Preface

This book is written for graduate and advanced undergraduate students of sciences, engineering, and mathematics as a tutorial on how to think about, organize, and implement programs in scientific computing. It may be used as a textbook for classroom instruction, or by individuals for self-directed learning. It is the outgrowth of a course that I have taught periodically over nearly 20 years at UMBC. In the beginning it was targeted to graduate students in Applied Mathematics to help them quickly acquire programming skills to implement and experiment with the ideas and algorithms mostly related to their doctoral researches. Over the years it has gained popularity among the Mechanical Engineering students. In recent years, about a quarter of the enrollment has come from the College of Engineering. Additionally, I have had the pleasure of having a number of advanced undergraduate student in the course; they have done quite well.

The course's, and by extension the book's, immediate goal is to provide an interesting and instructive set of problems—I call them *Projects*—each of which begins with the presentation of a problem and an algorithm for solving it and then leads the reader through implementing the algorithm in C and compiling and testing the results. The ultimate goal in my mind, however, is pedagogy, not programming per se. Most students can attest that there is a substantial gap between what one learns in an undergraduate course dedicated to programming and what is required to implement ideas and algorithms of scientific computing in a coherent fashion. This book aims to bridge that gap through a set of carefully thought-out and well-developed programming projects. The book does not "lecture" the reader; rather, it shows the way—at times by doing, and at times by prompting what to do—to lead him/her toward a goal. Paramount in my objectives is to instill a habit of, and an appreciation for, *modular program organization*. Breaking a large program into small and logically independent units makes it easier to understand, test/debug, and alter/expand, and—as demonstrated abundantly throughout—it makes the parts available for reuse elsewhere.

I hope that the reader will take away more than just programming techniques from this book. I have strived to make the projects interesting, intriguing, inviting, challenging, and illuminating on their own, apart from their programming aspects. The range of the topics inevitably reflects my tastes, but I hope that there is enough variety here to enable any reader to find several rewarding projects to work on. Some of my favorite projects are

- the *Nelder–Mead simplex algorithm* for minimizing functions in $\boldsymbol{R}^n$ (with or without constraints) with applications to computing finite deformations of trusses under large loads via minimizing the energy;
- the *Haar wavelet transform* in one and two dimensions, with applications to image analysis and image compression;

- a very simple yet intriguing model of *evolution through natural selection* and the effect of the environment on the emergence of genetically distinct species (speciation);
- the comparison/contrast of several *finite difference algorithms* for solving the time-dependent linear heat equation and extending one of the algorithms to solving the (nonlinear and degenerate) porous medium equation; and
- a minimal implementation of the *finite element method* for solving second order elliptic partial differential equations on arbitrary two-dimensional domains through unstructured triangular meshes and linear elements.

Additionally, I am particularly pleased with the *array.h* header file of Chapter 8 which provides a set of preprocessor macros for allocating and freeing memory for vectors and matrices of *arbitrary types* entirely within the bounds of standard C. That header file is used throughout the rest of the book.

To reach the book's intended readership, that is, the advanced undergraduate through beginning graduate students, I have made a consistent effort throughout to keep the mathematical prerequisites and jargon to a minimum and have not shunned from skirting technical issues to the extent that I could. For instance, the Galerkin approximation and the finite element method are introduced in Chapter 25 without explicit references to Hilbert or Sobolev spaces, although these concepts are brought up in the subsequent chapter. I have included plenty of references to the literature to help the curious reader to learn more. I believe that a good knowledge of undergraduate multivariable calculus and linear algebra is all that is needed to follow the topics in this book, but a graduate student's technical maturity certainly will help.

Part I of the book, consisting of Chapters 1 through 6, is a prerequisite for Part II, which makes up the rest of the book. A working familiarity with the concepts introduced in Part I is tacitly assumed throughout. The chapters of Part II consist of individual projects. Part II is *definitely not* intended for linear/sequential reading. A chart on page xi shows the chapter interdependencies. Chapter 18 on the Nelder–Mead simplex method, for instance, depends on Chapters 7 (memory allocation) and Chapter 8 (vectors and matrices). The way to read this book, therefore, is to pick a topic, look up its prerequisite in the chart, and then sequence your reading accordingly.

For classroom teaching, I select topics that reflect the interests of the majority of the class, which vary from semester to semester. The most recent semester's syllabus consisted of, in the order of coverage, the following:

- Chapter 7: allocating memory
- Chapter 8: constructing vectors and matrices
- Chapter 18: minimization through the Nelder–Mead simplex method
- Chapter 14: reading and writing digital images
- Chapter 23: using the *Triangle* library to triangulate two-dimensional polygonal domains
- Appendix A: an introduction to barycentric coordinates
- Chapter 24: integration over a triangulated domain
- Chapter 11: storage methods for sparse matrices
- Chapter 12: solving sparse linear systems using the UMFPACK library
- Chapter 25: a finite element method for solving the Poisson equation with zero Dirichlet boundary conditions
- Chapter 22: Gaussian quadrature

- Chapter 26: second order elliptic partial differential equation with arbitrary Dirichlet and Neumann boundary conditions

Naturally the syllabus and its pace should be adjusted to what the students can handle. Parts marked [optional] in a chapter's *Projects* section provide exercises that go beyond minimal learning objectives. Most students voluntarily carry out all the parts, regardless of the [optional] tags.

I should emphasize that neither the course nor this book is a primer on C. Most of my students have had at least one semester of an undergraduate course in C or a C-like low-level procedural programming language, although there have been a few whose prior programming experience has been nothing but MATLAB®, and they have succeeded through hard work, self-study, and some help from me and their classmates. In class I don't dwell on the basics of C programming. I do, however, devote time to pointing out the more subtle programming issues in anticipation of questions that may arise in particular projects. The topics in Part I reflect some of those class presentations. For a self-study, refresher, and reference on the C programming language I recommend Kochan's book [35].

Every program in this book is in full conformance with the 1999 ISO standard C, also known as C99. With minor changes, pointed out in Section 1.3, you may revert them to the 1989 standard, C89, if you so wish. The latest C standard, C11, was announced in 2011, but as of this writing there are no C11 compilers that fully support it; therefore I have avoided special features that were introduced in C11. See Section 1.3 for more on this.

Some of the projects call for supplementary files, mostly consisting of programs or program fragments, which may be obtained from the book's website at

`<www.siam.org/books/cs13/>`.

These supplemental programs are not difficult per se but may require specialized knowledge, such as the detailed syntax of the *PostScript* language or the interface to the *Triangle* library, which I do not wish to make prerequisites for completing the projects. The website also includes additional information, animations/demos, and other miscellany which may be of help with completing the projects.

It is customary in a book's preface to thank those who have been instrumental in bringing the book about. For the present book my thanks go first and foremost to the scores of students who have, over the years, put up with the loose sheets of printed paper which I have distributed weekly in class in lieu of a conventional textbook. I trust that having this book in hand will make for a less stressful—even pleasurable, I hope—learning experience. I am also indebted to the anonymous reviewers whose many constructive ideas and suggestions have been incorporated into the current presentation. Finally, my heartfelt thanks go to SIAM's amazing staff whose enthusiastic support and expert advice in all phases of this book's production have improved the original manuscript by an order of magnitude.

Rouben Rostamian
UMBC, March 2014