Principles of Scientific Computing
Preface

Jonathan Goodman

January 2, 2003

Last revised: **January 2, 2003**

This book grew out of a course in Scientific Computing for graduate students at New York University, a first course that covers the basic principles common to most applications. It addresses the question: What are the basics that everyone doing scientific computing needs to know? The answer is: some mathematics, the most basic algorithms, a bit about the workings of a computer, and an idea how to build software for scientific computing applications. Naturally, specific applications (e.g. computing stresses in a bridge) also require more advanced specific material (enginering mechanics, finite elements, etc.).

The principles of scientific computing are a collection of simple, almost obvious, ideas and points of view. The practitioner is hardly a cook relying printed recipies or downoaded software, but a creative problem solver who can devise algorithms and build trustworthy software for new computational challenges. I hope that the reader will come to share my delight in the simplicity and admiration of the power of these simple principles.

This book requires a facility with the mathematics that is common to most quantitative modeling: multivariate calculus, linear algebra, and basic probability. These subjects may be reviewed using books in the *Schaum's Outline* series, particularly, *Multivariate Calculus*, *Linear Algebra*, and *Probability.* The exercises require programming in C or C++. The differences between these are not so important for the small simple programs called for. The book is structured so that an ambitious student can learn programming as he or she goes. See Appendix I. It is possible to do the programming in Fortran, but students are discouraged from using a programming language, such as Java, Visual Basic, or Matlab, not designed for efficient large scale scientific computing.

Visualization and data analysis are an essential part of scientific computing. We recommend Matlab as a simple and reliable system for visualization and examination of computational results. Students familiar with other scientific visualization software are welcome to use it. However, we warn the student that Mathematica is often unreliable. Excel users will need to be sophisticated enough to turn off the default features intended for business presentations, such as shading on bar graphs.

Many students will want to compute on their personal computer. Any current laptop or desktop should be powerful enough. The student will need a C/C++ compiler and visualization software such as Matlab. Some of the exercises require the student to download files and software, but always plain text files or C/C++ source code.

Many of my views on scientific computing were formed during my association with the remarkable group of faculty and graduate students at Serra House, the numerical analysis group of the Computer Science Department of Stanford University, in the early 1980's. I mention in particularly Marsha Berger, Petter Björstad, Bill Coughran, Gene Golub, Bill Gropp, Eric Grosse, Bob Higdon, Randy LeVeque, Steve Nash, Joe Oliger, Michael Overton, Nick Trefethen, and Margaret Wright. Colleagues at the Courant Institute who have influenced this book include Leslie Greengard, Gene Isaacson, Peter Lax, Charlie Peskin, Luis Reyna, Mike Shelley, and Olof Widlund. I also acknowledge the lovely book

Numerical Methods by Germund Dahlquist and Åke Björk.