
Contents

1	Algorithms and Implementations	1
1.1	Finite Difference Methods	1
1.1.1	A Basic Model for Exponential Decay	1
1.1.2	The Forward Euler Scheme	2
1.1.3	The Backward Euler Scheme	7
1.1.4	The Crank–Nicolson Scheme	8
1.1.5	The Unifying θ -Rule	10
1.1.6	Constant Time Step	11
1.1.7	Mathematical Derivation of Finite Difference Formulas	11
1.1.8	Compact Operator Notation for Finite Differences	14
1.2	Implementations	15
1.2.1	Computer Language: Python	16
1.2.2	Making a Solver Function	17
1.2.3	Integer Division	18
1.2.4	Doc Strings	18
1.2.5	Formatting Numbers	19
1.2.6	Running the Program	20
1.2.7	Plotting the Solution	21
1.2.8	Verifying the Implementation	22
1.2.9	Computing the Numerical Error as a Mesh Function	25
1.2.10	Computing the Norm of the Error Mesh Function	26
1.2.11	Experiments with Computing and Plotting	28
1.2.12	Memory-Saving Implementation	33
1.3	Exercises	35
2	Analysis	39
2.1	Experimental Investigations	39
2.1.1	Discouraging Numerical Solutions	39
2.1.2	Detailed Experiments	41
2.2	Stability	44
2.2.1	Exact Numerical Solution	44
2.2.2	Stability Properties Derived from the Amplification Factor	45

2.3	Accuracy	46
2.3.1	Visual Comparison of Amplification Factors	46
2.3.2	Series Expansion of Amplification Factors	47
2.3.3	The Ratio of Numerical and Exact Amplification Factors	49
2.3.4	The Global Error at a Point	50
2.3.5	Integrated Error	50
2.3.6	Truncation Error	52
2.3.7	Consistency, Stability, and Convergence	53
2.4	Various Types of Errors in a Differential Equation Model	54
2.4.1	Model Errors	55
2.4.2	Data Errors	58
2.4.3	Discretization Errors	61
2.4.4	Rounding Errors	62
2.4.5	Discussion of the Size of Various Errors	64
2.5	Exercises	64
3	Generalizations	67
3.1	Model Extensions	67
3.1.1	Generalization: Including a Variable Coefficient	67
3.1.2	Generalization: Including a Source Term	68
3.1.3	Implementation of the Generalized Model Problem	69
3.1.4	Verifying a Constant Solution	70
3.1.5	Verification via Manufactured Solutions	71
3.1.6	Computing Convergence Rates	73
3.1.7	Extension to Systems of ODEs	75
3.2	General First-Order ODEs	76
3.2.1	Generic Form of First-Order ODEs	76
3.2.2	The θ -Rule	77
3.2.3	An Implicit 2-Step Backward Scheme	77
3.2.4	Leapfrog Schemes	78
3.2.5	The 2nd-Order Runge–Kutta Method	78
3.2.6	A 2nd-Order Taylor-Series Method	79
3.2.7	The 2nd- and 3rd-Order Adams–Bashforth Schemes	79
3.2.8	The 4th-Order Runge–Kutta Method	80
3.2.9	The Odespy Software	81
3.2.10	Example: Runge–Kutta Methods	82
3.2.11	Example: Adaptive Runge–Kutta Methods	85
3.3	Exercises	86
4	Models	91
4.1	Scaling	91
4.1.1	Dimensionless Variables	91
4.1.2	Dimensionless Numbers	92
4.1.3	A Scaling for Vanishing Initial Condition	92
4.2	Evolution of a Population	93
4.2.1	Exponential Growth	93
4.2.2	Logistic Growth	94
4.3	Compound Interest and Inflation	94

4.4	Newton's Law of Cooling	95
4.5	Radioactive Decay	95
4.5.1	Deterministic Model	96
4.5.2	Stochastic Model	96
4.5.3	Relation Between Stochastic and Deterministic Models	97
4.5.4	Generalization of the Radioactive Decay Modeling	98
4.6	Chemical Kinetics	98
4.6.1	Irreversible Reaction of Two Substances	98
4.6.2	Reversible Reaction of Two Substances	99
4.6.3	Irreversible Reaction of Two Substances into a Third	100
4.6.4	A Biochemical Reaction	101
4.7	Spreading of Diseases	101
4.8	Predator-Prey Models in Ecology	102
4.9	Decay of Atmospheric Pressure with Altitude	103
4.9.1	The General Model	103
4.9.2	Multiple Atmospheric Layers	104
4.9.3	Simplifications	104
4.10	Compaction of Sediments	105
4.11	Vertical Motion of a Body in a Viscous Fluid	106
4.11.1	Overview of Forces	106
4.11.2	Equation of Motion	107
4.11.3	Terminal Velocity	108
4.11.4	A Crank–Nicolson Scheme	108
4.11.5	Physical Data	109
4.11.6	Verification	109
4.11.7	Scaling	110
4.12	Viscoelastic Materials	110
4.13	Decay ODEs from Solving a PDE by Fourier Expansions	111
4.14	Exercises	112
5	Scientific Software Engineering	127
5.1	Implementations with Functions and Modules	127
5.1.1	Mathematical Problem and Solution Technique	128
5.1.2	A First, Quick Implementation	128
5.1.3	A More Decent Program	129
5.1.4	Prefixing Imported Functions by the Module Name	131
5.1.5	Implementing the Numerical Algorithm in a Function	133
5.1.6	Do not Have Several Versions of a Code	133
5.1.7	Making a Module	134
5.1.8	Example on Extending the Module Code	136
5.1.9	Documenting Functions and Modules	137
5.1.10	Logging Intermediate Results	138
5.2	User Interfaces	142
5.2.1	Command-Line Arguments	142
5.2.2	Positional Command-Line Arguments	145
5.2.3	Option-Value Pairs on the Command Line	146
5.2.4	Creating a Graphical Web User Interface	148

5.3	Tests for Verifying Implementations	151
5.3.1	Doctests	151
5.3.2	Unit Tests and Test Functions	153
5.3.3	Test Function for the Solver	157
5.3.4	Test Function for Reading Positional Command-Line Arguments	158
5.3.5	Test Function for Reading Option-Value Pairs	159
5.3.6	Classical Class-Based Unit Testing	160
5.4	Sharing the Software with Other Users	161
5.4.1	Organizing the Software Directory Tree	162
5.4.2	Publishing the Software at GitHub	163
5.4.3	Downloading and Installing the Software	164
5.5	Classes for Problem and Solution Method	166
5.5.1	The Problem Class	167
5.5.2	The Solver Class	168
5.5.3	Improving the Problem and Solver Classes	169
5.6	Automating Scientific Experiments	172
5.6.1	Available Software	172
5.6.2	The Results We Want to Produce	173
5.6.3	Combining Plot Files	174
5.6.4	Running a Program from Python	175
5.6.5	The Automating Script	176
5.6.6	Making a Report	178
5.6.7	Publishing a Complete Project	182
5.7	Exercises	183
	References	189
	Index	191

List of Exercises, Problems, and Projects

Exercise 1.1: Define a mesh function and visualize it	35
Problem 1.2: Differentiate a function	35
Problem 1.3: Experiment with divisions	36
Problem 1.4: Experiment with wrong computations	37
Problem 1.5: Plot the error function	37
Problem 1.6: Change formatting of numbers and debug	37
Problem 2.1: Visualize the accuracy of finite differences	64
Problem 2.2: Explore the θ -rule for exponential growth	65
Problem 2.3: Explore rounding errors in numerical calculus	66
Exercise 3.1: Experiment with precision in tests and the size of u	86
Exercise 3.2: Implement the 2-step backward scheme	86
Exercise 3.3: Implement the 2nd-order Adams–Bashforth scheme	87
Exercise 3.4: Implement the 3rd-order Adams–Bashforth scheme	87
Exercise 3.5: Analyze explicit 2nd-order methods	87
Project 3.6: Implement and investigate the Leapfrog scheme	87
Problem 3.7: Make a unified implementation of many schemes	89
Problem 4.1: Radioactive decay of Carbon-14	112
Exercise 4.2: Derive schemes for Newton’s law of cooling	112
Exercise 4.3: Implement schemes for Newton’s law of cooling	113
Exercise 4.4: Find time of murder from body temperature	114
Exercise 4.5: Simulate an oscillating cooling process	114
Exercise 4.6: Simulate stochastic radioactive decay	115
Problem 4.7: Radioactive decay of two substances	115
Exercise 4.8: Simulate a simple chemical reaction	116
Exercise 4.9: Simulate an n -th order chemical reaction	116
Exercise 4.10: Simulate a biochemical process	117
Exercise 4.11: Simulate spreading of a disease	118
Exercise 4.12: Simulate predator-prey interaction	118
Exercise 4.13: Simulate the pressure drop in the atmosphere	119
Exercise 4.14: Make a program for vertical motion in a fluid	119
Project 4.15: Simulate parachuting	120
Exercise 4.16: Formulate vertical motion in the atmosphere	121
Exercise 4.17: Simulate vertical motion in the atmosphere	122
Problem 4.18: Compute $y = x $ by solving an ODE	122

Problem 4.19: Simulate fortune growth with random interest rate	122
Exercise 4.20: Simulate a population in a changing environment	123
Exercise 4.21: Simulate logistic growth	124
Exercise 4.22: Rederive the equation for continuous compound interest	124
Exercise 4.23: Simulate the deformation of a viscoelastic material	124
Problem 5.1: Make a tool for differentiating curves	183
Problem 5.2: Make solid software for the Trapezoidal rule	184
Problem 5.3: Implement classes for the Trapezoidal rule	186
Problem 5.4: Write a doctest and a test function	186
Problem 5.5: Investigate the size of tolerances in comparisons	186
Exercise 5.6: Make use of a class implementation	186
Problem 5.7: Make solid software for a difference equation	187