

Preface

We are conducting ever more complex computations built upon the assumption that the underlying numerical methods are mature and reliable.

When we bundle existing algorithms into libraries and wrap them into packages to facilitate easy use, we create de facto standards that make it easy to ignore numerical analysis.

John Guckenheimer, president SIAM, in *SIAM News*, June 1998: *Numerical Computation in the Information Age*

When redrafting the book I was tempted to present the algorithms in ALGOL, but decided that the difficulties of providing procedures which were correct in every detail were prohibitive at this stage.

James Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, 1988.

This book is an introduction to *scientific computing*, the mathematical modeling in science and engineering and the study of how to exploit computers in the solution of technical and scientific problems. It is based on mathematics, numerical and symbolic/algebraic computations, parallel/distributed processing and visualization. It is also a popular and growing area — many new curricula in *computational science and engineering* have been, and continue to be, developed, leading to new academic degrees and even entire new disciplines.

A prerequisite for this development is the ubiquitous presence of computers, which are being used by virtually every student and scientist. While traditional scientific work is based on developing theories and performing experiments, the possibility to use computers at any time has created a third way of increasing our knowledge, which is through modeling and simulation. The use of simulation is further facilitated by the availability of sophisticated, robust and easy-to-use software libraries. This has the obvious advantage of shielding the user from the underlying numerics; however, this also has the danger of leaving the user unaware of the limitations of the algorithms, which can lead to incorrect results when used improperly. Moreover, some algorithms can be fast for certain types of problems but highly inefficient for others. Thus, it is important for the user to be able to make an informed decision on which algorithms to use, based on the properties of the problem to be solved. The goal of this book is to familiarize the reader with the basic

concepts of scientific computing and algorithms that form the workhorses of many numerical libraries. In fact, we will also emphasize the effective implementation of the algorithms discussed.

Numerical scientific computing has a long history; in fact, computers were first built for this purpose. *Konrad Zuse* [154] built his first (mechanical) computer in 1938 because he wanted to have a machine that would solve systems of linear equations that arise, e.g., when a civil engineer designs a bridge. At about the same time (and independently), *Howard H. Aiken* wanted to build a machine that would solve systems of ordinary differential equations [17].

The first high quality software libraries contained indeed numerical algorithms. They were produced in an international effort in the programming language ALGOL 60 [111], and are described in the handbook “Numerical Algebra” [148]. These fundamental procedures for solving linear equations and eigenvalue problems were developed further, rewritten in FORTRAN, and became the LINPACK [26] and EISPACK [47] libraries. They are still in use and available at www.netlib.org from Netlib. In order to help students to use this software, Cleve Moler created around 1980 a friendly interface to those subroutines, which he called MATLAB (Matrix Laboratory). MATLAB was so successful that a company was founded: MathWorks. Today, MATLAB is “the language of technical computing”, a very powerful tool in scientific computing.

Parallel to the development of numerical libraries, there were also efforts to do exact and algebraic computations. The first computer algebra systems were created some 50 years ago: At ETH, Max Engeli created SYMBAL, and at MIT, Joel Moses MACSYMA. MACSYMA is the oldest system that is still available. However, computer algebra computations require much more computer resources than numerical calculations. Therefore, only when computers became more powerful did these systems flourish. Today the market leaders are MATHEMATICA and MAPLE.

Often, a problem may be solved analytically (“exactly”) by a computer algebra system. In general, however, analytical solutions do not exist, and numerical approximations or other special techniques must be used instead. Moreover, computer Algebra is a very powerful tool for deriving numerical algorithms; we use MAPLE for this purpose in several chapters of this book. Thus, computer algebra systems and numerical libraries are complementary tools: working with both is essential in scientific computing. We have chosen MATLAB and MAPLE as basic tools for this book. Nonetheless, we are aware that the difference between pure computer algebra systems and numerical MATLAB-like systems is disappearing, and the two may merge and become indistinguishable by the user in the near future.

How to use this book

Prerequisites for understanding this book are courses in calculus and linear algebra. The content of this book is too much for a typical one semester course in scientific computing. However, the instructor can choose those sections that he wishes to teach and that fit his schedule. For example, for an introductory course in scientific computing, one can very well use the least squares chapter and teach only one of the methods for computing the QR decomposition. However, for an advanced course focused solely on least squares methods, one may also wish to consider the singular value decomposition (SVD) as a computational tool for solving least squares problems. In this case, the book also provides a detailed description on how to compute the SVD in the chapter on eigenvalues. The material is presented in such a way that a student can also learn directly from the book. To help the reader navigate the volume, we provide in section 1.2 some sample courses that have been taught by the authors at various institutions.

The focus of the book is algorithms: we would like to explain to the students how some fundamental functions in mathematical software are designed. Many exercises require programming in MATLAB or MAPLE, since we feel it is important for students to gain experience in using such powerful software systems. They should also know about their limitations and be aware of the issue addressed by John Guckenheimer. We tried to include meaningful examples and problems, not just academic exercises.

Acknowledgments

The authors would like to thank Oscar Chinellato, Ellis Whitehead, Oliver Ernst and Laurence Halpern for their careful proofreading and helpful suggestions.

Walter Gander is indebted to Hong Kong Baptist University (HKBU) and especially to its Vice President Academic, Franklin Luk, for giving him the opportunity to continue to teach students after his retirement at ETH. Several chapters of this book have been presented and improved successfully in courses at HKBU. We are also thankful to the University of Geneva, where we met many times to finalize the manuscript.

Geneva and Zürich, August 2013

Walter Gander, Martin J. Gander, Felix Kwok

Scientific Computing - An Introduction using Maple and
MATLAB

Gander, W.; Gander, M.J.; Kwok, F.

2014, XVIII, 905 p. 133 illus., 53 illus. in color.,

Hardcover

ISBN: 978-3-319-04324-1