An Introduction to Parallel and Vector Scientific Computing

In this text, students of applied mathematics, science, and engineering are introduced to fundamental ways of thinking about the broad context of parallelism. The authors begin by giving the reader a deeper understanding of the issues through a general examination of timing, data dependencies, and communication. These ideas are implemented with respect to shared memory, parallel and vector processing, and distributed memory cluster computing. Threads, OpenMP, and MPI are covered, along with code examples in Fortran, C, and Java.

The principles of parallel computation are applied throughout as the authors cover traditional topics in a first course in scientific computing. Building on the fundamentals of floating point representation and numerical error, a thorough treatment of numerical linear algebra and eigenvector/eigenvalue problems is provided. By studying how these algorithms parallelize, the reader is able to explore parallelism inherent in other computations, such as Monte Carlo methods.

Ronald W. Shonkwiler is a professor in the School of Mathematics at the Georgia Institute of Technology. He has authored or coauthored more than 50 research papers in areas of functional analysis, mathematical biology, image processing algorithms, fractal geometry, neural networks, and Monte Carlo optimization methods. His algorithm for monochrome image comparison is part of a U.S. patent for fractal image compression. He has coauthored two other books, *An Introduction to the Mathematics of Biology* and *The Handbook of Stochastic Analysis and Applications*.

Lew Lefton is the Director of Information Technology for the School of Mathematics and the College of Sciences at the Georgia Institute of Technology, where he has built and maintained several computing clusters that are used to implement parallel computations. Prior to that, he was a tenured faculty member in the Department of Mathematics at the University of New Orleans. He has authored or coauthored more than a dozen research papers in the areas of nonlinear differential equations and numerical analysis. His academic interests are in differential equations, applied mathematics, numerical analysis (in particular, finite element methods), and scientific computing.

Cambridge Texts in Applied Mathematics

All titles listed below can be obtained from good booksellers or from Cambridge University Press.
For a complete series listing, visit http:publishing.cambridge.org/stm/mathematics/ctam

Thinking about Ordinary Differential Equations
Robert E. O'Malley Jr

A Modern Introduction to the Mathematical Theory of Water Waves
Robin Stanley Johnson

Rarefied Gas Dynamics: From Basic Concepts to Actual Calculations
Carlo Cercignani

Symmetry Methods for Differential Equations: A Beginner's Guide
Peter E. Hydon

High Speed Flow
C. J. Chapman

Wave Motion
J. Billingham, and A. C. King

An Introduction to Magnetohydrodynamics
P. A. Davidson

Linear Elastic Waves
John G. Harris

Vorticity and Incompressible Flow
Andrew J. Majda, and Andrea L. Bertozzi

Infinite-Dimensional Dynamical Systems
James C. Robinson

Introduction to Symmetry Analysis
Brain J. Cantwell

Bäcklund and Darboux Transformations
C. Rogers, and W. K. Schief

Finite Volume Methods for Hyperbolic Problems
Randall J. LeVeque

Introduction to Hydrodynamic Stability
P. G. Drazin

Theory of Vortex Sound
M. S. Howe

Scaling
Grigory Isaakovich Barenblatt

Complex Variables: Introduction and Applications (2nd Edition)
Mark J. Ablowitz, and Athanassios S. Fokas

A First Course in Combinatorial Optimization
Jone Lee

Practical Applied Mathematics: Modelling, Analysis, Approximation
Sam Howison

An Introduction to Parallel and Vector Scientific Computing

RONALD W. SHONKWILER
Georgia Institute of Technology

LEW LEFTON
Georgia Institute of Technology

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
32 Avenue of the Americas, New York, NY 10013-2473, USA

www.cambridge.org
Information on this title: www.cambridge.org/9780521864787

First published 2006

Printed in the United States of America

A catalog record for this publication is available from the British Library.

Library of Congress Cataloging in Publication Data
Shonkwiler, Ronald W., 1942–
An introduction to parallel and vector scientific computing / by R. Shonkwiler and L. Lefton.
p. cm. – (Cambridge texts in applied mathematics ; no. 41)
Includes bibliographical references and index.
ISBN-13: 978-0-521-86478-7
ISBN-10: 0-521-86478-X
ISBN-13: 978-0-521-68337-1 (pbk.)
ISBN-10: 0-521-68337-8 (pbk.)
1. Parallel processing (Electronic computers) 2. Vector processing (Computer science)
I. Lefton, L. (Lew), 1960– II. Title.
III. Series: Cambridge texts in applied mathematics ; 41.
QA76.58.S545 2006
004.3′5–dc22 2006007798

ISBN-13: 978-0-521-86478-7 hardback
ISBN-10: 0-521-86478-X hardback

ISBN-13: 978-0-521-68337-1 paperback
ISBN-10: 0-521-68337-8 paperback

Contents

Preface

Numerical computations are a fundamental tool for engineers and scientists. The current practice of science and engineering demands that nontrivial computations be performed with both great speed and great accuracy. More and more, one finds that scientific insight and technologial breakthroughs are preceded by intense computational efforts such as modeling and simulation. It is clear that computing is, and will continue to be, central to the further development of science and technology.

As market forces and technological breakthroughs lowered the cost of computational power by several orders of magintude, there was a natural migration from large-scale mainframes to powerful desktop workstations. Vector processing and parallelism became possible, and this parallelism gave rise to a new collection of algorithms. Parallel architectures matured, in part driven by the demand created by the algorithms. Large computational codes were modified to take advantage of these parallel supercomputers. Of course, the term *supercomputer* has referred, at various times, to radically different parallel architectures. This includes vector processors, various shared memory architectures, distributed memory clusters, and even computational grids. Although the landscape of scientific computing changes frequently, there is one constant; namely, that there will always be a demand in the research community for high-performance computing.

When computations are first introduced in beginning courses, they are often straightforward "vanilla" computations, which are well understood and easily done using standard techniques and/or commercial software packages on desktop computers. However, sooner or later, a working scientist or engineer will be faced with a problem that requires advanced techniques, more specialized software (perhaps coded from scratch), and/or more powerful hardware. This book is aimed at those individuals who are taking that step, from a novice to intermediate or even from intermediate to advanced user of tools that fall under

the broad heading of scientific computation. The text and exercises have been shown, over many years of classroom testing, to provide students with a solid foundation, which can be used to work on modern scientific computing problems. This book can be used as a guide for training the next generation of computational scientists.

This manuscript grew from a collection of lecture notes and exercises for a senior-level course entitled "Vector and Parallel Scientific Computing." This course runs yearly at the Georgia Institute of Technology, and it is listed in both mathematics and computer science curricula. The students are a mix of math majors, computer scientists, all kinds of engineers (aerospace, mechanical, electrical, etc.), and all kinds of scientists (chemists, physicists, computational biologists, etc.). The students who used these notes came from widely varying backgrounds and varying levels of expertise with regard to mathematics and computer science.

Formally, the prerequisite for using this text is knowledge of basic linear algebra. We integrate many advanced matrix and linear algebra concepts into the text as the topics arise rather than offering them as a separate chapter. The material in Part II, Monte Carlo Methods, also assumes some familiarity with basic probability and statistics (e.g., mean, variance, t test, Markov chains).

The students should have some experience with computer programming. We do not teach nor emphasize a specific programming language. Instead, we illustrate algorithms through a pseudocode, which is very close to mathematics itself. For example, the mathematical expression $y = \sum_{i=1}^{n} x_i$ becomes

```
y=0;
loop i = 1 upto n
    y = y + x_i;
end loop
```

We provide many example programs in Fortran, C, and Java. We also have examples of code that uses MPI libraries. When this course was originally taught, it took several weeks for the students to get accounts and access to the Cray system available at that time. As a result, the material in the first two chapters provides no programming exercises. If one wishes to start programming right away, then he or she should begin with Chapter 3.

The purpose of the course is to provide an introduction to important topics of scientific computing including the central algorithms for numerical linear algebra such as linear system solving and eigenvalue calculation. Moreover, we introduce this material from the very beginning in the context of vector and parallel computation. We emphasize a recognition of the sources and propagation of numerical error and techniques for its control. Numerical error starts with

the limitations inherent in the floating point representation of numbers leading to round-off error and continues with algorithmic sources of error.

The material has evolved over time along with the machines called supercomputers. At present, shared memory parallel computation has standardized on the threads model, and vector computation has moved from the machine level to the chip level. Of course, vendors provide parallelizing compilers that primarily automatically parallelize loops that the programmer has requested, sometimes referred to as the DOACROSS model. This is a convenient model for engineers and scientists as it allows them to take advantage of parallel and vector machines while making minimal demands on their programming time. For the purpose of familiarily, we include a section on the basic concepts of distributed memory computation, including topological connectivity and communication issues.

In teaching the course, we employ a hands-on approach, requiring the students to write and execute programs on a regular basis. Over the years, our students have had time on a wide variety of supercomputers, first at National Centers, and more recently at campus centers or even on departmental machines. Of course, even personal computers today can be multiprocessor with a vector processing chipset, and many compiled codes implement threads at the operating system level.

We base our approach to parallel computation on its representation by means of a directed acyclic graph. This cuts to the essence of the computation and clearly shows its parallel structure. From the graph it is easy to explain and calculate the complexity, speedup, efficiency, communication requirements, and scheduling of the computation. And, of course, the graph shows how the computation can be coded in parallel.

The text begins with an introduction and some basic terminology in Chapter 1. Chapter 2 gives a high-level view of the theoretical underpinnings of parallelism. Here we discuss data dependencies and complexity, using directed acyclic graphs to more carefully demonstate a general way of thinking about parallelism. In Chapter 3, we have included a variety of machine implementations of parallelism. Although some of these architectures are not in widespread use any more (e.g., vector processors like the early Cray computers), there are still interesting and important ideas here. In fact, the Japanese Earth Simulator (the former world record holder for "fastest computer") makes heavy use of vector processing and pipelining. Chapter 3 includes an introduction to low-level implementations of parallelism by including material on barriers, mutexes, and threads. Of course, not every scientific computing application will require thread programming, but as mentioned earlier, these objects provide many useful ideas about parallelization that can be generalized to many different parallel codes.

We have even included a short introduction to quantum computing because this technology may one day be the future of parallel scientific computation.

In the second half of the book, we start with basic mathematical and computational background, presented as building blocks in Chapter 4. This includes material on floating point numbers, round-off error, and basic matrix arithmetic. We proceed to cover mathematical algorithms, which we have found are most frequently used in scientific computing. Naturally, this includes a large measure of numerical linear algebra. Chapters 5, 6, and 7 discuss direct methods for solving linear systems. We begin with classical Gaussian elimination and then move on to matrices with special structure and more advanced topics such as Cholesky decomposition and Givens' rotation. Iterative methods are covered in Chapter 8. We study Jacobi and Gauss-Seidel as well as relaxtion techniques. This chapter also includes a section on conjugate gradient methods. In Chapter 9, we examine eigenvalues and eigenvectors. This includes the power method and QR decomposition. We also cover the topics of Householder transformations and Hessenberg forms, since these can improve QR computations in practice.

Throughout all of Part II, our development of linear algebraic results relies heavily on the technique of partitioning matrices. This is introduced in Chapter 4 and continues through our presentation of Jordan form in Chapter 9.

The final section of the book is focused on Monte Carlo methods. We first develop classical quadrature techniques such as the Buffon Needle Problem in Chapter 10. We then advance in Chapter 11 to a presentation of Monte Carlo optimization, which touches on the ideas of simulated annealing, genetic algorithms, and iterated improvement with random restart.

Exercises are included at the end of every section. Some of these are meant to be done by hand, and some will require access to a computing environment that supports the necessary parallel architecture. This could be a vector machine, an SMP system supporting POSIX threads, a distributed memory cluster with MPI libraries and compilers, etc. We have attempted to isolate those exercises that require programming in a subsection of each exercise set. Exercises are followed by a number in parentheses, which is meant to be an indication of the level of difficulty.

Because scientific computing is often the result of significant research efforts by large distributed teams, it can be difficult to isolate meaningful self-contained exercises for a textbook such as this. We have found it very useful for students to work on and present a project as a substantial part of their course grade. A 10-minute oral presentation along with a written report (and/or a poster) is an excellent exercise for students at this level. One can ask them to submit a short project proposal in which they briefly describe the problem background, the

mathematical problem that requires computation, and how this computation may parallelize. Students do well when given the opportunity to perform a deeper study of a problem of interest to them.

Acknowledgments

Ron Shonkwiler would like to thank his coauthor, Dr. Lew Lefton, for bringing thoughtfulness and a fresh outlook to this project and for his "way with words," and Lauren Cowles of Cambridge for her encouragement. Most of all, he would like to thank the many students who have taken the course on which this book is based. Their enthusiasm, creativity, energy, and industry have made teaching the course a challenge and a pleasure.

Lew Lefton would first and foremost like to express his deepest gratitude to his coauthor, Dr. Ron Shonkwiler. When they began collaborating on the project of getting the course notes into a book manuscript several years ago, Lefton naively thought it would be an interesting but straightforward project. He was half right; it has been very interesting. The notes have changed (improved!) signifcantly since that early draft and, having taught the course himself and gone through the process of publishing a book, he feels very fortunate to have had Ron, with his expertise and experience, as a guide. Lefton would also like to thank his wife, Dr. Enid Steinbart, and his daughters, Hannah, Monica, and Natalie, who put up with him spending more than his usual amount of "screen time" in front of the computer. He is truly grateful for their love and support.