system obviously solve the scalar problem (9.38), $V_m(x,t)=V_{0m}(x+\lambda_m t)$, where $\boldsymbol{V}_0=(V_{01},V_{02},\ldots,V_{0M})^{\mathrm{T}}=S\boldsymbol{v}_0$ is the initial

condition for (9.64). This looks like a set of M waves propagating with phase velocities λ_m along the positive or negative x-axis, depending on the sign of the eigenvalues λ_m. The solution of the original problem (9.63) is

$$\boldsymbol{v}(x,t) = S^{-1}\boldsymbol{V} = S^{-1}\begin{pmatrix} V_{01}(x+\lambda_1 t) \\ V_{02}(x+\lambda_2 t) \\ \vdots \\ V_{0M}(x+\lambda_M t) \end{pmatrix}.$$

Hyperbolic Systems For hyperbolic systems of the form

$$\boldsymbol{v}_t = A\boldsymbol{v}_x + B_0\boldsymbol{v}, \quad x \in \mathbb{R},\ t > 0, \tag{9.65}$$

where A and B_0 are real constant $M \times M$ matrices, and A is diagonalizable (with eigenvalues λ_m), two explicit schemes are immediately at hand:

$$\boldsymbol{u}_j^{n+1} = RA\boldsymbol{u}_{j+1}^n + (I - RA)\boldsymbol{u}_j^n + \Delta t B_0 \boldsymbol{u}_j^n, \tag{9.66}$$

$$\boldsymbol{u}_j^{n+1} = (I + RA)\boldsymbol{u}_j^n - RA\boldsymbol{u}_{j-1}^n + \Delta t B_0 \boldsymbol{u}_j^n, \tag{9.67}$$

where $R = \Delta t/\Delta x$. The scheme (9.66) is stable in the $l_{2,\Delta x}$-norm if $\lambda_m \geq 0$ and $\lambda_m R \leq 1$ for each $m \in \{1, 2, \ldots, M\}$. In the same sense the scheme (9.67) is stable exactly when all $\lambda_m \leq 0$ and $|\lambda_m| R \leq 1$. For the system (9.65) in which the eigenvalues of A have different signs, we would like to use some "mixture" of the methods (9.66) and (9.67). We achieve this by *flux splitting* which is a popular trick in systems of non-linear PDE. We arrange the eigenvalues along the diagonal of the diagonal matrix $D = SAS^{-1}$ in decreasing order $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$ and construct matrices D_+ and D_- containing only the part of D with positive or negative eigenvalues, respectively. Then we use the matrices $A_+ = S^{-1}D_+S$ and $A_- = S^{-1}D_-S$ to form the split scheme

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n + RA_+\Delta_+^{(x)}\boldsymbol{u}_j^n + RA_-\Delta_-^{(x)}\boldsymbol{u}_j^n + \Delta t B_0 \boldsymbol{u}_j^n, \tag{9.68}$$

where $\Delta_+^{(x)}$ and $\Delta_-^{(x)}$ are defined by (9.8) and (9.9). There are also schemes in which we do not need to pay attention to the sign of the eigenvalues of A, for example, the Lax–Wendroff scheme,

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n + \frac{R}{2}A\Delta_0^{(x)}\boldsymbol{u}_j^n + \frac{R^2}{2}A^2\Delta_2^{(x)}\boldsymbol{u}_j^n + \Delta t B_0 \boldsymbol{u}_j^n, \tag{9.69}$$

which is of order $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)$. It is stable if $|\lambda_m| R \leq 1$. The matrix A should be evaluated at time $n\Delta t$, so $A = A(\boldsymbol{u}^n)$, $A^2 = [A(\boldsymbol{u}^n)]^2$.

Parabolic Systems For parabolic systems of the form

$$\boldsymbol{v}_t = B_2\boldsymbol{v}_{xx} + B_0\boldsymbol{v}, \quad x \in \mathbb{R},\ t > 0,$$

we could suggest the explicit scheme

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n + r B_2 \Delta_2^{(x)} \boldsymbol{u}_j^n + \Delta t B_0 \boldsymbol{u}_j^n,$$

where $r = \Delta t / \Delta x^2$ (the diffusion constant D, which we kept throughout the scalar case, can be absorbed in the matrix B_2). If B_2 is a real, symmetric, and positive definite $M \times M$ matrix (with eigenvalues λ_m), and B_0 is bounded, the above scheme is stable precisely when $r\lambda_m \le 1/2$ for each m.

Mixed-Type Systems We obtain a much more general class of systems of PDE if the diffusion (parabolic) and advection (hyperbolic) terms are included simultaneously,

$$\boldsymbol{v}_t = B_2 \boldsymbol{v}_{xx} + B_1 \boldsymbol{v}_x + B_0 \boldsymbol{v}, \quad x \in \mathbb{R},\ t > 0,$$

where B_0, B_1, and B_2 are real constant matrices and B_2 is positive definite. In such cases we should devise a sensible difference scheme on the basis of the relative magnitudes of the eigenvalues of B_1 and B_2. If the eigenvalues of B_2 are substantially larger than the eigenvalues of B_1, the parabolic nature of the equations will prevail over the hyperbolic, and the scheme of the form

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n + r B_2 \Delta_2^{(x)} \boldsymbol{u}_j^n + \frac{R}{2} B_1 \Delta_0^{(x)} \boldsymbol{u}_j^n + \Delta t B_0 \boldsymbol{u}_j^n \tag{9.70}$$

will behave decently (will be stable). In the opposite case, the hyperbolic character dominates and the scheme will become unstable, just as (9.41) was unstable for scalar equations. We face an even greater confusion if some eigenvalues of B_2 are smaller than those of B_1, while some are larger: in such instances the whole system must be carefully decoupled and components with the same regime rejoined. Systems of this type are too complicated to be discussed in this book. Yet a simplified variant of the parabolic-hyperbolic system,

$$\boldsymbol{v}_t = \boldsymbol{v}_{xx} + B_1 \boldsymbol{v}_x + B_0 \boldsymbol{v}, \quad x \in \mathbb{R},\ t > 0,$$

where B_1 and B_0 are real constant matrices, and where B_1 has M non-degenerate eigenvalues, is manageable and still interesting enough: a good scheme for it is (9.70) where one sets $B_2 = I$. It is stable if for all eigenvalues λ_m of B_1 we have $\lambda_m^2 R^2/2 \le r \le 1/2$. If for some m the opposite inequality $r < \lambda_m^2 R^2/2$ is fulfilled (dominating hyperbolic term), it is better to differentiate the term $B_1 \boldsymbol{v}_x$ "hyperbolically" (Table 9.2).

We do not discuss implicit schemes for systems of PDE. (Their main advantage are the improved stability properties: many of them are absolutely stable.) We also disregard systems of PDE with more than one spatial dimension. However, the problems in solving such systems are not immense; further reading can be found in [1].

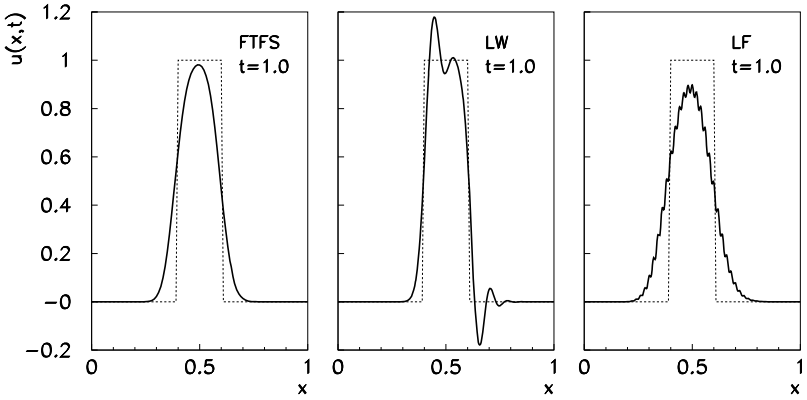


Fig. 9.10 The numerical solution of the hyperbolic problem $v_t - v_x = 0$ with periodic boundary conditions $v(0,t) = v(1,t)$ and the "step" initial condition (*dotted line*). Shown are the solutions by the FTFS scheme [*Left*], Lax–Wendroff scheme [*Center*] and Lax–Friedrichs scheme [*Right*] at time $t = 1.0$

9.12 Conservation Laws and High-Resolution Schemes ⋆

Section 9.10 taught us that seemingly well-designed difference schemes that are both consistent and stable, do not perform satisfactorily for certain classes of PDE: the deficiencies can usually be spotted in the suspicious behavior of the solution near discontinuities. Figure 9.10 shows the time evolution of the solution of $v_t - v_x = 0$ by using three difference schemes: regardless of the method, dispersion and dissipation distort the solutions already within a single period.

Nevertheless, by using a special approach to solve PDE expressed in conservative forms, it is possible to resolve and monitor the discontinuities evolving from the initial conditions, as well as their propagation velocities. In this section we discuss the solution of initial-boundary-value problems for PDE that can be written in the form of a conservation law for some quantity (e.g. mass, momentum, or energy). For such equations one can devise consistent, stable schemes with low dissipation and dispersion [5].

Initially we discuss conservation laws in the form

$$\boldsymbol{v}_t + \big[\boldsymbol{F}(\boldsymbol{v})\big]_x = \boldsymbol{0}, \tag{9.71}$$

where we assume that the function $\boldsymbol{F} : \mathbb{R}^N \to \mathbb{R}^N$ is convex ($\partial^2 F_j / \partial v_k^2 \geq 0$). A scalar example of such a law with $F(v) = v^2/2$ is the Burgers equation without diffusion,

$$v_t + v v_x = v_t + \left(\frac{1}{2} v^2\right)_x = 0.$$

The corresponding scheme is at hand: we integrate (9.71) from $x_{j-1/2}$ to $x_{j+1/2}$ over x and from $t_n = n\Delta t$ to $t_{n+1} = (n+1)\Delta t$ over t, resulting in

$$\Delta x\big(\boldsymbol{v}_j^{n+1} - \boldsymbol{v}_j^n\big) + \left[\int_{t_n}^{t_{n+1}} \boldsymbol{F}\big(\boldsymbol{v}(x_{j+1/2}, t)\big)\,\mathrm{d}t - \int_{t_n}^{t_{n+1}} \boldsymbol{F}\big(\boldsymbol{v}(x_{j-1/2}, t)\big)\,\mathrm{d}t\right] = \boldsymbol{0},$$

which has an obvious physical interpretation: the difference in the amount of "matter" entering the volume $[t_n, t_{n+1}] \times [x_{j-1/2}, x_{j+1/2}]$ or exiting it (through the boundary $t = t_n$ or $t = t_{n+1}$) is balanced by the amount of "matter" entering or exiting through $x = x_{j-1/2}$ or $x = x_{j+1/2}$. Such an equation can be approximated by the difference scheme

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n - R\big[\boldsymbol{h}_{j+1/2}^n - \boldsymbol{h}_{j-1/2}^n\big], \tag{9.72}$$

where $R = \Delta t/\Delta x$. In this approach, the discrete approximations for the fluxes are given in terms of the *numerical flux functions*

$$\boldsymbol{h}_{j+1/2}^n = \boldsymbol{h}\big(\boldsymbol{u}_{k-p}^n, \ldots, \boldsymbol{u}_{k+q}^n\big), \tag{9.73}$$

$$\boldsymbol{h}_{j-1/2}^n = \boldsymbol{h}\big(\boldsymbol{u}_{k-p-1}^n, \ldots, \boldsymbol{u}_{k+q-1}^n\big), \tag{9.74}$$

which in general depend on values at $p + q + 1$ points. The scheme (9.72) is consistent with the conservation law (9.71) when $\boldsymbol{h}(\boldsymbol{v}, \ldots, \boldsymbol{v}) = \boldsymbol{F}(\boldsymbol{v})$.

Difference schemes that can be written as (9.72) with the flux functions (9.73) and (9.74) are called *conservative*. But this transcription by itself does not guarantee any of the appealing properties advertised in the introduction! A non-linear generalization of the linear Lax–Wendroff scheme (9.42),

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n - \frac{R}{2}\Delta_0^{(x)}\boldsymbol{F}_j^n + \frac{R^2}{2}\big[J\big(\boldsymbol{u}_{j+1/2}^n\big)\Delta_+^{(x)}\boldsymbol{F}_j^n - J\big(\boldsymbol{u}_{j-1/2}^n\big)\Delta_-^{(x)}\boldsymbol{F}_j^n\big], \tag{9.75}$$

where $[J(\boldsymbol{u})]_{ab} = \partial F_a/\partial u_b$, can be written in conservative form if we write

$$\boldsymbol{h}_{j+1/2}^n = \frac{1}{2}\big[\boldsymbol{F}_{j+1}^n + \boldsymbol{F}_j^n\big] - \frac{R}{2}J\big(\boldsymbol{u}_{j+1/2}^n\big)\Delta_+^{(x)}\boldsymbol{F}_j^n.$$

Similarly we write the expression for $\boldsymbol{h}_{j-1/2}^n$, we just replace j by $j-1$ everywhere. Since we do not know the solution at $x_{j+1/2}$ and $x_{j-1/2}$ we approximate

$$J\big(\boldsymbol{u}_{j+1/2}^n\big) \approx J\left(\frac{1}{2}\big(\boldsymbol{u}_{j+1}^n + \boldsymbol{u}_j^n\big)\right), \qquad J\big(\boldsymbol{u}_{j-1/2}^n\big) \approx J\left(\frac{1}{2}\big(\boldsymbol{u}_j^n + \boldsymbol{u}_{j-1}^n\big)\right).$$

Thus, the scheme is conservative and consistent with the conservation law (9.71), since $\boldsymbol{h}(\boldsymbol{u}, \boldsymbol{u}) = \boldsymbol{F}(\boldsymbol{u})$, but its solution are still influenced by dispersion. The Lax–Friedrichs scheme

$$\boldsymbol{u}_j^{n+1} = \frac{1}{2}\big(\boldsymbol{u}_{j+1}^n + \boldsymbol{u}_{j-1}^n\big) - \frac{R}{2}\Delta_0^{(x)}\boldsymbol{F}_j^n,$$

which corresponds to the flux functions

$$\boldsymbol{h}^n_{j+1/2} = \frac{1}{2}\big(\boldsymbol{F}^n_{j+1} + \boldsymbol{F}^n_j\big) - \frac{1}{2R}\big(\boldsymbol{u}^n_{j+1} - \boldsymbol{u}^n_j\big), \qquad \boldsymbol{h}^n_{j-1/2} = \boldsymbol{h}^n_{j+1/2}[j \leftrightarrow j-1],$$

is also conservative and consistent, but strong dissipation is present in its solutions.

9.12.1 High-Resolution Schemes

Dispersion and dissipation of the solutions of difference schemes used to approximate PDE in the form of conservation laws can be harnessed by using *high-resolution schemes*. From here on we discuss only scalar problems

$$v_t + \big[F(v)\big]_x = 0. \tag{9.76}$$

High-resolution schemes can also be formulated for systems of linear or non-linear equations in the general form (9.71), but this is beyond our scope. Further reading can be found in [6, 7].

The first requirement that we impose on the scheme is that the solution in subsequent time steps changes less and less, in the sense

$$\sum_{j=-\infty}^{\infty} \big|\Delta^{(x)}_+ u^{n+1}_j\big| \le \sum_{j=-\infty}^{\infty} \big|\Delta^{(x)}_+ u^n_j\big| \quad \forall n \ge 0. \tag{9.77}$$

Schemes that fulfill (9.77) are known as *total variation diminishing* (TVD), and their typical representatives are the *flux-limiter methods*. In these methods we include a special "smoothing" function in the scheme in order to introduce artificial dissipation that prevents non-physical oscillations of the solutions near discontinuities. Their upgrade are the *slope-limiter methods* in which one attempts to prevent too large solution slopes. Another option are the *modified-flux methods* where one does not modify the numerical flux function h but rather the function F appearing in the conservation law. Here we get to know only the first type of schemes.

High-Resolution Flux-Limiter Schemes The numerical flux functions of high-resolution flux-limiter schemes are constructed [5] as a combination of the flux function ${}^{\mathrm{L}}h$ belonging to a low-order scheme, and the function ${}^{\mathrm{H}}h$ belonging to a high-order scheme:

$$h^n_{j+1/2} = {}^{\mathrm{L}}h^n_{j+1/2} + \phi^n_j\big[{}^{\mathrm{H}}h^n_{j+1/2} - {}^{\mathrm{L}}h^n_{j+1/2}\big].$$

This allows us to enjoy the best of the two worlds: the admixture of the low-order scheme introduces dissipation into the composite scheme, helping us reduce unwanted oscillations, while the admixture of the high-order scheme improves the

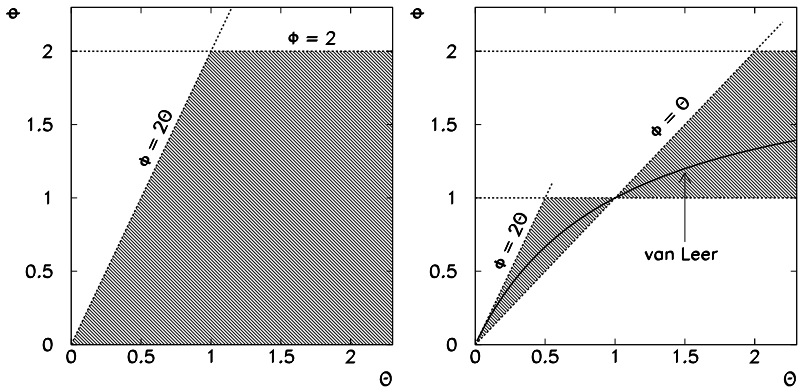


Fig. 9.11 [*Left*] The TVD region of the flux limiter $\phi(\theta)$ in a high-resolution scheme for the scalar conservation law $v_t + [F(v)]_x = 0$. [*Right*] Optimal region with the van Leer limiter drawn in. The schemes that utilize flux limiters that pass the point $(1, 1)$ are consistent with the conservation law to second order

accuracy. The ultimate success depends on the choice of the *flux-limiter function* ϕ_j^n, which is the weight of the anti-diffusive part of the flux,

$$^{\mathrm{H}}h_{j+1/2}^n - {}^{\mathrm{L}}h_{j+1/2}^n.$$

The limiter function to which we entrust the "smoothing" of the solution, is usually written as

$$\phi_j^n = \phi\big(\theta_j^n\big),$$

where θ_j^n is the *smoothing parameter* that can be computed from various components of the current solution.

It can be shown that a high-resolution scheme fulfills the TVD criterion (9.77) and is consistent with the conservation law to second order if the function $\phi(\theta)$ for positive θ satisfies the conditions

$$0 \le \frac{\phi(\theta)}{\theta} \le 2, \qquad 0 \le \phi(\theta) \le 2.$$

Figure 9.11 (left) shows the allowed region for the function $\phi(\theta)$. This basic requirement is met by many functions. The most frequently used are

$$\text{Superbee} \quad \phi(\theta) = \max\big\{0, \min\{1, 2\theta\}, \min\{\theta, 2\}\big\}, \tag{9.78}$$

$$\text{vanLeer} \quad \phi(\theta) = \big(|\theta| + \theta\big)/\big(|\theta| + 1\big), \tag{9.79}$$

$$\text{C/O} \quad \phi(\theta) = \max\big\{0, \min\{\theta, \psi\}\big\}, \tag{9.80}$$

$$\text{BW/LW} \quad \phi(\theta) = \max\big\{0, \min\{\theta, 1\}\big\}. \tag{9.81}$$

In the C/O function, $1 \le \psi \le 2$. The BW/LW limiter is a special case of C/O with $\psi = 1$. Functions whose graphs remain in the shaded area shown in Fig. 9.11 (right)

are preferable [8]. All forms enumerated above go through this region and also have the property $\phi(1) = 1$ which is crucial for second-order consistency [5]. The Superbee and van Leer flux limiters possess the additional symmetry

$$\frac{\phi(\theta)}{\theta} = \phi\left(\frac{1}{\theta}\right),$$

which is physically relevant: a high-resolution scheme with a symmetric flux-limiter function handles increasing and decreasing spatial gradients equivalently.

9.12.2 Linear Problem $v_t + cv_x = 0$

To form ${}^{\mathrm{L}}h$ for the scalar problem $v_t + cv_x = 0$, one uses a scheme that is insensitive to the sign of c, while to form ${}^{\mathrm{H}}h$, one takes the Lax–Wendroff scheme:

$$\begin{aligned} {}^{\mathrm{L}}h^n_{j+1/2} &= \frac{1}{2}c\big(u^n_j + u^n_{j+1}\big) - \frac{1}{2}|c|\big(u^n_{j+1} - u^n_j\big), \\ {}^{\mathrm{H}}h^n_{j+1/2} &= cu^n_j + \frac{1}{2}c(1 - cR)\Delta^{(x)}_+ u^n_j. \end{aligned} \tag{9.82}$$

(The method with the flux function (9.82) is known as the *upwind* scheme since it always differentiates in the direction opposite to the sense of advection: for $c < 0$ it turns into the FTFS scheme, and for $c > 0$ into the FTBS scheme.) The complete flux function is then

$$h^n_{j+1/2} = {}^{\mathrm{L}}h^n_{j+1/2} + \frac{1}{2}\phi^n_j c\big(\operatorname{sign}(c) - cR\big)\Delta^{(x)}_+ u^n_j,$$

and the corresponding high-resolution scheme becomes

$$\begin{aligned} u^{n+1}_j = u^n_j &- \frac{1}{2}cR\Delta^{(x)}_0 u^n_j + \frac{1}{2}|c|R\Delta^{(x)}_2 u^n_j \\ &- \frac{1}{2}cR\big(\operatorname{sign}(c) - cR\big)\big(\phi^n_j \Delta^{(x)}_+ u^n_j - \phi^n_{j-1}\Delta^{(x)}_- u^n_j\big). \end{aligned} \tag{9.83}$$

The smoothing parameters are computed as

$$\theta^n_j = \begin{cases} \Delta^{(x)}_- u^n_j / \Delta^{(x)}_+ u^n_j; & c > 0, \\ \Delta^{(x)}_+ u^n_{j+1} / \Delta^{(x)}_+ u^n_j; & c < 0. \end{cases}$$

Note that the differences in these formulas access points outside of the definition domain, and need to be handled separately. Trivial flux-limiter function ϕ do not bring anything new: $\phi = 0$ means just the FTFS or FTBS schemes, while $\phi = 1$ reproduces the Lax–Wendroff scheme. We therefore use a function from the set (9.78)–(9.81). How this works in practice is illustrated by Problem 9.13.2.

9.12.3 Non-linear Conservation Laws of the Form $v_t + [F(v)]_x = 0$

A non-linear conservation law obviously requires a non-linear high-resolution scheme. By analogy with the linear problem we use the upwind scheme for ${}^{\mathrm{L}}h$ and the non-linear Lax–Wendroff scheme to construct ${}^{\mathrm{H}}h$,

$$
\begin{aligned}
{}^{\mathrm{L}}h^n_{j+1/2} &= \frac{1}{2}\big(F^n_j + F^n_{j+1}\big) - \frac{1}{2}\big|a^n_{j+1/2}\big|\Delta^{(x)}_+ u^n_j, \\
{}^{\mathrm{H}}h^n_{j+1/2} &= \frac{1}{2}\big(F^n_j + F^n_{j+1}\big) - \frac{R}{2}\big(a^n_{j+1/2}\big)^2 \Delta^{(x)}_+ u^n_j.
\end{aligned}
\tag{9.84}
$$

The justification for this particular choice and many additional details on their properties can be found in [5]. We end up with the numerical flux function

$$
h^n_{j+1/2} = {}^{\mathrm{L}}h^n_{j+1/2} + \phi^n_j \big|a^n_{j+1/2}\big| \frac{1}{2}\big[1 - R\big|a^n_{j+1/2}\big|\big]\Delta^{(x)}_+ u^n_j,
\tag{9.85}
$$

where $\phi^n_j = \phi(\theta^n_j)$,

$$
a^n_{j+1/2} = \begin{cases} \Delta^{(x)}_+ F^n_j / \Delta^{(x)}_+ u^n_j; & \Delta^{(x)}_+ u^n_j \neq 0, \\ F'(u^n_j); & \Delta^{(x)}_+ u^n_j = 0, \end{cases}
\tag{9.86}
$$

and

$$
\theta^n_j = \begin{cases} \Delta^{(x)}_- u^n_j / \Delta^{(x)}_+ u^n_j; & a^n_{j+1/2} > 0, \\ \Delta^{(x)}_+ u^n_{j+1} / \Delta^{(x)}_+ u^n_j; & a^n_{j+1/2} < 0. \end{cases}
$$

9.13 Problems

9.13.1 Diffusion Equation

Use several difference methods to solve the linear diffusion equation

$$
v_t = D v_{xx}, \quad x \in [0, a],\ t > 0,\ D > 0,
$$

with specified initial and boundary conditions. The equation describes the heat transfer in an infinite slab of thickness a (or along a thin insulated wire).

⊙ Discretize the diffusion equation on the time-space mesh as described in Sect. 9.1. Solve the equation by using the explicit scheme (9.13) and pay attention to stability: keep $0 < r = D\Delta t/\Delta x^2 < 1/2$. Compare the solution at $r = 1/6$ to solutions at different r. The initial condition is: temperature zero everywhere except between $x = 0.2a$ and $x = 0.4a$, where the slab is at constant non-zero temperature

T_0. At time $t = 0$ we switch on additional heating between $x = 0.5a$ and $x = 0.75a$, with a heat release rate of $5T_0\lambda/a^2$. Plot the temperature profile $T(x,t) = v(x,t)$ at dimensionless times $Dt/a^2 = 0(0.3)3$!

Solve the diffusion equation by the implicit scheme (9.20) or the Crank–Nicolson scheme (9.21). Do you observe more freedom in choosing r? How does the choice of r influence the precision? What happens to the discrete "energy" (9.31) when you use the difference schemes (in stable and unstable regimes)?

⊕ Use the explicit scheme with local "freezing" of the function $D(v)$ described in Sect. 9.9 to solve the non-linear diffusion equation

$$v_t = \big(D(v)v_x\big)_x, \quad D(v) = \frac{1+3v^2}{1+v^2}, \ x \in [0,1],$$

with the initial condition $v(x,0) = \sin 3\pi x$ and boundary conditions $v(0,t) = v(1,t) = 0$. Use the discretization $N = 50$ ($\Delta x = 0.02$) and $\Delta t = 0.00005$, and monitor the solution until $T = 0.1$. By changing Δx and Δt confirm the stability criterion. Does the stability criterion remain valid for a different initial condition, e.g. $v(x,0) = 100x(1-x)|x-0.5|$? Write down and solve the corresponding implicit scheme. In all cases check whether (9.34) holds true.

9.13.2 Initial-Boundary Value Problem for $v_t + cv_x = 0$

We would like to solve numerically the first-order (hyperbolic) advection equation

$$v_t + cv_x = 0, \quad c = \pm 1,$$

on $x \in [0,1]$ with the initial condition $v(x,0) = \sin^{80}\pi x$ and periodic boundary condition $v(0,t) = v(1,t)$.

⊙ Solve the equation $v_t + v_x = 0$ by using the FTBS scheme (9.40) and the Lax–Wendroff scheme (9.42) with the discretization $N = 20$ and time step $\Delta t = 0.01$. Plot the solutions at $t = 0.0$, 0.1, 0.2, 0.4, and 1.0. Repeat the exercise with $N = 100$ and time step $\Delta t = 0.002$. Solve the equation $v_t - v_x = 0$ with the same discretization, but use the FTFS scheme (9.39) instead of FTBS. What happens to the amplitudes and phases of the solutions? (Compare them to analytic solutions at chosen times.) Discuss the solutions in the cases when the parameter R is near the stability limit (Table 9.2).

Repeat the exercise by using the Crank–Nicolson scheme (9.48). Pay attention to the correct treatment of the boundary conditions (periodicity) when you rewrite the system of difference equations in matrix form $F_1\boldsymbol{u}^{n+1} = \boldsymbol{r}^n$. (The matrix is cyclic tridiagonal.) Do you observe any peculiarities or limitations regarding the parameter R? (See Fig. 9.9.) Compare the amplitudes and phases of the solutions to those from the first part of the Problem. What happens when discretization is refined to, say, $\Delta x = 0.001$, $\Delta t = 0.0002$?

⊕ We put the difference schemes for hyperbolic PDE to a harsher test by including discontinuous initial conditions. We discuss the case $v_t - v_x = 0$ with a periodic boundary condition $v(0,t) = v(1,t)$ and a "step" initial condition

$$v(x,0) = \begin{cases} 1; & 0.4 \le x \le 0.6, \\ 0; & \text{otherwise.} \end{cases}$$

Use the same discretization as in the first part of the Problem and compute the solution by using the schemes: FTFS (9.39), Lax–Wendroff (9.42), Lax–Friedrichs (9.43), and Crank–Nicolson (9.48): the results should look like Fig. 9.10.

Finally, use the high-resolution scheme (9.83) with the flux-limiter function (9.79) and (9.78). If the formulas access the points x_{-1} or x_{N+1}, apply correctly the periodic boundary condition. If the expression for θ has a zero in the denominator, set $\theta = 0$. Use $(\Delta x, \Delta t) = (100, 0.002)$ and $(1000, 0.0002)$.

9.13.3 *Dirichlet Problem for a System of Non-linear Hyperbolic PDE*

We are interested in the solutions of a hyperbolic system (example from [1])

$$\boldsymbol{v}_t = A\boldsymbol{v}_x, \quad x \in [0,1],\ t > 0,$$

with the matrix

$$A = \frac{1}{3}\begin{pmatrix} 5 & 3 & 5 \\ -1 & -3 & -7 \\ 1 & -3 & 1 \end{pmatrix}.$$

⊙ Compute the matrices S and S^{-1} occurring in the similarity transformation $D = SAS^{-1}$ that diagonalizes A. Then solve the system by using the Lax–Wendroff scheme (9.69), with the initial condition

$$\boldsymbol{v}(x,0) = \left(\cos 4\pi x, (1-x)\cos \pi x, \mathrm{e}^{-x}\right)^{\mathrm{T}}$$

and discretization $N = 100$ ($\Delta x = 0.01$). From the eigenvalues of A it is clear that three boundary conditions may be assigned to this system, two at $x = 1$ and one at $x = 0$. We choose $v_1(1,t) = 1$, $v_2(1,t) = \sin 2t$, and $v_3(0,t) = \cos 2t$. Elsewhere, we prescribe numerical boundary conditions $u^n_{10} = u^n_{11}$, $u^n_{20} = u^n_{21}$, and $u^n_{3N} = u^n_{3N-1}$. Compute the solutions up to $t = 1.0$. Repeat the computation with the schemes (9.66) and (9.67) in the regime where they are expected to be stable.

Redo the calculation by using the initial condition

$$\boldsymbol{v}(x,0) = \left(\sin\frac{\pi}{2}x, \cos \pi x, x\mathrm{e}^{1-x}\right)^{\mathrm{T}}$$

and boundary conditions $v_1(0,t) = \sin 2t$, $v_1(1,t) = 1$, and $v_3(1,t) = \cos 2t$. Take $u^n_{20} = u^n_{21}$, $u^n_{2N} = u^n_{2N-1}$, and $u^n_{30} = u^n_{31}$ as numerical boundary conditions.

⊕ Repeat the exercise by using the flux-splitting scheme (9.68) without changing the spatial discretization or the length of the time step. Compare the results to those obtained in the first part of the Problem.

9.13.4 Second-Order and Fourth-Order Wave Equations

This Problem [4] deals with the one-dimensional second-order wave equation

$$v_{tt} = c^2 v_{xx}, \quad x \in [0, L],\ t > 0,\ c > 0,$$

describing the oscillations of a thin light string of length L with specified initial and boundary conditions. By substitution $x/L \to x$, $v/L \to v$, and $cT/L \to t$ we bring the equation to dimensionless form. The initial conditions are

$$v(x,0) = \frac{1}{2}x(1-x), \qquad v_t(x,0) = 0,$$

and the boundary solutions are $v(0,t) = v(1,t) = 0$. The analytic solution is

$$v(x,t) = \frac{2}{\pi^3}\sum_{k=1}^{\infty}\frac{1}{k^3}(1-\cos k\pi)\cos k\pi t \sin k\pi x.$$

⊙ Solve the wave equation by using the explicit difference scheme (9.50). Compute the deflections u^n_j after five oscillation periods. Pay attention to stability, which is ensured if $R = c\Delta t/\Delta x \le 1$. How do the results depend on the initialization scheme (Sect. 9.8.3)? Compare the numerical and analytic solutions, and find a good criterion for stopping the summation in the analytic solution. Test the implicit method (9.51) too.

Solve the Problem by treating the equation $v_{tt} = v_{xx}$ as a system of first-order hyperbolic equations. By denoting $p = u_x$ and $q = u_t$, it assumes a simple form $p_x - q_t = 0$, $q_x - p_t = 0$. To solve it, first use the difference scheme

$$\frac{p^n_{j+1} - p^n_{j-1}}{2\Delta x} = \frac{q^{n+1}_j - \frac{1}{2}(q^n_{j+1} + q^n_{j-1})}{\Delta t},$$
$$\frac{q^n_{j+1} - q^n_{j-1}}{2\Delta x} = \frac{p^{n+1}_j - \frac{1}{2}(p^n_{j+1} + p^n_{j-1})}{\Delta t}.$$

Along $t = 0$ we have $p = u_x = \frac{1}{2}(1-2x)$ and $q = u_t = 0$. Along $x = 0$ we have $u = 0$, thus $q = 0$ and $q_t = p_x = 0$. This provides the constraints for points outside of the mesh: $p^n_{-1} = p^n_1$, $p^n_{N+1} = p^n_{N-1}$, $q^n_{-1} = -q^n_1$, and $q^n_{N+1} = -q^n_{N-1}$. The solution at arbitrary time are the vectors $\boldsymbol{p}$ and $\boldsymbol{q}$. The deflections can then be computed

by simple quadrature, e.g. $\Delta u \approx \frac{1}{2}(p_j + p_{j+1})\Delta x$. Also solve the problem by the Lax–Wendroff method (9.42) in the form

$$p_j^{n+1} = p_j^n + \frac{R}{2}\left[q_{j+1}^n - q_{j-1}^n\right] + \frac{R^2}{2}\left[p_{j+1}^n - 2p_j^n + p_{j-1}^n\right],$$
$$q_j^{n+1} = q_j^n + \frac{R}{2}\left[p_{j+1}^n - p_{j-1}^n\right] + \frac{R^2}{2}\left[q_{j+1}^n - 2q_j^n + q_{j-1}^n\right].$$

Carefully determine the requirements at the boundary points x_0 and x_N as dictated by the scheme and the boundary conditions! What differences between the methods do you observe?

⊕ How would you approach the solution of the fourth-order wave equation

$$\left(c_1^2\frac{\partial^2}{\partial x^2} - \frac{\partial^2}{\partial t^2}\right)\left(c_2^2\frac{\partial^2}{\partial x^2} - \frac{\partial^2}{\partial t^2}\right)v = 0$$

with specified initial conditions for $v(x,0)$, $v_t(x,0)$, $v_{tt}(x,0)$, and $v_{ttt}(x,0)$ on $-\infty < x < \infty$? Such an equation describes the propagation of long shallow waves in an infinite channel with two fluids that do not mix [9]. In equilibrium (and at large distances from the waves) the fluid with density ρ_2 and height h_2 is at the bottom, and the fluid with density $\rho_1 < \rho_2$ and height h_1 on the top. The parameters are related by $c_1^2 + c_2^2 = g(h_1 + h_2)$ and $c_1^2 c_2^2 = g^2 h_1 h_2(1 - \rho_1/\rho_2)$. The solution $v(x,t)$ measures the deviation from the equilibrium height $h_1 + h_2$.

9.13.5 Burgers Equation

The one-dimensional Burgers equation

$$v_t + vv_x = Dv_{xx}$$

is widely used in modeling gas dynamics, acoustic phenomena, and turbulence. This is a mixed-type equation: for small values of D it is hyperbolic, while for large D its parabolic nature dominates. In this Problem we get to know the basic approaches to solving non-linear partial differential equations.

⊙ Solve the Burgers equation on $x \in [0,1]$ with the initial condition $v(x,0) = \sin 2\pi x$ and boundary conditions $v(0,t) = v(1,t) = 0$. Use several methods, e.g. the explicit one (see (9.55)), lagging the non-linear term (see (9.57)), or by expanding the current solution around the solution at the previous time step (see (9.59)). Use the discretization $N = 100$, $\Delta t = 0.007$, and the parameter $D = 0.1$. Compute the solution up to $T \approx 0.2$. Gradually decrease D. What do you observe? Try the discretization $N = 1000$, $\Delta t = 0.0007$ and reduce D down to $D \approx 0.0001$. Compute the solution at each time step of the implicit scheme (9.56) directly (without linearization) by using the multi-dimensional Newton method described in Sect. 9.9.

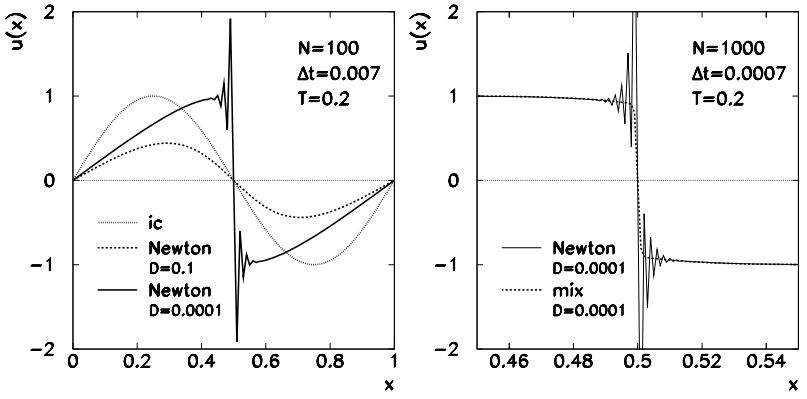


Fig. 9.12 Solutions of the Burgers equation. [*Left*] Initial condition (ic) and the solution by Newton's method with $D = 0.1$ and $D = 0.0001$ on a mesh with 100 subintervals. At small D a discontinuity forms that the scheme is unable to resolve. [*Right*] Solution by Newton's method and the mixed scheme (9.87) with $D = 0.0001$ by using a 10-times finer mesh and 10-times shorter time step (zoom-in on the discontinuity)

Use the generalization of the scheme for the linear problem $v_t + cv_x = Dv_{xx}$, with the usual notations $R = \Delta t/\Delta x$ and $r = D\Delta t/\Delta x^2$,

$$u_j^{n+1} = u_j^n - \frac{R}{2}u_j^n\Delta_0^{(x)}u_j^n + \left(r + \frac{R^2}{2}\left(u_j^n\right)^2\right)\Delta_2^{(x)}u_j^n.$$

In addition, try out the "mixed" scheme

$$u_j^{n+1} = u_j^n - Ru_j^n\left(\mathcal{D}^n u_j^n\right) + r\Delta_2^{(x)}u_j^n, \tag{9.87}$$

in which the diffusion term Dv_{xx} is approximated by the central difference, while the first difference in the non-linear term depends on the sign of the solution,

$$\left(\mathcal{D}^n u_j^n\right) = \begin{cases} u_{j+1}^n - u_j^n; & u_j^n < 0, \\ u_j^n - u_{j-1}^n; & u_j^n \geq 0. \end{cases}$$

$\bigoplus$ Solve the simplified equation $v_t + vv_x = 0$ (i.e. $D = 0$) for the same initial and boundary conditions by using the high-resolution scheme (9.85) with the flux limiters (9.79) and (9.78). Compare the solutions. The analytic solution is

$$v(x,t) = -2\sum_{k=1}^{\infty}\frac{J_k(-2\pi kt)}{2\pi kt}\sin(2\pi kx),$$

but it is only valid until $1/(2\pi)$ when the shock wave "breaks" (Fig. 9.12). Repeat the exercise on $x \in [-1, 1]$ with a discontinuous initial condition

$$v(x,0) = \begin{cases} 1.0; & x \leq 0, \\ 0.5; & x > 0. \end{cases}$$

9.13.6 The Shock-Tube Problem

Here we study the dynamics of a gas in an infinite tube, where a barrier at $x = 0$ initially separates two regions with different pressures and densities. We remove the barrier and study the time dependence of the density, pressure, and velocity of the gas. The system is described by three coupled non-linear PDE [1]:

$$\begin{aligned} \rho_t + (\rho v)_x &= 0, \\ (\rho v)_t + \left(\rho v^2 + p\right)_x &= 0, \\ E_t + \left[v(E+p)\right]_x &= 0, \end{aligned}$$

where ρ is the density, p is the pressure, v is the velocity of the gas, and E its total energy. The pressure and energy are related by the equation of state $p = (\kappa - 1)(E - \rho v^2/2)$, where $\kappa = c_\mathrm{p}/c_\mathrm{v} = 1.4$ is the specific heat ratio. For the solution we use the variables ρ, the momentum density $m = \rho v$, and E. The system can then be written in matrix form $\boldsymbol{v}_t = A\boldsymbol{v}_x$, where $\boldsymbol{v} = (\rho, m, E)^\mathrm{T}$ and

$$A = \begin{pmatrix} 0 & -1 & 0 \\ -\frac{\kappa-3}{2}\frac{m^2}{\rho^2} & (\kappa-3)\frac{m}{\rho} & -(\kappa-1) \\ \frac{m}{\rho^2}\left(\kappa E - (\kappa-1)\frac{m^2}{\rho}\right) & -\frac{1}{\rho}\left(\kappa E - \frac{3(\kappa-1)}{2}\frac{m^2}{\rho}\right) & -\kappa\frac{m}{\rho} \end{pmatrix}.$$

$\odot$ Solve the shock-tube problem on $x \in [-2, 2]$ with the boundary conditions $\rho_0 = \rho_1$, $m_0 = m_1$ and $E_0 = E_1$ at $x = -2$, $\rho_N = \rho_{N-1}$, $m_N = m_{N-1}$ and $E_N = E_{N-1}$ at $x = 2$, and the initial conditions

$$v(x,0) = 0, \qquad \rho(x,0) = p(x,0) = \begin{cases} 2; & -2 \le x < 0, \\ 1; & 0 \le x \le 2. \end{cases}$$

Compute the initial condition for E from those for p, ρ, and v, while the initial condition for $m = \rho v$ is $m(x,0) = 0$ (since $v(x,0) = 0$). Use the linearized Lax–Wendroff method (9.69) with the discretization $N = 400$ ($\Delta x = 0.01$) and compute the solution in steps of $\Delta t = 0.0025$ until $t = 1.0$.

$\oplus$ Use the non-linear variant of the Lax–Wendroff scheme (9.75) with the same discretization and compute the solution to $t = 1.0$. Pay attention to the signs in the term proportional to R: while (9.69) is written for systems of the form $\boldsymbol{v}_t = A\boldsymbol{v}_x$, (9.75) applies to $\boldsymbol{v}_t + [\boldsymbol{F}(\boldsymbol{v})]_x = F'(\boldsymbol{v})\boldsymbol{v}_x = 0$. Here you should therefore use $A = F'(\boldsymbol{v})$ and $\boldsymbol{F} = (\rho v, \rho v^2 + p, v(E+p))^\mathrm{T}$. Compare the propagation velocities of the shock waves. Do the results change if you extend the domain to $x \in [-8, 8]$ and trace the solution to $t = 4.0$? What about if you refine the discretization to $\Delta x = 0.001$, $\Delta t = 0.00025$? See also Fig. 9.13.

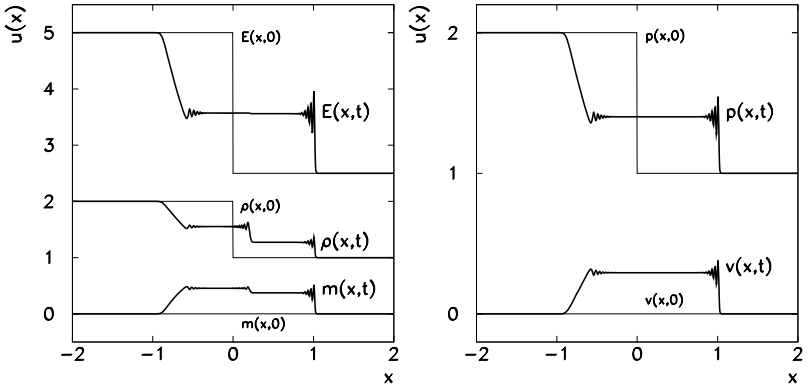


Fig. 9.13 Solution of the shock-tube problem by the non-linear Lax–Wendroff scheme with $\Delta t = 0.00125$ on a mesh with $N = 800$ subintervals. [*Left*] Spatial dependence of E, ρ, and m at $t = 0.75$. [*Right*] Spatial dependence of p and v at $t = 0.75$

9.13.7 Korteweg–de Vries Equation

Korteweg–de Vries equation [10] is a non-linear evolution equation useful in studying solitary waves (solitons). Solitons are waves that propagate with constant velocity and without change in form. In its original version, $v_t + cvv_x + \mu v_{xxx} = 0$, the equation is appropriate for the description of one-dimensional waves with small amplitudes in dispersive systems, like waves in shallow water or spilling over a submerged obstacle. The generalized Korteweg–de Vries equation

$$v_t + \frac{\beta}{\alpha+1}\left(v^{\alpha+1}\right)_x + \mu v_{xxx} = 0$$

with constant α, β, and μ has a much broader applicability [11]. Physically most relevant are the cases $\alpha = 1$ and $\alpha = 2$. In this Problem we analyze the simplest possibilities for the numerical solution of the generalized equation [12, 13].

$\odot$ We discretize the non-linear term $(v^{\alpha+1})_x$ and the dispersive term v_{xxx} at time $(n+1)\Delta t$ as

$$\left(v^{\alpha+1}\right)_x\big|_{x=x_j} \approx \frac{1}{2\Delta x}\left[\left(u_{j+1}^{n+1}\right)^{\alpha} u_{j+1}^{n+1} - \left(u_{j-1}^{n+1}\right)^{\alpha} u_{j-1}^{n+1}\right],$$

$$v_{xxx}|_{x=x_j} \approx \frac{1}{2\Delta x^3}\left[-u_{j-2}^{n+1} + 2u_{j-1}^{n+1} - 2u_{j+1}^{n+1} + u_{j+2}^{n+1}\right].$$

The equation is then linearized by using the approximation $(u_{j\pm1}^{n+1})^{\alpha} \approx (u_{j\pm1}^{n})^{\alpha}$. This discretization yields a (diagonally dominant) system of linear equations

$$-\frac{\mu\Delta t}{2\Delta x^3}u_{j-2}^{n+1}+\left(\frac{\mu\Delta t}{\Delta x^3}-\frac{\beta\Delta t\left(u_{j-1}^{n}\right)^{\alpha}}{2\Delta x(\alpha+1)}\right)u_{j-1}^{n+1}+u_{j}^{n+1}$$
$$+\left(-\frac{\mu\Delta t}{\Delta x^3}+\frac{\beta\Delta t\left(u_{j+1}^{n}\right)^{\alpha}}{2\Delta x(\alpha+1)}\right)u_{j+1}^{n+1}+\frac{\mu\Delta t}{2\Delta x^3}u_{j+2}^{n+1}=u_{j}^{n},$$

which has to be solved at each time step. (LAPACK/BLAS libraries contain optimized routines `SGBSV` (single) and `DGBSV` (double precision) for the solution of linear systems with banded matrices.) Use $\alpha=1$, $\beta=-6$, $\mu=1$, $\Delta x=0.1$, and $\Delta t=0.001$ with the initial condition $v(x,0)=-2k_x^2\operatorname{sech}^2 k_x x$, where $k_x=0.7$. The corresponding analytic solution is

$$v(x,t)=\left(\frac{2k_x^2(\alpha+1)(\alpha+2)}{\beta\alpha^2}\right)^{1/\alpha}\operatorname{sech}^{2/\alpha}k_x\left(x-\frac{4k_x^2}{\alpha}t\right).$$

Solve the equation numerically on the interval $x\in[-8,10]$ with the boundary conditions $u(-8,t)=u(10,t)=0$. Monitor the time evolution of the solution from the initial condition up to $t=1.0$. Also discuss the cases $\alpha=2$, $\alpha=3$, and $\alpha=4$. What is the influence of the (approximate) boundary conditions on the numerical error?

⊕ A slightly different discretization can be obtained by averaging the nonlinear and dispersion terms at times $n\Delta t$ and $(n+1)\Delta t$,

$$u_t+\frac{\beta}{\alpha+1}\frac{1}{2}\left[\left(u_j^{n+1}\right)^{\alpha+1}+\left(u_j^{n}\right)^{\alpha+1}\right]_x+\mu\frac{1}{2}\left[u_j^{n+1}+u_j^{n}\right]_{xxx}=0.$$

The equation can now be linearized by Taylor-expanding the expressions $(u_{j\pm1}^{n+1})^{\alpha+1}$ up to first order in time. This results in an implicit scheme which requires us to solve a pentadiagonal matrix system at each time step. Compare its solutions to those from the first part of the Problem.

9.13.8 Non-stationary Schrödinger Equation

The one-dimensional non-stationary Schrödinger equation

$$\left(\mathrm{i}\hbar\frac{\partial}{\partial t}-H\right)\psi(x,t)=0,\quad \psi(x,t_0)=\phi(x),$$

is the basic tool for a non-relativistic description of the time evolution of quantum states or wave packets in different potentials. Here we only discuss time-independent Hamiltonian operators

$$H=-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}+V(x).$$

(Time evolution involving time-dependent Hamiltonians is discussed in [14, 15].) The evolution of the state $\psi(x,t)$ to $\psi(x,t+\Delta t)$ is described by the approximation

$$\psi(x,t+\Delta t)=\mathrm{e}^{-\mathrm{i}H\Delta t/\hbar}\psi(x,t)\approx\frac{1-\frac{1}{2}\mathrm{i}H\Delta t/\hbar}{1+\frac{1}{2}\mathrm{i}H\Delta t/\hbar}\psi(x,t)+\mathcal{O}\big(\Delta t^3\big),$$

which is unitary. We compute the time evolution on the mesh $x_j=x_0+j\Delta x$ $(j=0,1,\ldots,N_x)$ at times $t_n=n\Delta t$ $(n=0,1,\ldots,N_t)$, approximating the spatial derivative by $\psi''(x)\approx(\psi(x+\Delta x)-2\psi(x)+\psi(x-\Delta x))/\Delta x^2$. The values of the wave-function and the potential at the mesh points at time t_n are denoted by $\psi(x_j,t_n)=\psi_j^n$ and $V(x_j)=V_j$, respectively, resulting in the equations

$$\begin{aligned}\psi_j^{n+1}-\frac{\mathrm{i}\hbar\Delta t}{4m\Delta x^2}\big[\psi_{j+1}^{n+1}-2\psi_j^{n+1}+\psi_{j-1}^{n+1}\big]+\frac{\mathrm{i}\Delta t}{2\hbar}V_j\psi_j^{n+1}\\=\psi_j^n+\frac{\mathrm{i}\hbar\Delta t}{4m\Delta x^2}\big[\psi_{j+1}^n-2\psi_j^n+\psi_{j-1}^n\big]-\frac{\mathrm{i}\Delta t}{2\hbar}V_j\psi_j^n.\end{aligned}$$

We arrange the values of the wave-function at x_j in the vector $\boldsymbol{\Psi}^n=(\psi_0^n,\ldots,\psi_{N_x}^n)^{\mathrm{T}}$ and rewrite the system in matrix form

$$A\boldsymbol{\Psi}^{n+1}=A^*\boldsymbol{\Psi}^n,\quad A=\begin{pmatrix}d_0&a&&&&\\a&d_1&a&&&\\&a&d_2&a&&\\&&\ddots&\ddots&\ddots&\\&&&a&d_{N_x-1}&a\\&&&&a&d_{N_x}\end{pmatrix},$$

where

$$b=\frac{\mathrm{i}\hbar\Delta t}{2m\Delta x^2},\qquad a=-\frac{b}{2},\qquad d_j=1+b+\frac{\mathrm{i}\Delta t}{2\hbar}V_j.$$

The matrix A and vector $\boldsymbol{\Psi}$ are complex, so the computation is best done in complex floating-point arithmetic. With the above approximation of the second derivative (which is of order $\mathcal{O}(\Delta x^2)$) we obtain a tridiagonal matrix. Had we used a difference of higher order, we would have obtained a banded matrix, but this would also improve the spatial precision: see [16, 17].

⊙ Compute the time evolution of the initial state

$$\psi(x,0)=\sqrt{\frac{\alpha}{\sqrt{\pi}}}\mathrm{e}^{-\alpha^2(x-\lambda)^2/2}$$

in the harmonic potential $V(x)=\frac{1}{2}kx^2$, where $\alpha=(mk/\hbar^2)^{1/4}$, $\omega=\sqrt{k/m}$. Set $\hbar=m=1$ and use $\omega=0.2$, $\lambda=10$. Place the spatial mesh on the interval $[x_0,x_{N_x}]=[-40,40]$ with $N_x=300$. The analytic solution is the coherent state

$$\psi(x,t)=\sqrt{\frac{\alpha}{\sqrt{\pi}}}\exp\bigg[-\frac{1}{2}(\xi-\xi_\lambda\cos\omega t)^2-\mathrm{i}\bigg(\frac{\omega t}{2}+\xi\xi_\lambda\sin\omega t-\frac{1}{4}\xi_\lambda^2\sin2\omega t\bigg)\bigg],$$

where $\xi=\alpha x$, $\xi_\lambda=\alpha\lambda$. In the chosen units the period is $T=10\pi$.

⊕ Compute the time evolution of the Gaussian wave packet

$$\psi(x,0) = \left(2\pi\sigma_0^2\right)^{-1/4} \mathrm{e}^{\mathrm{i}k_0(x-\lambda)} \mathrm{e}^{-(x-\lambda)^2/(2\sigma_0)^2}$$

in potential-free space ($V(x) = 0$). Set $\hbar = 1$, $m = 1/2$, $\sigma_0 = 1/20$, $k_0 = 50\pi$, $\lambda = 0.25$. The spatial interval is $[x_0, x_{N_x}] = [-0.5, 1.5]$ with $N_x = 2000$, and we choose $\Delta t = 2\Delta x^2$. Compute the solution until the center of the packet reaches $x \approx 0.75$. The analytic solution is

$$\psi(x,t) = \frac{(2\pi\sigma_0^2)^{-1/4}}{\sqrt{1 + \mathrm{i}\hbar t/(2m\sigma_0^2)}} \exp\left[\frac{-(x-\lambda)^2/(2\sigma_0)^2 + \mathrm{i}k_0(x-\lambda) - \mathrm{i}\hbar k_0^2 t/(2m)}{1 + \mathrm{i}\hbar t/(2m\sigma_0^2)}\right].$$

Section 1.2.2 tells you how the precision in the time variable can be improved.

9.13.9 Non-stationary Cubic Schrödinger Equation

The non-linear Schrödinger equation with a cubic non-linearity,

$$\mathrm{i}\frac{\partial\psi}{\partial t} + \frac{\partial^2\psi}{\partial x^2} + q|\psi|^2\psi = 0,$$

where q is a parameter [18], is used to describe numerous physical processes, like propagation of solitons in optical fibers, propagation of plasma waves, and even Bose–Einstein condensates. Splitting $\psi(x,t) = u(x,t) + \mathrm{i}v(x,t)$ turns the equation in a system of coupled non-linear PDE:

$$\frac{\partial u}{\partial t} + \frac{\partial^2 v}{\partial x^2} + q\left(u^2 + v^2\right)v = 0, \tag{9.88}$$

$$\frac{\partial v}{\partial t} - \frac{\partial^2 u}{\partial x^2} - q\left(u^2 + v^2\right)u = 0. \tag{9.89}$$

We are interested in the solutions of this system on $x \in [0, 2\pi]$ with the periodic boundary condition $\psi(x) = \psi(x + 2\pi)$. The spatial derivatives are approximated by differences of the form (9.6) with the discretization $x_j = j\Delta x$, $j = 0, 1, \ldots, N$, where $\Delta x = 2\pi/N$. The solution vectors are $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)^{\mathrm{T}}$, $\boldsymbol{v} = (v_1, v_2, \ldots, v_N)^{\mathrm{T}}$. When the boundary condition is accounted for, (9.88) and (9.89) can be written in canonical Hamiltonian form

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\boldsymbol{u}\\ \boldsymbol{v}\end{pmatrix} = \begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}\begin{pmatrix}\partial H/\partial\boldsymbol{u}\\ \partial H/\partial\boldsymbol{v}\end{pmatrix} = J^{-1}\begin{pmatrix}\partial H/\partial\boldsymbol{u}\\ \partial H/\partial\boldsymbol{v}\end{pmatrix},$$

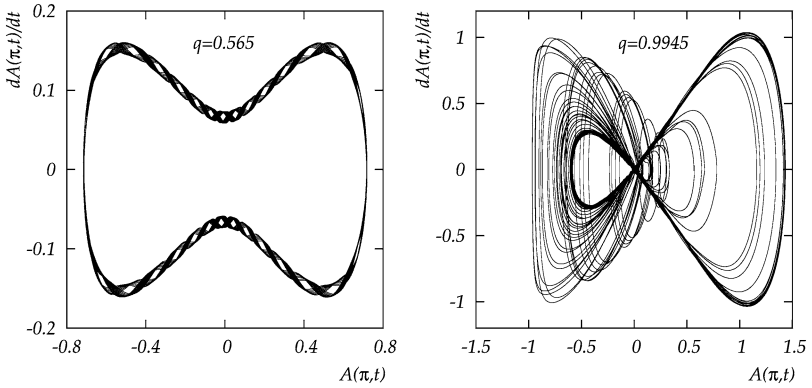


Fig. 9.14 Phase diagrams in solving the cubic Schrödinger equation where the solution at two values of the non-linearity parameter q is computed until time T. Shown is the amplitude $A(\pi,t) = |\psi(\pi,t)| - 1$ versus its time derivative $\mathrm{d}A(\pi,t)/\mathrm{d}t$. [*Left*] $q = 0.565$, $T = 400$. [*Right*] $q = 0.9945$, $T = 1000$

where $\partial H/\partial \boldsymbol{u} = (\partial H/\partial u_1, \partial H/\partial u_2, \ldots, \partial H/\partial u_N)^{\mathrm{T}}$ (analogously for $\partial H/\partial \boldsymbol{v}$), and

$$H(\boldsymbol{u},\boldsymbol{v}) = \frac{1}{2}(\boldsymbol{u}^{\mathrm{T}},\boldsymbol{v}^{\mathrm{T}}) \begin{pmatrix} F & 0 \\ 0 & F \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} + \frac{q}{4}\sum_{j=1}^{N}(u_j^2 + v_j^2)^2,$$

where

$$F = \frac{1}{\Delta x^2} \begin{pmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{pmatrix}.$$

One therefore needs to solve the system of equations $\dot{\boldsymbol{z}} = J^{-1}\nabla H = J^{-1}(\partial H/\partial \boldsymbol{z})$, where $\boldsymbol{z} = (\boldsymbol{u}^{\mathrm{T}}, \boldsymbol{v}^{\mathrm{T}})^{\mathrm{T}}$. This is best accomplished by the implicit symplectic Euler integrator

$$\boldsymbol{z}^{n+1} = \boldsymbol{z}^n + \Delta t\, J^{-1}(\nabla H)_{\boldsymbol{z}=\frac{1}{2}(\boldsymbol{z}^{n+1}+\boldsymbol{z}^n)}, \tag{9.90}$$

where the solution at each time step is computed iteratively.

⊙ By using (9.90) solve the non-linear Schrödinger equation with the initial condition $\psi(x,0) = 1 + \varepsilon \mathrm{e}^{\mathrm{i}\theta}\cos x$, where $\varepsilon = 0.1$ and $\theta = 45.18°$. Use the time step $\Delta t = 0.001$ and compute the solutions at a given parameter q until time T. Use the pairs of parameters $(q,T) = (0.01, 400)$, $(0.325, 400)$, $(0.565, 400)$, $(0.9825, 400)$, $(0.9945, 1000)$, $(1.0115, 1000)$, $(1.315, 1000)$, and $(1.630, 1000)$. Plot $A(\pi,t) = |\psi(\pi,t)| - 1$ and $A_t(\pi,t) = \mathrm{d}A(\pi,t)/\mathrm{d}t$ as functions of t, and the phase diagram $A_t(\pi,t)$ versus $A(\pi,t)$ (see examples in Fig. 9.14). You can use a simple approximation to compute the derivative A_t, e.g. $A_t \approx (A^{n+1} - A^n)/\Delta t$.

Plot the spectrum of the signals A and A_t (since you will be dealing with samples of several 100,000 points, use FFT). For the integration (9.90) set the tolerance for the difference of the current and previous iteration to less than $\approx 10^{-13}$. Otherwise the numerical errors in A and A_t will be too large.

⊕ Plot the regions $|\psi(x,t)|^2 = \text{const}$. When computing with parameters q at which you have observed irregular behavior, you may restrict the plots to the part of the interval $[0, T]$ with the most interesting structure.

References

1. J.W. Thomas, *Numerical Partial Differential Equations: Finite Difference Methods*. Springer Texts in Applied Mathematics, vol. 22 (Springer, Berlin, 1998)
2. D. Knoll, J. Morel, L. Margolin, M. Shashkov, Physically motivated discretization methods. Los Alamos Sci. **29**, 188 (2005)
3. A. Tveito, R. Winther, *Introduction to Partial Differential Equations*. Springer Texts in Applied Mathematics, vol. 29 (Springer, Berlin, 2005)
4. G.D. Smith, *Numerical Solution of Partial Differential Equations* (Oxford University Press, Oxford, 2003)
5. J.W. Thomas, *Numerical Partial Differential Equations: Conservation Laws and Elliptic Equations*. Springer Texts in Applied Mathematics, vol. 33 (Springer, Berlin, 1999)
6. E. Godlewski, P.-A. Raviart, *Numerical Approximations of Hyperbolic Systems of Conservation Laws* (Springer, Berlin, 1996)
7. R.J. LeVeque, *Numerical Methods for Conservation Laws* (Birkhäuser, Basel, 1990)
8. P.K. Sweby, High resolution schemes using flux limiters for hyperbolic conservation laws. SIAM J. Numer. Anal. **21**, 995 (1984)
9. D. Eisen, On the numerical analysis of a fourth order wave equation. SIAM J. Numer. Anal. **4**, 457 (1967)
10. D.J. Korteweg, G. de Vries, On the change of form of long waves advancing in a rectangular channel and on a new type of long stationary wave. Philos. Mag. Ser. 5 **39**, 422 (1895)
11. B. Fornberg, G.B. Whitham, A numerical and theoretical study of certain nonlinear wave phenomena. Philos. Trans. R. Soc. Lond. Ser. A **289**, 373 (1978)
12. X. Lai, Q. Cao, Some finite difference methods for a kind of GKdV equations. Commun. Numer. Methods Eng. **23**, 179 (2007)
13. Q. Cao, K. Djidjeli, W.G. Price, E.H. Twizell, Computational methods for some non-linear wave equations. J. Eng. Math. **35**, 323 (1999)
14. K. Kormann, S. Holmgren, O. Karlsson, Accurate time propagation for the Schrödinger equation with an explicitly time-dependent Hamiltonian. J. Chem. Phys. **128**, 184101 (2008)
15. C. Lubich, in *Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms*, ed. by J. Grotendorst, D. Marx, A. Muramatsu. NIC Series, vol. 10 (John von Neumann Institute for Computing, Jülich, 2002), p. 459
16. W. van Dijk, F.M. Toyama, Accurate numerical solutions of the time-dependent Schrödinger equation. Phys. Rev. E **75**, 036707 (2007)
17. T.N. Truong et al., A comparative study of time dependent quantum mechanical wave packet evolution methods. J. Chem. Phys. **96**, 2077 (1992)
18. X. Liu, P. Ding, Dynamic properties of cubic nonlinear Schrödinger equation with varying nonlinear parameter. J. Phys. A, Math. Gen. **37**, 1589 (2004)

Chapter 10
Difference Methods for PDE in Several Dimensions

10.1 Parabolic and Hyperbolic PDE

The basic concepts of difference methods for PDE in several dimensions are readily adopted from the discussion of one-dimensional initial-boundary-value problems (Chap. 9). We are seeking consistent, stable difference schemes, and corresponding discretizations of the initial and boundary conditions by which we obtain convergence of the numerical solution to the exact solution of the differential equation. Except in (10.21) and (10.22) we restrict the discussion to PDE with time dependence in two spatial coordinates.

10.1.1 Parabolic Equations

The basic model problem for two-dimensional parabolic PDE is the initial-boundary-value problem for the diffusion equation with an inhomogeneous term,

$$v_t = D\nabla^2 v + Q = D(v_{xx} + v_{yy}) + Q(x, y, t),$$
$$v(x, y, 0) = f(x, y),$$

which we solve for $t > 0$ in a region with a simple geometry, e.g. on a square $(x, y) \in [0, 1] \times [0, 1]$. More complicated geometries are mentioned in Sect. 10.4. Let us ignore the issue of boundary conditions for the moment. By analogy with the one-dimensional case (Fig. 9.1) we discretize the time axis and both spatial axes.

The exact solution $v(x_j, y_k, t_n) = v^n_{jk}$ at $(j\Delta x, k\Delta y)$ and time $n\Delta t$ corresponds to the numerical solution u^n_{jk}, where $j = 0, 1, \ldots, N_x$ and $k = 0, 1, \ldots, N_y$ (Fig. 10.1 (left), the time axis is perpendicular to the page). The most obvious bookkeeping choice for the vector of approximate solutions $\boldsymbol{u}$ is

$$\boldsymbol{u}^n = \left(u^n_{00}, u^n_{10}, \ldots, u^n_{N_x 0}, u^n_{01}, u^n_{11}, \ldots, u^n_{N_x 1}, \ldots, u^n_{0N_y}, u^n_{1N_y}, \ldots, u^n_{N_x N_y}\right)^{\mathrm{T}}. \tag{10.1}$$

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9_10, © Springer-Verlag Berlin Heidelberg 2012

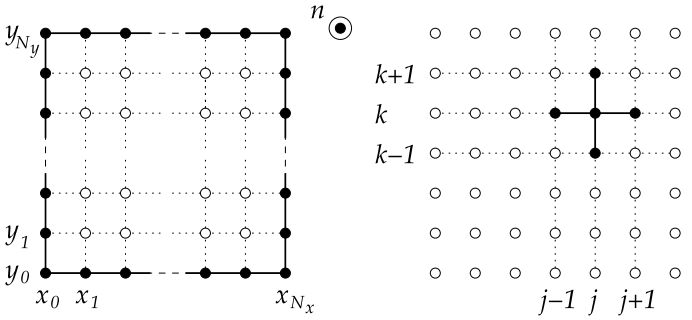


Fig. 10.1 [*Left*] The uniform mesh for the solution of the initial-boundary-value problem for a PDE with one time and two space dimensions on $[x_0, x_{N_x}] \times [y_0, y_{N_y}] = [0, 1] \times [0, 1]$. The discretization Δt, $\Delta x = 1/N_x$, $\Delta y = 1/N_y$ defines the mesh points $(x_j, y_k, t_n) = (j\Delta x, k\Delta y, n\Delta t)$ and in each of them the exact solution v^n_{jk} and its approximate value u^n_{jk}. [*Right*] The stencil of mesh points for the evaluation of $(v_{xx} + v_{yy})_{jk}$

Next, we adapt the formulas for the differences to two dimensions. The obvious generalizations of the approximation for the second derivative (9.6) are

$$\left.\frac{\partial^2 v}{\partial x^2}\right|_{(j\Delta x, k\Delta y, n\Delta t)} = \frac{v^n_{j+1k} - 2v^n_{jk} + v^n_{j-1k}}{\Delta x^2} + \mathcal{O}\big(\Delta x^2\big), \tag{10.2}$$

$$\left.\frac{\partial^2 v}{\partial y^2}\right|_{(j\Delta x, k\Delta y, n\Delta t)} = \frac{v^n_{jk+1} - 2v^n_{jk} + v^n_{jk-1}}{\Delta y^2} + \mathcal{O}\big(\Delta y^2\big), \tag{10.3}$$

which need to be evaluated in the characteristic *stencil* of mesh points (Fig. 10.1 (right)). Instead of one auxiliary notation (9.11) we now have two,

$$\Delta_2^{(x)} = u^n_{j+1k} - 2u_{jk} + u^n_{j-1k}, \qquad \Delta_2^{(y)} = u^n_{jk+1} - 2u_{jk} + u^n_{jk-1}. \tag{10.4}$$

10.1.2 Explicit Scheme

The two-dimensional generalization of the explicit FTCS scheme (see (9.13)) is

$$u^{n+1}_{jk} = u^n_{jk} + \big(r_x\Delta_2^{(x)} + r_y\Delta_2^{(y)}\big)u^n_{jk} + \Delta t q^n_{jk}, \tag{10.5}$$

where $r_x = D\Delta t/\Delta x^2$, $r_y = D\Delta t/\Delta y^2$, and $Q(x_j, y_k, t_n) = q^n_{jk}$. By analogy to the one-dimensional variant we conclude that the discretization error of the scheme is $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta y^2)$, and that the scheme can be written in matrix form

$$\boldsymbol{u}^{n+1} = F\boldsymbol{u}^n + \boldsymbol{r}^n + \Delta t\boldsymbol{q}^n.$$

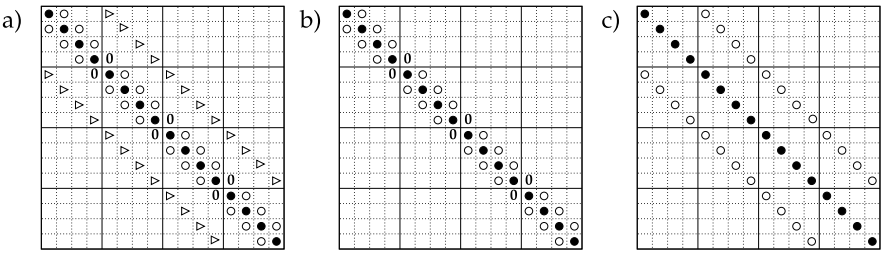


Fig. 10.2 Common structures of matrices appearing in explicit ($\boldsymbol{u}^{n+1} = F\boldsymbol{u}^n$) and implicit schemes ($F_1\boldsymbol{u}^{n+1} = F\boldsymbol{u}^n$) for the solution of two-dimensional diffusion equation. The values of the elements denoted by • / ○ / ▷ depend on the difference method (see text). Shown are matrices for the case $N_x = N_y = 5$

The inhomogeneous term (the last two terms at the right) has two contributions. The vector $\boldsymbol{r}^n$ contains the requirements at the boundary: for example, for homogeneous Dirichlet boundary conditions, the value u_{jk} on all boundaries of the region $[x_0, x_{N_x}] \times [y_0, y_{N_y}]$ is zero, and then $\boldsymbol{r}^n = 0$. The second part, the vector $\Delta t\boldsymbol{q}^n$, describes the sources. In practical problems the reader will have to figure out the structure of these terms for the chosen scheme and boundary conditions.

Let us inspect the typical cases. Dirichlet boundary conditions (the values of the function v specified on all four sides of the square) are expressed as

$$u^n_{0k} = f(0, k\Delta y, n\Delta t), \tag{10.6}$$

$$u^n_{N_x k} = f(1, k\Delta y, n\Delta t), \tag{10.7}$$

$$u^n_{j0} = f(j\Delta x, 0, n\Delta t), \tag{10.8}$$

$$u^n_{jN_y} = f(j\Delta x, 1, n\Delta t). \tag{10.9}$$

In this case the solution can be represented by a shorter vector $\boldsymbol{u}$, since all values at $j = 0$, $j = N_x$, $k = 0$, and $k = N_y$ are fixed by the boundary conditions. Instead of with (10.1) we work with a $(N_x - 1)(N_y - 1)$-dimensional vector

$$\boldsymbol{u}^n = \left(u^n_{11}, u^n_{21}, \ldots, u^n_{N_x-11}, \ldots, u^n_{1N_y-1}, u^n_{2N_y-1}, \ldots, u^n_{N_x-1N_y-1}\right)^{\mathrm{T}}. \tag{10.10}$$

The matrix F therefore has the size $(N_x - 1)(N_y - 1) \times (N_x - 1)(N_y - 1)$. If $N_x = N_y$ (which is not obligatory) it has the form shown in Fig. 10.2 (case (a)), with the elements $\bullet = 1 - 2r_x - 2r_y$, $\circ = r_x$, and $\triangleright = r_y$. Here we realize the practical constraints brought about by the additional space dimension: even a modest spatial discretization in $N_x = N_y = 100$ points in each coordinate implies manipulations of $(N_x - 1)(N_y - 1) \times (N_x - 1)(N_y - 1) \approx 10000 \times 10000$ matrices without simple structure (e.g. not tridiagonal). This becomes relevant in implicit schemes where we need to solve matrix systems of such sizes.

For Dirichlet boundary conditions we use the discrete Fourier transformation (by analogy to the one-dimensional case) to derive the sufficient condition for stability

of (10.5). We extend the discrete Fourier mode (9.29) to two dimensions,

$$u_{jk}^n = \xi^n \mathrm{e}^{\mathrm{i}jp\pi\Delta x} \mathrm{e}^{\mathrm{i}kq\pi\Delta y},$$

insert it into the homogeneous part of the difference equation (10.5), and divide the resulting equation by ξ^{n+1}, whence

$$\xi = 1 - 4r_x \sin^2 \frac{p\pi\Delta x}{2} - 4r_y \sin^2 \frac{q\pi\Delta y}{2}.$$

The discrete Neumann criterion $|\xi| \le 1$ means

$$r_x + r_y \le 1/2.$$

The stability condition for the two-dimensional scheme (10.5) is therefore more restrictive than in the one-dimensional case (first row of Table 9.1). If we use the uniform mesh, $\Delta x = \Delta y$, such a condition requires $r_x = r_y \le 1/4$ and forces us to take a twice smaller Δt (or $\sqrt{2}$-times larger Δx) to ensure stability.

Neumann conditions, as usual, require special care. If we have three Dirichlet conditions, e.g. $v(1, y, t) = v(x, 0, t) = v(x, 1, t) = 0$, and one Neumann, $v_x(0, y, t) = g(y, t)$, the joint order of the difference scheme depends on the order of the approximation for the Neumann condition. If the latter is discretized to first order, as in (9.17),

$$u_{1k}^n = u_{0k}^n + \Delta x g(k\Delta y, n\Delta t), \tag{10.11}$$

the FTCS scheme is only of order $\mathcal{O}(\Delta x)$. We retrieve the order $\mathcal{O}(\Delta x^2)$ if the Neumann condition is discretized to second order. Confirm that in this case

$$u_{0k}^{n+1} = u_{0k}^n + 2r_x\left(u_{1k}^n - u_{0k}^n\right) - 2r_x\Delta x g(k\Delta y, n\Delta t) + r_y\Delta_2^{(y)}u_{0k}^n, \tag{10.12}$$

by analogy to (9.18)! The stability condition for the scheme with Neumann conditions is just a necessary one, since the mapping matrix is not symmetric and we need to evaluate its spectral radius by Gershgorin's theorem [1].

10.1.3 Crank–Nicolson Scheme

The two-dimensional generalization of the Crank–Nicolson scheme (see (9.21)) is

$$\left(1 - \frac{r_x}{2}\Delta_2^{(x)} - \frac{r_y}{2}\Delta_2^{(y)}\right)u_{jk}^{n+1}$$
$$= \left(1 + \frac{r_x}{2}\Delta_2^{(x)} + \frac{r_y}{2}\Delta_2^{(y)}\right)u_{jk}^n + \frac{\Delta t}{2}\left(q_{jk}^n + q_{jk}^{n+1}\right). \tag{10.13}$$

The scheme has order $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta y^2)$, and can be written as

$$F_1 \boldsymbol{u}^{n+1} = F\boldsymbol{u}^n + \boldsymbol{r}^n + \frac{\Delta t}{2}\left(\boldsymbol{q}^n + \boldsymbol{q}^{n+1}\right),$$

where (for Dirichlet conditions (10.6)–(10.9)) the matrix F_1 has the form as in Fig. 10.2 (case (a)), with the elements $\bullet = 1 + r_x + r_y$, $\circ = -r_x/2$, and $\triangleright = -r_y/2$, while F has the same form except that all signs of r_x and r_y are flipped. The term $\boldsymbol{r}^n$ includes the boundary conditions at $j = 0$, $j = N_x$, $k = 0$, and $k = N_y$ that are not included in F_1 or F. The discrete Neumann criterion for stability is again derived by Fourier analysis as for the explicit scheme. For the symbol

$$\xi = \frac{1 - 2r_x \sin^2 p\pi\,\Delta x/2 - 2r_y \sin^2 q\pi\,\Delta y/2}{1 + 2r_x \sin^2 p\pi\,\Delta x/2 + 2r_y \sin^2 q\pi\,\Delta y/2}$$

we obviously get $|\xi| \leq 1$ for all non-negative r_x and r_y, so the two-dimensional Crank–Nicolson scheme is absolutely stable (the conclusion applies to Dirichlet and Neumann boundary conditions). As an exercise, generalize the implicit scheme (9.20) to two dimensions and check that its amplification factor is

$$\xi = \frac{1}{1 + 4r_x \sin^2 p\pi\,\Delta x/2 + 4r_y \sin^2 q\pi\,\Delta y/2}.$$

The two-dimensional scheme is absolutely stable since $|\xi| \leq 1$.

10.1.4 Alternating Direction Implicit Schemes

When implicit schemes discussed above (either in two or three space dimensions) are written in a matrix representation, the matrices become sparse. If the matrices are very large, the numerical solution of the corresponding matrix system may become too costly, even if one resorts to iterative methods (e.g. Gauss–Seidel, Jacobi, SOR, or multi-grid; see Sects. 10.2 and 10.4). The troublesome solution of such systems can be circumvented by *Alternating Direction Implicit* (ADI) schemes. In a two-dimensional ADI scheme we compute the spatial derivatives in one dimension explicitly, and implicitly in the other (or vice versa). This enables us to benefit from the good stability properties of implicit schemes, while keeping the tridiagonal structure of the matrix system.

Peaceman–Rachford Scheme A typical ADI method is the Peaceman–Rachford scheme: in the first half of the time step Δt we compute the spatial derivative implicitly in coordinate x and explicitly in coordinate y, and vice versa in the second half of Δt. This can be expressed as a two-level scheme

$$\left(1 - \frac{r_x}{2}\Delta_2^{(x)}\right) u_{jk}^{n+\frac{1}{2}} = \left(1 + \frac{r_y}{2}\Delta_2^{(y)}\right) u_{jk}^n + \frac{\Delta t}{2} q_{jk}^n, \tag{10.14}$$

$$\left(1-\frac{r_y}{2}\Delta_2^{(y)}\right)u_{jk}^{n+1}=\left(1+\frac{r_x}{2}\Delta_2^{(x)}\right)u_{jk}^{n+\frac{1}{2}}+\frac{\Delta t}{2}q_{jk}^{n+1} \tag{10.15}$$

which is absolutely stable. (Optionally, the q_{jk} terms in the inhomogeneous part may be evaluated at the same time, $(n+1/2)\Delta t$.) Moreover, the order of spatial derivatives in (10.14) and (10.15) can also be reversed: in this case we simply rewrite $x \longleftrightarrow y$. The scheme is of order $\mathcal{O}(\Delta t^2)+\mathcal{O}(\Delta x^2)+\mathcal{O}(\Delta y^2)$, but only if boundary conditions are treated consistently. For example, if we have a Dirichlet condition $v(0,y,t)=f(y,t)$ at $x=0$ ($j=0$), we do not spoil the nominal order of the scheme if the boundary condition at the intermediate time $(n+\frac{1}{2})\Delta t$ is computed as

$$\begin{aligned}u_{0k}^{n+\frac{1}{2}}&=\frac{1}{2}\left(1-\frac{r_y}{2}\Delta_2^{(y)}\right)f\big(0,k\Delta y,(n+1)\Delta t\big)\\&\quad+\frac{1}{2}\left(1+\frac{r_y}{2}\Delta_2^{(y)}\right)f(0,k\Delta y,n\Delta t).\end{aligned} \tag{10.16}$$

If the condition is time-independent, it clearly simplifies to $u_{0k}^{n+\frac{1}{2}}=f(0,k\Delta y)$. For the corresponding condition at $x=1$ ($j=N_x$) we compute $f(1,k\Delta y,\ldots)$ instead of $f(0,k\Delta y,\ldots)$; the discretization of the analogous conditions at $y=0$ ($k=0$) and $y=1$ ($k=N_y$) should be done by the reader as an exercise. (See Problem 10.9.1 which also involves Neumann boundary conditions.)

The steps (10.14) and (10.15) can be written in matrix form

$$F_1\boldsymbol{u}^{n+\frac{1}{2}}=F\boldsymbol{u}^n+\boldsymbol{r}_{(y)}^n+\boldsymbol{r}_{(x)}^{n+\frac{1}{2}}+\frac{\Delta t}{2}\boldsymbol{q}^n, \tag{10.17}$$

$$F_1'\boldsymbol{u}^{n+1}=F'\boldsymbol{u}^{n+\frac{1}{2}}+\boldsymbol{r}_{(x)}^{n+\frac{1}{2}}+\boldsymbol{r}_{(y)}^{n+1}+\frac{\Delta t}{2}\boldsymbol{q}^{n+1}. \tag{10.18}$$

The terms $\boldsymbol{r}_{(x)}$ and $\boldsymbol{r}_{(y)}$ contain the boundary conditions along x and y at times $n\Delta t$ and $(n+1/2)\Delta t$. The matrix F_1 is tridiagonal, as in Fig. 10.2 (case (b)), with the elements $\bullet=1+r_x$, $\circ=-r_x/2$, while the matrix F has the form shown in Fig. 10.2 (case c)), with the elements $\bullet=1-r_y$, $\circ=r_y/2$. We get F_1' from F by replacing $r_y\to-r_y$, and F' from F_1 by replacing $r_x\to-r_x$.

In each time step (10.17)–(10.18) we need to solve an $(N_x-1)(N_y-1)\times(N_x-1)(N_y-1)$ matrix system in which F_1' is not tridiagonal. But by reordering the solution components as $u_{jk}\to u_{kj}$, the system (10.18) becomes tridiagonal. The whole procedure then decouples to the solution of N_y-1 systems of size $(N_x-1)\times(N_x-1)$ in the first time step, and N_x-1 systems of size $(N_y-1)\times(N_y-1)$ in the second step:

$$F_x\boldsymbol{u}_k^{n+\frac{1}{2}}=\boldsymbol{r}_k+\frac{\Delta t}{2}\boldsymbol{q}_k^n,\quad k=1,2,\ldots,N_y-1, \tag{10.19}$$

$$F_y\boldsymbol{u}_j^{n+1}=\boldsymbol{r}_j+\frac{\Delta t}{2}\boldsymbol{q}_j^{n+1},\quad j=1,2,\ldots,N_x-1, \tag{10.20}$$

where the matrix F_x (or F_y) is tridiagonal, with the values $1+r_x$ (or $1+r_y$) on the main diagonal and $-r_x/2$ (or $-r_y/2$) on the subdiagonals. In the solution vectors $\boldsymbol{u}$ of dimension (N_x-1) (or (N_y-1)) we should pay attention to the ordering of components,

$$\begin{aligned}\boldsymbol{u}_k &= (u_{1k}, u_{2k}, \ldots, u_{N_x-1k})^{\mathrm{T}},\\ \boldsymbol{u}_j &= (u_{j1}, u_{j2}, \ldots, u_{jN_y-1})^{\mathrm{T}},\end{aligned}$$

and analogously for $\boldsymbol{q}_k$ and $\boldsymbol{q}_j$. The vectors containing the boundary conditions are

$$\boldsymbol{r}_k = \begin{pmatrix} (1+\frac{r_y}{2}\Delta_2^{(y)})u_{1k}^n + \frac{r_x}{2}u_{0k}^{n+\frac{1}{2}} \\ (1+\frac{r_y}{2}\Delta_2^{(y)})u_{2k}^n \\ \vdots \\ (1+\frac{r_y}{2}\Delta_2^{(y)})u_{N_x-2k}^n \\ (1+\frac{r_y}{2}\Delta_2^{(y)})u_{N_x-1k}^n + \frac{r_x}{2}u_{N_xk}^{n+\frac{1}{2}} \end{pmatrix},$$

$$\boldsymbol{r}_j = \begin{pmatrix} (1+\frac{r_x}{2}\Delta_2^{(x)})u_{j1}^{n+\frac{1}{2}} + \frac{r_y}{2}u_{j0}^{n+1} \\ (1+\frac{r_x}{2}\Delta_2^{(x)})u_{j2}^{n+\frac{1}{2}} \\ \vdots \\ (1+\frac{r_x}{2}\Delta_2^{(x)})u_{jN_y-2}^{n+\frac{1}{2}} \\ (1+\frac{r_x}{2}\Delta_2^{(x)})u_{jN_y-1}^{n+\frac{1}{2}} + \frac{r_y}{2}u_{jN_y}^{n+1} \end{pmatrix}.$$

The columns $\boldsymbol{r}_k$ (dimension N_x-1) and $\boldsymbol{r}_j$ (dimension N_y-1) contain expressions evaluated at times $n\Delta t$, $(n+1/2)\Delta t$, and $(n+1)\Delta t$: they are either boundary conditions u_{0k} and u_{N_xk}, computed at intermediate time $(n+1/2)\Delta t$ by using (10.16), boundary conditions u_{j0} and u_{jN_y} at time $(n+1)\Delta t$, or the solution values u_{jk} that we know from the previous step.

The key advantage of the decomposition of large matrix systems (10.17) and (10.18) to a multitude of smaller ones becomes clear at large N_x and N_y (solving systems with large matrices F_1 and F_1'). The subsystems of equations in (10.19) and (10.20) are independent: the solution of (10.19) is valid at $y_k = k\Delta y$ and it does not depend on values with other indices k, except those on the right-hand side of the equation, which is known from the previous time step. (Analogously for (10.20) at $x_j = j\Delta x$.) This decomposition can be exploited in vector and parallel-processing implementations.

D'yakonov Form The Peaceman–Rachford scheme can be expressed in a form that leads to a more efficient numerical implementation [1],

$$\left[1-\frac{r_x}{2}\Delta_2^{(x)}\right]u^\star_{jk}=\left[1+\frac{r_x}{2}\Delta_2^{(x)}\right]\left[1+\frac{r_y}{2}\Delta_2^{(y)}\right]u^n_{jk}+\frac{\Delta t}{2}\big(q^n_{jk}+q^{n+1}_{jk}\big),$$
$$\left[1-\frac{r_y}{2}\Delta_2^{(y)}\right]u^{n+1}_{jk}=u^\star_{jk}.$$

Its main feature is that the second step does not involve the inhomogeneous term q. Of course we still have to compute the boundary conditions corresponding to the equation for the intermediate value $u^\star_{jk}$, but we must be careful not to spoil the order of the methods by discretizing these conditions inappropriately. For example, if we have a Dirichlet boundary condition at $x=0$, the overall order of the scheme will remain $\mathcal{O}(\Delta t^2)+\mathcal{O}(\Delta x^2)+\mathcal{O}(\Delta y^2)$ if we evaluate the boundary condition as

$$u^\star_{0k}=-\frac{r_y}{2}f\big(0,(k+1)\Delta y,(n+1)\Delta t\big)+(1+r_y)f\big(0,k\Delta y,(n+1)\Delta t\big)$$
$$-\frac{r_y}{2}f\big(0,(k-1)\Delta y,(n+1)\Delta t\big)$$

(and similarly for other domain boundaries). In Problem 10.9.1 you can find the expressions for Neumann boundary conditions to first and second order.

10.1.5 Three Space Dimensions

Dealing with PDE in one time and three space dimensions is beyond our scope, but by using our knowledge accumulated so far it is easy to write at least the simplest schemes. An explicit scheme for the three-dimensional diffusion equation in Cartesian coordinates, $v_t=D\nabla^2 v+Q(x,y,z,t)$, is at hand: when we organize the numerical approximations u^n_{jkl} of the exact solution $v(x_j,y_k,z_l,t_n)=v(j\Delta x,k\Delta y,l\Delta z,n\Delta t)$ in a vector $\boldsymbol{u}^n$ of dimension $(N_x+1)\times(N_y+1)\times(N_z+1)$, a simple analogy to one dimension (9.13) and two dimensions (10.5) gives

$$u^{n+1}_{jkl}=u^n_{jkl}+\big(r_x\Delta_2^{(x)}+r_y\Delta_2^{(y)}+r_z\Delta_2^{(z)}\big)u^n_{jkl}+\Delta t\,Q^n_{jkl},\tag{10.21}$$

with the error of order $\mathcal{O}(\Delta t)+\mathcal{O}(\Delta x^2)+\mathcal{O}(\Delta y^2)+\mathcal{O}(\Delta z^2)$. This scheme is conditionally stable, with the condition $r_x+r_y+r_z\le 1/2$. Similarly, we generalize (10.13) to the absolutely stable three-dimensional Crank–Nicolson scheme

$$\left(1-\frac{r_x}{2}\Delta_2^{(x)}-\frac{r_y}{2}\Delta_2^{(y)}-\frac{r_z}{2}\Delta_2^{(z)}\right)u^{n+1}_{jkl}$$
$$=\left(1+\frac{r_x}{2}\Delta_2^{(x)}+\frac{r_y}{2}\Delta_2^{(y)}+\frac{r_z}{2}\Delta_2^{(z)}\right)u^n_{jkl}+\frac{\Delta t}{2}\big(q^n_{jkl}+q^{n+1}_{jkl}\big),\tag{10.22}$$

which is of order $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta y^2) + \mathcal{O}(\Delta z^2)$. At each time step this scheme requires us to solve a matrix equation of the form $F_1 \boldsymbol{u}^{n+1} = F\boldsymbol{u}^n + \cdots$ with $(N_x - 1)(N_y - 1)(N_z - 1) \times (N_x - 1)(N_y - 1)(N_z - 1)$ matrices F_1 and F. Obviously, even a coarse discretization may lead to practical (memory) obstacles. If we insist on difference methods for parabolic PDE, we also resort to ADI schemes in three dimensions, as the corresponding matrices have a simpler structure.

10.1.6 Hyperbolic Equations

We start our study of two-dimensional hyperbolic PDE by the initial-value problem

$$\begin{aligned} v_t + cv_x + dv_y &= 0, \\ v(x, y, 0) &= f(x, y). \end{aligned} \tag{10.23}$$

This problem has the analytic solution $v(x, y, t) = f(x - ct, y - dt)$ which tells us that at $t > 0$ the initial condition propagates without change of amplitude or phase with velocity c along the x-axis and velocity d along the y-axis. In the following we describe three classes of corresponding difference schemes.

10.1.7 Explicit Schemes

Again, the analogy to one-dimensional discussion allows us to augment the schemes (9.39), (9.40), and (9.41) by the additional dimension:

$$u_{jk}^{n+1} = \big(1 - R_x \Delta_+^{(x)} - R_y \Delta_+^{(y)}\big) u_{jk}^n, \tag{10.24}$$

$$u_{jk}^{n+1} = \big(1 - R_x \Delta_-^{(x)} - R_y \Delta_-^{(y)}\big) u_{jk}^n, \tag{10.25}$$

$$u_{jk}^{n+1} = \left(1 - \frac{R_x}{2} \Delta_0^{(x)} - \frac{R_y}{2} \Delta_0^{(y)}\right) u_{jk}^n, \tag{10.26}$$

where $R_x = c\Delta t/\Delta x$ and $R_y = d\Delta t/\Delta y$. The FTFS scheme (10.24) is consistent to order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x) + \mathcal{O}(\Delta y)$, and its symbol is $\rho(\xi, \eta) = 1 - R_x[\exp(\mathrm{i}\xi) - 1] - R_y[\exp(\mathrm{i}\eta) - 1]$. From the conditions $|\rho(\pm\pi, 0)| \le 1$ and $|\rho(0, \pm\pi)| \le 1$ (at the points where $|\rho|^2$ has the extrema) we obtain the conditions $-1 \le R_x \le 0$ and $-1 \le R_y \le 0$, while the condition $|\rho(\pm\pi, \pm\pi)| \le 1$ yields the final stability criterion $-1 \le R_x + R_y \le 0$. The FTBS scheme (10.25) has the same order as FTFS, and its symbol $\rho(\xi, \eta) = 1 - R_x[1 - \exp(-\mathrm{i}\xi)] - R_y[1 - \exp(-\mathrm{i}\eta)]$ implies the stability criterion $0 \le R_x + R_y \le 1$ with $0 \le R_x \le 1$ and $0 \le R_y \le 1$. (Check this as an exercise and use the discussion accompanying (9.23)!) The FTCS scheme (10.26) is absolutely unstable, just as (9.41) has been.

10.1.8 Schemes for Equations in the Form of Conservation Laws

The analogy to the one-dimensional case (9.76) also leads directly to two-dimensional difference schemes for hyperbolic differential equations that are written in the form of a conservation law

$$v_t + \big[F(v)\big]_x + \big[G(v)\big]_y = 0. \tag{10.27}$$

But recall the discussion in Sect. 9.12! The conservation law formulation by itself does not protect us from the unwanted effects of the difference scheme like dissipation and dispersion (Fig. 9.10), but it is a natural step on the path towards high-resolution schemes. In spite of this warning, we use the template (9.72) to construct the scheme

$$u_{jk}^{n+1} = u_{jk}^n - R_x\big[h_{j+1/2k}^{(x)n} - h_{j-1/2k}^{(x)n}\big] - R_y\big[h_{jk+1/2}^{(y)n} - h_{jk-1/2}^{(y)n}\big], \tag{10.28}$$

where $R_x = \Delta t/\Delta x$ and $R_y = \Delta t/\Delta y$, and where $h^{(x)}$ and $h^{(y)}$ are numerical flux functions in the x and y directions. For the linear conservative problem of the form $v_t + cv_x + dv_y = 0$ the flux functions may be simply

$$h_{j+1/2k}^{(x)n} = cu_{jk}^n, \qquad h_{jk+1/2}^{(y)n} = du_{jk}^n. \tag{10.29}$$

For a general non-linear conservative problem (10.27) the upwind scheme (9.84) is generalized by introducing the flux functions

$$h_{j+1/2k}^{(x)n} = \frac{1}{2}\big[F_{jk}^n + F_{j+1k}^n\big] - \frac{1}{2}\big|a_{j+1/2k}^n\big|\Delta_+^{(x)}u_{jk}^n,$$
$$h_{jk+1/2}^{(y)n} = \frac{1}{2}\big[G_{jk}^n + G_{jk+1}^n\big] - \frac{1}{2}\big|a_{jk+1/2}^n\big|\Delta_+^{(y)}u_{jk}^n,$$

except that the expressions $a_{j+1/2k}^n$ and $a_{jk+1/2}^n$ have to be computed such that in the former case we use the x coordinate (index j) as in (9.86), while in the latter we use the y coordinate (index k) analogously. The split (two-step) Lax–Wendroff scheme

$$u_{jk}^{n+1/2} = u_{jk}^n - R_x\Delta_-^{(x)}\big[h_{j+1/2k}^{(x)n}\big], \tag{10.30}$$
$$u_{jk}^{n+1} = u_{jk}^{n+1/2} - R_y\Delta_-^{(y)}\big[h_{jk+1/2}^{(y)n+1/2}\big], \tag{10.31}$$

is also very useful. It exploits the flux functions

$$h_{j+1/2k}^{(x)n} = cu_{jk}^n + \frac{1}{2}c(1 - cR_x)\Delta_+^{(x)}u_{jk}^n, \tag{10.32}$$

$$h_{jk+1/2}^{(y)n+1/2} = du_{jk}^{n+1/2} + \frac{1}{2}d(1 - dR_y)\Delta_+^{(y)}u_{jk}^{n+1/2}. \tag{10.33}$$

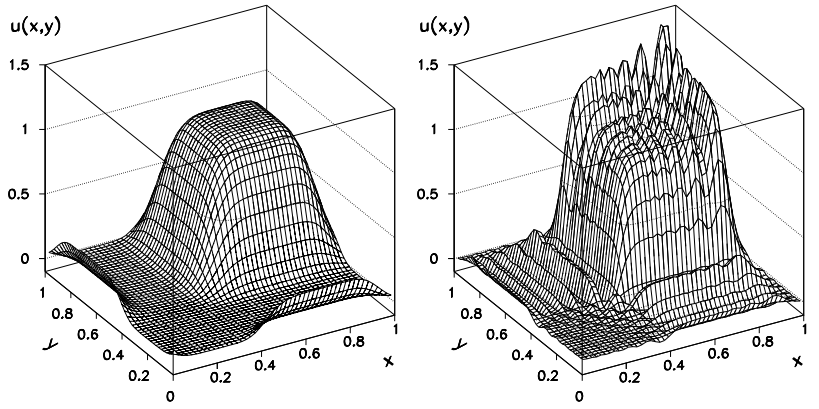


Fig. 10.3 The numerical solution of the hyperbolic problem $v_t + v_x + v_y = 0$ on $(x, y) \in [0, 1] \times [0, 1]$ with the initial condition $v(x, y, 0) = 1$ for $(x, y) \in [\frac{1}{4}, \frac{3}{4}] \times [\frac{1}{4}, \frac{3}{4}]$ at time $t = 0.2$. [*Left*] Dissipation of the solution in the upwind scheme. [*Right*] Strong dispersion of the solution in the split Lax–Wendroff scheme. The discretization in both cases is $N_x = N_y = 48$ and $\Delta t = 0.002$. Compare to Fig. 10.5

Dissipation and dispersion effects in difference schemes become apparent already in linear problems with discontinuous initial conditions: see Fig. 10.3 and compare it to Fig. 9.10! This comparison clearly demonstrates the need to build methods capable of resolving discontinuities. You will encounter the two-dimensional upwind scheme and the split Lax–Wendroff scheme in Problem 10.9.4, accompanied by high-resolution schemes from Sect. 10.3.

10.1.9 Implicit and ADI Schemes

Good stability properties of implicit methods represent a significant advantage that should certainly be exploited in schemes for two-dimensional hyperbolic PDE. For example, the two most obvious absolutely stable schemes for problems of the form (10.23), where we define $R_x = c\Delta t/\Delta x$ and $R_y = d\Delta t/\Delta y$, are the BTCS scheme

$$\left(1 + \frac{R_x}{2}\Delta_0^{(x)} + \frac{R_y}{2}\Delta_0^{(y)}\right)u_{jk}^{n+1} = u_{jk}^n,$$

which is convergent to order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta y^2)$, and the Crank–Nicolson scheme

$$\left(1 + \frac{R_x}{4}\Delta_0^{(x)} + \frac{R_y}{4}\Delta_0^{(y)}\right)u_{jk}^{n+1} = \left(1 - \frac{R_x}{4}\Delta_0^{(x)} - \frac{R_y}{4}\Delta_0^{(y)}\right)u_{jk}^n,$$

which is of order $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta y^2)$.

Similarly to their application to parabolic equations (p. 523), alternating direction implicit (ADI) methods allow us to avoid sparse matrices on the left-hand sides of the matrix equations that express the scheme in matrix notation. A typical representative is the absolutely stable Beam-Warming scheme of order $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta y^2)$,

$$\left(1 + \frac{R_x}{4}\Delta_0^{(x)}\right)u^{\star}_{jk} = \left(1 - \frac{R_x}{4}\Delta_0^{(x)}\right)\left(1 - \frac{R_y}{4}\Delta_0^{(y)}\right)u^{n}_{jk},$$
$$\left(1 + \frac{R_y}{4}\Delta_0^{(y)}\right)u^{n+1}_{jk} = u^{\star}_{jk}.$$

10.2 Elliptic PDE

Numerical solution of elliptic PDE is fundamentally different from the solution of parabolic and hyperbolic equations. While in the latter two cases we seek the time dependence of the solution on some spatial definition domain, the solution of elliptic problems is a time-independent function that solves the given PDE on its domain with specified boundary conditions. By far the most well-known elliptic PDE is the Poisson equation $-\nabla^2 v = Q$.

10.2.1 Dirichlet Boundary Conditions

The basic model problem with Dirichlet boundary conditions is the two-dimensional Poisson equation "on the square":

$$\begin{aligned} -\nabla^2 v &= Q(x,y), \quad (x,y) \in R = [0,1] \times [0,1], \\ v(x,y) &= f(x,y), \quad (x,y) \in \partial R. \end{aligned} \tag{10.34}$$

We discretize the equation along both space axes as shown in Fig. 10.1—we just ignore the time axis (index n). When we approximate the Laplace operator by the finite differences (10.2) and (10.3), the problem (10.34) becomes

$$-\frac{1}{\Delta x^2}\Delta_2^{(x)} u_{jk} - \frac{1}{\Delta y^2}\Delta_2^{(y)} u_{jk} = q_{jk}, \tag{10.35}$$

with the boundary conditions on four sides of the square

$$\begin{aligned} u_{0k} &= q_{0k}, \qquad u_{N_x k} = q_{N_x k}, \quad k = 0, 1, \ldots, N_y, \\ u_{j0} &= q_{j0}, \qquad u_{jN_y} = q_{jN_y}, \quad j = 0, 1, \ldots, N_x. \end{aligned} \tag{10.36}$$

In all Problems we will maintain $N_x = N_y = N$ ($\Delta x = \Delta y$) for simplicity; this will not deprive us of any essential ingredient.

We shall also encounter a much more general elliptic PDE,

$$av_{xx} + cv_{yy} + dv_x + ev_y + fv = Q, \tag{10.37}$$

where a, c, d, e, f, and Q may be functions of x and y (we require only $a < 0$, $c < 0$, and $f > 0$). We discretize it as

$$\frac{a}{\Delta x^2}\Delta_2^{(x)}u_{jk} + \frac{c}{\Delta y^2}\Delta_2^{(y)}u_{jk} + \frac{d}{2\Delta x}\Delta_0^{(x)}u_{jk} + \frac{e}{2\Delta y}\Delta_0^{(y)}u_{jk} + fu_{jk} = q_{jk}.$$

From the programming viewpoint—following the advice of [2]—it is worthwhile to prepare the ground for such a general scheme, and write it in the form

$$\alpha^1_{jk}u_{j+1k} + \alpha^2_{jk}u_{j-1k} + \alpha^3_{jk}u_{jk+1} + \alpha^4_{jk}u_{jk-1} - \alpha^0_{jk}u_{jk} = q_{jk}, \tag{10.38}$$

where the coefficients depend on the location (indices j and k).

We arrange the solution in a $(N_x + 1)(N_y + 1)$-dimensional vector of the form (10.1), we just drop the time index n. But due to the Dirichlet boundary conditions a shorter, $(N_x - 1)(N_y - 1)$-dimensional, vector (10.10) that does not include the components (10.36), suffices. The problems (10.34) and (10.37) can then be written as

$$A\boldsymbol{u} = \boldsymbol{q},$$

where A is a $(N_x - 1)(N_y - 1) \times (N_x - 1)(N_y - 1)$ matrix. By comparing the coefficients in (10.35) and (10.38) we find

$$\alpha^0_{jk} = -2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right), \qquad \alpha^1_{jk} = \alpha^2_{jk} = -\frac{1}{\Delta x^2}, \qquad \alpha^3_{jk} = \alpha^4_{jk} = -\frac{1}{\Delta y^2}. \tag{10.39}$$

The matrix A corresponding to problem (10.34) therefore has the form as in Fig. 10.2 (case (a)) with the elements $\bullet = 2(1/\Delta x^2 + 1/\Delta y^2)$, $\circ = -1/\Delta x^2$, and $\triangleright = -1/\Delta y^2$. The symmetric matrix A is strictly positive definite, thus invertible, and the problem (10.35) has a unique solution. On the other hand, the matrix A corresponding to problem (10.37) is not symmetric in general. It is invertible only if it is strictly diagonally dominant, i.e. when $|a_{jj}| > \sum_{k=1,k\neq j}^{N-1} |a_{jk}|$ for each $j = 1, 2, \ldots, N-1$. This requirement is met by fulfilling the conditions

$$0 < \Delta x < -\frac{2a_{jk}}{|d_{jk}|}, \qquad 0 < \Delta y < -\frac{2c_{jk}}{|e_{jk}|}.$$

The discrete scheme (10.35) is consistent with the differential equation (10.34) to order $\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta y^2)$. If we assume that the solution of the equation, v, is four

times continuously differentiable with respect to x and y on $R^0 = R \backslash \partial R$ (the definition domain minus the boundary), it can be shown that the scheme is convergent to the same order,

$$\|\boldsymbol{v} - \boldsymbol{u}\|_\infty \leq C\left(\Delta x^2 + \Delta y^2\right)\left\|\partial^4 \boldsymbol{v}\right\|_{\infty 0}. \tag{10.40}$$

Here $\|\partial^4 \boldsymbol{v}\|_{\infty 0} = \sup\{|\partial^4 v(x, y)/\partial x^p \partial y^q| : p + q = 4; 0 \leq p, q \leq 4; (x, y) \in R^0\}$. The scheme for problem (10.37) described above has the same order of convergence.

10.2.2 Neumann Boundary Conditions

The basic elliptic model problem involving Neumann boundary conditions is the two-dimensional Poisson equation "on the square":

$$-\nabla^2 v = Q(x, y), \quad (x, y) \in R = [0, 1] \times [0, 1],$$

$$\frac{\partial v}{\partial n}(x, y) = g(x, y), \quad (x, y) \in \partial R.$$

We discretize the operator side of the equation as in the case of Dirichlet boundary conditions, and the boundary conditions to first or second order (compare to (10.11) or (10.12) for the two-dimensional diffusion equation.) Most of the numerical methods described below for Dirichlet problems can be applied to Neumann problems equally well. A solution example is given in Problem 10.9.3.

10.2.3 Mixed Boundary Conditions

Difference schemes for two-dimensional elliptic PDE with mixed boundary conditions $\alpha v + \beta(\partial v/\partial n) = h(x, y)$ for $(x, y) \in \partial R$ become somewhat cumbersome even in simple geometries. Such problems do not offer major new insights and we do not discuss them here. Details can be found in [2], Sect. 10.8.

10.2.4 Relaxation Methods

Several classes of methods are available to solve the system $A\boldsymbol{u} = \boldsymbol{q}$. In *direct methods* we solve the system directly, by "inverting" A. Naive direct approaches (for example, variants of the Cholesky method) are time- and memory-consuming (see Table 10.1) and we do not discuss them here. Fast direct methods are based on the Fourier transformation or cyclic reduction (see Chap. 11). In *iterative methods* we reach the final answer in a sequence of steps in which the current approximate solution $\boldsymbol{w}$ (by some criterion) becomes increasingly similar to the final numerical

Table 10.1 Direct (D) and iterative (I) methods for solving the matrix problem $A\boldsymbol{u} = \boldsymbol{q}$ following from the discretization of the Poisson equation on a $N \times N$ mesh. The numerical cost (number of arithmetic operations and amount of memory) is given in terms of the *order of magnitude* in the power $n = N^2$: the actual cost in iterative methods depends on the criterion used to terminate the iteration

Method	Type	Number of operations	Memory
Jacobi	I	n^2	n
Gauss–Seidel	I	n^2	n
Conjugate gradients	I	$n^{3/2}$	n
SOR (SSOR+Chebyshev)	I	$n^{3/2}$ ($n^{5/4}$)	n
FFT	D	$n \log n$	n
Block cyclic reduction	D	$n \log n$	n
Multi-grid	I	n	n

Table 10.2 Forms of the matrix B in residual correction methods for iterative solution of the system $A\boldsymbol{u} = \boldsymbol{q}$. The iteration matrix is $I - BA$, where $A = L + D + U$, and ω is a free parameter

Method	Matrix B
Jacobi	D^{-1}
Gauss–Seidel	$(L + D)^{-1}$
SOR	$\omega(D + \omega L)^{-1}$
SSOR	$\omega(2 - \omega)(D + \omega U)^{-1} D (D + \omega L)^{-1}$

solution $\boldsymbol{u}$. We discuss *relaxation methods* in which we strive, in each step, to reduce the *algebraic error* $\boldsymbol{e} = \boldsymbol{u} - \boldsymbol{w}$ and the *residual error* $\boldsymbol{r} = \boldsymbol{q} - A\boldsymbol{w}$. We also mention *conjugate gradient methods*.

In *residual correction methods* we approximate the matrix A^{-1} and define the iteration

$$\boldsymbol{w}^{n+1} = \boldsymbol{w}^n + B\boldsymbol{r}^n, \quad n = 0, 1, \ldots,$$

where $\boldsymbol{r}^n = \boldsymbol{q} - A\boldsymbol{w}^n$ and B is some approximation of A^{-1}. On purpose, we have chosen n to denote the iteration index, just as in methods for parabolic and hyperbolic PDE that indeed contain the dependence on time. The errors $\boldsymbol{e}^n$ and $\boldsymbol{r}^n$ are related by the equation $A\boldsymbol{e}^n = A(\boldsymbol{u} - \boldsymbol{w}^n) = \boldsymbol{q} - A\boldsymbol{w}^n = \boldsymbol{r}^n$, so in "time" we iterate the solution as $\boldsymbol{w}^{n+1} = \boldsymbol{w}^n + BA\boldsymbol{e}^n$. The changing of the algebraic error $\boldsymbol{e}^{n+1} = (I - BA)\boldsymbol{e}^n$ is determined by the *iteration matrix* (also called the *error propagation matrix*) $R = I - BA$. We split the matrix A to the lower-triangular, diagonal, and upper-triangular part, $A = L + D + U$. Various relaxation schemes differ by the choice of the matrix B (Table 10.2).

Jacobi Method In the Jacobi method we iterate $\boldsymbol{w}^{n+1} = D^{-1}(\boldsymbol{q} - (L+U)\boldsymbol{w}^n)$. For the Dirichlet problem (10.35) this means

$$\begin{aligned} u_{jk}^{n+1} &= -\frac{1}{\alpha_{jk}^0}\left[q_{jk} - \alpha_{jk}^1 u_{j+1k}^n - \alpha_{jk}^2 u_{j-1k}^n - \alpha_{jk}^3 u_{jk+1}^n - \alpha_{jk}^4 u_{jk-1}^n\right] \\ &= \frac{1}{d}\left[q_{jk} + \frac{1}{\Delta x^2}\left(u_{j+1k}^n + u_{j-1k}^n\right) + \frac{1}{\Delta y^2}\left(u_{jk+1}^n + u_{jk-1}^n\right)\right], \end{aligned} \tag{10.41}$$

where α_{jk} are given by (10.39) and $d = -\alpha_{jk}^0 = 2(1/\Delta x^2 + 1/\Delta y^2)$. Equation (10.41) applies at $j = 1, 2, \ldots, N_x - 1$ and $k = 1, 2, \ldots, N_y - 1$, while the remaining values are specified by the boundary conditions. At each iteration step, we therefore replace the solution at (j, k) by the average of the solutions at four neighboring points (Fig. 10.1 (right)), and admix the source term q.

A common recommendation for implementing the method [2] is to code it along the first, not the second line of (10.41), even though such luxury appears to be redundant in such a simple method. This arms us for the attack on more general equations with space-dependent coefficients, as in (10.37), and on other types of boundary conditions. The coefficients $\alpha^{0,1,2,3,4}$ may be matrices (in this case relocate α_{jk}^0 to the left of the equation, in front of u_{jk}^{n+1}). The same advice applies to other relaxation methods.

Gauss–Seidel Method In the Gauss–Seidel method we acquire consecutive approximations of the solution by the iteration $\boldsymbol{w}^{n+1} = (L+D)^{-1}(\boldsymbol{q} - U\boldsymbol{w}^n)$. For the problem (10.35) this can be written as

$$\begin{aligned} u_{jk}^{n+1} &= -\frac{1}{\alpha_{jk}^0}\left[q_{jk} - \alpha_{jk}^1 u_{j+1k}^n - \alpha_{jk}^2 \underline{u_{j-1k}^{n+1}} - \alpha_{jk}^3 u_{jk+1}^n - \alpha_{jk}^4 \underline{u_{jk-1}^{n+1}}\right] \\ &= \frac{1}{d}\left[q_{jk} + \frac{1}{\Delta x^2}\left(u_{j+1k}^n + \underline{u_{j-1k}^{n+1}}\right) + \frac{1}{\Delta y^2}\left(u_{jk+1}^n + \underline{u_{jk-1}^{n+1}}\right)\right]. \end{aligned} \tag{10.42}$$

This iteration differs from the Jacobi only in the underlined terms. Even though the iteration index $n+1$ appears on both sides of the equation, the scheme is explicit: the values u_{j-1k}^{n+1} and u_{jk-1}^{n+1} in the loops with increasing j and k are always computed just before they are actually needed. The averaging over the neighbors of the point (j, k) takes place in the current iteration step.

SOR and SSOR Methods A tremendous improvement of the convergence speed can be obtained by the *successive over-relaxation* (SOR) scheme. In the first part of the iteration we perform one Gauss–Seidel step, and then form the average of the values from this step and the previously computed values:

$$\begin{aligned} u_{jk}^{\star} &= -\frac{1}{\alpha_{jk}^0}\left[q_{jk} - \alpha_{jk}^1 u_{j+1k}^n - \alpha_{jk}^2 u_{j-1k}^{n+1} - \alpha_{jk}^3 u_{jk+1}^n - \alpha_{jk}^4 u_{jk-1}^{n+1}\right], \\ u_{jk}^{n+1} &= u_{jk}^n + \omega\left(u_{jk}^{\star} - u_{jk}^n\right), \end{aligned}$$

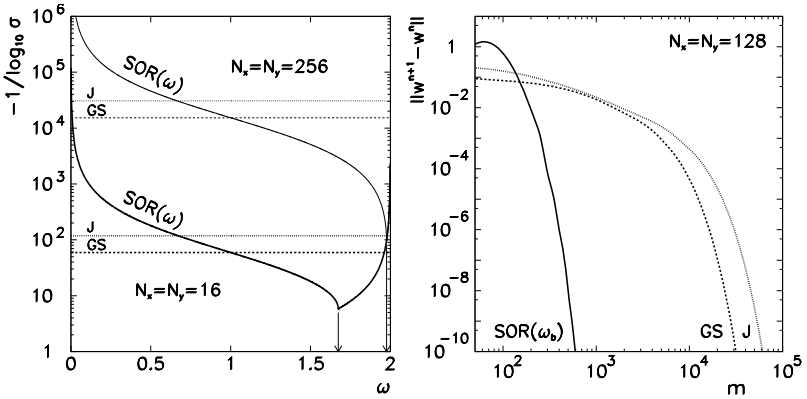


Fig. 10.4 [*Left*] The spectral radius σ of Jacobi, Gauss–Seidel, and SOR iteration matrices for the solution of the problem (10.35) on $N_x = N_y = 16$ and $N_x = N_y = 256$ meshes. Shown is the number of iteration steps needed to improve the precision of the result by one digit (the ordinate corresponds to (10.43) with $\zeta = 0.1$). The *vertical arrows* denote the optimal values of the parameter $\omega = \omega_b$ for the SOR method (1.67351 and 1.97575), where the number of steps drops dramatically. [*Right*] The number of steps m needed to achieve the chosen precision, measured by (10.46). In *both figures* we see the typical double convergence speed of the Gauss–Seidel method compared to the Jacobi method, and the much faster convergence of the SOR method when ω_b is optimal

where

$$0 < \omega < 2$$

is a free parameter; outside of this range the method does not converge. If ω is chosen carefully, the number of necessary iteration steps drops dramatically (Fig. 10.4). At $\omega = 1$ the SOR scheme is equivalent to the Gauss–Seidel scheme. The choice of the optimal ω is described in the following. The SOR scheme can also be written in symmetrized form (SSOR), which combs the discrete mesh in two directions, and is less sensitive to the choice of optimal ω. It can be further accelerated by a trick due to Chebyshev; we eschew these finesses [2, 3].

Example A residual correction method converges for any initial approximation $\boldsymbol{w}^0$ precisely when the spectral radius of its iteration matrix R is strictly less than one, $\sigma(R) < 1$ [4]. Yet we wish the method not only to converge, but to converge quickly! The speed of convergence is roughly determined by the largest eigenvalue λ_1 of the iteration matrix R: to reduce the algebraic error $\boldsymbol{e}$ by a factor $\zeta = \|\boldsymbol{e}^{n+m}\|/\|\boldsymbol{e}^n\|$, we must iterate

$$m \approx \frac{\log \zeta}{\log \sigma(R)} = \frac{\log \zeta}{\log |\lambda_1|} \tag{10.43}$$

times. The number of iterations m may be very large if $|\lambda_1| \approx 1$, or if there are several eigenvalues for which $|\lambda_j/\lambda_1| < 1$ while $|\lambda_j/\lambda_1| \approx 1$. For example, the spectral radius $\sigma(R_J)$ of the Jacobi iteration matrix $R_J = I - D^{-1}A$ for the scheme (10.35)

can be computed exactly:

$$\sigma(R_{\mathrm{J}}) = \frac{2}{d}\left(\frac{1}{\Delta x^2}\cos\frac{\pi}{N_x} + \frac{1}{\Delta y^2}\cos\frac{\pi}{N_y}\right) < 1. \tag{10.44}$$

For $N_x = N_y = 1000$ ($\Delta x = \Delta y = 1/1000$) we get $\sigma(R_{\mathrm{J}}) = 0.999995$, which means that in order to improve the precision of the result by one digit we should make $-1/\log\sigma(R_{\mathrm{J}}) \approx 466000$ iterations! The Gauss–Seidel iteration does not fare much better: its speed of convergence is namely just double that of Jacobi (since $\sigma(R_{\mathrm{GS}}) = \sigma(R_{\mathrm{J}})^2$).

Choosing the Parameter ω in the SOR Method By appropriately choosing the parameter ω in the SOR method we try to minimize the spectral radius of the corresponding iteration matrix (Table 10.2). Determining the optimal value ω_{b} is crucial for improving the speed of convergence (Fig. 10.4). In rare cases the spectral radius of the Jacobi iteration matrix can be computed analytically, e.g. as in (10.44). Then the optimal ω can be computed as

$$\omega_{\mathrm{b}} = \frac{2}{1+\sqrt{1-\overline{\lambda}_{\mathrm{J}}^2}}, \tag{10.45}$$

where $\overline{\lambda}_{\mathrm{J}} = \sigma(R_{\mathrm{J}})$. If $\sigma(R_{\mathrm{J}})$ cannot be computed, one may resort to the procedure given below (and justified in detail in Sect. 10.5.12 of [2]).

We start by choosing a value of ω to initialize the SOR scheme. After a large number of iteration steps the ratio of differences of consecutive approximations becomes almost equal to the largest eigenvalue of the SOR iteration matrix:

$$n \gg 1 \quad \Longrightarrow \quad \frac{w_j^{n+1} - w_j^n}{w_j^n - w_j^{n-1}} \approx \lambda_{1,\omega}.$$

(The expression applies to the jth component of the approximate solution vector $\boldsymbol{w}$ at three consecutive iteration steps $n-1$, n, and $n+1$; in practice we may indeed monitor just one component j or a set of different components.) We use this value to compute the approximation of the largest eigenvalue of the Jacobi iteration matrix,

$$\overline{\lambda}_{\mathrm{J}} \approx \frac{1-\omega-\lambda_{1,\omega}}{\omega\sqrt{\lambda_{1,\omega}}},$$

which can finally be used in (10.45). If we happen to choose $\omega > \omega_{\mathrm{b}}$ (which, unfortunately, we only realize once we try), the procedure does not work. One must keep on trying until $\omega \le \omega_{\mathrm{b}}$.

The convergence of iterative methods should be monitored in order to achieve the desired precision. The iteration may be terminated when the norm of the difference between the subsequent approximations,

$$\left\| \boldsymbol{w}^{n+1} - \boldsymbol{w}^n \right\|, \tag{10.46}$$

or the norm of the current residual error,

$$\left\| \boldsymbol{q} - A\boldsymbol{w}^n \right\|, \tag{10.47}$$

drop below the specified tolerance. For safety, we often check both convergences in various norms, e.g. l_2, $l_{2,\Delta x}$, or sup-norm. It makes no sense to choose a tolerance smaller than the discretization error (10.40) since this is just wasting CPU time. Because the error constants C of $\|\partial^4 v\|_{\infty 0}$ in (10.40) are not known, we may estimate the appropriate tolerance by trial-and-error, e.g. with analytically solvable cases. Further advice is offered by [2] in Sect. 10.5.4.

Finally, what is it that actually "relaxes" in relaxation methods? We may see the solution of $-\nabla^2 v = Q$ as the solution of $v_t = \nabla^2 v + Q$ at very long times: the initial temperature distribution $v(x, y, 0)$ relaxes to the equilibrium solution at $t \to \infty$ where $v_t = 0$. In other words, relaxation methods are, at least in principle, also applicable to non-stationary parabolic problems, since two-step difference schemes can be expressed in the form $F_1 \boldsymbol{u}^{n+1} = F\boldsymbol{u}^n + \Delta t \boldsymbol{q}$. At each time step we are therefore again solving the matrix equation $A\boldsymbol{u} = \boldsymbol{q}$. Yet simple iterative relaxation methods are much slower than alternating gradient implicit methods. Moreover, iterative methods are not suitable for the solution of hyperbolic equations $v_t + av_x + bv_x = v_{xx} + v_{yy}$, as the structure of the corresponding matrices becomes too complicated.

10.2.5 Conjugate Gradient Methods

Relaxation methods, like the SOR and its improved versions (symmetrized SOR with Chebyshev acceleration) are useful only if near-optimal convergence parameters can be determined accurately and effectively. This problem can be avoided by using *conjugate gradient methods* that belong to a broader class of Krilov subspaces methods [5]. In these methods we do not solve the equation $A\boldsymbol{u} = \boldsymbol{q}$ but rather seek the minimum of the scalar function $F(\boldsymbol{u}) = (A\boldsymbol{u})^{\mathrm{T}}\boldsymbol{u} - \boldsymbol{q}^{\mathrm{T}}\boldsymbol{u}$ or the zero of $F'(\boldsymbol{u}) = A\boldsymbol{u} - \boldsymbol{q}$ by using algorithms of steepest descent (to a minimum). Krilov subspaces methods are also useful for the computation of eigenvalues of sparse matrices. Naive formulations of conjugate gradient methods are not significantly better than the classical relaxation methods (see Table 10.1), while the *preconditioned conjugate gradient methods* truly out-perform them: in these methods, we regroup the eigenvalues of A prior to solving the system in order to improve the speed of convergence. Further reading can be found in the pedagogically polished review article [6] and in the book [7], while basic programming hints are given in [2].

10.3 High-Resolution Schemes ⋆

At the end of Sect. 10.1 we stopped mid-way in difference schemes for two-dimensional hyperbolic problems that can be written in the form of a conservation law $v_t + [F(v)]_x + [G(v)]_y = 0$. In two dimensions we encounter difficulties

closely resembling those from one space dimension (Fig. 9.10): especially for discontinuous initial conditions we observe dissipation and strong dispersion in the time evolution of the solutions. We tame them (for example) by using flux limiter methods from Sect. 9.12.1. In two dimensions we discuss only the linear problem $v_t + cv_x + dv_y = 0$ with $c > 0$ and $d > 0$, for which we describe two high-resolution schemes.

Two Basic Schemes A simple high-resolution scheme can be constructed if (10.28) is equipped by the numerical flux functions

$$h^{(x)n}_{j+1/2k} = cu^n_{jk} + \phi^{(x)n}_{jk} \frac{1}{2} c(1 - cR_x)\Delta^{(x)}_+ u^n_{jk}, \tag{10.48}$$

$$h^{(y)n}_{jk+1/2} = du^n_{jk} + \phi^{(y)n}_{jk} \frac{1}{2} d(1 - dR_y)\Delta^{(y)}_+ u^n_{jk}, \tag{10.49}$$

where

$$\phi^{(x)n}_{jk} = \phi^{(x)}\big(\theta^{(x)n}_{jk}\big), \quad \theta^{(x)n}_{jk} = \big(\Delta^{(x)}_- u^n_{jk}\big)/\big(\Delta^{(x)}_+ u^n_{jk}\big),$$
$$\phi^{(y)n}_{jk} = \phi^{(y)}\big(\theta^{(y)n}_{jk}\big), \quad \theta^{(y)n}_{jk} = \big(\Delta^{(y)}_- u^n_{jk}\big)/\big(\Delta^{(y)}_+ u^n_{jk}\big).$$

These are obvious generalizations of (10.32) and (10.33) with the limiter functions $\phi^{(x)}$ and $\phi^{(y)}$, which we choose for each direction separately (x or y), and according to the physical character of the problem, from the set (9.78)–(9.81). We should pay attention to zeros of the denominators in θ_{jk} and to cases when j or k reach outside of the definition domain (boundary condition). An example of the solution of a hyperbolic problem $v_t + v_x + v_y = 0$ on $(x, y) \in [0, 1] \times [0, 1]$ with this scheme is shown in Fig. 10.5.

We obtain a high-resolution split scheme of the Lax–Wendroff type by including the flux functions (10.48)–(10.49) with the chosen flux limiter functions into the expressions (10.30)–(10.31).

Zalesak–Smolarkiewicz Scheme In Problem 10.9.4 we test several difference methods to solve $v_t + cv_x + dv_y = 0$ with constant c and d, and discontinuous initial conditions. In this test we notice that both high-resolution schemes for linear problems discussed above fail with only a modest change in the coefficients c and d, for example, by allowing them to depend on x and y,

$$v_t + c(x, y)v_x + d(x, y)v_y = 0. \tag{10.50}$$

One of the reasons for this failure is the one-dimensional character of the limiter functions built into the two-dimensional scheme: the limiting of flux in one direction is decoupled from the limiting in the other. For transport problems of the form (10.50) one should therefore seek high-resolution schemes that limit the flux two-dimensionally. Here we describe the Zalesak–Smolarkiewicz method [8, 9] in the notation of [2]. The landmark feature of this method are its two components: a

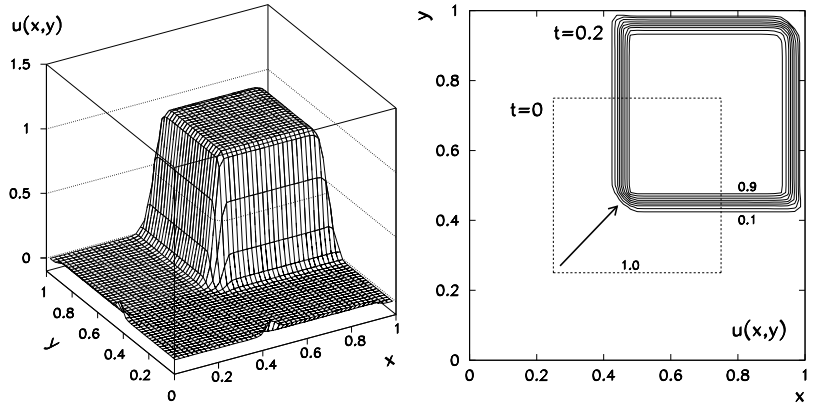


Fig. 10.5 Numerical solution of the linear hyperbolic problem $v_t + v_x + v_y = 0$ on $(x, y) \in [0, 1] \times [0, 1]$ with same conditions as in Fig. 10.3. [*Left*] Solution by (10.48) and (10.49). [*Right*] Solution isolines. The initial condition (*dashed square*) moves to the top right corner after $t = 0.2$ (a fifth of the period). In comparison to Fig. 10.3 (left and right) we observe significantly less dissipation and dispersion

low-order scheme (L) and a high-order (H) scheme. Its second feature is that at each point (j, k) we try to introduce enough anti-diffusive flux into the solution u_{jk} such that u_{jk}^{n+1} is still bounded by some values $u_{jk}^{\min}$ and $u_{jk}^{\max}$ to be determined on the fly. The low-order component,

$$ {}^{\mathrm{L}}u_{jk}^{n} = u_{jk}^{n} - R_x \Delta_{-}^{(x)} \big[{}^{\mathrm{L}}h_{j+1/2k}^{(x)n} \big] - R_y \Delta_{-}^{(y)} \big[{}^{\mathrm{L}}h_{jk+1/2}^{(y)n} \big], $$

contains the upwind scheme

$$ {}^{\mathrm{L}}h_{j+1/2k}^{(x)n} = \frac{1}{2} c \big(u_{jk}^{n} + u_{j+1k}^{n} \big) - \frac{1}{2} |c| \Delta_{+}^{(x)} u_{jk}^{n}, $$

$$ {}^{\mathrm{L}}h_{jk+1/2}^{(y)n} = \frac{1}{2} d \big(u_{jk}^{n} + u_{jk+1}^{n} \big) - \frac{1}{2} |d| \Delta_{+}^{(y)} u_{jk}^{n}. $$

At all j and k, only the values u_{jk} at time $n\Delta t$ are needed for the computation. The high-order component is constructed in two steps. In the first step, we adopt the flux functions of the non-split Lax–Wendroff scheme,

$$ {}^{\mathrm{H}}h_{j+1/2k}^{(x)n} = c u_{jk}^{n} + \frac{1}{2} c (1 - c R_x) \Delta_{+}^{(x)} u_{jk}^{n}, $$

$$ {}^{\mathrm{H}}h_{jk+1/2}^{(y)n} = d u_{jk}^{n} + \frac{1}{2} d (1 - d R_y) \Delta_{+}^{(y)} u_{jk}^{n}, $$

while in the second step, we use the differences

$$ {}^{\mathrm{D}}h_{j+1/2k}^{(x)n} = {}^{\mathrm{H}}h_{j+1/2k}^{(x)n} - {}^{\mathrm{L}}h_{j+1/2k}^{(x)n}, \qquad {}^{\mathrm{D}}h_{jk+1/2}^{(y)n} = {}^{\mathrm{H}}h_{jk+1/2}^{(y)n} - {}^{\mathrm{L}}h_{jk+1/2}^{(y)n}, $$

to form the solution at time $(n+1)\Delta t$:

$$u_{jk}^{n+1} = {}^{\mathrm{L}}u_{jk}^{n} - R_x \Delta_-^{(x)}\big[\phi_{jk}^{(x)n}\, {}^{\mathrm{D}}h_{j+1/2k}^{(x)n}\big] - R_y \Delta_-^{(y)}\big[\phi_{jk}^{(y)n}\, {}^{\mathrm{D}}h_{jk+1/2}^{(y)n}\big].$$

We define the limiter functions ϕ in the form

$$\phi_{jk}^{(x)n} = \begin{cases} \min\{b_{j+1k}^{+}, b_{jk}^{-}\}; & {}^{\mathrm{D}}h_{j+1/2k}^{(x)n} \geq 0, \\ \min\{b_{jk}^{+}, b_{j+1k}^{-}\}; & {}^{\mathrm{D}}h_{j+1/2k}^{(x)n} < 0, \end{cases}$$

$$\phi_{jk}^{(y)n} = \begin{cases} \min\{b_{jk+1}^{+}, b_{jk}^{-}\}; & {}^{\mathrm{D}}h_{jk+1/2}^{(y)n} \geq 0, \\ \min\{b_{jk}^{+}, b_{jk+1}^{-}\}; & {}^{\mathrm{D}}h_{jk+1/2}^{(y)n} < 0, \end{cases}$$

where b_{jk}^{+} is the smallest upper limit for the factor multiplying the anti-diffusive flux a_{jk}^{+} entering the mesh point (j,k), while b_{jk}^{-} is the largest upper limit for the factor multiplying the corresponding exiting flux a_{jk}^{-}:

$$b_{jk}^{+} = \begin{cases} \min\{1, (u_{jk}^{\max} - u_{jk}^{\mathrm{L}})/a_{jk}^{+}\}; & a_{jk}^{+} > 0, \\ 0; & a_{jk}^{+} = 0, \end{cases}$$

$$b_{jk}^{-} = \begin{cases} \min\{1, (u_{jk}^{\mathrm{L}} - u_{jk}^{\min})/a_{jk}^{-}\}; & a_{jk}^{-} > 0, \\ 0; & a_{jk}^{-} = 0. \end{cases}$$

The anti-diffusive fluxes are defined by

$$\begin{aligned} a_{jk}^{+} &= \max\big\{0, {}^{\mathrm{D}}h_{j-1/2k}^{(x)n}\big\} - \min\big\{0, {}^{\mathrm{D}}h_{j+1/2k}^{(x)n}\big\} \\ &\quad + \max\big\{0, {}^{\mathrm{D}}h_{jk-1/2}^{(y)n}\big\} - \min\big\{0, {}^{\mathrm{D}}h_{jk+1/2}^{(y)n}\big\}, \\ a_{jk}^{-} &= \max\big\{0, {}^{\mathrm{D}}h_{j+1/2k}^{(x)n}\big\} - \min\big\{0, {}^{\mathrm{D}}h_{j-1/2k}^{(x)n}\big\} \\ &\quad + \max\big\{0, {}^{\mathrm{D}}h_{jk+1/2}^{(y)n}\big\} - \min\big\{0, {}^{\mathrm{D}}h_{jk-1/2}^{(y)n}\big\}. \end{aligned}$$

Only $u_{jk}^{\min}$ and $u_{jk}^{\max}$ were left to be determined. We choose

$$\begin{aligned} u_{jk}^{\min} &= \min\big\{u_{j+1k}, u_{jk+1}, u_{jk}, u_{j-1k}, u_{jk-1}, u_{j+1k}^{\mathrm{L}}, u_{jk+1}^{\mathrm{L}}, u_{jk}^{\mathrm{L}}, u_{j-1k}^{\mathrm{L}}, u_{jk-1}^{\mathrm{L}}\big\}, \\ u_{jk}^{\max} &= \max\big\{u_{j+1k}, u_{jk+1}, u_{jk}, u_{j-1k}, u_{jk-1}, u_{j+1k}^{\mathrm{L}}, u_{jk+1}^{\mathrm{L}}, u_{jk}^{\mathrm{L}}, u_{j-1k}^{\mathrm{L}}, u_{jk-1}^{\mathrm{L}}\big\}. \end{aligned}$$

This choice embodies the two-dimensional nature of the method, as $u_{jk}^{\min}$ and $u_{jk}^{\max}$ at each point (j,k) couple the values from the four neighboring points.

10.4 Physically Motivated Discretizations

In each difference scheme for PDE or systems of PDE, regardless of the dimensionality, we use some procedure to discretize the space coordinates (and the time axis in

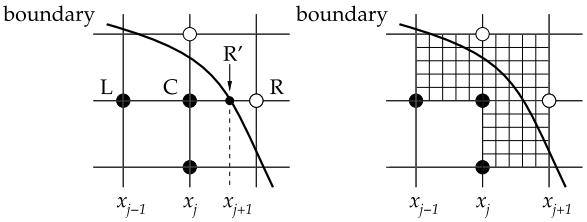


Fig. 10.6 [*Left*] Computing the approximations of u_x and u_{xx} in the vicinity of the domain boundary bypassing the mesh points. In the first order of the mesh spacings we use (10.51) and (10.52). [*Right*] Making the discrete mesh denser

evolution problems). Of course, for numerous realistic (two- and three-dimensional) physics problems, the basic Cartesian layout with square or rectangular domains is not the natural environment. In exceptional cases we may apply coordinate transformations to convert the problems with PDE in non-Cartesian frames to Cartesian ones, but most often the suitable transformation is very difficult to find. Computations in planar polar coordinates, for example, also introduce peculiarities not seen in Cartesian coordinates.

Another option is to solve the equation on the existing (geometrically irregular) domain: in this case the main obstacle is the proper implementation of the boundary conditions in parts of the domain boundary bypassing the mesh points, as shown in Fig. 10.6 (left). For example, if we wish to compute the derivatives u_x and u_{xx} at C, we may use a non-uniform mesh with $\Delta x_j = x_j - x_{j-1}$ and $\Delta x_{j+1} = x_{j+1} - x_j$ near the boundary, and use the differences evaluated at the points L, C, and R′ (instead of at L, C, and R):

$$(u_x)_j = \frac{u_{j+1} - u_{j-1}}{\Delta x_{j+1} + \Delta x_j} + \mathcal{O}\left(\frac{\Delta x_{j+1}^2}{\Delta x_{j+1} + \Delta x_j}\right) + \mathcal{O}\left(\frac{\Delta x_j^2}{\Delta x_{j+1} + \Delta x_j}\right), \tag{10.51}$$

$$(u_{xx})_j = 2\frac{\Delta x_j u_{j+1} - (\Delta x_{j+1} + \Delta x_j)u_j + \Delta x_{j+1} u_{j-1}}{\Delta x_{j+1}\Delta x_j(\Delta x_{j+1} + \Delta x_j)} + \mathcal{O}\left(\frac{\Delta x_{j+1}^2}{\Delta x_{j+1} + \Delta x_j}\right) + \mathcal{O}\left(\frac{\Delta x_j^2}{\Delta x_{j+1} + \Delta x_j}\right). \tag{10.52}$$

We may also make the mesh denser locally, or superpose a finer mesh on a coarser mesh, as in Fig. 10.6 (right), but such a procedure is hard to automate; dedicated programs exist for this task. Details on constructing overlapping discrete meshes can be found in [10] and [11] in the context of multi-grid methods (see also Sect. 10.8).

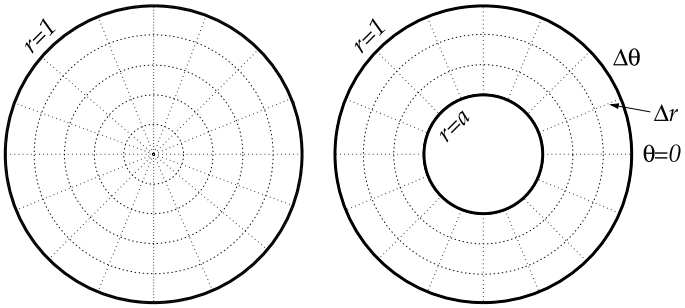


Fig. 10.7 [*Left*] The mesh for the solution of diffusion and Poisson equations in polar coordinates on $[r_0, r_{N_r}] \times [\theta_0, \theta_{N_\theta}] = [0, 1] \times [0, 2\pi]$. The discretization $\Delta\theta = 2\pi/N_\theta$, $\Delta r = (r_{N_r} - r_0)/N_r$, and Δt defines the mesh points $(r_j, \theta_k, t_n) = (r_0 + j\Delta r, k\Delta\theta, n\Delta t)$, and at each of them the true solution $v^n_{jk} = v(r_j, \theta_k, t_n)$ and its approximation u^n_{jk}. [*Right*] The mesh for the Poisson problem on $[a, 1] \times [0, 2\pi]$

10.4.1 Two-Dimensional Diffusion Equation in Polar Coordinates

Let us translate the difference scheme for the two-dimensional diffusion equation in Cartesian coordinates to polar coordinates. We are solving the equation

$$v_t = D\left(\frac{1}{r}(rv_r)_r + \frac{1}{r^2}v_{\theta\theta}\right) + Q(r, \theta, t) \tag{10.53}$$

on $(r, \theta) \in [0, 1] \times [0, 2\pi]$ for $t > 0$, with the initial and boundary conditions

$$\begin{aligned} v(r, \theta, 0) &= f(r, \theta), \quad 0 \le \theta < 2\pi,\ 0 \le r \le 1, \\ v(1, \theta, t) &= g(\theta, t), \quad 0 \le \theta < 2\pi,\ t > 0. \end{aligned}$$

These conditions are supplemented by the periodicity requirement

$$v(r, 0, t) = v(r, 2\pi, t).$$

As in Sect. 10.1 we count the time steps as $t = n\Delta t$, and segment the radial and angular coordinates uniformly:

$$r_j = j\Delta r, \quad j = 0, 1, \ldots, N_r,$$

so we have $r_0 = 0$ and $r_{N_r} = 1$, as well as

$$\theta_k = k\Delta\theta, \quad k = 0, 1, \ldots, N_\theta,$$

so that $\theta_0 = 0$ and $\theta_{N_\theta} = 2\pi$ (Fig. 10.7 (left)). Since the values at $j = 0$ do not depend on k, we abbreviate $u^n_{0k} = u^n_0$, and the continuity in θ implies $u^n_{jN_\theta} = u^n_{j0}$.

We replace the partial derivatives by the corresponding differences:

$$
\begin{aligned}
(v_t)^n_{jk} &\approx \frac{u^{n+1}_{jk} - u^n_{jk}}{\Delta t},\\
\left(\frac{1}{r^2} v_{\theta\theta}\right)^n_{jk} &\approx \frac{1}{r_j^2}\frac{1}{\Delta\theta^2}\Delta_2^{(\theta)} u^n_{jk},\\
\left(\frac{1}{r}(r v_r)_r\right)^n_{jk} &\approx \frac{1}{r_j}\frac{1}{\Delta r^2}\left[r_{j+\frac{1}{2}}\left(u^n_{j+1k} - u^n_{jk}\right) - r_{j-\frac{1}{2}}\left(u^n_{jk} - u^n_{j-1k}\right)\right],
\end{aligned}
$$

and $\Delta_2^{(\theta)}$ is defined by (10.4), where the role of y is now played by θ. We should be aware of the approximation of $(r v_r)_r/r$, in which the radial coordinate appears outside of the mesh points (where it is known): it turns out that such a difference is more precise than the discretization of the equivalent expression $v_r/r + v_{rr}$. When all terms are joined, an explicit difference scheme emerges,

$$
\begin{aligned}
u^{n+1}_{jk} = u^n_{jk} &+ \frac{1}{r_j}\frac{D\Delta t}{\Delta r^2}\left[r_{j+\frac{1}{2}}\left(u^n_{j+1k} - u^n_{jk}\right) - r_{j-\frac{1}{2}}\left(u^n_{jk} - u^n_{j-1k}\right)\right]\\
&+ \frac{1}{r_j^2}\frac{D\Delta t}{\Delta\theta^2}\left[u^n_{jk+1} - 2u^n_{jk} - \underline{u^n_{jk-1}}\right] + \Delta t\, q^n_{jk},
\end{aligned}
\tag{10.54}
$$

which applies at $j = 1, 2, \ldots, N_r - 1$ and $k = 1, 2, \ldots, N_\theta - 1$. For $k = 0$ the underlined term should be understood as $u^n_{jN_\theta - 1}$ (periodicity). We obtain the missing difference equation at $j = 0$ by integrating the differential equation (10.53),

$$
\int_r\int_\theta\int_t v_t \,\mathrm{d}\Omega = D\iiint\left(\frac{1}{r}(r v_r)_r + \frac{1}{r^2} v_{\theta\theta}\right)\mathrm{d}\Omega + \iiint Q(r,\theta,t)\,\mathrm{d}\Omega,
$$

on the time interval $t \in [t_n, t_{n+1}]$ and space intervals ("control volume") $r \in [0, \Delta r/2]$ and $\theta \in [0, 2\pi]$. (We have denoted $\mathrm{d}\Omega = r\,\mathrm{d}r\,\mathrm{d}\theta\,\mathrm{d}t$.) In the term containing the spatial derivatives we use the divergence theorem

$$
\int_0^{\Delta r/2}\left(\frac{1}{r}(r v_r)_r + \frac{1}{r^2} v_{\theta\theta}\right) r\,\mathrm{d}r = \frac{\Delta r}{2} v_r\left(\frac{\Delta r}{2}, \theta, t\right),
$$

and use the trapezoidal formula for the remaining integrals, whence

$$
\pi\frac{\Delta r^2}{4}\left(u^{n+1}_0 - u^n_0\right) \approx D\frac{\Delta r}{2}\Delta t\sum_{k=0}^{N_\theta - 1}\frac{u^n_{1k} - u^n_0}{\Delta r}\Delta\theta + \pi\frac{\Delta r^2\Delta t}{4} q^n_0.
$$

We then use $\sum_k u^n_0 = N_\theta u^n_0 = (2\pi/\Delta\theta) u^n_0$, which ultimately results in

$$
u^{n+1}_0 = \left(1 - \frac{4D\Delta t}{\Delta r^2}\right) u^n_0 + \frac{2D\Delta\theta\,\Delta t}{\pi\,\Delta r^2}\sum_{k=0}^{N_\theta - 1} u^n_{1k} + \Delta t\, q^n_0. \tag{10.55}
$$

The complete scheme is given by (10.54) and (10.55). Polar coordinates are no hindrance for implicit difference schemes, like the pure implicit or Crank–Nicolson by analogy to (10.13). The resulting matrix systems for solution vectors feature matrices with relatively ugly structures. However, these matrices can be transformed to triangular forms; details are given in [1].

10.4.2 Two-Dimensional Poisson Equation in Polar Coordinates

When two-dimensional elliptic PDE are solved in polar coordinates, further peculiarities appear. We discuss two model problems [2]: in the first, we solve the Poisson equation on the annulus,

$$-\nabla^2 v = Q(r,\theta),$$

where $a < r < 1$ and $0 \le \theta \le 2\pi$ (Fig. 10.7 (right)), with Dirichlet boundary conditions

$$\begin{aligned} v(a,\theta) &= f_1(\theta), \quad 0 \le \theta < 2\pi,\ a < r < 1, \\ v(1,\theta) &= f_2(\theta), \quad 0 \le \theta < 2\pi; \end{aligned} \tag{10.56}$$

in the second, we include the origin (thus $a = 0$), so that the first initial condition (10.56) does not apply. In both problems, the spatial part of the difference operator is the same as for the diffusion equation (10.53), so precisely that discretization can be adopted. We span the mesh (r_j, θ_k) on the annulus such that $r_0 = a$, $r_{N_r} = 1$, $\theta_0 = 0$, $\theta_{N_\theta} = 2\pi$, $\Delta r = (1-a)/N_r$, and $\Delta\theta = 2\pi/N_\theta$; if the origin is included, we have $r_0 = 0$ and $\Delta r = 1/N_r$. For either of the problems, this boils down to

$$-\frac{1}{\Delta r^2}\frac{1}{r_j}\left[r_{j+1/2}(u_{j+1k} - u_{jk}) - r_{j-1/2}(u_{jk} - u_{j-1k})\right] - \frac{1}{\Delta\theta^2}\frac{1}{r_j^2}\underline{\Delta_2^{(\theta)} u_{jk}} = q_{jk},$$

where $q_{jk} = Q(r_j, \theta_k)$ for $j = 1, 2, \ldots, N_r - 1$ and $k = 1, 2, \ldots, N_\theta - 1$. When $k = 0$, the scheme accesses the point u_{j-1} in the underlined term; periodicity in θ comes to rescue by setting $u_{j-1} = u_{jN_\theta - 1}$. The remaining components of the solution are given by the boundary conditions. In the annulus problem we have

$$\begin{aligned} u_{jN_\theta} &= u_{j0}, && j = 0, 1, \ldots, N_r, \\ u_{0k} &= f_1(\theta_k), && k = 0, 1, \ldots, N_\theta, \\ u_{N_r k} &= f_2(\theta_k), && k = 0, 1, \ldots, N_\theta. \end{aligned}$$

In the problem that includes the origin we must be careful about—the origin: the values u_{0k} and q_{0k} actually cannot depend on k, thus we abbreviate $u_{0k} = u_0$ and $q_{0k} = q_0$. The boundary conditions then become

$$u_{jN_\theta} = u_{j0}, \qquad j = 0, 1, \ldots, N_r,$$
$$u_{N_r k} = f_2(\theta_k), \quad k = 0, 1, \ldots, N_\theta,$$

and

$$\frac{4}{\Delta r^2} u_0 - \frac{2\Delta\theta}{\pi \Delta r^2} \sum_{k=0}^{N_\theta - 1} u_{1k} = q_0.$$

10.5 Boundary Element Method ⋆

The *boundary element method* (BEM) allows us to solve PDE in non-trivial geometries in which the discretization of the interior of the definition domain is difficult, while it is relatively easy to express (at least in some approximation) the boundary conditions on the boundaries of this domain. This is what makes the BEM method so appealing: just by using the information from the boundaries we solve the problem on the whole domain.

Here the basic outline of BEM is presented, following closely Ref. [12]. As an example we discuss the two-dimensional Laplace equation

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0, \tag{10.57}$$

which we try to solve in the xy-plane in the domain R bounded by the piecewise smooth closed curve C. Along the individual segments C_i of C, either Dirichlet or Neumann boundary conditions are specified:

$$\phi(x, y) = f_i(x, y), \quad (x, y) \in C_i,$$
$$\frac{\partial \phi(x, y)}{\partial n} = f_j(x, y), \quad (x, y) \in C_j,$$

as shown in Fig. 10.8 (left). The normal derivative $\partial\phi/\partial n = n_x(\partial\phi/\partial x) + n_y(\partial\phi/\partial y)$ is defined by the components of the unit normal vector $\boldsymbol{n} = (n_x, n_y)^{\mathrm{T}}$, which points away from the domain R. A physical example of such a problem is the stationary state of heat conduction in isotropic matter. The solution $\phi(x, y)$ describes the distribution of temperature in the domain R with the boundary C, some pieces of which are held at constant temperature, while the others experience a constant heat flux $-\lambda(\partial\phi/\partial n)$.

The particular solution of (10.57) is $\phi(x, y) = A \ln\sqrt{x^2 + y^2} + B$ for $(x, y) \neq (0, 0)$. We choose $A = 1/(2\pi)$, $B = 0$, and move the origin from $(0, 0)$ to (ξ, η). This results in the *fundamental solution* of the Laplace equation,

$$\Phi(x, y; \xi, \eta) = \frac{1}{4\pi} \ln\big[(x - \xi)^2 + (y - \eta)^2\big],$$

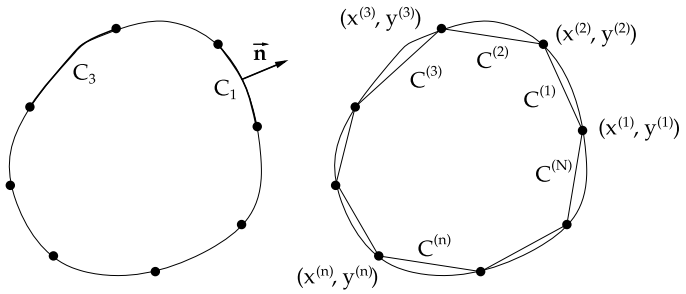


Fig. 10.8 [*Left*] The domain R with the smooth boundary C on which we solve the two-dimensional Poisson equation. As an example, we have Dirichlet boundary conditions on segment C_3 and Neumann boundary conditions on C_1. [*Right*] The approximation of the boundary C by the inscribed polygon with sides $C^{(n)}$

which is defined everywhere except at (ξ, η). Gauss theorem for vector functions, $\int_C \boldsymbol{V} \cdot \boldsymbol{n}\, \mathrm{d}s(x, y) = \int_R \nabla \cdot \boldsymbol{V}\, \mathrm{d}x\, \mathrm{d}y$, can be applied to show that for any two solutions ϕ_1 and ϕ_2 of (10.57), we have

$$\int_C \left[\phi_2(\partial\phi_1/\partial n) - \phi_1(\partial\phi_2/\partial n)\right] \mathrm{d}s(x, y) = 0.$$

We choose $\phi_1 = \Phi(x, y; \xi, \eta)$ and $\phi_2 = \phi(x, y)$, where $\phi(x, y)$ is the desired solution of (10.57). A few basic tricks of complex analysis are then needed to connect the desired solution and the fundamental solution by the *boundary* integral equation

$$\lambda(\xi, \eta)\phi(\xi, \eta) = \int_C \left[\phi(x, y)\frac{\partial}{\partial n}\Phi(x, y; \xi, \eta) - \Phi(x, y; \xi, \eta)\frac{\partial}{\partial n}\phi(x, y)\right] \mathrm{d}s, \tag{10.58}$$

where $\lambda(\xi, \eta) = 1/2$. In the boundary element method, the solution of the basic problem in the interior of the domain R is obtained by solving this integral equation *on the boundary* C. The form of the integral equation remains the same in all parts of the domain of (10.57), only the parameter $\lambda(\xi, \eta)$ changes:

$$\lambda(\xi, \eta) = \begin{cases} 0; & (\xi, \eta) \notin R \cup C, \\ \frac{1}{2}; & (\xi, \eta) \text{ on smooth part of } C, \\ 1; & (\xi, \eta) \in R. \end{cases}$$

We approximate the boundary C by a polygon with N sides, as shown in Fig. 10.8 (right), such that

$$C \approx C^{(1)} \cup C^{(2)} \cup \cdots \cup C^{(N)}.$$

Each side $C^{(n)}$ is a segment between the points $(x^{(n)}, y^{(n)})$ and $(x^{(n+1)}, y^{(n+1)})$. We assume that the values of the functions and their derivatives are constant along

individual sides $C^{(n)}$, i.e.

$$\phi(x,y) \approx v^{(n)}, \qquad \frac{\partial \phi(x,y)}{\partial n} \approx d^{(n)}, \quad (x,y) \in C^{(n)},\ n = 1,2,\ldots,N,$$

where $v^{(n)}$ is the value of ϕ and $d^{(n)}$ is the value of $\partial\phi/\partial n$ in the middle of $C^{(n)}$. Now the integral equation (10.58) can be approximately written as

$$\lambda(\xi,\eta)\phi(\xi,\eta) \approx \sum_{n=1}^{N}\left[v^{(n)}\mathcal{D}^{(n)}(\xi,\eta) - d^{(n)}\mathcal{V}^{(n)}(\xi,\eta)\right], \tag{10.59}$$

where

$$\mathcal{V}^{(n)}(\xi,\eta) = \int_{C^{(n)}} \Phi(x,y;\xi,\eta)\,\mathrm{d}s(x,y), \tag{10.60}$$

$$\mathcal{D}^{(n)}(\xi,\eta) = \int_{C^{(n)}} \frac{\partial \Phi}{\partial n}(x,y;\xi,\eta)\,\mathrm{d}s(x,y). \tag{10.61}$$

For some index n, the boundary condition defines either the value $v^{(n)}$ or the derivative $d^{(n)}$ (but not both), so there are N unknowns at the right-hand side of (10.59). We choose (ξ,η) to lie in the middle of the sides $C^{(n)}$ and obtain

$$\frac{1}{2}v^{(m)} = \sum_{n=1}^{N}\left[v^{(n)}\mathcal{D}^{(n)}\left(\overline{x}^{(m)},\overline{y}^{(m)}\right) - d^{(n)}\mathcal{V}^{(n)}\left(\overline{x}^{(m)},\overline{y}^{(m)}\right)\right], \quad m = 1,2,\ldots,N,$$

where $(\overline{x}^{(m)},\overline{y}^{(m)})$ is the midpoint of $C^{(m)}$. In (10.59) we have used $\lambda = 1/2$, since all midpoints lie on the smooth parts of the approximate boundary C. This is a system of N linear equation, which can be written as

$$\sum_{n=1}^{N} a_{mn} z_n = \sum_{n=1}^{N} b_{mn}, \quad m = 1,2,\ldots,N, \tag{10.62}$$

where

$$\begin{aligned} a_{mn} &= -\mathcal{V}^{(n)}\left(\overline{x}^{(m)},\overline{y}^{(m)}\right),\\ b_{mn} &= v^{(n)}\left[-\mathcal{D}^{(n)}\left(\overline{x}^{(m)},\overline{y}^{(m)}\right) + \frac{1}{2}\delta_{m,n}\right],\\ z_n &= d^{(n)}, \end{aligned}$$

if the boundary condition on $C^{(n)}$ prescribes the value ϕ, or

$$\begin{aligned} a_{mn} &= \mathcal{D}^{(n)}\left(\overline{x}^{(m)},\overline{y}^{(m)}\right) - \frac{1}{2}\delta_{m,n},\\ b_{mn} &= d^{(n)}\mathcal{V}^{(n)}\left(\overline{x}^{(m)},\overline{y}^{(m)}\right),\\ z_n &= v^{(n)}, \end{aligned}$$

if the boundary condition on $C^{(n)}$ prescribes the derivative $\partial\phi/\partial n$. When this system is solved, each component z_n contains precisely the missing information from $C^{(n)}$ that was not expressed by the boundary condition: if the derivative was specified, (10.62) provides the value of the function, and vice versa. We end up with N values of ϕ and N values of $\partial\phi/\partial n$ on N segments of C. Finally, the solution in the interior of R is obtained by (10.59), in which we set $\lambda = 1$,

$$\phi(\xi,\eta) \approx \sum_{n=1}^{N} \left[v^{(n)} \mathcal{D}^{(n)}(\xi,\eta) - d^{(n)} \mathcal{V}^{(n)}(\xi,\eta) \right], \quad (\xi,\eta) \in R.$$

The elegance of the method has thus been clearly revealed: we obtain the solution on the whole domain R by manipulating information from its boundary C. In constructing $\mathcal{V}(\xi,\eta)$ and $\mathcal{D}(\xi,\eta)$, the line integrals (10.60) and (10.61) need to be computed along individual segments $C^{(n)}$. The segments are parameterized as

$$x(t) = x^{(n)} - t l^{(n)} n_y^{(n)}, \qquad y(t) = y^{(n)} + t l^{(n)} n_x^{(n)}, \quad 0 \le t \le 1. \tag{10.63}$$

Here $l^{(n)}$ is the length of $C^{(n)}$, while the unit vector $\boldsymbol{n}^{(n)} = (n_x^{(n)}, n_y^{(n)})^{\mathrm{T}}$ perpendicular to this segment and pointing away from R has the components

$$n_x^{(n)} = \frac{1}{l^{(n)}} \left(y^{(n+1)} - y^{(n)} \right), \qquad n_y^{(n)} = \frac{1}{l^{(n)}} \left(x^{(n)} - x^{(n+1)} \right).$$

We define

$$\begin{aligned} A^{(n)} &= \left(l^{(n)} \right)^2, \\ B^{(n)}(\xi,\eta) &= 2 l^{(n)} \left(-n_y^{(n)} \left(x^{(n)} - \xi \right) + n_x^{(n)} \left(y^{(n)} - \eta \right) \right), \\ C^{(n)}(\xi,\eta) &= \left(x^{(n)} - \xi \right)^2 + \left(y^{(n)} - \eta \right)^2. \end{aligned}$$

In the parameterization (10.63), the quantity $4A^{(n)}C^{(n)} - (B^{(n)})^2$ is non-negative,

$$F^{(n)} \equiv 4A^{(n)}C^{(n)}(\xi,\eta) - \left[B^{(n)}(\xi,\eta) \right]^2 \ge 0, \quad \forall (\xi,\eta).$$

Elementary integration [12] brings us to the final expressions for the line integrals $\mathcal{V}^{(n)}$ and $\mathcal{D}^{(n)}$, which are distinguished according to the value of $F^{(n)}$. If $F^{(n)} > 0$, we evaluate

$$\begin{aligned} \mathcal{V}^{(n)} = \frac{l^{(n)}}{4\pi} \Bigg\{ & 2\left(\ln l^{(n)} - 1 \right) - \frac{B^{(n)}}{2A^{(n)}} \ln \left| \frac{C^{(n)}}{A^{(n)}} \right| + \left[1 + \frac{B^{(n)}}{2A^{(n)}} \right] \ln \left| 1 + \frac{B^{(n)}}{A^{(n)}} + \frac{C^{(n)}}{A^{(n)}} \right| \\ & + \frac{\sqrt{F^{(n)}}}{A^{(n)}} \left[\arctan \frac{2A^{(n)} + B^{(n)}}{\sqrt{F^{(n)}}} - \arctan \frac{B^{(n)}}{\sqrt{F^{(n)}}} \right] \Bigg\}, \end{aligned}$$

$$\mathcal{D}^{(n)} = \frac{l^{(n)}\left[n_x^{(n)}\left(x^{(n)}-\xi\right)+n_y^{(n)}\left(y^{(n)}-\eta\right)\right]}{\pi\sqrt{F^{(n)}}} \times\left[\arctan\frac{2A^{(n)}+B^{(n)}}{\sqrt{F^{(n)}}}-\arctan\frac{B^{(n)}}{\sqrt{F^{(n)}}}\right].$$

On the other hand, if $F^{(n)}=0$, we need to compute

$$\mathcal{V}^{(n)} = \frac{l^{(n)}}{2\pi}\left\{\ln l^{(n)}+\left[1+\frac{B^{(n)}}{2A^{(n)}}\right]\ln\left|1+\frac{B^{(n)}}{2A^{(n)}}\right|-\frac{B^{(n)}}{2A^{(n)}}\ln\left|\frac{B^{(n)}}{2A^{(n)}}\right|-1\right\},$$

$$\mathcal{D}^{(n)} = 0.$$

The boundary element method is also applicable to non-stationary problems: time integration is performed separately. A superb introduction is given in [12].

10.6 Finite-Element Method ⋆

Finite-element methods (FEM) are five decades old, yet they remain the basic tool of engineers and natural scientists, especially for problems in elastomechanics, hydro- and aero-dynamics in complex geometries. In difference methods we use finite differences to approximate the differential equation, i.e. its space and time derivatives, and impose boundary conditions appropriately. But in complex geometries this is very hard to accomplish.

In the finite-element approach, we find the integral of the differential equation on its definition domain, and thereby express the equation in its variational form. The domain is then divided into smaller units (finite elements) on which the solution of the equation is approximated by a linear combination of some basis functions. The individual elements may be positioned over the domain quite freely (Fig. 10.9), and this liberty represents the main charm and strength of the method. Finally, the variational integral is computed by summing the contributions from all elements; we end up with a system of algebraic equations for the coefficients multiplying the basis functions in the solution expansion.

Here we present the essence of FEM. For greater clarity, the basic concepts are introduced in one space dimension, while in applications the method is overwhelmingly used in two and three dimensions. Further reading can be found in the textbooks [13] and [14] which we follow closely. See also [15].

10.6.1 One Space Dimension

The finite-element method started to bloom in construction problems, so it is fair to introduce it by a civil-engineering example: we are interested in the transverse

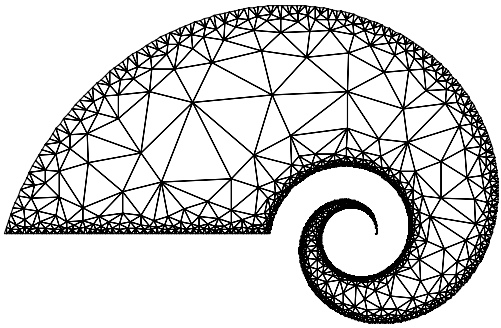


Fig. 10.9 Positioning the finite elements for the problem of propagating acoustic waves in the geometry of a "snail". Such complex geometries are virtually unmanageable by classical difference methods. Figure courtesy of M. Melenk and S. Langdon

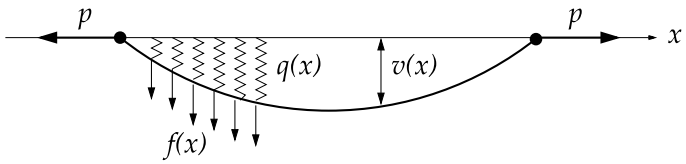


Fig. 10.10 The classical boundary-value problem illustrating the one-dimensional finite-element method: transverse deflections of a cable under non-uniform load

deflections $v(x)$ of a support cable pulled at its ends by the force p. There is another force (per unit length) acting in the transverse direction, $f(x)$, and the cable is held in equilibrium by springs with elastic modulus $q(x)$ (Fig. 10.10).

The deflection $v(x)$ from the horizontal is the solution of the boundary-value problem

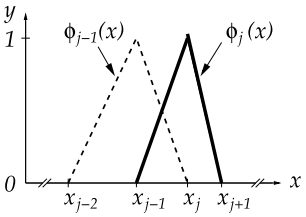
$$-p(x)v''(x) + q(x)v(x) = f(x), \quad x \in [0, 1], \tag{10.64}$$

with boundary conditions $v(0) = v(1) = 0$. For simplicity we assume constant $p > 0$ and $q \geq 0$. We have shown in Sect. 8.6 (see (8.82)) that seeking the solution of this problem is equivalent to determining the function v satisfying the equation

$$A(w, v) = \int_0^1 \left[w' p v' + w q v\right] \mathrm{d}x = \int_0^1 w f \, \mathrm{d}x = \langle w, f \rangle$$

for any weight function w.

We divide the interval $[0, 1]$ to N (not necessarily uniform) subintervals such that $0 = x_0 < x_1 < x_2 < \cdots < x_{N-1} < x_N = 1$. Each subinterval $[x_{j-1}, x_j]$ with length $h_j = x_j - x_{j-1}$ is called the *finite element*. Over the adjacent elements $[x_{j-1}, x_j]$ and $[x_j, x_{j+1}]$ we span the basis function ϕ_j with a peak at x_j and the *nodes* (zeros) at x_{j-1} and x_{j+1} (Fig. 10.11). We use piecewise linear basis functions here; quadratic and cubic functions are also widely used.



$$\phi_j(x) = \begin{cases} \frac{x - x_{j-1}}{x_j - x_{j-1}}; & x_{j-1} \le x < x_j, \\ \frac{x_{j+1} - x}{x_{j+1} - x_j}; & x_j \le x < x_{j+1}, \\ 0; & \text{otherwise.} \end{cases}$$

Fig. 10.11 Basis functions for the one-dimensional finite-element method. The function ϕ_j is non-zero only on the interval $(x_{j-1}, x_j] \cup [x_j, x_{j+1})$, and $\phi_j(x_k) = \delta_{j,k}$

The approximate solution is expanded in terms of the basis functions ϕ_j:

$$u(x) = \sum_{j=1}^{N-1} c_j \phi_j(x). \tag{10.65}$$

Since $\phi_j(x_k) = \delta_{j,k}$, we get $u(x_k) = \sum_j c_j \delta_{j,k} = c_k$. The coefficients c_k are therefore equal to the values of u at the interior nodes. For Dirichlet boundary conditions, $u(x_0) = c_0$ and $u(x_N) = c_N$, so we may include the contributions at the boundary nodes in (10.65) if we set $c_0 = c_N = 0$. The weight function is expanded similarly,

$$\widetilde{w}(x) = \sum_{j=1}^{N-1} d_j \psi_j(x).$$

We stay in the Galerkin framework, where the basis functions ϕ_j for the solution u are the same as the basis functions ψ_j for the weight function $\widetilde{w}$.

The coefficients c_k in the expansion (10.65) are computed by solving the variational equation (8.83), where the sum runs over all finite elements. The equation to be solved is

$$\sum_{j=1}^{N} \big[A_j(\widetilde{w}, u) - \langle \widetilde{w}, f \rangle_j \big] = 0 \quad \forall \widetilde{w}, \tag{10.66}$$

where u is the approximate solution (10.65), $\widetilde{w}$ is the weight function, and f is the right-hand side of (10.64). The subscript j denotes the element $[x_{j-1}, x_j]$ and runs from 1 to N, since all finite elements from $[x_0, x_1]$ through $[x_{N-1}, x_N]$ have to be considered. We split the expression for A_j in two parts,

$$A_j(\widetilde{w}, u) = A_j^{\mathrm{S}}(\widetilde{w}, u) + A_j^{\mathrm{M}}(\widetilde{w}, u) = \int_{x_{j-1}}^{x_j} p\, \widetilde{w}' u' \,\mathrm{d}x + \int_{x_{j-1}}^{x_j} q\, \widetilde{w} u \,\mathrm{d}x. \tag{10.67}$$

Stiffness Matrix, Mass Matrix, and Load Vector In engineering problems the first term of (10.67) corresponds to the internal energy of the body (due to compression or expansion), and the second term to external influences (e.g. due to the

change of potential energy under load). Over the element $[x_{j-1}, x_j]$, the approximate solution and the weight function have the forms

$$u(x) = c_{j-1}\phi_{j-1}(x) + c_j\phi_j(x) = (c_{j-1}, c_j)\begin{pmatrix}\phi_{j-1}(x)\\ \phi_j(x)\end{pmatrix},$$

$$\widetilde{w}(x) = d_{j-1}\phi_{j-1}(x) + d_j\phi_j(x) = (d_{j-1}, d_j)\begin{pmatrix}\phi_{j-1}(x)\\ \phi_j(x)\end{pmatrix},$$

with the derivatives

$$u'(x) = (c_{j-1}, c_j)\begin{pmatrix}-1/h_j\\ 1/h_j\end{pmatrix}, \qquad \widetilde{w}'(x) = (d_{j-1}, d_j)\begin{pmatrix}-1/h_j\\ 1/h_j\end{pmatrix},$$

where $h_j = x_j - x_{j-1}$. From these expressions one finds

$$A_j^{\mathrm{S}}(\widetilde{w}, u) = (d_{j-1}, d_j)\left[\int_{x_{j-1}}^{x_j}\begin{pmatrix}1/h_j^2 & -1/h_j^2\\ -1/h_j^2 & 1/h_j^2\end{pmatrix} p(x)\,\mathrm{d}x\right]\begin{pmatrix}c_{j-1}\\ c_j\end{pmatrix},$$

$$A_j^{\mathrm{M}}(\widetilde{w}, u) = (d_{j-1}, d_j)\left[\int_{x_{j-1}}^{x_j}\begin{pmatrix}\phi_{j-1}^2 & \phi_{j-1}\phi_j\\ \phi_{j-1}\phi_j & \phi_j^2\end{pmatrix} q(x)\,\mathrm{d}x\right]\begin{pmatrix}c_{j-1}\\ c_j\end{pmatrix}.$$

For constant p and q, as assumed for (10.64), only elementary integrals are involved (trivial integration in A_j^{S} and integration of quadratic functions in A_j^{M}). In this case we get

$$A_j^{\mathrm{S}}(\widetilde{w}, u) = (d_{j-1}, d_j)S_j\begin{pmatrix}c_{j-1}\\ c_j\end{pmatrix}, \quad S_j = \frac{p}{h_j}\begin{pmatrix}1 & -1\\ -1 & 1\end{pmatrix}, \tag{10.68}$$

$$A_j^{\mathrm{M}}(\widetilde{w}, u) = (d_{j-1}, d_j)M_j\begin{pmatrix}c_{j-1}\\ c_j\end{pmatrix}, \quad M_j = \frac{qh_j}{6}\begin{pmatrix}2 & 1\\ 1 & 2\end{pmatrix}. \tag{10.69}$$

The matrix S_j is known as the *element (local) stiffness matrix*, and the matrix M_j is the *element (local) mass matrix*.

In general, the integral $\langle\widetilde{w}, f\rangle_j$ in (10.66) (the scalar product of functions $\widetilde{w}$ and f on the jth finite element) is computed numerically. But a good approximation can be obtained if f on $[x_{j-1}, x_j]$ is replaced by a piecewise linear function, $f(x) \approx f_{j-1}\phi_{j-1}(x) + f_j\phi_j(x)$. Then

$$\langle\widetilde{w}, f\rangle_j \approx (d_{j-1}, d_j)\boldsymbol{g}_j, \quad \boldsymbol{g}_j = \frac{h_j}{6}\begin{pmatrix}2f_{j-1} + f_j\\ f_{j-1} + 2f_j\end{pmatrix}. \tag{10.70}$$

The vector $\boldsymbol{g}_j$ is the *element (local) load vector*, as it describes the external force or loading upon the system.

Assembly The next step is the *assembly* of the element matrices into the global stiffness matrix by summing the contributions of all elements, and adding all element load vectors into the global load vector. To simplify, we consider a uniform

mesh, $h_j = h = 1/N$. We sum the contributions (10.68), (10.69), and (10.70) over all elements $1, 2, \ldots, N$, where we consider the boundary condition $c_0 = d_0 = 0$ in the terms involving the element $j = 0$, and $c_N = d_N = 0$ in those involving $j = N$. We collect the coefficients c_j and d_j for the interior points of $[x_0, x_N]$ in the vectors $\boldsymbol{c} = (c_1, c_2, \ldots, c_{N-1})^{\mathrm{T}}$ and $\boldsymbol{d} = (d_1, d_2, \ldots, d_{N-1})^{\mathrm{T}}$, and put the components g_j in the global load vector $\boldsymbol{g} = (g_1, g_2, \ldots, g_{N-1})^{\mathrm{T}}$. The equation with the sum over the elements,

$$\sum_{j=1}^{N} \left[A_j^{\mathrm{S}}(\widetilde{w}, u) + A_j^{\mathrm{M}}(\widetilde{w}, u) \right] = \sum_{j=1}^{N} \langle \widetilde{w}, f \rangle_j ,$$

can then be written in matrix form $\boldsymbol{d}^{\mathrm{T}}[(S + M)\boldsymbol{c} - \boldsymbol{g}] = 0$, where

$$S = \frac{p}{h} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}, \qquad M = \frac{qh}{6} \begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{pmatrix},$$

are the $(N-1) \times (N-1)$ global stiffness and mass matrices, and

$$\boldsymbol{g} = \frac{h}{6} \begin{pmatrix} f_0 + 4f_1 + f_2 \\ f_1 + 4f_2 + f_3 \\ \vdots \\ f_{N-2} + 4f_{N-1} + f_N \end{pmatrix}$$

is the global load vector of dimension $N - 1$ (in the linear approximation on each element). This equation must hold true for any $\boldsymbol{d}$, so the coefficients c_j of the expansion (10.65) for $j = 1, 2, \ldots, N - 1$ are obtained by solving the system

$$(S + M)\boldsymbol{c} = \boldsymbol{g}.$$

10.6.2 Two Space Dimensions

In two dimensions the problem (10.64) is generalized to a planar region R,

$$-\frac{\partial}{\partial x}\left(p(x,y)\frac{\partial v}{\partial x} \right) - \frac{\partial}{\partial y}\left(p(x,y)\frac{\partial v}{\partial y} \right) + q(x,y)v = f(x,y), \quad (x,y) \in R,$$

with the boundary condition $v(x, y) = \alpha(x, y)$ for $(x, y) \in \partial R$. (A problem with a non-homogeneous condition $\alpha \neq 0$ can be turned into the problem with $\alpha = 0$ by shifting $\hat{v} = v - \alpha$.) The variational formulation of the problem is analogous to the one-dimensional case; things start to become complicated when we attempt to divide (partition) the region R to finite elements.

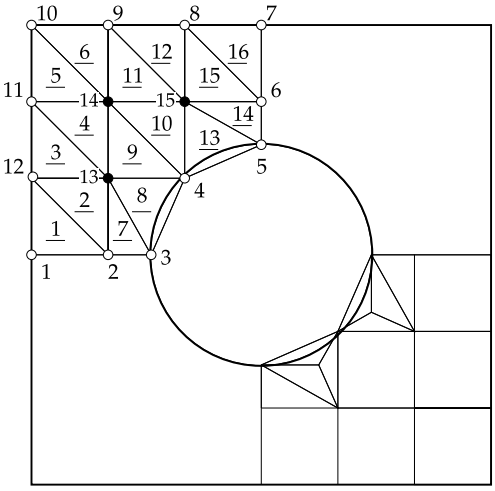


Fig. 10.12 Triangulation for the finite-element method in the geometry of a square with a cut-out circle. *Top left*: a coarse mesh of triangular elements with the enumeration of nodes and elements. *Bottom right*: in a part of the region we are allowed to use other shapes (e.g. squares) or refine the mesh in physically more interesting regions

In one dimension the finite elements are intervals; in a planar region the role of the elements is taken by various geometric shapes. Most frequently one uses (not necessarily equilateral or isosceles) triangles, rarely quadrilaterals (Fig. 10.12). The partitioning of the region is known as triangulation (or quadrangulation). For complex geometries (see Fig. 10.9) it almost amounts to art. Effective triangulation is accomplished by dedicated commercial programs (see e.g. [16]). Among the most well known is the Delaunay triangulation [17, 18]. A good triangulation guarantees that none of the interior angles in the triangles is too small (it maximizes the smallest used angles), thereby ensuring numerical stability.

On the chosen triangulation the corresponding basis functions are defined. Here we restrict the discussion to piecewise linear functions of x and y. The basis function on the nodes of the triangle $\boldsymbol{x}_j = (x_j, y_j)^{\mathrm{T}}$, $\boldsymbol{x}_{j+1} = (x_{j+1}, y_{j+1})^{\mathrm{T}}$, and $\boldsymbol{x}_{j+2} = (x_{j+2}, y_{j+2})^{\mathrm{T}}$, with an apex at $\boldsymbol{x}_j$ (Fig. 10.13(a)) has the form

$$\phi_j(x, y) = \left[\det \begin{pmatrix} 1 & x_j & y_j \\ 1 & x_{j+1} & y_{j+1} \\ 1 & x_{j+2} & y_{j+2} \end{pmatrix} \right]^{-1} \cdot \det \begin{pmatrix} 1 & x & y \\ 1 & x_{j+1} & y_{j+1} \\ 1 & x_{j+2} & y_{j+2} \end{pmatrix}. \tag{10.71}$$

The function ϕ_j is non-zero only over the triangle defined by the nodes $\boldsymbol{x}_j$, $\boldsymbol{x}_{j+1}$, and $\boldsymbol{x}_{j+2}$. It also holds that

$$\phi_j(x_k, y_k) = \delta_{j,k}.$$

Note that (10.71) expresses only the intersection of the planes defining the three side surfaces of the pyramid. For each pair of x and y one needs to check whether they lie inside the triangle defined by the three nodes.

Assembly When assembling the elements of local stiffness and mass matrices in the corresponding global matrices, one must distinguish between the local indices

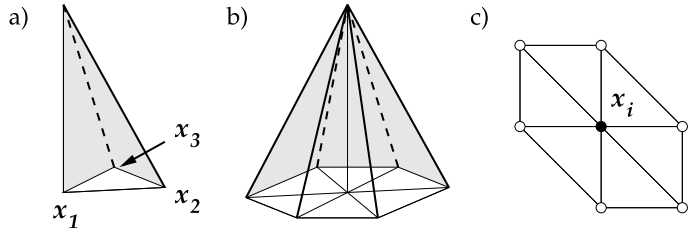


Fig. 10.13 Three-node element functions for the finite-element method on the plane: (**a**) the basic function above the triangular element with the value 1 at the node $\boldsymbol{x}_1$ and values 0 at the side nodes $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$; (**b**) the basis function above six elements; (**c**) the neighborhood of the node $\boldsymbol{x}_i$ as used in Problem 10.9.7

of nodes $m, n \in \{1, 2, 3\}$, and the global indices of nodes $j, k \in \{1, 2, \ldots, N_{\mathcal{N}}\}$ and elements $t \in \{1, 2, \ldots, N_{\mathcal{T}}\}$. Global indices label the whole triangulation with $N_{\mathcal{N}}$ nodes and $N_{\mathcal{T}}$ elements (triangles). In the variational requirement we sum over all triangles,

$$\sum_{t=1}^{N_{\mathcal{T}}} A^{(t)}(\widetilde{w}, u) = \sum_{t=1}^{N_{\mathcal{T}}} \langle \widetilde{w}, f \rangle^{(t)},$$

where t is the label of the triangle T_t. The surface integrals

$$A^{(t)}(\widetilde{w}, u) = \int_{T_t} \left[p(\nabla \widehat{w})^{\mathrm{T}} \cdot \nabla u + q \widetilde{w} u \right] \mathrm{d}x\, \mathrm{d}y, \qquad \langle \widetilde{w}, f \rangle^{(t)} = \int_{T_t} \widetilde{w} f \, \mathrm{d}x\, \mathrm{d}y,$$

are computed on each triangle T_t separately. This is relatively easily done in the Galerkin form of the method with piecewise linear basis functions, where $\phi_j = \psi_j$, as we need only the expressions for the functions ϕ_j (10.71) and their derivatives

$$\nabla \phi_j(x, y) = \frac{1}{2|T|} \begin{pmatrix} y_{j+1} - y_{j+2} \\ x_{j+2} - x_{j+1} \end{pmatrix},$$

where $|T| = \frac{1}{2}|(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)|$ is the surface area of the triangle. The local contributions of triplets of nodes to the stiffness matrix and the corresponding components of the local load vector are then

$$A^{(t)}_{mn}(\widetilde{w}, u) = \int_{T_t} \left[p(\nabla \phi_m)^{\mathrm{T}} \nabla \phi_n + q \phi_m \phi_n \right] \mathrm{d}x\, \mathrm{d}y,$$

$$\langle \widetilde{w}, f \rangle^{(t)}_m = \int_{T_t} \phi_m f \, \mathrm{d}x\, \mathrm{d}y.$$

Example How local matrices and vectors are assembled in the global matrix and global vector is best explained by an example. To make the discussion easier, we set $p = 1$ and $q = 0$ (we are solving the Poisson equation $-\nabla^2 v = f$). In this case the

surface integrals over the triangles T_t are simple,

$$A_{mn}^{(t)} = \frac{1}{4|T|}(y_{m+1} - y_{m+2},\, x_{m+2} - x_{m+1}) \begin{pmatrix} y_{n+1} - y_{n+2} \\ x_{n+2} - x_{n+1} \end{pmatrix},$$

$$\langle \widetilde{w},\, f \rangle_m^{(t)} \approx \frac{1}{6} \det \begin{pmatrix} x_{m+1} - x_m & x_{m+2} - x_m \\ y_{m+1} - y_m & y_{m+2} - y_m \end{pmatrix} f(x_{\mathrm{T}},\, y_{\mathrm{T}}),$$

where $(x_{\mathrm{T}},\, y_{\mathrm{T}})$ are the coordinates of the center of mass of the triangle T_t.

As an illustration of the method, the top left portion of Fig. 10.12 shows a coarse triangulation of a square region with a cut-out circle. This problem (adopted from [14]) is still manageable by classical difference methods, but we use it here to serve as a typical example. We arrange the nodes in the matrix

$$\mathcal{N} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & \cdots & 11 & 12 & 13 & 14 & 15 \\ 0 & 1 & 1.59 & 2 & 3 & \cdots & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1.41 & \cdots & 2 & 1 & 1 & 2 & 2 \end{pmatrix}^{\mathrm{T}}$$

that includes the triplets {label of node, x, y}: here we specify the actual locations of the nodes in the planar region. The matrix

$$\mathcal{T} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & \cdots & 12 & 13 & 14 & 15 & 16 \\ 0 & 1 & 10 & 10 & 9 & \cdots & 7 & 3 & 4 & 5 & 5 \\ 1 & 12 & 11 & 12 & 10 & \cdots & 8 & 4 & 5 & 7 & 6 \\ 11 & 11 & 12 & 13 & 13 & \cdots & 14 & 14 & 14 & 14 & 7 \end{pmatrix}^{\mathrm{T}}$$

contains the quadruplets {label of element, label of node 1, label 2, label 3} relating the elements to the global indices of their nodes. The connection between the space of physical coordinates and the space of elements must be provided in our own code by some mapping between the sets $\mathcal{N}$ and $\mathcal{T}$. It is recommendable that the main loop in the code runs over the triangles $t = 1, 2, \ldots, N_{\mathcal{T}}$, not over the nodes, which would also be possible. Namely, it turns out that fewer surface integrals (evaluated numerically in general) need to be computed by using the element loop than by looping over the nodes [14]. The components of the global stiffness matrix S and the global load vector $\boldsymbol{g}$ are obtained by summing

```
loop over t = 1, 2, ..., N_T
```

$$S_{\mathcal{T}(t,m),\mathcal{T}(t,n)} \mathrel{+}= A_{mn}^{(t)} \qquad (m, n = 1, 2, 3)$$

$$g_{\mathcal{T}(t,m)} \mathrel{+}= \langle \widetilde{w},\, f \rangle_m^{(t)} \qquad (m = 1, 2, 3)$$

```
end loop
```

By solving $S\boldsymbol{c} = \boldsymbol{g}$ (as in the one-dimensional case) we finally obtain the vector $\boldsymbol{c}$ containing the expansion coefficients of the solution, $u(x, y) = \sum_j c_j \phi_j(x, y)$.

Try not to confuse the labellings of the nodes and the elements, since in general the number of nodes $N_{\mathcal{N}}$ is different from the number of elements $N_{\mathcal{T}}$. While

the program loops over the elements t, the values are inserted in the global matrix, and the corresponding load vector according to the labeling of nodes. Compare Fig. 10.13(c) to Fig. 10.17: around the node $\boldsymbol{x}_i$ (in global enumeration) six triangles are arranged, and they form the support for the basis function shown in Fig. 10.13(b). Check your understanding in Problem 10.9.7!

We have merely traced the very first steps of FEM. A genuine expert use of the method only just starts here: non-uniform planar or spatial meshes are devised; basis functions are realized as splines of different degrees so that specific requirements at the boundaries between the elements are met; the mesh can be made denser during the computation if an increase in local precision is called for; discontinuities may be incorporated; non-stationary problems can be implemented; a road opens towards non-linear problems with a multitude of initial and boundary conditions. The finite-element method is already a part of modern numerical packages like MATLAB: nice pedagogical examples that can serve as a good vantage point for further study are given in [19, 20]. A myriad of commercial program packages is available for a more demanding use of the finite-element method [21]. For an excitingly propulsive free version, see [22].

10.7 Mimetic Discretizations ⋆

A powerful tool for solving PDE in complex geometries are the *mimetic (or compatible) discretizations* [23] that attempt to mimic the properties of the physical problem as closely as possible. A nice example is the diffusion in strongly heterogeneous and non-isotropic media described by the equation $v_t = \nabla \cdot D(v)\nabla v + Q$. The space is divided in "logically rectangular" convex cells onto which scalar and vector fields are attached (Fig. 10.14 (left)).

But the key step is the discretization of differential operators, like the diffusion operator $\nabla \cdot D(v)\nabla$ for the problem mentioned above [27, 28]. The discretization should, at least to some order, respect the symmetry and conservative properties of the underlying equation, for example, the conservation of mass, momentum, or energy (in studies of fluid flows) or Maxwell's equations (in electro-magnetic problems). Ultimately, the problems are translated to systems of algebraic equations and solved by preconditioned matrix relaxation methods.

A detailed presentation is beyond the scope of this book, but we let them lurk at the horizon due to their flexibility and development in the recent years. An introduction is given in [23] and the mathematical background in [29].

10.8 Multi-Grid and Mesh-Free Methods ⋆

Two further unique approaches to solving PDE in complex geometries should be mentioned. In the *multi-grid* approach the differential problem is discretized on

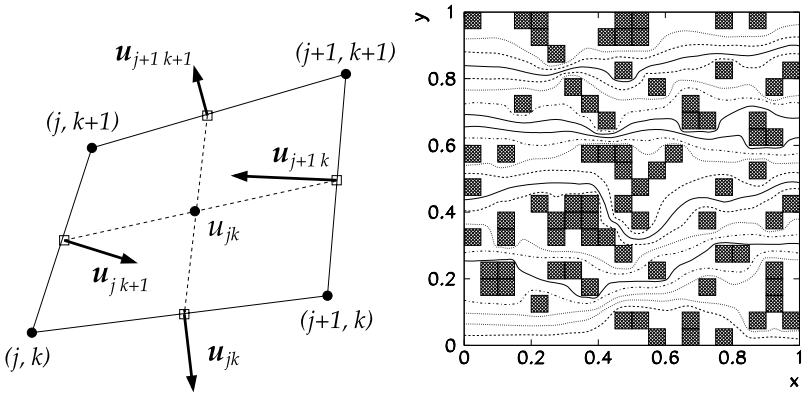


Fig. 10.14 [*Left*] A "logically rectangular" cell in a planar mimetic discretization. The scalar fields u_{jk} are defined at the cell centers, and the vector fields $\boldsymbol{u}$ are given by the components of vectors perpendicular to the cell sides. [*Right*] An example of solving the problem of a fluid flowing through a random arrangement of shale blocks in sand. The shale occupies 20 % of the total surface area (figures adapted from [24–26])

grids (meshes) of varying coarseness. The method is applied on ever finer meshes, and the resulting sequence of solutions converges very rapidly: even in complex geometries and with non-trivial boundary conditions, the multi-grid methods are among the fastest on the market, but each application of the method needs to be tailored carefully to the demands of the individual problem. Multi-grid methods were originally developed for boundary problems involving elliptic PDE, but they can be used to solve other types for PDE as well. An excellent introduction is given by [30]; further reading is offered by [31, 32].

The *mesh-free methods* liberate us from the shackles of discretization: we express the approximate solution as a function spanned over a set of nodes in the definition domain of the differential problem; there is no need for these nodes to be systematically related in any manner (as e.g. in the finite-element method). A certain algorithm leads to a system of equations relating the solution to the information at the nodes and on the domain boundaries. Mesh-free methods excel in problems calling for a large degree of flexibility, for example, in situations with non-fixed boundaries between regions with different physical properties. Examples include phase changes (boundaries between fluid and solid phases of alloys) or mechanical defects (cracks in materials). Especially in three space dimensions, mesh-free methods compete successfully with finite-element methods where adaptive triangulation may become too costly. We recommend [33] and [34] for further reading. In the following we describe the popular mesh-free method based on *radial basis functions*.

10.8.1 A Mesh-Free Method Based on Radial Basis Functions

Here we explain the basic idea of the radial basis functions (RBF) method by solving the Poisson equation on a square with Dirichlet boundary conditions,

$$\nabla^2 v = Q(x, y), \quad (x, y) \in R = [0, 1] \times [0, 1],$$
$$v(x, y) = f(x, y), \quad (x, y) \in \partial R.$$

On the definition domain we choose N points $\{(x_i, y_i)\}_{i=1}^{N}$, of which N_1 are in the interior of the domain, while the remaining $N_2 = N - N_1$ are on its boundary. We seek the approximate solution u in the form

$$u(x, y) = \sum_{j=1}^{N} a_j \phi_j(x, y), \tag{10.72}$$

where ϕ_j are the basis functions. A possible choice for the form of the basis functions is

$$\phi_j(x, y) = \sqrt{(x - x_j)^2 + (y - y_j)^2 + c^2} = \sqrt{r_j^2 + c^2}, \tag{10.73}$$

where c is an adjustable parameter [35, 36]. For such functions we have

$$\frac{\partial^2 \phi_j}{\partial x^2} = \frac{(y - y_j)^2 + c^2}{\left(r_j^2 + c^2\right)^{3/2}}, \qquad \frac{\partial^2 \phi_j}{\partial y^2} = \frac{(x - x_j)^2 + c^2}{\left(r_j^2 + c^2\right)^{3/2}}.$$

At the interior points we insert the ansatz for the solution u in the differential equation, while at the boundary points we insert it into the boundary conditions. This results in a system of linear equations for the coefficients a_j, represented by a $N \times N$ matrix:

$$\sum_{j=1}^{N} \left(\frac{\partial^2 \phi_j}{\partial x^2} + \frac{\partial^2 \phi_j}{\partial y^2} \right)(x_i, y_i) a_j = Q(x_i, y_i), \quad i = 1, 2, \ldots, N_1,$$
$$\sum_{j=1}^{N} \phi_j(x_i, y_i) a_j = f(x_i, y_i), \quad i = N_1 + 1, N_1 + 2, \ldots, N.$$

In the variables x and y, the basis functions (10.73) have a radial symmetry around the collocation points (x_j, y_j) which gave the method its name; but a large set of functions possessing this property is in wide use, for example,

$$\sqrt{r_j^2 + c^2}, \qquad \frac{1}{\sqrt{r_j^2 + c^2}}, \qquad \frac{1}{r_j^2 + c^2}, \qquad \mathrm{e}^{-(c r_j)^2}.$$

Radial basis function methods [37] are suitable for both time-dependent and time-independent problems. Instructive case studies of parabolic problems (diffusion equation) can be found in [38] and for hyperbolic problems (wave equation, Burgers equation) in [39]. Examples of solving elliptic problems (Poisson equation) are discussed in [40].

Example By using the mesh-free method utilizing radial basis functions [35, 36] we solve the Poisson equation on the unit square,

$$-\nabla^2 v = Q, \quad (x, y) \in [0, 1] \times [0, 1],$$

where $Q(x, y) = -13\mathrm{e}^{-2x+3y}$, with Dirichlet boundary conditions

$$v(0, y) = \mathrm{e}^{3y}, \qquad v(1, y) = \mathrm{e}^{-2+3y}, \qquad v(x, 0) = \mathrm{e}^{-2x}, \qquad v(x, 1) = \mathrm{e}^{-2x+3}.$$

The analytic solution is

$$v(x, y) = \mathrm{e}^{-2x+3y}.$$

On the definition domain we randomly choose $N_1 = 16$ collocation points on the boundary of the square (four on each side), and $N_2 = 24$ points in its interior (Fig. 10.15 (left)). In total, we have $N_1 + N_2 = 40$ collocation points, each corresponding to a basis function of the form (10.73), where we choose $c = 0.5$. When the ansatz (10.72) at the collocation points is inserted in the differential equation and in the equations specifying the boundary conditions, we obtain a system of equations for the expansion coefficients a_j, as described on p. 559. We define the absolute error of the numerical solution as

$$\mathrm{ERR} = \left[\sum_{i=1}^{N} \big(u(x_i, y_i) - v(x_i, y_i)\big)^2 \right]^{1/2}.$$

For $c = 0.5$ this error amounts to $\mathrm{ERR} \approx 0.91$ (Fig. 10.15 (right)).

The parameter c defines the scale of the radial basis function, determining whether the function is more spike-like or rather smooth. Obviously, the precision of the solution can be strongly influenced by changing c. Figure 10.15 (right) shows the error ERR in dependence of c: at certain values of c, the error may even explode! Some general instructions about the choice of the optimal c are given in [41].

10.9 Problems

10.9.1 Two-Dimensional Diffusion Equation

The initial-boundary-value problem for the inhomogeneous diffusion equation

$$v_t = D(v_{xx} + v_{yy}) + Q$$

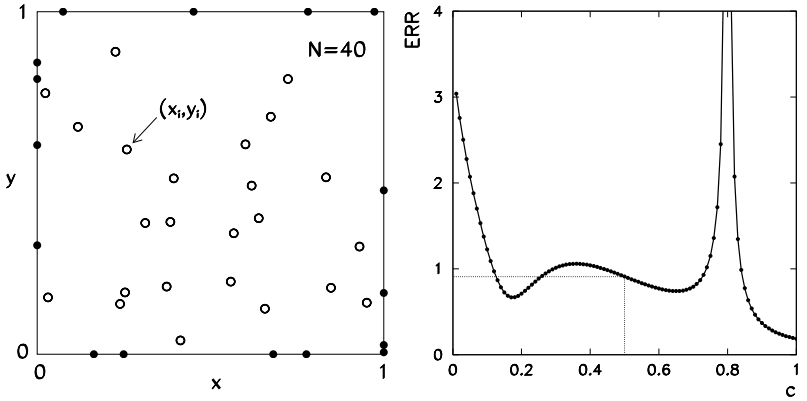


Fig. 10.15 Solving the Poisson equation on the square by using a mesh-free method based on radial basis functions. [*Left*] A random distribution of 16 points on the boundaries of the square (*symbols* •) and 24 points in its interior (*symbols* ∘). [*Right*] The error of the solution in dependence of the parameter c

on $(x, y) \in [0, 1] \times [0, 1]$ is the prototype parabolic problem in two dimensions that can be most easily solved by difference methods (Sect. 10.1). Particular attention should be paid to the consistent treatment of the boundary conditions, so that the order of the difference scheme is not spoiled.

⊙ By using the explicit (FTCS) scheme in two dimensions (10.5), solve the following problems with Dirichlet boundary conditions (examples from [1]):

$$(1) \quad v(x, y, 0) = \sin(\pi x) \sin(2\pi y),$$
$$v(0, y, t) = v(1, y, t) = v(x, 0, t) = v(x, 1, t) = Q(x, y, t) = 0;$$

compute the solutions at times $0 \le t \le 1$ (choose a few representative instants) with step $\Delta t = 0.0005$ (0.001), spatial discretization $N_x = N_y = 20$ and $D = 1.0$. What happens if the discretization in the x coordinate is twice as coarse?

$$(2) \quad v(x, y, 0) = 0,$$
$$v(0, y, t) = v(1, y, t) = v(x, 0, t) = v(x, 1, t) = 0,$$
$$Q(x, y, t) = \sin(2\pi x) \sin(4\pi y) \sin t;$$

compute the solution at times $0 \le t \le 2$ with the discretization $N_x = N_y = 20$, time step $\Delta t = 0.0005$ (or $N_x = N_y = 100$ and $\Delta t = 0.00002$), and parameter $D = 0.5$.

$$(3) \quad v(x, y, 0) = 0,$$
$$v(0, y, t) = \sin(\pi y) \sin t, \qquad v(x, 0, t) = \sin(\pi x) \sin t,$$
$$v(1, y, t) = v(x, 1, t) = Q(x, y, t) = 0;$$

use $N_x = N_y = 20$, $\Delta t = 0.001$, $D = 1.0$, and compute the solution at $0 \le t \le 10$. Compare the numerical result to the analytic one. What can you say about the stability of the schemes? Solve the three problems listed above by using the Crank–Nicolson scheme (10.13).

⊕ We are also interested in the solutions of the homogeneous equation ($Q(x, y, t) = 0$), with a Neumann boundary condition on one side of the square and Dirichlet conditions on the remaining three sides. A consistent inclusion of the boundary conditions is of key importance for the convergence of the chosen difference scheme. Discuss the problem

$$\begin{aligned} v(x, y, 0) &= \sin(\pi x)\sin(2\pi y), \\ v(0, y, t) &= v(x, 0, t) = v(x, 1, t) = 0, \\ v_x(1, y, t) &= g^{\mathrm{N}}(y, t) = -\pi \sin(2\pi y)\mathrm{e}^{-5D\pi^2 t}. \end{aligned}$$

We implement the Neumann conditions at $x = 1$ for the explicit scheme by analogy to (10.11) or (10.12) that apply at $x = 0$. At time $(n + 1)\Delta t$ we get, to first order in the space variable, $u^{n+1}_{N_x k} = u^{n+1}_{N_x - 1k} + \Delta x g^{\mathrm{N}}(k\Delta y, (n + 1)\Delta t)$, which is already the missing equation for the value at $j = N_x$. To second order, we write the Neumann condition at time $n\Delta t$ as $u^n_{N_x+1k} = u^n_{N_x-1k} + 2\Delta x g^{\mathrm{N}}(k\Delta y, n\Delta t)$. We insert the value $u^n_{N_x+1k}$ in the difference scheme at $j = N_x$, whence we again obtain the missing equation at $j = N_x$,

$$u^{n+1}_{N_x k} = (1 - 2r_x)u^n_{N_x k} + 2r_x u^n_{N_x - 1k} + r_y \Delta_2^{(y)} u^n_{N_x k} + 2r_x \Delta x g^{\mathrm{N}}(k\Delta y, n\Delta t).$$

For the Peaceman–Rachford scheme, the first-order Neumann condition is imposed in the same manner. The second-order condition is easier to compute if the order of spatial derivatives in the scheme is interchanged, i.e. by just substituting $x \longleftrightarrow y$ in (10.14) and (10.15). As in the implicit case, the condition is inserted in the scheme at $j = N_x$, but at time $(n + 1)\Delta t$. We obtain

$$\begin{aligned} u^{n+1}_{N_x k} = {} & \frac{r_x}{1 + r_x} u^{n+1}_{N_x - 1k} + \frac{1}{1 + r_x}\left(1 + \frac{r_y}{2}\Delta_2^{(y)}\right) u^{n+\frac{1}{2}}_{N_x k} \\ & + \frac{r_x}{1 + r_x}\Delta x g^{\mathrm{N}}\big(k\Delta y, (n + 1)\Delta t\big). \end{aligned}$$

In the D'yakonov variant of the scheme, the Neumann condition needs to be translated into a condition for $u^\star$. Check that for the first-order Neumann condition this means

$$u^\star_{N_x k} = u^\star_{N_x - 1k} + \Delta x\left(1 - \frac{r_x}{2}\Delta_2^{(x)}\right) g^{\mathrm{N}}\big(k\Delta y, (n + 1)\Delta t\big),$$

while for the second-order Neumann condition one should use

$$u^{\star}_{N_x k} = \frac{r_y}{1+r_y} u^{\star}_{N_x-1k} + \frac{1}{1+r_y}\left(1+\frac{r_y}{2}\Delta_2^{(y)}\right)\left(1+\frac{r_x}{2}\Delta_2^{(x)}\right)u^n_{N_x k} + \frac{r_y}{1+r_y}\Delta x\, g^{\mathrm{N}}\big(k\Delta y, (n+1)\Delta t\big).$$

Compute the solution by using the explicit scheme (10.5) at times $0 \le t \le 1$ with the step size $\Delta t = 0.001$ and discretization $N_x = N_y = 10$, and then with $\Delta t = 0.0005$, $N_x = N_y = 20$ and $\Delta t = 0.0001$, $N_x = N_y = 40$. Use $D = 1.0$. Repeat the exercise with the Peaceman–Rachford and D'yakonov scheme with $\Delta t = 0.01$. In all cases use first- and second-order approximations for the Neumann boundary condition. Compare the analytic and numerical solutions.

10.9.2 Non-linear Diffusion Equation

This problem acquaints us with a method of solving a diffusion-reaction problem from the field of biotechnology: we are interested in the formation and growth of bacterial biofilms on a nutritious substrate [42, 43]. The density of the created biomass $v(x, y, t)$ is determined by the partial differential equation

$$v_t = \nabla\big[D(v)\nabla v\big] + \kappa v, \quad (x, y) \in [0, 1] \times [0, 0.3],$$

where

$$D(v) = \delta\frac{v^b}{(1-v)^a}, \quad 0 < \delta \ll 1 \le a, b.$$

The quantity κ drives the rate of biofilm formation: a constant $\kappa > 0$ implies unlimited nutrients. In this case the bacterial culture spreads until it attains a homogeneous distribution (Fig. 10.16).

We monitor the approximate density of the biofilm $u(j\Delta x, k\Delta y, n\Delta t)$ on the uniform mesh with $\Delta x = 1/N_x$ and $\Delta y = 0.3/N_y$. We discretize the time derivative to second order, $v_t \approx (u^{n+1}_{jk} - u^n_{jk})/\Delta t$, while the reaction term κv is evaluated at time $(n+1)\Delta t$. For the diffusion term we apply the difference formula

$$\begin{aligned}\nabla\big[D(v)\nabla v\big] \approx{}& \frac{1}{2\Delta x^2}\Big\{\big[D\big(u^n_{jk}\big) + D\big(u^n_{j+1k}\big)\big]\big[u^{n+1}_{j+1k} - u^{n+1}_{jk}\big] \\ &\times \big[D\big(u^n_{jk}\big) + D\big(u^n_{j-1k}\big)\big]\big[u^{n+1}_{j-1k} - u^{n+1}_{jk}\big]\Big\} \\ &+ \frac{1}{2\Delta y^2}\Big\{\big[D\big(u^n_{jk}\big) + D\big(u^n_{jk+1}\big)\big]\big[u^{n+1}_{jk+1} - u^{n+1}_{jk}\big] \\ &\times \big[D\big(u^n_{jk}\big) + D\big(u^n_{jk-1}\big)\big]\big[u^{n+1}_{jk-1} - u^{n+1}_{jk}\big]\Big\}.\end{aligned}$$

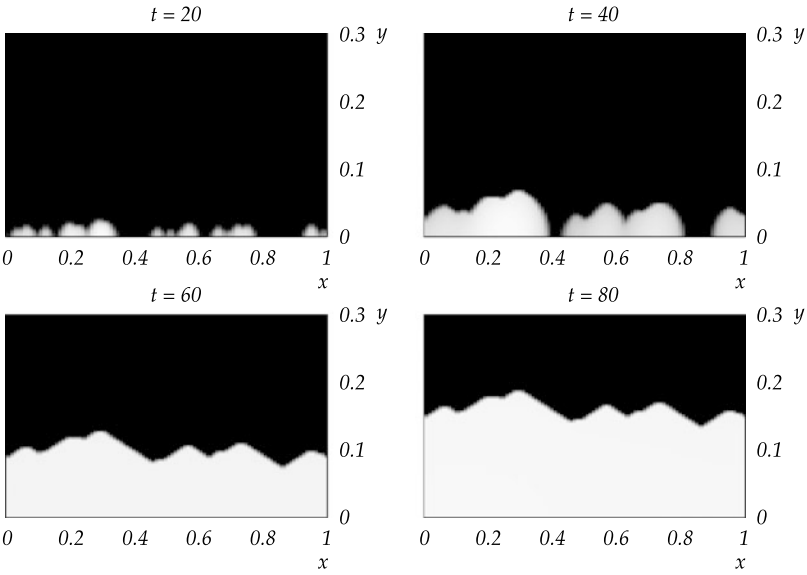


Fig. 10.16 Formation of the bacterial layer from an initial random distribution at $y = 0$. An unlimited supply of nutrients is available in the model, so the blotches grow, merge, and gradually expand to a homogeneous distribution of maximum density

If D is constant, this formula turns into the usual discretization of $\nabla^2 v$ in Cartesian coordinates (see (10.2) and (10.3)). Beware that the diffusion coefficients depend on the current values of u at times $n\Delta t$, while the values of the functions at the right-hand side are given at $(n+1)\Delta t$. The difference scheme is therefore implicit and does not suffer from stability problems.

⊙ Carefully join the terms v_t, $\nabla[D(v)\nabla v]$, and κv to form an implicit difference scheme that can be written in matrix form as

$$A\boldsymbol{u}^{n+1} = \boldsymbol{u}^n.$$

Here $\boldsymbol{u}^n$ is the solution vector at time $n\Delta t$ with the coordinates indexed as in (10.1). Solve the problem by using the parameters $a = b = 4$, $\kappa = 0.1$, and $\delta = 0.1$. Start with the spatial discretization $N_x = N_y = 60$ and the time step $\Delta t = 1$. Attention: even with such a coarse discretization, the matrix A has the size $(N_x+1)(N_y+1) \times (N_x+1)(N_y+1) = 3721 \times 3721$. Fortunately it is banded: only the diagonal, the first sub- and super-diagonal, and the (N_x+1)th sub- and super-diagonals have non-zero entries, so the computation is much faster if one resorts to methods for banded matrices, e.g. `DGBSV` from the LAPACK library (see Sect. 3.2.3).

The initial condition is N_i randomly distributed values of the density between 0 and 1 that appear at time $t = 0$ along $y = 0$ in $N_\mathrm{i} \approx N_x/2$ intervals of the mesh; everywhere else the density should be zero. Impose Neumann boundary conditions $v_x(x = 0) = v_x(x = 1) = v_y(y = 0) = 0$ on three boundaries of the domain, and a Dirichlet condition $v(y = 0.3) = 0$ on the fourth boundary. Compute the solution

until $t \approx 50$. If the speed of your computer (or the algorithm to solve the banded system) allows it, gradually increase N_x and N_y.

10.9.3 Two-Dimensional Poisson Equation

We discuss the elliptic boundary problem (adapted from [2])

$$\nabla^2 v = 2\pi^2 \big(\sin(\pi x)\cos(\pi y) + \cos(\pi x)\sin(\pi y)\big) \mathrm{e}^{\pi(x+y)}$$

on $(x, y) \in R = [0, 1] \times [0, 1]$ with Dirichlet boundary condition

$$v = 0, \quad (x, y) \in \partial R,$$

that has the analytic solution $v(x, y) = \sin(\pi x)\sin(\pi y)\mathrm{e}^{\pi(x+y)}$.

⊙ Discretize the problem on the unit square $(x, y) \in [0, 1] \times [0, 1]$ with a uniform mesh $N_x = N_y = 128$ and solve it by using the Jacobi (10.41) and Gauss–Seidel (10.42) iteration. Terminate the iteration when the value of the sup-norm (A.3) or the energy norm (A.5) of the difference between subsequent solution vectors (10.46) or the residual error (10.47) drops below a certain value, say, 10^{-6}. How does the number of iteration steps in both methods change when you refine the discretization (increase $N_x = N_y = N$)?

⊕ On the domain $(x, y) \in R = [0, 1] \times [0, 1]$, solve the following boundary problem:

$$\nabla^2 v = \mathrm{e}^{x+y} = Q(x, y)$$

with Neumann boundary conditions $(\partial v/\partial n)(x, y) = g(x, y)$, in detail:

$$\frac{\partial v}{\partial x}(0, y) = \frac{1}{2}\mathrm{e}^{y}, \qquad \frac{\partial v}{\partial y}(x, 0) = \frac{1}{2}\mathrm{e}^{x},$$
$$\frac{\partial v}{\partial x}(1, y) = \frac{1}{2}\mathrm{e}^{1+y}, \qquad \frac{\partial v}{\partial y}(x, 1) = \frac{1}{2}\mathrm{e}^{x+1}.$$

The operator side of the equation is discretized as in (10.35), while the boundary conditions are approximated at first order,

$$u_{1k} = u_{0k} + \Delta x g_{0k}, \qquad u_{j1} = u_{j0} + \Delta y g_{j0},$$
$$u_{N_x k} = u_{N_x - 1 k} + \Delta x g_{N_x k}, \qquad u_{jN_y} = u_{jN_y - 1} + \Delta y g_{jN_y},$$

where $j = 1, 2, \ldots, N_x - 1$ and $k = 1, 2, \ldots, N_y - 1$. We rewrite the complete system of equations (including the boundary conditions) as $A\boldsymbol{u} = \boldsymbol{q}$, where $\boldsymbol{u}$ is the solution vector of the form (10.1) that does not contain components $j = 0$, $j = N_x$,

$k = 0$, and $k = N_y$ (its dimension is $(N_x - 1)(N_y - 1)$). The matrix of the system,

$$A = \begin{pmatrix} T_1 & -Y & 0 & & \\ -Y & T & -Y & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & -Y & T & -Y \\ & & 0 & -Y & T_1 \end{pmatrix},$$

is symmetric block-tridiagonal with $(N_y - 1) \times (N_y - 1)$ blocks, where the non-zero blocks lie only along the diagonal and the first sub- and super-diagonal. The matrices T_1, T, and Y have dimension $(N_x - 1) \times (N_x - 1)$. By abbreviating $a = 1/\Delta x^2$ and $b = 1/\Delta y^2$, the matrices T_1 and T are

$$T_1 = \begin{pmatrix} a+b & -a & 0 & & \\ -a & 2a+b & -a & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & -a & 2a+b & -a \\ & & 0 & -a & a+b \end{pmatrix}$$

and

$$T = \begin{pmatrix} a+2b & -a & 0 & & \\ -a & 2a+2b & -a & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & -a & 2a+2b & -a \\ & & 0 & -a & a+2b \end{pmatrix},$$

while $Y = (1/\Delta y^2)I$. The right-hand side of the equation is a vector of dimension $(N_x - 1) \times (N_y - 1)$,

$$\boldsymbol{q} = \boldsymbol{Q} + \boldsymbol{b}_x + \boldsymbol{b}_y,$$

with three contributions: the source term and two terms picking up those parts of the expressions for boundary conditions that have not been absorbed in the matrix A. The components are arranged in the usual "$j - k$" ordering:

$$\boldsymbol{Q} = (Q_{11}, Q_{21}, \ldots, Q_{N_x-11}, Q_{12}, Q_{22}, \ldots, Q_{1N_y-1}, Q_{2N_y-1}, \ldots, Q_{N_x-1N_y-1})^{\mathrm{T}},$$

$$\boldsymbol{b}_x = \frac{1}{\Delta x}(g_{01}, 0, \ldots, 0, g_{N_x1}, g_{02}, 0, \ldots, g_{0N_y-1}, 0, \ldots, 0, g_{N_xN_y-1})^{\mathrm{T}},$$

$$\boldsymbol{b}_y = \frac{1}{\Delta y}(g_{10}, g_{20}, \ldots, 0, g_{N_x-10}, 0, 0, \ldots, g_{1N_y}, g_{2N_y}, \ldots, 0, g_{N_x-1N_y})^{\mathrm{T}}.$$

Discretize the problem on a reasonably fine uniform mesh with $N_x, N_y \approx 100$. Solve the matrix equation by the Gauss–Seidel and SOR method. Because the boundary conditions have been discretized only to first order, the convergence of the solution will also be just of order $\mathcal{O}(\Delta x) + \mathcal{O}(\Delta y)$. Use the convergence tolerance of $\approx 10^{-6}$

in both schemes. The problem can be simplified by setting $N_x = N_y = N$, since in this case the value of the optimal relaxation parameter ω for the SOR scheme can be determined by

$$\omega_{\mathrm{b}} = \frac{2}{1 + \sqrt{1 - \frac{1}{4}\left(1 + \cos\frac{\pi}{N}\right)^2}}.$$

10.9.4 High-Resolution Schemes for the Advection Equation

This Problem (taken from [2]) involves the two-dimensional advection equation

$$v_t + c(x, y)v_x + d(x, y)v_y = 0, \quad (x, y) \in [0, 1] \times [0, 1],$$

with the initial condition

$$v(x, y, 0) = f(x, y) = \begin{cases} 1; & (x, y) \in [\frac{1}{4}, \frac{3}{4}] \times [\frac{1}{4}, \frac{3}{4}], \\ 0; & \text{otherwise.} \end{cases}$$

The analytic solution $v(x, y, t) = f(x - ct, y - dt)$ at constant c and d does not bother about the discontinuity in the initial condition: it just advances the initial "jump" in time. In contrast, propagation of discontinuities in difference schemes is plagued by typical annoyances discussed in this Problem.

$\odot$ Set $c = d = 1$ and impose periodic boundary conditions $v(0, y, t) = v(1, y, t)$ for $y \in [0, 1]$ and $v(x, 0, t) = v(x, 1, t)$ for $x \in [0, 1]$. Use the scheme (10.28) with the flux functions (10.29) and discretization $N_x = N_y = 100$ ($\Delta x = 0.01$). Compute the solution until $t = 1.0$ in time steps of $\Delta t = 0.002$. In same conditions, apply the split Lax–Wendroff method (see (10.30) and (10.31)) with the flux functions (10.32) and (10.33). Compute the solution until $t = 0.2$. Finally, solve the problem by using the flux functions (10.48) and (10.49); once in the scheme (10.28), and once in the split Lax–Wendroff scheme. Compare the solutions and describe their behavior.

$\bigoplus$ For all schemes mentioned above, non-constant coefficients c and d are a much harder nut to crack. You can see that by running the programs from the first part of this Problem with a pair of seemingly harmless coefficient functions

$$c(x, y) = \sqrt{2}\left(y - \frac{1}{2}\right), \qquad d(x, y) = \sqrt{2}\left(x - \frac{1}{2}\right).$$

This part of the Problem can be solved well by the high-resolution Zalesak–Smolarkiewicz method described in Sect. 10.3.

10.9.5 Two-Dimensional Diffusion Equation in Polar Coordinates

Here we try to solve the diffusion equation [2]

$$v_t = \frac{1}{r}(r v_r)_r + \frac{1}{r^2} v_{\theta\theta}, \quad v = v(r, \theta, t),$$

on $(r, \theta) \in [0, 1] \times [0, 2\pi]$ with pairs of initial and boundary conditions

$$v(r, \theta, 0) = 0, \qquad v(1, \theta, t) = \sin(4\theta) \sin t,$$

or

$$v(r, \theta, 0) = \left(1 - r^2\right) \sin(2\theta), \qquad v(1, \theta, t) = 0.$$

⊙ Use the explicit scheme (10.54) and (10.55) with $N_r = 20$, $N_\theta = 32$, $\Delta t = 0.001$. For the first condition pair, compute the solution at $t = 0.1, 0.5, 1.5, 3.0$, and 6.0; for the other pair, compute it at $t = 0.1, 0.5$, and 1.0.

10.9.6 Two-Dimensional Poisson Equation in Polar Coordinates

We would like to solve the Poisson equation in planar polar coordinates

$$-\nabla^2 v = Q,$$

by considering in two geometries (an annulus and a disk including the origin) and different boundary conditions. (This problem is adapted from [2].)

⊙ In the annular geometry (Fig. 10.7 (right)), let $a = 0.1$ and $Q(r, \theta) = 0$, and impose boundary conditions $f_1(\theta) = \sin 2\theta$ and $f_2(\theta) = \sin 3\theta$ for $\theta \in [0, 2\pi]$. Solve the problem by using the difference scheme described in Sect. 10.4.2. Use the discretization $N_r = N_\theta = 20$ and solve the resulting matrix system by using the Gauss–Seidel method. Repeat the exercise by using a finer discretization $N_r = N_\theta = 100$ with the Gauss–Seidel and the optimal SOR method. In addition, use SOR with $N_r = N_\theta = 100$, $a = 0.5$, $Q(r, \theta) = \exp(r) \sin 2\pi\theta$, and boundary conditions $f_1(\theta) = \sin 4\theta$, $f_2(\theta) = \sin 3\theta$ for $\theta \in [0, 2\pi]$. Examine the difference between the subsequent solutions (10.46) or the residual (10.47) to ascertain convergence.

⊕ Solve the Poisson equation in the disk geometry (Fig. 10.7 (left)). Discuss the case $Q(r, \theta) = 0$ with the boundary condition $f_2(\theta) = \sin 2\theta$ for $\theta \in [0, 2\pi]$. Use the Jacobi and SOR schemes with $N_r = N_\theta = 20$. Additionally, use the SOR method to solve the problem with $Q(r, \theta) = \cos(\pi r) \cos(2\pi\theta)$ and the boundary condition $f_2(\theta) = \sin 4\theta$ on the mesh $N_r = N_\theta = 100$. Use the quantities (10.46) and (10.47) when determining when to stop the iteration.

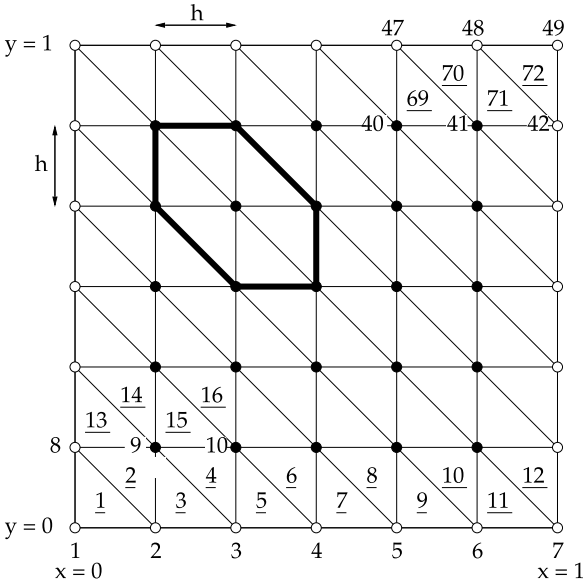


Fig. 10.17 Regular triangulation for solving the Poisson equation on the square by the finite-element method. The indices of the elements are underlined; the indices of the nodes are not (only one of the many options is drawn)

10.9.7 Finite-Element Method

We would like to use the finite-element method to solve the Poisson equation

$$-\nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = f(x, y), \quad (x, y) \in R = [0, 1] \times [0, 1],$$

with $f(x, y) = 8\pi^2 \sin(2\pi x) \sin(2\pi y)$ and homogeneous Dirichlet boundary conditions on all sides of R. The analytic solution is $v(x, y) = \sin(2\pi x) \sin(2\pi y)$. (Example adapted from [14].)

⊙ First, use uniform triangulation with the lengths of intervals h on each axis (Fig. 10.17), in which you span piecewise continuous basis functions on the individual elements, as described in Sect. 10.6. Set $h = 0.1$, 0.01, and 0.001. In all cases, compute the error

$$\left[\int_R |\nabla v - \nabla u|^2 \, \mathrm{d}\boldsymbol{x} \right]^{1/2}.$$

⊕ Solve the problem by using a different enumeration of nodes and/or elements. By doing this, you obtain a different stiffness matrix and a different load vector. Is the matrix sparse? How does this influence the speed at which the matrix system can be solved?

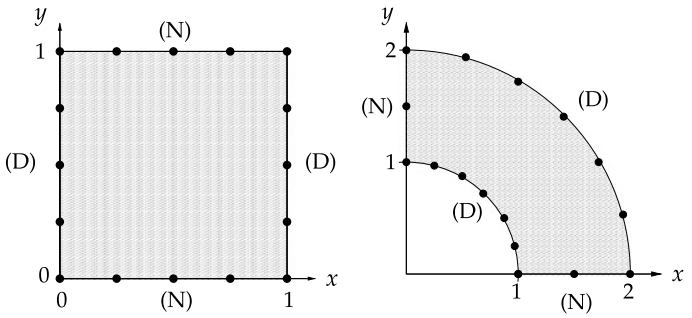


Fig. 10.18 [*Left*] Square geometry for the solution of the planar Laplace equation. The letters at the boundaries denote the type of the boundary condition: (D) Dirichlet, (N) Neumann. [*Right*] The geometry of a section of an annulus

10.9.8 Boundary Element Method for the Two-Dimensional Laplace Equation

In this example (adapted from [12]) we are solving the two-dimensional Laplace equation

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

in the geometry of a square (see Fig. 10.18 (left)).

Along $x = 0$ and $x = 1$, impose Dirichlet boundary conditions, while along $y = 0$ and $y = 1$, impose Neumann conditions:

$$\begin{aligned}
&\phi = 0 \quad \text{on boundary } x = 0 \text{ for } 0 < y < 1,\\
&\phi = \cos(\pi y) \quad \text{on boundary } x = 1 \text{ for } 0 < y < 1,\\
&\frac{\partial \phi}{\partial n} = 0 \quad \text{on boundaries } y = 0 \text{ and } y = 1 \text{ for } 0 < x < 1.
\end{aligned}$$

The analytic solution is

$$\phi(x, y) = \frac{\sinh(\pi x)\cos(\pi y)}{\sinh \pi}.$$

⊙ Use the boundary element method to solve the Laplace equation on the square with the conditions specified above. Divide each side of the square into N_1 boundary elements, so that the boundary points are at

$$\begin{aligned}
&\left(x^{(n)}, y^{(n)}\right) = \left((n-1)l, 0\right) && \text{(bottom boundary)},\\
&\left(x^{(N_1+n)}, y^{(N_1+n)}\right) = \left(1, (n-1)l\right) && \text{(right boundary)},
\end{aligned}$$

$$\left(x^{(2N_1+n)}, y^{(2N_1+n)}\right) = \left(1-(n-1)l, 1\right) \quad \text{(top boundary)},$$
$$\left(x^{(3N_1+n)}, y^{(3N_1+n)}\right) = \left(0, 1-(n-1)l\right) \quad \text{(left boundary)},$$

for $n = 1, 2, \ldots, N_1$, where $l = 1/N_1$ is the length of each element. We have a total of $N = 4N_1$ boundary elements, and clearly

$$\left(x^{(N+1)}, y^{(N+1)}\right) = \left(x^{(1)}, y^{(1)}\right).$$

$\bigoplus$ Use the boundary element method to solve the Laplace equation in the geometry shown in Fig. 10.18 (right). Impose Neumann conditions on the straight boundaries of the domain, and Dirichlet conditions on the circular arcs:

$$\frac{\partial \phi}{\partial n} = 0 \quad \text{on boundary } x = 0 \text{ for } 1 < y < 2,$$
$$\frac{\partial \phi}{\partial n} = 0 \quad \text{on boundary } y = 0 \text{ for } 1 < x < 2,$$
$$\phi = \cos\left(4 \arctan \frac{y}{x}\right) \quad \text{on arc } x^2 + y^2 = 1 \text{ for } x, y > 0,$$
$$\phi = 3\cos\left(4 \arctan \frac{y}{x}\right) \quad \text{on arc } x^2 + y^2 = 4 \text{ for } x, y > 0.$$

The analytic solution is

$$\phi(x, y) = \left\{\frac{16}{85}\left[r^4 - \frac{1}{r^4}\right] - \frac{16}{255}\left[\frac{r^4}{16} - \frac{16}{r^4}\right]\right\} \cos\left(4 \arctan \frac{y}{x}\right),$$

where $r^2 = x^2 + y^2$. Divide the straight sections of the boundary into N_1, the exterior arc into $8N_1$, and the interior arc into $2N_1$ elements, so that the total number of the boundary elements is $N = 12N_1$:

$$\left(x^{(n)}, y^{(n)}\right) = \left(1 + \frac{n-1}{N_1}, 0\right), \qquad n = 1, 2, \ldots, N_1,$$
$$\left(x^{(N_1+n)}, y^{(N_1+n)}\right) = \left(2\cos\frac{(n-1)\pi}{16N_1}, 2\sin\frac{(n-1)\pi}{16N_1}\right), \quad n = 1, 2, \ldots, 8N_1,$$
$$\left(x^{(9N_1+n)}, y^{(9N_1+n)}\right) = \left(0, 2 - \frac{n-1}{N_1}\right), \qquad n = 1, 2, \ldots, N_1,$$
$$\left(x^{(10N_1+n)}, y^{(10N_1+n)}\right) = \left(\sin\frac{(n-1)\pi}{4N_1}, \cos\frac{(n-1)\pi}{4N_1}\right), \quad n = 1, 2, \ldots, 2N_1.$$

References

1. J.W. Thomas, *Numerical Partial Differential Equations: Finite Difference Methods*. Springer Texts in Applied Mathematics, vol. 22 (Springer, Berlin, 1998)

2. J.W. Thomas, *Numerical Partial Differential Equations: Conservation Laws and Elliptic Equations*. Springer Texts in Applied Mathematics, vol. 33 (Springer, Berlin, 1999)
3. W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 3rd edn. (Cambridge University Press, Cambridge, 2007). See also the equivalent handbooks in Fortran, Pascal and C, as well as http://www.nr.com
4. R.A. Horn, C.R. Johnson, *Matrix Analysis* (Cambridge University Press, Cambridge, 1985)
5. J.W. Demmel, *Applied Numerical Linear Algebra* (SIAM, Philadelphia, 1997)
6. J.R. Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain* (Carnegie Mellon University, Pittsburgh, 1994) (unpublished, but accessible through numerous websites)
7. G.H. Golub, C.F. Van Loan, *Matrix Computations*, 3rd edn. (Johns Hopkins University Press, Baltimore, 1996)
8. S.T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids. J. Comput. Phys. **31**, 335 (1979)
9. P.K. Smolarkiewicz, A fully multidimensional positive definite advection transport algorithm with small implicit diffusion. J. Comput. Phys. **54**, 325 (1984)
10. N.A. Peterson, An algorithm for assembling overlapping grid systems. SIAM J. Sci. Comput. **20**, 1995 (1999)
11. W.D. Henshaw, On multigrid for overlapping grids. SIAM J. Sci. Comput. **26**, 1547 (2005)
12. W.T. Ang, *A Beginner's Course in Boundary Element Methods* (Universal, Boca Raton, 2007)
13. J.E. Flaherty, *Finite Element Analysis*. CSCI, MATH Lecture Notes, vol. 6860 (Rensselaer Polytechnic Institute, Troy, 2000)
14. Z. Chen, *Finite Element Methods and Their Applications* (Springer, Berlin, 2005)
15. M.S. Gockenbach, *Understanding and Implementing the Finite Element Method* (SIAM, Philadelphia, 2006)
16. Computational Geometry Algorithms Library. http://www.cgal.org. The algorithms from this library are also built into MATLAB
17. M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd edn. (Springer, Berlin, 2008)
18. J.R. Shewchuk, Delaunay refinement algorithms for triangular mesh generation. Comput. Geom. **22**, 21 (2002)
19. J. Alberty, C. Carstensen, S.A. Funken, Remarks around 50 lines of Matlab: short finite element implementation. Numer. Algorithms **20**, 117 (1999)
20. J. Alberty, C. Carstensen, S.A. Funken, R. Klose, Matlab implementation of the finite element method in elasticity. Computing **69**, 239 (2002)
21. http://en.wikipedia.org/wiki/List_of_finite_element_software_packages
22. F. Hecht, O. Pironneau, J. Morice, A. Le Hyaric, K. Ohtsuka, FreeFem++. http://www.freefem.org/ff++
23. D. Knoll, J. Morel, L. Margolin, M. Shashkov, Physically motivated discretization methods. Los Alamos Sci. **29**, 188 (2005)
24. M. Shashkov, S. Steinberg, Solving diffusion equations with rough coefficients on rough grids. J. Comput. Phys. **129**, 383 (1996)
25. J. Hyman, M. Shashkov, S. Steinberg, The numerical solution of diffusion problems in strongly heterogeneous non-isotropic materials. J. Comput. Phys. **132**, 130 (1997)
26. J. Hyman, M. Shashkov, Mimetic discretizations for Maxwell's equations. J. Comput. Phys. **151**, 881 (1999)
27. J. Hyman, J. Morel, M. Shashkov, S. Steinberg, Mimetic finite difference methods for diffusion equations. Comput. Geosci. **6**, 333 (2002)
28. Y. Kuznetsov, K. Lipnikov, M. Shashkov, The mimetic finite difference method on polygonal meshes for diffusion-type problems. Comput. Geosci. **8**, 301 (2004)
29. P. Bochev, J. Hyman, Principles of mimetic discretizations of differential operators, in *Compatible Spatial Discretizations*, ed. by D.N. Arnold, P.B. Bochev, R.B. Lehoucq, R.A. Nicolaides, M. Shashkov. The IMA Volumes in Mathematics and Its Applications, vol. 142 (Springer, Berlin, 2006), p. 89

30. Multiple authors, Special issue of Comput. Sci. Eng. **Nov/Dec** (2006)
31. P. Wesseling, *An Introduction to Multigrid Methods* (Edwards, Philadelphia, 2004)
32. W.L. Briggs, H. van Emden, S.F. McCormick, *A Multigrid Tutorial*, 2nd edn. (SIAM, Philadelphia, 2000)
33. S. Li, W.K. Liu, *Mesh-Free Particle Methods* (Springer, Berlin, 2004)
34. G.R. Liu, *Mesh-Free Methods: Moving Beyond the Finite-Element Method* (CRC Press, Boca Raton, 2003)
35. E.J. Kansa, Multiquadrics—a scattered data approximation scheme with applications to computational fluid dynamics, I: surface approximations and partial derivative estimates. Comput. Math. Appl. **19**, 127 (1990)
36. E.J. Kansa, Multiquadrics—a scattered data approximation scheme with applications to computational fluid dynamics, II: solutions of parabolic, hyperbolic and elliptic partial differential equations. Comput. Math. Appl. **19**, 147 (1990)
37. N. Flyer, B. Fornberg, Radial basis functions: developments and applications to planetary scale flows. Comput. Fluids **46**, 23 (2011)
38. M. Tatari, M. Dehghan, A method for solving partial differential equations via radial basis functions: application to the heat equation. Eng. Anal. Bound. Elem. **34**, 206 (2010)
39. E.J. Kansa, Exact explicit time integration of hyperbolic partial differential equations with mesh free radial basis functions. Eng. Anal. Bound. Elem. **31**, 577 (2007)
40. E. Larsson, B. Fornberg, A numerical study of some radial basis function based solution methods for elliptic PDEs. Comput. Math. Appl. **46**, 891 (2003)
41. C.-S. Huang, H.-D. Yen, A.H.-D. Cheng, On the increasingly flat radial basis function and optimal shape parameter for the solution of elliptic PDEs. Eng. Anal. Bound. Elem. **34**, 802 (2010)
42. H.J. Eberl, L. Demaret, A finite difference scheme for a degenerated diffusion equation arising in microbial biology. Electron. J. Differ. Equ. **15**, 77 (2007)
43. I. Klapper, J. Dockery, Mathematical description of microbial biofilms. SIAM Rev. **52**, 221 (2010)

Chapter 11
Spectral Methods for PDE

In finite-difference methods the exact solution v of the differential equation is approximated by low-order polynomials interpolating v at several nearby mesh points. For example, (9.3) is an approximation of the derivative at x_j obtained by parabolic interpolation between x_{j-1}, x_j, and x_{j+1}. The solution on the whole interval is constructed by superposing many such overlapping polynomials as the weighted sum of the function values at the interpolation points.

In spectral methods [1] we approximate the solution *on the whole interval* by a *single* high-degree polynomial and establish conditions at which this polynomial approximates the exact solution as closely as possible. Different methods exploit different classes of polynomials and ways of realizing these conditions.

Linear Stationary Problems Spectral methods represent a subclass of the *methods of weighted residuals*. Initially we discuss linear PDE (e.g. of the form (9.1)) on the domain Ω, with the boundary condition specified at the domain boundary $\partial\Omega$,

$$Lv = Q \quad \text{in } \Omega, \tag{11.1}$$

$$Bv = 0 \quad \text{on } \partial\Omega, \tag{11.2}$$

where L and B are linear operators. It is easier to grasp the basic ideas of spectral methods by placing them in the context of mappings between vector spaces. The operator L in (11.1) acts in the Hilbert space X, which is the space of real or complex functions defined on Ω. These functions are square-integrable with respect to a continuous positive weight function. We present the operator L in its discrete form L_N, which is defined on $X_N \subset X$ and maps to X.

At the heart of all spectral methods is the condition for the *spectral approximation* $u^N \in X_N$ or for the *residual* $R = L_N u^N - Q$. We require that the linear projection with the projector P_N of the residual from the space $Z \subseteq X$ to the subspace $Y_N \subset Z$ is zero,

$$P_N\left(L_N u^N - Q\right) = 0. \tag{11.3}$$

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9_11,

The operator P_N is an orthogonal projector. With the scalar product $\langle u, v\rangle_N$ in the space Y_N this means $\langle z - P_N z, v\rangle_N = 0$ for $\forall v \in Y_N$. Under this assumption the basic spectral requirement can be written in its *variational form*

$$\left\langle L_N u^N - Q, v\right\rangle_N = 0 \quad \forall v \in Y_N. \tag{11.4}$$

The way in which the residual R is minimized depends on the choice of spaces X_N, Y_N, and the form of the projector P_N. Clearly X_N and Y_N should have equal dimensions for the solution of (11.3) to be unique, but in general these spaces need not be identical. In Galerkin and collocation methods for problems involving Dirichlet boundary conditions, usually $X_N = Y_N$, while in tau methods $X_N \neq Y_N$. The choice of the vector spaces and the definition of the scalar product (i.e. the projection of the differential equation to the corresponding subspace) depends on the nature of the problem and on the individual method.

In one dimension the spectral solution is sought in the form of a finite sum

$$u^N(x) = \sum_n u_n \phi_n(x), \tag{11.5}$$

where ϕ_n are the *trial* (or *expansion*, or *approximating*) *functions* that form the basis of X_N. The range of the index n depends on N and on the type of the method. In addition to the trial functions, we also utilize *test* or *weight functions* ψ_n, which are used to minimize the residual $R = L_N u^N - Q$ in the sense of the scalar product (11.4), thus

$$\langle \psi_n, R\rangle = 0 \quad \forall n. \tag{11.6}$$

In Galerkin and tau methods, the trial and test functions are equal ($\psi_n = \phi_n$), but in the Galerkin approach we choose them such that they fulfill boundary conditions by themselves; in tau methods, boundary conditions are imposed by supplemental equations. In collocation methods, the test functions are the delta-"functions" $\psi_n = \delta(x - x_n)$ applied at the collocation points x_n.

Linear Evolution Problems In the analysis of linear evolution problems

$$\begin{aligned} v_t + Lv &= Q \quad \text{in } \Omega \times (0, \infty), \\ Bv &= 0 \quad \text{on } \partial\Omega \times (0, \infty), \\ v &= v_0 \quad \text{in } \Omega \text{ at } t = 0, \end{aligned} \tag{11.7}$$

the spectral approximation is usually realized in the spatial part only. We treat time as a separate variable and compute the time evolution by using the methods for initial-value problems (Chap. 7).

The spectral approximation u^N for all $t \geq 0$ is a continuously differentiable function with the values in X_N. At $t = 0$ it satisfies the initial condition, while at $t > 0$ it satisfies

$$P_N\left(\frac{\mathrm{d}u^N}{\mathrm{d}t} + L_N u^N - Q\right) = 0,$$

or, in variational form,

$$\left\langle \frac{\mathrm{d}u^N}{\mathrm{d}t} + L_N u^N - Q, v \right\rangle_N = 0 \quad \forall v \in Y_N, \tag{11.8}$$

which is analogous to (11.4) for stationary problems. We will see that the conditions (11.6) translate to systems of algebraic equations, and conditions (11.8) to systems of differential equations.

Comparison of Difference and Spectral Methods Trial functions distinguish difference and finite element methods (Chaps. 9 and 10) from spectral methods. In the former two classes of methods, they are overlapping low-degree local polynomials, thus they are more suitable for irregular geometries. In spectral methods, they are infinitely differentiable global functions (trigonometric functions, Chebyshev or Legendre polynomials), so their applicability is limited to regular geometries without discontinuities. Still, spectral methods have also invaded the field of complex geometries [2].

11.1 Spectral Representation of Spatial Derivatives

In spectral methods for PDE the spatial derivatives can be evaluated in several ways, in configuration (physical) or transformed space. In transformed space the computation of the derivatives becomes very simple.

11.1.1 Fourier Spectral Derivatives

The Fourier series (4.8) corresponds to the Fourier series for the derivative of u,

$$Su' = \sum_{k=-\infty}^{\infty} \mathrm{i}k\widehat{u}_k \phi_k,$$

which is known as the spectral (Fourier–Galerkin) derivative. Deriving with respect to x in configuration space is equivalent to multiplying each Fourier coefficient by $\mathrm{i}k$ in transform space. (Recall the momentum operator $\boldsymbol{p} = \hbar\boldsymbol{k} = -\mathrm{i}\hbar\nabla$ in non-relativistic quantum mechanics.) The derivation and the truncation of the series ($Su \to S_N u$) commute, thus $(S_N u)' = S_N u'$. We obtain ever higher derivatives by multiplying the coefficients repeatedly by $\mathrm{i}k$,

$$\frac{\mathrm{d}^m u(x)}{\mathrm{d}x^m} \longleftrightarrow (\mathrm{i}k)^m \widehat{u}_k. \tag{11.9}$$

In the discrete case, the derivative is computed by using the function values $u_j = u(x_j)$ given at the Fourier collocation points

$$x_j = 2\pi j/N, \quad j = 0, 1, \ldots, N-1, \quad N \text{ even}.$$

First, we compute the discrete coefficients $\widetilde{u}_k$, multiply them by $\mathrm{i}k$, and transform back to configuration space by using (4.12). The approximation for the derivative at x_j is then

$$(\mathcal{D}_N u)_j = \sum_{k=-N/2}^{N/2-1} \widetilde{u}_k^{(1)} \mathrm{e}^{2\mathrm{i}jk\pi/N}, \quad j = 0, 1, \ldots, N-1,$$

where

$$\widetilde{u}_k^{(1)} = \mathrm{i}k\widetilde{u}_k = \frac{\mathrm{i}k}{N} \sum_{j=0}^{N-1} u(x_j)\mathrm{e}^{-2\mathrm{i}jk\pi/N}, \quad k = -N/2, -N/2+1, \ldots, N/2-1.$$

The $(\mathcal{D}_N u)$ is called the *Fourier collocation (or interpolation) derivative*, as the values $(\mathcal{D}_N u)_j$ are equal to the derivatives of the discrete Fourier transform at the mesh points. By changing the order of summation over k and j, the Fourier derivative can be expressed as a matrix multiplication:

$$(\mathcal{D}_N u)_l = \sum_{k=-N/2}^{N/2-1} \left(\frac{\mathrm{i}k}{N} \sum_{j=0}^{N-1} u(x_j)\mathrm{e}^{-2\pi\mathrm{i}jk/N} \right) \mathrm{e}^{2\pi\mathrm{i}kl/N} = \sum_{j=0}^{N-1} \left(D_N^{(1)}\right)_{lj} u_j, \quad (11.10)$$

where

$$\left(D_N^{(1)}\right)_{lj} = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \mathrm{i}k\mathrm{e}^{2\mathrm{i}k(l-j)\pi/N}. \quad (11.11)$$

If u is a real function, the term with $k = -N/2$ makes a purely imaginary contribution to the sum. In other words, if the Fourier coefficient $\widehat{u}_{-N/2}$ has a non-zero imaginary component, u^N is not a real function. As a rule, we therefore drop the term with $k = -N/2$, i.e. we set $\widehat{u}_{-N/2} \equiv 0$. This asymmetry between the lower and upper limit for k and the mentioned artifact originate in numerous implementations of the FFT calling for meshes with even numbers of points. The sum in (11.11) can then be computed analytically, yielding [3]

$$\left(D_N^{(1)}\right)_{lj} = \begin{cases} 0; & l = j, \\ \frac{1}{2}(-1)^{l+j} \operatorname{ctg} \frac{(l-j)\pi}{N}; & l \neq j. \end{cases} \quad (11.12)$$

An analogous procedure gives the matrix corresponding to the second derivative,

$$\left(D_N^{(2)}\right)_{lj} = \begin{cases} -\frac{1}{12}(N-1)(N-2); & l = j, \\ \frac{1}{4}(-1)^{l+j} N + \frac{1}{2}(-1)^{l+j+1} \sin^{-2} \frac{(l-j)\pi}{N}; & l \neq j. \end{cases} \quad (11.13)$$

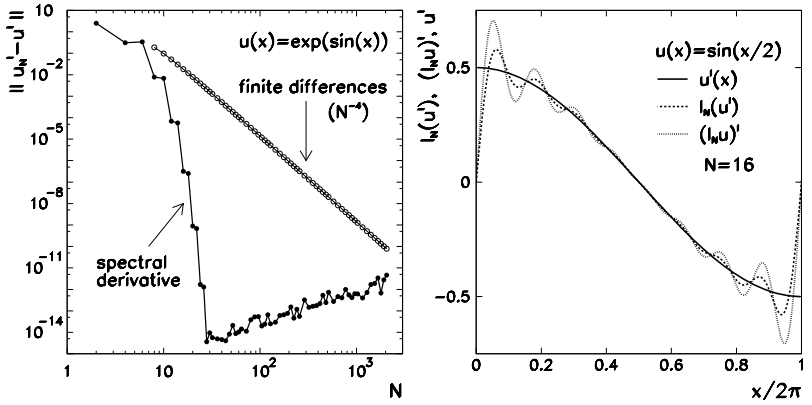


Fig. 11.1 [*Left*] The precision of the derivative of the function $u(x) = \exp(\sin x)$ on $[0, 2\pi]$. Finite-difference methods converge as $\mathcal{O}(N^{-m})$, where m is the order of the method (here $m = 4$). The spectral derivative converges faster than any power N^{-m}. [*Right*] The spectral derivatives of $\sin(x/2)$ compared to the analytical derivative. Due to the discontinuities at the boundaries of the interval $[0, 2\pi]$ Gibbs oscillations appear. Here I_N denotes the interpolants at N points (see (4.13))

This matrix can be used to compute the values of the second derivative at the collocation points by simple multiplication,

$$\left(\mathcal{D}_N^2 u\right)_l = \sum_{j=0}^{N-1} \left(D_N^{(2)}\right)_{lj} u_j. \tag{11.14}$$

Example One can get a feel for the beauty of spectral derivation in the case of a simple function $u(x) = \exp(\sin x)$ which is periodic on $x \in [0, 2\pi]$. We compute its derivative first by using finite differences, then by using (11.10). The fourth-order finite difference $u'(x_j) \approx [u(x_{j-2}) - 8u(x_{j-1}) + 8u(x_{j+1}) - u(x_{j+2})]/(12h)$ corresponds to the five-diagonal circulant matrix (write it down). The error with respect to the analytical solution $\|u'_N - u'\|$ decreases as $\mathcal{O}(N^{-4})$ (Fig. 11.1 (left)). All elements of the matrix $D_N^{(1)}$ in (11.12) except the diagonal ones are non-zero, but the error drops to round-off level already at very small N. For sufficiently smooth functions the error decreases as $\mathcal{O}(N^{-m})$ for *any* m. This dramatic fall-off that gave the methods of this chapter their name is known as *spectral convergence*. Do not expect it for functions with discontinuities! For such functions—regardless of N—typical oscillations occur throughout the domain, in particular at its edges (Gibbs phenomenon). An example for the function $u(x) = \sin(x/2)$ with the discontinuity in the derivative $u'(x) = \frac{1}{2}\cos(x/2)$ is shown in Fig. 11.1 (right).

The collocation derivative (or second derivative) can be computed naively, by multiplying the $N \times N$ matrix and the N-dimensional vector u_j according to (11.10) and (11.14), which takes $2N^2$ operations. There is a much faster way. First we use the inverse FFT to compute the interior sums, multiply the computed Fourier coeffi-

cients by ik (or $(ik)^2$ for the second derivative), and finally use FFT to compute the exterior sum. For a real function u the total cost is $N(5\log_2 N - 5)$ operations if the classic FFT is used (see pp. 168–169 and Fig. 4.7 (right)).

We have written the matrix representations of Fourier derivatives for an even number of mesh points N. The formulas for odd N are given in [4].

11.1.2 Legendre Spectral Derivatives

The spectral derivatives of functions that can be expanded in orthogonal polynomials can also be computed in configuration (physical) space or transform space. In transform space, the first and the second derivative of a function with the Legendre expansion (4.29) are

$$\begin{aligned} u'(x) &= \sum_{k=0}^{\infty} \widehat{u}_k^{(1)} P_k(x), \quad \widehat{u}_k^{(1)} = (2k+1) \sum_{\substack{p=k+1 \\ p+m \text{ odd}}}^{\infty} \widehat{u}_p, \\ u''(x) &= \sum_{k=0}^{\infty} \widehat{u}_k^{(2)} P_k(x), \quad \widehat{u}_k^{(2)} = \left(k+\frac{1}{2}\right) \sum_{\substack{p=k+2 \\ p+k \text{ even}}}^{\infty} \big[p(p+1) - k(k+1)\big] \widehat{u}_p. \end{aligned} \tag{11.15}$$

These formulas follow from the recurrence relations for Legendre polynomials [5]. The finite sum $(P_N u)'$ corresponding to the first derivative is known as the *Legendre–Galerkin derivative*. In contrast to the Fourier expansion, this derivative and the truncation of the sum do not converge, thus $(P_N u)' \neq P_{N-1} u'$. Asymptotically $P_{N-1} u'$ is a better approximation of u' than $(P_N u)'$ [3].

If the values of a function are given at the Gauss, Gauss–Radau, or Gauss–Lobatto quadrature nodes (see (4.30), (4.31), and (4.33)), it can be differentiated in physical space by differentiating the interpolation polynomial $I_N u$ of degree N and evaluating it at these points. The *Legendre collocation derivative* is defined as the derivative of the discrete finite series for the function u,

$$\mathcal{D}_N u = (I_N u)', \tag{11.16}$$

and is a polynomial of degree $N-1$. We compute the collocation derivative $(\mathcal{D}_N u)_l = (\mathcal{D}_N u)(x_l)$ at $x_0, x_1, \ldots, x_N$ from the values $u(x_j)$ as

$$(\mathcal{D}_N u)_l = \sum_{j=0}^{N} \big(D_N^{(1)}\big)_{lj} u(x_j), \quad l = 0, 1, \ldots, N. \tag{11.17}$$

The matrix elements $(D_N^{(1)})_{lj}$ can be computed explicitly for all three types of quadrature [3]. Most frequently, the Gauss–Lobatto variant (4.33) is used, for which

$$\left(D_N^{(1)}\right)_{lj} = \begin{cases} -\frac{N(N+1)}{4}; & j = l = 0, \\ 0; & 1 \le j = l \le N-1, \\ \frac{N(N+1)}{4}; & j = l = N, \\ \frac{P_N(x_l)}{P_N(x_j)} \frac{1}{x_l - x_j}; & j \ne l. \end{cases}$$

The matrix corresponding to the second derivative is

$$\left(D_N^{(2)}\right)_{lj} = \begin{cases} \frac{N(N+1)(N^2+N-2)}{24}; & j = l = 0,\ j = l = N, \\ \frac{(-1)^N}{P_N(x_j)} \frac{N(N+1)(1+x_j)-4}{2(1+x_j)^2}; & l = 0,\ 1 \le j \le N, \\ \frac{1}{P_N(x_j)} \frac{N(N+1)(1-x_j)-4}{2(1-x_j)^2}; & l = N,\ 0 \le j \le N-1, \\ \frac{1}{3} \frac{P_N''(x_j)}{P_N(x_j)}; & 1 \le j = l \le N-1, \\ -\frac{P_N(x_l)}{P_N(x_j)} \frac{2}{(x_l - x_j)^2}; & 1 \le l \le N-1, \\ & 0 \le j \le N,\ j \ne l. \end{cases}$$

A fast Legendre transform is not known, so in order to compute the Legendre spectral derivative, one should use the matrix multiplication (11.17).

11.1.3 Chebyshev Spectral Derivatives

In transform space, the first and second Chebyshev–Galerkin derivative of the function with the expansion (4.39) can be computed by using the formulas

$$u'(x) = \sum_{k=0}^{\infty} \widehat{u}_k^{(1)} T_k(x), \quad \widehat{u}_k^{(1)} = \frac{2}{c_k} \sum_{\substack{p=k+1 \\ p+k \text{ odd}}}^{\infty} p \widehat{u}_p, \tag{11.18}$$

$$u''(x) = \sum_{k=0}^{\infty} \widehat{u}_k^{(2)} T_k(x), \quad \widehat{u}_k^{(2)} = \frac{1}{c_k} \sum_{\substack{p=k+2 \\ p+k \text{ even}}}^{\infty} p\left(p^2 - k^2\right) \widehat{u}_p. \tag{11.19}$$

(The formulas for the third and fourth derivative are listed in [4].) The expansion coefficients of the derivative u' and of the function u are related by

$$c_k \widehat{u}_k^{(1)} = \widehat{u}_{k+2}^{(1)} + 2(k+1)\widehat{u}_{k+1}, \quad k = 0, 1, \ldots, N-1. \tag{11.20}$$

This allows us to effectively differentiate a degree-N polynomial in Chebyshev transform space: since $\widehat{u}_k^{(1)} = 0$ for all $k \ge N$, the expansion coefficients of u' can

be computed from the expansion of u by using (11.20). The generalization of this relation is

$$c_k \widehat{u}_k^{(q)} = \widehat{u}_{k+2}^{(q)} + 2(k+1)\widehat{u}_{k+1}^{(q-1)}, \quad k \geq 0.$$

The Chebyshev interpolation derivative in the sense of (11.16) can be represented in matrix form (11.17) just as in the case of Legendre polynomials. Here we list only the matrices corresponding to the first and second derivative for the Gauss–Lobatto nodes [3]. The matrix for the first derivative is

$$\left(D_N^{(1)}\right)_{lj} = \begin{cases} \frac{1}{6}(2N^2+1); & j=l=0, \\ -\frac{1}{2}\frac{x_j}{1-x_j^2}; & 1 \leq j=l \leq N-1, \\ -\frac{1}{6}(2N^2+1); & j=l=N, \\ \frac{\overline{c}_l}{\overline{c}_j}\frac{(-1)^{l+j}}{x_l-x_j}; & j \neq l. \end{cases} \tag{11.21}$$

The matrix corresponding to the second derivative is

$$\left(D_N^{(2)}\right)_{lj} = \begin{cases} \frac{1}{15}(N^4-1); & j=l=0,\ j=l=N, \\ \frac{2}{3}\frac{(-1)^j}{\overline{c}_j}\frac{(2N^2+1)(1-x_j)-6}{(1-x_j)^2}; & l=0,\ 1 \leq j \leq N, \\ \frac{2}{3}\frac{(-1)^{j+N}}{\overline{c}_j}\frac{(2N^2+1)(1+x_j)-6}{(1+x_j)^2}; & l=N,\ 0 \leq j \leq N-1, \\ -\frac{(N^2-1)(1-x_l^2)+3}{3(1-x_l^2)^2}; & 1 \leq j=l \leq N-1, \\ \frac{(-1)^{j+l}}{\overline{c}_j}\frac{x_l^2+x_jx_l-2}{(1-x_l^2)(x_j-x_l)^2}; & 1 \leq l \leq N-1, \\ & 0 \leq j \leq N,\ j \neq l. \end{cases}$$

(The coefficients $\overline{c}_i$ are defined below (4.41).) In computing the denominators $x_l - x_j$ and $1 - x_j^2$ for large N round-off errors may occur due to subtraction of almost equal values. The following expressions that can be derived from the definition of Chebyshev nodes, $x_j = \cos(\pi j/N)$, are more stable:

$$\frac{1}{1-x_j^2} \longrightarrow \frac{1}{\sin^2(j\pi/N)},$$

$$\frac{1}{x_l - x_j} \longrightarrow -\frac{1}{2}\frac{1}{\sin[(l+j)\pi/(2N)]\sin[(l-j)\pi/(2N)]}.$$

For further hints on improving the precision of matrix differentiation see [6].

As with Legendre polynomials, the Chebyshev series truncation and interpolation do not commute with differentiation. This implies that asymptotically $P_{N-1}u'$ and $I_{N-1}u'$ are better approximations of u' than $(P_N u)'$ or $(I_N u)'$.

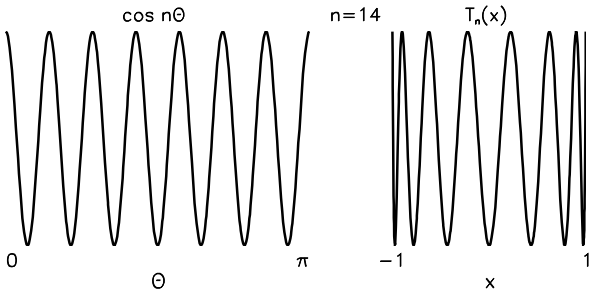


Fig. 11.2 The cosine function $\cos n\theta$ and the Chebyshev polynomial T_n for $n = 14$

11.1.4 Computing the Chebyshev Spectral Derivative by Fourier Transformation

The Chebyshev collocation spectral derivative can be computed efficiently by using the fast Fourier transformation. We see this from the structure of the interpolation polynomial in dependence on x or θ, where $x = \cos\theta$. In the following we restrict the discussion to the Chebyshev–Gauss–Lobatto collocation (4.40). The polynomial

$$p(x) = \sum_{n=0}^{N} a_n T_n(x), \quad x \in [-1, 1], \tag{11.22}$$

interpolates an arbitrary function f on the interval $x \in [-1, 1]$ at Chebyshev collocation points $x_j = \cos(\pi j/N)$, where $j = 0, 1, \ldots, N$. On the other hand, the polynomial

$$P(\theta) = \sum_{n=0}^{N} a_n \cos n\theta, \quad \theta \in \mathbb{R}, \tag{11.23}$$

interpolates an arbitrary even and 2π-periodic function F at equidistant points $\theta_j = \pi j/N$. With the transformation $x = \cos\theta$ we have $F(\theta) = f(x) = f(\cos\theta)$, which implies $P(\theta) = p(x) = p(\cos\theta)$. The Chebyshev transform (11.22) in x is therefore nothing but the Fourier (cosine) transform (11.23) in θ. In other words, if one "wraps" the cosine function $\cos n\theta$ around a half-cylinder with unit radius and look at it from the side, the Chebyshev polynomial T_n emerges (Fig. 11.2).

The procedure is [7]: compute the derivative of the Chebyshev interpolation polynomial p of the function f by first finding the trigonometric interpolation polynomial P of the corresponding function F; compute the derivative in Fourier space; compute the inverse Fourier transform on the uniform mesh; finally, display the obtained values on the mesh of Chebyshev collocation points.

We write the values u_j at the Chebyshev collocation points $x_j = \cos(\pi j/N)$, $j = 0, 1, \ldots, N$, as an array $U = \{U_1, U_2, \ldots, U_{2N}\}$ of dimension $2N$,

$$U_{j+1} = u_j, \quad j = 0, 1, \ldots, N,$$
$$U_{2N-j+1} = u_j, \quad j = 1, 2, \ldots, N-1.$$

By fast Fourier transformation we calculate

$$\widehat{U}_k = \frac{\pi}{N} \sum_{j=1}^{2N} \mathrm{e}^{-\mathrm{i}k\theta_j} U_j, \quad k = -N+1, -N+2, \ldots, N.$$

The polynomial P depends on the transformed values $\widehat{U}_k$ and has the form

$$P(\theta) = \frac{1}{2\pi} \sum_{k=-N+1}^{N} \mathrm{e}^{\mathrm{i}k\theta} \widehat{U}_k = \sum_{n=0}^{N} a_n \cos n\theta.$$

At this point we actually execute the differentiation by using (11.9) at $m = 1$, thus

$$\widehat{V}_k = \begin{cases} \mathrm{i}k\widehat{U}_k; & k = -N+1, -N+2, \ldots, N-1, \\ 0; & k = N, \end{cases} \tag{11.24}$$

and compute the inverse Fourier transform

$$W_j = \frac{1}{2\pi} \sum_{k=-N+1}^{N} \mathrm{e}^{\mathrm{i}k\theta_j} \widehat{V}_k, \quad j = 1, 2, \ldots, 2N.$$

This yields the derivative of the trigonometric interpolation polynomial P on the equidistant mesh θ_j. In the last step the derivative of the interpolation polynomial p is computed on the mesh x_j. This is accomplished by using the chain rule

$$p'(x) = \frac{\mathrm{d}p}{\mathrm{d}x} = \frac{\mathrm{d}P}{\mathrm{d}\theta} \frac{\mathrm{d}\theta}{\mathrm{d}x} = -\frac{P'(\theta)}{\sin\theta} = \frac{\sum_{n=0}^{N} a_n n \sin n\theta}{\sqrt{1-x^2}}.$$

Denote $p'(x_j) = w_j$ and $P'(\theta_j) = W_j$. At the interior collocation points we get

$$w_j = -\frac{W_j}{\sqrt{1-x_j^2}}, \quad j = 1, 2, \ldots, N-1. \tag{11.25}$$

Note that at $x_0 = 1$ ($\theta = 0$) and $x_N = -1$ ($\theta = \pi$), the expressions for w_0 and w_N are of the form $0/0$ and must be harnessed by the l'Hôpital rule, resulting in

$$w_0 = \frac{1}{\pi} \left[\sum_{n=1}^{N-1} n^2 \widehat{U}_n + \frac{N^2}{2} \widehat{U}_N \right], \qquad w_N = \frac{1}{\pi} \left[\sum_{n=1}^{N-1} (-1)^{n+1} n^2 \widehat{U}_n - \frac{N^2}{2} \widehat{U}_N \right].$$

Example Compute the Chebyshev spectral derivative of the function

$$u(x) = \left(x^2 - 1\right) \exp\left(x^3\right),$$

given at the Chebyshev nodes $x_j = \cos(\pi j/N)$, $j = 0, 1, \ldots, N$ where its values are determined by the interpolation polynomial (11.22); see Fig. 11.3 (left). The

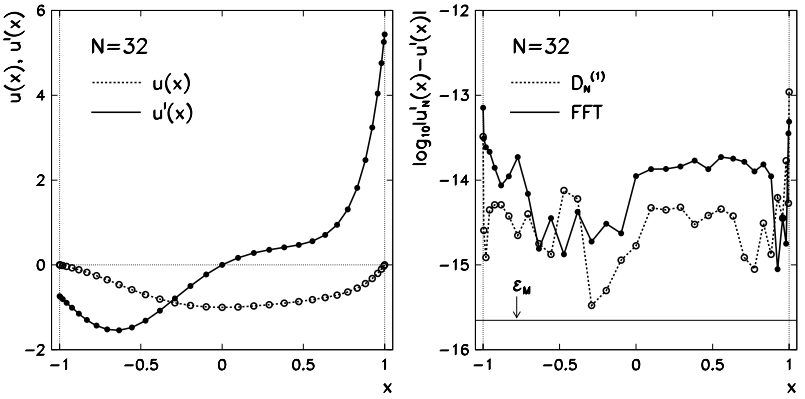


Fig. 11.3 [*Left*] The graphs of $u(x) = (x^2 - 1)\exp(x^3)$ and its derivative $u'(x)$. The *symbols* $\circ$ and $\bullet$ denote their values at the Chebyshev points $x_j = \cos(\pi j/N)$, $j = 0, 1, \ldots, N$, where $N = 32$. [*Right*] The absolute error of the derivative, computed via multiplication by the matrix (11.21) and according to (11.25) via FFT

derivatives at x_j can be computed simply by multiplying u_j by the matrix $D_N^{(1)}$ of (11.21), but a more elegant way to the derivatives $p'(x_j) = w_j$ leads through (11.25) and two additional equations for w_0 and w_N by using FFT. Figure 11.3 (right) shows the absolute error of both methods.

Higher derivatives can be computed similarly. To compute the mth derivative in Fourier space, step (11.24) should be repeated m-times, as shown in (11.9). We define

$$\widehat{V}_k^{(m)} = (\mathrm{i}k)^m \widehat{U}_k$$

and set $\widehat{V}_{-N} = 0$, if m is odd. Let us compute the second derivative:

$$p''(x) = \frac{\mathrm{d}P}{\mathrm{d}\theta}\frac{\mathrm{d}^2\theta}{\mathrm{d}x^2} + \frac{\mathrm{d}^2P}{\mathrm{d}\theta^2}\left(\frac{\mathrm{d}\theta}{\mathrm{d}x}\right)^2 = -\frac{x}{(1-x^2)^{3/2}}P'(\theta) + \frac{1}{1-x^2}P''(\theta).$$

By denoting $p''(x_j) = w_j^{(2)}$ and $P''(\theta_j) = W_j^{(2)}$ we obtain at the interior points

$$w_j^{(2)} = -\frac{x_j}{(1-x_j^2)^{3/2}}W_j + \frac{1}{1-x_j^2}W_j^{(2)}, \quad j = 1, 2, \ldots, N-1,$$

while at the boundary points (check this as an exercise) we get

$$w_0^{(2)} = \frac{1}{3\pi}\left[\sum_{n=1}^{N-1}\left(n^4 - n^2\right)\widehat{U}_n + \frac{N^4 - N^2}{2}\widehat{U}_N\right],$$

$$w_N^{(2)} = \frac{1}{3\pi}\left[\sum_{n=1}^{N-1}(-1)^n\left(n^4 - n^2\right)\widehat{U}_n + \frac{N^4 - N^2}{2}\widehat{U}_N\right].$$

11.2 Galerkin Methods

In Galerkin methods for stationary problems of the form (11.1) with boundary conditions (11.2), the residual R is minimized by requiring (11.6). This means

$$\left\langle \psi_n, L_N u^N - Q\right\rangle = \left\langle \psi_n, L_N \sum_k u_k\phi_k - Q\right\rangle = \sum_k u_k\langle\psi_n, L_N\phi_k\rangle - \langle\psi_n, Q\rangle = 0.$$

In the classical Galerkin methods, the trial and test functions are the same, so $\psi_n = \phi_n$, and the equations above become

$$\sum_k L_{nk}u_k = \langle\phi_n, Q\rangle, \quad n = 0, 1, \ldots, N,$$

$$L_{nk} = \langle\phi_n, L_N\phi_k\rangle.$$

By solving this linear system we get $N + 1$ coefficients u_k of the spectral expansion (11.5) of the function u^N. We have not specified in detail the range of the index k. For example, depending on the test function, it may run from $-N/2$ to $N/2 - 1$ (trigonometric polynomials) or from 0 to N (orthogonal polynomials).

11.2.1 Fourier–Galerkin

The classic model stationary problem is the one-dimensional Helmholtz equation

$$-\frac{\mathrm{d}^2 v}{\mathrm{d}x^2} + \lambda v = Q, \quad x \in (0, 2\pi), \tag{11.26}$$

where $\lambda \geq 0$. We seek solutions that are periodic on the interval $[0, 2\pi]$. The equation is of the form $Lv = Q$, where $L = -\mathrm{d}^2/\mathrm{d}x^2 + \lambda$ is a linear differential operator. This is not a "genuine" partial differential equation as it only contains the spatial derivative, but we use it to learn the basics of Galerkin methods. Later on we add another spatial dimension and time dependence.

First let us construct the spectral solution in the space of trigonometric polynomials. The trial space X_N is spanned on a set of such polynomials (basis functions) ϕ_k of degree $\leq N/2$,

$$\left\{\phi_k(x) = \mathrm{e}^{\mathrm{i}kx};\ -N/2 \leq k \leq N/2 - 1\right\}.$$

The spectral solution $u = u^N$ will then have the form of a finite Fourier series,

$$u(x) = u^N(x) = \sum_{k=-N/2}^{N/2-1} \widehat{u}_k \phi_k(x) = \sum_{k=-N/2}^{N/2-1} \widehat{u}_k \mathrm{e}^{\mathrm{i}kx}. \tag{11.27}$$

We interpret L as L_N, and insert the solution ansatz in the Galerkin condition,

$$\left\langle \psi_n, L_N u^N \right\rangle = \langle \psi_n, Q \rangle, \tag{11.28}$$

where

$$\psi_n(x) = \frac{1}{2\pi} \mathrm{e}^{-\mathrm{i}nx} \tag{11.29}$$

are the test functions. The trial and the test functions are orthogonal,

$$\langle \psi_n, \phi_k \rangle = \int_0^{2\pi} \psi_n(x) \phi_k^*(x) \, \mathrm{d}x = \delta_{k,n}. \tag{11.30}$$

From the system (11.28) one therefore reaps only the simple relation

$$\widehat{u}_k = \frac{\widehat{Q}_k}{k^2 + \lambda}, \quad -\frac{N}{2} \le k \le \frac{N}{2} - 1, \tag{11.31}$$

where the coefficients $\widehat{Q}_k$ are

$$\widehat{Q}_k = \frac{1}{2\pi} \int_0^{2\pi} Q(x) \mathrm{e}^{-\mathrm{i}kx}. \tag{11.32}$$

The value of the coefficient $\widehat{u}_0$ is arbitrary when $\lambda = 0$ (a non-zero value means just a translation of the solution along the dependent axis). In order to exploit the standard Fourier transformation with even N, we set $\widehat{Q}_{-N/2} = 0$ and thus damp only the highest Fourier component that has no match in $\widehat{Q}_{N/2}$ (as argued on p. 578).

We have described the *Fourier–Galerkin method*. It is a "Fourier" method because the trial and test functions are trigonometric polynomials; it is also a "Galerkin" method because the minimization of the residual occurs in the characteristic manner (11.28). For certain functions Q the Fourier coefficients (11.32) can be computed analytically; otherwise numerical integration must be used. An example is discussed in Problem 11.9.1.

11.2.2 Legendre–Galerkin

The Fourier–Galerkin method is applicable to the problem (11.26) with periodic solutions. For the same problem with homogeneous Dirichlet conditions,

$$-\frac{\mathrm{d}^2 v}{\mathrm{d}x^2} + \lambda v = Q, \quad x \in (-1, 1), \; v(-1) = v(1) = 0, \tag{11.33}$$

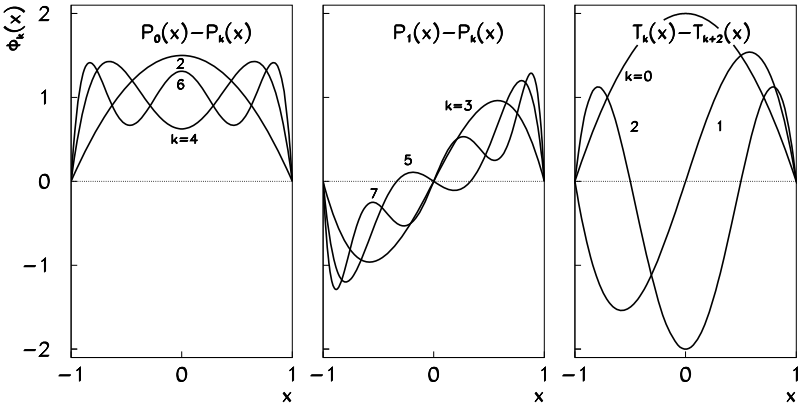


Fig. 11.4 [*Left*, *Center*] The basis functions (11.35) for the Legendre–Galerkin method with $k \in \{2, 4, 6\}$ and $k \in \{3, 5, 7\}$. [*Right*] The basis functions (11.40) for the Chebyshev–Galerkin methods with $k \in \{0, 1, 2\}$

the Legendre polynomials also offer a very natural basis. (More general guidelines on choosing the basis functions can be found in Sect. 11.7.) Following [3] one can seek the spectral solution on the interval $[-1, 1]$ in the form

$$u(x) = \sum_{k=2}^{N} \breve{u}_k \phi_k(x), \tag{11.34}$$

where the basis functions are

$$\phi_k(x) = \begin{cases} P_0(x) - P_k(x); & k \text{ even and } \geq 2, \\ P_1(x) - P_k(x); & k \text{ odd and } \geq 3 \end{cases} \tag{11.35}$$

(see Fig. 11.4 (left and center)).

Legendre polynomials have the property $P_k(\pm 1) = (\pm 1)^k$, so the particular choice of the basis functions (11.35) automatically fulfills the boundary conditions $u(-1) = u(1) = 0$, even though ϕ_k are *not* orthogonal. Because the spectral solution is expanded in terms of the functions ϕ_k instead of the polynomials P_k, we use the notation $\breve{u}_k$ for the expansion coefficients instead of the usual $\widehat{u}_k$ as in (4.29). We set the test functions to be identical to the trial functions, thus $\psi_k = \phi_k$. The Galerkin condition then becomes

$$-\left\langle \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}, \phi_l \right\rangle + \lambda \langle u, \phi_l \rangle = \langle Q, \phi_l \rangle = b_l, \quad l = 2, 3, \ldots, N, \tag{11.36}$$

where the scalar products should be understood as

$$\langle u, v \rangle = \int_{-1}^{1} u(x) v(x) \, \mathrm{d}x$$

(see (A.1) with the Legendre weight $w(x) = 1$). After integrating by parts and taking into account the boundary conditions, one obtains

$$-\left\langle \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}, \phi_l \right\rangle = -\underbrace{\left[\phi_l(x) \frac{\mathrm{d}u}{\mathrm{d}x} \right]\Big|_{-1}^{1}}_{0} + \left\langle \frac{\mathrm{d}u}{\mathrm{d}x}, \frac{\mathrm{d}\phi_l}{\mathrm{d}x} \right\rangle.$$

We write the expansion coefficients $\breve{u}_k$ and the expressions for b_l as vectors $\breve{\boldsymbol{u}} = (\breve{u}_2, \breve{u}_3, \ldots, \breve{u}_N)^{\mathrm{T}}$ and $\boldsymbol{b} = (b_2, b_3, \ldots, b_N)^{\mathrm{T}}$. When the solution ansatz is inserted in (11.36), the system can be written as a matrix equation for $\breve{\boldsymbol{u}}$:

$$(K + \lambda M)\breve{\boldsymbol{u}} = \boldsymbol{b}, \tag{11.37}$$

where the matrix elements of K and M are

$$K_{lk} = \left\langle \frac{\mathrm{d}\phi_k}{\mathrm{d}x}, \frac{\mathrm{d}\phi_l}{\mathrm{d}x} \right\rangle = \begin{cases} \min(k,l)[\min(k,l)+1]; & k,l \text{ even and } \geq 2, \\ \min(k,l)[\min(k,l)+1] - 2; & k,l \text{ odd and } \geq 3, \\ 0; & \text{otherwise}, \end{cases}$$

$$M_{lk} = \langle \phi_k, \phi_l \rangle = \begin{cases} 2 + \frac{2}{2k+1}\delta_{k,l}; & k,l \text{ even and } \geq 2, \\ \frac{2}{3} + \frac{2}{2k+1}\delta_{k,l}; & k,l \text{ odd and } \geq 3, \\ 0; & \text{otherwise}. \end{cases}$$

(Compute the elements K_{kl} and M_{kl} as an exercise. You only need the basic properties of Legendre polynomials.) Through solving (11.37) we obtain the "Legendre–Galerkin" spectral approximation of u by (11.34).

The matrix $K + \lambda M$ is sparse (empty subdiagonals interchange with the full), so the solution of the system (11.37) may be time-consuming and prone to round-off errors. With a different choice of the basis functions,

$$\phi_k(x) = \frac{1}{\sqrt{4k+6}}\big[P_k(x) - P_{k+2}(x)\big], \quad k \geq 0, \tag{11.38}$$

that fulfill the boundary conditions, and the solution ansatz

$$u(x) = \sum_{k=0}^{N-2} \breve{u}_k \phi_k(x) \tag{11.39}$$

we obtain a tridiagonal system. Details can be found in [2], and a concrete example is given in Problem 11.9.1.

11.2.3 Chebyshev–Galerkin

Legendre polynomials are not the only option to solve problems of the form (11.33). They can be also attacked by Chebyshev polynomials. The spectral approximation

is written in the form (11.39), with the basis functions

$$\phi_k(x) = T_k(x) - T_{k+2}(x), \quad k = 0, 1, \ldots, N-2, \tag{11.40}$$

which again are not orthogonal, but they do fulfill the boundary conditions $u(-1) = u(1) = 0$ due to the property $T_k(\pm 1) = (\pm 1)^k$ (Fig. 11.4 (right)). The Galerkin condition now reads

$$-\left\langle \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}, \phi_l \right\rangle_w + \lambda \langle u, \phi_l \rangle_w = \langle Q, \phi_l \rangle_w = b_l, \quad l = 0, 1, \ldots, N-2,$$

where the scalar products involve the Chebyshev weight:

$$\langle u, v \rangle_w = \int_{-1}^{1} u(x) v(x) w(x)\, \mathrm{d}x, \quad w(x) = \frac{1}{\sqrt{1-x^2}}.$$

When the expansion coefficients $\breve{u}_k$ and the expressions for b_l at the right-hand side of the Galerkin equation are arranged as vectors $\breve{\boldsymbol{u}} = (\breve{u}_0, \breve{u}_1, \ldots, \breve{u}_{N-2})^{\mathrm{T}}$ and $\boldsymbol{b} = (b_0, b_1, \ldots, b_{N-2})^{\mathrm{T}}$, we again obtain a matrix system of the form (11.37), where

$$K_{lk} = \begin{cases} 2\pi(l+1)(l+2); & k = l, \\ 4\pi(l+1); & k = l+2, l+4, l+6, \ldots, \\ 0; & k < l \text{ or } k+l = \text{odd}, \end{cases}$$

$$M_{lk} = \begin{cases} \frac{\pi}{2}(c_l + 1); & k = l, \\ -\frac{\pi}{2}; & k = l \pm 2, \\ 0; & \text{otherwise}, \end{cases}$$

where $c_0 = 2$ and $c_k = 1$ for $k \geq 1$. In this case the matrix system is asymmetric (K is upper triangular and M is tridiagonal). The b_l can be computed numerically, and they can be expressed in terms of the usual Chebyshev coefficients,

$$b_l = \frac{\pi}{2}\left(c_l \widehat{Q}_l - c_{l+2} \widehat{Q}_{l+2}\right), \quad l = 0, 1, \ldots, N-2.$$

Once the coefficients $\breve{u}_k$ are known, the solution is given by the sum (11.39), or else we transform the coefficients,

$$\widehat{u}_k = \begin{cases} \breve{u}_k; & k = 0, 1, \\ \breve{u}_k - \breve{u}_{k-2}; & k = 2, 3, \ldots, N-2, \end{cases}$$

and apply the expansion in terms of the Chebyshev polynomials,

$$u(x) = \sum_{k=0}^{N-2} \widehat{u}_k T_k(x).$$

Figure 11.6 shows the error of the Chebyshev–Galerkin method for the problem with the function $Q(x) = (16\pi^2 + \lambda)\sin(4\pi x)$ in comparison to other methods.

11.2.4 Two Space Dimensions

Extending the Galerkin method for our chosen model problem to two space dimensions is straightforward. The Helmholtz equation (11.26) becomes

$$-\nabla^2 v + \lambda v = Q, \quad x \in \Omega = (0, 2\pi) \times (0, 2\pi),$$

and instead of (11.27) the spectral solution is sought in the form

$$u(x, y) = \sum_{kl} \widehat{u}_{kl} \mathrm{e}^{\mathrm{i}(kx+ly)}.$$

The test functions, the scalar product (11.30), and the Galerkin condition (11.28) should be generalized to two dimensions as well. Instead of (11.31) we get

$$\widehat{u}_{kl} = \frac{\widehat{Q}_{kl}}{k^2 + l^2 + \lambda}, \quad k, l = -N/2, -N/2 + 1, \ldots, N/2 - 1,$$

with the coefficients

$$\widehat{Q}_{kl} = \int_\Omega Q(x, y) \mathrm{e}^{-\mathrm{i}(kx+ly)} \, \mathrm{d}x \, \mathrm{d}y.$$

To solve the equation $-\nabla^2 v + \lambda v = Q$ with Dirichlet boundary conditions (two-dimensional generalization of (11.33)) by Legendre–Galerkin or Chebyshev–Galerkin method, the basis functions (11.35) and (11.40) are used in product,

$$\phi_{kl}(x, y) = \phi_k(x)\phi_l(y).$$

Only the book-keeping becomes slightly more complicated: the expansion coefficients are organized as $\approx N \times N$-dimensional vectors, and the matrices K and M of the corresponding matrix systems blow up to sizes $\approx N^2 \times N^2$. (Recall the vector (10.1) that holds the two-dimensional finite-difference solution.)

11.2.5 Non-stationary Problems

The spectral approach changes substantially when time dependence is included. Following [3], the basic model problem is the linear hyperbolic equation

$$\frac{\partial v}{\partial t} - \frac{\partial v}{\partial x} = 0, \quad x \in (0, 2\pi), \ t > 0,$$

with a periodic boundary condition $v(0, t) = v(2\pi, t)$ for each t, and the initial condition $v(x, 0) = f(x)$. The ansatz for the solution is

$$u(x, t) = \sum_{k=-N/2}^{N/2-1} \widehat{u}_k(t)\phi_k(x), \tag{11.41}$$

where the basis functions $\phi_k(x) = \mathrm{e}^{\mathrm{i}kx}$ depend on the space coordinate while the expansion coefficients depend on time. We keep (11.29) as test functions.

In evolution problems of the form (11.7) the residual R is minimized by the variational condition (11.8). The Galerkin condition gives

$$\frac{1}{2\pi}\int_0^{2\pi}\left[\left(\frac{\partial}{\partial t}-\frac{\partial}{\partial x}\right)\sum_{k=-N/2}^{N/2-1}\widehat{u}_k(t)\mathrm{e}^{\mathrm{i}kx}\right]\mathrm{e}^{-\mathrm{i}nx}\,\mathrm{d}x = 0.$$

When the terms $\widehat{u}_k(t)\mathrm{e}^{\mathrm{i}kx}$ are differentiated with respect to t and x, this becomes

$$\frac{1}{2\pi}\int_0^{2\pi}\left[\sum_{k=-N/2}^{N/2-1}\left(\frac{\mathrm{d}\widehat{u}_k}{\mathrm{d}t}-\mathrm{i}k\widehat{u}_k\right)\mathrm{e}^{\mathrm{i}kx}\right]\mathrm{e}^{-\mathrm{i}nx}\,\mathrm{d}x = 0.$$

The left-hand side of the equation can be integrated analytically. In fact, due to the orthogonality (11.30), the integration is trivial, yielding a system of differential equations for the coefficients $\widehat{u}_k(t)$,

$$\frac{\mathrm{d}\widehat{u}_k}{\mathrm{d}t} = \mathrm{i}k\widehat{u}_k, \quad -N/2 \le k \le N/2-1, \tag{11.42}$$

with the initial conditions

$$\widehat{u}_k(0) = \frac{1}{2\pi}\int_0^{2\pi} f(x)\mathrm{e}^{-\mathrm{i}kx}\,\mathrm{d}x. \tag{11.43}$$

The system (11.42) can be solved by some method for initial-value problems described in Chap. 7. The complete solution at time t is given by the sum (11.41).

How the Fourier–Galerkin method works in practice can be experienced in Problem 11.9.2. In the case of a Dirichlet boundary condition (the equation is only allowed one boundary condition) the expansion in terms of Chebyshev polynomials can be used (the Chebyshev–Galerkin method).

Example In this example of solving time-dependent problems by the Fourier method, an important advantage of spectral approaches over the finite-difference methods emerges. We seek the solution of the equation

$$v_t - v_x = 0, \quad x \in (0, 2\pi),\ t > 0,$$

with the boundary condition $v(0,t) = v(2\pi,t)$ and the initial condition $v(x,0) = \exp(\sin x)$, after long times. The analytic solution is $v(x,t) = \exp(\sin(x+t))$. In a finite-difference approach, one writes the equation in terms of *local* differences,

$$\frac{u_j^{n+1}-u_j^n}{\Delta t} = \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x},$$

as in (9.7) and (9.3), and then finds the solution at consecutive times $n\Delta t$ by, say, the explicit Euler method (7.5). The solution at time 400π on a mesh with $N = 200$

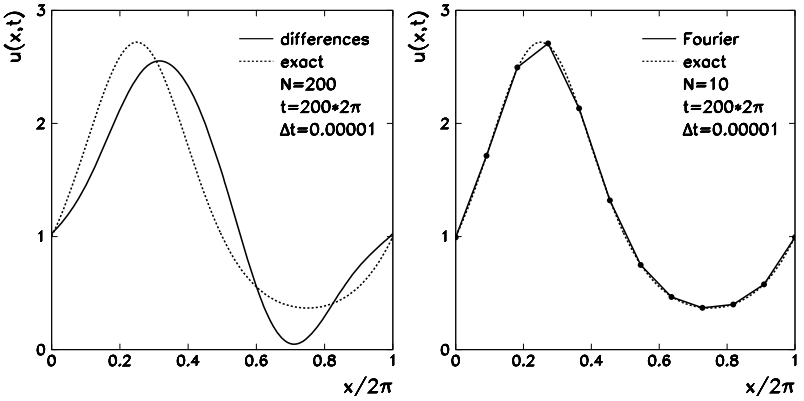


Fig. 11.5 The numerical solution of the equation $v_t = v_x$ with the initial condition $v(x,0) = \exp(\sin x)$ and periodic boundary conditions. Shown is the solution at $t = 400\pi$ obtained by the explicit Euler method. [*Left*] The finite-difference solution develops typical phase errors. [*Right*] The solution by the Fourier spectral method

points with time step $\Delta t = 10^{-5}$ is shown in Fig. 11.5 (left). The solution develops an error in both amplitude and phase.

A computation on a very coarse mesh with $N = 10$ using the Fourier collocation derivative (11.10), which approximates the derivative by a *global* sum of basis functions, gives the result shown in Fig. 11.5 (right), all other parameters unchanged. (With $N = 200$ the differences between the numerical and analytic solution would no longer be noticeable on this scale.)

If the differential equations are linear, adding terms or increasing the order of derivatives causes only minor complications. For example, the solution of the convection-diffusion problem

$$\frac{\partial v}{\partial t} = c\frac{\partial v}{\partial x} + D\frac{\partial^2 v}{\partial x^2}, \quad x \in (0, 2\pi),\ t > 0,$$

with a smooth initial condition $v(x,0) = f(x)$ and periodic boundary condition, can again be sought in the form (11.41). Instead of the system (11.42) we get

$$\frac{\mathrm{d}\widehat{u}_k}{\mathrm{d}t} = \big(\mathrm{i}kc - k^2 D\big)\widehat{u}_k, \quad -N/2 \le k \le N/2 - 1, \tag{11.44}$$

with the initial condition (11.43). The $c = 0$ case is discussed in Problem 11.9.3.

Fourier–Galerkin methods work best for linear evolution problems or problems with constant coefficients. They are less suitable for non-linear or variable-coefficient problems, as the resulting systems of differential equations for the evolution coefficients become too complicated, if they can be written down compactly at all. (If you doubt that, write the corresponding systems for the equations $v_t = vv_x$ and $v_t = \mathrm{e}^v v_x$, see also Sect. 11.5 and [4].)

Problem 11.9.3 also teaches us about the usefulness of Galerkin method using orthogonal polynomials. Solving evolution problems by the Legendre–Galerkin or Chebyshev–Galerkin method leads to a matrix system of ordinary differential equations for the evolution coefficients $\boldsymbol{a}$ of the form

$$M\frac{\mathrm{d}\boldsymbol{a}(t)}{\mathrm{d}t} = S\boldsymbol{a}(t),$$

where the matrix M depends on the chosen basis (and is easy to compute), while S depends on the nature of the differential problem. All basis functions must satisfy the boundary conditions, time-dependent or not. A more flexible treatment of the boundary conditions is offered by tau methods (Sect. 11.3).

11.3 Tau Methods

Tau methods are representatives of weighted residual methods, in which solving the differential equation leads to the minimization (11.6) as in Galerkin methods. But in contrast to Galerkin methods, the test functions in tau methods do not satisfy the boundary conditions of the differential problem; additional equations are needed to impose boundary conditions. (Slightly different interpretations of tau methods can be found in literature; see e.g. Chap. 21 in [8].)

The relatively simple implementation of the boundary conditions—even time-dependent ones—is a major advantage of tau methods. Still, each problem requires us to derive a distinct system of equations for the expansion coefficients of the spectral solution. Tau methods are optimally suited for linear elliptic problems with Chebyshev or Legendre polynomials as test functions. In the former case, the corresponding scalar product is in the sense of (4.20) with the weight function $w(x) = 1/\sqrt{1-x^2}$, while in the latter, $w(x) = 1$.

11.3.1 Stationary Problems

To explain the method we stick to the model problem (11.33) and use Chebyshev polynomials. The expansion of an arbitrary function u on $x \in [-1, 1]$ has the form (4.39). When we insert this expansion at finite N in (11.6), consider the expression for u'' (see (11.19)) and orthogonality (4.21), we get

$$-\widehat{u}_k^{(2)} + \lambda\widehat{u}_k = \widehat{Q}_k, \quad k = 0, 1, \ldots, N-2. \tag{11.45}$$

The boundary conditions are expressed by the additional equations

$$u(-1)=0 \quad \Longrightarrow \quad \sum_{k=0}^{N} \widehat{u}_k T_k(-1) = \sum_{k=0}^{N} \widehat{u}_k (-1)^k = 0,$$

$$u(1)=0 \quad \Longrightarrow \quad \sum_{k=0}^{N} \widehat{u}_k T_k(1) = \sum_{k=0}^{N} \widehat{u}_k (1)^k = 0,$$

where we have used $T_k(\pm 1) = (\pm 1)^k$. These equations can also be written as

$$\sum_{\substack{k=0 \\ k \text{ even}}}^{N} \widehat{u}_k = 0, \qquad \sum_{\substack{k=1 \\ k \text{ odd}}}^{N} \widehat{u}_k = 0. \tag{11.46}$$

This is typical of tau methods: when the minimization condition is imposed, the residual $Lu - Q$ is projected on a $(N - N_\mathrm{b})$-dimensional space, where N_b is the number of boundary conditions (here $N_\mathrm{b} = 2$). The polynomial approximation u has degree N, but N_b degrees of freedom are reserved for the boundary conditions. The coefficients $\widehat{u}_k^{(2)}$ in (11.45) can be expressed by (11.19), yielding

$$-\frac{1}{c_k} \sum_{\substack{p=k+2 \\ p+k \text{ even}}}^{N} p\big(p^2 - k^2\big)\widehat{u}_p + \lambda \widehat{u}_k = \widehat{Q}_k, \quad k = 0, 1, \ldots, N-2, \tag{11.47}$$

where $c_0 = 2$ and $c_k = 1$ for $k \geq 1$, and

$$\widehat{Q}_k = \frac{2}{\pi c_k} \int_{-1}^{1} Q(x) T_k(x) \frac{1}{\sqrt{1-x^2}} \, \mathrm{d}x.$$

Equations (11.47) and (11.46) constitute a matrix system for the coefficients $\widehat{u}_k$. The matrix is upper triangular (with additional rows for the boundary condition) and can be ill-conditioned if λ is small. The method to convert the system to a more robust quasi-tridiagonal form can be found in [3]. The error of the Chebyshev tau method for the Helmholtz problem with $Q(x) = (16\pi^2 + \lambda)\sin(4\pi x)$ and analytic solution $v(x) = \sin(4\pi x)$ is shown in Fig. 11.6.

More general (mixed) boundary conditions

$$\alpha_1 v(-1) + \beta_1 v_x(-1) = c_-,$$
$$\alpha_2 v(1) + \beta_2 v_x(1) = c_+$$

can be incorporated in the tau method quite unobtrusively: the system of equations for the expansion coefficients $\widehat{u}_k$ for $k = 0, 1, \ldots, N-2$ is still given by (11.47), only the last two rows of the matrix change: since

$$T_k(\pm 1) = (\pm 1)^k, \qquad T_k'(\pm 1) = (\pm 1)^{k+1} k^2,$$

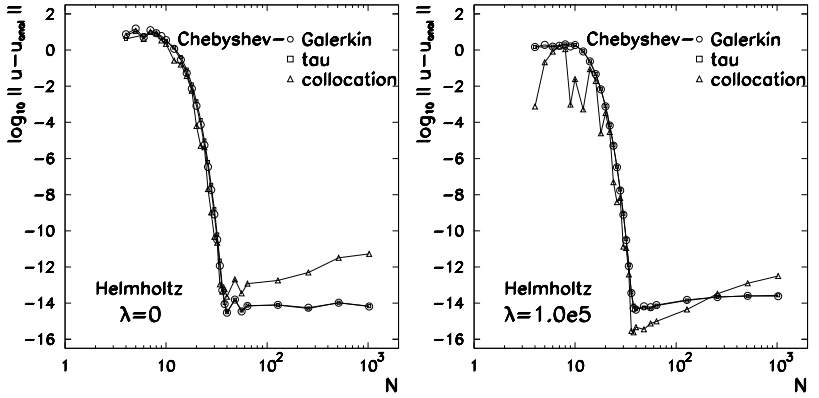


Fig. 11.6 The error of the numerical solution of the Helmholtz problem (11.33) in dependence of the number of basis functions in Chebyshev–Galerkin and Chebyshev tau methods, and in dependence of the number of collocation points in Chebyshev collocation. The function Q corresponds to the analytic solution $v(x) = \sin(4\pi x)$. [*Left*] $\lambda = 0$ (Poisson equation). [*Right*] $\lambda = 10^5$ (a typical value encountered in computations of incompressible flows)

(11.46) are replaced by

$$\sum_{k=0}^{N} \widehat{u}_k \left(\alpha_1 (-1)^k + \beta_1 (-1)^{k+1} k^2\right) = c_-,$$

$$\sum_{k=0}^{N} \widehat{u}_k \left(\alpha_2 + \beta_2 k^2\right) = c_+.$$

11.3.2 Non-stationary Problems

Tau methods easily accommodate non-stationary problem as well: time dependence is assigned to the expansion coefficients. As an example we discuss the Chebyshev tau method for the diffusion equation

$$v_t = v_{xx}, \quad x \in (-1, 1),\ t > 0,$$

with mixed boundary conditions

$$\alpha_1 v(-1, t) + \beta_1 v_x(-1, t) = c_-(t), \tag{11.48}$$

$$\alpha_2 v(1, t) + \beta_2 v_x(1, t) = c_+(t), \tag{11.49}$$

and the initial condition $v(x,0) = f(x)$. We seek the solution in the form

$$u(x,t) = \sum_{k=0}^{N} \widehat{u}_k(t) T_k(x). \tag{11.50}$$

By (11.8) we require that the residual $R = u_t - u_{xx}$ is orthogonal to all functions from the space of Chebyshev polynomials of degree $N-2$, so

$$\frac{2}{\pi c_k} \int_{-1}^{1} \big(u_t(x,t) - u_{xx}(x,t)\big) T_k(x) \frac{1}{\sqrt{1-x^2}} \,\mathrm{d}x = 0, \quad k = 0, 1, \ldots, N-2.$$

The coefficients of the second space derivative are again given by (11.19), whence a system of linear differential equations for the expansion coefficients emerges,

$$\frac{\mathrm{d}\widehat{u}_k}{\mathrm{d}t} = \frac{1}{c_k} \sum_{\substack{p=k+2 \\ p+k \text{ even}}}^{N} p\big(p^2 - k^2\big)\widehat{u}_p(t), \quad k = 0, 1, \ldots, N-2. \tag{11.51}$$

The initial conditions are

$$\widehat{u}_k(0) = \frac{2}{\pi c_k} \int_{-1}^{1} f(x) T_k(x) \frac{1}{\sqrt{1-x^2}} \,\mathrm{d}x.$$

The system (11.51) yields the first $N-1$ solution coefficients at time t, while the remaining coefficients $\widehat{u}_{N-1}(t)$ and $\widehat{u}_N(t)$ are given by two supplementary equations for the boundary conditions (11.48) and (11.49), in which again the properties $T_k(\pm 1) = (\pm 1)^k$ and $T_k'(\pm 1) = k^2(\pm 1)^{k+1}$ are exploited:

$$\sum_{k=0}^{N} \widehat{u}_k(t)\big(\alpha_1(-1)^k + \beta_1(-1)^{k+1}k^2\big) = c_-(t),$$

$$\sum_{k=0}^{N} \widehat{u}_k(t)\big(\alpha_2 + \beta_2 k^2\big) = c_+(t).$$

The boundary conditions therefore remain detached from the representation of the differential problem. Apart from the time evolution of the coefficients, the method is no harder than in the stationary case.

11.4 Collocation Methods

In collocation (also called pseudospectral) methods [9] the solution u is represented by the values $u(x_j) = u_j$ at specific *collocation points* or *nodes*. In Fourier collocation methods for periodic problems on the interval $[0, 2\pi]$, the nodes are the Fourier

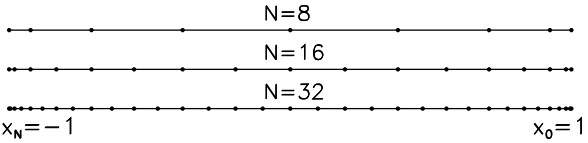


Fig. 11.7 Chebyshev–Gauss–Lobatto nodes (4.40) on the interval $[x_N, x_0] = [-1, 1]$ for $N = 8$, 16, and 32 (See also Fig. 4.10)

points (4.10). In methods with orthogonal polynomials, the location of nodes on the interval $[-1, 1]$ depends on the type of collocation. For example, the points (4.32) are the nodes of Legendre–Gauss–Lobatto collocation, while the points (4.40) are the nodes of Chebyshev–Gauss–Lobatto collocation (see Fig. 11.7). Between the nodes we span the interpolation polynomial $I_N u$, and represent the derivatives of u by analytic derivatives of this polynomial.

11.4.1 Stationary Problems

Let us keep the Helmholtz periodic problem (11.26) as our baseline example, closely following [9]. The Fourier collocation solution is given by the equations

$$\left[-\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} + \lambda u - Q\right]\Big|_{x=x_j} = 0, \quad j = 0, 1, \ldots, N-1, \tag{11.52}$$

where N is even and x_j are the Fourier collocation points (4.10). This is equivalent to the requirement for the minimization of the residual (11.6) with $L_N = L = -\mathrm{d}^2/\mathrm{d}x^2 + \lambda$, test functions $\psi_n(x) = \delta(x - x_n)$, and the usual scalar product on $[0, 2\pi]$. When the discrete Fourier transform (4.12) is inserted in the equations above, we get

$$k^2 \widetilde{u}_k + \lambda \widetilde{u}_k = \widetilde{Q}_k, \quad -\frac{N}{2} \le k \le \frac{N}{2} - 1,$$

and

$$\widehat{Q}_k = \frac{1}{N} \sum_{j=0}^{N-1} Q(x_j) \mathrm{e}^{-\mathrm{i}k x_j},$$

which closely resembles (but is not identical) to (11.31) and (11.32). In practice, we evaluate Q at the Fourier collocation points x_j and use the transformation (4.11), in FFT form, to compute the coefficients $\widetilde{Q}_k$, whence we obtain $\widetilde{u}_k = \widetilde{Q}_k/(k^2 + \lambda)$. Finally, the transformation (4.12) is applied, again as FFT, to turn $\widetilde{u}_k$ into the function values $u(x_j)$ which represent our final solution. If we wish to know the values $u(x)$ on the subintervals *between* the collocation points, the complete Lagrange interpolation polynomial can be plotted.

The problem with Dirichlet boundary conditions can be handled just as efficiently. Consider the case of Chebyshev collocation with Gauss–Lobatto nodes. The

form of the collocation condition (11.52) remains unchanged, but x_j are now given by (4.40), resulting in the system of equations

$$\sum_{k=0}^{N}\left[-\left(D_N^{(2)}\right)_{kj}+\lambda\delta_{k,j}\right]u(x_j)=Q(x_j),$$

where the matrix $D_N^{(2)}$ represents the Chebyshev collocation second derivative (see p. 582). If the solution u and the right-hand sides of the equation Q at the collocation points are arranged in the vectors

$$\boldsymbol{u}=\left(u(x_0),u(x_1),\ldots,u(x_N)\right)^{\mathrm{T}},\qquad \boldsymbol{Q}=\left(Q(x_0),Q(x_1),\ldots,Q(x_N)\right)^{\mathrm{T}},$$

we obtain, in short,

$$Z_N\left[-D_N^{(2)}+\lambda I\right]\boldsymbol{u}=\boldsymbol{Q}.$$

Dirichlet boundary conditions are imposed by setting the first and last column of the system matrix to zero: this is accomplished by the matrix Z_N. We solve the system by classical methods (the system matrix is sparse). An alternative to matrix multiplication leads through Chebyshev transformation and FFT (Sect. 11.1.3). The error of the Chebyshev collocation method for the Helmholtz problem with $Q(x)=(16\pi^2+\lambda)\sin(4\pi x)$ is shown in Fig. 11.6.

11.4.2 Non-stationary Problems

A typical non-stationary periodic problem is the advection-diffusion equation

$$v_t=c(x)v_x+D(x)v_{xx},\quad x\in(0,2\pi),\ t>0,$$

with the initial condition $v(x,0)=f(x)$ and the boundary condition $v(0,t)=v(2\pi,t)$. The Fourier collocation solution can be sought in the form [4]

$$u(x,t)=\sum_{j=0}^{N-1}u(x_j,t)g_j(x),\tag{11.53}$$

where g_j is the Lagrange interpolation polynomial with the property $g_j(x_i)=\delta_{i,j}$ (see (4.15) and Fig. 4.2). As in the stationary case we require that the residual $u_t-cu_x-Du_{xx}$ is zero at the Fourier collocation points (4.10). This requirement leads to the system of differential equations

$$\begin{aligned}\frac{\mathrm{d}u(x_j,t)}{\mathrm{d}t}&=c(x_j)I_N\frac{\partial}{\partial x}\left[I_Nu(x_j,t)\right]+D(x_j)I_N\frac{\partial^2}{\partial x^2}\left[I_Nu(x_j,t)\right]\\&=\sum_{k=0}^{N-1}\left[c(x_j)\left(D_N^{(1)}\right)_{jk}+D(x_j)\left(D_N^{(2)}\right)_{jk}\right]u(x_k,t),\end{aligned}$$

describing the time evolution of $u(x_j, t)$. The first row of the equation fulfills the promises of the introduction to this section: the derivatives of a function are represented by analytic derivatives of its interpolation polynomial. The second row contains the instructions on how to solve the system. The derivatives at the right can be evaluated by direct multiplication of matrices (11.12) and (11.13) by the vector $\boldsymbol{u}(t)$, but one is better off by using the discrete Fourier transformation: we first transform the values $u(x_j, t)$ to Fourier space, multiply the obtained coefficients $\widetilde{u}_k(t)$ by $\mathrm{i}k$ and $(\mathrm{i}k)^2$ (for first and second derivative, respectively), and map the computed products back to configuration space, where we ultimately multiply them by $c(x_j)$ or $D(x_j)$, and the right-hand side is ready to go. Finally, we follow the time evolution of the values $u(x_j, t)$ with the initial condition

$$u(x_j, 0) = f(x_j)$$

by using methods for initial-value problems for systems of ordinary differential equations. The examples of solving the diffusion equation (the above problem with $c = 0$) can be found in Problems 11.9.6 and 11.9.7. Collocation methods are ideally suited for the solution of non-linear problems: this is discussed in Sect. 11.5.

11.4.3 Spectral Elements: Collocation with B-Splines

In the ansatz for the collocation solution one is not limited to trigonometric functions or orthogonal polynomials. A special class of methods is based on approximating the solution by a sum of localized low-degree polynomials. Instead of global, local functions can be used that are defined only on specific parts of the domain, and zero elsewhere. We are referring to *spectral element methods* [10]. For the lowest polynomial degrees (linear functions) they are identical to finite element methods described in Sect. 10.6.

In the following we present a method in which the spectral elements are cubic *B-splines*. Because the solution approximation is expressed by values at characteristic points (maxima) of these polynomial splines, we in fact still witness "collocation". As an example, again take the diffusion equation $v_t = Dv_{xx}$ on the interval $x \in [0, L]$ with Dirichlet boundary conditions $v(0, t) = v(L, t) = 0$ (see also Problem 11.9.6). The solution is sought in the form [11]

$$u(x, t) = \sum_{k=-1}^{N+1} a_k(t) B_k(x), \tag{11.54}$$

where B_j is the cubic spline centered at $x = x_j$ on the mesh $x_j = j\Delta x$ ($j = 0, 1, \ldots, N$ and $\Delta x = L/N$). This special class of basis functions already appeared in Chap. 8 on scalar boundary-value problems with ordinary differential equations (see (8.64), Fig. 8.9, and Sect. 8.5 in general). We require that the expansion (11.54)

satisfies the differential equation and the boundary conditions. Inserting the expansion (11.54) in the differential equation and evaluating the result at $x = x_j$, we get

$$\sum_{k=-1}^{N+1} \frac{\mathrm{d}a_k(t)}{\mathrm{d}t} B_k(x_j) = D \sum_{k=-1}^{N+1} a_k(t) B_k''(x_j), \quad j = 0, 1, \ldots, N,$$

where $'$ denotes the derivative with respect to x. When properties of B-splines are accounted for, one ends up with a system of differential equations for the coefficients $a_j(t)$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\big[a_{j-1}(t) + 4a_j(t) + a_{j+1}(t)\big] = \frac{6D}{\Delta x^2}\big(a_{j-1}(t) - 2a_j(t) + a_{j+1}(t)\big),$$

where $j = 0, 1, \ldots, N$. From the boundary condition at $x = 0$ we determine $a_{-1} = -4a_0 - a_1$. When the above equation is used at $j = 0$, it also follows that $a_0 = 0$ and $a_{-1} = -a_1$. Similarly, at $x = L$ we see that $a_N = 0$ and $a_{N+1} = -a_{N-1}$, and end up with a system of linear equations

$$A\frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} = B\boldsymbol{a},$$

where $\boldsymbol{a}(t) = (a_1(t), a_2(t), \ldots, a_{N-1}(t))^{\mathrm{T}}$ and

$$A = \begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{pmatrix}, \qquad B = \frac{6D}{\Delta x^2} \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix}.$$

The initial condition for the differential equation is $u(x_j, 0) = g(x_j)$, so the initial approximation for the collocation approximation is

$$A\boldsymbol{a}(0) = 6\boldsymbol{g},$$

where $\boldsymbol{g} = (g(x_1), g(x_2), \ldots, g(x_{N-1}))^{\mathrm{T}}$. The coefficients $a_k(t)$ can then be evolved in time by using some method discussed in Chap. 7. The solution at arbitrary time and location is given by the sum (11.54).

11.5 Non-linear Equations

The spectral expansion coefficients in Galerkin and tau methods result from solving systems of algebraic equations and systems of differential equations, respectively. For differential equations with general boundary conditions, non-constant coefficients, and in particular for non-linear equations, these systems become cumbersome or even impossible to write down (e.g. Fourier–Galerkin treatment of

$v_t = \mathrm{e}^v v_x$). Non-linear problems are therefore most frequently solved by collocation (pseudospectral) methods. Galerkin and tau methods do turn out to be effective for certain non-linear problems, although complications appear that are not present when solving linear problems.

Burgers Equation In the following we describe the typical approaches to non-linear problems in the case of the Burgers equation,

$$Lv = \frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} - D\frac{\partial^2 v}{\partial x^2} = 0, \quad x \in \Omega,\ t > 0,$$

where D is a positive constant and $v(x,0) = f(x)$ is the initial condition. We are interested in periodic solutions on $\Omega = [0, 2\pi]$ and in non-periodic solutions with Dirichlet boundary conditions on $\Omega = [-1, 1]$. For the periodic problem both Fourier–Galerkin and Fourier collocation methods are suitable; for the Dirichlet problem we will use the Chebyshev tau and collocation methods.

Fourier–Galerkin Method In this method the approximation u of the exact solution v has the form (11.41). When the Galerkin condition $\langle Lu, \mathrm{e}^{-\mathrm{i}kx}\rangle = 0$ is enforced and the equation for the Fourier coefficients (4.7) is used, we obtain a system of differential equations for the coefficients $\widehat{u}_k(t)$:

$$\frac{\mathrm{d}\widehat{u}_k}{\mathrm{d}t} + \widehat{\left(u\frac{\partial u}{\partial x}\right)}_k + Dk^2\widehat{u}_k = 0, \quad -N/2 \le k \le N/2 - 1. \tag{11.55}$$

The corresponding initial conditions are

$$\widehat{u}_k(0) = \frac{1}{2\pi}\int_0^{2\pi} f(x)\mathrm{e}^{-\mathrm{i}kx}\,\mathrm{d}x. \tag{11.56}$$

The diffusion term $-Dv_{xx}$ is linear and turns to $Dk^2\widehat{u}_k$ in transform space, as expected from (11.9), while the non-linear advection term vv_x generates the characteristic "compound coefficient"

$$\widehat{\left(u\frac{\partial u}{\partial x}\right)}_k = \frac{1}{2\pi}\int_0^{2\pi} u\frac{\partial u}{\partial x}\mathrm{e}^{-\mathrm{i}kx}\,\mathrm{d}x. \tag{11.57}$$

This is a special case of the more general coefficient

$$(\widehat{uv})_k = \frac{1}{2\pi}\int_0^{2\pi} u(x)v(x)\mathrm{e}^{-\mathrm{i}kx}\,\mathrm{d}x,$$

where u and v are trigonometric polynomials. By using discrete Fourier expansion and the orthogonality of basis functions, one immediately realizes

$$(\widehat{uv})_k = \frac{1}{2\pi}\int_0^{2\pi}\sum_l \widehat{u}_l\mathrm{e}^{\mathrm{i}lx}\sum_m \widehat{u}_m\mathrm{e}^{\mathrm{i}mx}\mathrm{e}^{-\mathrm{i}kx} = \sum_{l,m}\widehat{u}_l\widehat{u}_m\delta_{l+m,k} = \sum_{l+m=k}\widehat{u}_l\widehat{v}_m,$$

which is nothing but a convolution of discrete functions. A naive evaluation of this formula requires $\mathcal{O}(N^2)$ operations, while a computation exploiting FFT requires only $\mathcal{O}(N \log_2 N)$ (see Sect. 6.6 and Sect. 3.4 in [3]). This logarithmic discount is particularly relevant in two- and three-dimensional problems.

Another trap is hidden in the terms with non-constant coefficients in the differential equation (e.g. $a(x)v_x$) or in the non-linear terms (e.g. vv_x). If high Fourier modes are present in the expansions of the functions a and v, the terms $a(x)v_x$ and vv_x correspond to even higher modes, which may result in aliasing effects (Sect. 4.2.3). For example, if $v(x) = \sin kx$, the non-linear term becomes $vv_x = (k/2)\sin 2kx$: we witness frequency doubling. Aliasing can be removed by two relatively simple tricks described in Sect. 3.4 of [3].

Chebyshev Tau Method The tau method involving Chebyshev polynomials can be applied to the Burgers equation with Dirichlet boundary conditions $v(-1,t) = v_-(t)$ and $v(1,t) = v_+(t)$, and the initial condition $v(x,0) = f(x)$. The solution is sought in the form (11.50). We require that the expression $u_t + uu_x - Du_{xx}$ is orthogonal to all test functions T_k, thus

$$\int_{-1}^{1} \left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - D\frac{\partial^2 u}{\partial x^2} \right) T_k(x) \frac{1}{\sqrt{1-x^2}}\,\mathrm{d}x = 0, \quad k = 0, 1, \ldots, N-2.$$

Think: why does the index k run only up to $N-2$? We obtain a system of linear differential equations for the expansion coefficients,

$$\frac{\mathrm{d}\widehat{u}_k}{\mathrm{d}t} + \widehat{\left(u\frac{\partial u}{\partial x} \right)}_k - D\widehat{u}_k^{(2)} = 0, \quad k = 0, 1, \ldots, N-2,$$

where $\widehat{u}_k^{(2)}$ is given by (11.19). This system for $N-1$ coefficients is supplemented by the relations expressing the boundary conditions

$$\sum_{k=0}^{N} \widehat{u}_k(t) = v_-(t), \qquad \sum_{k=0}^{N} (-1)^k \widehat{u}_k(t) = v_+(t),$$

as well as by the initial condition for all $N+1$ coefficients,

$$\widehat{u}_k(0) = \frac{2}{\pi c_k} \int_{-1}^{1} f(x) T_k(x) \frac{1}{\sqrt{1-x^2}}\,\mathrm{d}x, \quad k = 0, 1, \ldots, N.$$

The contribution of the non-linear term,

$$\widehat{\left(u\frac{\partial u}{\partial x} \right)}_k = \frac{2}{\pi c_k} \int_{-1}^{1} \left(u\frac{\partial u}{\partial x} \right) T_k(x) \frac{1}{\sqrt{1-x^2}}\,\mathrm{d}x,$$

is again a special case of an expression that is evaluated as the convolution sum,

$$(\widehat{uv})_k = \frac{2}{\pi c_k} \int_{-1}^{1} u(x)v(x)T_k(x) \frac{1}{\sqrt{1-x^2}}\,\mathrm{d}x = \frac{1}{2} \sum_{l+m=k} \widehat{u}_l \widehat{v}_m + \sum_{|l-m|=k} \widehat{u}_l \widehat{v}_m.$$

Effective means to compute this sum and protect against aliasing are discussed by [3] in Sect. 3.4. With some effort (and by resorting to recurrence relations for Chebyshev polynomials) even the non-linear term can be expressed with the coefficients $\widehat{u}_k^{(1)}$ from (11.18), and thus the evaluation of convolutions can be avoided: see Sect. 7.2 in [4].

Fourier Collocation The collocation solution of the Burgers equation with the initial condition $v(x,0) = f(x)$ and periodic boundary conditions is represented by the values at the collocation points $x_j = 2\pi j/N$, where $j = 0, 1, \ldots, N-1$, and by its interpolation (11.53) if needed. The collocation requirement is

$$\left[\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - D\frac{\partial^2 u}{\partial x^2}\right]\Bigg|_{x=x_j} = 0, \quad j = 0, 1, \ldots, N-1,$$

while the initial conditions are $u(x_j, 0) = f(x)$. If the solution components are arranged in the vector $\boldsymbol{u} = (u_0(t), u_1(t), \ldots, u_{N-1}(t))^{\mathrm{T}}$, the collocation condition translates to a system of equations

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} + \boldsymbol{u} \otimes D_N^{(1)}\boldsymbol{u} - DD_N^{(2)}\boldsymbol{u} = \boldsymbol{0},$$

where $D_N^{(1)}$ and $D_N^{(2)}$ are the Fourier collocation derivative matrices (see (11.12) and (11.13)). The corresponding contributions can be computed directly by matrix multiplication, or by transformation to Fourier space, multiplication by $(\mathrm{i}k)$ or $(-k^2)$, followed by the inverse transformation (see Sect. 11.1.1).

Chebyshev Collocation In a similar manner one can tackle the Burgers equation with Dirichlet boundary conditions by Chebyshev collocation. We represent the solution by the values at the Gauss–Lobatto collocation points (4.40) and require that the differential equation is fulfilled at those points,

$$\left[\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - D\frac{\partial^2 u}{\partial x^2}\right]\Bigg|_{x=x_j} = 0, \quad j = 1, 2, \ldots, N-1.$$

This requirement is supplemented by the boundary conditions $u(x_0, t) = u_+(t)$ and $u(x_N, t) = u_-(t)$, and the initial condition

$$u(x_j, 0) = f(x_j), \quad j = 0, 1, \ldots, N.$$

When the solution is arranged in the vector $\boldsymbol{u} = (u_0(t), u_1(t), \ldots, u_N(t))^{\mathrm{T}}$, the collocation condition gives the system of equations

$$Z_N\left(\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} + \boldsymbol{u} \otimes D_N^{(1)}\boldsymbol{u} - DD_N^{(2)}\boldsymbol{u}\right) = \boldsymbol{0},$$

where $D_N^{(1)}$ and $D_N^{(2)}$ are the first and second Chebyshev collocation derivative matrices (see equations on p. 582). Here Z_N is the matrix that, upon multiplication with a vector, sets its first and last component to zero.

11.6 Time Integration ⋆

In spectral methods for PDE we invariably use a spectral representation of the spatial part of the differential operator, while the time evolution is traced by using finite differences. (The spectral expansion can also be performed in the time coordinate: see [4].) All systems of differential equations for the evolution coefficients in this chapter are of the form $\mathrm{d}\boldsymbol{u}/\mathrm{d}t = L\boldsymbol{u}$, where L is a differential operator. The choice of the time step size Δt used in the integration schemes for such systems is closely related to the spectrum (the set of eigenvalues) of matrices that represent spectral derivatives in L. The necessary condition for the stability of such schemes is that the product of the spectral radius (maximum eigenvalue) and Δt is bounded,

$$\Delta t \rho(L) \leq C.$$

The explicit form of L and its spectrum depend on the type of the spectral method. The spectrum of the matrix (11.12) for the Fourier collocation derivative is

$$\big\{\mathrm{i}(-N/2+1), \ldots, -\mathrm{i}, 0, \mathrm{i}, \ldots, \mathrm{i}(N/2-1)\big\},$$

while the spectrum of the matrix (11.13) for the second derivative is

$$\big\{(-N/2+1)^2, \ldots, -1, 0, 1, \ldots, (N/2-1)^2\big\}.$$

The spectral radius for the mth derivative is

$$\rho\big(D_N^{(m)}\big) = \max\big|\lambda^{(m)}\big| = \left(\frac{N}{2}-1\right)^m = \mathcal{O}\big(N^m\big).$$

This means that the step Δt needed for a stable time integration of Fourier approximations of advection problems (terms with v_x) scales as $\Delta t \sim 1/N$, and for diffusion problems (terms with v_{xx}) as $\Delta t \sim 1/N^2$: if we double the number of Fourier basis functions (or collocation points) N, we must correspondingly decrease Δt.

The limitations in spectral methods based on orthogonal polynomials are even more severe: for explicit integration of Chebyshev or Legendre approximation of advection problems we have $\Delta t \sim 1/N^2$, while for diffusion problems $\Delta t \sim 1/N^4$. This behavior can be seen in Fig. 11.8 (left). It shows the spectrum of the differential operator $\mathrm{d}/\mathrm{d}x$, i.e. the set of eigenvalues of the matrix $D_N^{(1)}$ for the Chebyshev collocation derivative with Gauss–Lobatto nodes (see (11.21)) and boundary condition $u(x_0) = 0$ (the first row and first column in the matrix are zero). The eigenvalues are rescaled such that the maximum eigenvalue of $D_N^{(1)}$ remains in the absolute stability region of the RK4 method (see also Fig. 7.3). These runaway eigenvalues that asymptotically behave as $\lambda \sim \mathcal{O}(N^2)$ force us to advance in very short time steps Δt. Spectral methods therefore call for implicit integrators with unlimited stability regions.

Figure 11.8 (right) shows the eigenvalues of the differential operator $-\mathrm{d}^2/\mathrm{d}x^2$ represented by the Chebyshev collocation second derivative matrix $-D_N^{(2)}$. Its eigenvalues are real and positive. They are bounded by $0 < c_1 \leq \lambda \leq c_2 N^4$, so they asymptotically increase as $\lambda \sim \mathcal{O}(N^4)$.

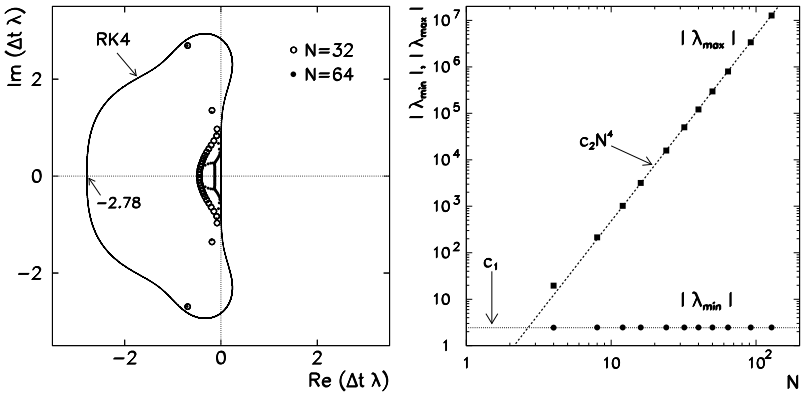


Fig. 11.8 [*Left*] The rescaled spectrum of the differential operator $\mathrm{d}/\mathrm{d}x$ in Chebyshev–Gauss–Lobatto collocation, embedded in the absolute stability region of the RK4 method. [*Right*] Minimal and maximal eigenvalues of the operator $-\mathrm{d}^2/\mathrm{d}x^2$

One is also allowed to use different time discretizations for different spatial parts of the differential operator. Instructive graphical representations of the spectral properties of differential operators can be found in Sect. 4.3 of [3], while the details on time discretization are discussed in [8] in Chaps. 9, 12, 13, and 14. A true point of contact between discrete and continuous differential operators is established by the concept of *pseudospectra* [12].

11.7 Semi-Infinite and Infinite Definition Domains ⋆

If spectral methods are used to solve problems with periodic solutions on $[0, 2\pi]$ or non-periodic solutions on $[-1, 1]$, the choice of basis functions is simple. Boyd [8] (see Chap. 1 in that book) advises:

1. When in doubt, use Chebyshev polynomials unless the solution is spatially periodic, in which case an ordinary Fourier series is better.
2. Unless you are sure another set of basis functions is better, use Chebyshev polynomials.
3. Unless you are really, really sure that another set of basis functions is better, use Chebyshev polynomials.

Spectral methods can also be used with problems on semi-infinite and infinite domains, but neither trigonometric not orthogonal polynomials are suitable basis functions. Variable transformations can be applied to map such problems to a finite interval on which Jacobi polynomials can be used [8], but it is much wiser to choose basis functions that are natural for the given definition domain.

On the interval $[-\infty, \infty]$ we can use the $\sin(nx)/x$ functions discussed in the review article [13]. A lovely set of model problems in more space dimensions has

been collected in [14]. Another option are the Hermite functions [15], as they are the eigenfunctions of the linear harmonic oscillator appearing in innumerable disguises in all fields of physics. Instructive examples of their use are the solution of the Burgers equation [16, 17] and the Dirac equation [18]. According to [19] the best choice could be the rational Chebyshev polynomials TB_k.

The natural choice for a basis on $[0, \infty]$ are the Laguerre functions [20]. These functions solve the radial part of the Schrödinger equation with the Coulomb potential: they are the staple diet in quantum physics and chemistry. Rational Chebyshev polynomials TL_k also lend themselves ideally to semi-infinite intervals [21]. A nice comparison of numerical treatments of the hydrogen atom in the Laguerre and TL_k polynomial bases has been made by [22]; see also [23].

11.8 Complex Geometries ⋆

In this chapter we discussed almost exclusively one-dimensional problems. We encountered two space dimensions only with the Fourier–Galerkin method in Sect. 11.2.4, where the extension to multiple dimensions (from the interval to the square or cube) was conceptually simple. In Problem 11.9.5 one can obtain a good two-dimensional impression while solving the Poisson equation by the Legendre tau method.

Cylindrical and spherical geometries, as well as all non-trivial geometries appearing in advanced physics problems require, of course, much greater involvement. Computations in cylindrical and spherical geometries are introduced by [8] in Chap. 18 and [9] in Chap. 6, but our Problem 11.9.4 may also serve as an appetizer. The book [7] gives appealing two-dimensional examples solved by MATLAB. As a very complete textbook on spectral methods in complex geometries we recommend [2].

11.9 Problems

11.9.1 Galerkin Methods for the Helmholtz Equation

This Problem is devoted to solving the one-dimensional Helmholtz equation

$$-\frac{\mathrm{d}^2 v}{\mathrm{d}x^2} + \lambda v = Q, \quad \lambda \geq 0,$$

by Galerkin spectral methods. The solution of the problem on $x \in [0, 2\pi]$ with the periodic boundary condition $v(0, t) = v(2\pi, t)$ can be expanded in terms of trigonometric polynomials, while the solution of the problem on $x \in [-1, 1]$ with homogeneous Dirichlet conditions $v(-1, t) = v(1, t) = 0$ can be constructed as a linear combination of Legendre or Chebyshev polynomials.

⊙ Solve the periodic problem by using the Fourier–Galerkin method described in Sect. 11.2.1, with the function

$$Q(x) = 4(\pi - x)\cos x + \big[(1+\lambda)(\pi - x)^2 - 2\big]\sin x.$$

The analytic solution is $v(x) = (\pi - x)^2 \sin x$ and does not depend on λ. Determine the Fourier coefficients (11.32) by analytic or numerical integration. Is the method of integration relevant for the precision of the final result if N is relatively small, for example, $N = 8$ or $N = 16$?

⊙ Solve the Dirichlet version of the problem with $Q(x) = x\sin(\pi x)$ by using the Legendre–Galerkin method from Sect. 11.2.2 and the Chebyshev–Galerkin method from Sect. 11.2.3. Try to solve the Legendre–Galerkin part by using the optimized basis functions (11.38).

11.9.2 Galerkin Methods for the Advection Equation

The one-dimensional linear hyperbolic equation

$$\frac{\partial v}{\partial t} - \frac{\partial v}{\partial x} = 0, \quad x \in (0, 2\pi),$$

with the periodic boundary condition $v(0,t) = v(2\pi,t)$ and specified initial condition $v(x,0)$ is a benchmark problem for spectral methods (e.g. Fourier–Galerkin) for non-stationary problems. The Galerkin condition with the approximation (11.41) leads to an uncoupled system of ordinary differential equations (11.42) that can be solved by methods for initial-value problems.

⊙ Apply the Fourier–Galerkin method to solve the problem outlined above, with the initial condition $v(x,0) = \sin(\pi\cos x)$. The analytic solution [3] is

$$v(x,t) = \sin\big[\pi\cos(x+t)\big],$$

and it corresponds to the Fourier expansion

$$v(x,t) = \sum_{k=-\infty}^{\infty} \widehat{v}_k(t)\mathrm{e}^{\mathrm{i}kx}, \quad \widehat{v}_k(t) = \sin\left(\frac{k\pi}{2}\right) J_k(\pi)\mathrm{e}^{\mathrm{i}kt}.$$

Of course, the spectral approximation will have a finite N, e.g. $N = 16$. Compare the numerical and analytic solutions at long times (large multiple of 2π). Determine the speed of convergence of the numerical solution to the analytic one when N increases. How important is the choice of the integration method? What role does the time step size play? Solve the problem by using some difference method and establish a comparison similar to that in Fig. 11.5!

⊙ Use the Chebyshev–Galerkin method to solve the problem

$$\frac{\partial v}{\partial t} - \frac{\partial v}{\partial x} = 0, \quad x \in (-1, 1),$$

with the boundary condition $v(1,t)=0$ and initial condition $v(x,0)=\sin\pi x$. This time the spectral solution can be cast in the form [4]

$$u(x,t)=\sum_{k=1}^{N} a_k(t)\phi_k(x),$$

where the basis functions are $\phi_k(x)=T_k(x)-1$ (since $\phi_0(x)=0$, the summation starts at $k=1$). According to Galerkin we require the residual $R=u_t-u_x$ to be orthogonal to ϕ_n in the sense of the scalar product

$$\langle R,\phi_n\rangle_w=\frac{2}{\pi}\int_{-1}^{1}(u_t-u_x)\phi_n(x)\frac{1}{\sqrt{1-x^2}}\,\mathrm{d}x=0,\quad n=1,2,\ldots,N.$$

When orthogonality relations are applied, the following system of differential equations emerges:

$$\sum_{k=1}^{N} M_{nk}\frac{\mathrm{d}a_k(t)}{\mathrm{d}t}=\sum_{k=1}^{N} S_{nk}a_k(t),\quad n=1,2,\ldots,N,$$

where the coefficient matrices are

$$M_{nk}=\frac{2}{\pi}\int_{-1}^{1}\phi_n(x)\phi_k(x)\frac{1}{\sqrt{1-x^2}}\,\mathrm{d}x=2+\delta_{n,k},$$

$$S_{nk}=\frac{2}{\pi}\int_{-1}^{1}\phi_n(x)\phi_k'(x)\frac{1}{\sqrt{1-x^2}}\,\mathrm{d}x=2k\sum_{\substack{p=0\\ p+k\ \text{odd}}}^{k-1}(\delta_{n,p}-\delta_{0,p}).$$

When the expansion coefficients are arranged in the vector $\boldsymbol{a}=(a_1,a_2,\ldots,a_N)^{\mathrm{T}}$ the system of differential equations can be written in matrix form

$$\frac{\mathrm{d}\boldsymbol{a}(t)}{\mathrm{d}t}=M^{-1}S\boldsymbol{a}(t),\quad a_k(0)=\frac{2}{\pi}\int_{-1}^{1}f(x)\phi_k(x)\frac{1}{\sqrt{1-x^2}}\,\mathrm{d}x.$$

Solve it by using standard methods for initial-value problems.

11.9.3 Galerkin Method for the Diffusion Equation

Check your understanding of Galerkin methods by solving the diffusion equation

$$v_t=Dv_{xx},\quad x\in(0,2\pi),$$

with boundary condition $v(0,t)=v(2\pi,t)$ and initial condition $v(x,0)=f(x)$.

⊙ Using the ansatz (11.41) in the Galerkin condition leads to the system (11.44) (with the simplification $c = 0$) that should be solved with the initial condition (11.43). Solve it by using the initial condition $f(x) = x^2(2\pi - x)^2$ and $D = 1.0$. Plot the solution at selected times $t \in [0, 10]$.

⊙ To solve the diffusion equation on $x \in (-1, 1)$ with Dirichlet boundary conditions $v(-1, t) = v(1, t) = 0$, we choose the Legendre–Galerkin approach. The solution ansatz is

$$u(x,t) = \sum_{n=1}^{N-1} a_n(t)\phi_n(x), \tag{11.58}$$

with the basis functions $\phi_n(x) = P_{n+1}(x) - P_{n-1}(x)$, where $n \geq 1$. We require the residual $u_t - u_{xx}$ to be orthogonal to all functions ϕ_k, hence

$$\frac{2k+1}{2}\int_{-1}^{1}\left(\frac{\mathrm{d}a_n}{\mathrm{d}t}\phi_n(x) - a_n\frac{\mathrm{d}^2\phi_n(x)}{\mathrm{d}x^2}\right)\phi_k(x)\,\mathrm{d}x = 0, \quad k = 1, 2, \ldots, N-1.$$

By considering the orthogonality of Legendre polynomials, we get the system

$$\sum_{n=1}^{N-1} M_{kn}\frac{\mathrm{d}a_n(t)}{\mathrm{d}t} = \sum_{n=1}^{N-1} S_{kn}a_n(t), \quad k = 1, 2, \ldots, N-1,$$

with the matrices

$$M_{kn} = \frac{2k+1}{2}\int_{-1}^{1}\phi_k(x)\phi_n(x)\,\mathrm{d}x,$$
$$S_{kn} = \frac{2k+1}{2}\int_{-1}^{1}\phi_k(x)\frac{\mathrm{d}^2\phi_n(x)}{\mathrm{d}x^2}\,\mathrm{d}x.$$

When the expansion coefficients are arranged in the vector $\boldsymbol{a} = (a_1, a_2, \ldots, a_{N-1})^{\mathrm{T}}$, this system of differential equations can be written in matrix form

$$\frac{\mathrm{d}\boldsymbol{a}(t)}{\mathrm{d}t} = M^{-1}S\boldsymbol{a}(t),$$

and solved by standard methods for initial-value problems. The initial condition vector can be computed from the expansion of $u(x, 0) = f(x)$ in terms of the functions ϕ_n as in (11.58), which leads to the recurrence

$$a_{n+1}(0) = a_{n-1}(0) - \frac{2n+1}{2}\int_{-1}^{1} f(x)P_n(x)\,\mathrm{d}x, \quad a_{-1}(0) = a_0(0) = 0.$$

Solve this part of the Problem by using $f(x) = (1-x)^2\cos^4(\pi x/2)$.

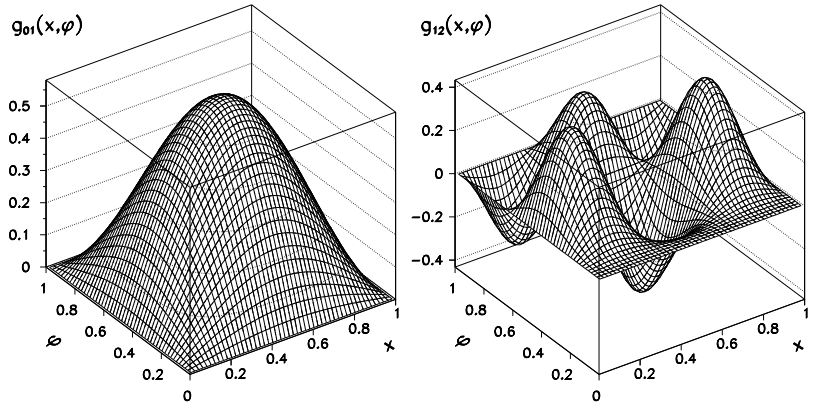


Fig. 11.9 Examples of eigenfunctions (11.59) for the Poisson equation in semi-circular geometry. [*Left*] The function $g_{01}(x, \varphi)$. [*Right*] The function $g_{12}(x, \varphi)$

11.9.4 Galerkin Method for the Poisson Equation: Poiseuille Law

The flow of a viscous fluid in a long straight pipe driven by a pressure gradient p' is described by the Poisson equation $\nabla^2 v = -p'/\eta$, where v is the longitudinal velocity component that depends only on the coordinates in the plane of the pipe's cross-section, and η is the viscosity. The fluid velocity at the pipe's walls is zero. The flux is given by the Poiseuille law

$$\Phi = \int v \, \mathrm{d}S = \frac{C}{8\pi} \frac{S^2}{\eta} p',$$

where the coefficient C depends on the shape of the cross-section. For a circular pipe, $C = 1$; but what is the coefficient for a *semi-circular* pipe with the radius R? Changing to variables $x = r/R$ and $u = v\eta/(p'R^2)$, the problem becomes

$$\nabla^2 u(x, \varphi) = -1, \quad u(1, \varphi) = u(x, 0) = u(x, \pi) = 0,$$

and the flux coefficient is given by

$$C = 8\pi \int_0^1 \int_0^\pi \frac{u(x, \varphi) x \, \mathrm{d}x \, \mathrm{d}\varphi}{(\pi/2)^2}.$$

The eigenfunctions of the Poisson equation for the semi-circular geometry are

$$g_{ms}(x, \varphi) = J_{2m+1}(y_{ms} x) \sin(2m + 1)\varphi, \tag{11.59}$$

where y_{ms} is the sth zero of the $(2m + 1)$th Bessel function. The functions g_{01} and g_{12} are shown as examples in Fig. 11.9.

By using these functions the solution can be written as [24]

$$u(x,\varphi)=\sum_{ms}\frac{A_{ms}g_{ms}(x,\varphi)}{y_{ms}^2},\quad m=0,1,2,\ldots,\ s=1,2,\ldots,$$

where

$$A_{ms}=\frac{\langle 1,g_{ms}\rangle}{\langle g_{ms},g_{ms}\rangle},$$

and the scalar product is $\langle u,v\rangle=\iint u(x,\varphi)v(x,\varphi)x\,\mathrm{d}x\,\mathrm{d}\varphi$. The flux coefficient is then

$$C=8\sum_{ms}\left[\frac{8}{\pi}\frac{I_{ms}}{(2m+1)y_{ms}J_{2m+2}(y_{ms})}\right]^2.$$

To compute the integral $I_{ms}=\frac{2m+1}{2}\langle 1,g_{ms}\rangle$, use the relation

$$I_{ms}=\int_0^1 xJ_{2m+1}(y_{ms}x)\,\mathrm{d}x=\frac{4(2m+1)}{y_{ms}}\sum_{k=m+1}^{\infty}\frac{kJ_{2k}(y_{ms})}{(4k^2-1)}.$$

⊙ Had we not known Bessel functions (and had we been ignorant of their eigenfunction property), we could appeal to Galerkin to find an approximate solution. We write the solution as a linear combination of trial functions

$$u(x,\varphi)=\sum_k a_k\phi_k. \tag{11.60}$$

The functions ϕ_k need not be orthogonal, but they should satisfy the boundary conditions so that they are also satisfied by the sum (11.60). We keep the exact functions $\sin(2m+1)\varphi$ for the angular part, while we take $x^{2m+1}(1-x)^n$ for the radial part. According to Galerkin, the residual

$$R(x,\varphi)=\nabla^2u(x,\varphi)+1$$

should be orthogonal to all test functions $\psi_j=\phi_j$, hence $\langle R,\psi_j\rangle=0$. This leads to the system of equations for the coefficients a_i,

$$\sum_j A_{ij}a_j=b_i,\quad A_{ij}=\langle\psi_i,\nabla^2\psi_j\rangle,\ b_i=\langle -1,\psi_i\rangle,$$

and the flux coefficient is

$$C=-\frac{32}{\pi}\sum_i\sum_j b_iA_{ij}^{-1}b_j.$$

The indices i and j are "double": both run through the complete set of m and n. Due to orthogonality in m the matrix A has a block structure. How does the precision of C depend on the number of included angular and radial terms?

⊙ Try to compute the coefficient C for a rectangular pipe.

11.9.5 Legendre Tau Method for the Poisson Equation

Tau methods are well suited for solving the two-dimensional Poisson equation

$$-\nabla^2 v = Q$$

on the square $(x, y) \in [-1, 1] \times [-1, 1]$ with Dirichlet boundary conditions

$$v(x, -1) = B_1(x), \qquad v(x, 1) = B_2(x),$$
$$v(-1, y) = B_3(y), \qquad v(1, y) = B_4(y).$$

Both Chebyshev and Legendre polynomials can be used as trial functions, even if they do not satisfy the boundary conditions: in tau methods they are established by separate equations. In this Problem the two-dimensional trial functions are products of Legendre polynomials

$$\phi_{jk}(x, y) = P_j(x) P_k(y), \quad j, k = 0, 1, \ldots, N,$$

and we seek the solution in the form

$$u(x, y) = \sum_{j=0}^{N} \sum_{k=0}^{N} a_{jk} \phi_{jk}(x, y). \tag{11.61}$$

In this case we need one set of test functions for the representation of the differential equation, and another set to impose the boundary conditions. The test functions for the equation are

$$\psi_{jk}(x, y) = \widetilde{P}_j(x) \widetilde{P}_k(y), \quad j, k = 0, 1, \ldots, N-2,$$

where

$$\widetilde{P}_k(x) = \left(k + \frac{1}{2}\right) P_k(x),$$

(the factor $k + 1/2$ originates in the relation $(k + \frac{1}{2}) \int_{-1}^{1} P_j(x) P_k(x)\, \mathrm{d}x = \delta_{j,k}$). The test functions for the boundary conditions depend only on one variable,

$$\psi_j^{(1)}(x) = \psi_j^{(2)}(x) = \widetilde{P}_j(x), \quad j = 0, 1, \ldots, N,$$
$$\psi_k^{(3)}(y) = \psi_k^{(4)}(y) = \widetilde{P}_k(y), \quad k = 0, 1, \ldots, N.$$

From the minimization condition for the residual we obtain

$$\int_{-1}^{-1} \int_{-1}^{-1} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + Q(x, y) \right) \psi_{jk}(x, y)\, \mathrm{d}x\, \mathrm{d}y = 0, \quad j, k = 0, 1, \ldots, N-2,$$
$$\int_{-1}^{-1} B_1(x) \psi_j^{(1)}(x)\, \mathrm{d}x = \int_{-1}^{-1} B_2(x) \psi_j^{(2)}(x)\, \mathrm{d}x = 0, \quad j = 0, 1, \ldots, N,$$

$$\int_{-1}^{-1} B_3(y)\psi_k^{(3)}(x)\,\mathrm{d}y = \int_{-1}^{-1} B_4(y)\psi_k^{(4)}(x)\,\mathrm{d}y = 0, \quad k = 0, 1, \ldots, N.$$

With the chosen expansion of u the integrals can be computed analytically [2]. For the part pertaining to the differential equation one gets

$$-\left[a_{jk}^{[2,0]} + a_{jk}^{[0,2]}\right] = Q_{jk}, \quad j, k = 0, 1, \ldots, N-2, \tag{11.62}$$

where

$$a_{jk}^{[2,0]} = \left(j + \frac{1}{2}\right) \sum_{\substack{p=j+2 \\ p+j \text{ even}}}^{N} \left[p(p+1) - j(j+1)\right] a_{pk}, \tag{11.63}$$

$$a_{jk}^{[0,2]} = \left(k + \frac{1}{2}\right) \sum_{\substack{q=k+2 \\ q+k \text{ even}}}^{N} \left[q(q+1) - k(k+1)\right] a_{jq}, \tag{11.64}$$

$$Q_{jk} = \int_{-1}^{-1} \int_{-1}^{-1} Q(x, y)\psi_{jk}(x, y)\,\mathrm{d}x\,\mathrm{d}y.$$

This is a generalization of the coefficients $\widehat{u}_k^{(2)}$ from (11.15) to two dimensions: the second derivative in x (and zeroth in y) corresponds to the coefficients (11.63); the second derivative in y (and zeroth in x) corresponds to (11.64). Both include linear combinations of coefficients in (11.61), and (11.62) is the system for these coefficients. Boundary conditions generate constraints supplementing the system of equations, rendering the final solution unique:

$$\sum_{k=0}^{N} a_{jk} = \sum_{k=0}^{N} (-1)^k a_{jk} = 0, \quad j = 0, 1, \ldots, N,$$

$$\sum_{j=0}^{N} a_{jk} = \sum_{j=0}^{N} (-1)^j a_{jk} = 0, \quad k = 0, 1, \ldots, N.$$

⊙ Solve the Dirichlet problem for the Poisson equation with the Legendre tau method described above, by using $Q(x, y) = 2\pi^2 \sin(\pi x)\sin(\pi y)$. The analytic solution is $u(x, y) = \sin(\pi x)\sin(\pi y)$.

11.9.6 Collocation Methods for the Diffusion Equation I

In this Problem we are solving the one-dimensional diffusion equation that describes the temperature profile $T(x, t)$ in a homogeneous infinite slab of thickness L:

$$\frac{\partial T}{\partial t} = D\frac{\partial^2 T}{\partial x^2} + \frac{Q}{\rho c}, \quad 0 < x < L, \; D = \frac{\lambda}{\rho c}.$$

For the time being we do not consider additional heat sources, $Q(x,t) = 0$. The temperature at an arbitrary point x at time t is given by the Fourier series

$$T(x,t) \approx \sum_{k=0}^{N-1} \widehat{T}_k(t) \mathrm{e}^{2\pi \mathrm{i} f_k x},$$

where $f_k = k/L$. When this ansatz is inserted in the Galerkin requirement, we get the evolution equation for the Fourier coefficients:

$$\frac{\mathrm{d}\widehat{T}_k(t)}{\mathrm{d}t} = D\big(-4\pi^2 f_k^2\big)\widehat{T}_k(t).$$

The explicit Euler method is used to advance in time,

$$\widehat{T}_k(t + \Delta t) = \widehat{T}_k(t) + \Delta t\, D\big(-4\pi^2 f_k^2\big)\widehat{T}_k(t).$$

The temperature profile $T(x,t)$ at an arbitrary time can be computed by using the inverse Fourier transformation.

⊙ Use the Fourier method to compute the time evolution of the temperature profile with the initial condition $T(x,0) = \sin^2(\pi x/L)$. By using the Fourier transformation you get the initial condition for the evolution equation. Pay attention to the stability of the Euler difference scheme: at any time step maintain

$$\left|\frac{\widehat{T}_k(t + \Delta t)}{\widehat{T}_k(t)}\right| = \left|1 + \Delta t\, D\big(-4\pi^2 f_k^2\big)\right| < 1.$$

The discretization also requires some care: in FFT one should keep $f_k < f_{\mathrm{Nyquist}}$ for each k (see (4.6)).

In addition, solve the Dirichlet problem: the boundary conditions are $T(0,t) = T(L,t) = 0$, while at time zero the slab should have the ambient temperature T_0 everywhere, except between $0.2L$ and $0.4L$ where it has been heated up to temperature $T_1 > T_0$. Moreover, at time zero we switch on a heater between $0.5L$ and $0.75L$ with the power density of $5T_0\lambda/L^2$. The suitable eigenfunctions are the sine functions with multiples of half-waves on the interval. FFT implies an expansion in terms of sines *and* cosines, but it can still be used with Dirichlet boundary conditions if the function to be transformed is extended to an odd function on the interval $[-L, L]$.

⊙ Solve the problem by using the collocation method with B-splines discussed in Sect. 11.4.3. Use the initial condition $T(x,0) = g(x) = \sin(\pi x/L)$ and homogeneous Dirichlet boundary conditions $T(0,t) = T(L,t) = 0$. Evolve the expansion coefficients $a_j(t)$ in time by using the explicit Euler method: the initial condition for the vector of coefficients $\boldsymbol{a}$ is $A\boldsymbol{a}^0 = 6\boldsymbol{g}$. At subsequent times $n\Delta t$ we get

$$\boldsymbol{a}^{n+1} = \boldsymbol{a}^n + \Delta t\, A^{-1} B \boldsymbol{a}^n = \big(I + \Delta t\, A^{-1} B\big)\boldsymbol{a}^n.$$

The time profile at any later time can be computed by evaluating the sum (11.54). The implicit method is more stable, but at every step it requires you to solve

$$\left(A - \frac{\Delta t}{2} B\right) \boldsymbol{a}^{n+1} = \left(A + \frac{\Delta t}{2} B\right) \boldsymbol{a}^{n}.$$

11.9.7 Collocation Methods for the Diffusion Equation II

The homogeneous Dirichlet problem for the diffusion equation on $x \in [-1, 1]$,

$$\frac{\partial v}{\partial t} = \frac{\partial^2 v}{\partial x^2}, \quad v(-1, t) = v(1, t) = 0,$$

and initial condition $v(x, 0) = f(x)$ can be efficiently solved by collocation methods with orthogonal polynomials. The first choice for the trial functions are the Chebyshev polynomials, $\phi_k = T_k$, and the collocation solution acquires the form

$$u(x, t) = \sum_{k=0}^{N} a_k(t) T_k(x) = \sum_{j=0}^{N} u_j(t) l_j(x), \tag{11.65}$$

where $l_j(x)$ are the polynomials (4.42) interpolating at the Chebyshev–Gauss–Lobatto collocation points $x_j = \cos(j\pi/N)$ (see (4.40)). Their property is $l_j(x_k) = \delta_{j,k}$. Since the complete solution is represented as the weighted sum of the values at the nodes, we have denoted $u_j(t) = u(x_j, t)$.

One requires that the differential equation $u_t - u_{xx} = 0$ is fulfilled exactly at the collocation points [4],

$$\left[\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2}\right]\Bigg|_{x=x_j} = 0, \quad j = 1, 2, \ldots, N-1,$$

while the boundary conditions are $u(x_0, t) = u(x_N, t) = 0$. By using the expressions for the (second) Chebyshev collocation derivative (see p. 582) one obtains a system of differential equations for the values $u_j(t)$ at the nodes,

$$\frac{\mathrm{d}u_j(t)}{\mathrm{d}t} = \sum_{l=0}^{N} \left(D_N^{(2)}\right)_{jl} u_l(t), \quad j = 1, 2, \ldots, N-1, \tag{11.66}$$

that should be solved with the initial conditions

$$u_k(0) = u(x_k, 0) = f(x_k), \quad k = 0, 1, \ldots, N.$$

⊙ Solve the Dirichlet problem for the diffusion equation by using the Chebyshev collocation method with Gauss–Lobatto nodes. The initial condition

is $v(x,0) = f(x) = \sin \pi x$. The analytic solution is $v(x,t) = \mathrm{e}^{-\pi^2 t} \sin \pi x$ and it corresponds to the expansion

$$v(x,t) = \sum_{k=0}^{\infty} b_k(t) T_k(x), \quad b_k(t) = \frac{2}{c_k} \sin\left(\frac{k\pi}{2}\right) J_k(\pi) \mathrm{e}^{-\pi^2 t},$$

where $c_0 = 2$ and $c_k = 1$ for $k \geq 1$.

⊙ Collocation is not restricted to Dirichlet boundary conditions. This part of the Problem deals with the solution of the diffusion equation with more general boundary conditions

$$\alpha_1 v(-1,t) - \beta_1 v_x(-1,t) = g(t),$$
$$\alpha_2 v(1,t) + \beta_2 v_x(1,t) = h(t).$$

The solution again has the form (11.65), but Chebyshev polynomials should be substituted by Legendre polynomials, so for the Gauss–Lobatto nodes the corresponding interpolation polynomials are (4.35). By requiring that the differential equation is fulfilled at the collocation points, we obtain an equation like (11.66), except that the collocation derivative matrix is given by the expression on p. 581. The boundary conditions are embodied by two additional equations:

$$\alpha_1 u_0(t) - \beta_1 \sum_{j=0}^{N} \left(D_N^{(1)}\right)_{0j} u_j(t) = g(t),$$

$$\alpha_2 u_N(t) + \beta_2 \sum_{j=0}^{N} \left(D_N^{(1)}\right)_{Nj} u_j(t) = h(t).$$

A system of two equations for the remaining unknowns $u_0(t)$ and $u_N(t)$ follows:

$$\left[\alpha_1 - \beta_1 \left(D_N^{(1)}\right)_{00}\right] u_0(t) - \beta_1 \left(D_N^{(1)}\right)_{0N} u_N(t) = g(t) + \beta_1 \sum_{j=1}^{N-1} \left(D_N^{(1)}\right)_{0j} u_j(t),$$

$$\beta_2 \left(D_N^{(1)}\right)_{N0} u_0(t) + \left[\alpha_2 + \beta_2 \left(D_N^{(1)}\right)_{NN}\right] u_N(t) = h(t) - \beta_2 \sum_{j=1}^{N-1} \left(D_N^{(1)}\right)_{Nj} u_j(t).$$

11.9.8 Burgers Equation

An nice example of a non-linear advection-diffusion problem that can be solved by almost all spectral methods in this chapter is the Burgers equation

$$v_t + v v_x - D v_{xx} = 0$$

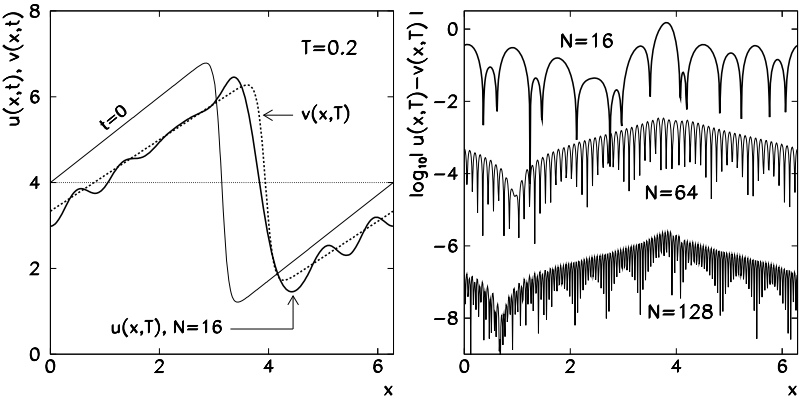


Fig. 11.10 Solution of the Burgers equation with periodic boundary conditions by the Fourier—Galerkin method. [*Left*] The initial condition and the comparison of numerical ($N = 16$) and analytic solution at $T = 0.2$. [*Right*] The error of the numerical solution along $[0, 2\pi]$ for different approximation levels (different number of basis functions N)

with the parameter $D > 0$ and initial condition $v(x, 0) = f(x)$. We are interested in the periodic solutions on the interval $[0, 2\pi]$ and non-periodic solutions on $[-1, 1]$ computed by using the methods discussed in Sect. 11.5.

⊙ Use the Fourier–Galerkin method described on p. 602 to solve the Burgers equation with periodic solutions on $[0, 2\pi]$. In computing the "compound coefficient" (11.57), recall the simple relation (11.9): the system of differential equations for the expansion coefficients $\widehat{u}_k(t)$ (11.55) then becomes

$$\frac{\mathrm{d}\widehat{u}_k}{\mathrm{d}t} + \mathrm{i} \sum_{m+l=k} m \widehat{u}_m \widehat{u}_l + Dk^2 \widehat{u}_k = 0, \quad -N/2 \le k \le N/2 - 1,$$

with the initial conditions (11.56). The analytic solution is given by

$$v(x, t) = c + v_\mathrm{b}(x - ct, t + t_0),$$

where

$$v_\mathrm{b}(x, t) = -2D \frac{1}{\phi_\mathrm{b}} \frac{\partial \phi_\mathrm{b}}{\partial x}, \quad \phi_\mathrm{b}(x, t) = \frac{1}{\sqrt{4\pi Dt}} \sum_{n=-\infty}^{\infty} \mathrm{e}^{-(x-2\pi n)^2/(4Dt)}.$$

Use $N = 16$, 32, and 64 with $D = 0.2$, $c = 4$, and $t_0 = 1$. The initial condition is the analytic solution at $t = 0$. Compute the solution until $t = \pi/8$ (Fig. 11.10 might serve as an example). Solve the problem by Fourier collocation as well.

⊙ Use the Chebyshev tau and Chebyshev collocation methods to solve the Burgers equation on $[-1, 1]$ with homogeneous Dirichlet boundary conditions

$v(-1,t) = v(1,t) = 0$. The analytic solution in this case is

$$v(x,t) = 4\pi D\left[\sum_{n=1}^{\infty} n a_n \mathrm{e}^{-n^2\pi^2 Dt}\sin n\pi x\right]\left[a_0 + 2\sum_{n=1}^{\infty} a_n \mathrm{e}^{-n^2\pi^2 Dt}\cos n\pi x\right]^{-1},$$

where the coefficients a_n are given by the modified Bessel functions of the first kind as $a_n = (-1)^n I_n(1/(2\pi D))$. The initial condition is the analytic solution at $t = 0$. Use $D = 0.1$ and compute with $N = 16$, 32, and 64 in both Chebyshev tau and collocation methods. Compute the solutions until $t = 1$.

References

1. D. Gottlieb, S.A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications* (SIAM, Philadelphia, 1987)
2. C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zang, *Spectral Methods. Evolution to Complex Geometries and Applications to Fluid Mechanics* (Springer, Berlin, 2007)
3. C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zang, *Spectral Methods. Fundamentals in Single Domains* (Springer, Berlin, 2006)
4. J. Hesthaven, S. Gottlieb, D. Gottlieb, *Spectral Methods for Time-Dependent Problems* (Cambridge University Press, Cambridge, 2007)
5. M. Abramowitz, I.A. Stegun, *Handbook of Mathematical Functions*, 10th edn. (Dover, Mineola, 1972)
6. R. Baltensperger, M.R. Trummer, Spectral differencing with a twist. SIAM J. Sci. Comput. **24**, 1465 (2002)
7. L.N. Trefethen, *Spectral Methods in Matlab* (SIAM, Philadelphia, 2000)
8. J.P. Boyd, *Chebyshev and Fourier Spectral Methods*, 2nd edn. (Dover, Mineola, 2000); the web edition and supplemental materials are available at http://www-personal.engin.umich.edu/~jpboyd
9. B. Fornberg, *A Practical Guide to Pseudospectral Methods* (Cambridge University Press, Cambridge, 1998). The concepts of "collocation" and "pseudospectral" methods are identical in our context, while a part of the expert community distinguishes between them and states that the only true pseudospectral methods are those involving global basis functions; see Sect. 3.5 in [3] and Sect. 3.1 in [8]
10. D. Funaro, *Spectral Elements for Transport-Dominated Equations*. Lecture Notes in Computational Science and Engineering, vol. 1 (Springer, Heidelberg, 1997)
11. M.H. Holmes, *Introduction to Numerical Methods in Differential Equations* (Springer, New York, 2007)
12. L.N. Trefethen, M. Embree, *Spectra and Pseudospectra* (Princeton University Press, Princeton, 2005)
13. F. Stenger, Numerical methods based on Whittaker cardinal, or sinc functions. SIAM Rev. **23**, 165 (1981)
14. K.M. McArthur, K.L. Bowers, J. Lund, The sinc method in multiple space dimensions: model problems. Numer. Math. **56**, 789 (1990)
15. D. Funaro, O. Kavian, Approximation of some diffusion evolution equations in unbounded domains by Hermite functions. Math. Comput. **57**, 597 (1991)
16. B.-Y. Guo, C.-L. Xu Math, Hermite pseudospectral method for nonlinear partial differential equations. Model. Numer. Anal. **34**, 859 (2000)
17. B.-Y. Guo, Error estimation of Hermite spectral method for nonlinear partial differential equations. Math. Comput. **68**, 1067 (1999)

18. B.-Y. Guo, J. Shen, C.-L. Xu, Spectral and pseudospectral approximations using Hermite functions: application to the Dirac equation. Adv. Comput. Math. **19**, 35 (2003)
19. J.P. Boyd, Spectral methods using rational basis functions on an infinite interval. J. Comput. Phys. **69**, 112 (1987)
20. J. Shen, Stable and efficient spectral methods in unbounded domains using Laguerre functions. SIAM J. Numer. Anal. **38**, 1113 (2000)
21. J.P. Boyd, Orthogonal rational functions on a semi-infinite interval. J. Comput. Phys. **70**, 63 (1987)
22. J.P. Boyd, C. Rangan, P.H. Bucksbaum, Pseudospectral methods on a semi-infinite interval with application to the hydrogen atom. J. Comput. Phys. **188**, 56 (2003)
23. V. Iranzo, A. Falqués, Some spectral approximations for differential equations in unbounded domains. Comput. Methods Appl. Mech. Eng. **98**, 105 (1992)
24. I. Kuščer, A. Kodre, H. Neunzert, *Mathematik in Physik und Technik* (Springer, Berlin, 1993)

Appendix A
Mathematical Tools

A.1 Asymptotic Notation

In asymptotic analysis we use several sets of functions that acquire their special meaning in the limit of a real parameter $x \to a$, where a is finite or infinite.

Symbol $\mathcal{O}(\cdot)$ The $\mathcal{O}(f)$ symbol denotes the set of functions $g \in \mathcal{O}(f)$ which, in the limit $x \to a$, are bounded from above in magnitude as $|g(x)| \leq C|f(x)|$, with a finite positive constant C. For a function g we thus have

$$\lim_{x \to a} \frac{|g(x)|}{|f(x)|} \in (0, C].$$

We say that the function g is *asymptotically bounded from above* by the function f up to a constant factor.

Symbol $o(\cdot)$ The $o(f)$ symbol denotes the set of functions $g \in o(f)$, where f dominates over g such that $|g(x)| < C|f(x)|$ for each $C > 0$, when $x \to a$. So for a function $g \in o(f)$ we have

$$\lim_{x \to a} \frac{|g(x)|}{|f(x)|} = 0.$$

We say that the function f is *asymptotically dominant* with respect to g. The following also holds:

$$o(f) + o(f) \subseteq o(f), \qquad o(f)o(g) \subseteq o(fg),$$
$$o\big(o(f)\big) \subseteq o(f), \qquad o(f) \subset \mathcal{O}(f).$$

Symbol $\Theta(\cdot)$ The $\Theta(f)$ symbol denotes the set of functions $g \in \Theta(f)$ which, in the limit $x \to a$, are bounded as $C_1|f(x)| \leq |g(x)| \leq C_2|f(x)|$ for two positive constants C_1 and C_2. Then for $g \in \Theta(f)$ we have

$$\lim_{x \to a} \frac{|g(x)|}{|f(x)|} \in [C_1, C_2],$$

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9, © Springer-Verlag Berlin Heidelberg 2012

i.e. f *asymptotically tightly bounds* g, which we also denote by $g \asymp f$.

Symbol $(\cdot \sim \cdot)$ The $\sim f(x)$ symbol denotes the set of functions $g \sim f$ that are asymptotically equivalent to the function f, so

$$\lim_{x \to a} \frac{f(x)}{g(x)} = 1.$$

The symbols defined above represent sets of functions, but in practice they are often used to express the relations among their elements. For example, we interpret $g \in \mathcal{O}(f)$ as $g = \mathcal{O}(f)$.

A.2 The Norms in Spaces $L^p(\Omega)$ and $L^p_w(\Omega)$, $1 \le p \le \infty$

Over a bounded open domain $\Omega \in \mathbb{R}^d$ $(d = 1, 2, 3)$ we define the space $L^p_w(\Omega)$ of measurable functions $u : \Omega \to \mathbb{R}$ which are Lebesgue-integrable on Ω, i.e. for which $\int_\Omega |u(\boldsymbol{x})|^p \, w(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} < \infty$, where w is a continuous, strictly positive and integrable weight function on Ω. The space $L^p_w(\Omega)$ for $1 \le p < \infty$, furnished with the norm

$$\|u\|_{L^p_w(\Omega)} = \left(\int_\Omega \left| u(\boldsymbol{x}) \right|^p w(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \right)^{1/p},$$

is a Banach space. (Most often we encounter one-dimensional cases $\Omega = [a, b]$, e.g. $\Omega = [0, 2\pi]$ in Fourier analysis, or $\Omega = [-1, 1]$ when working with orthogonal polynomials. The simplifications $\Omega \to [a, b]$ and $\boldsymbol{x} \to x$ are trivial.) Among these, the space $L^2_w(\Omega)$ is virtually ubiquitous: it is also a Hilbert space with the scalar product

$$\langle u, v \rangle_w = \int_\Omega u(\boldsymbol{x}) v(\boldsymbol{x}) w(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \tag{A.1}$$

and the induced weighted norm

$$\|u\|_{L^2_w(\Omega)} = \left(\int_\Omega \left| u(\boldsymbol{x}) \right|^2 w(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \right)^{1/2}.$$

The corresponding expressions for the spaces $L^p(\Omega)$ can be obtained simply by dropping all occurrences of the weight function w. We also encounter spaces of complex measurable functions $u : \Omega \to \mathbb{C}$. In this case, the expressions remain as they are, except that $|\cdot|$ stands for the modulus of the complex number instead of the usual absolute value; for example, in the case of $L^2(a, b)$, the scalar product and the norm are

$$\langle u, v \rangle_2 = \int_a^b u(x) v^*(x) \, \mathrm{d}x, \qquad \|u\|_2 = \int_a^b \left| u(x) \right|^2 \mathrm{d}x. \tag{A.2}$$

(We drop the index 2 when there is no danger of confusion with another norm.) In the family of spaces $L^p(\Omega)$, the extreme case $p = \infty$ represents a Banach space of measurable functions $u : \Omega \to \mathbb{R}$ such that $|u|$ is bounded outside of a set with measure zero, and is furnished with the norm

$$\|u\|_{L^\infty(\Omega)} = \sup_{\boldsymbol{x}\in\Omega} \big|u(\boldsymbol{x})\big| = M.$$

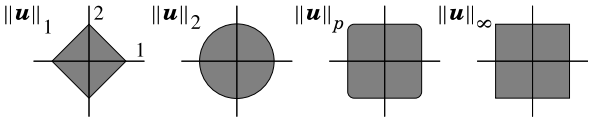
In other words, M is the smallest real number for which $|u(\boldsymbol{x})| \le M$ holds true outside of a set with measure zero.

A.3 Discrete Vector Norms

We are mostly working with n-dimensional vectors $\boldsymbol{u} \in \mathbb{R}^n$ and square matrices $A \in \mathbb{R}^{n\times n}$. Several vector, matrix, and operator norms can be defined (see Sect. A.4). The most important class of vector norms are the p-norms

$$\|\boldsymbol{u}\|_p = \left(\sum_{j=1}^{n} |u_j|^p\right)^{1/p}, \quad 1 \le p < \infty.$$

The unit balls ("circles") with $n = 2$ for the norms $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_p$ for some large p are the first three shapes in this figure:



A commonly used norm is the "max"-norm $\|\boldsymbol{u}\|_\infty = \max_{1\le j\le n} |u_j|$, for which the unit "circle" is shown by the fourth shape. In the numerical methods for partial differential equations, where the space axes are discretized in $N + 1$ points (for example, x_j with indices from $j = 0$ to $j = N$), the "max"-norm appears in two disguises. One of them includes all points, while the other excludes the boundary points:

$$\|\boldsymbol{u}\|_\infty = \max_{0\le j\le N} |u_j|, \qquad \|\boldsymbol{u}\|_{\infty 0} = \max_{1\le j\le N-1} |u_j|. \tag{A.3}$$

Various inequalities exist among the vector norms. They can be used in establishing the hierarchy of the estimates of differences, errors, and remainders in virtually all numerical problems. Among others, the following relations hold:

$$\|\boldsymbol{u}\|_2 \le \|\boldsymbol{u}\|_1 \le \sqrt{n}\|\boldsymbol{u}\|_2, \qquad \|\boldsymbol{u}\|_\infty \le \|\boldsymbol{u}\|_2 \le \sqrt{n}\|\boldsymbol{u}\|_\infty,$$
$$\|\boldsymbol{u}\|_\infty \le \|\boldsymbol{u}\|_1 \le n\|\boldsymbol{u}\|_\infty.$$

Of all p-norms, by far the most widely used is the Euclidean (l_2):

$$\|\boldsymbol{u}\|_2 = \sqrt{\sum_{j=1}^{n} |u_j|^2}. \tag{A.4}$$

(If the subscript 2 is missing, it is usually precisely 2 that is meant.) If we compute the norm by naively coding the expression (A.4) in a computer program, about a half of all representable numbers in floating-point arithmetic will cause an overflow or an underflow. This is how LAPACK's experts circumvent this problem (the DNRM2 routine from the level-1 BLAS library) [1]:

Input: Values $\{u_j\}_{j=1}^n$
$t = 0$; $s = 1$;
for $i = 1$ **step** 1 **to** n **do**
  if $u_j \neq 0$ **then**
    if $|u_j| > t$ **then**
      $s = 1 + s(t/u_j)^2$;
      $t = |u_j|$;
    else
      $s = s + (u_j/t)^2$;
    end
  end
end
Output: $\|\boldsymbol{u}\|_2 = t\sqrt{s}$

In the chapters on differential equations, we sometimes use the so-called "energy" ($l_{2,\Delta x}$) norm, which takes into account the discretization Δx in one of the coordinates,

$$\|\boldsymbol{u}\|_{2,\Delta x} = \sqrt{\sum_{j=1}^{n} |u_j|^2 \Delta x}. \tag{A.5}$$

Namely, by using the basic l_2-norm it is quite awkward to distinguish among different discretization of the function values. Why? An infinitely long vector describing the discrete approximation of a function defined on the whole real axis,

$$\boldsymbol{u}_{\Delta x} = \big(\ldots, u(-\Delta x), u(0), u(\Delta x), \ldots\big)^{\mathrm{T}},$$

has the l_2-norm of $\|\boldsymbol{u}_{\Delta x}\|_2$. The approximation $\boldsymbol{u}_{\Delta x/2}$ on a mesh twice as dense (with a spacing of $\Delta x/2$) has twice as many components as $\boldsymbol{u}_{\Delta x}$, and its l_2-norm is $\|\boldsymbol{u}_{\Delta x/2}\|_2 \approx \sqrt{2}\|\boldsymbol{u}_{\Delta x}\|_2$. For any smooth function therefore $\|\boldsymbol{u}_{\Delta x}\|_2 \to \infty$ if $\Delta x \to 0$. In such cases we prefer to use the $l_{2,\Delta x}$-norm, which possesses all formal properties of the l_2-norm, and which approximates the norm (A.2) for square-integrable functions.

The generalization to vectors from spaces $\mathbb{R}^n \oplus \mathbb{R}^n \oplus \cdots$ is straightforward. For example, for $\boldsymbol{u} \in \mathbb{R}^n \oplus \mathbb{R}^n$ we have

$$\|\boldsymbol{u}\|_{2,\Delta x} = \sqrt{\sum_{j=1}^{n}\sum_{k=1}^{n} |u_{jk}|^2 \Delta x \Delta y}.$$

In infinitely dimensional spaces the sums run from $-\infty$ to $+\infty$. Such examples can be found in discussing initial problems for ordinary and partial differential equations, where $\boldsymbol{u} = (\ldots, u_{-1}, u_0, u_1, \ldots)^{\mathrm{T}}$. In such cases the norm (A.5) is computed as

$$\|\boldsymbol{u}\|_{2,\Delta x} = \left(\sum_{j=-\infty}^{\infty} \|u_j\|_2 \Delta x \right)^{1/2}.$$

A.4 Matrix and Operator Norms

Matrix norms can be introduced by analogy with the vector norms. We are primarily dealing with real square $n \times n$ matrices. A commonly used norm is the "max"-norm

$$\|A\|_{\max} = \max_{ij} |A_{ij}|,$$

but this "norm" does not satisfy one of the mathematical requirements for the matrix norm: namely, the inequality $\|AB\|_{\max} \leq \|A\|_{\max}\|B\|_{\max}$ is not always fulfilled. The Euclidean norm

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_{ij} |A_{ij}|^2},$$

also known as the Frobenius or Hilbert–Schmidt norm, is a "genuine" matrix norm. We also use the norm $\|A\|_1$, which measures the largest column sum of the absolute values of the matrix elements, and the norm $\|A\|_\infty$, which measures the largest row sum of the absolute values:

$$\|A\|_1 = \max_j \sum_i |A_{ij}|, \qquad \|A\|_\infty = \max_i \sum_j |A_{ij}|.$$

Note that $\|A\|_1 = \|A^{\mathrm{T}}\|_\infty$. The operator norms

$$\|A\|_p = \max_{\boldsymbol{u} \neq \boldsymbol{0}} \frac{\|A\boldsymbol{u}\|_p}{\|\boldsymbol{u}\|_p}$$

(which are also matrix norms) are defined according to the underlying vector norm, and are thus also known as the subordinate or induced norms. In general, operator

norms are difficult to compute. The operator norm corresponding to the Euclidean vector norm is

$$\|A\|_2 = \max_{\boldsymbol{u}\neq\boldsymbol{0}} \frac{\|A\boldsymbol{u}\|_2}{\|\boldsymbol{u}\|_2} = \sqrt{\lambda_{\max}},$$

where $\lambda_{\max}$ is the largest eigenvalue of the matrix $A^{\mathrm{T}}A$. Various relations exist among different matrix norms, for example

$$\|A\|_2 \le \|A\|_{\mathrm{F}} \le \sqrt{n}\,\|A\|_2 \le \|A\|_\infty, \qquad \frac{1}{n}\|A\|_\infty \le \|A\|_1 \le n\|A\|_\infty \le n^{3/2}\|A\|_2.$$

The spectral radius of the matrix A with the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_s$ is defined as

$$\rho(A) = \max_{1\le k\le s} \big\{ \big|\lambda_k(A)\big| \big\}.$$

For Hermitian matrices the Euclidean matrix norm is equal to the spectral radius of the matrix, $\rho(A) = \|A\|_2$, while in general $\rho(A) \le \|A\|_2$. These considerations apply equally well to vectors in $\mathbb{C}^n$ and matrices in $\mathbb{C}^{n\times n}$: the absolute value signs denote the magnitudes of the complex numbers, while for matrices A^{T} (transposition) should be substituted by $A^\dagger$ (transposition and complex conjugation).

A.5 Eigenvalues of Tridiagonal Matrices

Here we list the eigenvalues λ_s and eigenvectors $\boldsymbol{v}_s = (v_1, v_2, \ldots, v_k, \ldots, v_N)_s^{\mathrm{T}}$ of some special $N \times N$ matrices appearing in problems with partial differential equations. The eigenvalues of a tridiagonal matrix

$$T(a,b,c) = \begin{pmatrix} b & c & 0 & & \\ a & b & c & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & a & b & c \\ & & 0 & a & b \end{pmatrix} \tag{A.6}$$

are

$$\lambda_s = b + 2c\sqrt{\frac{a}{c}}\cos\frac{s\pi}{N+1}, \quad s = 1, 2, \ldots, N, \tag{A.7}$$

while the corresponding components of the eigenvectors are

$$(\boldsymbol{v}_s)_k = 2\left(\sqrt{\frac{a}{c}}\right)^k \sin\frac{ks\pi}{N+1}, \quad k = 1, 2, \ldots, N. \tag{A.8}$$

On p. 479 we also refer to the tridiagonal symmetric matrix

$$T_{N_1 D} = \begin{pmatrix} 1 & -1 & 0 & & \\ -1 & 2 & -1 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & -1 & 2 & -1 \\ & & 0 & -1 & 2 \end{pmatrix}, \tag{A.9}$$

with eigenvalues

$$\lambda_s = 2 - 2\cos\frac{(2s-1)\pi}{2N+1}, \quad s = 1, 2, \ldots, N,$$

and eigenvector components

$$(\boldsymbol{v}_s)_k = \cos\frac{(2k-1)(2s-1)\pi}{2(2N+1)}, \quad k = 1, 2, \ldots, N.$$

A.6 Singular Values of X and Eigenvalues of $X^{\mathrm{T}}X$ and XX^{T}

The singular values and the corresponding singular vectors of the $n \times m$ matrix X of rank r are closely related to the eigenvalues of the products $X^{\mathrm{T}}X$ and XX^{T}. Within this textbook, this connection is exploited in multivariate statistical methods, where it is sometimes preferable to use the covariance matrix $X^{\mathrm{T}}X$ rather than the data matrix X (Sects. 5.6, 5.7, 5.10, and 5.11). Let

$$X = U\Sigma V^{\mathrm{T}}, \quad U \in \mathbb{R}^{n\times r},\ \Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r) \in \mathbb{R}^{r\times r},\ V \in \mathbb{R}^{m\times r},$$

be the singular decomposition of the matrix X. Then the following holds:

1. The matrix $X^{\mathrm{T}}X \in \mathbb{R}^{m\times m}$ is symmetric and has r positive eigenvalues $\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_r^2\}$, with the corresponding eigenvectors $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r\}$, and $m - r$ zero eigenvalues. The singular values $\{\sigma_1, \sigma_2, \ldots, \sigma_r\}$ of X are therefore just the positive square roots of the positive eigenvalues of $X^{\mathrm{T}}X$, and the columns of V are the corresponding eigenvectors.
2. The matrix $XX^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ is symmetric and has r positive eigenvalues $\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_r^2\}$, with the corresponding eigenvectors $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_r\}$, and $n - r$ zero eigenvalues. The singular values $\{\sigma_1, \sigma_2, \ldots, \sigma_r\}$ of X are therefore just the positive square roots of the positive eigenvalues of XX^{T}, and the columns of U are the corresponding eigenvectors.

This implies that the positive eigenvalues of the matrices $X^{\mathrm{T}}X$ or XX^{T} are equal, and the eigenvectors of these matrix products are related by

$$\boldsymbol{u}_i = \frac{1}{\sigma_i} X\boldsymbol{v}_i, \qquad \boldsymbol{v}_i = \frac{1}{\sigma_i} X^{\mathrm{T}}\boldsymbol{u}_i, \quad i = 1, 2, \ldots, r,$$

or, in matrix form,

$$U = XV\Sigma^{-1}, \qquad V = X^{\mathrm{T}}U\Sigma^{-1}.$$

A.7 The "Square Root" of a Matrix

The "square root" of a symmetric positive semi-definite matrix $A \in \mathbb{R}^{n\times n}$ is computed by first finding its Cholesky decomposition $A = GG^{\mathrm{T}}$, then computing the singular decomposition $G = U\Sigma V^{\mathrm{T}}$ of the matrix G, and finally form $X = U\Sigma U^{\mathrm{T}}$. For the matrix X obtained in this way we have $X^2 = A$, and thus $A^{1/2} = X$ or $A^{-1/2} = X^{-1}$ [2].

References

1. LAPACK, The Linear Algebra PACKage. http://www.netlib.org/lapack
2. G.H. Golub, C.F. Van Loan, *Matrix Computations*, 3rd edn. (Johns Hopkins University Press, Baltimore, 1996)

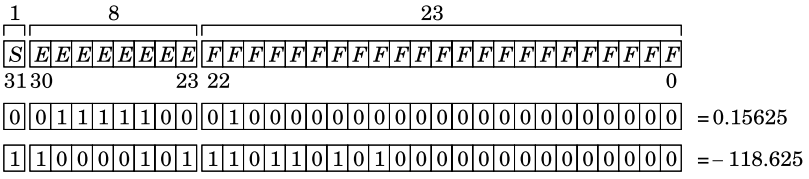
Appendix B
Standard Numerical Data Types

In serious computer programming, memory manipulations are needed (e.g. in dynamic allocation). Moreover, the characteristics of data structures should be well matched to the compiler in order to optimize execution speed (e.g. by aligning the structures to the boundaries of 32-bit regions on 32-bit architectures). To be able to do this, we should know how real and integer numbers are stored.

B.1 Real Numbers in Floating-Point Arithmetic

The IEEE 754-2008 standard for floating-point arithmetic was devised by the IEEE [1]. This set of rules prescribes the computer representation of numbers in single (32 bits) and double precision (64 bits) as well as the rules for arithmetic operations upon these types of numbers (algorithmic prescriptions for addition, subtraction, multiplication, division, and taking the square root). In the following we discuss the numbers in the *little-endian* notation (see Sect. B.2).

Single Precision (C/C++ `float`) The single-precision floating-point representation allows us to represent normal numbers in the range of $\approx 10^{\pm 38}$. According to the IEEE standard, a 32-bit word (4 bytes) is sufficient for the complete representation: it can be written as a sequence of 32 bits from 31 to 0 from left to right. The bit S at the extreme left determines the sign of the whole number (the *sign bit*). It is followed by 8 *exponent bits* E specifying the exponent of the number, and finally by 23 bits representing the *mantissa* or the *fraction* F:



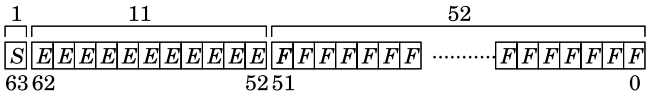
S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9, © Springer-Verlag Berlin Heidelberg 2012

The numerical value of V can be determined by using the scheme

$$\begin{array}{lcl} E = 255 \text{ and } F \neq 0 & \Longrightarrow & V = \text{NaN } (\textit{Not a Number}), \\ E = 255 \text{ and } F = 0 \text{ and } S = 1 & & V = -\infty, \\ E = 255 \text{ and } F = 0 \text{ and } S = 0 & & V = +\infty, \\ 0 < E < 255 & & V = (-1)^S \, 2^{E-127} \, 1.F_{10}, \\ E = 0 \text{ and } F \neq 0 & & V = (-1)^S \, 2^{-126} \, 0.F_{10}, \\ E = 0 \text{ and } F = 0 \text{ and } S = 1 & & V = -0, \\ E = 0 \text{ and } F = 0 \text{ and } S = 0 & & V = +0. \end{array}$$

The values $0.F_{10}$ and $1.F_{10}$ are obtained by taking the negative powers of 2 corresponding to each bit of F to the right of the decimal point, and summing them to form the decimal value, e.g. $1.011_{10} = 1 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 1.875$ (obviously always $0 \leq 0.F_{10} < 1$ and $1 \leq 1.F_{10} < 2$). Two examples are shown in the figure: in the first case $S = 0$, $E = 124$, and $F = 01$ (thus $1.F_{10} = 1.25$) and we obtain $V = (-1)^0 2^{124-127} 1.25 = 0.15625$; in the second case $S = 1$, $E = 133$, and $F = 110110101$ (thus $1.F_{10} = 1.85351562$) and therefore $V = (-1)^1 2^{133-127} 1.85351562 = -118.625$. The zeroth (rightmost) bit is known as the *least significant bit* (LSB), in contrast to the *most significant bit* (MSB) at the extreme left: if the LSB changes, the whole value V changes least. The notation $S = 0$, $E = 0$, and just LSB $= 1$ therefore corresponds to the smallest representable positive number in single-precision, $V = 2^{-126} 2^{-23} = 2^{-149} \approx 1.4 \cdot 10^{-45}$.

Double Precision (C/C++ `double`) The representation of a double-precision floating-point number by the IEEE standard requires a 64-bit word (8 bytes) which can be written as a sequence of 64 bits from 63 to 0 from left to right. This packaging allows us to represent normal numbers in the range of $\approx 10^{\pm 308}$. By analogy to the single-precision representation, the leftmost bit S determines the sign of the whole number, followed by 11 exponent bits and 52 bits for the mantissa:



The numerical value of V is determined as before,

$$\begin{array}{lcl} E = 2047 \text{ and } F \neq 0 & \Longrightarrow & V = \text{NaN } (\textit{Not a Number}), \\ E = 2047 \text{ and } F = 0 \text{ and } S = 1 & & V = -\infty, \\ E = 2047 \text{ and } F = 0 \text{ and } S = 0 & & V = +\infty, \\ 0 < E < 2047 & & V = (-1)^S \, 2^{E-1023} \, 1.F_{10}, \\ E = 0 \text{ and } F \neq 0 & & V = (-1)^S \, 2^{-1022} \, 0.F_{10}, \\ E = 0 \text{ and } F = 0 \text{ and } S = 1 & & V = -0, \\ E = 0 \text{ and } F = 0 \text{ and } S = 0 & & V = +0. \end{array}$$

Quadruple Precision (C/C++ `long double`) In exceptional cases we need quadruple-precision numbers requiring 96 bits of memory (1 sign bit, 31 exponent

bits, and 64 bits for the mantissa). This data type enables us to represent normal numbers in the range of $\approx 10^{\pm 4932}$. If our algorithm does not work in single or double precision, we should not resort blindly to quadruple precision: rather, discover the deficiency of the algorithm or find the error in the code.

Exceptions in Floating-Point Arithmetic A computer code performing operations in floating-point arithmetic may generate certain types of critical errors which should be made known to the user. Locating such errors is made easier by tracing *exceptions*. The IEEE 745 standard defines several critical errors and the corresponding exceptions that these errors should trigger. The most commonly encountered exceptions occurring in computations by a real number R, are

- `FE_DIVBYZERO`: division by zero ($R/0$),
- `FE_INVALID`: invalid conversion of data to R (its floating-point form),
- `FE_OVERFLOW`: $|R|$ exceeds the maximum value (range overflow),
- `FE_UNDERFLOW`: $|R|$ is smaller than the smallest non-zero value (range underflow),
- `FE_INEXACT`: inexact conversion of data to R.

For most errors, triggering an exception is not automatic. In a program written in C/C++, exceptions must be enabled explicitly: for example, the exceptions `FE_DIVBYZERO`, `FE_INVALID`, or `FE_OVERFLOW`) are activated by the command

```
#include <fenv.h>
feenableexcept(FE_DIVBYZERO | FE_INVALID | FE_OVERFLOW);
```

and are deactivated by the command `fedisableexcept()`. If an exception is triggered, the operating system terminates the program with the SIGFPE signal (*floating-point exception*).

Normal and Subnormal Numbers and Their Limits The numbers V corresponding to "non-extreme" E (in the ranges $0 < E < 255$ or $0 < E < 2047$) are known as *normal* or *normalized*, since their mantissa has the form $1.F_{10}$. In the case $E = 0$ and $F \neq 0$ (mantissa in the form $0.F_{10}$) we are dealing with *subnormal* or *denormalized* numbers. Both types of numbers appear in a variety of applications. A simple program (listed below in C++) may be used to establish their upper and lower limits:

```
using namespace std;
#include <stdlib.h>
#include <iostream>
#include <limits>
#include <sstream>

template <typename T> string S(string name) {
  ostringstream os;
  os << "Data type: " << name << endl
     << " mantissa: " << numeric_limits<T>::digits << endl
     << "Normal: " << endl
```

```
      << "   min: " << numeric_limits<T>::min() << endl
      << "   max: " << numeric_limits<T>::max() << endl
      << "   exponent in ["
      <<                 numeric_limits<T>::min_exponent << ","
      <<                 numeric_limits<T>::max_exponent << "]"
      << endl
      << "Subnormal: \n"
      << "   min: " <<  numeric_limits<T>::denorm_min() << endl;
  return os.str();
}

int main() {
  cout << S<float>("float") << endl
       << S<double>("double") << endl
       << S<long double>("long double") << endl;
  return EXIT_SUCCESS;
}
```

B.1.1 Combining Types with Different Precisions

On all computer architectures, computation with more precise data types is more time- and memory-consuming than computing with less precise ones. By knowing their inherent limitations, however, various data types can be sometimes combined such that the precision of the final result is dictated by the more precise type, yet the computation proceeds relatively faster.

Let us discuss such a case in which `float` and `double` data types are combined. Suppose we have a real number x of type `double` and we wish to compute the value of a function f at x. We represent x as a sum of the nearest number s of type `float` and the remainder ε of type `double`:

$$x(\text{double}) = s(\text{float}) + \varepsilon(\text{double}).$$

In double-precision floating-point arithmetic we have $1 \oplus \varepsilon^3 = 1$, so that certain functions can be evaluated faster and sometimes even more precisely. In the case of the exponential function, $f(x) = \exp(x) = \exp(s)\exp(\varepsilon)$, we can use the power expansion $\exp(\varepsilon) \approx 1 + \varepsilon + \varepsilon^2/2$, since ε is small and the third order of the expansion does not contribute anything at the specified precision. In the case of $f(x) = \exp(x^2)$, we use the expansion $x^2 = s^2 + \varepsilon(x + s)$. Similar conclusions follow in the evaluation of polynomials of low degrees or smooth functions f that can be approximated by $f(x) \approx f(s) + f'(s)\varepsilon + f''(s)\varepsilon^2/2$, if one can ascertain that $f(s)$ is computed to sufficient precision.

The speed of the operations with different numerical types strongly depends on the computer architecture and the programming language. In the order of increasing numerical cost, addition is followed by subtraction (about the same cost as addition), multiplication, and division (about the same cost as multiplication). The speed of the algorithm is determined mostly by the number of multiplications and divisions.

Multiplication and division of `double` types are about twice as time-consuming as multiplication and division of `float` types.

B.2 Integer Numbers

The representation of integer numbers is more transparent than the representation of floating-point numbers. The most frequently used types in programming languages related to C/C++ are `char` (8 bits in memory), `short int` (at least 16 bits), and `int`, where "int" is an abbreviation for "integer". According to the C language standard, the type `int` has the same size as a typical processor register and hence occupies 32 (64, 128, ...) bits on 32 (64, 128, ...) bit architectures. The binary representation of an integer with p bits $b_i \in \{0, 1\}$ further depends on an additional declaration qualifier. If the variable is `unsigned`, it describes a non-negative number of the form

$$b_p 2^p + b_{p-1} 2^{p-1} + \cdots + b_1 2^1 + b_0 2^0 = (b_p, b_{p-1}, \ldots, b_0)_2,$$

which is represented by the vector of bits $(b_p, b_{p-1}, \ldots, b_0)$. If it is declared as `signed`, it can stand for either positive or negative numbers of the form

$$\pm(b_{p-1}, b_{p-2}, \ldots, b_0)_2.$$

A positive integer can be represented by the vector of bits $(0, b_{p-1}, \ldots, b_0)$, i.e. by a non-negative number $x = (0, b_{p-1}, \ldots, b_0)_2$, while a negative integer can be represented by $(1, \neg b_{p-1}, \ldots, \neg b_0)_2 + 1$, also known as the *two's binary complement* of x. The leftmost bit (MSB) is the *sign bit*. This rather unusual notation for negative integers allows for a faster summation of positive and negative numbers. The 32-bit computer architecture permits the following integer ranges:

	`unsigned`	`signed`
`char`	0...255	−128...127
`short int`	0...65535	−32768...32767
`int`	0...4294967295	−2147483648...2147483647

Two further types, `long int` and `long long int`, are also in use. Their implementation depends on the compiler, computer architecture, and even the operating system: on UNIX and Linux systems, see system variable `__WORDSIZE`) and other declarations in the file `/usr/include/limits.h`.

An even more important characteristic of the representation of integers is the ordering of bits within a byte or word. In *big-endian* ordering the most significant bit (MSB) is stored at the lowest memory address. In *little-endian* ordering the least significant bit (LSB) is stored first. The distinction between *big-endian* and *little-endian* may apply even at the level of words, not just at the level of bits: the number 1025 in integer type of length 4 bytes can be represented as 00000000|00000000|00000100|00000001, but this sequence of bits will be stored differently in the two orderings:

```
address    big-endian     little-endian
00         00000000       00000001
01         00000000       00000100
10         00000100       00000000
11         00000001       00000000
```

Most modern computers use *little-endian* ordering, while *big-endian* ordering is popular on (super)computers. The PowerPC architecture is *bi-endian* as it understands both systems. Problems may appear when one attempts to port the programs between architectures with different orderings. This is known as the "NUXI problem": the word "UNIX", written in two two-byte words, is stored in memory as "UNIX" on *big-endian* systems (in the sense of ordering bytes within a word), or as "NUXI" on *little-endian* systems. Note that within an individual byte, even the bits can be stored in one ordering or another. Moreover, on some systems the order of significance of bits in a byte can be opposite to the order of significance of bytes in a word!

Until we live in the safety of integer arithmetic at the level of a programming language, we need not fear these peculiarities. But when we directly manipulate bits, e.g. in cyclic permutations of bits or executing logical operations between sequences of bits, a detailed understanding of the "NUXI problem" is essential. The following program allows us to determine the way in which integer and floating-point numbers are stored on our computer:

```
#include <stdlib.h>
#include <iostream>
using namespace std;

template <typename T> string binary(const T v) {
  typedef unsigned char Ti;
  string s;
  for (Ti *d = (Ti *)&v; d != sizeof(v) + (Ti *)&v; ++d)
    for (Ti mask = 1; mask; mask <<= 1)
      if ((*d) & mask) s += '1'; else s += '0';
  return s;
}

int main() {
  int i = 1025;
  double d = 1025;
  cout << "int    " << i << ", binary " << binary(i) << endl
       << "double " << d << ", binary " << binary(d) << endl;
  return EXIT_SUCCESS;
}
```

B.3 (Almost) Arbitrary Precision

Computations in number theory and numerous physical application sometimes require arbitrary precision in integer or floating-point arithmetic. Standard libraries

exist for this purpose. Important libraries for integer arithmetic are BigDigits [2], where the interval $[L_{\min}, L_{\max}]$ of integers is practically unbounded. Of course, with increasing interval length $L = L_{\max} - L_{\min} + 1$, memory consumption also increases, and amounts to $\lceil \log_2 L \rceil$ bits in the optimal case. In order to ensure good portability, well-defined use of memory, and transparency of computations, we sometimes also use the character representation of numbers. In this representation, numbers in some base B (e.g. decimal, $B = 10$) are stored as arrays of characters, e.g. the number 123 is stored as the array "123". Such storage requires $8\lceil \log_B L \rceil$ bits (which is a lot). Another important library for integer arithmetic in arbitrary precision is NTL [3].

Arbitrary integer and floating-point arithmetic are implemented in the GMP library (Gnu Multi-Precision, "*The fastest bignum library on the planet*") [4, 5]. Implementations in various precisions (`double-double precision`, `quad-double precision`, and arbitrary precision) for Fortran77, Fortran90, and C++ are listed in [6].

References

1. IEEE Standard 754-2008 for Binary Floating-Point Arithmetic (IEEE, 2008); see also http://grouper.ieee.org/groups/754/
2. BigDigits multiple-precision arithmetic. http://www.di-mgt.com.au/bigdigits.html
3. NTL: a library for doing number theory. http://shoup.net/ntl
4. GNU multi precision (GMP), free library for arbitrary precision arithmetic. http://gmplib.org
5. L. Fosse et al., MPFR: a multiple-precision binary floating-point library with correct rounding. ACM Trans. Math. Softw. **33**, 13 (2007)
6. http://crd.lbl.gov/~dhbailey/mpdist

Appendix C
Generation of Pseudorandom Numbers

Generation of *pseudorandom* (or *quasirandom*) numbers is a deterministic process of computing sequences appearing to be random. The degree of their randomness is established by special statistical tests. *Random number generators* (RNG) are used in a variety of computations, and each individual application may require these generators to possess specific statistical properties. In the following, generation of pseudorandom numbers is called simply *drawing*, and the generated numbers are called *random numbers*. The mathematical basis of random numbers is presented in [1], the analysis of generators from the mathematical perspective is given by [2, 3], and from the viewpoint of theoretical computing by [4]. Details on the use of generators can be found in [5, 6, 7].

C.1 Uniform Generators: From Integers to Reals

To generate random numbers with a uniform probability distribution, we use *uniform generators*. A discrete integer random variable $X \in \mathbb{Z}_m = [0, m-1] \subset \mathbb{N}_0$ is distributed uniformly if it assumes any value on the interval with equal probability $\mathcal{P}$, thus

$$\mathcal{P}(X = i) = \frac{1}{m}, \quad i = 0, 1, \ldots, m-1.$$

A continuous real random variable $X \in [a, b] \subset \mathbb{R}$ is distributed uniformly if its probability density is

$$p(x) = \begin{cases} (b-a)^{-1}; & x \in [a, b], \\ 0; & \text{otherwise.} \end{cases}$$

We denote the uniform distribution on the real or integer axis by $U(a, b)$.

A good uniform generator that yields the sequence of numbers $\{x_i\}$ is expected to generate *uncorrelated sequences*: this means that the vectors of subsequences $(x_i, x_{i+1}, \ldots, x_{k+i})$ are as weakly correlated as possible, for any k. The sequence

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9, © Springer-Verlag Berlin Heidelberg 2012

$\{x_i\}$ should also have a *long period* (it should not repeat itself for as long as possible) and should be *uniform and unbiased* (equal number of points should fall into spaces of equal sizes). In other words, we require a uniform distribution of points $v_i^{(k)} = (x_i, x_{i+1}, \ldots, x_{k+i-1})$ in a hypercube with dimension k as large as possible; this is known as *serial uniformity of the sequence*. Moreover, the generator should be numerically efficient [8].

Most modern uniform random generators are based on integer arithmetic. Such generators typically return numbers with equal probabilities on the interval $[0, m-1]$, where $m = 2^{32}$ or 2^{64}. Uniform generators are standard components of general libraries and tools, e.g. the command `rand()` in MATLAB and C/C++, `gsl_rng_rand` in GSL, or `Random[]` in MATHEMATICA. Random integers $x_k \in \mathbb{Z}_m$ returned by the generator can be converted to uniformly distributed random real numbers y_k from $U(0, 1)$ by using the transformations [2, 3]:

$$\begin{array}{lll} y_k = x_k/m & \text{approximately uniform in} & [0, 1), \\ y_k = x_k/(m-1) & & [0, 1], \\ y_k = (x_k+1)/m & & (0, 1], \\ y_k = (x_k+1/2)/m & & (0, 1). \end{array}$$

The real numbers obtained in this manner have $b = \log_2 m$ random most significant bits. Frequently this is not enough (and, at any rate, less than what is supported by the mantissa of the real data type). But there is another way of converting x_k to y_k. In real arithmetic with a n-bit mantissa, the precision of a number is 2^{-n} with $n > b$. An approximation of a random real number y on the interval $[0, 1)$ with all bits random can be obtained by independently drawing the integers $\{x_k \in \mathbb{Z}_m\}_{k=1}^{h}$ and using the formula

$$y = x_1 m^{-1} + x_2 m^{-2} + \cdots + x_h m^{-h},$$

where h is chosen such that $(h-1)b < n < (h+1)b$. In the case of the 32-bit integer random generator `int32()` and the real data type `double`, the right-hand side of this formula is (see Chap. 7 in [5]):

```
2.3283064365386962E-10 * (int32() + 2.3283064365386962E-10
                          * int32())
```

C.2 Transformations Between Distributions

A generator of random numbers with an arbitrary discrete or continuous distribution can be obtained by transforming the numbers returned by a uniform generator. Here we present the most frequently encountered transformations.

C.2.1 Discrete Distribution

Let X be a random variable that can assume n distinct values. An example is the interval of integers $I = [1, n]$, where X assumes each value with a probability $p_i = \mathcal{P}(X = i)$ for $i \in I$ and $\sum_{i \in I} p_i = 1$ holds true. The probabilities p_i represent a discrete probability distribution with the cumulative distribution

$$P_i = \mathcal{P}(X \leq i) = \sum_{j \leq i} p_j \quad \text{for } i \in I \cup \{0\}.$$

A discrete random variable X can be expressed by a continuous random variable U from $U(0, 1)$ by inverting the cumulative distribution, by using the formula $X = P^{-1}(U)$: this means that $X = i$ precisely when $P_{i-1} \leq U < P_i$.

This statement can be turned into a method of generating numbers, where for each drawn number u from $U(0, 1)$ we find an interval $R_i = [P_i, P_{i+1})$ such that $u \in R_i$, and then i is the sought value of the random variable X. Finding the corresponding interval can be accomplished linearly in $\mathcal{O}(n)$ operations at worst, or in $\mathcal{O}(\log n)$ by using search trees [9].

In [7] and [10] other methods of generation of random numbers according to a discrete distribution are presented. Here we mention the Walker alias method [11] with the computational cost of $\mathcal{O}(1)$ (independent of n) and memory requirement of just $\mathcal{O}(n)$ [12]. In this method, any discrete random variable $X \in [1, n]$ with the probability distribution $\{p_i\}_{i=1}^n$ can be expressed as a random variable $Y \in [1, n]$ by n two-point probability densities

$$\mathcal{P}(Y = y_{ij}) = q_{ij}, \quad i = 1, 2, \ldots, n, \ j = 1, 2,$$

where $q_{i1} + q_{i2} = 1$, and we denote

$$y_{i1} = i, \qquad y_{i2} = L_i, \qquad q_{i1} = F_i, \qquad q_{i2} = 1 - F_i.$$

This transformation of random variables is not unique. We draw the value of the random variable X by using the algorithm

Input: Constants F_i and L_i for $i = 1, 2, \ldots, n$.
Draw u from $U(0, n)$;
$i = \lceil u \rceil$;
$v = i - u$;
if $(v > F_i)$ **then**
  $i = L_i$;
Output: i is one realization of the discrete random variable X with the distribution p_i.

The constants F_i and L_i depend on the distribution p_i and are determined below. The constant F_i represents the limit value to which we compare the randomly chosen value u from $U(0, 1)$, while L_i is the alias into which i is transformed if the

step	L	F	G	S
1	(0, 0, 0, 0)	(1.6, 1.2, 0.8, 0.4)	{1, 2}	{3, 4}
2	(0, 0, 1, 0)	(1.4, 1.2, 0.8, 0.4)	{1, 2}	{4}
3	(0, 0, 1, 1)	(0.8, 1.2, 0.8, 0.4)	{2}	{1}
4	(2, 0, 1, 1)	(0.8, 1.0, 0.8, 0.4)	{2}	{}

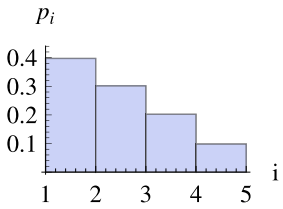


Fig. C.1 The sequence of vectors of constants L and F, and the sets G and S during the Walker algorithm for the discrete distribution $p = (p_i)_{i=1}^4 = (0.4, 0.3, 0.2, 0.1)$ of numbers $\{1, 2, 3, 4\}$. Step 1 represents the status of (L, F, G, S) prior to entering the loop

comparison fails. The F_i and L_i can be determined by using the procedure

Input: Discrete distribution given by vector $p = (p_i)_{i=1}^n$.
Define vectors $F = (F_i)_{i=1}^n$ and $L = (L_i)_{i=1}^n$.
for $i = 1, 2, \ldots, n$ **do**
  $F_i = np_i$;
  $L_i = 0$;
Define sets $G = \{i : F_i > 1\}$ and $S = \{i : F_i < 1\}$.
if ($|S| = 0$) **then**
  All p_i are equal to $1/n$, so terminate the routine, since the values of the random variable X can be drawn from the uniform distribution.
repeat
  Choose an element from G (denoted by k) and an element from S (denoted by j).
  $L_j = k$; `// alias assigned to element` j
  $S = S \backslash \{j\}$;
  $F_k = F_k - (1 - F_j)$; `// change` k`th limit value`
  if ($F_k < 1$) **then**
    $G = G \backslash \{k\}$;
    $S = S \cup \{k\}$;
until ($|S| > 0$);
Output: Vectors of limit values F and aliases L for distribution p.

Here $|\cdot|$ denotes cardinality (the number of elements in the set). An illustration of the Walker algorithm in operation is given in Fig. C.1. Walker's method is implemented in the GSL library (commands `gsl_ran_discrete*`) and the R project (commands `sample`[·]).

C.2.2 Continuous Distribution

If a real random variable X is distributed according to the continuous probability density p, the probability of an event $X \in A = [a, b]$ is given by the integral

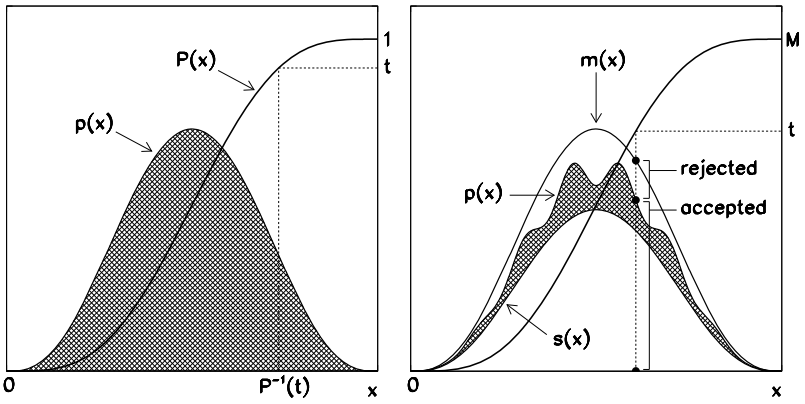


Fig. C.2 Drawing [*Left*] by the inverse method; [*Right*] by the rejection method

$$\mathcal{P}(X \in A) = \int_A p(t)\,\mathrm{d}t,$$

The function p is integrable and usually at least piecewise continuous. The cumulative distribution function is

$$P(x) = \int_{-\infty}^{x} p(t)\,\mathrm{d}t, \quad 0 \le P(x) \le 1.$$

Random values of the variable X can be obtained by using uniform generators and transformations between distributions of various variables. In the following we present the most commonly used methods.

The Inverse Method Assume that X is a random variable with the probability density $p(x)$ and U is a variable distributed uniformly according to $U(0, 1)$. Then X can be expressed as

$$X = P^{-1}(U)$$

(Fig. C.2 (left)). This method of generating the values of X is useful if the inverse of the cumulative distribution function $P^{-1}(y)$ is easy to compute.

If a random variable $X \in \Omega$ has the probability density p_X and h is a differentiable function with the inverse $g = h^{-1}$, the random variable $Y = h(X)$ has the probability density

$$p_Y(y) = \int_\Omega \delta\big(h(t) - y\big) p_X(t)\,\mathrm{d}t = p_X\big(g(y)\big)\big|g'(y)\big|.$$

If independent random variables X and Y are distributed according to the densities p_X and p_Y, the probability density of the sum $Z = X + Y$ is given by the

convolution of p_X and p_Y,

$$p_Z(z) = \int_{\mathbb{R}} p_X(t-z)\, p_Y(t)\, \mathrm{d}t.$$

A few examples follow.

• A variable X distributed according to the Weibull probability density

$$p(t) = kt^{k-1} \exp\bigl(-t^k\bigr), \quad t > 0,$$

can be obtained by the transformation $X = (-\log U)^{1/k}$. We except the value 0 from the domain of U, so that X is always bounded. A special case of the Weibull density is the exponential density $p(t) = \exp(-t)$.

• If U_1 and U_2 are independent random variables distributed according to $U(0,1)$, the Box–Muller transformation [13]

$$X_1 = \sqrt{-2\log U_1}\, \cos(2\pi U_2), \qquad X_2 = \sqrt{-2\log U_1}\, \sin(2\pi U_2),$$

yields independent random variables X_1 and X_2 distributed according to the normal distribution $N(0,1)$. The variables U_1 and U_2 define the length $R = \sqrt{-2\log U_1}$ and angle $\theta = 2\pi U_2$ of the two-dimensional vector $(X_1, X_2)^{\mathrm{T}}$. The costly computation of trigonometric functions can be avoided in the Marsaglia implementation of the Box–Muller method (see [4], Chap. 7, Algorithm P):

repeat
  Independently draw u_1 and u_2 from $U(0,1)$.
  $\boldsymbol{v} = 2(u_1, u_2)^{\mathrm{T}} - (1,1)^{\mathrm{T}}$;
  $s = |\boldsymbol{v}|^2$;
until $(s \geq 1 \vee s \neq 0)$;
$(x_1, x_2)^{\mathrm{T}} = \sqrt{-2\log(s)/s}\, \boldsymbol{v}$;
Output: Realization of two independent random variables (x_1, x_2) distributed according to $N(0,1)$.

On the average, the drawn vector $\boldsymbol{v}$ covers the unit circle, while approximately $1 - \pi/4 \approx 21.5$ % of generated points are discarded: for one pair (x_1, x_2) we draw $2/(\pi/4) \approx 2.54$ uniformly distributed numbers. This algorithm is suitable for the generation of complex Gaussian-distributed numbers $x_1 + \mathrm{i}\, x_2$ for the construction of random matrices from the GUE (Sect. 3.5.2).

• The values $\boldsymbol{x}$ of a random vector $\boldsymbol{X} \in \mathbb{R}^d$ distributed according to the multivariate normal probability density

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2} (\det \Sigma)^{1/2}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}} \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}} \right)$$

with the mean $\boldsymbol{\mu}$ and correlation matrix Σ can be generated by independently drawing d components of the vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_d)^{\mathrm{T}}$ according to the standard normal distribution $N(0, 1)$, and computing

$$\boldsymbol{x} = L\boldsymbol{y} + \boldsymbol{\mu},$$

where L is the lower-triangular $d \times d$ matrix from the Cholesky decomposition of the correlation matrix, $\Sigma = LL^{\mathrm{T}}$.

• The values of a random variable X with the probability density in the form of a symmetric trapezoid

$$p(x) = \frac{1}{ab} \begin{cases} (a+b)/2 - |t|; & 2|t| \in [|a-b|, (a+b)], \\ \min(a, b); & 2|t| \in [0, |a-b|], \\ 0; & \text{otherwise}, \end{cases}$$

can be obtained by combining the values of two random variables U_1 and U_2 uniformly distributed on the unit interval [0, 1], by using the formula

$$X = a\left(U_1 - \frac{1}{2}\right) + b\left(U_2 - \frac{1}{2}\right).$$

• The points $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^{\mathrm{T}} \in \mathbb{R}^d$ that are uniformly distributed over the $(d-1)$-dimensional real sphere $\boldsymbol{S}_{d-1} \in \mathbb{R}^d$ can be generated [4] by independently drawing the components of the vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_d)^{\mathrm{T}}$ with probability density $N(0, 1)$, and normalizing it: $x_i = y_i / \|\boldsymbol{y}\|_2$, where $\|\boldsymbol{y}\|_2^2 = \sum_{i=1}^d y_i^2$.

• The points $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^{\mathrm{T}}$, $x_i > 0$, which are uniformly distributed over the plane defined by $\sum_{i=1}^d a_i x_i = b$ with positive real constants a_i and b, are generated by independently drawing d components of the vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_d)^{\mathrm{T}}$ with exponential probability density $p(y) = \exp(-y)$, and computing [14]

$$S = \sum_{i=1}^{d} a_i y_i, \qquad x_i = \frac{b}{S} a_i y_i.$$

Rejection Method Suppose a function m exists that bounds the probability density function p from above as tightly as possible (Fig. C.2 (right)), and that the integral of m is $M = \int_{\mathbb{R}} m(x)\,\mathrm{d}x$. If we possess an efficient method to generate random numbers distributed according to $\frac{1}{M} m$ and the computation of p is too costly, we introduce a function $s \geq 0$ that bounds p as tightly as possible from below, and follow the algorithm

Input: Probability density p, functions m and s.
repeat
  Draw x with probability density $\frac{1}{M}m(x)$ and u with density $U(0,1)$.
  if $(\ s(x) > u\,m(x)\ )$ **then**
    Terminate the loop.
until $(\ p(x) < u\,m(x)\ )$
Output: x is the value of one realization of the random variable X.

The function s increases the rate at which the values are accepted, which is known as *squeezing*. If we do not know the function s, we simply remove the **if** conditional statement from the loop. We discard $\approx 1 - \frac{1}{M}$ drawn pairs (x, v), so the algorithm becomes the most efficient when M is close to 1. If m is a piecewise constant function, the Walker method described previously can be used to draw the values according to $\frac{1}{M}m$. Some examples follow.

• The Cauchy–Lorentz (Breit–Wigner) distribution

$$p(x) = \frac{1}{\pi(1+x^2)} \tag{C.1}$$

is a frequent occurrence: it describes shapes of spectral lines and nuclear resonances in quantum physics, or resonance curves in classical forced oscillations. The corresponding cumulative distribution function P and its inverse P^{-1} are

$$P(x) = \frac{1}{\pi}\arctan x + \frac{1}{2}, \qquad P^{-1}(t) = \tan\left[\pi\left(t - \frac{1}{2}\right)\right]. \tag{C.2}$$

The values of the variable X distributed according to (C.1) could be generated by the inverse method where in (C.2) one would use a random variable U distributed uniformly on $[-\pi/2, \pi/2]$, and compute $X = \tan U$. Instead of the costly evaluation of $\tan U = \sin U / \cos U$, we prefer to see the value of X as the ratio of the projections of a point within a circle onto the x- and y-axis. These points are uniformly distributed over the angles. We use the algorithm

repeat
  Draw u_1 from $U(-1,1)$ and u_2 from $U(0,1)$.
until $(\ u_1^2 + u_2^2 > 1 \vee u_2 = 0\ )$;
$x = u_1/u_2$;
Output: x is a realization of a random variable distributed according to Cauchy.

• Assume that a random variable X has the probability density p defined on the interval $[s, s+h]$ such that

$$\int_s^{s+h} p(x)\,\mathrm{d}x = \int_0^h p(x+s)\,\mathrm{d}x = 1,$$

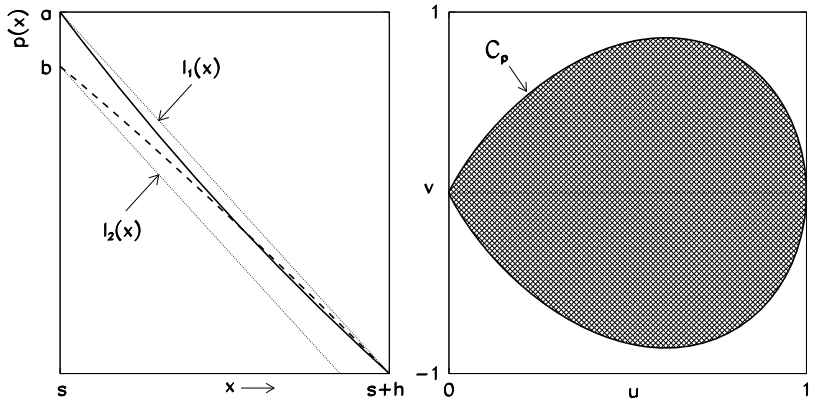


Fig. C.3 [*Left*] An almost linear portion of the probability density $p(x)$ on the interval $[s, s+h]$, which is bounded from above and below by straight lines (C.3). [*Right*] The region C_p where points are accepted in the method of uniform deviates, for the standard normal distribution $p(x) = (2\pi)^{-1/2} \exp(-x^2/2)$. The boundary of the region is defined by $v = \pm 2u\sqrt{-\log u}$

and that it is bounded from above and below by the straight lines

$$l_1(x) = a - \frac{b}{h}(x - s), \qquad l_2(x) = b - \frac{b}{h}(x - s), \tag{C.3}$$

as shown in Fig. C.3 (left). The values of the variable X can be drawn by using the following algorithm [4] based on the rejection method:

Input: Distribution p, interval $[s, s+h]$, constants a and b.
repeat
  Independently draw u and v from $U(0, 1)$.
  if *(* $u > v$ *)* **then**
    Swap u and v. `// implies` $u \le v$.
  $x = s + hu$;
  if *(* $v \le a/b$ *)* **then**
    Terminate the loop.
until *(* $v > u + p(x)/b$ *)*;
Output: $x \in [s, s+h]$ is a realization of a random variable distributed according to p.

Method of the Ratio of Uniform Deviates Assume that the random variable X has the probability density p, to which we assign the region

$$C_p = \left\{(u, v) : 0 \le u \le \sqrt{p(v/u)}\right\}.$$

If (U, V) is a random variable uniformly distributed over C_p, we have $X = V/U$ [15]. This transformation is embodied in the algorithm

Input: Distribution p, constants $a = \sup_x \sqrt{p(x)}$, $b = \inf_x x\sqrt{p(x)}$, and $c = \sup_x x\sqrt{p(x)}$.
repeat
  Independently draw u and v from $U(0, 1)$.
  $u_1 = au$;
  $u_2 = b + (c - b)v$;
  $x = v/u$;
until $(u^2 \leq p(x))$;
Output: x is the value of one realization of the random variable distributed according to p.

This algorithm is a variation of the rejection method, and can be optimized similarly. An example of optimized generation of random numbers from $N(0, 1)$ is described in [16]: the region C_p where the points are accepted is shown in Fig. C.3 (right). This method requires $\approx$2.74 draws from $U(0, 1)$ to generate one number from $N(0, 1)$ [5], which is worse than in the Box–Muller method.

C.3 Random Number Generators and Tests of Their Reliability

Generators of random numbers $x_i \in \mathbb{Z}_m = [0, m - 1]$ where $i \in \mathbb{N}_0 = \{0, 1, \ldots\}$ [4, 5] are defined by the *transition function* F and the relation

$$x_i = F(x_{i-1}, \ldots, x_{i-k}) \mod m.$$

The function F is thus restricted to $\mathbb{Z}_m$ by the congruence relation [2, 3]. The initial state $\{x_0, x_1, \ldots, x_{k-1}\}$ of the generator is a unique function of a number known as the *seed* which completely determines the generated sequence: a generator initialized with the same seed always yields the same sequence of numbers. The properties of F define two classes of generators. If F is linear in its parameters, the generator is *linear*; in other cases, it is *non-linear*.

Random number generators are implemented in all major numerical packages (MATLAB, MATHEMATICA, MAPLE, the R project) and in libraries (NUMERICAL RECIPES, GSL, BOOST). Interesting thoughts on the implementations of generators are preserved in [17] and in Appendices A–C of [18]. A pedagogically systematic overview of random generator classes is offered by [7] and [2, 3].

C.3.1 Linear Generators

Typical linear generators are the *linear congruential generators* (LCG):

$$x_{n+1} = ax_n + c \mod m,$$

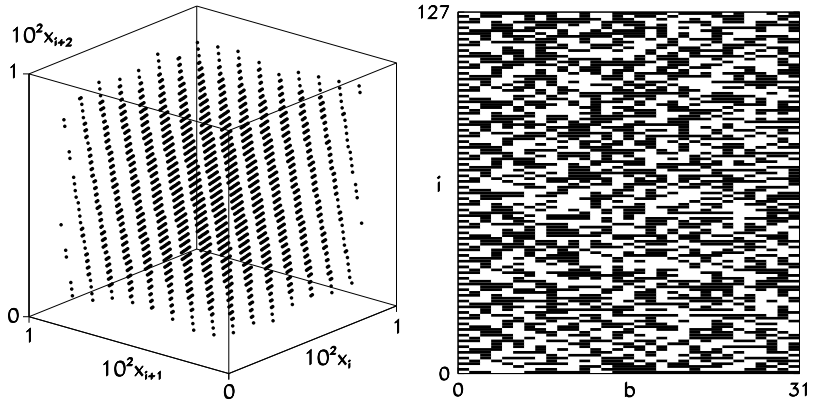


Fig. C.4 [*Left*] Zoom-in of the phase space $[0, 1]^3$ of the points $2^{-31}(x_i, x_{i+1}, x_{i+2})$ picked from the sequence $\{x_i\}$ generated by the standard 32-bit `glibc` generator with $x_0 = 12345$. [*Right*] The bits b of numbers x_i (*black* $= 1$, *white* $= 0$)

where $x_n \in \mathbb{Z}_m$. The result $\{x_0, x_1, \ldots\}$ is called the Lehmer sequence. The *multiplier* a and the *carry* $c \in \mathbb{Z}_m$ are adjustable constants, while the initial value x_0 is the seed. A LCG generator for $c \neq 0$ attains full periods of length m precisely when c and m have no common factors except 1, when $(a - 1)$ is divisible by all prime factors m, and when $(a - 1)$ is a multiple of 4, if m is a multiple of 4 [4]. In the case $c = 0$ we attain the longest period of length $m - 1$ if m is prime. On the average, the period of the LCG-type generator is increased by using a non-zero c.

If $c = 0$, the points $v_i^{(k)} = \frac{1}{m}\{x_i, \ldots, x_{i+k-1}\}$ for given k and x_0 do not fill the whole k-dimensional hypercube but lie on at most $(mk!)^{1/k}$ hyperplanes (similarly for $c \neq 0$ [19]). A good generator should generate numbers over many hyperplanes [20]. It also turns out that the least significant bits are less random than the more significant ones [5, 21]: Fig. C.4 shows the numbers generated by the default generator in the 32-bit `glibc` library. It belongs to the LCG family with the parameters $m = 2^{32}$, $a = 1103515245$, and $c = 123454$. The figure clearly shows that the points are distributed in planes and that the less significant bits are not random.

The LCG generators are therefore not suitable for certain applications. In those cases where the deficiencies discussed above are irrelevant, we nevertheless use them extensively, since they are supported by all programming languages and because they are simple and fast.

Further members of the LCG family are the generators Add-with-Carry (AWC), Subtract-with-Borrow (SWB), and Multiply-with-Carry (MWC) [22–24]:

$$\begin{array}{lll} \text{AWC:} & x_n = x_{n-r} + x_{n-k} + c_{n-1} \mod m, & c_n = \lfloor (x_{n-r} + x_{n-k} + c_{n-1})/m \rfloor, \\ \text{SWB:} & x_n = x_{n-r} - x_{n-k} - c_{n-1} \mod m, & c_n = \lfloor (x_{n-r} - x_{n-k} - c_{n-1})/m \rfloor, \\ \text{MWC:} & x_n = a x_{n-r} + c_{n-1} \mod m, & c_n = \lfloor (a x_{n-r} + c_{n-1})/m \rfloor. \end{array}$$

The SWB algorithm is at the heart of the `RANLUX` generator [25,26] implemented in the GSL library. Almost just as popular are the *multiple recursive generators*

(MRG):

$$x_n = a_1 x_{n-1} + \cdots + a_k x_{n-k} + c_n \mod m,$$

with constants $a_k \in \mathbb{Z}_m$. The carry c_n may or may not depend on the step, or may be even zero, all of which strongly influences the properties of the generator. This family consists of the well-known *lagged Fibonacci generators* (LFG) [4] of the form

$$x_n = x_{n-r} \circ x_{n-k} \mod m,$$

where $\circ$ denotes the operations of addition, subtraction, multiplication, or the exclusive OR (XOR) within $\mathbb{Z}_m$. A typical representative is the popular `ran3` generator from the NUMERICAL RECIPES library (up to its third edition).

C.3.2 Non-linear Generators

In general, non-linear generators are less predictable (more random) than linear, but they are also slower. In specific application, e.g. in Monte-Carlo integrations, the generator speed may be the decisive factor. The main representatives of non-linear generators are the *inversive congruential generators* (ICG) defined by the recurrence

$$x_n = a\overline{x}_{n-1} + b \qquad \mod m,$$

where $(x\overline{x} = 1 \mod m)$, and the *explicit inversive congruential generators* (EICG) [27] based on the relation

$$x_n = \overline{a\, n + b} \mod m.$$

For prime m, the generators from the ICG and EICG families generate points that avoid clustering in planes typical of the LCG generators. However, modular inversion is time consuming, and the filling of space is less uniform [7, 28]. Further members of the non-linear group are the LFSR, NLFSR, and GFSR generators (*linear*, *non-linear*, and *generalized feedback shift register*). A typical LFSR generator is XorShift [29–31]. Details can be found in [2, 3, 5, 7, 32].

The Authors' preference is the *Mersenne twister* [33] (algorithm MT19937) from the family of "twisted" GFSR algorithms. It is implemented in 32-bit integer arithmetic, theoretically well understood, and accessible in standard libraries. Its period is $2^{19937} - 1$ and has been proven to be serially uniform for dimensions $k \in [1, 623]$. Its main deficiency is the somewhat lower randomness of the consecutive bits between subsequently generated numbers.

C.3.3 Using and Testing Generators

There are many random number generators branded as "good" or "best" by specialists. But every generator has its own deficiencies, some of them very specific. If we

(as non-specialists) must invoke a generator very frequently in our code, we may be guided by the following advice [4, 6, 34, 35].

Choose only generators that were created and tested by the experts in the field. The code should be terse and preferably based on integer arithmetic in favor of greater speed. Choose a generator with the largest period and best serial uniformity for as many dimensions as possible. Preferably use generators that are accessible in source code, since modern compilers allow us to link short code fragments in the main program by using "inline" functions. Prior to its use, study the statistical property of a generator and determine whether any of its known deficiencies may endanger the correctness of your results. Individual rounds of computation should be performed by using different generators and seeds.

The quality of random number generators can be checked by statistically founded *batteries of empirical tests*. The best-known batteries—the tests contained within coincide in many cases—can be found in the classical collection due to Knuth [4], in the slightly more restrictive set DIEHARD by Marsaglia [36], in the NIST Statistical Test Suite [37, 38], and in the large set TESTU01 due to L'Ecuyer [28]. See also Appendices A–C in [18].

References

1. L. Devroye, *Non-uniform Random Variate Generation* (Springer, Berlin, 1986)
2. P. L'Ecuyer, Random number generation, in *The Handbook of Computational Statistics*, ed. by J.E. Gentle, W. Haerdle, Y. Mori (Springer, Berlin, 2004), p. 35
3. P. L'Ecuyer, Uniform random number generators and non-uniform random variate generation, in *International Encyclopedia of Statistical Science*, ed. by M. Lovric (Springer, Berlin, 2011)
4. D. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 3rd edn. (Addison-Wesley, Reading, 1998)
5. W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 3rd edn. (Cambridge University Press, Cambridge, 2007). See also the equivalent handbooks in Fortran, Pascal and C, as well as http://www.nr.com
6. H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods* (SIAM, Philadelphia, 1992)
7. J. E. Gentle, *Random Number Generation and Monte Carlo Methods* (Springer, Berlin, 2003)
8. P. L'Ecuyer, Uniform random number generation. Ann. Oper. Res. **53**, 77 (1994)
9. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 3rd edn. (MIT Press, Cambridge, 2009)
10. A.V. Peterson Jr., R.A. Kronmal, On mixture methods for the computer generation of random variables. Am. Stat. **36**, 184 (1982)
11. A.J. Walker, An efficient method for generating discrete random variables with general distributions. ACM Trans. Math. Softw. **3**, 253 (1977)
12. R.A. Kronmal, A.V. Peterson, On the alias method for generating random variables from a discrete distribution. Am. Stat. **33**, 214 (1979)
13. G.E.P. Box, M.E. Muller, A note on the generation of random normal deviates. Ann. Math. Stat. **29**, 610 (1958)
14. M. Horvat, The ensemble of random Markov matrices. J. Stat. Mech. **2009**, P07005 (2009)
15. A.J. Kinderman, J.F. Monahan, Computer generation of random variables using the ratio of uniform deviates. ACM Trans. Math. Softw. **3**, 257 (1977)

16. J.L. Leva, A fast normal random number generator. ACM Trans. Math. Softw. **18**, 449 (1992)
17. G. Marsaglia, `Usenet` newsgroup `sci.stat.math`, contributions dated 12 January 1999 and 1 August 1994
18. J.C. Collins, Testing, selection, and implementation of random number generators. Army Research Laboratory, Report ARL-TR-4498, 2008
19. G. Marsaglia, Random numbers fall mainly in the planes. Proc. Natl. Acad. Sci. **61**, 25 (1968)
20. U. Dieter, How to calculate shortest vectors in a lattice. Math. Comput. **29**, 827 (1975)
21. S.K. Park, K.W. Miller, Random number generators: good ones are hard to find. Commun. ACM **31**, 1192 (1988)
22. G. Marsaglia, A. Zaman, A new class of random number generators. Ann. Appl. Probab. **1**, 462 (1991)
23. R. Couture, P. L'Ecuyer, Distribution properties of MWC random number generators. Math. Comput. **66**, 591 (1997)
24. G. Marsaglia, Random number generators. J. Mod. Appl. Stat. Methods **2**, 2 (2003)
25. M. Lüscher, A portable high-quality random number generator for lattice field theory simulations. Comput. Phys. Commun. **79**, 100 (1994)
26. F. James, RANLUX: a Fortran implementation of the high-quality pseudorandom number generator of Lüscher. Comput. Phys. Commun. **79**, 111 (1994)
27. J. Eichenauer-Herrmann, K. Ickstadt, Explicit inversive congruential pseudorandom numbers with power of two modulus. Math. Comput. **62**, 787 (1994)
28. P. L'Ecuyer, R. Simard, TestU01: a C library for empirical testing of random number generators. ACM Trans. Math. Softw. **33**, 22 (2007), see also http://www.iro.umontreal.ca/~simardr/testu01/tu01.html
29. G. Marsaglia, Xorshift RNGs. J. Stat. Softw. **8**, 1 (2003)
30. F. Panneton, P. L'Ecuyer, R. P. Brent, Note on Marsaglia's Xorshift random number generators. J. Stat. Softw. **11**, 1 (2004)
31. F. Panneton, P. L'Ecuyer, On the Xorshift random number generators. ACM Trans. Model. Comput. Simul. **15**, 346 (2005)
32. T. G. Lewis, W. H. Payne, Generalized feedback shift register pseudorandom number algorithms. J. ACM **20**, 456 (1973)
33. M. Matsumoto, T. Nishimura, Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. **8**, 3 (1998)
34. S. Tezuka, *Uniform Random Numbers: Theory and Practice* (Kluwer Academic, Norwell, 1995)
35. O. Goldreich, *Modern Cryptography, Probabilistic Proofs and Pseudo-Randomness* (Springer, Berlin, 1999)
36. G. Marsaglia, DIEHARD battery of tests of randomness. http://www.stat.fsu.edu/pub/diehard/
37. A. L. Rukhin, Testing randomness: a suite of statistical procedures. Theor. Probab. Appl. **45**, 111 (2001)
38. A. Rukhin et al., NIST—a statistical test suite for random and pseudorandom number generators for cryptographical applications. http://csrc.nist.gov/groups/ST/toolkit/rng/index.html

Appendix D
Convergence Theorems for Iterative Methods

This appendix contains the basic theorems on convergence properties of iterative methods of the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{\phi}(\boldsymbol{x}_k) \tag{D.1}$$

used to solve non-linear equations

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}, \quad \boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x}) \in X, \tag{D.2}$$

where X is a vector space ($\mathbb{R}^n$ or $\mathbb{C}^n$). The convergence of the sequence of approximations $\{\boldsymbol{x}_k\}_{k\in\mathbb{N}_0}$ to the exact solution $\boldsymbol{\xi}$ of (D.2) depends on the local properties of the iteration function $\boldsymbol{\phi}$ defined by $\boldsymbol{f}$ and its derivatives (see Chap. 2). The sequence of vectors $\{\boldsymbol{x}_k\}$ converges to $\boldsymbol{\xi}$ if for any $\varepsilon > 0$, a $N(\varepsilon)$ exists such that

$$\|\boldsymbol{x}_k - \boldsymbol{\xi}\| < \varepsilon \quad \forall k \geq N(\varepsilon), \tag{D.3}$$

or $\lim_{k\to\infty} \boldsymbol{x}_k = \boldsymbol{\xi}$. Cauchy's definition of convergence is slightly different: the sequence of vectors $\{\boldsymbol{x}_k\}$ is convergent precisely when for any $\varepsilon > 0$ a $N(\varepsilon)$ exists such that

$$\|\boldsymbol{x}_k - \boldsymbol{x}_l\| < \varepsilon \quad \forall k, l \geq N(\varepsilon).$$

The definition (D.3) is independent of the choice of norm and is equivalent to the Cauchy formulation in complete metric spaces like $\mathbb{R}^n$ and $\mathbb{C}^n$. (In a complete space any Cauchy sequence is convergent.)

D.1 General Theorems

Theorem *Assume that* $\boldsymbol{\phi} : X \to X$ *has a fixed point* $\boldsymbol{\xi}$, *i.e.* $\boldsymbol{\phi}(\boldsymbol{\xi}) = \boldsymbol{\xi}$, *and that its neighborhood* $U(\boldsymbol{\xi})$, $p > 1$, *and* $C \geq 0$ *exist* (*or* $p = 1$ *and* $C \in [0, 1)$), *such that for all* $\boldsymbol{x} \in U(\boldsymbol{\xi})$ *we have* $\|\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\xi}\| \leq C\|\boldsymbol{x} - \boldsymbol{\xi}\|^p$. *Then a neighborhood* $V(\boldsymbol{\xi}) \subset U(\boldsymbol{\xi})$ *exists such that for each initial* $\boldsymbol{x}_0 \in V(\boldsymbol{\xi})$, *the points* $\boldsymbol{x}_k$ *obtained by the iteration* $\boldsymbol{x}_{k+1} = \boldsymbol{\phi}(\boldsymbol{x}_k)$ *converge to* $\boldsymbol{\xi}$ *at least with order* p.

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9, © Springer-Verlag Berlin Heidelberg 2012

The process in which $V(\boldsymbol{\xi}) \subset \mathbb{R}^n$ is locally convergent, while if $V(\boldsymbol{\xi}) = \mathbb{R}^n$ it is globally convergent. For the sequence to converge, $\boldsymbol{\phi}$ must fulfill certain conditions. The sufficient condition for the convergence of the iteration process (D.1) is given by the following theorem.

Theorem *Assume that $\boldsymbol{\xi}$ is the fixed point of $\boldsymbol{\phi}$. Let $S = \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{\xi}\| < r\}$ be the open neighborhood of $\boldsymbol{\xi}$, on which $\boldsymbol{\phi}$ is contractive: this means that a $K < 1$ exists such that*

$$\left\|\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\phi}(\boldsymbol{y})\right\| \leq K\|\boldsymbol{x} - \boldsymbol{y}\| \quad \forall \boldsymbol{x}, \boldsymbol{y} \in S \tag{D.4}$$

(Lipschitz inequality). Then we can use any initial approximation $\boldsymbol{x}_0 \in S$ in the iteration $\boldsymbol{x}_{k+1} = \boldsymbol{\phi}(\boldsymbol{x}_k)$ to generate the sequence $\{\boldsymbol{x}_k\}$, $k > 0$, for which $\boldsymbol{x}_k \in S$ holds true for $k = 1, 2, \ldots$, and

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{\xi}\| \leq K\|\boldsymbol{x}_k - \boldsymbol{\xi}\| \leq K^{k+1}\|\boldsymbol{x}_0 - \boldsymbol{\xi}\|.$$

This means that the sequence $\{\boldsymbol{x}_k\}$ converges to $\boldsymbol{\xi}$ at least linearly. In one dimension the open neighborhood in (D.4) is replaced by the open interval $I = (a, b)$, and the Lipschitz condition can be written as

$$\left|\phi(x) - \phi(y)\right| \leq \alpha|x - y|, \quad x, y \in I,\ 0 \leq \alpha < 1.$$

This theorem summarizes the necessary conditions for convergence, but it does not ensure that the limit exists. The sufficient condition for its existence is specified by the following theorem.

Theorem *Suppose we have the mapping $\boldsymbol{\phi} : X \to X$ and the initial point $\boldsymbol{x}_0$ of the sequence $\{\boldsymbol{x}_k : \boldsymbol{x}_{k+1} = \boldsymbol{\phi}(\boldsymbol{x}_k)\}$, $k \in \mathbb{N}_0$. If a neighborhood $S_r = \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}_0\| < r\}$ and a constant $K \in (0, 1)$ exist such that*

$$\left\|\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\phi}(\boldsymbol{y})\right\| \leq K\|\boldsymbol{x} - \boldsymbol{y}\| \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \overline{S}_r,$$
$$\|\boldsymbol{x}_1 - \boldsymbol{x}_0\| = \left\|\boldsymbol{\phi}(\boldsymbol{x}_0) - \boldsymbol{x}_0\right\| \leq (1 - K)r,$$

then $\boldsymbol{\phi}$ has precisely one fixed point ($\boldsymbol{\xi} = \boldsymbol{\phi}(\boldsymbol{\xi})$) in $\overline{S}_r$, to which the sequence $\{\boldsymbol{x}_k\}$ converges monotonously (and is contained in S_r).

For differentiable mappings $\boldsymbol{\phi}$ the theorem simplifies to the Banach contraction principle:

Theorem *Assume that for the mapping $\boldsymbol{\phi} : X \to X$ a region $\Omega \subset X$ exists that maps onto itself upon this mapping ($\boldsymbol{\phi}(\Omega) \subseteq \Omega$), and on which the derivative $\boldsymbol{J}(\boldsymbol{x}) = \boldsymbol{\phi}'(\boldsymbol{x})$ exists with the property*

$$\left\|\boldsymbol{J}(\boldsymbol{x})\right\|_2 < 1 \quad \forall \boldsymbol{x} \in \Omega.$$

Then in Ω there is exactly one fixed point $\boldsymbol{\xi}$ of $\boldsymbol{\phi}$. For an arbitrary initial point $\boldsymbol{x}_0 \in \Omega$ the corresponding sequence $\{\boldsymbol{x}_k : \boldsymbol{x}_{k+1} = \boldsymbol{\phi}(\boldsymbol{x}_k)\}$ remains in Ω and converges to the fixed point $\boldsymbol{\xi}$.

D.2 Theorems for the Newton–Raphson Method

Theorem *Let f be a real function. Choose $x_0 \in \mathbb{R}$ such that $f(x_0)f'(x_0) \neq 0$ and define $h_0 = f(x_0)/f'(x_0)$. Assume that $f''(x)$ exists on the interval $I_0 = [x_0, x_0 + 2h_0]$, and that $M = \sup_{x \in I_0} |f''(x)|$ and $2M|h_0| \leq |f'(x_0)|$. Then the elements of the sequence $\{x_k\}_{k \in \mathbb{N}_0}$ generated by the Newton–Raphson iteration,*

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \tag{D.5}$$

satisfy $x_k \in I_0$, and $\xi = \lim_{k \to \infty} x_k$ is the only zero of f in I_0. The method is of first order except if $\xi = x_0 + 2h_0$. In that case, f is a quadratic function and ξ is its double root. For consecutive approximations we have

$$|x_{k+1} - x_k| \text{ and } |\xi - x_{k+1}| \leq \frac{M}{2|f'(x_k)|}|x_k - x_{k-1}|^2, \quad k \in \mathbb{N}_0,$$

signifying a quadratic convergence of the sequence.

The following theorem describes the requirements on the function f that ensure absolute convergence of the Newton method. It teaches us that the initial approximation should always be picked from that side of the interval on which the function and its second derivative have equal signs.

Theorem *Assume that f is twice continuously differentiable on $[a, b]$, that $f(a)f(b) < 0$ (a zero on the interval), and $f''(x) \neq 0$ for any $x \in [a, b]$. If we choose the initial approximation such that $x_0 \in \{a, b\}$ and $f(x_0)f''(x_0) > 0$, the sequence of approximations $\{x_k\}$ in iteration (D.5) converges monotonously to ξ, which is the only zero of f on (a, b). The convergence is quadratic.*

In connection to the Maehly–Newton–Raphson method (Sect. 2.4.11) we also use the following theorem:

Theorem *Let p be a polynomial of degree $n \geq 2$ with exclusively real zeros. Order the zeros in magnitude, $\xi_1 \geq \xi_2 \geq \cdots \geq \xi_n$. Let α_1 be the largest zero of the derivative p': $\xi_1 \geq \alpha_1 \geq \xi_2$. If $n = 2$, we assume $\xi_1 \geq \xi_2$. Then for all $z > \xi_1$, the quantities defined by*

$$z' = z - \frac{p(z)}{p'(z)}, \qquad y = z - 2\frac{p(z)}{p'(z)}, \qquad y' = y - \frac{p(y)}{p(y)},$$

satisfy the inequalities

$$\alpha_1 < y, \qquad \xi_1 \leq y' \leq z'. \tag{D.6}$$

In the case $n = 2$ and $\xi_1 = \xi_2$, we have $y = \xi_1$ for all $z > \xi_1$ [1].

References

1. J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, 3rd edn. Texts in Appl. Math., vol. 12 (Springer, Berlin, 2002)

Appendix E
Numerical Integration

When integrands are tame functions of a single variable and the error of the numerical integral is not our main concern, basic quadrature formulas from any textbook of numerical methods can be used. In serious work one should pay attention to the errors: in particular when we build integration routines into larger program units, we should be aware of the required error tolerance and the error that is in fact attained. At the specified tolerance, one would like to perform as few arithmetic operations as possible, in particular the evaluations of the function being integrated.

We should therefore always use methods with error estimates, for example, the Simpson's formula to integrate a function f given on the interval $[a, b]$ at $(N + 1)$ equidistant points (where N is even) spaced $h = (b - a)/N$ apart,

$$\int_a^b f(x)\,\mathrm{d}x = I_N + R_N,$$

where

$$I_N = \frac{h}{3}\left[f(a) + 4\sum_{j=1}^{N/2} f\big(a + (2j-1)h\big) + 2\sum_{j=1}^{N/2-1} f(a + 2jh) + f(b) \right].$$

The remainder has the form $R_N = \sum_{n=2}^{\infty} b_n h^{2n}$. Its leading term is

$$R_N \approx -\frac{b-a}{180} h^4 f^{(4)}(\xi), \quad \xi \in [a, b],$$

which can be used to calculate a better estimate for the value of the integral and the error. The integral is first computed with $(N + 1)$ and then with $(2N + 1)$ points, so that we can write $I = I_N + R_N = I_{2N} + R_{2N}$, where

$$|R_N| = \frac{b-a}{180} h^4 \big| f^{(4)}(\xi_1) \big|, \qquad |R_{2N}| = \frac{b-a}{180} \left(\frac{h}{2}\right)^4 \big| f^{(4)}(\xi_2) \big|.$$

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9, © Springer-Verlag Berlin Heidelberg 2012

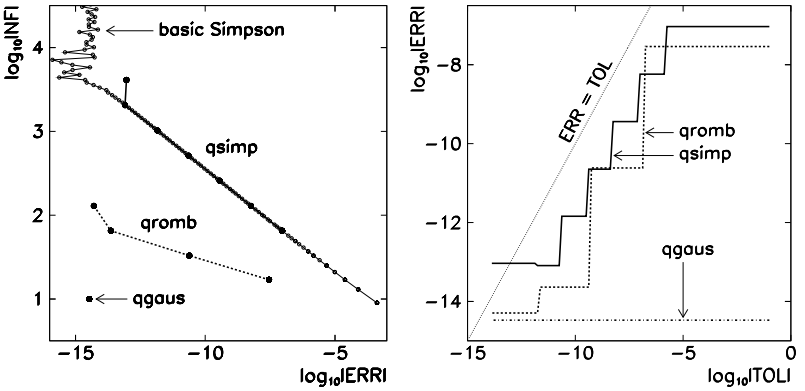


Fig. E.1 Numerical integration of the function $f(x) = x^2(x^2 - 2)\sin x$ on $[0, \pi/2]$ by using the basic Simpson's method and by using routines `qsimp`, `qromb` and `qgaus` from the NUMERICAL RECIPES library [1]. [*Left*] The number NF of integrand evaluations as a function of the required relative error ERR of the integral. [*Right*] The achieved relative error compared to the required tolerance TOL in subsequent halving of the subinterval. Due to rounding errors, the value of ERR can be even larger than TOL if we require TOL to be very small (`qsimp`, lower left corner)

If we assume $|f^{(4)}(\xi_1)| \approx |f^{(4)}(\xi_2)|$, the ratio of the remainders is

$$\left|\frac{R_{2N}}{R_N}\right| \approx \frac{1}{16}.$$

It follows that

$$|I_{2N} - I_N| = |R_{2N} - R_N| \approx \frac{15}{16}|R_N|.$$

The quantity I_{2N} is therefore an improved numerical estimate for the integral, while a naive (and conservative) estimate for the error of the integral I_N is

$$\frac{16}{15}|I_{2N} - I_N|.$$

Simpson's method requires many function evaluations (see Fig. E.1 (left)). By recursive halving of the subintervals, which is repeated until the relative error drops below a specified value, the error of the final result can be reduced by many orders of magnitude (see Fig. E.1 (right) corresponding to `qsimp` from [1]). Extrapolation methods (like the Romberg algorithm, see `qromb` in the figure) are even faster. But quadrature formulas (like the Gauss quadrature, `qgaus` in the figure), are equally tempting: they require few evaluations of $f(x)$ and achieve breathtaking precision for smooth functions.

Table E.1 Nodes x_j and weights w_j in quadrature formulas for different weight functions W. The Gauss–Legendre quadrature corresponds to the polynomials P_n, while the Gauss–Chebyshev quadrature of the first and second kind to the polynomials T_n and U_n. The nodes and weights for the Gauss–Legendre quadrature are listed in Table 25.4 of [2]

Interval	$W(x)$	Polynomial	x_j	w_j
$[-1, 1]$	1	P_n	solution of $P_n(x_j) = 0$	$\frac{2}{(1-x_j^2)(P_n'(x_j))^2}$
$(-1, 1)$	$\frac{1}{\sqrt{1-x^2}}$	T_n	$\cos\frac{(2j-1)\pi}{2n}$	$\frac{\pi}{n}$
$[-1, 1]$	$\sqrt{1-x^2}$	U_n	$\cos\frac{j\pi}{n+1}$	$\frac{\pi}{n+1}\sin^2\frac{j\pi}{n+1}$

E.1 Gauss Quadrature

Quadrature formulas are used to compute definite integrals $\int_a^b W(x) f(x)\,\mathrm{d}x$, where f is a given function and W a non-negative weight. The integral is approximated by the sum

$$\int_a^b W(x) f(x)\,\mathrm{d}x \approx \sum_{j=1}^{n} w_j f(x_j).$$

For certain weight functions W, such nodes x_j and weights w_j can be found that the quadrature formula is exact for polynomial functions f up to the polynomial degrees of $2n-1$. Each of these unique weight functions W belongs to a particular class of orthogonal polynomials, and the nodes x_j are the zeros of these polynomials. The most important ones are shown in Table E.1.

Let us focus on the Gauss–Legendre quadrature of order n on $[-1, 1]$,

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx G_n = \sum_{j=1}^{n} w_j f(x_j). \tag{E.1}$$

The numerical values for the nodes and weights of order $n = 7$ are listed to a precision of 15 digits in the upper part of Table E.2. If the function f is integrated on $[a, b]$ instead of on $[-1, 1]$, the argument of f needs to be modified, and the sum weighted by the length of the interval:

$$\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{2}\sum_{j=1}^{n} w_j f\left(\frac{b-a}{2}x_j + \frac{a+b}{2}\right) + R_n. \tag{E.2}$$

The error R_n in expression (E.2) is (see [3]):

$$R_n = \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n)}(\xi), \quad a < \xi < b. \tag{E.3}$$

Table E.2 Nodes and weights for the Gauss–Kronrod quadrature (G_7, K_{15}) on $[-1, 1]$

Nodes for G_7	Weights for G_7
±0.949107912342759	0.129484966168870
±0.741531185599394	0.279705391489277
±0.405845151377397	0.381830050505119
0.000000000000000	0.417959183673469
Nodes for K_{15}	**Weights for K_{15}**
±0.991455371120813	0.022935322010529
±0.949107912342759	0.063092092629979
±0.864864423359769	0.104790010322250
±0.741531185599394	0.140653259715525
±0.586087235467691	0.169004726639267
±0.405845151377397	0.190350578064785
±0.207784955007898	0.204432940075298
0.000000000000000	0.209482141084728

E.1.1 Gauss–Kronrod Quadrature

The estimate (E.3) is not very useful. It requires us to compute the $2n$th derivative of f and tends to grossly over-estimate the error. The error can be reduced in a controlled manner such that the quadrature weights are adaptively condensed or rearranged: new nodes are inserted between nodes of the previous division, and the integration rules are used on these sub-meshes. The integration precision is increased by covering the problematic regions of the integrand by ever denser quadrature formulas of lower orders. But such "adaptivity" greatly diminishes the symmetric beauty of Gauss formulas.

For a minimum number of evaluations of f we prefer to consider the "optimal case": we make the existing mesh denser by embedding one mesh into another. Such optimal embedding is realized in the Gauss–Kronrod quadrature. First, we compute the Gauss quadrature G_n of order n by using (E.1). Second, we add $n+1$ new nodes such that the corresponding quadrature formula has a higher order. We compute the sums

$$G_n = \sum_{j=1}^{n} w_j f(x_j), \tag{E.4}$$

$$K_{2n+1} = \sum_{j=1}^{n} a_j f(x_j) + \sum_{k=1}^{n+1} \alpha_k f(\xi_k). \tag{E.5}$$

The quadratures G_n and K_{2n+1} share the n nodes x_j, but differ in the weights corresponding to these nodes: in the first sum of (E.5) we have a_j instead of w_j

from (E.4). The second sum in (E.5) requires $n+1$ additional nodes ξ_k and weights α_k. In total, we need $2n+1$ evaluations of the function f, but the resulting quadrature formula K_{2n+1} has order $3n+1$! The value of K_{2n+1} is therefore a spectacularly improved estimate of the integral. For the error estimate of the integral G_n, Ref. [4] recommends

$$\left(200|G_n - K_{2n+1}|\right)^{3/2}.$$

The numerical values of the additional nodes and weights for $n = 7$ (the standard Gauss–Kronrod pair (G_7, K_{15})) are listed to 15-digit precision in the lower part of Table E.2. The method to compute the nodes and the weights is described in [5,6]. An interesting alternative to Gauss quadrature with certain numerical advantages are the Fejér and Clenshaw–Curtis formulas [7,8].

E.1.2 Quadrature in Two Dimensions

A generalization of the one-dimensional formulas to two dimensions is not trivial. In square geometry $(x, y) \in [-1, 1] \times [-1, 1]$, Gauss quadrature (of equal orders in x and y directions) has the form

$$\int_{-1}^{1}\int_{-1}^{1} f(x, y)\,\mathrm{d}x\,\mathrm{d}y \approx \sum_{k=1}^{n} w_k f(x_k, y_k),$$

where the nodes and the weights satisfy a system of non-linear equations

$$\sum_{k=1}^{n} w_k x_k^i y_k^j = \frac{1+(-1)^i}{i+1} \cdot \frac{1+(-1)^j}{j+1}, \tag{E.6}$$

for $i = 0, 1, \ldots, m$, $j = 0, 1, \ldots, m-i$, and $n = \frac{1}{6}(m^2+3m+2)$. Some popular sets of nodes and weights for quadrature in two and three dimensions are collected in [2] (pp. 891–895), but those formulas are not optimal. The system (E.6) is hard to solve, in particular for high orders, and we tend to use a simple Cartesian product of one-dimensional formulas. For example, the integral on the square $(x, y) \in [-1, 1] \times [-1, 1]$ can be computed by using the formula

$$\int_{-1}^{1}\int_{-1}^{1} f(x, y)\,\mathrm{d}x\,\mathrm{d}y \approx \sum_{j=1}^{n}\sum_{k=1}^{m} w_j^{(n)} w_k^{(m)} f(x_j, y_k),$$

where quadratures in x and y directions may be of different orders. This simple product formula has mn nodes and requires just as many evaluations of $f(x, y)$. The weight corresponding to the node (x_j, y_k) is simply a product of the weights from individual one-dimensional quadratures. Optionally, one can even apply different integration rules in different directions, for example, if functions are tame in one independent variable but wild in another.

E.2 Integration of Rapidly Oscillating Functions

Introductory textbooks on numerical methods rarely discuss recipes for integration of rapidly oscillating functions although they are widely used in physics, chemistry, and engineering. Even in well-known libraries, such algorithms are scarce. Here, we present some basic approaches. We are mostly interested in the integrals of the form

$$I[f] = \int_a^b f(x)\, \mathrm{e}^{\mathrm{i}\omega g(x)}\, \mathrm{d}x, \quad -\infty < a < b < \infty, \tag{E.7}$$

where f and g are smooth functions and $|\omega|$ is large. A well-known example is the Fourier integral, in which the oscillation function or the *oscillator* is $g(x) = x$. For such integrals, the quadrature and extrapolation methods of the previous section are woefully inadequate: when $1/|\omega|$ becomes much smaller than the number of nodes, quadrature formulas fail. To compute the integrals of the form (E.7) we use the asymptotic method, the Filon method, or their numerous improvements and generalizations. See also Sect. 1.3.

E.2.1 Asymptotic Method

Let us first discuss the case where $g'(x) \neq 0$ for $x \in [a, b]$. Then one can demonstrate [9,10] that asymptotically the integral behaves as $I[f] = \mathcal{O}(1/\omega)$ when $|\omega| \to \infty$, and that the value of the integral can be expanded in powers of $1/\omega$ as

$$I[f] \sim -\sum_{m=0}^{\infty} \frac{1}{(-\mathrm{i}\omega)^{m+1}} \left[\frac{\mathrm{e}^{\mathrm{i}\omega g(x)}}{g'(x)}\, f_m(x) \right]_a^b, \quad \omega \to \infty, \tag{E.8}$$

where for the functions f_m we have the recurrence

$$\begin{aligned} f_0(x) &= f(x), \\ f_{m+1}(x) &= \frac{\mathrm{d}}{\mathrm{d}x} \left(\frac{f_m(x)}{g'(x)} \right), \quad m = 0, 1, \ldots. \end{aligned}$$

Even though this recurrence is analytically simple, it generates a clumsy sequence of functions which is hard to implement in computer code for non-trivial functions g, unless one exploits symbolic computation. In the special case of the Fourier oscillator, $g(x) = x$, i.e. in computing the integrals of the form

$$\int_a^b f(x) \mathrm{e}^{\mathrm{i}\omega x}\, \mathrm{d}x, \tag{E.9}$$

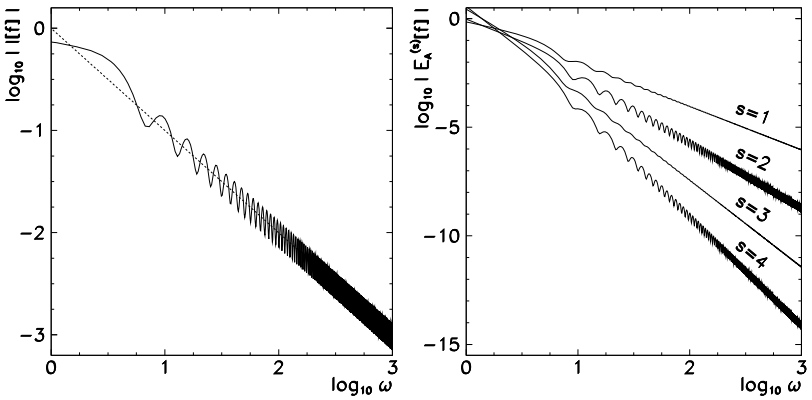


Fig. E.2 Numerical calculation of the integral $I[f] = \int_0^1 \cos^2 x \, \mathrm{e}^{\mathrm{i}\omega x} \, \mathrm{d}x$ by the asymptotic method. [*Left*] Exact value of $I[f]$ with the asymptotics $|I[f]| \sim \mathcal{O}(1/\omega)$. [*Right*] Error of the asymptotic method for $s = 1, 2, 3, 4$ with the behavior $|E_{\mathrm{A}}^{(s)}[f]| \sim \mathcal{O}(\omega^{-s-1})$

this recurrence simplifies considerably. We have $g'(x) = 1$ and $g^{(r)}(x) = 0$ for $r > 1$, thus $f_m(x) = f^{(m)}(x)$, and the integral has the asymptotic expansion

$$I[f] \sim -\sum_{m=0}^{\infty} \frac{[\mathrm{e}^{\mathrm{i}\omega x} f^{(m)}(x)]_a^b}{(-\mathrm{i}\omega)^{m+1}} = \sum_{m=0}^{\infty} \frac{\mathrm{e}^{\mathrm{i}\omega a} f^{(m)}(a) - \mathrm{e}^{\mathrm{i}\omega b} f^{(m)}(b)}{(-\mathrm{i}\omega)^{m+1}}.$$

We should stress again that this expansion in general does not converge when $m \to \infty$, while it certainly converges when $|\omega| \to \infty$ (see Sect. 1.3.2).

We generate an asymptotic quadrature formula of order s if we keep only s terms in (E.8). For a general function g this means

$$I[f] \approx Q_{\mathrm{A}}^{(s)}[f] = \frac{\mathrm{e}^{\mathrm{i}\omega g(a)}}{g'(a)} \sum_{m=0}^{s-1} \frac{f_m(a)}{(-\mathrm{i}\omega)^{m+1}} - \frac{\mathrm{e}^{\mathrm{i}\omega g(b)}}{g'(b)} \sum_{m=0}^{s-1} \frac{f_m(b)}{(-\mathrm{i}\omega)^{m+1}}. \tag{E.10}$$

For the Fourier oscillator $g(x) = x$ we set $f_m(a) = f^{(m)}(a)$, $f_m(b) = f^{(m)}(b)$ and $g'(a) = g'(b) = 1$. The error of the asymptotic quadrature is

$$E_{\mathrm{A}}^{(s)}[f] = Q_{\mathrm{A}}^{(s)}[f] - I[f] = -\frac{I[f_s]}{(-\mathrm{i}\omega)^s} \sim \omega^{-s-1}. \tag{E.11}$$

Figure E.2 shows the numerical calculation of the integral $\int_0^1 \cos^2 x \, \mathrm{e}^{\mathrm{i}\omega x} \, \mathrm{d}x$ by the asymptotic method (E.10). The left side shows the exact value of the integral as a function of ω, while the right side shows the dependence of the error with the characteristic $\sim\omega^{-s-1}$ trend. The reader might be particularly impressed by two observations. The precision of the quadrature *improves* when ω in increased, which is counter-intuitive. The second surprise is the decent precision we obtain, for large

enough ω, already at first order ($s = 1$): for $\omega \approx 1000$, the trivial formula

$$Q_{\mathrm{A}}^{(1)}[f] = \frac{\mathrm{i}}{\omega}\left[\mathrm{e}^{\mathrm{i}\omega a} f(a) - \mathrm{e}^{\mathrm{i}\omega b} f(b)\right]$$

is precise to about six digits!

The asymptotic method becomes only moderately more complicated when $g'(x) = 0$ on $x \in (a, b)$, or when $g'(a) = 0$ and/or $g'(b) = 0$: the formulas for such cases can be found in [11]. The largest obstacle in these methods is the annoying recursive computation of the functions f_m which is hard to automate for non-trivial g. The approach due to Filon [12, 13] offers an appealing alternative.

E.2.2 Filon's Method

The basic idea of the Filon's method is to remove the oscillatory part of the integral $\mathrm{e}^{\mathrm{i}\omega g(x)}$ from the integrand in (E.7), and to translate the problem to simpler integrals that play the role of weights in a quadrature formula. The function f is replaced by a polynomial approximation $p(x) = \sum_{k=0}^{n} c_k x^k$ interpolating f at the points $x_0, x_1, \ldots, x_n$. The quadrature formula then becomes

$$I[f] \approx Q_{\mathrm{F}}[f] = I[p] = \int_a^b \left(\sum_{k=0}^{n} c_k x^k\right) \mathrm{e}^{\mathrm{i}\omega g(x)}\, \mathrm{d}x = \sum_{k=0} c_k\, \mu_k(\omega).$$

To evaluate it, we need the coefficients c_k and $(n + 1)$ values of the integrals of the form

$$\mu_k(\omega) = \int_a^b x^k\, \mathrm{e}^{\mathrm{i}\omega g(x)}\, \mathrm{d}x, \tag{E.12}$$

which are called *moments*. We compute the coefficients c_k by solving the corresponding Vandermonde system: see Sect. 3.2.5 and [14–16]. (In MATHEMATICA this is accomplished by the commands `InterpolatingPolynomial` and `CoefficientList`.)

The error of the Filon's formula $E_{\mathrm{F}}[f]$ can be determined by using the difference function $E(x) = f(x) - p(x)$ that measures the deviation of f from its interpolation polynomial p. We have $E_{\mathrm{F}}[f] = I[f] - Q_{\mathrm{F}}[f] = I[f] - I[p] = I[f - p] = I[E]$. By analogy with (E.8), one immediately sees that

$$E_{\mathrm{F}}[f] = I[E] \sim -\sum_{m=0}^{\infty} \frac{1}{(-\mathrm{i}\omega)^{m+1}} \left[\frac{\mathrm{e}^{\mathrm{i}\omega g(x)}}{g'(x)}\, E(x)\right]_a^b,$$

where we have assumed $g'(x) \neq 0$ for $x \in [a, b]$. Different asymptotic orders s can be achieved by using the Filon quadrature. If we use an interpolation polynomial p for which

$$p^{(m)}(a) = f^{(m)}(a), \qquad p^{(m)}(b) = f^{(m)}(b), \quad m = 0, 1, \ldots, s - 1, \tag{E.13}$$

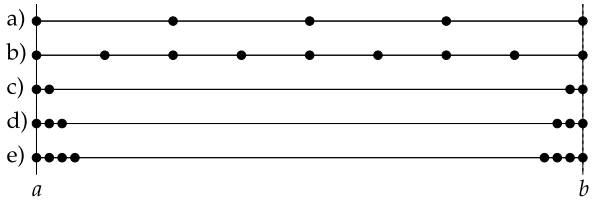


Fig. E.3 The points used to interpolate f in the Filon's method. (*a*) Equidistant mesh with 5 points. (*b*) Mesh with 9 points. (*c*) Inversely spaced mesh $\{a, a + 1/\omega, b - 1/\omega, b\}$. (*d*) Inverse mesh $\{a, a + 1/\omega, a + 2/\omega, b - 2/\omega, b - 1/\omega, b\}$. (*e*) Inverse mesh $\{a, a + 1/\omega, a + 2/\omega, a + 3/\omega, b - 3/\omega, b - 2/\omega, b - 1/\omega, b\}$. See also Fig. E.4

i.e. if p interpolates all function's values and derivatives up to order $(s - 1)$ at the boundary points of the interval $[a, b]$, the error is $E_{\mathrm{F}}[f] = \mathcal{O}(\omega^{-s-1})$, just like the error (E.11) of the asymptotic quadrature (see Fig. E.4 (right)). In this case, Filon's quadrature $Q_{\mathrm{F}}^{(s)}$ of order s is exact for all polynomials of degree $\leq 2s - 1$.

The integrals μ_k contain rapidly oscillating functions and are hard to compute for general g; to compute them, we can use the asymptotic method described earlier. In the following, we again restrict the discussion to the Fourier integrals (E.9) with $g(x) = x$. In this case, the moments (E.12) can be easily computed, and they are related by the recurrence

$$\mu_0 = \frac{\mathrm{e}^{\mathrm{i}\omega b} - \mathrm{e}^{\mathrm{i}\omega a}}{\mathrm{i}\omega}, \qquad \mu_k = \frac{\mathrm{e}^{\mathrm{i}\omega b} b^k - \mathrm{e}^{\mathrm{i}\omega a} a^k - k\mu_{k-1}}{\mathrm{i}\omega}, \quad k = 1, 2, \ldots, n.$$

Figure E.4 shows the error of the numerical calculation of the integral $\int_0^1 \mathrm{e}^x \, \mathrm{e}^{\mathrm{i}\omega x} \, \mathrm{d}x$ by using the Filon's method. Higher quadrature orders can be attained by trying to fulfill the requirements (E.13) at a chosen s as closely as possible. We add further ω-dependent points (Fig. E.3, cases c–e) in the vicinity of the boundary points. By doing this, not only do we capture the correct function's values $f(a)$ and $f(b)$, but also the derivatives $f'(a)$, $f'(b)$, $f''(a)$, $f''(b)$, and so on.

Typically, we encounter only a limited set of oscillator functions $g(x)$, like x, x^2, $s_0 + s_1 x + s_2 x^2$, x^p, or $\sum_{j=0}^{p} s_j x^j$. Extensive derivations, a collection of formulas for the moments μ_k, and several numerical tricks to speed up the quadrature with such sets of functions can be found in [11]. A slightly different quadrature method to compute integrals of the form (E.7), utilizing collocation and the usual two-point Gauss–Legendre formula, is presented in [17]. The integrals containing Bessel functions of the type

$$\int_a^b f(x) \, J_m(sx) \, \mathrm{d}x \quad \text{and} \quad \int_a^b f(x) \cos(rx) J_m(sx) \, \mathrm{d}x$$

are described in [18]. An efficient approach to compute integrals of the form (E.7) with $b = \infty$ is discussed in the papers [19, 20]. Robust subroutines for the numerical integration of oscillating functions on finite or infinite intervals can be found in the QUADPACK package [21] which is also implemented in the GSL library.

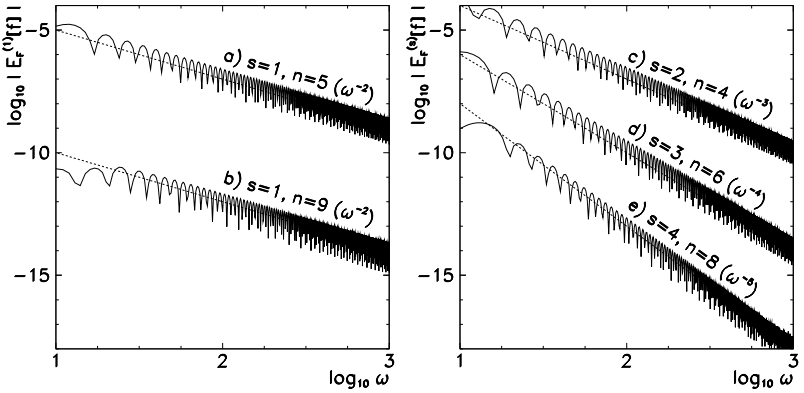


Fig. E.4 Numerical calculation of the integral $I[f] = \int_0^1 \mathrm{e}^x \, \mathrm{e}^{\mathrm{i}\omega x} \, \mathrm{d}x$ by the Filon's method. The quadrature nodes are defined in Fig. E.3. [*Left*] Error of the first-order quadrature $|E_{\mathrm{F}}^{(1)}[f]|$ with five and nine equidistant points on $[a, b]$. By adding further interpolation points for f we can reduce the leading constant of the error, but the order of convergence remains the same. [*Right*] Error of the quadrature $|E_{\mathrm{F}}^{(s)}[f]|$ with three inverse quadrature meshes. By adding nodes in the vicinity of the boundaries ($x = a$ and $x = b$) we approximate higher derivatives of f, thus the order of convergence increases

E.3 Integration of Singular Functions

In everyday work one frequently encounters integrals of singular functions, which need to be understood as principal values

$$I(\lambda) = P \int_a^b \frac{f(x)}{x - \lambda} \, \mathrm{d}x = \lim_{\varepsilon \to 0} \left[\int_a^{\lambda - \varepsilon} \frac{f(x)}{x - \lambda} \, \mathrm{d}x + \int_{\lambda + \varepsilon}^b \frac{f(x)}{x - \lambda} \, \mathrm{d}x \right].$$

We split such integrals into three parts,

$$I(\lambda) = \int_a^{\lambda - \Delta} \frac{f(x)}{x - \lambda} \, \mathrm{d}x + \underbrace{P \int_{\lambda - \Delta}^{\lambda + \Delta} \frac{f(x)}{x - \lambda} \, \mathrm{d}x}_{I_\Delta(\lambda)} + \int_{\lambda + \Delta}^b \frac{f(x)}{x - \lambda} \, \mathrm{d}x,$$

so that the singular part of the integrand is framed by a symmetric interval $[\lambda - \Delta, \lambda + \Delta]$. By substituting $x - \lambda = X$ and $x = X/\Delta$, the integral $I_\Delta(\lambda)$ can be rewritten in a form that can be handled by a suitable quadrature formula,

$$\begin{aligned} I_\Delta(\lambda) &= P \int_{-\Delta}^{\Delta} \frac{f(X + \lambda) - f(\lambda)}{X} \, \mathrm{d}X = P \int_{-1}^{1} \frac{f(x\Delta + \lambda) - f(\lambda)}{x} \, \mathrm{d}x \\ &\approx \sum_j \frac{w_j}{x_j} \big[f(x_j \Delta + \lambda) - f(\lambda) \big]. \end{aligned}$$

It makes sense to take a quadrature formula with an order as high as possible, and with an even number of points in order to avoid the calculation of the derivative

of f [22, 23]. See also [24] and Sect. 4.5. Other reliable numerical methods for the integration of singular functions are implemented in QUADPACK [21].

References

1. W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 3rd edn. (Cambridge University Press, Cambridge, 2007). See also the equivalent handbooks in Fortran, Pascal and C, as well as http://www.nr.com
2. M. Abramowitz, I. A. Stegun, *Handbook of Mathematical Functions*, 10th edn. (Dover, Mineola, 1972)
3. J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, 3rd edn. Texts in Appl. Math., vol. 12 (Springer, Berlin, 2002)
4. D. Kahaner, C. Moler, S. Nash, *Numerical Methods and Software* (Prentice-Hall, Englewood Cliffs, 1989)
5. D. Calvetti, G. H. Golub, W. B. Gragg, L. Reichel, Computation of Gauss–Kronrod quadrature rules. Math. Comput. **69**, 1035 (2000)
6. D.P. Laurie, Calculation of Gauss–Kronrod quadrature rules. Math. Comput. **66**, 1133 (1997)
7. J. Waldvogel, Fast construction of the Fejér and Clenshaw–Curtis quadrature rules. BIT Numer. Math. **46**, 195 (2006)
8. L.N. Trefethen, Is Gauss quadrature better than Clenshaw–Curtis? SIAM Rev. **50**, 67 (2008)
9. A. Iserles, S.P. Nørsett, On quadrature methods for highly oscillatory integrals and their implementation. BIT Numer. Math. **44**, 755 (2004)
10. A. Iserles, S.P. Nørsett, Efficient quadrature of highly oscillatory integrals using derivatives. Proc. R. Soc. A **461**, 1383 (2005)
11. J. Vanbiervliet, Numerical methods for highly oscillatory integrals and integral equations. MSc thesis, KU Leuven, Leuven, 2005
12. A. Iserles, On the numerical quadrature of highly-oscillating integrals, I: Fourier transforms. IMA J. Numer. Anal. **24**, 365 (2004)
13. A. Iserles, On the numerical quadrature of highly-oscillating integrals, II: Irregular oscillators. IMA J. Numer. Anal. **25**, 25 (2005)
14. N.J. Higham, Fast solution of Vandermonde-like systems involving orthogonal polynomials. IMA J. Numer. Anal. **8**, 473 (1988)
15. Å. Björck, V. Pereyra, Solution of Vandermonde systems of equations. Math. Comput. **24**, 893 (1970)
16. D. Calvetti, L. Reichel, Fast inversion of Vandermonde-like matrices involving orthogonal polynomials. BIT Numer. Math. **33**, 473 (1993)
17. S. Xiang, Efficient quadrature for highly oscillatory integrals involving critical points. J. Comput. Appl. Math. **206**, 688 (2007)
18. S. Xiang, Numerical analysis of a fast integration method for highly oscillatory functions. BIT Numer. Math. **47**, 469 (2007)
19. T. Sauter, Computation of irregularly oscillating integrals. Appl. Numer. Math. **35**, 245 (2000)
20. T. Sauter, Integration of highly oscillatory functions. Comp. Phys. Commun. **125**, 119 (2000)
21. R. Piessens, E. De Doncker-Kapenga, C.W. Überhuber, *QUADPACK: A Subroutine Package for Automatic Integration* (Springer, Berlin, 1983). See also http://www.netlib.org/quadpack/
22. J.V. Noble, Gauss–Legendre principal value integration. Comput. Sci. Eng. **Jan/Feb**, 92 (2000)
23. W.J. Thompson, Principal-value integrals by a simple and accurate finite-interval method. Comput. Phys. **12**, 94 (1998)
24. P. Rabinowitz, Gauss–Kronrod integration rules for Cauchy principal value integrals. Math. Comput. **41**, 63 (1983)

Appendix F
Fixed Points and Stability ⋆

F.1 Linear Stability

In Chap. 7 we discuss autonomous systems of ordinary differential equations with specified initial conditions:

$$\boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{y}), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0 \in \mathbb{R}^M, \ x \geq x_0. \tag{F.1}$$

In explicit one-step difference methods we approximate the system (F.1) by the discrete mapping

$$\boldsymbol{y}_{n+1} = \boldsymbol{F}(\boldsymbol{y}_n), \tag{F.2}$$

where $\boldsymbol{F}$ is a known function (e.g. $\boldsymbol{F}(\boldsymbol{y}) = \boldsymbol{y} + h\boldsymbol{f}(\boldsymbol{y})$ in the explicit Euler scheme). When analyzing the stability of the solutions of (F.1) one would like to know whether two solutions of the equation that are nearby at some time, remain nearby at later times, or perhaps even asymptotically approach each other. We define two types of stability visualized in Fig. F.1. (The reader can easily generalize these definitions to the mapping (F.2).)

Definition The solution $\boldsymbol{y}^*$ corresponding to (F.1) is locally attractive (*linearly stable*) *in Lyapunov sense* if for some $\varepsilon > 0$ such $\delta > 0$ exists that for any other solution $\boldsymbol{z}$ of (F.1) that satisfies $\|\boldsymbol{y}^*(x_0) - \boldsymbol{z}(x_0)\| < \delta$, also $\|\boldsymbol{y}^*(x) - \boldsymbol{z}(x)\| < \varepsilon$ holds true for $x > x_0$.

Definition The solution $\boldsymbol{y}^*$ corresponding to (F.1) is *asymptotically stable* if it is Lyapunov stable and for any other solution $\boldsymbol{z}$ of (F.1) such $\beta > 0$ exists that $\lim_{x\to\infty} \|\boldsymbol{y}^*(x) - \boldsymbol{z}(x)\| = 0$ when $\|\boldsymbol{y}^*(x_0) - \boldsymbol{z}(x_0)\| < \beta$.

Suppose that the true solution (flow) $\boldsymbol{y}$ approaches a constant value $\boldsymbol{y}^*$ when $x \to \infty$. Then clearly $\boldsymbol{f}(\boldsymbol{y}^*) = 0$, and $\boldsymbol{y}^*$ is called the *fixed point* of the flow $\boldsymbol{y}$ corresponding to (F.1). In the case of a mapping the fixed point is the point $\boldsymbol{y}^*$ for which $\boldsymbol{F}(\boldsymbol{y}^*) = \boldsymbol{y}^*$. The study of stability of differential equations and difference schemes focuses primarily on the stability near the fixed points.

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9, © Springer-Verlag Berlin Heidelberg 2012

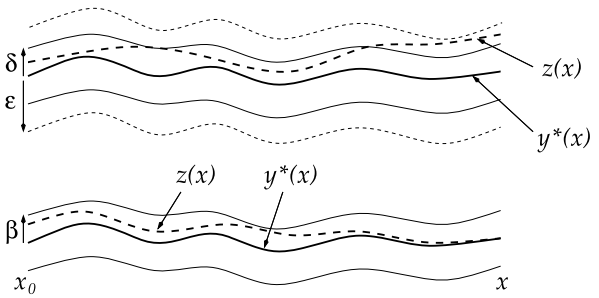


Fig. F.1 Geometric illustration of stability in the Lyapunov sense [*Top*] and asymptotic stability [*Bottom*]

In the vicinity of the fixed point $\boldsymbol{y}^*$ we denote $\boldsymbol{y} = \boldsymbol{y}^* + \boldsymbol{\delta}$ when dealing with a differential equation, or $\boldsymbol{y}_n = \boldsymbol{y}^* + \boldsymbol{\delta}_n$ when we think in terms of a discrete mapping. Then (F.1) can be linearized,

$$\boldsymbol{\delta}' = \boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{y}) \approx \underbrace{\boldsymbol{f}(\boldsymbol{y}^*)}_{0} + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{y}^*} \boldsymbol{\delta},$$

and similarly for the mapping (F.2),

$$\boldsymbol{y}^* + \boldsymbol{\delta}_{n+1} = \boldsymbol{F}(\boldsymbol{y}_n) \approx \underbrace{\boldsymbol{F}(\boldsymbol{y}^*)}_{\boldsymbol{y}^*} + \left.\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{y}^*} \boldsymbol{\delta}_n.$$

In the following we shift the fixed point to the origin, so $\boldsymbol{\delta} \to \boldsymbol{y}$ for equations and $\boldsymbol{\delta}_n \to \boldsymbol{y}_n$ for mappings. Then we have, to first order,

$$\boldsymbol{y}' = A\boldsymbol{y}, \quad A_{ij} = \left[\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{0}}\right]_{ij}, \tag{F.3}$$

$$\boldsymbol{y}_{n+1} = A\boldsymbol{y}_n, \quad A_{ij} = \left[\left.\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{0}}\right]_{ij}. \tag{F.4}$$

The behavior at large x for equations (or large n for mappings) is determined by the eigenvalues of the Jacobi matrix $\partial \boldsymbol{f}/\partial \boldsymbol{y}$ (or $\partial \boldsymbol{F}/\partial \boldsymbol{y}$). The system (F.3) is stable in the Lyapunov sense precisely when all roots of the equation

$$\det(\lambda I - A) = a_0\lambda^n + a_1\lambda^{n-1} + \cdots + a_{n-1}\lambda + a_n = 0$$

satisfy $\operatorname{Re}\lambda_i \le 0$, and all multiple roots satisfy $\operatorname{Re}\lambda_i < 0$. A similar argument applies to the asymptotic stability of the linearized system (F.4): the sufficient condition for it is that the spectral radius of the Jacobi matrix for the mapping (F.2) is bounded,

$$\rho\left(\left.\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{y}^*}\right) < 1.$$

The Routh–Hurwitz criterion [1] allows us to determine whether the eigenvalues satisfy the stability condition from the coefficients a_i alone, i.e. without actually solving the characteristic equation. Since the computation of the characteristic polynomial for large matrices is difficult and the search for zeros is plagued by round-off errors, it is recommendable to use dedicated, numerically stable algorithms to find the eigenvalues.

If the matrix elements of A are not constant (non-autonomous systems), as in

$$\boldsymbol{y}' = A(x)\boldsymbol{y},$$

and we wish to examine their asymptotic stability, it *does not suffice* to check the criteria $\mathrm{Re}\,\lambda_i \leq 0$ for simple roots or $\mathrm{Re}\,\lambda_i < 0$ for multiple roots of the characteristic equation. But specific cases exist [1] for which these checks are sufficient. Asymptotic stability is ensured if $A_{ii}(x) < 0$ and $A(x)$ is diagonally dominant; or if a decomposition $A(x) = B + C(x)$ exists, where B is a constant matrix and all its eigenvalues satisfy $\mathrm{Re}\,\lambda_i < 0$ and $\|C(x)\| < \varepsilon$.

Example Examine the stability of the system of differential equations

$$y_1' = \frac{1}{3}(y_1 - y_2)(1 - y_1 - y_2), \tag{F.5}$$

$$y_2' = y_1(2 - y_2). \tag{F.6}$$

Linearize the system, compute the eigenvalues and eigenvectors for the problem $\boldsymbol{y}' = A\boldsymbol{y}$, and confirm that the system has four fixed points in the (y_1, y_2) plane: an unstable focus $(0, 0)$, a saddle $(0, 1)$, an unstable knot $(2, 2)$, and an unstable point $(-1, 2)$ with the eigenspace spanned by the vectors whose y_2-components are zero. Based on (F.3), classify the linear stability (or instability) in the vicinity of fixed points. Figure F.2 shows the numerical solution of the system. Reproduce it by randomly selecting initial conditions $(y_1(0), y_2(0))$ near the fixed points $(\pm 1, \pm 1)$, and run the integration to $x \approx 0.5$. Use the same case to check the stability of the explicit Euler difference scheme. Such studies aid us, on a case-by-case basis, in deciding to what extent linear measures of stability can be trusted in non-linear systems.

F.2 Spurious Fixed Points

Are the fixed points of the chosen difference scheme the same as the fixed points of the differential equation it approximates? We examine the stability of the (non-linear) logistic equation

$$y' = \lambda y(1 - y), \quad y(0) = y_0,$$

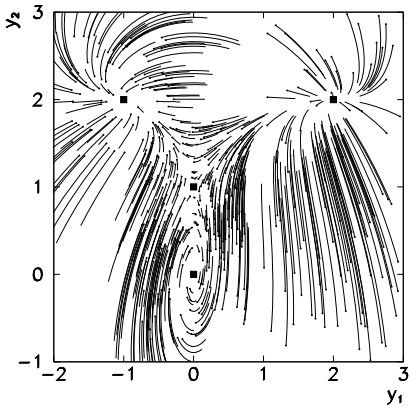


Fig. F.2 The numerical solution (flow) of the non-linear system (F.5)–(F.6), with 500 randomly chosen initial conditions (*small filled circles*), integrated to $x = 0.7$ by the adaptive RK4 method. The fixed points are denoted by *filled squares*

with $\lambda \in \mathbb{C}$ and of the corresponding difference approximations for it by using the criteria of *linear* stability [2]. The analytic solution of the equation is

$$y(x) = \frac{y_0}{y_0 + (1 - y_0)\mathrm{e}^{-\lambda x}},$$

and the stable fixed points are obviously $y^* = 0$ for $\mathrm{Re}\,\lambda < 0$ and $y^* = 1$ for $\mathrm{Re}\,\lambda > 0$. The Jacobi matrix in one dimension is just the derivative $\partial f/\partial y = \lambda - 2\lambda y$, so at the fixed points $[\partial f/\partial y](0) = \lambda$ or $[\partial f/\partial y](1) = -\lambda$. The fixed point $y^* = 0$ is linearly stable when $\mathrm{Re}\,\lambda < 0$, and the fixed point $y^* = 1$ is linearly stable if $\mathrm{Re}\,\lambda > 0$.

The corresponding explicit Euler scheme is

$$y_{n+1} = y_n + h\lambda y_n(1 - y_n) = F(y_n),$$

with the same fixed points as before, $y^* = 0$ and $y^* = 1$. But now the Jacobi derivative is $\partial F/\partial y = 1 + h\lambda - 2h\lambda y$, thus $[\partial F/\partial y](0) = 1 + h\lambda$ and $[\partial F/\partial y](1) = 1 - h\lambda$. The fixed point $y^* = 0$ corresponds to the absolute stability region which is exactly the same as in the linear case (the interior of the circle denoted by $p = 1$ in Fig. 7.3). The point $y^* = 1$ corresponds to this circle mirrored across the imaginary axis. Let us look one order higher, to the improved Euler method (7.6). When we write the difference scheme

$$y_{n+1} = y_n + hf\big(y_n + (h/2)f(y_n)\big),$$

for the logistic equation, we get

$$\begin{aligned} y_{n+1} &= y_n + h\lambda\left[y_n + \frac{h\lambda}{2}y_n(1 - y_n)\right]\left[1 - y_n - \frac{h\lambda}{2}y_n(1 - y_n)\right] \\ &= y_n + h\lambda y_n(1 - y_n)\left[1 + \frac{h\lambda}{2} - \frac{h\lambda}{2}\,y_n\right]\left[1 - \frac{h\lambda}{2}\,y_n\right]. \end{aligned}$$

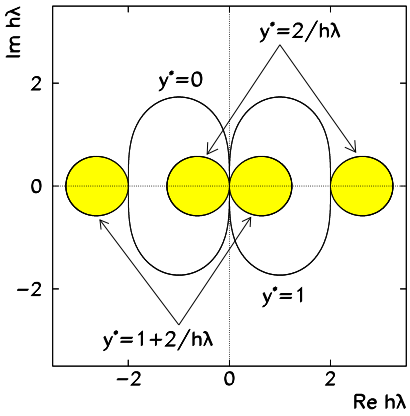


Fig. F.3 Stability regions with respect to genuine and spurious fixed points for the improved Euler method (second-order explicit RK method). See also Fig. 7.3

This expression reveals *four* fixed points: the familiar $y^* = 0$ and $y^* = 1$, as well as two new ones, $2/h\lambda$ and $1 + 2/h\lambda$! The fixed points that are not seen in the differential equation, but appear in the corresponding difference scheme, are called *spurious* or *ghost points*. Characteristically, spurious fixed points depend on the step size h. When one computes $\partial F/\partial y$, finding the stability region of different fixed points translates to a search of the set of points $h\lambda$ in the complex plane satisfying

$$\left|1 \pm h\lambda \pm \frac{(h\lambda)^2}{2}\right| < 1,$$

similar to what we have done in Sect. 7.5. The fixed point $y^* = 0$ corresponds to the interior of the shape denoted by $p = 2$ in Fig. 7.3 (and its mirror image corresponds to $y^* = 1$). To each of the spurious fixed point corresponds a pair of smaller stability regions: the complete pattern of stability regions for all four fixed points is shown in Fig. F.3.

F.3 Non-linear Stability

When we studied a non-linear system by using methods of linear stability we learned that one should not apply linear measures to non-linear systems. The stability region of a linear discrete mapping used to approximate a linear problem is not necessarily the same as the stability region of the same scheme applied to a non-linear problem. The theory of stability for non-linear systems requires us to understand the concepts of *contractivity* and *convergence*.

Definition If $\widetilde{\boldsymbol{y}}$ and $\boldsymbol{y}$ are any solutions of the system $\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y})$ corresponding to different initial conditions, the system is *contractive* if

$$\left\|\widetilde{\boldsymbol{y}}(x_2) - \boldsymbol{y}(x_2)\right\| \leq \left\|\widetilde{\boldsymbol{y}}(x_1) - \boldsymbol{y}(x_1)\right\|$$

for any pair (x_1, x_2) from the interval $a \le x_1 \le x_2 \le b$. The corresponding difference scheme is contractive if

$$\|\widetilde{\boldsymbol{y}}_{n+1} - \boldsymbol{y}_{n+1}\| \le \|\widetilde{\boldsymbol{y}}_n - \boldsymbol{y}_n\|.$$

Linear Problems Contractivity is easy to understand for the linear problem

$$\boldsymbol{y}' = A\boldsymbol{y}.$$

What is the norm of the numerical solution $\boldsymbol{y}_{n+1} = S(hA)\boldsymbol{y}_n$ if S is the stability function of the applied difference method? Since $\|\boldsymbol{y}_{n+1}\| \le \|S(hA)\|\,\|\boldsymbol{y}_n\|$, we have contractivity if $\|S(hA)\| \le 1$. For functions $S(z)$ that ensure stability throughout the complex half-plane (A-stability, Sect. 7.9), we have contractivity when, in the Euclidean norm, $\mathrm{Re}\,\langle \boldsymbol{y}, A\boldsymbol{y}\rangle \le 0$ for each $\boldsymbol{y}$ [3].

Semi-linear Problems For semi-linear systems of the form

$$\boldsymbol{y}' = A\boldsymbol{y} + \boldsymbol{g}(x, \boldsymbol{y}),$$

where $\boldsymbol{g}(x, \boldsymbol{y})$ is a small non-linear perturbation, we can resort to similar measures as in the linear case. If all eigenvalues of A satisfy $\mathrm{Re}\,\lambda_i < 0$, and if for any $\varepsilon > 0$ a $\delta > 0$ exists such that $\|\boldsymbol{g}(x, \boldsymbol{y})\| \le \varepsilon\|\boldsymbol{y}\|$ for $\|\boldsymbol{y}\| < \delta$ and $x \ge x_0$, then the origin is asymptotically stable in the Lyapunov sense. But the parameter ε must be sufficiently small. This means that in non-linear systems, one may expect absolute stability to occur only in the vicinity of stable fixed points. Contractivity is achieved when

$$\langle \boldsymbol{y}, A\boldsymbol{y}\rangle \le \mu\|\boldsymbol{y}\|^2, \qquad \big\|\boldsymbol{g}(x, \boldsymbol{y}) - \boldsymbol{g}(x, \boldsymbol{z})\big\| \le \Lambda\|\boldsymbol{y} - \boldsymbol{z}\|,$$

where μ and Λ are constants that must satisfy $\mu + \Lambda \le 0$ (with Λ small). Similar conditions apply to numerical solutions of difference schemes, e.g. the implicit Runge–Kutta methods [3].

Non-linear Problems The study of stability of "genuine" non-linear systems

$$\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y})$$

rests upon one-sided Lipschitz conditions of the form

$$\mathrm{Re}\big\langle \boldsymbol{f}(x, \boldsymbol{y}) - \boldsymbol{f}(x, \boldsymbol{z}), \boldsymbol{y} - \boldsymbol{z}\big\rangle \le \nu\|\boldsymbol{y} - \boldsymbol{z}\|^2, \tag{F.7}$$

where ν is the one-sided Lipschitz constant of $\boldsymbol{f}$ (compare (F.7) to (7.4)). When $\nu \le 0$, the distance between any two solutions of the equation $\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y})$ is a non-increasing function of x. We wish the numerical solution to possess the same property, e.g. when using implicit Runge–Kutta methods. For this purpose we define B-stability and B-convergence, which we refer to in Sect. 7.9.

Definition A method of the Runge–Kutta type is B-stable if from the contractivity condition $\mathrm{Re}\langle \boldsymbol{f}(x,\boldsymbol{y}) - \boldsymbol{f}(x,\boldsymbol{z}),\ \boldsymbol{y}-\boldsymbol{z}\rangle \le 0$ for all $h \ge 0$, we can conclude

$$\|\widetilde{\boldsymbol{y}}_{n+1} - \boldsymbol{y}_{n+1}\| \le \|\widetilde{\boldsymbol{y}}_n - \boldsymbol{y}_n\|,$$

where $\boldsymbol{y}_{n+1}$ (or $\widetilde{\boldsymbol{y}}_{n+1}$) are the numerical solutions obtained from $\boldsymbol{y}_n$ (or $\widetilde{\boldsymbol{y}}_n$) in one step. ($B$-stability implies A-stability, see [3].)

Definition A method of the Runge–Kutta type is B-convergent of order r for the non-linear problem $\boldsymbol{y}' = \boldsymbol{f}(x,\boldsymbol{y})$ satisfying the one-sided Lipschitz condition (F.7), if the global error can be estimated as

$$\left\|\boldsymbol{y}_n - \boldsymbol{y}(x_n)\right\| \le h^r \gamma(x_n - x_0, \nu) \max_j \max_x \left\|\boldsymbol{y}^{(j)}(x)\right\|$$

for $h\nu \le \alpha$, where $h = \max_i h_i$, $1 \le j \le j_{\max}$, and $a \le x \le b$. The function γ and the parameter α depend only on the method.

For stiff problems, the measures of convergence have to be adapted. Namely, if we use A-stable methods to solve large systems of stiff non-linear differential equations, some may yield unstable results. Besides, it may happen that the precision of the solution has no connection to the order of the method used. A decrease of the order of convergence occurs [4], i.e. the order of the method for a stiff problem with large λ becomes much smaller than the order for a non-stiff problem. Through B-convergence one can also introduce global error estimates that do not depend on the degree of stiffness [5]. The implicit method Radau 5 (p. 360) is B-stable and B-convergent of order 3 [3].

References

1. E. Hairer, S.P. Nørsett, G. Wanner, *Solving Ordinary Differential Equations I; Nonstiff Problems*. Springer Series in Computational Mathematics, vol. 8 (Springer, Berlin, 2000)
2. J.H.E. Cartwright, O. Piro, The dynamics of Runge–Kutta methods. Int. J. Bifur. Chaos **2**, 427 (1992)
3. E. Hairer, G. Wanner, *Solving Ordinary Differential Equations II; Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics, vol. 14 (Springer, Berlin, 2004)
4. A. Prothero, A. Robinson, On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. Math. Comput. **28**, 145 (1974)
5. R. Frank, J. Schneid, C.W. Ueberhuber, The concept of B-convergence. SIAM J. Numer. Anal. **18**, 753 (1981)

Appendix G
Construction of Symplectic Integrators ⋆

To study time evolution of dynamical systems, an operator formalism can be developed [1] that leads to a special family of integrators. We have seen in Chap. 7 that the evolution amounts to solving the system of differential equations

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y})$$

over some phase space χ, where $\boldsymbol{y}$ is a vector in this space, and $\boldsymbol{f}$ is a vector of scalar functions.[1] The solutions of this equations are the trajectories traced by $\boldsymbol{y}$ in time t, and described by the mapping $M^t(\boldsymbol{y})$. In general, we would like to learn about the dynamics of some variable (observable) of the system $\boldsymbol{\phi}(\boldsymbol{y})$, which can be very simple (for example, the "position" $\boldsymbol{\phi}(\boldsymbol{y}) = \boldsymbol{y}$). The time dynamics is defined by the *map* M, $\boldsymbol{\phi}^t(\boldsymbol{y}) = \boldsymbol{\phi}(M^t(\boldsymbol{y}))$, thus

$$\dot{\boldsymbol{\phi}} = (\boldsymbol{f} \cdot \nabla)\boldsymbol{\phi}.$$

The linear differential operator $\boldsymbol{f} \cdot \nabla = D$ is called the *dynamics generator* and represents the derivative in the direction tangential to the trajectory at the point $\boldsymbol{y}$ (Fig. G.1). The equation above can be formally integrated, yielding

$$\boldsymbol{\phi}^t(\boldsymbol{y}) = \exp(tD)\boldsymbol{\phi}(\boldsymbol{y}) = \boldsymbol{\phi}\big(M^t(\boldsymbol{y})\big), \tag{G.1}$$

where the exponential acts in the operator sense. This equation embodies the *evolution operator* or *integrator* describing the time evolution of $\boldsymbol{\phi}$.

Hamiltonian Systems Let us apply the procedure outlined above to Hamiltonian systems with the Hamiltonian function $H(\boldsymbol{p}, \boldsymbol{q})$, where $(\boldsymbol{p}, \boldsymbol{q}) \in \chi = \mathbb{R}^{2n}$. The dynamics (and the trajectory) is governed by the Hamiltonian equations

$$\dot{\boldsymbol{y}} = J^{-1}\nabla H(\boldsymbol{y}), \quad J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \ \nabla = \begin{pmatrix} \nabla_p \\ \nabla_q \end{pmatrix},$$

[1] In this appendix and in Sect. 7.12 we use the pair of variables $(t, \boldsymbol{y})$ instead of $(x, \boldsymbol{y})$, while we express $\boldsymbol{y}$ by the Hamiltonian canonical variables $\boldsymbol{p}$ and $\boldsymbol{q}$.

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9, © Springer-Verlag Berlin Heidelberg 2012

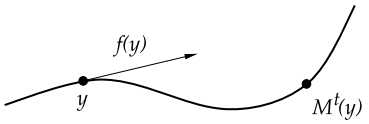


Fig. G.1 The trajectory of a dynamical system is described by the mapping $M^t(\boldsymbol{y})$ which corresponds to the solution of a system of differential equations $\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y})$

while the time evolution of an arbitrary observable $\boldsymbol{\phi}$ is given by the Liouville equation

$$\dot{\boldsymbol{\phi}} = \{\boldsymbol{\phi}, H\},$$

where $\{A, B\} = (\partial_q A)(\partial_p B) - (\partial_q B)(\partial_p A)$ is the Poisson bracket. We use the notation $D_H = \{\cdot, H\}$ to write the equation in the form $\dot{\boldsymbol{\phi}} = D_H \boldsymbol{\phi}$ and formally integrate it over an infinitesimally short time $t = \tau$:

$$\boldsymbol{\phi}^\tau(\boldsymbol{y}) = \exp(\tau D_H)\boldsymbol{\phi}(\boldsymbol{y}) = \exp\bigl(-\tau(\nabla H\, J)\nabla\bigr)\boldsymbol{\phi}(\boldsymbol{y}) = \boldsymbol{\phi}\bigl(M^\tau(\boldsymbol{y})\bigr).$$

The exponential operator in this equation is unitary in the function space $\boldsymbol{\phi} : \chi \to \mathbb{C}^n$, thus the norm $\int_\chi |\boldsymbol{\phi}|^2 \,\mathrm{d}\boldsymbol{y}$ along the solution of the system remains constant. In the case $\boldsymbol{\phi}(\boldsymbol{y}) = \boldsymbol{y} = (\boldsymbol{p}, \boldsymbol{q})$ this means that the mapping from the point $(\boldsymbol{p}, \boldsymbol{q})$ in phase space at $t = 0$ to the point $(\widetilde{\boldsymbol{p}}, \widetilde{\boldsymbol{q}})$ at a later time along the solution preserves the volume in phase space, i.e. the form $\sum \mathrm{d}\boldsymbol{p} \wedge \mathrm{d}\boldsymbol{q} = \sum \mathrm{d}\widetilde{\boldsymbol{p}} \wedge \mathrm{d}\widetilde{\boldsymbol{q}}$: the mapping is *symplectic* [2]. Moreover, it preserves the energy, $H(\boldsymbol{p}, \boldsymbol{q}) = H(\widetilde{\boldsymbol{p}}, \widetilde{\boldsymbol{q}})$.

In practice we rarely sum more than just a few terms in the exponential series

$$\boldsymbol{\phi}^\tau(\boldsymbol{y}) = \sum_{n=0}^{N} \frac{\tau^n}{n!} D_H^n \boldsymbol{\phi}(\boldsymbol{y}) + \mathcal{O}\bigl(\tau^{N+1}\bigr),$$

where $D_H^n \boldsymbol{\phi} = \{\{\ldots\{\boldsymbol{\phi}, H\}, \ldots, H\}, H\}$. The truncated series still preserves the energy and possible other constants (integrals) of motion $I_k(\boldsymbol{p}, \boldsymbol{q})$, for which $\{H, I_k\} = 0$ applies, but the truncated operator is no longer unitary and violates the symplectic nature of the dynamics. Nevertheless, for short times τ even the finite series provides a good description.

Symplectic Integrators As we have shown in Sect. 7.12, the traditional integrators (for example, Euler or Runge–Kutta methods) in general do not preserve the symplectic structure, nor the energy. In the following we chart a path to a special class of explicit symplectic integrators with which we attempt to fulfill at least one of these requirements. We focus on solving the equations of motion for Hamiltonian systems,

$$\dot{q} = \frac{\partial H}{\partial p}, \qquad \dot{p} = -\frac{\partial H}{\partial q},$$

where the Hamiltonian is separable (the kinetic energy depending only on p and the potential only on q), thus

$$H(p,q) = T(p) + V(q).$$

(For greater clarity, we discuss only the scalar case and write ∂_p, ∂_q instead of ∇_p, ∇_q.) When the equations of motion are rewritten in the form $\dot{y} = \{y, H\}$, the dynamics of the system is described by

$$y(\tau) = \exp(\tau D_H)\, y(0) = \exp\big(\tau(D_T + D_V)\big)\, y(0). \tag{G.2}$$

The exponential operators composed of partial generators $D_T = (\partial_p T)\partial_q$ and $D_V = -(\partial_q V)\partial_p$, which in general do not commute, have a particularly appealing property: they generate shifts in the canonical variables,

$$\exp(\tau D_T)\phi(q,p) = \phi\big(q + \tau(\partial_p T), p\big) = \phi\big(S_\tau(y)\big),$$
$$\exp(\tau D_V)\phi(q,p) = \phi\big(q, p - \tau(\partial_q V)\big) = \phi\big(C_\tau(y)\big),$$

which we denote as new mappings $S_\tau, C_\tau : \chi \to \chi$. (This should be familiar from quantum mechanics: the linear momentum operator $\boldsymbol{p} = -\mathrm{i}\hbar\nabla_x$ is the generator of infinitesimal translations, while the orbital angular momentum operator $\boldsymbol{L} = \boldsymbol{r} \times \boldsymbol{p}$ is the generator of infinitesimal rotations.) We approximate the exact time evolution operator by the product of the exponential operators

$$\exp\big(\tau(D_T + D_V)\big) = \prod_{i=1}^{k} \exp(c_i \tau D_T) \exp(d_i \tau D_V) + \mathcal{O}\big(\tau^{r+1}\big), \tag{G.3}$$

where c_i and d_i are real numbers, and r is an integer determining the quality of the approximation and the order of the integrator. We have thus replaced the full evolution operator by a compositum of shifts in the phase space:

$$\begin{aligned} \mathrm{e}^{\tau D_H}\phi(y) &= \mathrm{e}^{c_1\tau D_T}\mathrm{e}^{d_1\tau D_V}\cdots\mathrm{e}^{c_k\tau D_T}\mathrm{e}^{d_k\tau D_V}\phi(y) + \mathcal{O}\big(\tau^{r+1}\big) \\ &= \phi\big(C_{d_k\tau}\circ S_{c_k\tau}\circ\cdots\circ C_{d_1\tau}\circ S_{c_1\tau}(y)\big) + \mathcal{O}\big(\tau^{r+1}\big). \end{aligned}$$

By comparison to (G.1), we infer that we have thus also obtained an approximation for the true mapping $M^t(y)$ of the dynamical system,

$$M^\tau = C_{d_k\tau}\circ S_{c_k\tau}\circ\cdots\circ C_{d_1\tau}\circ S_{c_1\tau} + \mathcal{O}\big(\tau^{r+1}\big).$$

The scheme for the integrator is now at hand [3, 4]. When expression (G.3) is inserted in (G.2), we obtain a sequence of simpler mappings

$$q_i = q_{i-1} + \tau c_i \left(\frac{\partial T}{\partial p}\right)_{p=p_{i-1}}, \qquad p_i = p_{i-1} - \tau d_i \left(\frac{\partial V}{\partial q}\right)_{q=q_i}, \tag{G.4}$$

which map the initial configuration $y(0) = (p_0, q_0)$ in k steps into the final configuration $y(\tau) = (p_k, q_k)$. This sequence of mappings is symplectic, since it is composed of smaller steps which are all symplectic by themselves. We now know how to integrate the equation of motion; the main challenge is how to find k as small as possible (as few operations as possible) with the order of the integrator r as large as possible.

In principle, the procedure is straightforward: we expand the left-hand side of (G.3) in powers of τ and compare the coefficients of the corresponding powers of τ at the right (up to including order k). We end up with a system of non-linear algebraic equations for the unknowns c_i and d_i, which is easily solvable only for small k. In the simplest non-trivial case we obtain $k = 1$, $c_1 = d_1 = 1$, therefore

$$\exp(\tau D_T)\exp(\tau D_V) = \exp(\tau D_{\widetilde{H}_1}). \tag{G.5}$$

For small τ, the left-hand side of the equation represents a simple iterative scheme which requires a single reference to q and p for one step in time τ according to (G.4). The right-hand side can be computed by using the Baker–Campbell–Hausdorff (BCH) formula describing the decomposition of a product of exponential functions of two *non-commuting* operators A and B,

$$\exp(A)\exp(B) = \exp(C),$$

where

$$C = A + B + \frac{1}{2}[A, B] + \frac{1}{12}\big([A, [A, B]] + [B, [B, A]]\big) + \frac{1}{24}\big[A, [B, [B, A]]\big] + \cdots$$

and $[A, B] = AB - BA$. At the right of (G.5) we then obtain

$$D_{\widetilde{H}_1} = D_T + D_V + \frac{\tau}{2}[D_T, D_V] + \frac{\tau^2}{12}\big([D_T, [D_T, D_V]] + [D_V, [D_V, D_T]]\big) + \frac{\tau^3}{24}\big[D_T, [D_V, [D_V, D_T]]\big] + \cdots,$$

and it follows that

$$\widetilde{H}_1 = T + V + \frac{\tau}{2}\{V, T\} + \frac{\tau^2}{12}\big(\{\{T, V\}, V\} + \{\{V, T\}, T\}\big) + \frac{\tau^3}{12}\big(\{\{\{T, V\}, V\}, T\}\big) + \cdots. \tag{G.6}$$

(Note that a change in the signs occurs when passing from the usual commutators to Poisson brackets of odd orders, $[D_T, D_V] \to -\{T, V\}$, $[D_T, [D_V, [D_T, D_V]]] \to -\{\{\{T, V\}, T\}, V\}$, and so on. For details see [5].) We see that the expansion (G.6) has given us a first-order symplectic integrator preserving the Hamiltonian structure of the system, but which evolves according to a slightly different Hamiltonian operator $\widetilde{H}_1$. Only when $\tau \to 0$ do we recover the original Hamiltonian dynamics governed by $H = T + V$.

A Second-Order Integrator We obtain an explicit symplectic integrator of the second order involving two stages ($k = 2$) by using the constants

$$c_1 = c_2 = \frac{1}{2}, \qquad d_1 = 1, \qquad d_2 = 0.$$

Let us write down the formulas for this integrator explicitly and use the notation of Chap. 7: instead of τ we write h, and replace the index i by n. The solutions at time t_n are p_n and q_n. The solution at the next time $t_{n+1} = t_n + h$ is obtained by computing

$$\begin{aligned} q^* &= q_n + hc_1 \left(\frac{\partial T}{\partial p}\right)_{p=p_n}, \\ p^* &= p_n - hd_1 \left(\frac{\partial V}{\partial q}\right)_{q=q^*}, \\ q_{n+1} &= q^* + hc_2 \left(\frac{\partial T}{\partial p}\right)_{p=p^*}, \\ p_{n+1} &= p^* - hd_2 \left(\frac{\partial V}{\partial q}\right)_{q=q_{n+1}}, \end{aligned}$$

where q^* and p^* are auxiliary variables. In integrators of higher orders, there are even more such variables which form a chain-like progression to consecutive parts of an individual time step. Because $d_2 = 0$, only a single evaluation of the force $F = -\partial_q V$ is needed. This integrator corresponds to the decomposition

$$\exp\left(\frac{1}{2} h D_T\right) \exp(h D_V) \exp\left(\frac{1}{2} h D_T\right) = \exp(h D_{\widetilde{H}_2}),$$

whence it follows that

$$\widetilde{H}_2 = T + V + \frac{h^2}{12}\Big(\big\{\{T, V\}, V\big\} - \frac{1}{2}\big\{\{V, T\}, T\big\}\Big) + \mathcal{O}\big(h^4\big).$$

A Fourth-Order Integrator A very popular fourth-order symplectic integrator [6] requiring four stages ($k = 4$) has the parameters

$$c_1 = c_4 = \frac{1}{2(2 - 2^{1/3})}, \qquad c_2 = c_3 = \frac{1 - 2^{1/3}}{2(2 - 2^{1/3})}, \tag{G.7}$$

$$d_1 = d_3 = \frac{1}{2 - 2^{1/3}}, \qquad d_2 = -\frac{2^{1/3}}{2 - 2^{1/3}}, \qquad d_4 = 0, \tag{G.8}$$

and has the highest order for which the coefficients could still be determined analytically. Again, the last coefficient is $d_4 = 0$: at each time step h only three evaluations of the force $F = -\partial_q V$ are needed, one less than, for example, in the standard RK4

method calling for four evaluations. The coefficients (G.7) and (G.8) do not solve the system of equations for c_i and d_i at fourth order uniquely: they can be further optimized with respect to a certain aspect of the numerical problem, for example, for optimal conservation of energy [7].

A Sixth-Order Integrator The path to practically useful symplectic integrators also leads through the BCH formula, but the procedure of determining the coefficients c_i and d_i becomes more complicated and they can no longer be given analytically. The coefficients (in double precision) for the sixth-order integrator with eight stages ($k = 8$) are

$$
\begin{aligned}
c_1 = c_8 &= 0.392256805238780,\\
c_2 = c_7 &= 0.510043411918458,\\
c_3 = c_6 &= -0.471053385409758,\\
c_4 = c_5 &= 0.0687531682525198,\\
d_1 = d_7 &= 0.784513610477560,\\
d_2 = d_6 &= 0.235573213359357,\\
d_3 = d_5 &= -1.17767998417887,\\
d_4 &= 1.31518632068391,\\
d_8 &= 0.
\end{aligned}
$$

An eighth-order integrator requiring 16 stages is described in [8].

References

1. A.J. Dragt, J.M. Finn, Lie series and invariant functions for analytic symplectic maps. J. Math. Phys. **17**, 2215 (1976)
2. V.I. Arnol'd, *Mathematische Methoden der klassischen Mechanik* (VEB Deutscher Verlag der Wissenschaften, Berlin, 1988)
3. H. Yoshida, Recent progress in the theory and application of symplectic integrators. Celest. Mech. Dyn. Astron. **56**, 27 (1993)
4. H. Kinoshita, H. Yoshida, H. Nakai, Symplectic integrators and their application to dynamical astronomy. Celest. Mech. Dyn. Astron. **50**, 59 (1991)
5. S.R. Scuro, S.A. Chin, Forward symplectic integrators and the long-time phase error in periodic motions. Phys. Rev. E **71**, 056703 (2005)
6. E. Forest, R.D. Ruth, Fourth-order symplectic integration. Physica D **43**, 105 (1990)
7. D. Donnelly, E. Rogers, Symplectic integrators: an introduction. Am. J. Phys. **73**, 938 (2005)
8. H. Yoshida, Construction of higher order symplectic integrators. Phys. Lett. A **150**, 262 (1990)

Appendix H
Transforming PDE to Systems of ODE: Two Warnings

Partial differential equations can be transformed to a system of ordinary differential equations. This conversion reveals some new possibilities to construct difference schemes with—in principle—arbitrary precision that can be implemented in modern symbolic computing programs. We follow [1] to show that in this process there are some unforeseen obstacles.

H.1 Diffusion Equation

The first example is the one-dimensional linear diffusion equation $v_t = Dv_{xx}$ with the initial condition $v(x,0) = f(x)$ and homogeneous Dirichlet boundary conditions at $x = 0$ and $x = 1$. We approximate the time derivative at time t by the central difference,

$$\frac{\mathrm{d}v(x,t)}{\mathrm{d}t} = \frac{D}{\Delta x^2}\big(v(x-\Delta x,t) - 2v(x,t) + v(x+\Delta x,t)\big) + \mathcal{O}\big(\Delta x^2\big),$$

and do this at each $x_j = j\Delta x$. We obtain a system of ordinary differential equations, which can be written in matrix form

$$\frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} = A\boldsymbol{u}(t) + \boldsymbol{b}(t). \tag{H.1}$$

Here $\boldsymbol{u}(t) = (u_1, u_2, \ldots, u_{N-1})^{\mathrm{T}}$ is the solution vector, while the boundary conditions are contained in the vector $\boldsymbol{b}(t) = (D/\Delta x^2)(u_0(t), 0, \ldots, 0, u_N(t))^{\mathrm{T}}$. The matrix of the system is

$$A = \frac{D}{\Delta x^2}\begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix}.$$

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics, DOI 10.1007/978-3-642-32478-9, © Springer-Verlag Berlin Heidelberg 2012

Table H.1 Padé approximations $[L/M]$ of order (L, M) of the function $\exp(X)$, and their leading error terms. If X is a matrix, the denominators of expressions $[L/M]$ should be understood in the sense of matrix inverses: we write $\boldsymbol{u}(t+\Delta t) = (A/B)\boldsymbol{u}(t)$, while we actually solve the system $B\boldsymbol{u}(t+\Delta t) = A\boldsymbol{u}(t)$

(L, M)	$[L/M](X)$	Error
$(0, 1)$	$1/(1-X)$	$-\frac{1}{2}X^2$
$(1, 0)$	$1+X$	$\frac{1}{2}X^2$
$(0, 2)$	$1/(1-X+\frac{1}{2}X^2)$	$\frac{1}{6}X^3$
$(1, 1)$	$(1+\frac{1}{2}X)/(1-\frac{1}{2}X)$	$-\frac{1}{12}X^3$
$(2, 0)$	$1+X+\frac{1}{2}X^2$	$\frac{1}{6}X^3$
$(1, 2)$	$(1+\frac{1}{3}X)/(1-\frac{2}{3}X+\frac{1}{6}X^2)$	$\frac{1}{72}X^4$
$(2, 1)$	$(1+\frac{2}{3}X+\frac{1}{6}X^2)/(1-\frac{1}{3}X)$	$-\frac{1}{72}X^4$
$(2, 2)$	$(1+\frac{1}{2}X+\frac{1}{12}X^2)/(1-\frac{1}{2}X+\frac{1}{12}X^2)$	$\frac{1}{720}X^5$

If the boundary conditions $\boldsymbol{b}$ do not depend on time, the solution of the system (H.1) with the initial condition $\boldsymbol{u}(0) = (f_1, f_2, \ldots, f_{N-1})^{\mathrm{T}} = \boldsymbol{f}$ has the form $\boldsymbol{u}(t) = -A^{-1}\boldsymbol{b} + \exp(tA)(\boldsymbol{f} + A^{-1}\boldsymbol{b})$. The solutions at times t and $t+\Delta t$ are then related by the evolution equation

$$\boldsymbol{u}(t+\Delta t) = -A^{-1}\boldsymbol{b} + \exp(\Delta t A)\big(\boldsymbol{u}(t) + A^{-1}\boldsymbol{b}\big),$$

which becomes even simpler in the case of homogeneous Dirichlet boundary conditions ($\boldsymbol{b} = \boldsymbol{0}$).

We can expand the exponential function with a matrix argument in the Taylor series $\exp(\Delta t A) = I + \Delta t A + \frac{1}{2}\Delta t^2 A^2 + \cdots$ or use a Padé approximation. The latter can be used to approximate the exponential as a ratio of two polynomials (rational approximation)

$$\exp(X) = \underbrace{\frac{1 + p_1 X + p_2 X^2 + \cdots + p_L X^L}{1 + q_1 X + q_2 X^2 + \cdots + q_M X^M}}_{[L/M](X)} + c_{L+M+1} X^{L+M+1} + \mathcal{O}\big(X^{L+M+2}\big),$$

where p_l, q_m, and c are real coefficients that can be determined uniquely. The function $[L/M](X)$ is the Padé approximation of order (L, M) of the function $\exp(X)$ with the leading error term $c_{L+M+1}X^{L+M+1}$ (see also Sect. 1.2.2). Table H.1 shows a few lowest orders of the Padé approximations.

In difference schemes for partial differential equations, X are matrices: how one should interpret "matrix division" in the expression for $\exp(X)$, will become clear instantly. The Padé approximation $[1/0]$ means

$$\boldsymbol{u}(t+\Delta t) = (I + \Delta t A)\,\boldsymbol{u}(t), \tag{H.2}$$

which is nothing but the well-known explicit FTCS scheme. In the case of implicit schemes where $M \geq 1$, the denominators in Table H.1 should be understood in the sense of matrix inverses. The $[0/1]$ approximation is the implicit BTCS scheme,

$$(I - \Delta t A)\boldsymbol{u}(t+\Delta t) = \boldsymbol{u}(t),$$

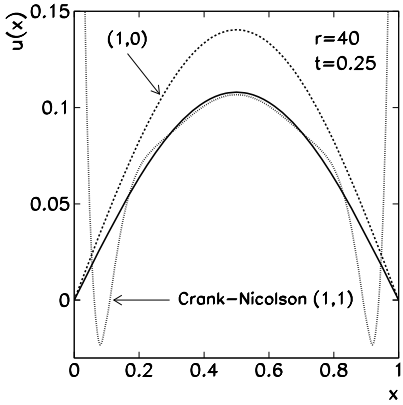


Fig. H.1 Numerical solution of the diffusion equation $v_t = Dv_{xx}$ on $0 \le x \le 1$ with the initial condition $v(x,0) = 1$ and boundary conditions $v(0,t) = v(1,t) = 0$. (Example adapted from [1].) Shown are the analytic solution (*full curve*), the numerical solution ($D = 1$, $\Delta x = \Delta t = 0.025, r = 40$) by the Padé approximation of order $(1,0)$ (see (H.2)), and the solution of order $(1,1)$ (Crank–Nicolson, (H.3)), in which typical oscillations appear in the vicinity of the discontinuity between the initial and boundary condition. For the example shown in this figure we have set $\Delta t = 0.025 > \Delta x/\pi = 0.008$, in disagreement with the requirement (H.4)

while the $[1/1]$ approximation is the Crank–Nicolson scheme,

$$\left(I - \frac{1}{2}\Delta t A\right)\boldsymbol{u}(t + \Delta t) = \left(I + \frac{1}{2}\Delta t A\right)\boldsymbol{u}(t). \tag{H.3}$$

The compact notation hides a trap shaking our conviction that from the viewpoint of stability (the value of the parameter $r = D\Delta t/\Delta x^2$), we may rely on the Crank–Nicolson scheme without second thoughts. The difference scheme with the approximation of order (L, M) maps the initial condition $\boldsymbol{u}(0)$ to $\boldsymbol{u}(t_n) = ([L/M](\Delta t A))^n \boldsymbol{u}(0)$ at time t_n. The non-degenerate eigenvalues λ_s of the matrix A (A.7) correspond to the linearly independent eigenvectors $\boldsymbol{v}_s$ (A.8), which we can use to expand the initial approximation as $\boldsymbol{u}(0) = \sum_s c_s \boldsymbol{v}_s$. Since $A\boldsymbol{v}_s = \lambda_s \boldsymbol{v}_s$, we also have $F(A)\boldsymbol{v}_s = F(\lambda_s)\boldsymbol{v}_s$, thus

$$\boldsymbol{u}(t_n) = \sum_{s=1}^{N-1} c_s \big([L/M](\Delta t \lambda_s)\big)^n \boldsymbol{v}_s.$$

The solution remains stable, $\lim_{n\to\infty} \boldsymbol{u}(t_n) = \boldsymbol{0}$, if $|[L/M](\Delta t\lambda_s)| < 1$ for each s. This is clearly true for the Crank–Nicolson scheme, since $\lambda_s < 0$ for all s and the growth factor is $\mu_s = [1/1](\Delta t\lambda_s) = (1 + \Delta t\lambda_s)/(1 - \Delta t\lambda_s) < 1$. Yet $[1/1](\Delta t\lambda_s)$ may come close to the value -1 for large $\Delta t\lambda_s = -4r \sin^2(s\pi/2N)$, i.e. when r is large and $s\pi/2N \approx \pi/2$ (both N and s large). This can occur when there are discontinuities in the initial condition or between the initial and the boundary condition, and may cause oscillations of the solution in the vicinity of such discontinuities. Figure H.1 shows an example.

Oscillations originate in the "high-frequency" terms $c_{N-1}\boldsymbol{v}_{N-1}, c_{N-2}\boldsymbol{v}_{N-2}, \ldots$, which become $c_{N-1}\mu_{N-1}^n\boldsymbol{v}_{N-1}, c_{N-2}\mu_{N-2}^n\boldsymbol{v}_{N-2}, \ldots$ in the nth time step. These terms alternate in sign when n increases, and are only slowly damped. Still, it can be shown [1] that such instabilities can be harnessed with a sufficiently small time step

$$\Delta t \lesssim \Delta x/\pi. \tag{H.4}$$

H.2 Advection Equation

As an example of hyperbolic equations, we discuss $v_t + cv_x = 0$ with the parameter $c > 0$ at $x \geq 0$, the initial condition $v(x,0) = f(x)$, and the boundary condition $v(0,t) = b(t)$. According to the sign of c it is sensible to discretize the spatial derivative as $v_x(x,t) \approx (v(x,t) - v(x-\Delta x,t))/\Delta x$. As in the preceding section, we establish an ordinary differential equation at each point $x_j = j\Delta x$,

$$\frac{\mathrm{d}v(x,t)}{\mathrm{d}t} = -\frac{c}{\Delta x}\big(v(x,t) - v(x-\Delta x,t)\big) + \mathcal{O}(\Delta x),$$

so that the whole domain can be covered by the matrix system of the form

$$\frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} = -cB\boldsymbol{u}(t) + c\boldsymbol{b}(t),$$

where $\boldsymbol{u}(t) = (u_1, u_2, \ldots, u_N)^{\mathrm{T}}$ is the solution vector, the boundary condition is contained in the vector $\boldsymbol{b}(t) = (1/\Delta x)(b(t), 0, \ldots, 0)^{\mathrm{T}}$, and B is bidiagonal,

$$B = \frac{1}{\Delta x}\begin{pmatrix} 1 & 0 & & & \\ -1 & 1 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 1 & 0 \\ & & & -1 & 1 \end{pmatrix}.$$

The solution of this system is $\boldsymbol{u}(t) = B^{-1}\boldsymbol{b}(t) + \exp(-c\Delta t B)(\boldsymbol{f} - B^{-1}\boldsymbol{b}(t))$. In subsequent time steps we therefore obtain

$$\boldsymbol{u}(t+\Delta t) = B^{-1}\boldsymbol{b}(t) + \exp(-c\Delta t B)\big(\boldsymbol{u}(t) - B^{-1}\boldsymbol{b}(t)\big).$$

The exponential function $\exp(-c\Delta t B)$ can now be replaced by a Padé approximation. For example, the order $(0,1)$ approximation represents the implicit scheme $(I + c\Delta t B)\boldsymbol{u}(t+\Delta t) = \boldsymbol{u}(t) + c\Delta t\boldsymbol{b}(t)$, which can be transformed into explicit form. At the points x_j, $j = 1, 2, \ldots, N$, the solution is

$$u_j^{n+1} = \frac{1}{1+cR}\big(cRu_{j-1}^{n+1} + u_j^n\big),$$

where $R = \Delta t/\Delta x$ (check this as an exercise). At order (1, 1), one again retrieves the Crank–Nicolson scheme. We may also increase the precision of the schemes by using a three-point difference approximation for the spatial derivative,

$$\frac{\mathrm{d}v(x,t)}{\mathrm{d}t} = -\frac{c}{2\Delta x}\big(3v(x,t) - 4v(x-\Delta x,t) + v(x-2\Delta x,t)\big) + \mathcal{O}\big(\Delta x^2\big).$$

In this case, the matrix system becomes

$$\frac{\mathrm{d}\boldsymbol{u}(t)}{\mathrm{d}t} = -\frac{1}{2}cB\boldsymbol{u}(t) + \frac{1}{2}c\boldsymbol{b}(t), \tag{H.5}$$

the boundary condition is contained in $\boldsymbol{b}(t) = (1/\Delta x)(2b(t), -b(t), 0, \ldots, 0)^{\mathrm{T}}$, and the matrix B is

$$B = \frac{1}{\Delta x}\begin{pmatrix} 2 & & & & & & & \\ -4 & 3 & & & & & & \\ 1 & -4 & 3 & & & & & \\ & 1 & -4 & 3 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & 1 & -4 & 3 & & \\ & & & & 1 & -4 & 3 & \\ & & & & & 1 & -4 & 3 \end{pmatrix}.$$

The solution of this system is $\boldsymbol{u}(t) = B^{-1}\boldsymbol{b}(t) + \exp(-\frac{1}{2}c\Delta t B)(\boldsymbol{f} - B^{-1}\boldsymbol{b}(t))$. In subsequent time steps we obtain

$$\boldsymbol{u}(t+\Delta t) = B^{-1}\boldsymbol{b}(t) + \exp\left(-\frac{1}{2}c\Delta t B\right)\big(\boldsymbol{u}(t) - B^{-1}\boldsymbol{b}(t)\big).$$

The (0, 1) Padé approximation results in the stable implicit scheme

$$\left(I + \frac{1}{2}c\Delta t B\right)\boldsymbol{u}(t+\Delta t) - \frac{1}{2}c\Delta t\boldsymbol{b}(t+\Delta t) = \boldsymbol{u}(t),$$

which can also be rewritten in explicit form.

An unpleasant surprise is in store at the very end [1]. If we use the symmetric difference for the spatial derivative,

$$\frac{\partial v(x,t)}{\partial x} \approx \frac{v(x+\Delta x,t) - v(x-\Delta x,t)}{2\Delta x}, \tag{H.6}$$

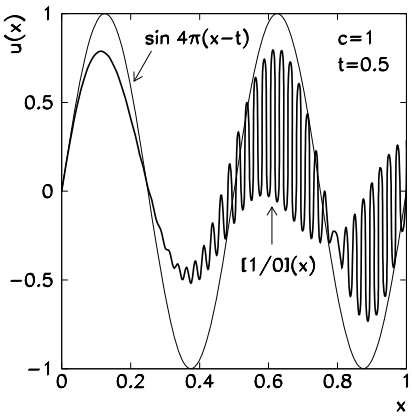


Fig. H.2 Solution of $v_t + cv_x = 0$ ($c = 1$) on $0 \le x \le 1$ with the initial condition $v(x,0) = \sin 4\pi x$, at $t = 0.5$. Shown are the analytic solution $v(x,t) = \sin 4\pi(x-t)$ and the numerical solution ($\Delta x = 0.0125$, $\Delta t = 0.025$) in the $(1,0)$ Padé approximation. The symmetric approximation for v_x (see (H.6)) causes strong oscillations

we obtain the matrix system of the same form as before (see (H.5)), with the matrix

$$B = \frac{1}{\Delta x}\begin{pmatrix} 0 & 1 & & & & & \\ -1 & 0 & 1 & & & & \\ & -1 & 0 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 0 & 1 & \\ & & & & -1 & 0 & 1 \\ & & & & & -1 & 0 \end{pmatrix}$$

and vector $\boldsymbol{b}(t) = (1/\Delta x)(b(t), 0, 0, \ldots, -u_{N+1}(t))^{\mathrm{T}}$ containing the boundary condition. Due to the symmetric difference the eigenvalues of B are complex. The solution $\boldsymbol{u}(t)$ therefore includes oscillatory terms that completely spoil the solution (Fig. H.2).

References

1. G.D. Smith, *Numerical Solution of Partial Differential Equations* (Oxford University Press, Oxford, 2003)

Appendix I
Numerical Libraries, Auxiliary Tools, and Languages

I.1 Important Numerical Libraries

A detailed explanation of individual methods and algorithms is neither the only nor the main purpose of this textbook: we are primarily interested in the mathematical and physical background of a problem rather than the details of the numerical method used to solve it. In such cases we build the skeleton of the program code, but to efficiently and reliably manage the core of the computations, we resort to specialized collection of numerical routines called *libraries*.

Linear Algebra Packages The routines for the manipulation of vectors and matrices (for example, to compute scalar product of vectors, multiply matrices, solve systems of equations, $Ax = b$, or eigenvalue problems, $Ax = \lambda x$) should never be coded by ourselves. Almost without exception our program, even if literally adopted from a textbook on numerical methods, will be inferior to a carefully optimized algorithm from a dedicated library.

The BLAS (Basic Linear Algebra Subprograms) library [1, 2] contains basic routines for vector and matrix operations used by more general libraries. At level 1 of BLAS we find routines for scalar, vector, and vector-vector operations; level 2 contains routines for matrix-vector operations; level 3 allows for matrix-matrix manipulations.

The basic building blocks of BLAS are used by higher-level libraries, the most famous of which is LAPACK (Linear Algebra PACKage) [3]. It contains a collection of subprograms to solve systems of linear equations, eigenvalue problems, and problems involving singular value decomposition. Most of the routines have been prepared in real and complex versions (vectors and matrices), in single- and double-precision floating-point arithmetic. The library is well tuned to dense and banded matrices, while it is less appropriate for use with general sparse matrices of high ranks. To solve matrix problems with dense and banded matrices on multiprocessor systems (clusters) with distributed memory, a scalable version SCALAPACK is available [4]. The same family of libraries also encompasses packages for the solution of eigenvalue problems with very large sparse matrices ARPACK [5],

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9,

for direct solution of sparse systems (CAPSS, MFACT), as well as preconditioning routines used in iterative methods for large sparse systems (ParPre). See also [6].

Libraries for Specific Processor Architectures The basic versions of LAPACK and BLAS libraries are intended for use with the Fortran77 programming language, but versions for Fortran95, C/C++, and Java also exist. They are incorporated in numerous distributions of Linux/UNIX systems. If we wish faster libraries, they need to be tuned to the individual processor architecture. For the BLAS library and a subset of routines from the LAPACK library, this can be done by the ATLAS (Automatically Tuned Linear Algebra Software) package [7]. This tool compiles the libraries and adapts them to the chosen computer platform. Precompiled optimized codes are readily available for most processors.

Perhaps the most widely spread and powerful among them is the Intel package MKL (Intel Math Kernel Library) [8] offering highly optimized mathematical routines for a broad spectrum of scientific, engineering, and financial applications with an interface to Fortran and C. It contains BLAS/LAPACK, SCALAPACK, and LINPACK, as well as the routines for direct and iterative solution of systems with sparse matrices. It performs particularly well in algorithms for direct solution of systems with large symmetric sparse matrices (of sizes 10000×10000 or more) [9]. MKL also includes a set of (up to seven-dimensional) Fourier transformations for digital signal analysis, image processing, and solving partial differential equations. Moreover, the MKL package contains well tuned vector implementations of computationally intensive elementary mathematical functions (above all, trigonometric and hyperbolic, where MKL boasts several times larger speeds than those attained in standard versions). The MKL package is available for Windows, Linux, and Mac OS operating systems.

For AMD processors we use ACML (AMD Core Math Library) [10]. In addition to the optimized BLAS/LAPACK routines, this package offers a set of routines for fast Fourier transformations and random number generation. On alpha systems (portable to other platforms) by HP (Compaq) the following packages are available: MLIB (HP Mathematical Software Library) [11], CPML (Compaq Portable Math Library), and CXML (Compaq Extended Math Library). Users of Apple computers may reach for the Velocity Engine libraries [12] that contain tools for digital processing of signals and images, in addition to LAPACK and BLAS.

Special Packages for Fourier Transformation The "working horse" for the discrete Fourier transformation (especially for very long data arrays) is the fast Fourier transform FFTW [13] (Fastest Fourier Transform in the West). It is written in C and is well portable between different architectures and operating systems. (In one form or another, FFTW is implemented in most libraries for specific architectures mentioned above.) One of the advantages of FFTW is the capability to adapt to the hardware used, allowing for improvement in speed (see Fig. 4.7). The method is of order $\mathcal{O}(N \log N)$ for any N, in contrast to numerous other implementations that are restricted to a subset of sizes N (e.g. to powers of two, $N = 2^n$) or that become

of order $\mathcal{O}(N^2)$ for specific values of N (e.g. when N is prime). While the majority of competitive FFT libraries is limited to one-, two-, and three-dimensional transformations, FFTW can be applied in arbitrary dimensions.

Packages for Boundary-Value Problems Two modern advanced packages for solving Sturm–Liouville boundary-value problems for ordinary differential equations involving eigenvalues and eigenfunctions are SLEIGN2 [14] and the cluster of subroutines D02KAF, D02KDF, and D02KEF from the extensive and much more general library NAG (see description below). These packages use the shooting method based on the Prüfer transformation (see Sect. 8.7.2) and can be applied to regular and singular Sturm–Liouville problems with separated or unseparated boundary conditions, as well as for the approximation of the continuous spectrum in the case of singular problems. The core ideas are described in [15]. Also available are the SLEDGE [16] and SL02F [17] packages based on the Pruess method (see Sect. 8.7.3).

Large General Libraries The collections of routines mentioned above are specialized and highly optimized for a particular kind of numerical problems or algorithms. We therefore use them with a very specific purpose or goal. For everyday use, much more general and extensive libraries are available that can be applied to a variety of problems.

The longest tradition among them is claimed by the NAG (Numerical Analysis Group) corpus [18]. The NAG C version is currently the largest existing commercial collection of mathematical and statistical algorithms and corresponding routines (more than 1000) for programming in C/C++ languages. In the mathematical part, it includes subprograms from the area of optimization and minimization; ordinary and partial differential equations; finding zeros of non-linear equations; solving dense, banded, and sparse matrix systems and corresponding eigenvalue problems; special functions; and approximation and interpolation. In the statistical part, it offers routines for random number generation; correlation and regression analysis; multivariate methods; variance analysis; as well as time series analysis. Versions for Fortran77 and Fortran95 also exist. Special branches of the collection, NAG SMP (Symmetric Multi-Processor) and NAG PARALLEL, predominantly contain routines for matrix-vector algebra and FFT optimized for multi-processor and distributed-memory platforms. These libraries are available for all relevant processor architectures and operating systems; an interface to MATLAB is also available (NAG TOOLBOX for MATLAB).

A similar collection, NUMERICAL RECIPES (NR) [19], is a practical tool for everyday numerical computations that is not extremely intensive. For decades, physicists considered the NR library (in versions for Pascal, Fortran77, Fortran90, and C) to be the first choice for numerical work, while the current (third) edition for C++ (NR3E) became commercial, with relatively severe copyright restrictions in academia. This may be one of the reasons that especially users of Linux/UNIX systems prefer the open-code GSL (GNU Scientific Library) [20]. We recommend it for general use in the context of this textbook. The NR3E collection also offers an interface to MATLAB.

A project rapidly ascending to its zenith is BOOST [21], which is a kind of a library of libraries. It contains a large collection of algorithms and tools for general structured and object-oriented programming, numerous data types, structures, classes, mathematical routines, and input-output tools. Some components of BOOST are gradually being integrated in standard C++ libraries.

I.2 Basics of Program Compilation

On standard Linux systems, programs are compiled by GCC (the *GNU Compiler Collection*) encompassing the compilers for the languages C, C++, Java, and Fortran. The languages in C are compiled by `gcc`, while for C++ we use `c++` or `g++`. The compiler for Java is `javac`, and for Fortran it is `gfortran` (also `f95`, on older systems `fort77` or `f77`). For example, a simple program in C++ without using libraries can be compiled and executed like this:

```
c++ PROG.cc -o PROG
./PROG
```

Libraries are linked to the main code during compilation. A library that exists on the system in the file `libALGEBRA.a` (static version) or `libALGEBRA.so` (dynamic version) residing in `/lib` or `/usr/lib` directories (these are always looked up first by the compiler) can be incorporated like this:

```
c++ [options] PROG.cc -lALGEBRA -o PROG
```

The compiler accepts numerous `[options]`. The most useful are:

`-o`	output file (otherwise `a.out`)
`-O`	basic speed optimization (higher levels are `-O2`, `-O3`)
`-g`	allows a debugger to be used (e.g. `gdb`)
`-pg`	generates profile information code for `gprof` (Appendix J)
`-Wall`	include all warnings (that are not syntax errors)
`-pedantic`	more pedantic warnings
`-ansi`	enforce ANSI standard for C
`-I`	include `*.h` files not in the standard search path, for example: `-I/home/users/my/path header.h`
`-L`	include `*.a` or `*.so` libraries not in the standard search path, for example: `-L/home/users/my/path -llibrary`

To get an impression, have a glance at the help pages (`man gcc` or `info gcc`)! This multi-thousand line documentation is not just a set of instructions on how to use the compiler: it also contains rich information on C and C++ programming languages, operating systems, and computer architectures. As was the case with numerical libraries, compilers exist that are tuned to a particular architecture. Excellent Intel compilers for C++ and Fortran [22] are available for the operating systems Windows, Linux (free for non-commercial use), and Mac OS.

I.3 Using Libraries in C/C++ and Fortran

Our code can make use of almost any library, regardless of the language in which it has been written. As an example, we describe a typical use of libraries to solve the system of equations $Ax = b$ by LU decomposition. First we show the solution with the GSL library in C or C++, then we do the same by including a Fortran library LAPACK in a program written in C or C++. Finally we give a possible solution with a Fortran library included in a program written in Fortran95 (which is the current standard version).

I.3.1 Solving Systems of Equations $Ax = b$ by Using the GSL *Library*

```
#include <gsl/gsl_linalg.h>

const int N = 256;
gsl_vector *x, *b;
gsl_matrix *A;
gsl_permutation *p;
A = gsl_matrix_alloc(N, N);
x = gsl_vector_alloc(N);
b = gsl_vector_alloc(N);

for (int i = 0; i < N; i++) {
  gsl_vector_set(b, i, ...);
  for (int j = 0; j < N; j++) gsl_matrix_set(A, i, j, ...);
}
int s;
p = gsl_permutation_alloc(N);
gsl_linalg_LU_decomp(A, p, &s);
gsl_linalg_LU_solve(A, p, b, x);
for (i = 0; i < N; i++)
  cout << gsl_vector_get(x, i) << endl;

gsl_permutation_free(p);
gsl_matrix_free(A); gsl_vector_free(x); gsl_vector_free(b);
```

This program (`Axb_GSL.cc`) is compiled and executed as follows:

```
c++ Axb_GSL.cc -lgsl -lblas -o Axb_GSL
./Axb_GSL
```

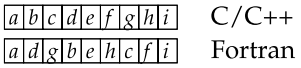


Fig. I.1 Storage of a 3×3 matrix in computer memory in the case of C/C++ languages (row-major mode) and Fortran (column-major mode)

I.3.2 Solving the System $Ax = b$ in C/C++ Language and Fortran Libraries

When we call routines written in Fortran from programs written in C/C++, we must realize that multi-dimensional data arrays are organized differently in memory. For example, a 3×3 matrix,

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix},$$

is stored in *column-major mode* (i.e. as $adg\text{-}beh\text{-}cfi$) in Fortran, while it is stored in *row-major mode* (i.e. in rows $abc\text{-}def\text{-}ghi$) in C/C++, as shown in Fig. I.1. When calling a library routine, the matrix therefore needs to be transposed. Moreover, there is a difference in indexing: Fortran indices of one-dimensional arrays (and individual components of multi-dimensional arrays) start at 1, while indices in C/C++ start at 0.

Before the beginning of the main program in C++ the subroutine from the Fortran library needs to be announced by the `extern "C"` declaration. The name of the subroutine in the declaration should be written in lower-case letters and an underscore (_) attached to it, so instead of `DGESV(...)` we write `dgesv_(...)`

```
extern "C" {
  void dgesv_(int *, int *, double *, int *, int *, double *,
              int *, int *);
}
```

(In the C language the declaration `extern "C"` is not necessary.) The lower-case name with the underscore is the one we use to actually invoke the subroutine:

```
const int N = 256;

int i, j, k, CN, NRHS, LDAF, INFO;
int *IPIV  = new int[N];
double  *B = new double[N];
double **A = new double* [N];
for (i = 0; i < N; i++) A[i] = new double[N];

double *AF = new double[N*N];
for (i = 0; i < N; i++) {
  for(j = 0; j < N; j++) {
    A[i][j] = ...              // matrix A[i][j] ("row-major")
    AF[j+N*i] = A[j][i];       // transposition ("column-major")
  }
}
```

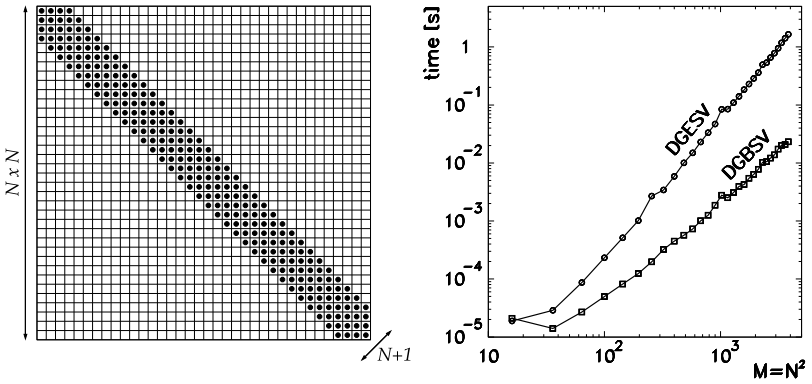


Fig. I.2 [*Left*] Typical form of a $M \times M$ banded matrix (where $M = N^2$) encountered e.g. in solving two-dimensional partial differential equations or boundary-value problems. Non-zero elements are present on the main diagonal and N subdiagonals, or there may be gaps with zeros between the most distant subdiagonals. [*Right*] Execution times for the solution of a banded system $Ax = b$ involving A and b with random elements. We have used the routines DGESV (classical LU decomposition) and DGBSV (algorithm optimized for banded matrices) from the LAPACK library. The computation time was measured by the gettimeofday() method from Appendix J

```
for (i = 0; i < N; i++) B[i] = ...
                                  // right-hand side of Ax = b
CN = LDAF = N;
NRHS = 1;
dgesv_(&CN, &NRHS, AF, &LDAF, IPIV, B, &CN, &INFO);
                                  // result is B
for (i = 0; i < N; i++) delete [] A[i];
delete [] AF;
delete [] B;
```

This program (`Axb_dgesv.cc`) is compiled and executed as follows:

```
c++ Axb_dgesv.cc -llapack -lblas -lg2c -o Axb_dgesv
./Axb_dgesv
```

On some systems `-lblas` and `-lg2c` are not necessary since they are already included in `-llapack`, or else various (dynamically allocated) libraries call each other automatically during the execution of the compiled and linked program.

The routines that are the most appropriate for a given task (for example, solving the system $Ax = b$) should always be chosen carefully. We must weigh and confront the analytic structure of the problem, the required precision of the results, and anticipated execution times (numerical costs). Figure I.2 (left) shows a typical banded matrix for a system of equations that frequently occurs in numerical solution of ordinary and partial differential equations. Such systems, especially those with large matrices, should always be solved by algorithms optimized for banded matrices. A careless choice of an unspecialized algorithm may imply execution times which are orders of magnitude longer (see Fig. I.2 (right)); in addition, it may lead to larger numerical errors.

I.3.3 Solving the System $Ax = b$ in Fortran95 by Using a Fortran77 Library

```
integer N, i, j, CN, NRHS, LDAF, INFO
integer, dimension(:), allocatable :: IPIV
real*8, dimension(:), allocatable :: B
real*8, dimension(:,:), allocatable :: A

N = 256
allocate(A(N,N), B(N), IPIV(N))

do i = 1, N
  B(i) = ...
  do j = 1, N
    A(i, j) = ...
  enddo
enddo

CN = N
LDAF = N
NRHS = 1
call DGESV(CN, NRHS, A, LDAF, IPIV, B, CN, INFO)

deallocate(A, B, IPIV)
```

This program (`Axb_dgesv.f`) can be compiled and executed as follows:

```
f95 Axb_dgesv.f -llapack -o Axb_dgesv
./Axb_dgesv
```

I.4 Auxiliary Tools

MATLAB and Octave The core of the commercial program package MATLAB (MATrix LABoratory) [23] is the high-level programming language adapted to technical use, and a powerful interactive environment for numerical computations, analysis and visualization of data, and algorithm development. MATLAB is used by scientists and engineers in innumerable applications, e.g. in designing static constructions, solving problems with partial differential equations, control systems, signal and image processing, and solving biochemical problems. With the supplementary *add-on toolboxes* the MATLAB environment can be adapted to more specific classes of problems. Due to its high-level language interface, MATLAB allows us to solve numerous problems faster than by programming in classical programming languages. It is also possible to merge the code written in MATLAB with the code written in some other language. The MATLAB Compiler can issue an executable code or library that can be included in an external program as explained in Sect. I.3. The reverse direction is also possible: external libraries may be used in MATLAB.

A skinnier free version of MATLAB (lacking the additional power of the toolboxes) is OCTAVE [24].

Mathematica A commercial tool with a two-decade tradition, MATHEMATICA [25], is comparable to MATLAB in terms of utility and flexibility. Yet these tools tend to emphasize various mathematical physics or numerical aspects very differently, and choosing among them is largely a matter of taste. MATLAB is more suited to numerical computations, especially matrix-vector algebra (for which it was originally devised), as well as for manipulation and insightful visualization of complex data sets. MATHEMATICA is more powerful in symbolic computing, in solving all kinds of linear and non-linear equations, as well as symbolic differentiation and integration. An interesting comparison of MATLAB, MATHEMATICA, and MAPLE is given in [26–28].

Graphics Tools General packages like MATLAB, OCTAVE, or MATHEMATICA already come with graphical tools built in. But good pictures should be created by using dedicated plotting programs. For fast everyday use one may recommend GNUPLOT [29]. The ROOT package [30] is much more than just a plotting program: it is a powerful interactive environment for object-oriented data analysis. ROOT is rooted in the C++ language and is a descendant of the famous Fortran-based tool PAW [31], which retains some popularity. Lovely graphics can be formed by the program grace (or xmgrace) [32]. All packages mentioned above are freely accessible. A powerful commercial tool for data analysis and plotting is ORIGIN [33].

Pedagogical Remark The major software packages mentioned in this section (some of them with similar functionalities are listed in[1]) can hardly be called "auxiliary tools". We are dealing with such an extensive and complete collections of algorithms, methods, and visualization options that an average user cannot possibly fully understand and exploit them. Their purpose is the quick solution of daily scientific and engineering problems, and as such we heartily recommend them! On the other hand, physicists encounter numerous non-daily tasks that require particular attention on how to formulate the problem, choose among the solution options, and control the quality of the computation (stability, convergence). Therefore, we should always ponder between the obvious appeal of the ready-to-use tool, and the prospect of perhaps learning more by a more dedicated engagement.

[1] MAPLE (http://www.maplesoft.com) is a powerful tool for mathematical modeling; SAGE (http://www.sagemath.org) is a free alternative to commercial packages; SCILAB (http://www.scilab.org) is another powerful free environment for scientific work and technical applications; MAGMA (http://magma.maths.usyd.edu.au/magma) is a package for symbolic solution of problems from algebra, number theory, geometry, and combinatorics. The tools enumerated here are available for Windows, Mac OS, and Linux.

I.5 Choosing the Programming Language

Fortran More than half a century has passed since its first steps, but the FORTRAN (The IBM FORmula TRAnslating System) language in its newer versions Fortran77 and Fortran95 remains *by far* the most widely spread and used language for numerically intensive computing applications, such as: study of fluid dynamics; weather predictions; finite element method analyses in civil engineering; theoretical physical chemistry (ab initio calculations of molecular orbitals); and simulations of nuclear explosions.

In Fortran, basic scientific and engineering ideas can be expressed very naturally and directly. Excellent compilers exist, coupled to an exceptionally rich assortment of libraries and packages. In Fortran95, the latest standard version, most of the constructs offered by C++ and Java can be realized, including pointers, dynamical memory allocation, and classes, with the exception of inheritance (relating general and particular properties of classes) and dynamical polymorphism (using equal names for abstract referencing to families of data types or routines). Further explanations can be found in [34, 35].

Apart from the Intel compiler mentioned above [22], the standard Linux/UNIX compilers are `gfortran` [36], which is a component of the GNU Compiler Collection `gcc`, and `g95` [37]. Important commercial versions of compilers for high-performance numerical computing (also for Windows and Mac OS) are offered by Absoft, Lahey, and NAG. More information can be found on the web.

C/C++ The C and C++ languages are closely related. The syntactically and semantically relatively simple language C [38] was developed in 1972 to cater to the needs of low-level programming, allowing for direct access to memory and other hardware components. A code written in C is easily portable among different architectures and operating systems, so C can be found everywhere, from supercomputers to drivers steering the lowest levels of electronic circuits.

The C++ language [39–41] was designed as a broad, object-oriented extension of the C language whose nature is markedly procedural, with the added option of *exception handling* during program execution. Another important enrichment is the possibility of dynamical resource control (allocation and de-allocation of memory, opening and closing files, establishing connections to databases and libraries depending on the context (*scoped resource management*). The C++ language derives part of its strength from generic programming: algorithms can be written in a form that does not depend on the type of objects occurring in the algorithm. It also benefits from a relatively rich standard library facilitating the work with arrays, iterators, input-output operations, algorithms of general and numerical types, and diagnostic tools. With few exceptions, the C language is a subset of C++, so a valid code written in C is valid in C++ as well (but not the other way around). The standard open-code compiler for C/C++ on Linux/UNIX systems is `gcc` [42], while an excellent commercial compiler that can be used freely in the academic environment is offered by Intel [22].

Java Similar to C++, Java [43] was developed for object-oriented programming, originally in the context of network computing and the desire for easy portability. Yet syntactically it is compatible with neither C++ or C (although it resembles them in many ways). Perhaps the most essential benefit of using Java is the Java Runtime Environment (JRE), an environment in which user application or libraries actually run. An application in Java "communicates" with the physical computer or the actual operating system by means of the *Java Virtual Machine* (JVM). The virtual machine represents a certain abstraction of the actual system. The program in Java therefore need not be written for each architecture or operating system separately, since to do this, one would have to know their numerous details and peculiarities: we write the program only once, and then it can be embedded in any platform on which JRE is running.

Two types of Java compilers exist. The first type allows us to compile the source code (the `.java` files) to bit code (the `.class` files) that is interpreted by the JVM. The best-known compiler from this family is `javac` packaged in any distribution of the Java Development Kit (JDK). The second type of compilers directly translates the source code to machine code for the corresponding architecture. Modern implementations of the JVM can transform the source code to machine code during execution. The GNU compiler `gcj` can do the same [44].

References

1. L.S. Blackford et al., BLAS, basic linear algebra subprograms. ACM Trans. Math. Softw. **28**, 135 (2002)
2. J. Dongarra, Basic linear algebra subprograms technical forum standard. Int. J. High Perform. Comput. Appl. **16**, 1 (2002). The reference (unoptimized) library is available at http://www.netlib.org/blas
3. LAPACK, The Linear Algebra PACKage. http://www.netlib.org/lapack (version for Fortran77), `../lapack95` (version for Fortran95), `../clapack` (version for C), `../lapack++` (version for C++), `../java/f2j` (version for Java)
4. ScaLAPACK, http://www.netlib.org/scalapack
5. ARPACK, http://www.caam.rice.edu/software/ARPACK. The P_ARPACK version also exists: it is appropriate for use in parallel computers. There is also the ARPACK++ interface to C++
6. J.J. Dongarra, V. Eijkhout, Numerical linear algebra algorithms and software. J. Comput. Appl. Math. **123**, 489 (2000)
7. R.C. Whaley, A. Petitet, J.J. Dongarra (ATLAS Collaboration), Automated empirical optimization of software and the ATLAS project. Parallel Comput. **27**, 3 (2001); see also http://math-atlas.sourceforge.net
8. Intel Math Kernel Library, http://www.intel.com/software/products/mkl
9. N.I.M. Gould, J.A. Scott, Y. Hu, A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations. ACM Trans. Math. Softw. **33**, 10 (2007)
10. AMD Core Math Library, http://developer.amd.com/cpu/Libraries/acml
11. MLIB, HP's Mathematical Software Library, http://www.hp.com/go/mlib
12. Apple Velocity Engine, http://developer.apple.com/hardware/ve
13. M. Frigo, S.G. Johnson, The design and implementation of FFTW3. Proc. IEEE **93**, 216 (2005); see also http://www.fftw.org

14. P.B. Bailey, W.N. Everitt, A. Zettl, Algorithm 810: the SLEIGN2 Sturm–Liouville code. ACM Trans. Math. Softw. **27**, 143 (2001); see also http://www.math.niu.edu/SL2
15. P.B. Bailey, M.K. Gordon, L.F. Shampine, Automatic solution of the Sturm–Liouville problem. ACM Trans. Math. Softw. **3**, 193 (1978)
16. S. Pruess, C. Fulton, Mathematical software for Sturm–Liouville problems. ACM Trans. Math. Softw. **19**, 360 (1993)
17. M. Marletta, J.D. Pryce, Automatic solution of Sturm–Liouville problems using the Pruess method. J. Comput. Appl. Math. **39**, 57 (1992)
18. NAG (Numerical Algorithms Group), http://www.nag.co.uk
19. W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 3rd edn. (Cambridge University Press, Cambridge, 2007). See also the equivalent handbooks in Fortran, Pascal and C, as well as http://www.nr.com
20. GSL (GNU Scientific Library), http://www.gnu.org/software/gsl
21. BOOST C++ Libraries, http://www.boost.org
22. Intel Compilers (versions for Fortran and C++), http://www.intel.com/software/products/compilers
23. MATLAB, The MathWorks, http://www.mathworks.com
24. OCTAVE, http://www.octave.org
25. S. Wolfram, WOLFRAM MATHEMATICA, http://www.wolfram.com
26. N. Chonacky, D. Winch, 3Ms for instruction. Reviews of Maple, Mathematica, and Matlab. Comput. Sci. Eng. **May/Jun**, 7 (2005)
27. N. Chonacky, D. Winch, 3Ms for instruction, part 2. Comput. Sci. Eng. **Jul/Aug**, 14 (2005)
28. N. Chonacky, D. Winch, 3Ms: a response. Comput. Sci. and Eng. **Sep/Oct**, 7 (2005)
29. GNUPLOT, http://www.gnuplot.info
30. ROOT, http://root.cern.ch
31. PAW, http://wwwasd.web.cern.ch/wwwasd/paw
32. GRACE (xmgrace), http://plasma-gate.weizmann.ac.il/Grace
33. ORIGINLAB, http://www.originlab.com
34. J. Reid, The future of Fortran. Comput. Sci. Eng. **Jul/Aug**, 59 (2003)
35. V.K. Decyk, C.D. Norton, H.J. Gardner, Why Fortran? Comput. Sci. Eng. **Jul/Aug**, 68 (2007)
36. GNU Fortran Compiler, http://gcc.gnu.org/wiki/GFortran
37. G95, http://www.g95.org
38. B.W. Kernighan, D.M. Ritchie, *C Programming Language (ANSI C)*, 2nd edn. (Prentice Hall, Englewood Cliffs, 1988); the current official language standard is ISO/IEC 9899:1999, also adopted by ANSI; see also http://www.open-std.org/jtc1/sc22/wg14
39. B. Stroustrup, *The C++ Programming Language*, 3rd edn. (Addison-Wesley, Reading, 2000)
40. N. M. Josuttis, *The C++ Standard Library: A Tutorial and Reference* (Addison-Wesley, Reading, 1999)
41. S. Prata, *C++ Primer Plus*, 5th edn. (Sams/Pearson Education, Indianapolis, 2004); the current official language standard is ISO/IEC 14882:2003, see also http://www.open-std.org/jtc1/sc22/wg21
42. GNU Project C/C++ Compiler, http://gcc.gnu.org
43. Java Platform (Standard edition), http://java.sun.com/javase
44. GNU Java Compiler, http://gcc.gnu.org/java

Appendix J
Measuring Program Execution Times on Linux Systems

Here we describe the methods to measure program execution times on Linux and UNIX systems. We use them to identify bottlenecks in the code which helps us optimize its speed. The methods can be used on the command line or within the C++ programming language. Coding in C requires only minor modifications.

time The command `time` for a program `prog` is issued on the command line:

```
$ time prog
real    0m4.52s
user    0m4.15s
sys     0m0.04s
```

The command `time` measures the execution time of the whole program. The `real` value gives the total elapsed time from the beginning of the execution to its end. The processor (CPU) time has two components, `user` and `sys`. The `user` value is the time needed by the program for its own execution and for library calls. The `sys` value is the time spent on system calls. The difference between the total and CPU times, `real` − (`user`+`sys`), is caused by several factors slowing down the program, such as I/O operations, memory accesses, and the time spent by other programs and the operating system.

clock() is a *low-resolution* timer function that can be used within the program itself. The processor time has type `clock_t` and counts the number of oscillations of the CPU clock relative to an arbitrary starting time which does not change within the program. The first call of `clock()` should immediately precede the timed program fragment, while the second call should be just after it:

```
#include <ctime>
clock_t start, end;
start = clock();
/* --- measured code fragment --- */
end = clock();
double elapsed = ((double) (end - start)) / CLOCKS_PER_SEC;
```

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9, © Springer-Verlag Berlin Heidelberg 2012

The factor CLOCKS_PER_SEC (CPU clock ticks per second) converts the difference between end and start to the total elapsed time in seconds. The way of recording the CPU time is not the same on all architectures and operating systems. Typical resolutions of CPU clocks are between hundredths and millionths of a second. The POSIX standard defines CLOCKS_PER_SEC = 1000000 regardless of the actual resolution. The value returned by clock() rolls over after a certain time: on 32-bit POSIX systems this occurs approximately once every 72 minutes. On GNU systems the type clock_t is equivalent to long int, and the type of CLOCKS_PER_SEC is equivalent to int; on other systems both might be of type float. We recommend an explicit conversion to type double (see man clock).

gettimeofday() The timeval structure has two components, both of type int. The first one is tv_sec and measures the time in seconds since January 1, 1970 (the same as time()). The second one is tv_usec and returns the number of microseconds elapsed within the last second measured by tv_sec. On many systems the number of microseconds is accurate to a few decimal places less than the element timeval.tv_usec indicates, so we may expect resolutions of $\approx 1/100$ s. An example of use in the program:

```
#include <sys/time.h>
timeval tim; // in C++
//struct timeval tim; // in C
double t1, t2;

gettimeofday(&tim, NULL);
t1 = tim.tv_sec + (tim.tv_usec/1000000.0);
/* --- measured code fragment --- */
gettimeofday(&tim, NULL);
t2 = tim.tv_sec + (tim.tv_usec/1000000.0);
double elapsed = t2 - t1;
```

For additional information, see man gettimeofday.

rdtsc() For x86-family processors (Pentium and newer), the number of CPU cycles from the last reboot is obtained by the rdtsc() system call, which offers a very precise stop-watch. This is an example of how it can be used in the program:

```
// for 32-bit architecture
unsigned long long int rdtsc() {
  unsigned long long int x;
  asm volatile("rdtsc": "=A" (x));
  return x; }

// for 64-bit architecture
unsigned long long int rdtscll() {
  unsigned int __a, __d;
  asm volatile("rdtsc": "=a" (__a), "=d" (__d));
  return ((unsigned long long)__a)
           |(((unsigned long long)__d)<<32); }
```

```
unsigned long int start, end, elapsed;
start = rdtsc();
/* --- measured code fragment --- */
end = rdtsc();
elapsed = end - start;
```

For additional information, see [1].

getrusage() The `getrusage()` function returns the data on the use of system resources of the running process. The `rusage.ru_utime` structure measures the elapsed user time for the process, while the `rusage.ru_stime` structure measures the elapsed system time. The elements of the structure corresponding to seconds and microseconds needs to be added up, as in the following example:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/resource.h>

double seconds_usr() {
  static struct rusage temp;
  getrusage(RUSAGE_SELF, &temp);
  return temp.ru_utime.tv_sec + temp.ru_utime.tv_usec
           /1000000.0; }
double seconds_sys() {
  static struct rusage temp;
  getrusage(RUSAGE_SELF, &temp);
  return temp.ru_stime.tv_sec + temp.ru_stime.tv_usec
           /1000000.0; }

double start_usr, start_sys, elapsed_usr, elapsed_sys;
start_usr = seconds_usr();
start_sys = seconds_sys();
/* --- measured code fragment --- */
elapsed_usr = seconds_usr() - start_usr;
elapsed_sys = seconds_sys() - start_sys;
```

This function works on most Linux/UNIX systems. For further information, see `man getrusage`.

gprof is an extremely useful command-line tool enabling us to precisely analyze the whole program. The program should be compiled with the `-pg` option and executed; this stores the profile information in the object file and creates the `gmon.out` file. When the program terminates, we run `gprof` over the `gmon.out` file. We obtain a comprehensive display of the computational costs of individual routines. The `gprof` program (in conjunction with the GNU Compiler Collection `gcc` on Linux systems) works with C, C++, Pascal, and Fortran. Example of use:

```
$ c++ -pg prog.cc -o prog && ./prog
$ gprof prog gmon.out > gmon.log
```

This produces the file `gmon.log`, which looks approximately like this:

```
  %   cumulative  self              self     total
 time   seconds  seconds     calls  s/call  s/call name
 23.49     48.86   48.86 806392282    0.00    0.00  aj1_
 21.34     93.24   44.38        65    0.68    1.30  make_kae2c2__
 11.93    118.07   24.82  10607923    0.00    0.00  swint_
 10.97    140.89   22.82        65    0.35    0.54  make_kam1pi__
 10.93    163.63   22.74        65    0.35    1.01  make_ae2c2__
  8.60    181.51   17.89 537095882    0.00    0.00  aj0_
  5.00    191.92   10.41  30406567    0.00    0.00  xint_
  4.01    200.25    8.33        65    0.13    0.24  make_am1pi__
```

The `%` column gives the fraction of total execution time spent in the function (routine) `name` listed at the extreme right. The `cumulative seconds` value is the cumulative time spent in all functions (routines) down to this point in the list. The `self seconds` value measures the time for all executions of the individual routine: this value reveals the most time-consuming routine, and allows us to potentially optimize the program at this point. The `calls` entry specifies the number of calls of a routine, while the `self s/call` and `total s/call` give the corresponding times spent for each call. This table is followed by an even more detailed hierarchy of function calls. For further options see `man gprof`.

In the sample output listed above almost a quarter of time is spent on computing the spherical Bessel function $j_1(z)$, where $z = kx = 2\pi x/\lambda$, by using the routine `double aj1(double z) {return (sin(z)/z-cos(z))/z;}`. Physical circumstances might allow for a long-wavelength approximation,

$$j_1(z) = \frac{1}{z}\left(\frac{\sin z}{z} - \cos z\right) = \frac{z}{3}\left(1 - \frac{z^2}{10}\right) + \mathcal{O}(z^5),$$

when $\lambda \gg x$ and thus $z \ll 1$: if this approximation were allowed, the time-consuming computation of two trigonometric functions could be replaced by just a handful of elementary operations.

References

1. http://en.wikipedia.org/wiki/Time_Stamp_Counter

Index

A
A-stability, 359
$A(\alpha)$-stability, 367
Abel summation, 45
Absolute convergence, 32
Absolute stability, 347–349
Acceleration of convergence, 36–38, 72
ACML, 688
Action function, 27
Adams method, 352
Adams–Bashforth–Moulton method, 354
Adaptive step size, 343, 346
Admissibility condition, 196
Advection equation, 567, 684
Affine invariance, 68
Affinity matrix, 258
Airy functions, 48
Aitken's Δ^2 method, 37, 72
Algebraic multiplicity, 127
Algebraic variety, 91
Algorithm
  Buchberger's, 92
  Burg's, 313
  Cooley–Tukey, 169
  Coppersmith-Winograd, 110
  distillation, 36
  divide-and-conquer, 169
  EM, 256
  Euclid's, 78
  Faugère's (F_5), 92
  FFT, 169
  for M-estimate of location, 215
  for M-estimate of scale, 216
  hierarchical clustering, 251
  Horner's, 74
  Kahan's, 34
  Linz's, 35
  QR, 128
  QZ, 134
  Remes, 6
  Ruffini's, 74
  Strassen, 110
  Wynn, 12
Aliasing, 166
Almost optimal approximation, 8
Alternating series, 38, 51
Amplification factor, 480
Amplification matrix, 497
Amplitude function, 27
Analysis
  asymptotic, 16–31
  auto-regression, 310–319
  canonical correlation, 263, 264
  cluster, 249–259
  factor, 265–269
  linear discriminant, 259–261
  logistic discriminant, 261
  of errors in solving $Ax = b$, 111
  principal component, 244–249
Analytic domain of dependence, 487
Analytic function, 17
Analytic signal, 186
Anderson localization, 150
Angular flux, 426
Approximation
  almost polynomial, 8
  mean-field, 96
  minimax, 6
  of stationary phase, 24
  optimal polynomial, 6
  rational (Padé), 9–15
  spectral, 575
  with Chebyshev polynomials, 8
Arbitrary precision, 4, 377, 634

S. Širca, M. Horvat, *Computational Methods for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-642-32478-9, © Springer-Verlag Berlin Heidelberg 2012

Area of triangle, 5
Arenstorf orbits, 386
Arithmetic mean, 208
Arnold's cat, 372
Asymptotic
  analysis, 16–31, 621, 622
  diffusion limit, 426
  discretization scheme, 424–429
  expansion, 16, 17
    of Fourier integral, 21
  quadrature, 660
  series, 18
  stability, 347–349
ATLAS, 688
Auto-correlation, 305
Auto-covariance, 282
Auto-regression analysis, 310–319
Autonomous system, 335, 364, 667
  Hamiltonian, 368
Avoidance of level crossing, 142, 153

B
B-convergence, 673
B-splines, 430–432, 600
B-stability, 362, 672
Backward difference, 352
Bairstow's method, 84
Balancing, 118
Ballistic diffusion, 289
Banach contraction principle, 652
Banded matrices, 115
Base two, 2
Bayes theorem, 259
BCH formula, 678
Beam-Warming scheme, 530
BEM, 545–549, 570
Bernoulli's
  equation, 100
  method, 82
Berry–Esséen theorem, 284
Bessel functions, 50
Bifurcation, 325
Big-endian ordering, 633
Binary base, 629–633
Binary classification, 259
  efficiency, 261
Binary operations, 2
Biofilms, 563
Bisection, 59
BLAS, 687
BOOST, 689
Borel (re)summation, 44, 52
Boundary condition, 401
  discretization, 402–404, 408, 427, 431, 444
  eigenvalue-dependent, 453, 454
  numerical, 484
Boundary elements, 545–549, 570
Boundary layer, 459
Boundary-value problems, 455
  boundary layer, 459
  scalar, 402–407, 413–416, 418, 419, 430–438, 441–454
  systems, 408–413, 416, 417, 419–421, 438
  with eigenvalues, 441–454
    fourth-order, 463
Box diagram, 212
Box–Muller transformation, 642
Branch statements, 4
Breit–Wigner distribution, 644
Brownian noise, 301
Broyden's method, 69
BTBS, 489
BTCS, 475, 476, 489, 683
  2-dim, 529
BTFS, 489
Burgers equation, 492, 493, 500, 509–511, 602–604, 606, 617
Burg's algorithm, 313
Butcher tableau, 340

C
C/C++, 696
Canonical
  correlation analysis, 263, 264
  parameters, 432
  variable, 264
  variate score, 264
Carry, 646–648
Cauchy
  definition of convergence, 651
  estimate of r, 75
  function, 214
  polynomial, 75
  principal value, 184
  square-root test, 32
Cauchy–Lorentz distribution, 644
Central limit theorem, 283
  generalized, 287
Chaotic scattering, 391
Chapman–Kolmogorov equation, 293
Characteristic
  function, 278
  polynomial, 127
    of AR model, 314
  variables, 497
Characteristics, 485
Chebyshev polynomial, 8
Chebyshev–Galerkin method, 589, 594, 608

Chirik's map, 326
Cholesky method, 114
Chord method, 64
Circular law, 136
Clenshaw–Curtis formulas, 659
Cluster analysis, 249–259
  Gaussian mixture, 256, 257
  hierarchical, 250, 251
  k-means, 253–255
  non-hierarchical, 253–259
  spectral, 258
CN (Crank–Nicolson), 475, 476, 489, 683
  2-dim, 522, 529
CO_2 in atmosphere, 381
Cocktail-party problem, 319
Colebrook–White equation, 101
Collocation
  for boundary-value problems, 429–438
  Gauss, 175, 179
  Gauss–Lobatto, 175, 179
  Gauss–Radau, 175, 179
  method, 576, 597–601, 604
  points, 173, 432, 578, 597
Comparison test, 32
Compatible discretizations, 557
Complete linkage, 253
Complete pivoting, 114
Condition number
  for eigenproblem, 128
  for least-squares problem, 119
  for matrix inverse, 111, 118, 123
    Hager's estimator, 118
Confidence interval
  for correlation coefficient, 226
  for sample mean, 218
  for sample variance, 220
Confidence level, 217
Conjugate gradients, 537
Conservation
  of invariants, 368–372
  of symplectic structure, 372, 373
Conservation law
  $v_t = -[F(v)]_x$, 500, 502, 505
  $v_t = -[F(v)]_x - [G(v)]_y$, 528, 537
Consistency
  of difference scheme, 404, 473
Constant of motion, 374
Contractivity, 672
Convergence
  absolute, 32
  acceleration, 72
  almost sure, 138
  linear, 58
  of a series, 31
  of difference scheme, 404, 476–480
  of sequence, 651
  quadratic, 58
  radius, 17, 33
Convolution, 171, 184, 193, 195
Cooley–Tukey algorithm, 169
Coppersmith-Winograd algorithm, 110
Corrector, 353
  Milne–Simpson, 354
Correlation
  coefficient, 225, 241
    canonical, 264
    rank, 227
  linear, 225, 263
  non-parametric, 226
  time, 304–310
    sample, 306
Correlogram, 317
Coulomb scattering amplitude, 52
Courant–Friedrichs–Lewy criterion, 487
Covariance
  ellipse, 231
  matrix, 236, 246, 263
  of sample, 230
Cowell's method, 359
Crank–Nicolson scheme, 475, 476, 489, 683
  2-dim, 522, 529
Cubic splines, 430–432, 600
Cumulative distribution, 207

D

Dahlquist barrier, 367
Darcy–Weisbach equation, 101
Decomposition
  LU, 114
  QR, 120
Decorrelation of variables, 246
Defective eigenvalue, 127
Defective matrix, 127
Deficient rank, 119, 125
Deflection of a beam, 459
Degrees of freedom, 217
Delaunay triangulation, 554
Dendrogram, 251
Dense output, 340, 345
Descartes' rule, 77
Detailed balance, 293
Determinant
  of matrix, 111
  Wronski, 446
Deterministic process, 277
DFT, 163, 168
Diagonalizable matrix, 127
Diagonalization of a matrix, 127–136

Difference schemes
eigenvalue problems, 443–446
increasing order, 445
for advection equation, 486–490, 527–530, 537–540
for Burgers equation, 492, 493
for diffusion equation, 469–480, 484, 485, 519–527, 542–544
for Poisson equation, 530–537, 544
for wave equation, 508, 509
PDE
alternating direction implicit, 523–526
Beam-Warming, 530
BTBS, 489
BTCS, 475, 476, 489, 683
BTCS (2-dim), 529
BTFS, 489
conservative, 501
Crank–Nicolson, 475, 476, 489, 683
Crank–Nicolson (2-dim), 522, 529
Dufort–Frankel, 484
flux-limiter, 502–505
for wave equation, 490, 491
FTBS, 486
FTBS (2-dim), 527
FTCS, 471, 476, 683
FTCS, $\mathcal{O}(\Delta x^4)$, 484
FTCS (2-dim), 520, 527
FTFS, 486
FTFS (2-dim), 527
high-resolution, 500–505, 537–540
implicit, 489
initialization, 490
Lax–Friedrichs, 489
Lax–Friedrichs (non-linear), 501
Lax–Wendroff, 488
Lax–Wendroff (2-dim), 528
Lax–Wendroff (implicit), 489
Lax–Wendroff (non-linear), 501
leapfrog, 484
modified-flux, 502
Peaceman–Rachford, 523–526
slope-limiter, 502
upwind, 504
upwind (2-dim), 528
Zalesak–Smolarkiewicz, 538
scalar boundary-value problems, 402–407
increasing order, 405
systems of boundary-value problems, 408–413
increasing order, 411–413
systems of PDE, 497–499
Diffusion equation, 426, 427, 469, 505, 593, 596, 599, 609, 614, 616, 681
2-dim (Cartesian coord), 519, 560
2-dim (polar coord), 542–544, 568
energy estimates, 481, 482
non-linear, 482, 563
theorems on maxima, 482–484
Diffusion-reaction kinetics, 457
Discrete
Fourier expansion (DFT), 163, 168
Neumann criterion, 480, 522
polynomial transformation, 173
Discretization
asymptotic, 424–429
compatible, 557
diamond, 427
error, 341, 402, 404, 473–475
mimetic, 557
of boundary conditions, 402–404, 408, 422, 427, 431, 444, 462, 463
for PDE, 471–473, 521, 522, 530–532, 542, 544
of boundary-value problem, 402, 405, 408, 411, 412, 427–429, 462, 463
of derivatives, 402, 403, 462, 463
for PDE, 470, 471, 520, 530–532, 543, 544
of eigenvalue problem, 443
of initial conditions
for PDE, 471–473
physically motivated, 540–560
upwind, 403, 427
Discriminant analysis
linear, 259–261
logistic, 261
multi-class, 262
Discriminant function, 261
Dispersion, 486, 494–496
relation, 494–496
Displacement structure, 115
Dissipation, 486, 494–496
Distillation algorithm, 36
Distribution
Breit–Wigner, 644
Cauchy–Lorentz, 644
χ^2, 220
exponential, 642
F, 221
function, 207
Gauss, 209
multivariate, 642
Maxwell, 207
normal, 209
multivariate, 256
stable, 284–286
Student, 217

Distribution (*cont.*)
uniform, 637
Weibull, 642
Wigner, 142
Divergence of a series, 31
Domain of dependence, 487
Doppler effect, 200
Dufort–Frankel scheme, 484
Dynamical system, 277
Dynamics generator, 675

E
ε_{M} (machine precision), 2–4
Eigenfunctions
of Sturm–Liouville problems, 442
Eigenvalues, 127
of Sturm–Liouville problems, 442
repulsion, 142, 153
Eigenvectors, 127
Embedded method, 344
Ensemble
cyclic orthogonal, 142
cyclic unitary, 142
Gaussian orthogonal, 139, 152
Gaussian unitary, 139, 152
Epidemic of measles, 456
Equi-oscillation theorem, 6
Ergodicity, 279, 282
in auto-correlation, 282
on average, 282
Error
aliasing, 166
discretization, 341, 402, 404
of difference scheme
global, 341
local, 341
relative backward, 113
relative component backward, 118
round-off, 341
Essential singularity, 17
Estimates of condition number, 118
Euclidean distance, 251
Euclid's algorithm, 78
Euler's methods, 337, 338, 360
Euler's transformation, 39
Evolution operator, 675
Expansion
asymptotic, 16, 17
of Fourier integral, 21
discrete Fourier (DFT), 163, 168
in Chebyshev polynomials, 8, 179
in Legendre polynomials, 175
Laurent, 17
power, 16
SVD, 124
Taylor, 17, 336
Exponent of a real number, 629
Exponential operators, 677
Extrapolation methods, 349–351

F
Factor analysis, 265–269
score, 265
weight, 265
Fast Fourier transformation (FFT), 169
Fat tails, 286
Fejér formulas, 659
FEM, 549–557, 569
FFT, 169
FFTW library, 688
Filon quadrature, 662
Finite elements, 549–557, 569
Fitting a constant, 233
Fixed decimal point, 1
Fixed point, 667
ghost (spurious), 671
Flicker noise, 301
Floating decimal point, 1, 629–633
Flux splitting, 498
Flux-limiter function, 503
Fortran, 696
Fourier
collocation derivative, 578–580
series, 162, 587, 615
discrete, 164
for derivative, 577
theorem, 159
transformation, 159
discrete (DFT), 163, 168
fast (FFT), 169
uncertainty, 160
Fourier–Galerkin derivative, 577
Fourier–Galerkin method, 586, 587, 592, 602, 607–609
Fractional part of a real number, 1
Frobenius norm, 625
FTBS, 486
2-dim, 527
FTCS, 471, 476, 683
2-dim, 520, 527
FTCS, $\mathcal{O}(\Delta x^4)$, 484
FTFS, 486
2-dim, 527
Fujiwara's estimate, 75

G
Galactic dynamics, 397
Galerkin condition, 576, 586–588, 590, 592, 602, 609, 610, 612
Galerkin methods, 586–594
Gauss
  distribution, 46, 209
    integral, 46
  elimination, 113
  quadrature, 657
Gauss–Kronrod quadrature, 658, 659
  2-dim, 659
Gauss–Seidel method, 119, 534
Gelfand–Bratu equation, 455
Generalized
  eigenvalue problem, 133
  inverse of a matrix, 126
  linear regression, 236
Generator
  of dynamics, 675
  of random numbers, 279, 637–649
    linear, 646–648
    non-linear, 648
Geometric integration, 368
Geometric multiplicity, 127
Gershgorin's theorem, 480
Ghost point, 472
Gibbs phenomenon, 579
Girko's law, 136
Givens transformation, 122
Global error of difference scheme, 341
GNUPLOT, 695
Gram–Schmidt orthogonalization, 120
Greatest common divisor, 80
Gröbner basis, 89–94, 106
Growth factor, 348
GSL (GNU scientific library), 689

H
Haar
  function, 196
  measure, 143
Hager's estimator, 118
Hamiltonian system, 368, 675, 676
Hampel identifier, 213
Harmonic oscillator, 148, 369
Hearing the shape of the drum, 454
Heisenberg's model, 96
Helmholtz equation, 586–591, 594, 598, 607
Hermite functions, 606
Hermitian eigenproblem, 131
Heron's formula, 5
Hessian matrix, 105
High-resolution schemes, 500–505, 537–540
Hilbert transformation, 184, 190
  discrete, 192
Hilbert–Schmidt norm, 625
Hofstadter's butterfly, 133
Horner's algorithm, 74
Horner's method
  linear, 83
  quadratic, 84
Householder transformation, 122, 131

I
IEEE 754 standard, 2, 4, 629–633
Impact parameter, 392
Initial-value problems, 375
Initialization scheme, 490
Instantaneous
  angular frequency, 187
  complex frequency, 187
  complex phase, 187
  radial frequency, 187
  signal power, 187
Integer numbers, 633, 634
Integral of motion, 358
Integral test, 32
Integrator, 337
  geometric, 368
  Lie-series, 375–379
  reversibility, 373
  symmetry, 373
Inter-quartile range (IQR), 212
Interaction of electric dipoles, 16, 24
Interest rates, 6
Interpolation polynomial, 174
Invariants, 368–372
Inverse problems, 454
Ising model, 328
Isospectral problems, 454
Iteration
  QL, 131
  QR, 128, 131
Iteration matrix, 533

J
Jacobi
  matrix, 68, 85, 336, 362, 406, 409, 421, 668
  method, 130
    iterative, 534
  polynomials, 606
Java, 696
Jenkins–Traub method, 82
Jordan canonical form, 135
Jordan decomposition, 135

K
k-means, 253–255
Kahan's algorithm, 34
Kepler's problem, 57, 395
Korteweg–de Vries equation, 512, 513
Kramers–Kronig relations, 187
Kummer's acceleration, 38
Kummer's test, 32
Kuramoto model, 383

L
Lagging the non-linear term, 493
Lagrange
  interpolation polynomial, 177, 180
  trigonometric polynomial, 164
Laguerre functions, 607
Laguerre's method, 87
Lambert's function, 57, 95
LAPACK, 687
Laplace
  approximation, 21
  equation (2-dim), 570
  method, 21
  transformation, 181
    inverse, 183
Laplace–Runge–Lenz vector, 396
Laurent expansion, 17
Lax theorem, 476
Lax–Friedrichs scheme, 489
  non-linear, 501
Lax–Wendroff scheme, 488
  2-dim, 528
  implicit, 489
  non-linear, 501
LCG, 646
LDA, 259–261
Leakage, 167
Leapfrog, 484
Least median of squares (LMS), 238
Least significant bit (LSB), 630
Least-squares method, 228
Least-squares problem, 9
Legendre
  collocation derivative, 580
  polynomial, 174
Legendre–Galerkin derivative, 580
Legendre–Galerkin method, 587–589, 594, 608, 610
Lehmer sequence, 646
Leibniz's test, 33
Level repulsion, 142
Levin's transformations, 42
Lévy flights, 289
Lexicographic order, 90
Libraries
  ACML, 688
  ATLAS, 688
  BLAS, 687
  BOOST, 689
  FFTW, 688
  for Sturm–Liouville problems, 689
  GMP, 4, 377
  GSL (GNU Scientific Library), 689
  LAPACK, 687
  LINPACK, 688
  MKL, 688
  NAG, 689
  NUMERICAL RECIPES, 689
  SCALAPACK, 688
Lie-series integrator, 375–379
Limit comparison test, 33
Line search, 71
Linear congruence generator, 646
Linear correlation, 225, 263
Linear discriminant analysis, 259–261
Linear prediction, 314
Linear regression, 228–239
  errors in both coordinates, 232
LINPACK, 688
Linz's algorithm, 35
Liouville
  equation, 676
  normal form, 442
  transformation, 442
Lipschitz
  constant, 337, 342, 672
  inequality, 337, 652, 672
    one-sided, 673
Little-endian ordering, 633
Load vector, 552–557
Local error of difference scheme, 341
Log-normal diffusion, 289
Logistic map, 325
Lorentz force, 390
Lorenz system, 377, 388
  Lie-series solution, 377
Loss of significance, 4
LSB, 630
Lyapunov exponent, 325, 378

M
M-estimates
  of location, 213–215
  of scale, 216
Machine precision (ε_M), 2–4
MAD, MADN, 211
Maehly–Newton–Raphson's method, 88
Magnitude of complex number, 5

Manhattan street distance, 251
Mantissa, 1, 629
Marčenko–Pastur theorem, 137
Markov
  chain, 292–299
  matrix, 292
Mass matrix, 552–557
MATHEMATICA, 695
Mathieu equation, 447
MATLAB, 694
Matrix
  affinity, 258
  amplification, 497
  banded, 115
  covariance, 236, 246, 263
  defective, 127
  diagonalizable, 127
  Hessian, 105
  iteration, 533
  Jacobi, 68, 85, 336, 362, 406, 409, 421, 668
  Markov, 292
  mass, 552–557
  multiplication, 109
  non-defective, 127
  non-diagonalizable, 127
  normal, 128
  proximity, 251
  random, 136
  sparse, 118, 136
  stiffness, 552–557
  stochastic, 292
  symmetric, 130
  taking "square root" of, 628
  Toeplitz, 115, 311
  Vandermonde, 116
Maximum entropy principle, 318
Maxwell's distribution, 207
Measles, 456
Median, 211
  absolute deviation (MAD), 211
  of sample, 211
  of squares of residuals (LMS), 238
Meromorphic function, 17
Mersenne twister, 648
Mesh-free methods, 557
Method
  Adams, 351, 352
    stability, 355
  Adams–Bashforth–Moulton, 354
  backward differentiation, 356
    stability, 357
  Bairstow's, 84
  Bernoulli's, 82
  boundary element, 545–549, 570
  Broyden's, 69
  Chebyshev–Galerkin, 589, 594, 608
  Cholesky, 114
  chord, 64
  collocation, 576, 597–601
  conjugate gradients, 537
  CORVAR, 313
  Cowell's, 359
  divide and conquer, 131
  Dormand–Prince, 345
    stability, 348
  embedded, 344
  Euler's explicit, 369, 670
    non-symplecticity, 372–375
    stability, 348
  Euler's implicit, 360
    non-symplecticity, 372
  explicit, 337
    stability, 348
  explicit Euler's, 337
  extrapolation, 349–351
  Fehlberg (Cash–Karp), 345
  finite element, 549–557, 569
  Fourier–Galerkin, 586, 587, 592, 602, 607–609
  Galerkin, 440, 441, 576, 586
  Gauss–Seidel, 119, 534
  Horner's
    linear, 83
    quadratic, 84
  implicit, 337, 359
    Euler's, 360
    midpoint, 360
    multi-step, 367
    trapezoidal, 360
  improved Euler's, 338
  inverse, 641
  Jacobi, 130
    iterative, 534
  Jenkins–Traub, 82
  k-means, 253–255
  Laguerre's, 87
  Laplace, 21
  least-squares, 228
    auto-correlation, 312
    covariance, 313
    iteratively reweighted (IRWLS), 238
  Legendre–Galerkin, 587–589, 594, 608, 610
  Maehly–Newton–Raphson's, 88
  maximum entropy, 318
  mesh-free, 557
  midpoint, 360
  Müller's, 65

Method (*cont.*)
multi-grid, 557
multi-step, 337, 351
Newton–Raphson's, 60
for vector equations, 67
Newton's double-step, 88
Numerov–Cowell, 444
Numerov's, 359
of bisection, 59
of characteristics, 485
of false position, 64
of steepest descent, 27
one-point, 64
Petrov–Galerkin, 440
predictor–corrector, 351, 353
Pruess, 449–452
pseudospectral, 597–601
ratio of uniform deviates, 645
regula falsi, 64
rejection, 643
relaxation, 532–537
reversible, 373
Rosenbrock, 363
Runge–Kutta, 339–349, 359–364
explicit, 339
implicit, 339, 360
order 4 (RK4), 339, 369
order 4 (RK4), stability, 348
partitioned, 371
Runge–Kutta–Nyström, 358
saddle-point, 27
secant, 64
for vector equations, 69
shooting, 413–424, 446–449
single-step, 337, 359
SOR/SSOR, 119, 534–537
spectral, 577
time integration, 605, 606
Störmer–Verlet, 371–373
Störmer's, 359
Sturm's, 77
symmetric, 373
symmetrized Euler's, 338
tangent, 61
tau, 576, 594–597, 603, 613
trapezoidal, 360
Walker, 639
weighted residuals, 439–441, 575–577
with memory, 65
WKB, 29
Mexican hat, 150
Mimetic discretizations, 557
Minimal polynomials, 128
Minimax approximation, 6
Mixing, 279
MKL, 688
Modified Hamiltonians, 374, 375
Monomial, 90
Moody diagram, 101
Moore–Penrose inverse, 126
Morlet wavelet, 197
Morris–Lecar model, 384, 385
Most significant bit (MSB), 630
MSB, 630
Müller's method, 65
Multi-grid methods, 557
Multiple (parallel) shooting, 421–424
Multiple regression, 240–244
Multiplication
of matrices, 109
of polynomials by FFT, 170
Multiplicity of a root, 58

N
NAG, 689
Natural logarithm, 6
Neumann criterion, 477
discrete, 480, 522
Newton–Raphson's method, 60, 653
for vector equations, 67
Newton's
double-step method, 88
law, 336, 357, 368, 389, 395
sum, 79
Nodes, 173, 432, 597
Noise, 299–304
Brownian, 301
flicker, 301
generation, 302
pink, 301
red, 301
white, 300
Non-autonomous system, 335, 364, 669
Non-defective matrix, 127
Non-diagonalizable matrix, 127
Non-linear extrapolation, 37
Non-linear regression, 239
Non-parametric correlation, 226
Norm
Frobenius, 625
Hilbert–Schmidt, 625
in spaces $L^p(\Omega)$ and $L^p_w(\Omega)$, 622
induced, 622
l_2 (Euclidean), 624
$l_{2,\Delta x}$ ("energy"), 624
matrix, 625
max, 623, 625

Norm (*cont.*)
operator, 625
vector, 623
Normal diffusion, 289
Normal distribution, 209
Normal matrix, 128
Normal numbers, 2, 629
Normal system, 119
Northern lights, 396
Nuclear reactions in the Sun, 392–394
Numerical domain of dependence, 487
Numerical flux function, 501
Numerical integration, 655–665
by Gauss quadrature, 657–659
of rapidly oscillating functions, 660–663
of singular functions, 664
NUMERICAL RECIPES, 689
Numerov–Cowell method, 444
Nyquist frequency, 161

O
OCTAVE, 694
Offset grid, 475
One-point method, 64
Optical mouse, 305
Optimal polynomial approximation, 6
Order
lexicographic, 90
of a root, 58
of convergence, 58
total, 90
Oregonator, 394
ORIGIN, 695
Orthogonal polynomials, 172
Orthogonalization of a matrix, 119–126
Oscillations of inhomogeneous string, 460
Outliers, 210, 212
Over-determined system, 119

P
Padé approximation, 9–15
diagonal, 10, 14
duality, 11
existence, 11
of e^z, 11, 14
of matrix function e^X, 682
unitarity, 12
Padé–Borel resummation, 45
Parseval's equality, 160, 172
Partial differential equations
elliptic, 468, 530–537
hyperbolic, 467, 485–491, 527–530, 684–686
mixed type, 468, 491–494
non-linear, 491–494, 601–604
parabolic, 468–485, 519–527, 681–684
Partial pivoting, 114
Partial sum, 31
PAW, 695
PCA, 244–249
Peaceman–Rachford scheme, 523–526
Penning trap, 391
Percentage of explained variance, 247
Percolation, 144
Periodic orbit, 325
Phase space, 372
volume, 372, 676
Pink noise, 301
Pivot growth factor, 114
Pivoting, 114
Planck's law, 94
Poiseuille law, 611
Poisson
bracket, 676, 678
equation
2-dim (Cartesian coord), 530–537, 565, 613
2-dim (polar coord), 544, 545, 568
sum, 159
summation, 44
Polynomial
Cauchy, 75
Chebyshev, 42, 178, 606
orthogonality by points, 178
rational, 607
conditioning of zeros, 81
greatest common divisor, 80
interpolation, 174, 180
Jacobi, 606
Lagrange
interpolation, 177, 180
trigonometric, 164
Legendre, 174
optimal, 7
orthogonal, 172
real, 73
Wilkinson's, 81
Polynomial ideal, 91
Polynomial ring, 90
Polynomial term, 90
Power expansion, 16
Power spectral density, 171, 300
double-sided (PSD), 171
single-sided, 171
Power tails, 286
Predictor, 353
Nyström, 354
Predictor–corrector, 351

Preservation
of invariants, 368–372
of symplectic structure, 372, 373
Primitive variables, 497
Principal component analysis, 244–249
score, 246
Principal value, 184, 664
Probability density function, 207
Problem
$Ax = b$, 111–119
eigenvalue, 127–136
for sparse matrices, 136
generalized, 133
inhomogeneous, 134
NUXI, 634
of least squares, 9, 119–126
minimal solution, 126
three-body, 386
Proximity matrix, 251
Pruess method, 449–452
Prüfer transformation, 447, 448
PSD, 171, 300
Pseudoinverse, 126, 243
Pseudorandom numbers, 279, 637–649
Pseudospectral method, 597–601

Q
QR decomposition, 120
Quadrature, 173, 657–659
Gauss, 173
Gauss–Kronrod, 657–659
Gauss–Lobatto, 173
Gauss–Radau, 173
weights, 173, 657–659
Quarter-circular law, 138
Quasi-linearization, 437
Quotient of convergence, 58
Quotient test, 32

R
ρ-reversibility, 373
Raabe's test, 32
Radial basis functions, 558
Random lattice, 144
Random matrices, 136–144
Random numbers, 279, 637–649
Random processes, 277–304
stationarity, 281
without memory, 293
Random signal, 277
Random variables, 278
Random walks, 287–292
continuous-time, 290
discrete-time, 287
Ratio of complex numbers, 5
Rational approximation, 9–15
Recursive summation of series, 33
Red noise, 301
Region of absolute stability, 347
Regression
linear, 228–239
errors in both coordinates, 232
generalized, 236
multiple, 240–244
principal component, 243
pseudo-inverse, 243
non-linear, 239
robust, 237
with orthogonal polynomials, 229
Regularization, 126
Relative backward error, 113
component, 118
Relatively robust representation, 131
Relaxation methods, 532–537
Remes algorithm, 6
Repulsion of eigenvalues, 142, 153
Residual sum of squares, 241
Residual variance, 241
Reversibility, 373
Richardson extrapolation, 36
Risk level, 217
Robust statistics, 210–216
ROOT, 695
Roots of quadratic equation, 5
Rosenbrock linearization, 363
Round-off error, 3, 34, 36
Rousseeuw identifier, 213
Routh–Hurwitz criterion, 669
Ruffini's algorithm, 74
Runge phenomenon, 180
Runge–Kutta–Nyström method, 358

S
Sample
correlation, 306
covariance, 230
mean, 208
median, 211
path, 280
standard deviation, 208
variance, 208
SCALAPACK, 688
Scalar boundary-value problems, 402–407, 413–416, 418, 419, 430–438, 441–454
non-linear, 405–407
Scalar flux, 426

Schrödinger equation, 29, 98, 148, 152, 442, 452, 462, 513, 515, 606
Secant method, 64
  for vector equations, 69
Seed, 646
Semi-circular law, 140, 153
Semi-classical approach, 29
Semi-linear extrapolation, 37
Sequence
  Bulirsch, 350
  harmonic, 350
  Lehmer, 646
  Romberg, 350
Serial uniformity of sequence, 638
Shock wave in a tube, 511
Shooting methods, 413–424, 446–449
Sign bit, 633
Signal power, 171
Significant, 1
Simpson's formula, 655, 656
Sine pendulum, 389
Single linkage, 253
Singular value decomposition (SVD), 122
Singular values, 123, 627
Singular vectors, 123
Smoothing parameter, 503
SOR/SSOR, 119, 533–537
Space mixing, 326
Sparse matrix, 118, 136
Spectral
  approximation, 575
  convergence, 579
  derivative, 577
    Chebyshev, 581–586
    Chebyshev, computation by FFT, 583–586
    Fourier–Galerkin, 577
    Legendre–Galerkin, 580
  elements, 600
  method, 577
    infinite domains, 606
    semi-infinite domains, 606
  representation of derivatives
    Chebyshev, 581–586
    Fourier, 577–580
    Legendre, 580, 581
Spectrum of a matrix, 127
  unfolding, 140
Stability
  absolute (asymptotic), 347–349
  function, 348
  linear, 667
  Lyapunov, 667–669, 672
  non-linear, 671
  of difference scheme, 404, 476–480
  of multi-step methods, 354
  of one-step methods, 347–349
  semi-linear system, 672
Stable distributions, 284–286
Standard
  deviation of sample, 208
  IEEE 754, 629–633
  map, 326
Standardization of data, 248, 251
Stationary distribution, 294
Stationary phase, 24
Stationary random process, 281
Statistic
  χ^2, 220, 223, 224
  χ^2, reduced, 224
  F, 221
  s_d, t_d, 219
  t, 217, 219
Statistical average, 278
Statistical tests, 217–225
Stencil of mesh points, 520
Step size control, 343, 346
Stieltjes transformation, 137
Stiff problems, 365, 366
Stiffness matrix, 552–557
Stirling approximation, 53
Stochastic matrix, 292
Stochastic variable, 278
Störmer–Verlet method, 371–373
Störmer's method, 359
Strassen algorithm, 110
Student's distribution, 217
Sturm–Liouville problem, 174, 178, 441, 453
  eigenfunctions, 442
  eigenvalues, 442
  program packages, 689
  singular, 452
Sturm's
  comparison theorem, 442
  method, 77
  sequence of polynomials, 78
Subnormal numbers, 2, 631
Super-diffusion, 289
SVD expansion, 124
SVD (singular value decomposition), 122
Symbol of difference scheme, 477
Symmetric matrices, 130
Symplectic integration, 675–680
Symplectic structure, 372, 373, 676
Synchronization of coupled oscillators, 383
Synthetic division, 73
Systems of boundary-value problems, 408–413, 416, 417, 419–421, 438

T
Tangent method, 61
Taylor expansion, 17, 336, 339, 376
Test
  Cauchy square-root, 32
  χ^2, 220, 223
  comparison, 32
  F, 222
  integral, 32
  Kolmogorov–Smirnov, 225
  Kummer's, 32
  Leibniz's, 33
  limit comparison, 33
  of convergence, 32, 33
  quotient, 32
  Raabe's, 32
  statistical, 217–225
  t, 217
Test function, 439, 576
Theorem
  Bayes, 259
  Berry–Esséen, 284
  equi-oscillation, 6
  Fourier, 159
  Gershgorin's, 480
  implicit-function, 60
  interlacing of zeros, 442
  Lax, 476
  Marčenko–Pastur, 137
  mean-value, 17
  on maxima, 482–484
  sampling, 161
  Sturm's comparison, 442
Thomas equation, 391
Three-body problem, 386
Time correlation, 304–310
Time series, 277
Titchmarsh theorem, 184
Toeplitz matrix, 115, 311
Total order, 90
Total sum of squares, 241
Transformation
  Box–Muller, 642
  Euler's, 39
  Fourier, 159
    discrete (DFT), 163, 168
    fast (FFT), 169
  Hilbert, 184, 190
    discrete, 192
  Laplace, 181
  Levin's, 42
  Liouville, 442
  polynomial discrete, 173
  Prüfer, 447, 448
  Stieltjes, 137
  wavelet, 195
    continuous, 195
    discrete, 199
Transition function, 646
Transition point, 30
Transport equation, 425
Trial function, 439, 576
Triangulation, 554
Trigonometric interpolation, 164
Tukey function, 214
Turning point, 30

U
Unfolding of spectrum, 140

V
Van Wijngaarden's trick, 39
Vandermonde matrix, 116
Variance of sample, 208
Viète's formulas, 79
Volume in phase space, 372, 676

W
Walker's alias method, 639
Wave equation, 490, 491, 508, 509
  fourth-order, 508, 509
Wavelet transformation, 195
Weibull probability density, 642
Weight function, 172, 439, 576
Weighted mean, 209, 210
Weighted standard deviation, 209
White noise, 300
Wien's law, 94
Wigner's distribution, 142
Wigner's semi-circular law, 140, 153
Wilkinson's polynomial, 81
Wynn's ϵ-algorithm, 12

Y
Yule–Walker equation, 115, 311

Z
Zalesak–Smolarkiewicz scheme, 538