THE IMA VOLUMES
IN MATHEMATICS
AND ITS APPLICATIONS VOLUME 14

J.R. Rice
Editor

Mathematical Aspects of Scientific Software

Springer-Verlag

J.R. Rice
Editor

Mathematical Aspects of Scientific Software

With 46 Illustrations

Springer-Verlag
New York Berlin Heidelberg
London Paris Tokyo

J.R. Rice
Department of Computer Science
Purdue University
West Lafayette, IN 47907, USA

Mathematics Subject Classification (1980): 68Q20

Library of Congress Cataloging-in-Publication Data
Mathematical aspects of scientific software.
(The IMA volumes in mathematics and its applications; v. 14)
Bibliography: p.
1. Computer software—Development. 2. Science—Data processing. I. Rice, John Richard. II. Series.
QA76.76.D47M366 1988 502'.85'53 88-3237

ISBN 0-387-96706-0/1988 $0.00 + 0.20.

Camera-ready text prepared by the editor.
Printed and bound by Edwards Brothers Inc., Ann Arbor, Michigan.
Printed in the United States of America.

9 8 7 6 5 4 3 2 1

ISBN 0-387-96706-0 Springer-Verlag New York Berlin Heidelberg
ISBN 3-540-96706-0 Springer-Verlag Berlin Heidelberg New York

The IMA Volumes in Mathematics and Its Applications

Volume 14

Series Editors
Hans Weinberger Willard Miller, Jr.

Institute for Mathematics and Its Applications
IMA

The **Institute for Mathematics and Its Applications** was established by a grant from the National Science Foundation to the University of Minnesota in 1982. The IMA seeks to encourage the development and study of fresh mathematical concepts and questions of concern to the other sciences by bringing together mathematicians and scientists from diverse fields in an atmosphere that will stimulate discussion and collaboration.

The IMA Volumes are intended to involve the broader scientific community in this process.

Hans Weinberger, Director
Willard Miller, Jr., Associate Director

IMA Programs

1982–1983 Statistical and Continuum Approaches to Phase Transition

1983–1984 Mathematical Models for the Economics of Decentralized Resource Allocation

1984–1985 Continuum Physics and Partial Differential Equations

1985–1986 Stochastic Differential Equations and Their Applications

1986–1987 Scientific Computation

1987–1988 Applied Combinatorics

1988–1989 Nonlinear Waves

1989–1990 Dynamical Systems and Their Applications

Springer Lecture Notes from the IMA

The Mathematics and Physics of Disordered Media
Editors: Barry Hughes and Barry Ninham
(Lecture Notes in Mathematics, Volume 1035, 1983)

Orienting Polymers
Editor: J. L. Ericksen
(Lecture Notes in Mathematics, Volume 1063, 1984)

New Perspectives in Thermodynamics
Editor: James Serrin
(Springer-Verlag, 1986)

Models of Econoic Dynamics
Editor: Hugo Sonnenschein
(Lecture Notes In Economics, Volume 264, 1986)

The IMA Volumes in Mathematics and Its Applications

Current Volumes:

Volume 1: Homogenization and Effective Moduli of Materials and Media
Editors: Jerry Ericksen, David Kinderlehrer, Robert Kohn, and J.-L. Lions
Volume 2: Oscillation Theory, Computation, and Methods of Compensated Compactness
Editors: Constantine Dafermos, Jerry Ericksen, David Kinderlehrer, and Marshall Slemrod
Volume 3: Metastability and Incompletely Posed Problems
Editors: Stuart Antman, Jerry Ericksen, David Kinderlehrer, and Ingo Müller
Volume 4: Dynamical Problems in Continuum Physics
Editors: Jerry Bona, Constantine Dafermos, Jerry Ericksen, and David Kinderlehrer
Volume 5: Theory and Application of Liquid Crystals
Editors: Jerry Erickson and David Kinderlehrer
Volume 6: Amorphous Polymers and Non-Newtonian Fluids
Editors: Constantine Dafermos, Jerry Ericksen, and David Kinderlehrer
Volume 7: Random Media
Editor: George Papanicolaou
Volume 8: Percolation Theory and Ergodic Theory of Infinite Particle Systems
Editor: Harry Kesten
Volume 9: Hydrodynamic Behavior and Interacting Particle Systems
Editor: George Papanicolaou
Volume 10: Stochastic Differential Systems, Stochastic Control Theory and Applications
Editors: Wendell Fleming and Pierre-Louis Lions
Volume 11: Numerical Simulation in Oil Recovery
Editor: Mary Fanett Wheeler
Volume 12: Computational Fluid Dynamics and Reacting Gas Flows
Editors: Bjorn Engquist, Mitchell Luskin, and Andrew Majda
Volume 13: Numerical Algorithms for Modern Parallel Computer Architectures
Editor: Martin Schultz
Volume 14: Mathematical Aspects of Scientific Software
Editor: J.R. Rice

Forthcoming Volumes:

1986–1987: *Scientific Computation*

The Modeling of Fractures, Heterogeneities, and Viscous Fingering in Flow in Porous Media
Computational Fluid Dynamics and Reacting Gas Flows
Numerical Algorithms for Modern Parallel Computer Architectures
Atomic and Molecular Structure and Dynamics

FOREWORD

This IMA Volume in Mathematics and its Applications

MATHEMATICAL ASPECTS OF SCIENTIFIC SOFTWARE

is in part the proceedings of a workshop which was an integral part of the 1986-87 IMA program on SCIENTIFIC COMPUTATION. We are grateful to the Scientific Committee: Bjorn Engquist (Chairman), Roland Glowinski, Mitchell Luskin and Andrew Majda for planning and implementing an exciting and stimulating year-long program. We especially thank the Workshop Organizer, John R. Rice for organizing a workshop which brought together many of the major figures in a variety of research fields connected with scientific software for a fruitful exchange of ideas.

Willard Miller, Jr.

Hans Weinberger

PREFACE

Scientific software is the fuel that drives today's computers to solve a vast range of problems. Huge efforts are being put into developing new software, new systems and new algorithms for scientific problem solving. The ramifications of this effort echo throughout science and, in particular, into mathematics. This book explores how scientific software impacts the structure of mathematics, how it creates new subfields and how new classes of mathematical problems arise.

The focus is on five topics where the impact is currently being felt and where important new challenges for mathematics exist. These topics are the new subfields of parallel and geometric computations, the emergence of symbolic computation systems into "general" use, the potential emergence of new, high-level mathematical systems, and the crucial question of how to measure the performance of mathematical problem solving tools.

This workshop brought together research workers from the borders between mathematics and computer science, people who were able to see and discuss the interactions between mathematics and its applications. The editor greatly appreciates the efforts made by the authors and other participants in the workshop. Special thanks are due to Carl de Boor, Clarence Lehman, Bradley Lucier and Richard McGehee who gave stimulating and insightful panel presentations on the nature and needs for high level mathematical systems.

CONTENTS

MATHEMATICAL ASPECTS OF SCIENTIFIC SOFTWARE

JOHN R. RICE*
Department of Computer Science
Purdue University
West Lafayette, Indiana 47907

Abstract

The goal is to survey the impact of scientific software on mathematics. Three types of impacts are identified and several topics from each are discussed in some depth. First is the impact on the structure of mathematics through its role as the scientific tool for problem solving. Scientific software leads to new assessments of what algorithms are, how well they work, and what a solution really is. Second is the initiation of new mathematical endeavors. Numerical computation is already very widely known, we discuss the important future roles of symbolic and geometric computation. Finally, there are particular mathematical problems that arise from scientific software. Examples discussed include round-off errors and the validation of computations, mapping problems and algorithms into machines, and adaptive methods. There is considerable discussion of the shortcommings of mathematics in providing an adequate model for the scientific analysis of scientific software.

1. The Impact of Scientific Software on Mathematics

The goal of this paper is to survey the impact of scientific software on mathematics. Three areas are identified:

1) *The effect on the structure of mathematics*, on what mathematicians do and how they view their activities,
2) *New mathematical endeavors that arise*, new specialities or subspecialities of mathematics that may be created,

*This work supported in part by the Air Force Office of Scientific Research grant 84-0385.

3) *Mathematical problems that arise*, difficult mathematical problems or groups of problems arise from efforts to understand certain methods or phenomena of scientific software.

About 15 topics are presented which illustrate these impacts. No attempt has been made to be encyclopedic, the choices are those that appeal to the author. This introductory section is a summary of the survey, about a dozen of the topics are discussed in more depth in the later sections of this paper and only mentioned here. A few topics not discussed later are included here with a few remarks.

In considering the structure of mathematics, it is important to realize that not only does mathematics grow but that it also changes nature. Large subfields die out and not just because all the problems are solved or questions answered. There subfields become irrelevant to new directions that mathematics take. An example of this exists in scientific computation, namely making tables of mathematical functions. This endeavor started almost in the antiquity of mathematics and grew until there were a large number of practitioners. Volumes and volumes of tables were prepared and the methodology of creating, checking and making them easy to use became quite sophisticated. This endeavor is now in a steep decline because computers and scientific software have made it easier and more reliable to compute values from "first principles" than to look them up in tables.

Scientific software will lead mathematics to focus again more heavily on *problem solving and algorithms*. We need to analyze the intrinsic nature of problems, how hard they are to solve and the strengths of classes of algorithms. We need to examine again what it means to solve a problem, a step that will show many previous "solutions" to be of little value.

We need to examine again what it means to prove a result, and what techniques are reliable. Some groups in computer science have a substantially different view of proof then modern mathematical practice. They view proofs much more formally, somewhat reminiscent of Russell and Whitehead's Principia Mathematica. If computer programs are going to prove theorems, how does one prove the programs themselves are correct? For a delightful and insightful analysis of the pitfalls here, see [Davis, 1972]. I believe that we have learned several important things about proofs and scientific software. First, scientific software is not amenable to proofs as a whole because it contains many heuristics (i.e., algorithm fragments for which there is no underlying formal model, we just hope they work). Second, it is very difficult, often impossible, to say what scientific

software is supposed to do. Finally, the computing requirements for this approach are truly enormous.

We discuss the impact on mathematics of the creation and widespread use of *mathematical systems*. This development is now overdue and will have an enormous impact on education and practice of mathematics. It is plausible that one can automate large parts (the algorithmic parts) of mathematics from the middle elementary years to the middle undergraduate years. The results will be much more reliable abilities, less cost, more power for problem solving and more time to learn the mysteries of problem solving rather than the rote.

The largest and most visible new mathematical endeavor resulting from scientific software is that of *numerical computation*. It has a lot of structure and activity. The principle components are *numerical analysis*, a large subfield of mathematics and computer science, *mathematical software*, a smaller subfield of computer science, a part of *applied mathematics*, and *computational analysis*, a huge subfield of science and engineering. This endeavor has not yet had a large impact on most of mathematics. Perhaps this is because mathematics has turned away from problem solving and has been content to let other disciplines appropriate this endeavor. As the attention of mathematics is turned toward problem solving, interest in this area will rise substantially.

Sections 6 to 8 discuss the newer endeavors of *symbolic computation* and *geometric computation*. These are currently much smaller than numerical computation but may have a more immediate impact on mathematics because they are closer to currently active areas. Geometric computation is still quite immature and offers a host of challenges for mathematics.

Perhaps the best known problem area arising from scientific software is that of *round-off error analysis*. In 1948 John von Newmann and Herbert Goldstine pursued the idea of following the effect of individual errors and bounding the results. This is very tedious and overall this approach is a failure. A second and more subtle idea is to estimate the change in the problem data so the computed result is exact. This is often very practical and it gives very useful information, once one accepts that a realistic estimate of the error due to round-off is not going to be obtained. The third and most widely used idea is to do all computations with much higher precision than is thought to be required. One can even use exact arithmetic. This approach is quite, but not extremely, reliable. It is also expensive. The final approach has been to introduce *condition numbers*, these are essentially norms of the Frechet derivative of the solution with respect to the data. Note

that these do not take into account actual round-off errors but rather estimate the uncertainty of the solution in terms of the uncertainty of the problem data.

Round-off error analysis is somewhat unpopular because it is so frustrating and many people hope they can succeed in ignoring it. That is the real attraction of buying computers with long (e.g., 64 bit) word lengths, one gets high precision and, hopefully, freedom from worrying about round-off. Unfortunately there is a general lack of understanding of the nature of uncertainty effects in problem solving. There is confusion about the difference between the condition of a problem and that of an algorithm to solve it. If a problem is badly conditioned (the condition number is large) then nothing can be done about it while a badly conditioned algorithm might be replaced by a better one.

Condition numbers are sometimes misleading in that the estimates derived are grossly pessimistic. In my own work of solving elliptic PDEs, I see condition numbers like 10^5, 10^{10} or 10^{15} and yet observe almost no round-off effects. This leads to more confusion which is further compounded by the fact that scaling problems (simply changing the units of measurement) can have dramatic effects on round-off. This phenomena is poorly understood and difficult to analyze.

The problem of round-off error is just one aspect of a more general question: *How do you know the computed results are correct?* The mathematical results here are much less than satisfactory. Most problems addressed by scientific software are unsolvable within the framework of current mathematics. For example, given a program to compute integrals numerically, it is easy to construct a function (with as many derivatives as one wants) where the result is as inaccurate as one wants. Most theorems that apply to prove correctness of computed results have unverifiable hypotheses. And many theorems have hypotheses that are obviously violated in common applications. Most scientific software contains several heuristic code fragments. Indeed, it is usually not possible to give a precise mathematical statement of what the software is supposed to do.

The search for techniques to give better confidence in computed results is still on. A posteriori techniques still are not fully explored (it is easy to tell if x_o solves $f(x) = 0$). Computing multiple solutions efficiently is another technique that holds promise and which uses the old idea: Solve the problem 3 times (or k times) and with 3 methods and compare the results. The application of several techniques is usually required to achieve really high confidence in correctness. A rule of thumb is that it costs as much to verify the correctness of a computed result as to compute it in the first place.

Four other topics are discussed in Sections 9 to 12: 1) mapping problems and algorithms into the new parallel machines, 2) the analysis of adaptive algorithms, 3) how well mathematics models real problems, can one find theorems that are useful in assessing real computations, 4) the role mathematics plays in the experimental performance evaluation of scientific software. The final topic is particularly frustrating. Much like the weather, everyone talks about it but few do anything about it. One frequently hears statements "method x is the best way to solve problem y" which are in fact, little more than conjectures. Scientific and systematic performance evaluation is a lot of work, much of it is tedious and the work is not highly regarded by one's peers. No wonder that people prefer to do other things. We have the puzzling situation where research managers and funding agencies are always looking for "better" methods and yet they are uninterested in supporting work to measure which methods are actually good or bad.

2. Problem Solving and Algorithms

Historically, mathematics has arisen from the need to solve problems. It was recognized about a thousand years ago that one can codify the steps needed to solve some problems. These steps can be written down and someone can be told "*If you have this kind of problem, then follow these steps and you will have the solution.*" This idea matured into two of the most fundmental concepts inn mathematics: *models* and *algorithms*. Models arise from the need to make precise the phrase "*this kind of problem*" and algorithms make precise the phrase "*follow these steps*". Recall that, intuitively speaking, a model is an abstract system using axioms, assumptions and definitions which represents (well, one hopes) a real would system. An algorithm is a set of precise instructions to operate an abstract machine.

Mathematics has evolved through several well identified levels of problem solving. The lowest several of these are presented below along with typical problems and solutions.

Arithmetic

Problem	*Solution*
What is 2 + 2?	4
What is 7 × 8?	56
What is 1327/83?	15.9879518...
What is 3/2 × (1/8 − 3/5 + 1/12) / (37/8)?	47/310
What is $\sqrt{86}$?	9.273618...

Notes. There are many algorithms including memorization (table look-up in computer science terms) taught in school. It is significant that some studies suggest that about 80% of the entering college freshman cannot do long division, i.e., do not know an algorithm for computing 1327/83 as a decimal number. I would guess that a greater percentage of professional mathematicians and scientists cannot take square roots.

Algebra

Problem	*Solution*
What is $3x + 2y - x - 3y$?	$2x - y$
What is $(3x + 2y) \times (x + 3y)$?	$3x^2 + 11xy + 6y^2$
Solve $3x^2 - x - 7 = 0$	$x = 1.703...$
Solve $x^3 - 7x^2 + 3x - 110 = 0$	$x = 8.251...$

Note. Very few mathematicians know the algorithms for all of these problems.

Calculus

Problem	*Solution*
What is the derivative of e^x?	e^x
What is the integral of $\cot x$?	$\log \lvert \sin x \rvert + c$
What is the integral of $(\cos x) / x$?	$\log \lvert x \rvert + c - \sum_{i=1}^{\infty} \frac{(-1)^i x^{2i}}{2i\,(2i)!}$
What is the area under the curve $y = 1/(\sqrt{x}\,(1 + x))$ for x in $[0,\infty]$?	π
What is the series expansion of *erf* (x)	$\frac{2\pi}{\sqrt{x}} \sum_{i=0}^{\infty} \frac{(-1)^i x^{2i}}{(2i + 1)i!}$

Notes. The algorithms learned in calculus are rarely explicitly stated or taught. Problems are often called "solved" when one symbolic expression symbolic expression (say, a function or integral) is shown equal to another symbolic expression (say, an infinite series or product), neither of which can be computed exactly (even assuming one can do exact arithmetic with real numbers).

Beyond these three levels there are linear algebra, ordinary differential equations, combinatorics and others.

Even though the essense of mathematics is model construction and problem solving, a large percentage of the teaching effort is devoted to learning algorithms (often with neither the teacher or student being aware of this). I believe that it is much more important to *learn how to use knowledge* than to *learn knowledge*. In mathematical terms, it is more important to learn how to use algorithms than to memorize them.

The current situation is as follows:

(a) Enormous effort is invested in teaching people algorithms.

(b) People forget most of the algorithms they learn.

(c) Many algorithms of arithmetic, algebra, calculus, linear algebra, etc., can be implemented as scientific software and run on cheap machines.

(d) Many educators who expound the virtues of learning algorithms routinely use concepts, processes and models for which they do not know any relevent algorithms.

This situation is unstable and portends great changes in the educational system. This change will be slow but profound. Almost twenty years ago a compute program could make a grade of B on some calculus exams at MIT. Surely we cannot continue this when the cost of machines and software to do algorithms is becoming negligible.

It is fascinating to contemplate what the first two years of college mathematics would be if it were based on a mathematical system which includes the standard (and not so standard) algorithms of arithmetic, algebra, calculus, linear algebra, numerical analysis, geometry and combinatorics. The National Academy of Sciences is sponsoring a new study *Calculus for a New Century*, perhaps it will make some steps of change.

3. How Hard are Problems to Solve?

Since problem solving is one focus of mathematics, a central question is to determine just how hard various problems are to solve. Being hard to solve is measured by how much computation an algorithm must do, not by how hard it is to discover the algorithm. The problem of multiplying two integers, compute $a \times b$, illustrates the idea.