

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>

Einar Smith

Introduction to the Tools of Scientific Computing



Springer

Einar Smith
Fraunhofer Institut für Algorithmen
und Wissenschaftliches Rechnen SCAI
Sankt Augustin, Germany

Institut für Numerische Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn
Bonn, Germany

ISSN 1611-0994 ISSN 2197-179X (electronic)
Texts in Computational Science and Engineering
ISBN 978-3-030-60807-1 ISBN 978-3-030-60808-8 (eBook)
<https://doi.org/10.1007/978-3-030-60808-8>

Mathematics Subject Classification (2010): 97N80

© Springer Nature Switzerland AG 2020

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

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, expressed 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.

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

Preface

This book provides an introduction to common programming tools and methods in numerical mathematics and scientific computing. In contrast to widespread standard approaches, it does not focus on a specific language, but rather aims to explain the central underlying concepts.

In general, new concepts are first introduced in the particularly user-friendly Python language and then transferred and expanded in various scientific programming environments from C / C ++, Julia and MATLAB to Maple.

This approach can best be illustrated with a recurring leitmotif: the numerical approximation of differential equations.

In the basic Python chapter, we introduce the function concept and illustrate how derivatives can be approximated using discrete difference quotients.

In the chapter on Scientific Python, we expand this idea in the approximation of ordinary differential equations. In comparison we show how the same sample problems can be solved with integrated solver operators.

We then extend the manual approximation methods to partial equations and their solution using the finite difference method. To prepare for this, the necessary linear algebra algorithms and matrix constructions, such as Poisson matrices, are developed along the way. In particular, we show that the use of sparse matrices leads to a significant acceleration of the computation.

The chapter on Symbolic Python shows how symbolic solvers can facilitate or replace the numerical solution process.

In addition, in this symbolic context we can easily explain the Galerkin method and thus pave the way for the introduction of the finite element method later.

In the C chapter we show how the sparse matrices that were already used in Python can be generated by machine-oriented, low-level programming using pointer constructions to implement linked lists.

The Julia chapter illustrates how the fundamental programming techniques discussed so far can be formulated in a promising newcomer to mathematical programming, which aims to combine the elegance of Python with the performance of C/C++.

We then show how corresponding ideas are implemented in commercial programming environments such as `MATLAB` and `Maple`.

In the `Maple` chapter, we expand the Galerkin method introduced in `Python` to the finite element method. In this way the reader is already well prepared for the general discussion in the `FEniCS` chapter.

In the chapters on distributed programming we show how classical sequential methods can be adopted to distributed and parallel computation, which is becoming increasingly important in the recent machine development, where multicore processors have found their way even into standard home computers. We discuss different approaches in `Python`, `C/C++` and `Julia`.

The book closes with an advanced programming topic, the `FEniCS` project, which combines many of the previously developed techniques for the automated solution of partial differential equations using the finite element method.

The book is based on material from courses held by the author in the Department of Mathematics at the University of Bonn. Originally primarily intended for students of mathematics – at both bachelor and master level – the courses also attracted participants from other fields, including computer science, physics and geology.

The book is primarily aimed at students of mathematics and disciplines in which mathematical methods play an important role. A certain level of mathematical maturity is recommended. Technically, however, only very basic ideas from linear algebra and analysis are assumed, so that the book can also be read by anyone with a solid high-school education in mathematics who wants to understand how mathematical algorithms can be performed by digital computers. Programming experience is not required.

The book is written in such a way that it can also serve as a text for private self-study. With the exception of a few advanced examples in the `MATLAB` and `Maple` chapters, you can run all programs directly on your home computer, based on free open source programming environments.

The book can therefore also serve as a repetition and to improve the understanding of basic numerical algorithms.

Acknowledgments

The author wishes to thank Helmut Griebel and Marc Alexander Schweitzer from the Institute for Numerical Simulation at the University of Bonn for the opportunity to hold the programming courses and for their help in contacting Springer Verlag.

I would like to thank the course participants for their lively collaboration and critical comments, which have helped to transform the loose lecture notes into a comprehensive presentation. In particular, I would like to thank Angelina Steffens for proofreading the manuscript.

Very useful was also the correspondence with Chris Rackauckas, the author of the Julia differential equation package in Chapter 8, and Lisandro Dalcin, the author of the Python MPI implementation in Chapter 11.

I am also grateful for helpful suggestions from the anonymous referees.

My special thanks go to Martin Peters, Ruth Allewelt and Leonie Kunz from Springer-Verlag for their support and encouragement while preparing the book.

Bonn, September 2020

Einar Smith

Contents

1	Introduction	1
Part I Background		
2	Mathematical Foundations of Programming	9
2.1	A Simple Machine Model	9
2.2	Digital Computers	14
Part II Core Languages		
3	Python, the Fundamentals	19
3.1	Python Interpreter	20
3.2	Elementary Data Types	21
3.3	Variables and Value Assignments	25
3.4	Control Structures	26
3.5	Collection Types: Lists, Tuples, Dictionaries and Sets	30
3.6	Functions	34
3.7	String Formatting	38
3.8	Writing and Reading Files	39
3.9	Object-Oriented Programming and Classes	42
3.10	Exercises	47
4	Python in Scientific Computation	51
4.1	NumPy	51
4.2	Conjugate Gradient	58
4.3	SciPy	60
4.4	Linear Algebra	60
4.5	Graphics with Matplotlib	68
4.6	Nonlinear Equations, Optimization	70
4.7	Numerical Integration, Ordinary Differential Equations	73
4.8	Partial Differential Equations	81

4.9	Round off: Random Numbers	87
4.10	Exercises	88
5	Python in Computer Algebra	93
5.1	Symbolic Calculation, Numbers	93
5.2	Equation Systems	99
5.3	Linear Algebra	101
5.4	Calculus	106
5.5	Ordinary Differential Equations	108
5.6	Galerkin Method	109
5.7	Exercises	113
6	The C Language	115
6.1	Basics	116
6.2	Control Structures: Branches, Loops	118
6.3	Functions	121
6.4	Arrays	122
6.5	Pointers	125
6.6	Structures	128
6.7	Files, Input and Output	131
6.8	Conclusion	132
7	The C++ Language	133
7.1	Transfer from C	133
7.2	Basics	134
7.3	Lambda Expressions	136
7.4	Data Type vector	137
7.5	Reference Operator	139
7.6	Classes	140
7.7	Header Files	144
7.8	Summary and Outlook	145
7.9	Exercises	146
8	Julia	149
8.1	Basics	149
8.2	Control Structures: Branching, Loops	152
8.3	Functions	154
8.4	Collection Types	158
8.5	Composite Types	162
8.6	Linear Algebra	165
8.7	Ordinary Differential Equations	170
8.8	Partial Differential Equations	175
8.9	Working with Files	177
8.10	Exercises	179

Part III Commercial Computing Environments

9	MATLAB	185
9.1	Basics	185
9.2	Vectors and Matrices	187
9.3	Control Structures: Branching, Loops	192
9.4	Functions	194
9.5	M-Files	196
9.6	Linear Algebra	199
9.7	Ordinary Differential Equations	201
9.8	Partial Differential Equations	205
9.9	Exercises	207
10	Maple	211
10.1	Basics	211
10.2	Functions	213
10.3	Linear Algebra	216
10.4	Calculus	221
10.5	Interpolation with Spline Functions	224
10.6	Differential Equations	227
10.7	Galerkin Method	229
10.8	Finite Element Method	230
10.9	Exercises	233

Part IV Distributed Computing

11	A Python Approach to Message Passing	239
11.1	Introduction to the Message Passing Interface	239
11.2	Communicating Processes	241
11.3	Integral Approximation	246
11.4	Vector Dot Product	248
11.5	Laplace Equations	251
11.6	Conjugate Gradient Method	254
11.7	Exercises	256
12	Parallel Computing in C/C++	259
12.1	Integral Approximation of π	260
12.2	Scatter and Gather	262
12.3	Conjugate Gradient	263
12.4	Shared Memory Programming	265
12.5	Parallelizing Loops	268
12.6	Integral Approximation of π	271
12.7	Parallelized Function Modules	271
12.8	Hybrid Message Passing and Shared Memory Programming	273
12.9	Exercises	274

13 Distributed Processing in Julia	277
13.1 Point-to-Point Communication	278
13.2 Distributed Loops, Reductions	278
13.3 Monte Carlo Method	279
13.4 Fibonacci Function	281
13.5 Shared Arrays.....	281
13.6 Distributed Arrays	283
13.7 Exercises	286

Part V Specialized Programming Environments

14 Automated Solution of PDEs with FEniCS	289
14.1 Finite Element Method, One-Dimensional	290
14.2 FEniCS Implementation, One-Dimensional	293
14.3 Poisson Equations	296
14.4 Time-Dependent Poisson Equation	300
14.5 Nonlinear Equations	304
14.6 Neumann Boundary Conditions	308
14.7 Stokes Equation	311
14.8 Adaptive Mesh Refinement	316
14.9 User Defined Meshes	318
14.10 Final Note: Parallel Processing in FEniCS	323
14.11 Exercises	323

References	327
-------------------------	-----

Index	329
Subjects and Persons	329
Python	332
C/C++	337
Julia	339
MATLAB	342
Maple	343