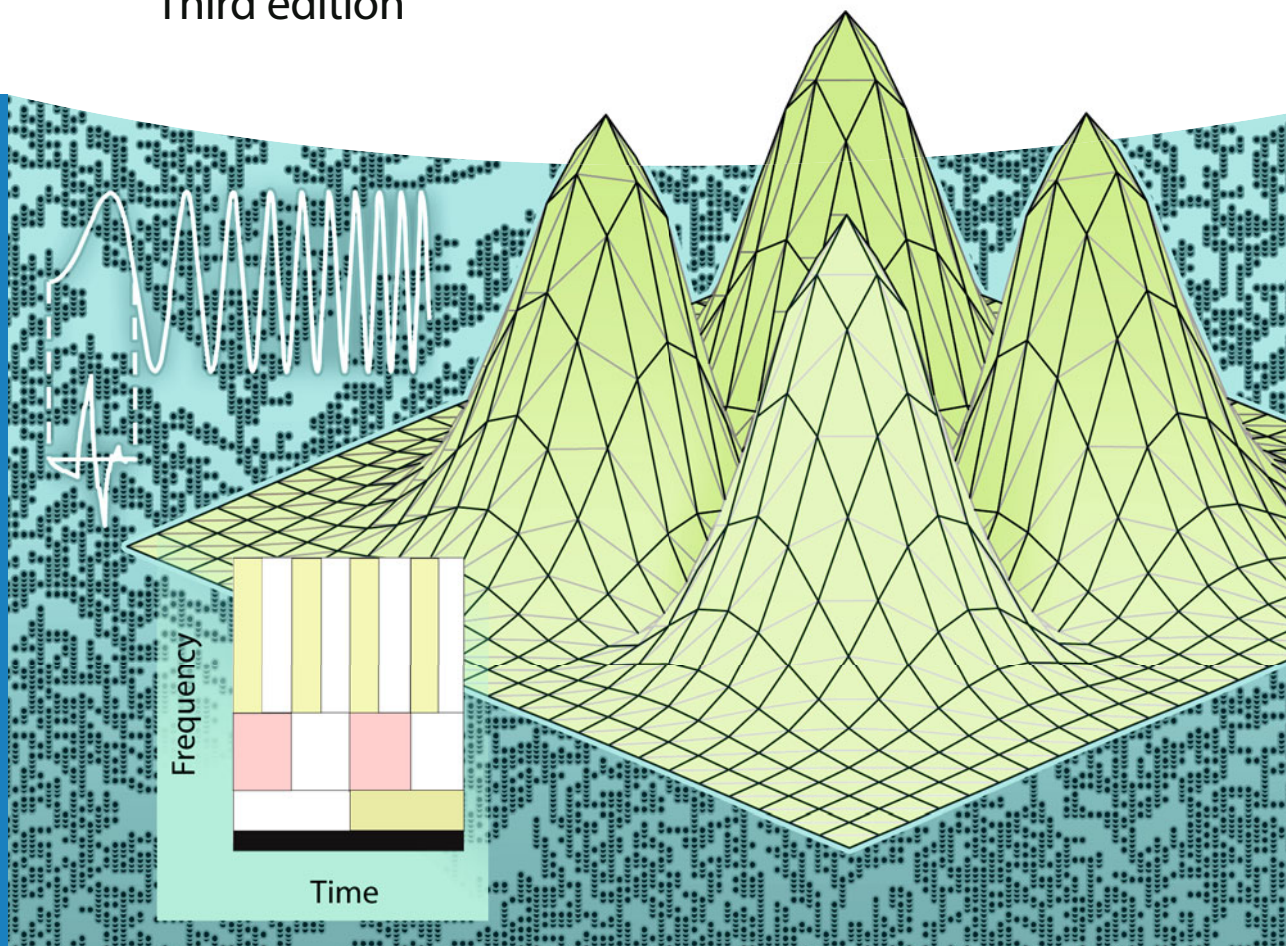


Rubin H. Landau, Manuel J. Páez
and Cristian C. Bordeianu

Computational Physics

Problem Solving with Python

Third edition



Rubin H. Landau

Manuel J. Páez

Cristian C. Bordeianu

Computational Physics

Related Titles

Paar, H.H.

An Introduction to Advanced Quantum Physics

2010

Print ISBN: 978-0-470-68675-1; also available in electronic formats

Har, J.J., Tamma, K.K.

Advances in Computational Dynamics of Particles, Materials and Structures

2012

Print ISBN: 978-0-470-74980-7; also available in electronic formats ISBN: 978-1-119-96589-3

Cohen-Tannoudji, C., Diu, B., Laloe, F.

Quantum Mechanics

2 Volume Set

1977

Print ISBN: 978-0-471-56952-7; also available in electronic formats

Schattke, W., Díez Muiño, R.

Quantum Monte-Carlo Programming

for Atoms, Molecules, Clusters, and Solids

2013

Print ISBN: 978-3-527-40851-1; also available in electronic formats

Zelevinsky, V.

Quantum Physics 1&2

2 Volume Set

2011

Print ISBN: 978-3-527-41057-6; also available in electronic formats

Rubin H. Landau
Manuel J. Páez
Cristian C. Bordeianu

Computational Physics

Problem Solving with Python

3rd completely revised edition

WILEY-VCH
Verlag GmbH & Co. KGaA

Authors

Rubin H. Landau

Oregon State University
97331 Corvallis OR
United States

Manuel J. Pérez

Universad de Antioquia
Departamento Fisica
Medellin
Colombia

Cristian C. Bordeianu

National Military College "Ștefan cal Mare"
Campulung Moldovenesc
Romania

All books published by Wiley-VCH are carefully produced. Nevertheless, authors, editors, and publisher do not warrant the information contained in these books, including this book, to be free of errors. Readers are advised to keep in mind that statements, data, illustrations, procedural details or other items may inadvertently be inaccurate.

Library of Congress Card No.:
applied for

British Library Cataloguing-in-Publication Data:
A catalogue record for this book is available from the British Library.

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at <http://dnb.d-nb.de>.

© 2015 WILEY-VCH Verlag GmbH & Co. KGaA,
Boschstr. 12, 69469 Weinheim, Germany

All rights reserved (including those of translation into other languages). No part of this book may be reproduced in any form – by photoprinting, microfilm, or any other means – nor transmitted or translated into a machine language without written permission from the publishers. Registered names, trademarks, etc. used in this book, even when not specifically marked as such, are not to be considered unprotected by law.

Typesetting le-tex publishing services GmbH,
Leipzig, Deutschland

Cover Design Formgeber, Mannheim, Deutschland

Print and Binding Markono Print Media
Pte Ltd, Singapore

Print ISBN 978-3-527-41315-7

ePDF ISBN 978-3-527-68466-3

ePub ISBN 978-3-527-68469-4

Mobi ISBN 978-3-527-68467-0

Printed on acid-free paper.

To the memory of Jon Maestri

Contents

Dedication *V*

Preface *XIX*

1	Introduction	<i>1</i>
1.1	Computational Physics and Computational Science	<i>1</i>
1.2	This Book's Subjects	<i>3</i>
1.3	This Book's Problems	<i>4</i>
1.4	This Book's Language: The Python Ecosystem	<i>8</i>
1.4.1	Python Packages (Libraries)	<i>9</i>
1.4.2	This Book's Packages	<i>10</i>
1.4.3	The Easy Way: Python Distributions (Package Collections)	<i>12</i>
1.5	Python's Visualization Tools	<i>13</i>
1.5.1	Visual (VPython)'s 2D Plots	<i>14</i>
1.5.2	VPython's Animations	<i>17</i>
1.5.3	Matplotlib's 2D Plots	<i>17</i>
1.5.4	Matplotlib's 3D Surface Plots	<i>22</i>
1.5.5	Matplotlib's Animations	<i>24</i>
1.5.6	Mayavi's Visualizations Beyond Plotting	<i>26</i>
1.6	Plotting Exercises	<i>30</i>
1.7	Python's Algebraic Tools	<i>31</i>
2	Computing Software Basics	<i>33</i>
2.1	Making Computers Obey	<i>33</i>
2.2	Programming Warmup	<i>35</i>
2.2.1	Structured and Reproducible Program Design	<i>36</i>
2.2.2	Shells, Editors, and Execution	<i>37</i>
2.3	Python I/O	<i>39</i>
2.4	Computer Number Representations (Theory)	<i>40</i>
2.4.1	IEEE Floating-Point Numbers	<i>41</i>
2.4.2	Python and the IEEE 754 Standard	<i>47</i>
2.4.3	Over and Underflow Exercises	<i>48</i>
2.4.4	Machine Precision (Model)	<i>49</i>

2.4.5	Experiment: Your Machine's Precision	50
2.5	Problem: Summing Series	51
2.5.1	Numerical Summation (Method)	51
2.5.2	Implementation and Assessment	52
3	Errors and Uncertainties in Computations	53
3.1	Types of Errors (Theory)	53
3.1.1	Model for Disaster: Subtractive Cancellation	55
3.1.2	Subtractive Cancellation Exercises	56
3.1.3	Round-off Errors	57
3.1.4	Round-off Error Accumulation	58
3.2	Error in Bessel Functions (Problem)	58
3.2.1	Numerical Recursion (Method)	59
3.2.2	Implementation and Assessment: Recursion Relations	61
3.3	Experimental Error Investigation	62
3.3.1	Error Assessment	65
4	Monte Carlo: Randomness, Walks, and Decays	69
4.1	Deterministic Randomness	69
4.2	Random Sequences (Theory)	69
4.2.1	Random-Number Generation (Algorithm)	70
4.2.2	Implementation: Random Sequences	72
4.2.3	Assessing Randomness and Uniformity	73
4.3	Random Walks (Problem)	75
4.3.1	Random-Walk Simulation	76
4.3.2	Implementation: Random Walk	77
4.4	Extension: Protein Folding and Self-Avoiding Random Walks	79
4.5	Spontaneous Decay (Problem)	80
4.5.1	Discrete Decay (Model)	81
4.5.2	Continuous Decay (Model)	82
4.5.3	Decay Simulation with Geiger Counter Sound	82
4.6	Decay Implementation and Visualization	84
5	Differentiation and Integration	85
5.1	Differentiation	85
5.2	Forward Difference (Algorithm)	86
5.3	Central Difference (Algorithm)	87
5.4	Extrapolated Difference (Algorithm)	87
5.5	Error Assessment	88
5.6	Second Derivatives (Problem)	90
5.6.1	Second-Derivative Assessment	90
5.7	Integration	91
5.8	Quadrature as Box Counting (Math)	91
5.9	Algorithm: Trapezoid Rule	93
5.10	Algorithm: Simpson's Rule	94

5.11	Integration Error (Assessment)	96
5.12	Algorithm: Gaussian Quadrature	97
5.12.1	Mapping Integration Points	98
5.12.2	Gaussian Points Derivation	99
5.12.3	Integration Error Assessment	100
5.13	Higher Order Rules (Algorithm)	103
5.14	Monte Carlo Integration by Stone Throwing (Problem)	104
5.14.1	Stone Throwing Implementation	104
5.15	Mean Value Integration (Theory and Math)	105
5.16	Integration Exercises	106
5.17	Multidimensional Monte Carlo Integration (Problem)	108
5.17.1	Multi Dimension Integration Error Assessment	109
5.17.2	Implementation: 10D Monte Carlo Integration	110
5.18	Integrating Rapidly Varying Functions (Problem)	110
5.19	Variance Reduction (Method)	110
5.20	Importance Sampling (Method)	111
5.21	von Neumann Rejection (Method)	111
5.21.1	Simple Random Gaussian Distribution	113
5.22	Nonuniform Assessment \odot	113
5.22.1	Implementation \odot	114
6	Matrix Computing	117
6.1	Problem 3: N-D Newton–Raphson; Two Masses on a String	117
6.1.1	Theory: Statics	118
6.1.2	Algorithm: Multidimensional Searching	119
6.2	Why Matrix Computing?	122
6.3	Classes of Matrix Problems (Math)	122
6.3.1	Practical Matrix Computing	124
6.4	Python Lists as Arrays	126
6.5	Numerical Python (NumPy) Arrays	127
6.5.1	NumPy’s linalg Package	132
6.6	Exercise: Testing Matrix Programs	134
6.6.1	Matrix Solution of the String Problem	137
6.6.2	Explorations	139
7	Trial-and-Error Searching and Data Fitting	141
7.1	Problem 1: A Search for Quantum States in a Box	141
7.2	Algorithm: Trial-and-Error Roots via Bisection	142
7.2.1	Implementation: Bisection Algorithm	144
7.3	Improved Algorithm: Newton–Raphson Searching	145
7.3.1	Newton–Raphson with Backtracking	147
7.3.2	Implementation: Newton–Raphson Algorithm	148
7.4	Problem 2: Temperature Dependence of Magnetization	148
7.4.1	Searching Exercise	150
7.5	Problem 3: Fitting An Experimental Spectrum	150

7.5.1	Lagrange Implementation, Assessment	152
7.5.2	Cubic Spline Interpolation (Method)	153
7.6	Problem 4: Fitting Exponential Decay	156
7.7	Least-Squares Fitting (Theory)	158
7.7.1	Least-Squares Fitting: Theory and Implementation	160
7.8	Exercises: Fitting Exponential Decay, Heat Flow and Hubble's Law	162
7.8.1	Linear Quadratic Fit	164
7.8.2	Problem 5: Nonlinear Fit to a Breit–Wigner	167
8	Solving Differential Equations: Nonlinear Oscillations	171
8.1	Free Nonlinear Oscillations	171
8.2	Nonlinear Oscillators (Models)	171
8.3	Types of Differential Equations (Math)	173
8.4	Dynamic Form for ODEs (Theory)	175
8.5	ODE Algorithms	177
8.5.1	Euler's Rule	177
8.6	Runge–Kutta Rule	178
8.7	Adams–Bashforth–Moulton Predictor–Corrector Rule	183
8.7.1	Assessment: rk2 vs. rk4 vs. rk45	185
8.8	Solution for Nonlinear Oscillations (Assessment)	187
8.8.1	Precision Assessment: Energy Conservation	188
8.9	Extensions: Nonlinear Resonances, Beats, Friction	189
8.9.1	Friction (Model)	189
8.9.2	Resonances and Beats: Model, Implementation	190
8.10	Extension: Time-Dependent Forces	190
9	ODE Applications: Eigenvalues, Scattering, and Projectiles	193
9.1	Problem: Quantum Eigenvalues in Arbitrary Potential	193
9.1.1	Model: Nucleon in a Box	194
9.2	Algorithms: Eigenvalues via ODE Solver + Search	195
9.2.1	Numerov Algorithm for Schrödinger ODE \odot	197
9.2.2	Implementation: Eigenvalues via ODE Solver + Bisection Algorithm	200
9.3	Explorations	203
9.4	Problem: Classical Chaotic Scattering	203
9.4.1	Model and Theory	204
9.4.2	Implementation	206
9.4.3	Assessment	207
9.5	Problem: Balls Falling Out of the Sky	208
9.6	Theory: Projectile Motion with Drag	208
9.6.1	Simultaneous Second-Order ODEs	209
9.6.2	Assessment	210
9.7	Exercises: 2- and 3-Body Planet Orbits and Chaotic Weather	211
10	High-Performance Hardware and Parallel Computers	215
10.1	High-Performance Computers	215

10.2	Memory Hierarchy	216
10.3	The Central Processing Unit	219
10.4	CPU Design: Reduced Instruction Set Processors	220
10.5	CPU Design: Multiple-Core Processors	221
10.6	CPU Design: Vector Processors	222
10.7	Introduction to Parallel Computing	223
10.8	Parallel Semantics (Theory)	224
10.9	Distributed Memory Programming	226
10.10	Parallel Performance	227
10.10.1	Communication Overhead	229
10.11	Parallelization Strategies	230
10.12	Practical Aspects of MIMD Message Passing	231
10.12.1	High-Level View of Message Passing	233
10.12.2	Message Passing Example and Exercise	234
10.13	Scalability	236
10.13.1	Scalability Exercises	238
10.14	Data Parallelism and Domain Decomposition	239
10.14.1	Domain Decomposition Exercises	242
10.15	Example: The IBM Blue Gene Supercomputers	243
10.16	Exascale Computing via Multinode-Multicore GPUs	245
11	Applied HPC: Optimization, Tuning, and GPU Programming	247
11.1	General Program Optimization	247
11.1.1	Programming for Virtual Memory (Method)	248
11.1.2	Optimization Exercises	249
11.2	Optimized Matrix Programming with NumPy	251
11.2.1	NumPy Optimization Exercises	254
11.3	Empirical Performance of Hardware	254
11.3.1	Racing Python vs. Fortran/C	255
11.4	Programming for the Data Cache (Method)	262
11.4.1	Exercise 1: Cache Misses	264
11.4.2	Exercise 2: Cache Flow	264
11.4.3	Exercise 3: Large-Matrix Multiplication	265
11.5	Graphical Processing Units for High Performance Computing	266
11.5.1	The GPU Card	267
11.6	Practical Tips for Multicore and GPU Programming	267
11.6.1	CUDA Memory Usage	270
11.6.2	CUDA Programming	271
12	Fourier Analysis: Signals and Filters	275
12.1	Fourier Analysis of Nonlinear Oscillations	275
12.2	Fourier Series (Math)	276
12.2.1	Examples: Sawtooth and Half-Wave Functions	278
12.3	Exercise: Summation of Fourier Series	279
12.4	Fourier Transforms (Theory)	279

12.5	The Discrete Fourier Transform	281
12.5.1	Aliasing (Assessment)	285
12.5.2	Fourier Series DFT (Example)	287
12.5.3	Assessments	288
12.5.4	Nonperiodic Function DFT (Exploration)	290
12.6	Filtering Noisy Signals	290
12.7	Noise Reduction via Autocorrelation (Theory)	290
12.7.1	Autocorrelation Function Exercises	293
12.8	Filtering with Transforms (Theory)	294
12.8.1	Digital Filters: Windowed Sinc Filters (Exploration) ⊙	296
12.9	The Fast Fourier Transform Algorithm ⊙	299
12.9.1	Bit Reversal	301
12.10	FFT Implementation	303
12.11	FFT Assessment	304
13	Wavelet and Principal Components Analyses: Nonstationary Signals and Data Compression	307
13.1	Problem: Spectral Analysis of Nonstationary Signals	307
13.2	Wavelet Basics	307
13.3	Wave Packets and Uncertainty Principle (Theory)	309
13.3.1	Wave Packet Assessment	311
13.4	Short-Time Fourier Transforms (Math)	311
13.5	The Wavelet Transform	313
13.5.1	Generating Wavelet Basis Functions	313
13.5.2	Continuous Wavelet Transform Implementation	316
13.6	Discrete Wavelet Transforms, Multiresolution Analysis ⊙	317
13.6.1	Pyramid Scheme Implementation ⊙	323
13.6.2	Daubechies Wavelets via Filtering	327
13.6.3	DWT Implementation and Exercise	330
13.7	Principal Components Analysis	332
13.7.1	Demonstration of Principal Component Analysis	334
13.7.2	PCA Exercises	337
14	Nonlinear Population Dynamics	339
14.1	Bug Population Dynamics	339
14.2	The Logistic Map (Model)	339
14.3	Properties of Nonlinear Maps (Theory and Exercise)	341
14.3.1	Fixed Points	342
14.3.2	Period Doubling, Attractors	343
14.4	Mapping Implementation	344
14.5	Bifurcation Diagram (Assessment)	345
14.5.1	Bifurcation Diagram Implementation	346
14.5.2	Visualization Algorithm: Binning	347
14.5.3	Feigenbaum Constants (Exploration)	348
14.6	Logistic Map Random Numbers (Exploration) ⊙	348

14.7	Other Maps (Exploration)	348
14.8	Signals of Chaos: Lyapunov Coefficient and Shannon Entropy \odot	349
14.9	Coupled Predator–Prey Models	353
14.10	Lotka–Volterra Model	354
14.10.1	Lotka–Volterra Assessment	356
14.11	Predator–Prey Chaos	356
14.11.1	Exercises	359
14.11.2	LVM with Prey Limit	359
14.11.3	LVM with Predation Efficiency	360
14.11.4	LVM Implementation and Assessment	361
14.11.5	Two Predators, One Prey (Exploration)	362
15	Continuous Nonlinear Dynamics	363
15.1	Chaotic Pendulum	363
15.1.1	Free Pendulum Oscillations	364
15.1.2	Solution as Elliptic Integrals	365
15.1.3	Implementation and Test: Free Pendulum	366
15.2	Visualization: Phase-Space Orbits	367
15.2.1	Chaos in Phase Space	368
15.2.2	Assessment in Phase Space	372
15.3	Exploration: Bifurcations of Chaotic Pendulums	374
15.4	Alternate Problem: The Double Pendulum	375
15.5	Assessment: Fourier/Wavelet Analysis of Chaos	377
15.6	Exploration: Alternate Phase-Space Plots	378
15.7	Further Explorations	379
16	Fractals and Statistical Growth Models	383
16.1	Fractional Dimension (Math)	383
16.2	The Sierpiński Gasket (Problem 1)	384
16.2.1	Sierpiński Implementation	384
16.2.2	Assessing Fractal Dimension	385
16.3	Growing Plants (Problem 2)	386
16.3.1	Self-Affine Connection (Theory)	386
16.3.2	Barnsley’s Fern Implementation	387
16.3.3	Self-Affinity in Trees Implementation	389
16.4	Ballistic Deposition (Problem 3)	390
16.4.1	Random Deposition Algorithm	390
16.5	Length of British Coastline (Problem 4)	391
16.5.1	Coastlines as Fractals (Model)	392
16.5.2	Box Counting Algorithm	392
16.5.3	Coastline Implementation and Exercise	393
16.6	Correlated Growth, Forests, Films (Problem 5)	395
16.6.1	Correlated Ballistic Deposition Algorithm	395
16.7	Globular Cluster (Problem 6)	396
16.7.1	Diffusion-Limited Aggregation Algorithm	396

16.7.2	Fractal Analysis of DLA or a Pollock	399
16.8	Fractals in Bifurcation Plot (Problem 7)	400
16.9	Fractals from Cellular Automata	400
16.10	Perlin Noise Adds Realism \odot	402
16.10.1	Ray Tracing Algorithms	404
16.11	Exercises	407
17	Thermodynamic Simulations and Feynman Path Integrals	409
17.1	Magnets via Metropolis Algorithm	409
17.2	An Ising Chain (Model)	410
17.3	Statistical Mechanics (Theory)	412
17.3.1	Analytic Solution	413
17.4	Metropolis Algorithm	413
17.4.1	Metropolis Algorithm Implementation	416
17.4.2	Equilibration, Thermodynamic Properties (Assessment)	417
17.4.3	Beyond Nearest Neighbors, 1D (Exploration)	419
17.5	Magnets via Wang–Landau Sampling \odot	420
17.6	Wang–Landau Algorithm	423
17.6.1	WLS Ising Model Implementation	425
17.6.2	WLS Ising Model Assessment	428
17.7	Feynman Path Integral Quantum Mechanics \odot	429
17.8	Feynman’s Space–Time Propagation (Theory)	429
17.8.1	Bound-State Wave Function (Theory)	431
17.8.2	Lattice Path Integration (Algorithm)	432
17.8.3	Lattice Implementation	437
17.8.4	Assessment and Exploration	440
17.9	Exploration: Quantum Bouncer’s Paths \odot	440
18	Molecular Dynamics Simulations	445
18.1	Molecular Dynamics (Theory)	445
18.1.1	Connection to Thermodynamic Variables	449
18.1.2	Setting Initial Velocities	449
18.1.3	Periodic Boundary Conditions and Potential Cutoff	450
18.2	Verlet and Velocity–Verlet Algorithms	451
18.3	1D Implementation and Exercise	453
18.4	Analysis	456
19	PDE Review and Electrostatics via Finite Differences and Electrostatics via Finite Differences	461
19.1	PDE Generalities	461
19.2	Electrostatic Potentials	463
19.2.1	Laplace’s Elliptic PDE (Theory)	463
19.3	Fourier Series Solution of a PDE	464
19.3.1	Polynomial Expansion as an Algorithm	466
19.4	Finite-Difference Algorithm	467

- 19.4.1 Relaxation and Over-relaxation 469
- 19.4.2 Lattice PDE Implementation 470
- 19.5 Assessment via Surface Plot 471
- 19.6 Alternate Capacitor Problems 471
- 19.7 Implementation and Assessment 474
- 19.8 Electric Field Visualization (Exploration) 475
- 19.9 Review Exercise 476

- 20 Heat Flow via Time Stepping 477**
- 20.1 Heat Flow via Time-Stepping (Leapfrog) 477
- 20.2 The Parabolic Heat Equation (Theory) 478
- 20.2.1 Solution: Analytic Expansion 478
- 20.2.2 Solution: Time Stepping 479
- 20.2.3 von Neumann Stability Assessment 481
- 20.2.4 Heat Equation Implementation 483
- 20.3 Assessment and Visualization 483
- 20.4 Improved Heat Flow: Crank–Nicolson Method 484
- 20.4.1 Solution of Tridiagonal Matrix Equations \odot 487
- 20.4.2 Crank–Nicolson Implementation, Assessment 490

- 21 Wave Equations I: Strings and Membranes 491**
- 21.1 A Vibrating String 491
- 21.2 The Hyperbolic Wave Equation (Theory) 491
- 21.2.1 Solution via Normal-Mode Expansion 493
- 21.2.2 Algorithm: Time Stepping 494
- 21.2.3 Wave Equation Implementation 496
- 21.2.4 Assessment, Exploration 497
- 21.3 Strings with Friction (Extension) 499
- 21.4 Strings with Variable Tension and Density 500
- 21.4.1 Waves on Catenary 501
- 21.4.2 Derivation of Catenary Shape 501
- 21.4.3 Catenary and Frictional Wave Exercises 503
- 21.5 Vibrating Membrane (2D Waves) 504
- 21.6 Analytical Solution 505
- 21.7 Numerical Solution for 2D Waves 508

- 22 Wave Equations II: Quantum Packets and Electromagnetic 511**
- 22.1 Quantum Wave Packets 511
- 22.2 Time-Dependent Schrödinger Equation (Theory) 511
- 22.2.1 Finite-Difference Algorithm 513
- 22.2.2 Wave Packet Implementation, Animation 514
- 22.2.3 Wave Packets in Other Wells (Exploration) 516
- 22.3 Algorithm for the 2D Schrödinger Equation 517
- 22.3.1 Exploration: Bound and Diffracted 2D Packet 518
- 22.4 Wave Packet–Wave Packet Scattering 518

22.4.1	Algorithm	520
22.4.2	Implementation	520
22.4.3	Results and Visualization	522
22.5	E&M Waves via Finite-Difference Time Domain	525
22.6	Maxwell's Equations	525
22.7	FDTD Algorithm	526
22.7.1	Implementation	530
22.7.2	Assessment	530
22.7.3	Extension: Circularly Polarized Waves	531
22.8	Application: Wave Plates	533
22.9	Algorithm	534
22.10	FDTD Exercise and Assessment	535
23	Electrostatics via Finite Elements	537
23.1	Finite-Element Method \odot	537
23.2	Electric Field from Charge Density (Problem)	538
23.3	Analytic Solution	538
23.4	Finite-Element (Not Difference) Methods, 1D	539
23.4.1	Weak Form of PDE	539
23.4.2	Galerkin Spectral Decomposition	540
23.5	1D FEM Implementation and Exercises	544
23.5.1	1D Exploration	547
23.6	Extension to 2D Finite Elements	547
23.6.1	Weak Form of PDE	548
23.6.2	Galerkin's Spectral Decomposition	548
23.6.3	Triangular Elements	549
23.6.4	Solution as Linear Equations	551
23.6.5	Imposing Boundary Conditions	552
23.6.6	FEM 2D Implementation and Exercise	554
23.6.7	FEM 2D Exercises	554
24	Shocks Waves and Solitons	555
24.1	Shocks and Solitons in Shallow Water	555
24.2	Theory: Continuity and Advection Equations	556
24.2.1	Advection Implementation	558
24.3	Theory: Shock Waves via Burgers' Equation	559
24.3.1	Lax-Wendroff Algorithm for Burgers' Equation	560
24.3.2	Implementation and Assessment of Burgers' Shock Equation	561
24.4	Including Dispersion	562
24.5	Shallow-Water Solitons: The KdeV Equation	563
24.5.1	Analytic Soliton Solution	563
24.5.2	Algorithm for KdeV Solitons	564
24.5.3	Implementation: KdeV Solitons	565
24.5.4	Exploration: Solitons in Phase Space, Crossing	567
24.6	Solitons on Pendulum Chain	567

24.6.1	Including Dispersion	568
24.6.2	Continuum Limit, the Sine-Gordon Equation	570
24.6.3	Analytic SGE Solution	571
24.6.4	Numeric Solution: 2D SGE Solitons	571
24.6.5	2D Soliton Implementation	573
24.6.6	SGE Soliton Visualization	574
25	Fluid Dynamics	575
25.1	River Hydrodynamics	575
25.2	Navier–Stokes Equation (Theory)	576
25.2.1	Boundary Conditions for Parallel Plates	578
25.2.2	Finite-Difference Algorithm and Overrelaxation	580
25.2.3	Successive Overrelaxation Implementation	581
25.3	2D Flow over a Beam	581
25.4	Theory: Vorticity Form of Navier–Stokes Equation	582
25.4.1	Finite Differences and the SOR Algorithm	584
25.4.2	Boundary Conditions for a Beam	585
25.4.3	SOR on a Grid	587
25.4.4	Flow Assessment	589
25.4.5	Exploration	590
26	Integral Equations of Quantum Mechanics	591
26.1	Bound States of Nonlocal Potentials	591
26.2	Momentum–Space Schrödinger Equation (Theory)	592
26.2.1	Integral to Matrix Equations	593
26.2.2	Delta-Shell Potential (Model)	595
26.2.3	Binding Energies Solution	595
26.2.4	Wave Function (Exploration)	597
26.3	Scattering States of Nonlocal Potentials \odot	597
26.4	Lippmann–Schwinger Equation (Theory)	598
26.4.1	Singular Integrals (Math)	599
26.4.2	Numerical Principal Values	600
26.4.3	Reducing Integral Equations to Matrix Equations (Method)	600
26.4.4	Solution via Inversion, Elimination	602
26.4.5	Scattering Implementation	603
26.4.6	Scattering Wave Function (Exploration)	604
	Appendix A Codes, Applets, and Animations	607
	Bibliography	609
	Index	615

Preface

Seventeen years have past since Wiley first published Landau and Páez's *Computational Physics* and twelve years since Cristian Bordeianu joined the collaboration for the second edition. This third edition adheres to the original philosophy that the best way to learn computational physics (CP) is by working on a wide range of projects using the text and the computer as partners. Most projects are still constructed using a computational, scientific problem-solving paradigm:

$$\text{Problem} \rightarrow \text{Theory/Model} \rightarrow \text{Algorithm} \leftrightarrow \text{Visualization} \quad (0.1)$$

Our guiding hypothesis remains that CP is a computational science, which means that to understand CP you need to understand some physics, some applied mathematics, and some computer science. What is different in this edition is the choice of Python for sample codes and an increase in the number of topics covered. We now have a survey of CP which is more than enough for a full-year's course.

The use of Python is more than just a change of language, it is taking advantage of the Python ecosystem of base language plus multiple, specialized libraries to provide all computational needs. In addition, we find Python to be the easiest and most accessible language for beginners, while still being excellent for the type of interactive and exploratory computations now popular in scientific research. Furthermore, Python supplemented by the Visual package (VPython) has gained traction in lower division physics teaching, and this may serve as an excellent segue to a Python-based CP course. Nevertheless, the important aspects of computational modeling and thinking transcends any particular computer language, and so having a Python alternative to our previous use of Fortran, C and Java may help promote this view (codes in all languages are available).

As before, we advocate for the use of a compiled or interpreted programming language when learning CP, in contrast to a higher level problem-solving environment like Mathematica or Maple, which we use in daily work. This follows from our experiences that if you want to *understand* how to compute scientifically, then you must look inside a program's black box and get your hands dirty. Otherwise, the algorithms, logic, and the validity of solutions cannot be ascertained, and that is not a good physics. Not surprisingly, we believe all physicists should know how to read programs how to write them as well.

Notwithstanding our beliefs about programming, we appreciate how time-consuming and frustrating debugging programs often is, and especially for beginners. Accordingly, rather than make the learner write all codes from scratch, we have placed a large number of codes within the text and often ask the learner only to run, modify, and extend them. This not only leaves time for exploration and analysis, but also provides experience in the modern work environment in which one must incorporate new developments into the preexisting codes of others. Be that as it may, for this edition we have added problems in which the relevant codes are not in the text (but are available to instructors). This should permit an instructor to decide on the balance of new and second-hand codes with which their students should work.

In addition to the paper version of the text, there is also an eBook of it that incorporates many of the multimodal enhancements possible with modern technologies: video lecture modules, active simulations, editable codes, animations, and sounds. The eBook is available as a Web (HTML5) document appropriate for both PCs or mobile devices. The lecture modules, which can be viewed separately from the eBook, cover most of the topics in the text, are listed in Appendix B, and are available online. They may provide avenues for alternative understanding the text (either as a preview or a review), for an online course, or for a blended course that replaces some lecture time with lab time. This latter approach, which we recommend, provides time for the instructor to assist students more personally with their projects and their learning issues. The studio-produced lectures are truly “modules,” with active slides, a dynamic table of context, excellent sound (except maybe for a Bronx accent), and with occasional demonstrations replacing the talking head.

The introductory chapter includes tables listing all of the problems and exercises in the text, their locations in the text, as well as the physics courses in which these problems may be used as computational examples. Although we think it better to have entire courses in CP rather than just examples in the traditional courses, the inclusion of examples may serve as a valuable first step towards modernization.

The entire book has been reedited to improve clarity and useability. New materials have also been added, and this has led to additional and reorganized chapters. Specific additions not found in the second edition include: descriptions of the Python language and its packages, demonstrations of several visualization packages, discussions of algebraic tools, an example on protein folding, a derivation of the Gaussian quadrature rule, searching to obtain the temperature dependence of magnetization, chaotic weather patterns, planetary motion, matrix computing with Numerical Python, expanded and updated discussion of parallel computing including scalability and domain composition, optimized matrix computing with NumPy, GPU computing, CUDA programming, principal components analysis, digital filtering, the fast Fourier transform (FFT), an entire chapter on wavelet analysis and data compression, a variety of predator–prey models, signals of chaos, nonlinear behavior of double pendulum, cellular automata, Perlin noise, ray tracing, Wang–Landau sampling for thermodynamic simulations, fi-

nite *element* (in addition to *difference*) solutions of 1D and 2D PDEs, waves on a catenary, finite-difference-time-domain solutions for E&M waves, advection and shock waves in fluids, and a new chapter on fluid dynamics. We hope you enjoy it all!

Redmond, Oregon, June 2014

RHL, rubin@science.oregonstate.edu

Acknowledgments

*Immature poets imitate;
mature poets steal.
T.S. Elliot*

This book and the courses it is based upon could not have been created without continued financial support from the National Science Foundation's CCLI, EPIC, and NPACI programs, as well as support from the Oregon State University. Thank you all and we hope we have done you proud.

Our CP developments have followed the pioneering path paved by the books of Thompson, Gould and Tobochnik, Koonin and Press *et al.*; indubitably, we have borrowed material from them and made it our own with no further thought. We wish to acknowledge valuable contributions by Hans Kowallik, Sally Haerer (video-lecture modules), Paul Fink, Michel Vallières, Joel Wetzels, Oscar A. Restrepo, Jaime Zuluaga, Pavel Snopok, and Henri Jansen. It is our pleasure to acknowledge the invaluable friendship, encouragement, helpful discussions, and experiences we have had with many colleagues and students over the years. We are particularly indebted to Guillermo Avendaño-Franco, Saturo S. Kano, Melanie Johnson, Jon Maestri (deceased), David McIntyre, Shashikant Phatak, Viktor Podolskiy, C.E. Yaguna, Zlatco Dimcovic, and Al Stetz. The new work on principal component analysis resulted from a wonderful collaboration with Jon Wright and Roy Schult in 1997. Our gratitude also goes to the reviewers for their thoughtful and valuable suggestions, and to Bruce Sherwood, who has assisted us in making the Python codes run faster and look better. And finally, Martin Preuss, Nina Stadthaus, Ann Seidel, and Vera Palmer at Wiley-VCH have been a pleasure to work with.

In spite of everyone's best efforts, there are still errors and confusing statements in the book and codes for which we are to blame.

Finally, we extend our gratitude to the wives, Jan and Lucia, whose reliable support and encouragement are lovingly accepted, as always.

1

Introduction

Beginnings are hard.

Chaim Potok

Nothing is more expensive than a start.

Friedrich Nietzsche

This book is really two books. There is a rather traditional paper one with a related Web site, as well as an eBook version containing a variety of digital features best experienced on a computer. Yet even if you are reading from paper, you can still avail yourself of many of digital features, including video-based lecture modules, via the book's Web sites: <http://physics.oregonstate.edu/~rubin/Books/CPbook/eBook/Lectures/> and www.wiley.com/WileyCDA.

We start this chapter with a description of how computational physics (CP) fits into physics and into the broader field of computational science. We then describe the subjects we are to cover, and present lists of all the problems in the text and in which area of physics they can be used as computational examples. The chapter finally gets down to business by discussing the Python language, some of the many packages that are available for Python, and some detailed examples of the use of visualization and symbolic manipulation packages.

1.1

Computational Physics and Computational Science

This book presents computational physics (CP) as a subfield of computational science. This implies that CP is a multidisciplinary subject that combines aspects of physics, applied mathematics, and computer science (CS) (Figure 1.1a), with the aim of solving realistic and ever-changing physics problems. Other computational sciences replace physics with their discipline, such as biology, chemistry, engineering, and so on. Although related, computational science is *not* part of computer science. CS studies computing for its own intrinsic interest and develops the hardware and software tools that computational scientists use. Likewise, applied mathematics develops and studies the algorithms that computational scientists use. As much as we also find math and CS interesting for their own sakes,

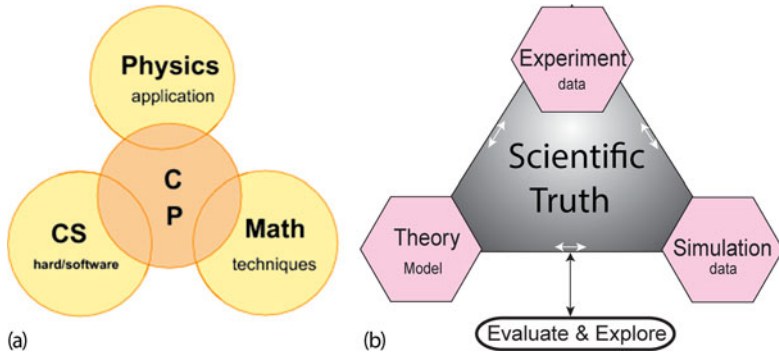


Figure 1.1 (a) A representation of the multi-disciplinary nature of computational physics as an overlap of physics, applied mathematics and computer science, and as a bridge among them. (b) Simulation has been added to experiment and theory as a basic approach in the search for scientific truth. Although this book focuses on simulation, we present it as part of the scientific process.

our focus is on helping the reader do better physics for which you need to understand the CS and math well enough to solve your problems correctly, but not to become an expert programmer.

As CP has matured, we have come to realize that it is more than the overlap of physics, computer science, and mathematics. It is also a bridge among them (the central region in Figure 1.1a) containing core elements of its own, such as computational tools and methods. To us, CP's commonality of tools and its problem-solving mindset draws it toward the other computational sciences and away from the subspecialization found in so much of physics. In order to emphasize our computational science focus, to the extent possible, we present the subjects in this book in the form of a *Problem* to solve, with the components that constitute the solution separated according to the scientific problem-solving paradigm (Figure 1.1b). In recent times, this type of problem-solving approach, which can be traced back to the post-World War II research techniques developed at US national laboratories, has been applied to science education where it is called something like *computational scientific thinking*. This is clearly related to what the computer scientists more recently have come to call *Computational Thinking*, but the former is less discipline specific. Our computational scientific thinking is a hands-on, inquiry-based project approach in which there is problem analysis, a theoretical foundation that considers computability and appropriate modeling, algorithmic thinking and development, debugging, and an assessment that leads back to the original problem.

Traditionally, physics utilizes both experimental and theoretical approaches to discover scientific truth. Being able to transform a theory into an algorithm requires significant theoretical insight, detailed physical and mathematical understanding, and a mastery of the art of programming. The actual debugging, testing, and organization of scientific programs are analogous to experimentation, with the numerical simulations of nature being virtual experiments. The synthesis of

numbers into generalizations, predictions, and conclusions requires the insight and intuition common to both experimental and theoretical science. In fact, the use of computation and simulation has now become so prevalent and essential a part of the scientific process that many people believe that the scientific paradigm has been extended to include simulation as an additional pillar (Figure 1.1b). Nevertheless, as a science, CP must hold experiment supreme, regardless of the beauty of the mathematics.

1.2

This Book's Subjects

This book starts with a discussion of Python as a computing environment and then discusses some basic computational topics. A simple review of computing hardware is put off until Chapter 10, although it also fits logically at the beginning of a course. We include some physics applications in the first third of this book, but put off most CP until the latter two-thirds of the book.

This text has been written to be accessible to upper division undergraduates, although many graduate students without a CP background might also benefit, even from the more elementary topics. We cover both ordinary and partial differential equation (PDE) applications, as well as problems using linear algebra, for which we recommend the established subroutine libraries. Some intermediate-level analysis tools such as discrete Fourier transforms, wavelet analysis, and singular value/principal component decompositions, often poorly understood by physics students, are also covered (and recommended). We also present various topics in fluid dynamics including shock and soliton physics, which in our experience physics students often do not see otherwise. Some more advanced topics include integral equations for both the bound state and (singular) scattering problem in quantum mechanics, as well as Feynman path integrations.

A traditional way to view the materials in this text is in terms of its use in courses. In our classes (CPUG, 2009), we have used approximately the first third of the text, with its emphasis on computing tools, for a course called *Scientific Computing* that is taken after students have acquired familiarity with some compiled language. Typical topics covered in this one-quarter course are given in Table 1.1, although we have used others as well. The latter two-thirds of the text, with its greater emphasis on physics, has typically been used for a two-quarter (20-week) course in CP. Typical topics covered for each quarter are given in Table 1.2. What with many of the topics being research level, these materials can easily be used for a full year's course or for extended research projects.

The text also uses various symbols and fonts to help clarify the type of material being dealt with. These include:

⊙	Optional material
Monospace font	Words as they would appear on a computer screen
Vertical gray line	Note to reader at the beginning of a chapter saying

Table 1.1 Topics for one-quarter (10 Weeks) scientific computing course.

Week	Topics	Chapter	Week	Topics	Chapter
1	OS tools, limits	1, (10)	6	Matrices, N -D search	6
2	Visualization, Errors	1, 3	7	Data fitting	7
3	Monte Carlo,	4, 4	8	ODE oscillations	8
4	Integration, visualization	5, (1)	9	ODE eigenvalues	8
5	Derivatives, searching	5, 7	10	Hardware basics	10

Table 1.2 Topics for two-quarters (20 Weeks) computational physics course.

Computational Physics I			Computational Physics II		
Week	Topics	Chapter	Week	Topics	Chapter
1	Nonlinear ODEs	8, 9	1	Ising model, Metropolis	17
2	Chaotic scattering	9	2	Molecular dynamics	18
3	Fourier analysis, filters	12	3	Project completions	—
4	Wavelet analysis	13	4	Laplace and Poisson PDEs	19
5	Nonlinear maps	14	5	Heat PDE	19
6	Chaotic/double pendulum	15	6	Waves, catenary, friction	21
7	Project completion	15	7	Shocks and solitons	24
8	Fractals, growth	16	8	Fluid dynamics	25
9	Parallel computing, MPI	10, 11	9	Quantum integral equations	26
10	More parallel computing	10, 11	10	Feynman path integration	17

1.3

This Book's Problems

For this book to contribute to a successful learning experience, we assume that the reader will work through what we call the *Problem* at the beginning of each discussion. This entails studying the text, writing, debugging, and running programs, visualizing the results, and then expressing in words what has been performed and what can be concluded. As part of this approach, we suggest that the learner write up a mini lab report for each problem containing sections on

Equations solved	Numerical method	Code listing
Visualization	Discussion	Critique

Although we recognize that programming is a valuable skill for scientists, we also know that it is incredibly exacting and time-consuming. In order to lighten the workload, we provide “bare bones” programs. We recommend that these be used

as guides for the reader's own programs, or tested and extended to solve the problem at hand. In any case, they should be understood as part of the text.

While we think it is best to take a course, or several courses, in CP, we recognize that this is not always possible and some instructors may only be able to include some CP examples in their traditional courses. To assist in this latter endeavor, in this section we list the location of each problem distributed throughout the text and the subject area of each problem. Of course this is not really possible with a multidisciplinary subject like CP, and so there is an overlap. The code used in the table for different subjects is: QM = quantum mechanics or modern physics, CM = classical mechanics, NL = nonlinear dynamics, EM = electricity and magnetism, SP = statistical physics, MM = mathematical methods as well as tools, FD = fluid dynamics, CS = computing fundamentals, Th = thermal physics, and BI = biology. As you can see from the tables, there are many problems and exercises, which reflects our view that you learn computing best by doing it, and that many problems cover more than one subject.

Problems and exercises in computational basics

Subject	Section	Subject	Section	Subject	Section
MM, CS	1.6	CS	2.2.2	CS	2.2.2
CS	2.4.3	CS	2.4.5	CS	2.5.2
CS	3.1.2	CS	3.2	CS	3.2.2
CS	3.3	CS	3.3.1	CS	4.2.2
MM, CS	6.6	CS	10.13.1	CS	10.14.1
CS	11.3.1	CS	11.1.2	CS	11.2.1

Problems and exercises in thermal physics and statistical physics

Subject	Section	Subject	Section	Subject	Section
SP, MM	4.3	SP, MM	4.5	QM, SP	4.6
Th, SP	7.4	Th, SP	7.4.1	NL, SP	16.3.3
NL, SP	16.4.1	NL, SP	16.7.1	NL, SP	16.7.1
NL, SP	16.8	NL, SP	16.11	SP, QM	17.4.1
SP, QM	17.4.2	SP, QM	17.6.2	Th, MM	20.2.4
Th, MM	20.3	TH, MM	20.4.2	TH, MM	20.1
TH, MM	17.1	SP	16.2	SP, BI	16.3
SP	16.4	SP, MM	16.5	SP	16.6
SP	16.7				

Problems and exercises in electricity and magnetism

Subject	Section	Subject	Section	Subject	Section
EM, MM	19.6	EM, MM	19.7	EM, MM	19.8
EM, MM	19.9	EM, MM	23.2	EM, MM	23.5
EM, MM	23.5.1	EM, MM	23.6.6	EM, MM	22.7.2
EM, MM	22.10	EM, MM	19.2		

Problems and exercises in quantum mechanics

Subject	Section	Subject	Section	Subject	Section
QM, SP	4.6	QM, MM	7.1	QM, MM	7.2.1
QM, MM	7.3.2	QM, MM	9.1	QM, MM	9.2
QM, MM	9.2.1	QM, MM	9.3	QM	13.6.3
QM, MM	17.7	QM, MM	26.1	QM, MM	26.3
QM, MM	22.1				

Problems and exercises in classical mechanics and nonlinear dynamics

Subject	Section	Subject	Section	Subject	Section
CM, NL	5.16	CM	6.1	CM, NL	8.1
CM, NL	8.7.1	CM, NL	8.8	CM, NL	8.9
CM, NL	8.10	CM, NL	9.4	CM, NL	9.4.3
CM	9.5	CM	9.7	CM	9.7
NL, FD	9.7	CM	9.7	CM, MM	6.6.2
CM, MM	6.6.1	CM, NL	12.1	BI, NL	14.3
CM, MM	6.6.1	BI, NL	14.4	BI, NL	14.5.2
BI, NL	14.5.3	BI, NL	14.10	BI, NL	14.11.1
BI, NL	14.11.4	BI, NL	14.11.5	CM, NL	15.1.3
CM, NL	15.1	NL, BI	14.1	NL, BI	14.9
CM, NL	15.2.2	CM, NL	15.3	CM, NL	15.4
CM, NL	15.5	CM, NL	15.6	CM, NL	15.7
CM, NL	15.7	NL, MM	16.2.1	NL, MM	16.3.3
NL, MM	16.4.1	NL, MM	16.5.3	NL, MM	16.7.1
NL, MM	16.7.1	NL, MM	16.8	NL, MM	16.11
CM, MM	21.2.4	CM, MM	21.3	CM, MM	21.4.3
CM, MM	24.6	CM, MM	21.1	CM, MM	21.5