Texts in Computational Science and Engineering 25

Editors

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

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

Part III Commercial Computing Environments

Part IV Distributed Computing