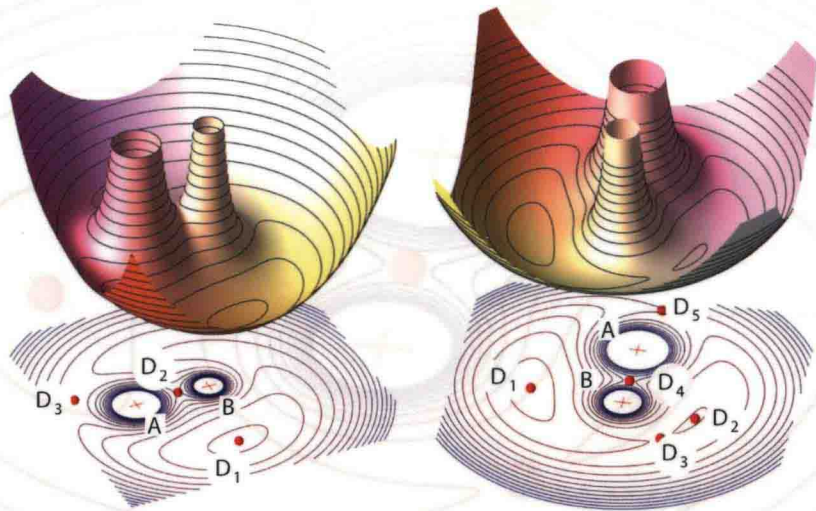


ROUBEN ROSTAMIAN

Programming Projects in C

*for Students of Engineering,
Science, and Mathematics*



siam
Computational Science & Engineering

ROUBEN ROSTAMIAN

University of Maryland, Baltimore County
Baltimore, Maryland

Programming Projects in C
*for Students of Engineering,
Science, and Mathematics*

siam.

Society for Industrial and Applied Mathematics
Philadelphia

Copyright © 2014 by the Society for Industrial and Applied Mathematics

10 9 8 7 6 5 4 3 2 1

All rights reserved. Printed in the United States of America. No part of this book may be reproduced, stored, or transmitted in any manner without the written permission of the publisher. For information, write to the Society for Industrial and Applied Mathematics, 3600 Market Street, 6th Floor, Philadelphia, PA 19104-2688 USA.

Trademarked names may be used in this book without the inclusion of a trademark symbol. These names are used in an editorial context only; no infringement of trademark is intended.

Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Mac is a trademark of Apple Computer, Inc., registered in the United States and other countries. *Programming Projects in C for Students of Engineering, Science, and Mathematics* is an independent publication and has not been authorized, sponsored, or otherwise approved by Apple Computer, Inc.

Maple is a trademark of Waterloo Maple, Inc.

MATLAB is a registered trademark of The MathWorks, Inc. For MATLAB product information, please contact The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098 USA, 508-647-7000, Fax: 508-647-7001, info@mathworks.com, www.mathworks.com.

PostScript is a registered trademark of Adobe Systems Incorporated in the United States and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries.

Figures 15.1 (right image) and 15.2 courtesy of Stockvault.

Figure 19.2 courtesy of the Library of Congress.

Library of Congress Cataloging-in-Publication Data

Rostamian, Rouben, 1949-

Programming projects in C for students of engineering, science, and mathematics /
Rouben Rostamian.

pages cm. – (Computational science and engineering series ; 13)

Includes bibliographical references and index.

ISBN 978-1-611973-49-5

1. Science—Data processing. 2. Engineering—Data processing. 3. Mathematics—Data processing.
4. C (Computer program language) I. Title.

Q183.9R67 2014

502.85'5133—dc23

2014012614

Programming Projects in C

*for Students of Engineering,
Science, and Mathematics*

Computational Science & Engineering

The SIAM series on Computational Science and Engineering publishes research monographs, advanced undergraduate- or graduate-level textbooks, and other volumes of interest to an interdisciplinary CS&E community of computational mathematicians, computer scientists, scientists, and engineers. The series includes both introductory volumes aimed at a broad audience of mathematically motivated readers interested in understanding methods and applications within computational science and engineering and monographs reporting on the most recent developments in the field. The series also includes volumes addressed to specific groups of professionals whose work relies extensively on computational science and engineering.

SIAM created the CS&E series to support access to the rapid and far-ranging advances in computer modeling and simulation of complex problems in science and engineering, to promote the interdisciplinary culture required to meet these large-scale challenges, and to provide the means to the next generation of computational scientists and engineers.

Editor-in-Chief

Donald Estep
Colorado State University

Editorial Board

Omar Ghattas
University of Texas at Austin

Max Gunzburger
Florida State University

Des Higham
University of Strathclyde

Michael Holst
University of California, San
Diego

David Keyes
Columbia University and KAUST

Max D. Morris
Iowa State University

Alex Pothen
Purdue University

Padma Raghavan
Pennsylvania State University

Karen Willcox
Massachusetts Institute
of Technology

Series Volumes

Rostamian, Rouben, *Programming Projects in C for Students of Engineering, Science, and Mathematics*

Smith, Ralph C., *Uncertainty Quantification: Theory, Implementation, and Applications*

Dankowicz, Harry and Schilder, Frank, *Recipes for Continuation*

Mueller, Jennifer L. and Siltanen, Samuli, *Linear and Nonlinear Inverse Problems with Practical Applications*

Shapira, Yair, *Solving PDEs in C++: Numerical Methods in a Unified Object-Oriented Approach, Second Edition*

Borzi, Alfio and Schulz, Volker, *Computational Optimization of Systems Governed by Partial Differential Equations*

Ascher, Uri M. and Greif, Chen, *A First Course in Numerical Methods*

Layton, William, *Introduction to the Numerical Analysis of Incompressible Viscous Flows*

Ascher, Uri M., *Numerical Methods for Evolutionary Differential Equations*

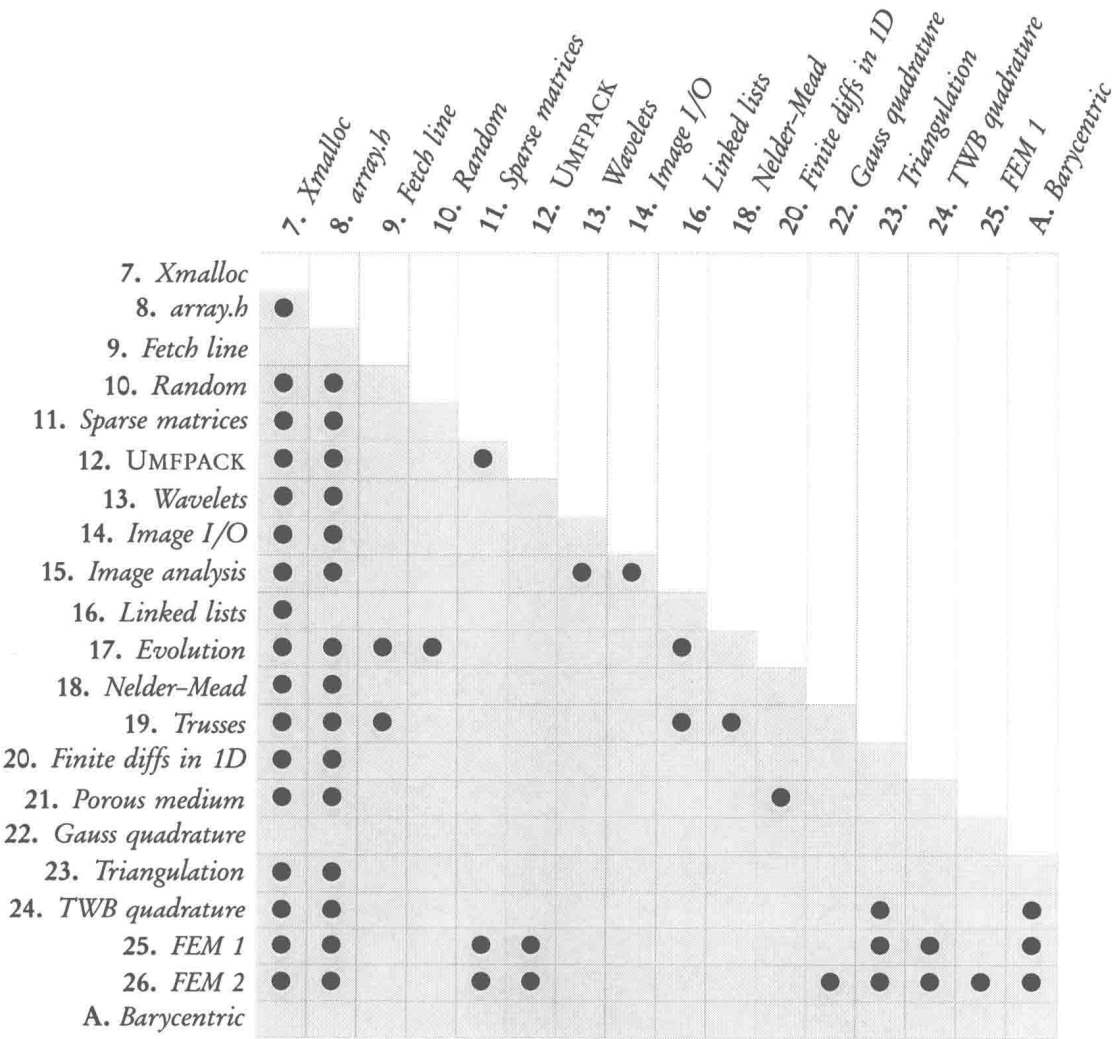
Zohdi, T. I., *An Introduction to Modeling and Simulation of Particulate Flows*

Biegler, Lorenz T., Ghattas, Omar, Heinkenschloss, Matthias, Keyes, David, and van Bloemen Waanders, Bart, Editors, *Real-Time PDE-Constrained Optimization*

Chen, Zhangxin, Huan, Guanren, and Ma, Yuanle, *Computational Methods for Multiphase Flows in Porous Media*

Shapira, Yair, *Solving PDEs in C++: Numerical Methods in a Unified Object-Oriented Approach*

Chapter interdependencies



To find the prerequisites of a chapter, find the chapter title along the left edge, go across horizontally to the bullet marks, and then go vertically to the prerequisite chapters. For instance, **Chapter 23: *Triangulation*** depends on **Chapter 7: *Xmalloc***, and **Chapter 8: *array.h***. Chapters prior to Chapter 7 are not listed; these provide a general background for the entire book and should be considered prerequisites for everything.

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 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

Contents

Chapter interdependencies	xi
Preface	xiii
I A common background	1
1 Introduction	3
1.1 An overview of the book	3
1.2 Why C?	3
1.3 Which version of C?	4
1.4 Operating systems	7
1.5 The compiler and other software	7
1.6 Interfaces and implementations	9
1.7 Advice on writing	11
1.8 Special notations	11
2 File organization	13
3 Streams and the Unix shell	15
4 Pointers and arrays	19
4.1 Pointers	19
4.2 Pointer types	20
4.3 The pointer to void	20
4.4 Arrays	21
4.5 Multidimensional arrays	23
4.6 Strings	24
4.7 The command-line arguments	25
5 From strings to numbers	27
5.1 The function <code>strtod()</code>	27
5.2 The function <code>strtol()</code>	28
5.3 The functions <code>atof()</code> , <code>atol()</code> , and friends	29
6 Make	31
6.1 Multifile programs	31
6.2 Separate compilation and linking	31
6.3 File dependencies	32

6.4	<i>Makefile</i> version 1	34
6.5	How to run make	35
6.6	<i>Makefile</i> version 2	35
6.7	<i>Makefile</i> version 3	36
6.8	<i>Makefile</i> : The final version	38
6.9	Linking with external libraries	40
6.10	Multiple executables in one <i>Makefile</i>	40
II	Projects	43
7	Allocating memory: <i>xmalloc()</i>	45
7.1	Introduction	45
7.2	A review of <i>malloc()</i>	45
7.3	The program	48
7.4	The interface and the implementation	49
7.5	<i>Project Xmalloc</i>	52
8	Dynamic memory allocation for vectors and matrices: <i>array.h</i>	53
8.1	Introduction	53
8.2	Constructing vectors of arbitrary types	54
8.3	A scheme for dynamically allocated matrices	56
8.4	Constructing matrices of arbitrary types	57
8.5	<i>Project array.h</i>	59
9	Reading lines: <i>fetch_line()</i>	63
9.1	Introduction	63
9.2	Reading one line at a time with <i>fgets()</i>	63
9.3	Trimming whitespace and comments	65
9.4	The program	66
9.5	The files <i>fetch-line.[ch]</i>	67
9.6	<i>Project fetch_line</i>	70
10	Generating random numbers	71
10.1	The <i>rand()</i> and <i>srand()</i> functions	71
10.2	Bitmap images	73
10.3	The program	74
10.4	The file <i>random-pbm.c</i>	75
10.5	<i>Project Random Bitmaps</i>	78
11	Storing sparse matrices	79
11.1	Introduction	79
11.2	The CCS format	80
11.3	The program	81
11.4	The files <i>sparse.[ch]</i>	81
11.5	<i>Project Sparse Matrix</i>	82
12	Sparse systems: The UMFPACK library	85
12.1	Introduction	85
12.2	The basics	85
12.3	The program	86

12.4	<i>umfpack-demo1.c</i>	87
12.5	<i>umfpack-demo2.c</i>	89
12.6	<i>umfpack-demo3.c</i> and the triplet form	90
12.7	<i>Project UMFPACK</i>	92
13	Haar wavelets	93
13.1	A brief background	93
13.2	The space $L^2(0,1)$	93
13.3	Haar's construction	94
13.4	The decomposition $V_j = V_{j-1} \oplus W_{j-1}$	97
13.5	From functions to vectors	98
13.6	The Haar wavelet transform	100
13.7	Functions of two variables	102
13.8	An overview of the <i>wavelet</i> module	104
13.9	The file <i>wavelet.h</i>	104
13.10	The file <i>wavelet.c</i>	105
13.11	<i>Project Wavelets</i>	109
14	Image I/O	113
14.1	Digital images and image file formats	113
14.2	Bitmaps and the PBM image format	114
14.3	Grayscale images and the PGM image format	116
14.4	Color images and the PPM image format	117
14.5	The <i>libnetpbm</i> library	118
14.6	A no-frills demo of <i>libnetpbm</i>	121
14.7	The interface of the <i>image-io</i> module	121
14.8	The implementation of the <i>image-io</i> module	124
14.9	The file <i>image-io-test-0.c</i>	129
14.10	<i>Project Image I/O</i>	131
15	Image analysis	135
15.1	Introduction	135
15.2	The truncation error in a grayscale image	137
15.3	The truncation error in a color image	138
15.4	Image reconstruction	138
15.5	The program	139
15.6	The implementation of <i>image-analysis.c</i>	139
15.7	<i>Project Image Analysis</i>	144
16	Linked lists	147
16.1	Linked lists	147
16.2	The program	148
16.3	The function <i>ll_push()</i>	148
16.4	The function <i>ll_pop()</i>	150
16.5	The function <i>ll_free()</i>	151
16.6	The function <i>ll_reverse()</i>	152
16.7	The function <i>ll_sort()</i>	153
16.8	The function <i>ll_filter()</i>	157
16.9	The function <i>ll_length()</i>	159
16.10	<i>Project Linked Lists</i>	159

17	The evolution of species	161
17.1	Introduction	161
17.2	A more detailed description	162
17.3	The <i>World Definition File</i>	165
17.4	The program's user interface	166
17.5	The program's components	167
17.6	The file <i>evolution.h</i>	168
17.7	The files <i>read.[ch]</i>	169
17.8	The files <i>write.[ch]</i>	174
17.9	The files <i>world-to-eps.[ch]</i>	175
17.10	Interlude (and a mini-project)	176
17.11	The file <i>evolution.c</i>	177
17.12	Experiments	188
17.13	Animation	190
17.14	<i>Project Evolution</i>	191
18	The Nelder–Mead downhill simplex	193
18.1	Introduction	193
18.2	The algorithm	193
18.3	Problems with the Nelder–Mead algorithm	197
18.4	An overview of the program	198
18.5	The interface	198
18.6	The implementation	200
18.7	<i>Project Nelder–Mead: Unconstrained optimization</i>	207
18.8	Constrained optimization	211
18.9	<i>Project Nelder–Mead: Constrained optimization</i>	212
18.10	Appendix: Orthogonal projection onto $Ax = b$	213
19	Trusses	215
19.1	Introduction	215
19.2	One-dimensional elasticity	217
19.3	From energy to force	221
19.4	The energy of a truss	222
19.5	From energy to equilibrium	223
19.6	The <i>Truss Description File (TDF)</i>	224
19.7	An overview of the program	226
19.8	The interface	227
19.9	Reading and writing: <i>truss-io.[ch]</i>	230
19.10	The files <i>truss-to-eps.[ch]</i>	240
19.11	Interlude (and a mini-project)	241
19.12	The file <i>truss.c</i>	242
19.13	The file <i>truss-demo.c</i>	247
19.14	<i>Project Truss</i>	249
20	Finite difference schemes for the heat equation in one dimension	251
20.1	The basic idea of finite differences	251
20.2	An explicit scheme for the heat equation	253
20.3	An implicit scheme for the heat equation	256
20.4	The Crank–Nicolson scheme for the heat equation	258
20.5	The Seidman sweep scheme for the heat equation	260

20.6	Test problems	263
20.7	The program	266
20.8	The files <i>problem-spec.[ch]</i>	267
20.9	The file <i>heat-implicit.c</i>	272
20.10	<i>Project Finite Differences in One Dimension</i>	280
21	The porous medium equation	283
21.1	Introduction	283
21.2	Barenblatt's solution	283
21.3	Generalizations	284
21.4	The finite difference scheme	285
21.5	The program	286
21.6	The files <i>problem-spec.[ch]</i>	288
21.7	The file <i>pme-seidman-sweep.c</i>	288
21.8	<i>Project Porous Medium</i>	289
21.9	Appendix: The porous medium equation as a population dynamics model	289
22	Gaussian quadrature	291
22.1	Introduction	291
22.2	Lagrange interpolation	292
22.3	Legendre polynomials	294
22.4	The Gaussian quadrature formula	295
22.5	The program	296
22.6	The files <i>gauss-quad.[ch]</i>	297
22.7	<i>Project Gaussian Quadrature</i>	299
23	Triangulation with the <i>Triangle</i> library	301
23.1	Introduction	301
23.2	The program	302
23.3	The file <i>problem-spec.h</i>	303
23.4	The file <i>problem-spec.c</i>	305
23.5	The files <i>mesh.h</i> and <i>mesh.c</i>	311
23.6	The file <i>mesh-demo.c</i>	313
23.7	Installing <i>Triangle</i>	315
23.8	<i>Project Triangulate</i>	316
24	Integration on triangles	319
24.1	Introduction	319
24.2	The Taylor, Wingate, and Bos (TWB) quadrature	321
24.3	The files <i>twb-quad.[ch]</i>	322
24.4	The program	326
24.5	The files <i>plot-with-geomview.[ch]</i>	328
24.6	Modifying the file <i>problem-spec.c</i>	329
24.7	The file <i>twb-quad-demo.c</i>	330
24.8	<i>Project TWB Quadrature</i>	334
25	Finite elements	337
25.1	The Poisson equation	337
25.2	The weak formulation	338
25.3	The Galerkin approximation	340

25.4	An overview of the FEM	341
25.5	Error analysis	345
25.6	The program	347
25.7	Changes in <i>problem-spec.c</i>	349
25.8	The file <i>poisson.h</i>	350
25.9	The file <i>poisson.c</i>	351
25.10	The file <i>fem-demo.c</i>	358
25.11	Further reading	359
25.12	<i>Project FEM 1</i>	360
26	Finite elements: Nonzero boundary data	361
26.1	The problem	361
26.2	The weak formulation	362
26.3	The Galerkin approximation	364
26.4	The program	366
26.5	The file <i>problem-spec.[ch]</i>	366
26.6	The file <i>poisson.c</i>	369
26.7	<i>Project FEM 2</i>	373
A	Barycentric coordinates	375
A.1	Barycentric coordinates	375
A.2	Calculus on a triangle	377
	Bibliography	381
	Index	387