Texts in Computational Science and Engineering

16

Editors

Timothy J. Barth
Michael Griebel
David E. Keyes
Risto M. Nieminen
Dirk Roose
Tamar Schlick

More information about this series at
http://www.springer.com/series/5151

Hans Petter Langtangen · Svein Linge

Finite Difference Computing with PDEs

A Modern Software Approach

Springer Open

Hans Petter Langtangen
Lysaker, Norway

Svein Linge
Process, Energy & Environmental Technology
University College of Southeast Norway
Porsgrunn, Norway

ISSN 1611-0994
Texts in Computational Science and Engineering
ISBN 978-3-319-55455-6 ISBN 978-3-319-55456-3 (eBook)
DOI 10.1007/978-3-319-55456-3

Library of Congress Control Number: 2017942747

Mathematic Subject Classification (2010): 34, 35, 65, 68

© The Editor(s) (if applicable) and The Author(s) 2017 This book is an open access publication
Open Access This book is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, duplication, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.
The images or other third party material in this book are included in the work's Creative Commons license, unless indicated otherwise in the credit line; if such material is not included in the work's Creative Commons license and the respective action is not permitted by statutory regulation, users will need to obtain permission from the license holder to duplicate, adapt or reproduce the material.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

There are so many excellent books on finite difference methods for ordinary and partial differential equations that writing yet another one requires a different view on the topic. The present book is not so concerned with the traditional academic presentation of the topic, but is focused at teaching the practitioner how to obtain reliable computations involving finite difference methods. This focus is based on a set of learning outcomes:

1. understanding of the ideas behind finite difference methods,
2. understanding how to transform an algorithm to a well-designed computer code,
3. understanding how to test (verify) the code,
4. understanding potential artifacts in simulation results.

Compared to other textbooks, the present one has a particularly strong emphasis on computer implementation and verification. It also has a strong emphasis on an intuitive understanding of constructing finite difference methods. To learn about the potential non-physical artifacts of various methods, we study exact solutions of finite difference schemes as these give deeper insight into the physical behavior of the numerical methods than the traditional (and more general) asymptotic error analysis. However, asymptotic results regarding convergence rates, typically truncation errors, are crucial for testing implementations, so an extensive appendix is devoted to the computation of truncation errors.

Why finite differences? One may ask why we do finite differences when finite element and finite volume methods have been developed to greater generality and sophistication than finite differences and can cover more problems. The finite element and finite volume methods are also the industry standard nowadays. Why not just those methods? The reason for finite differences is the method's simplicity, both from a mathematical and coding perspective. Especially in academia, where simple model problems are used a lot for teaching and in research (e.g., for verification of advanced implementations), there is a constant need to solve the model problems from scratch with easy-to-verify computer codes. Here, finite differences are ideal. A simple 1D heat equation can of course be solved by a finite element package, but a 20-line code with a difference scheme is just right to the point and provides an understanding of all details involved in the model and the solution method. Everybody nowadays has a laptop and the natural method to attack a 1D heat equation is a simple Python or Matlab program with a difference scheme. The conclusion goes for other fundamental PDEs like the wave equation and Poisson equation as long as the geometry of the domain is a hypercube. The present book contains all the practical information needed to use the finite difference tool in a safe way.

Various pedagogical elements are utilized to reach the learning outcomes, and these are commented upon next.

Simplify, understand, generalize The book's overall pedagogical philosophy is the three-step process of first *simplifying* the problem to something we can *understand* in detail, and when that understanding is in place, we can *generalize* and hopefully address real-world applications with a sound scientific problem-solving approach. For example, in the chapter on a particular family of equations we first simplify the problem in question to a 1D, constant-coefficient equation with simple boundary conditions. We learn how to construct a finite difference method, how to implement it, and how to understand the behavior of the numerical solution. Then we can generalize to higher dimensions, variable coefficients, a source term, and more complicated boundary conditions. The solution of a compound problem is in this way an assembly of elements that are well understood in simpler settings.

Constructive mathematics This text favors a constructive approach to mathematics. Instead of a set of definitions followed by popping up a method, we emphasize how to think about the construction of a method. The aim is to obtain a good intuitive understanding of the mathematical methods.

The text is written in an easy-to-read style much inspired by the following quote.

> *Some people think that stiff challenges are the best device to induce learning, but I am not one of them. The natural way to learn something is by spending vast amounts of easy, enjoyable time at it. This goes whether you want to speak German, sight-read at the piano, type, or do mathematics. Give me the German storybook for fifth graders that I feel like reading in bed, not Goethe and a dictionary. The latter will bring rapid progress at first, then exhaustion and failure to resolve.*
>
> *The main thing to be said for stiff challenges is that inevitably we will encounter them, so we had better learn to face them boldly. Putting them in the curriculum can help teach us to do so. But for teaching the skill or subject matter itself, they are overrated.* [18, p. 86]
> Lloyd N. Trefethen, Applied Mathematician, 1955-.

This book assumes some basic knowledge of finite difference approximations, differential equations, and scientific Python or MATLAB programming, as often met in an introductory numerical methods course. Readers without this background may start with the light companion book "Finite Difference Computing with Exponential Decay Models" [9]. That book will in particular be a useful resource for the programming parts of the present book. Since the present book deals with partial differential equations, the reader is assumed to master multi-variable calculus and linear algebra.

Fundamental ideas and their associated scientific details are first introduced in the simplest possible differential equation setting, often an ordinary differential equation, but in a way that easily allows reuse in more complex settings with partial differential equations. With this approach, new concepts are introduced with a minimum of mathematical details. The text should therefore have a potential for use early in undergraduate student programs.

All nuts and bolts Many have experienced that "vast amounts of easy, enjoyable time", as stated in the quote above, arises when mathematics is implemented on a computer. The implementation process triggers understanding, creativity, and curiosity, but many students find the transition from a mathematical algorithm to a working code difficult and spend a lot of time on "programming issues".

Most books on numerical methods concentrate on the mathematics of the subject while details on going from the mathematics to a computer implementation are less in focus. A major purpose of this text is therefore to help the practitioner by providing *all nuts and bolts* necessary for safely going from the mathematics to a well-designed and well-tested computer code. A significant portion of the text is consequently devoted to programming details.

Python as programming language While MATLAB enjoys widespread popularity in books on numerical methods, we have chosen to use the Python programming language. Python is very similar to MATLAB, but contains a lot of modern software engineering tools that have become standard in the software industry and that should be adopted also for numerical computing projects. Python is at present also experiencing an exponential growth in popularity within the scientific computing community. One of the book's goals is to present an up-to-date Python eco system for implementing finite difference methods.

Program verification Program testing, called *verification*, is a key topic of the book. Good verification techniques are indispensable when debugging computer code, but also fundamental for achieving reliable simulations. Two verification techniques saturate the book: exact solution of discrete equations (where the approximation error vanishes) and empirical estimation of convergence rates in problems with exact (analytical or manufactured) solutions of the differential equation(s).

Vectorized code Finite difference methods lead to code with loops over large arrays. Such code in plain Python is known to run slowly. We demonstrate, especially in Appendix C, how to port loops to fast, compiled code in C or Fortran. However, an alternative is to vectorize the code to get rid of explicit Python loops, and this technique is met throughout the book. Vectorization becomes closely connected to the underlying array library, here `numpy`, and is often thought of as a difficult subject by students. Through numerous examples in different contexts, we hope that the present book provides a substantial contribution to explaining how algorithms can be vectorized. Not only will this speed up serial code, but with a library that can produce parallel code from `numpy` commands (such as Numba[1]), vectorized code can be automatically turned into parallel code and utilize multi-core processors and GPUs. Also when creating tailored parallel code for today's supercomputers, vectorization is useful as it emphasizes splitting up an algorithm into plain and simple

[1] http://numba.pydata.org

array operations, where each operation is trivial to parallelize efficiently, rather than trying to develop a “smart” overall parallelization strategy.

Analysis via exact solutions of discrete equations Traditional asymptotic analysis of errors is important for verification of code using convergence rates, but gives a limited understanding of how and why a correctly implemented numerical method may give non-physical results. By developing exact solutions, usually based on Fourier methods, of the discrete equations, one can obtain a physical understanding of the behavior of a numerical method. This approach is favored for analysis of methods in this book.

Code-inspired mathematical notation Our primary aim is to have a clean and easy-to-read computer code, and we want a close one-to-one relationship between the computer code and mathematical description of the algorithm. This principle calls for a mathematical notation that is governed by the natural notation in the computer code. The unknown is mostly called u, but the meaning of the symbol u in the mathematical description changes as we go from the exact solution fulfilling the differential equation to the symbol `u` that is naturally used for the associated data structure in the code.

Limited scope The aim of this book is not to give an overview of a lot of methods for a wide range of mathematical models. Such information can be found in numerous existing, more advanced books. The aim is rather to introduce basic concepts and a thorough understanding of how to think about computing with finite difference methods. We therefore go in depth with only the most fundamental methods and equations. However, we have a multi-disciplinary scope and address the interplay of mathematics, numerics, computer science, and physics.

Focus on wave phenomena Most books on finite difference methods, or books on theory with computer examples, have their emphasis on diffusion phenomena. Half of this book (Chap. 1, 2, and Appendix C) is devoted to wave phenomena. Extended material on this topic is not so easy find in the literature, so the book should be a valuable contribution in this respect. Wave phenomena is also a good topic in general for choosing the finite difference method over other discretization methods since one quickly needs fine resolution over the entire mesh and uniform meshes are most natural.

Instead of introducing the finite difference method for diffusion problems, where one soon ends up with matrix systems, we do the introduction in a wave phenomena setting where explicit schemes are most relevant. This slows down the learning curve since we can introduce a lot of theory for differences and for software aspects in a context with simple, explicit stencils for updating the solution.

Independent chapters Most book authors are careful with avoiding repetitions of material. The chapters in this book, however, contain some overlap, because we want the chapters to appear meaningful on their own. Modern publishing technology makes it easy to take selected chapters from different books to make a new book tailored to a specific course. The more a chapter builds on details in other chapters, the more difficult it is to reuse chapters in new contexts. Also, most readers find it convenient that important information is explicitly stated, even if it was already met in another chapter.

Supplementary materials All program and data files referred to in this book are available from the book's primary web site: URL: http://github.com/hplgit/fdm-book/.

Acknowledgments Professor Kent-Andre Mardal at the University of Oslo has kindly contributed to enlightening discussions on several issues. Many students have provided lots of useful feedback on the exposition and found many errors in the text. Special efforts in this regard were made by Imran Ali, Shirin Fallahi, Anders Hafreager, Daniel Alexander Mo Søreide Houshmand, Kristian Gregorius Hustad, Mathilde Nygaard Kamperud, and Fatemeh Miri. The collaboration with the Springer team, with Dr. Martin Peters, Thanh-Ha Le Thi, and their production staff has always been a great pleasure and a very efficient process.

Finally, want really appreciate the strong push from the COE of Simula Research Laboratory, Aslak Tveito, for publishing and financing books in open access format, including this one. We are grateful for the laboratory's financial contribution as well as to the financial contribution from the Department of Process, Energy and Environmental Technology at the University College of Southeast Norway.

Oslo, July 2016 Hans Petter Langtangen, Svein Linge

Contents

1 Vibration ODEs 1

1.1 Finite Difference Discretization 1

1.1.1 A Basic Model for Vibrations 1

1.1.2 A Centered Finite Difference Scheme 2

1.2 Implementation 4

1.2.1 Making a Solver Function 4

1.2.2 Verification 6

1.2.3 Scaled Model 10

1.3 Visualization of Long Time Simulations 11

1.3.1 Using a Moving Plot Window 11

1.3.2 Making Animations 13

1.3.3 Using Bokeh to Compare Graphs 15

1.3.4 Using a Line-by-Line Ascii Plotter 18

1.3.5 Empirical Analysis of the Solution 19

1.4 Analysis of the Numerical Scheme 21

1.4.1 Deriving a Solution of the Numerical Scheme 21

1.4.2 The Error in the Numerical Frequency 22

1.4.3 Empirical Convergence Rates and Adjusted ω 24

1.4.4 Exact Discrete Solution 24

1.4.5 Convergence 24

1.4.6 The Global Error 25

1.4.7 Stability 26

1.4.8 About the Accuracy at the Stability Limit 27

1.5 Alternative Schemes Based on 1st-Order Equations 29

1.5.1 The Forward Euler Scheme 29

1.5.2 The Backward Euler Scheme 30

1.5.3 The Crank-Nicolson Scheme 30

1.5.4 Comparison of Schemes 32

1.5.5 Runge-Kutta Methods 33

1.5.6 Analysis of the Forward Euler Scheme 34

1.6 Energy Considerations 36

1.6.1 Derivation of the Energy Expression 36

1.6.2 An Error Measure Based on Energy 38

1.7 The Euler-Cromer Method 40
1.7.1 Forward-Backward Discretization 40
1.7.2 Equivalence with the Scheme for the Second-Order ODE 42
1.7.3 Implementation 43
1.7.4 The Störmer-Verlet Algorithm 45
1.8 Staggered Mesh 46
1.8.1 The Euler-Cromer Scheme on a Staggered Mesh 46
1.8.2 Implementation of the Scheme on a Staggered Mesh . . . 48
1.9 Exercises and Problems 50
1.10 Generalization: Damping, Nonlinearities, and Excitation 57
1.10.1 A Centered Scheme for Linear Damping 57
1.10.2 A Centered Scheme for Quadratic Damping 58
1.10.3 A Forward-Backward Discretization of the Quadratic Damping Term 59
1.10.4 Implementation 59
1.10.5 Verification 60
1.10.6 Visualization 61
1.10.7 User Interface 62
1.10.8 The Euler-Cromer Scheme for the Generalized Model . . 63
1.10.9 The Störmer-Verlet Algorithm for the Generalized Model 64
1.10.10 A Staggered Euler-Cromer Scheme for a Generalized Model 64
1.10.11 The PEFRL 4th-Order Accurate Algorithm 65
1.11 Exercises and Problems 66
1.12 Applications of Vibration Models 67
1.12.1 Oscillating Mass Attached to a Spring 67
1.12.2 General Mechanical Vibrating System 69
1.12.3 A Sliding Mass Attached to a Spring 70
1.12.4 A Jumping Washing Machine 71
1.12.5 Motion of a Pendulum 71
1.12.6 Dynamic Free Body Diagram During Pendulum Motion . 74
1.12.7 Motion of an Elastic Pendulum 79
1.12.8 Vehicle on a Bumpy Road 83
1.12.9 Bouncing Ball 85
1.12.10 Two-Body Gravitational Problem 85
1.12.11 Electric Circuits 88
1.13 Exercises 88

2 Wave Equations 93
2.1 Simulation of Waves on a String 93
2.1.1 Discretizing the Domain 94
2.1.2 The Discrete Solution 94
2.1.3 Fulfilling the Equation at the Mesh Points 94
2.1.4 Replacing Derivatives by Finite Differences 95
2.1.5 Formulating a Recursive Algorithm 96
2.1.6 Sketch of an Implementation 98
2.2 Verification 99
2.2.1 A Slightly Generalized Model Problem 99
2.2.2 Using an Analytical Solution of Physical Significance . . 99

2.2.3 Manufactured Solution and Estimation of Convergence Rates . . . 100
2.2.4 Constructing an Exact Solution of the Discrete Equations 102
2.3 Implementation . . . 104
2.3.1 Callback Function for User-Specific Actions . . . 104
2.3.2 The Solver Function . . . 105
2.3.3 Verification: Exact Quadratic Solution . . . 106
2.3.4 Verification: Convergence Rates . . . 107
2.3.5 Visualization: Animating the Solution . . . 108
2.3.6 Running a Case . . . 112
2.3.7 Working with a Scaled PDE Model . . . 113
2.4 Vectorization . . . 114
2.4.1 Operations on Slices of Arrays . . . 115
2.4.2 Finite Difference Schemes Expressed as Slices . . . 117
2.4.3 Verification . . . 118
2.4.4 Efficiency Measurements . . . 119
2.4.5 Remark on the Updating of Arrays . . . 121
2.5 Exercises . . . 122
2.6 Generalization: Reflecting Boundaries . . . 125
2.6.1 Neumann Boundary Condition . . . 126
2.6.2 Discretization of Derivatives at the Boundary . . . 126
2.6.3 Implementation of Neumann Conditions . . . 127
2.6.4 Index Set Notation . . . 128
2.6.5 Verifying the Implementation of Neumann Conditions . . 130
2.6.6 Alternative Implementation via Ghost Cells . . . 132
2.7 Generalization: Variable Wave Velocity . . . 135
2.7.1 The Model PDE with a Variable Coefficient . . . 135
2.7.2 Discretizing the Variable Coefficient . . . 136
2.7.3 Computing the Coefficient Between Mesh Points . . . 137
2.7.4 How a Variable Coefficient Affects the Stability . . . 138
2.7.5 Neumann Condition and a Variable Coefficient . . . 138
2.7.6 Implementation of Variable Coefficients . . . 139
2.7.7 A More General PDE Model with Variable Coefficients . 140
2.7.8 Generalization: Damping . . . 140
2.8 Building a General 1D Wave Equation Solver . . . 141
2.8.1 User Action Function as a Class . . . 142
2.8.2 Pulse Propagation in Two Media . . . 144
2.9 Exercises . . . 148
2.10 Analysis of the Difference Equations . . . 155
2.10.1 Properties of the Solution of the Wave Equation . . . 155
2.10.2 More Precise Definition of Fourier Representations . . . 157
2.10.3 Stability . . . 158
2.10.4 Numerical Dispersion Relation . . . 160
2.10.5 Extending the Analysis to 2D and 3D . . . 163
2.11 Finite Difference Methods for 2D and 3D Wave Equations . . . 167
2.11.1 Multi-Dimensional Wave Equations . . . 167
2.11.2 Mesh . . . 168
2.11.3 Discretization . . . 169

2.12 Implementation . . . 171
2.12.1 Scalar Computations . . . 172
2.12.2 Vectorized Computations . . . 174
2.12.3 Verification . . . 176
2.12.4 Visualization . . . 177
2.13 Exercises . . . 181
2.14 Applications of Wave Equations . . . 183
2.14.1 Waves on a String . . . 183
2.14.2 Elastic Waves in a Rod . . . 186
2.14.3 Waves on a Membrane . . . 186
2.14.4 The Acoustic Model for Seismic Waves . . . 186
2.14.5 Sound Waves in Liquids and Gases . . . 188
2.14.6 Spherical Waves . . . 189
2.14.7 The Linear Shallow Water Equations . . . 190
2.14.8 Waves in Blood Vessels . . . 192
2.14.9 Electromagnetic Waves . . . 194
2.15 Exercises . . . 195

3 Diffusion Equations . . . 207
3.1 An Explicit Method for the 1D Diffusion Equation . . . 208
3.1.1 The Initial-Boundary Value Problem for 1D Diffusion . . . 208
3.1.2 Forward Euler Scheme . . . 208
3.1.3 Implementation . . . 210
3.1.4 Verification . . . 212
3.1.5 Numerical Experiments . . . 215
3.2 Implicit Methods for the 1D Diffusion Equation . . . 218
3.2.1 Backward Euler Scheme . . . 219
3.2.2 Sparse Matrix Implementation . . . 223
3.2.3 Crank-Nicolson Scheme . . . 224
3.2.4 The Unifying θ Rule . . . 226
3.2.5 Experiments . . . 227
3.2.6 The Laplace and Poisson Equation . . . 227
3.3 Analysis of Schemes for the Diffusion Equation . . . 229
3.3.1 Properties of the Solution . . . 229
3.3.2 Analysis of Discrete Equations . . . 233
3.3.3 Analysis of the Finite Difference Schemes . . . 233
3.3.4 Analysis of the Forward Euler Scheme . . . 234
3.3.5 Analysis of the Backward Euler Scheme . . . 236
3.3.6 Analysis of the Crank-Nicolson Scheme . . . 237
3.3.7 Analysis of the Leapfrog Scheme . . . 237
3.3.8 Summary of Accuracy of Amplification Factors . . . 238
3.3.9 Analysis of the 2D Diffusion Equation . . . 239
3.3.10 Explanation of Numerical Artifacts . . . 241
3.4 Exercises . . . 242
3.5 Diffusion in Heterogeneous Media . . . 245
3.5.1 Discretization . . . 245
3.5.2 Implementation . . . 246
3.5.3 Stationary Solution . . . 247

3.5.4 Piecewise Constant Medium . . . 247
3.5.5 Implementation of Diffusion in a Piecewise Constant Medium . . . 248
3.5.6 Axi-Symmetric Diffusion . . . 251
3.5.7 Spherically-Symmetric Diffusion . . . 252
3.6 Diffusion in 2D . . . 254
3.6.1 Discretization . . . 254
3.6.2 Numbering of Mesh Points Versus Equations and Unknowns . . . 255
3.6.3 Algorithm for Setting Up the Coefficient Matrix . . . 259
3.6.4 Implementation with a Dense Coefficient Matrix . . . 260
3.6.5 Verification: Exact Numerical Solution . . . 264
3.6.6 Verification: Convergence Rates . . . 265
3.6.7 Implementation with a Sparse Coefficient Matrix . . . 266
3.6.8 The Jacobi Iterative Method . . . 270
3.6.9 Implementation of the Jacobi Method . . . 273
3.6.10 Test Problem: Diffusion of a Sine Hill . . . 274
3.6.11 The Relaxed Jacobi Method and Its Relation to the Forward Euler Method . . . 276
3.6.12 The Gauss-Seidel and SOR Methods . . . 277
3.6.13 Scalar Implementation of the SOR Method . . . 277
3.6.14 Vectorized Implementation of the SOR Method . . . 278
3.6.15 Direct Versus Iterative Methods . . . 282
3.6.16 The Conjugate Gradient Method . . . 285
3.6.17 What Is the Recommended Method for Solving Linear Systems? . . . 287
3.7 Random Walk . . . 287
3.7.1 Random Walk in 1D . . . 288
3.7.2 Statistical Considerations . . . 288
3.7.3 Playing Around with Some Code . . . 290
3.7.4 Equivalence with Diffusion . . . 293
3.7.5 Implementation of Multiple Walks . . . 294
3.7.6 Demonstration of Multiple Walks . . . 300
3.7.7 Ascii Visualization of 1D Random Walk . . . 300
3.7.8 Random Walk as a Stochastic Equation . . . 303
3.7.9 Random Walk in 2D . . . 304
3.7.10 Random Walk in Any Number of Space Dimensions . . . 305
3.7.11 Multiple Random Walks in Any Number of Space Dimensions . . . 307
3.8 Applications . . . 308
3.8.1 Diffusion of a Substance . . . 308
3.8.2 Heat Conduction . . . 309
3.8.3 Porous Media Flow . . . 312
3.8.4 Potential Fluid Flow . . . 312
3.8.5 Streamlines for 2D Fluid Flow . . . 313
3.8.6 The Potential of an Electric Field . . . 313
3.8.7 Development of Flow Between Two Flat Plates . . . 313
3.8.8 Flow in a Straight Tube . . . 314

3.8.9 Tribology: Thin Film Fluid Flow 315
3.8.10 Propagation of Electrical Signals in the Brain 316
3.9 Exercises 316

4 Advection-Dominated Equations 323
4.1 One-Dimensional Time-Dependent Advection Equations 323
4.1.1 Simplest Scheme: Forward in Time, Centered in Space . . 324
4.1.2 Analysis of the Scheme 327
4.1.3 Leapfrog in Time, Centered Differences in Space 328
4.1.4 Upwind Differences in Space 331
4.1.5 Periodic Boundary Conditions 333
4.1.6 Implementation 333
4.1.7 A Crank-Nicolson Discretization in Time and Centered Differences in Space 337
4.1.8 The Lax-Wendroff Method 339
4.1.9 Analysis of Dispersion Relations 340
4.2 One-Dimensional Stationary Advection-Diffusion Equation 344
4.2.1 A Simple Model Problem 344
4.2.2 A Centered Finite Difference Scheme 345
4.2.3 Remedy: Upwind Finite Difference Scheme 347
4.3 Time-dependent Convection-Diffusion Equations 349
4.3.1 Forward in Time, Centered in Space Scheme 349
4.3.2 Forward in Time, Upwind in Space Scheme 349
4.4 Applications of Advection Equations 350
4.4.1 Transport of a Substance 350
4.4.2 Transport of Heat in Fluids 350
4.5 Exercises 351

5 Nonlinear Problems 353
5.1 Introduction of Basic Concepts 353
5.1.1 Linear Versus Nonlinear Equations 353
5.1.2 A Simple Model Problem 354
5.1.3 Linearization by Explicit Time Discretization 355
5.1.4 Exact Solution of Nonlinear Algebraic Equations 356
5.1.5 Linearization 357
5.1.6 Picard Iteration 357
5.1.7 Linearization by a Geometric Mean 359
5.1.8 Newton's Method 360
5.1.9 Relaxation 361
5.1.10 Implementation and Experiments 362
5.1.11 Generalization to a General Nonlinear ODE 365
5.1.12 Systems of ODEs 367
5.2 Systems of Nonlinear Algebraic Equations 368
5.2.1 Picard Iteration 369
5.2.2 Newton's Method 369
5.2.3 Stopping Criteria 371
5.2.4 Example: A Nonlinear ODE Model from Epidemiology . 372

5.3 Linearization at the Differential Equation Level 373
5.3.1 Explicit Time Integration 373
5.3.2 Backward Euler Scheme and Picard Iteration 374
5.3.3 Backward Euler Scheme and Newton's Method 375
5.3.4 Crank-Nicolson Discretization 377
5.4 1D Stationary Nonlinear Differential Equations 378
5.4.1 Finite Difference Discretization 378
5.4.2 Solution of Algebraic Equations 379
5.5 Multi-Dimensional Nonlinear PDE Problems 384
5.5.1 Finite Difference Discretization 384
5.5.2 Continuation Methods 386
5.6 Operator Splitting Methods 387
5.6.1 Ordinary Operator Splitting for ODEs 387
5.6.2 Strang Splitting for ODEs 388
5.6.3 Example: Logistic Growth 388
5.6.4 Reaction-Diffusion Equation 391
5.6.5 Example: Reaction-Diffusion with Linear Reaction Term 392
5.6.6 Analysis of the Splitting Method 400
5.7 Exercises 401

A Useful Formulas 409
A.1 Finite Difference Operator Notation 409
A.2 Truncation Errors of Finite Difference Approximations 410
A.3 Finite Differences of Exponential Functions 411
A.4 Finite Differences of t^n 411
A.4.1 Software 412

B Truncation Error Analysis 415
B.1 Overview of Truncation Error Analysis 415
B.1.1 Abstract Problem Setting 415
B.1.2 Error Measures 416
B.2 Truncation Errors in Finite Difference Formulas 417
B.2.1 Example: The Backward Difference for $u'(t)$ 417
B.2.2 Example: The Forward Difference for $u'(t)$ 418
B.2.3 Example: The Central Difference for $u'(t)$ 419
B.2.4 Overview of Leading-Order Error Terms in Finite Difference Formulas 420
B.2.5 Software for Computing Truncation Errors 421
B.3 Exponential Decay ODEs 422
B.3.1 Forward Euler Scheme 422
B.3.2 Crank-Nicolson Scheme 423
B.3.3 The θ-Rule 424
B.3.4 Using Symbolic Software 424
B.3.5 Empirical Verification of the Truncation Error 425
B.3.6 Increasing the Accuracy by Adding Correction Terms . . 430
B.3.7 Extension to Variable Coefficients 432
B.3.8 Exact Solutions of the Finite Difference Equations 433
B.3.9 Computing Truncation Errors in Nonlinear Problems . . . 434

B.4 Vibration ODEs . . . 434
B.4.1 Linear Model Without Damping . . . 434
B.4.2 Model with Damping and Nonlinearity . . . 437
B.4.3 Extension to Quadratic Damping . . . 438
B.4.4 The General Model Formulated as First-Order ODEs . . . 439
B.5 Wave Equations . . . 440
B.5.1 Linear Wave Equation in 1D . . . 440
B.5.2 Finding Correction Terms . . . 441
B.5.3 Extension to Variable Coefficients . . . 442
B.5.4 Linear Wave Equation in 2D/3D . . . 444
B.6 Diffusion Equations . . . 445
B.6.1 Linear Diffusion Equation in 1D . . . 445
B.6.2 Nonlinear Diffusion Equation in 1D . . . 446
B.7 Exercises . . . 447

C Software Engineering; Wave Equation Model . . . 451
C.1 A 1D Wave Equation Simulator . . . 451
C.1.1 Mathematical Model . . . 451
C.1.2 Numerical Discretization . . . 451
C.1.3 A Solver Function . . . 452
C.2 Saving Large Arrays in Files . . . 455
C.2.1 Using `savez` to Store Arrays in Files . . . 455
C.2.2 Using `joblib` to Store Arrays in Files . . . 457
C.2.3 Using a Hash to Create a File or Directory Name . . . 458
C.3 Software for the 1D Wave Equation . . . 459
C.3.1 Making Hash Strings from Input Data . . . 460
C.3.2 Avoiding Rerunning Previously Run Cases . . . 460
C.3.3 Verification . . . 461
C.4 Programming the Solver with Classes . . . 462
C.4.1 Class Parameters . . . 463
C.4.2 Class Problem . . . 465
C.4.3 Class Mesh . . . 465
C.4.4 Class Function . . . 468
C.4.5 Class Solver . . . 471
C.5 Migrating Loops to Cython . . . 475
C.5.1 Declaring Variables and Annotating the Code . . . 476
C.5.2 Visual Inspection of the C Translation . . . 478
C.5.3 Building the Extension Module . . . 479
C.5.4 Calling the Cython Function from Python . . . 480
C.6 Migrating Loops to Fortran . . . 480
C.6.1 The Fortran Subroutine . . . 481
C.6.2 Building the Fortran Module with f2py . . . 482
C.6.3 How to Avoid Array Copying . . . 483
C.7 Migrating Loops to C via Cython . . . 485
C.7.1 Translating Index Pairs to Single Indices . . . 485
C.7.2 The Complete C Code . . . 486
C.7.3 The Cython Interface File . . . 486
C.7.4 Building the Extension Module . . . 487

C.8 Migrating Loops to C via f2py 488
C.8.1 Migrating Loops to C++ via f2py 489
C.9 Exercises 490

References 493

Index 495

List of Exercises, Problems and Projects

Problem 1.1: Use linear/quadratic functions for verification 50
Exercise 1.2: Show linear growth of the phase with time 51
Exercise 1.3: Improve the accuracy by adjusting the frequency 51
Exercise 1.4: See if adaptive methods improve the phase error 52
Exercise 1.5: Use a Taylor polynomial to compute u^1 52
Problem 1.6: Derive and investigate the velocity Verlet method 52
Problem 1.7: Find the minimal resolution of an oscillatory function 52
Exercise 1.8: Visualize the accuracy of finite differences for a cosine function 53
Exercise 1.9: Verify convergence rates of the error in energy 53
Exercise 1.10: Use linear/quadratic functions for verification 53
Exercise 1.11: Use an exact discrete solution for verification 53
Exercise 1.12: Use analytical solution for convergence rate tests 53
Exercise 1.13: Investigate the amplitude errors of many solvers 54
Problem 1.14: Minimize memory usage of a simple vibration solver 54
Problem 1.15: Minimize memory usage of a general vibration solver 56
Exercise 1.16: Implement the Euler-Cromer scheme for the generalized model 56
Problem 1.17: Interpret $[D_tD_tu]^n$ as a forward-backward difference 56
Exercise 1.18: Analysis of the Euler-Cromer scheme 56
Exercise 1.19: Implement the solver via classes 66
Problem 1.20: Use a backward difference for the damping term 66
Exercise 1.21: Use the forward-backward scheme with quadratic damping . . 67
Exercise 1.22: Simulate resonance 88
Exercise 1.23: Simulate oscillations of a sliding box 88
Exercise 1.24: Simulate a bouncing ball 88
Exercise 1.25: Simulate a simple pendulum 89
Exercise 1.26: Simulate an elastic pendulum 89
Exercise 1.27: Simulate an elastic pendulum with air resistance 90
Exercise 1.28: Implement the PEFRL algorithm 91
Exercise 2.1: Simulate a standing wave 122
Exercise 2.2: Add storage of solution in a user action function 123
Exercise 2.3: Use a class for the user action function 123
Exercise 2.4: Compare several Courant numbers in one movie 123
Exercise 2.5: Implementing the solver function as a generator 124
Project 2.6: Calculus with 1D mesh functions 124

Exercise 2.7: Find the analytical solution to a damped wave equation 148
Problem 2.8: Explore symmetry boundary conditions 148
Exercise 2.9: Send pulse waves through a layered medium 148
Exercise 2.10: Explain why numerical noise occurs 149
Exercise 2.11: Investigate harmonic averaging in a 1D model 149
Problem 2.12: Implement open boundary conditions 149
Exercise 2.13: Implement periodic boundary conditions 151
Exercise 2.14: Compare discretizations of a Neumann condition 151
Exercise 2.15: Verification by a cubic polynomial in space 152
Exercise 2.16: Check that a solution fulfills the discrete model 181
Project 2.17: Calculus with 2D mesh functions 181
Exercise 2.18: Implement Neumann conditions in 2D 182
Exercise 2.19: Test the efficiency of compiled loops in 3D 182
Exercise 2.20: Simulate waves on a non-homogeneous string 195
Exercise 2.21: Simulate damped waves on a string 195
Exercise 2.22: Simulate elastic waves in a rod 195
Exercise 2.23: Simulate spherical waves 195
Problem 2.24: Earthquake-generated tsunami over a subsea hill 196
Problem 2.25: Earthquake-generated tsunami over a 3D hill 198
Problem 2.26: Investigate Mayavi for visualization 199
Problem 2.27: Investigate visualization packages 199
Problem 2.28: Implement loops in compiled languages 199
Exercise 2.29: Simulate seismic waves in 2D 200
Project 2.30: Model 3D acoustic waves in a room 200
Project 2.31: Solve a 1D transport equation 201
Problem 2.32: General analytical solution of a 1D damped wave equation . . 204
Problem 2.33: General analytical solution of a 2D damped wave equation . . 205
Exercise 3.1: Explore symmetry in a 1D problem 242
Exercise 3.2: Investigate approximation errors from a $u_x = 0$ boundary condition 243
Exercise 3.3: Experiment with open boundary conditions in 1D 243
Exercise 3.4: Simulate a diffused Gaussian peak in 2D/3D 244
Exercise 3.5: Examine stability of a diffusion model with a source term 245
Exercise 3.6: Stabilizing the Crank-Nicolson method by Rannacher time stepping 316
Project 3.7: Energy estimates for diffusion problems 316
Exercise 3.8: Splitting methods and preconditioning 318
Problem 3.9: Oscillating surface temperature of the earth 319
Problem 3.10: Oscillating and pulsating flow in tubes 320
Problem 3.11: Scaling a welding problem 320
Exercise 3.12: Implement a Forward Euler scheme for axi-symmetric diffusion 322
Exercise 4.1: Analyze 1D stationary convection-diffusion problem 351
Exercise 4.2: Interpret upwind difference as artificial diffusion 351
Problem 5.1: Determine if equations are nonlinear or not 401
Problem 5.2: Derive and investigate a generalized logistic model 401
Problem 5.3: Experience the behavior of Newton's method 402
Exercise 5.4: Compute the Jacobian of a 2×2 system 402
Problem 5.5: Solve nonlinear equations arising from a vibration ODE 402

Exercise 5.6: Find the truncation error of arithmetic mean of products 402
Problem 5.7: Newton's method for linear problems 404
Problem 5.8: Discretize a 1D problem with a nonlinear coefficient 404
Problem 5.9: Linearize a 1D problem with a nonlinear coefficient 404
Problem 5.10: Finite differences for the 1D Bratu problem 404
Problem 5.11: Discretize a nonlinear 1D heat conduction PDE by finite differences 405
Problem 5.12: Differentiate a highly nonlinear term 405
Exercise 5.13: Crank-Nicolson for a nonlinear 3D diffusion equation 406
Problem 5.14: Find the sparsity of the Jacobian 406
Problem 5.15: Investigate a 1D problem with a continuation method 406
Exercise B.1: Truncation error of a weighted mean 447
Exercise B.2: Simulate the error of a weighted mean 447
Exercise B.3: Verify a truncation error formula 447
Problem B.4: Truncation error of the Backward Euler scheme 447
Exercise B.5: Empirical estimation of truncation errors 448
Exercise B.6: Correction term for a Backward Euler scheme 448
Problem B.7: Verify the effect of correction terms 448
Problem B.8: Truncation error of the Crank-Nicolson scheme 448
Problem B.9: Truncation error of $u' = f(u, t)$ 448
Exercise B.10: Truncation error of $[D_t D_t u]^n$ 449
Exercise B.11: Investigate the impact of approximating $u'(0)$ 449
Problem B.12: Investigate the accuracy of a simplified scheme 449
Exercise C.1: Explore computational efficiency of numpy.sum versus built-in sum 490
Exercise C.2: Make an improved `numpy.savez` function 490
Exercise C.3: Visualize the impact of the Courant number 491
Exercise C.4: Visualize the impact of the resolution 491