Future Directions for Research in Symbolic Computation

Report of a Workshop on Symbolic and Algebraic Computation
April 29–30, 1988
Washington, DC

Ann Boyle
B. F. Caviness
Editors

Anthony C. Hearn
Workshop Chairperson

The preparation of this report was partially supported by grant CCR-8814224 from the National Science Foundation and by the U.S. Army Research Office through the Mathematical Sciences Institute, Cornell University. This is a report to the National Science Foundation and other agencies and is not a report by or of NSF or any other agency.

Published by the
Society for Industrial and Applied Mathematics
Philadelphia
1990

Copyright ©1990 by the Society for Industrial and Applied Mathematics

For additional copies write:
Society for Industrial and Applied Mathematics
3600 University City Science Center
Philadelphia, PA 19104-2688

Reprinted as a University of Delaware technical report with permission from the Society for Industrial and Applied Mathematics. Copyright 1990 by the Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania. All rights reserved.

For additional copies of the technical report write:
Department of Computer and Information Sciences
103 Smith Hall
University of Delaware
Newark, DE 19716
USA

Participants at the Workshop on Symbolic and Algebraic Computation

Anthony C. Hearn, RAND Corporation, Workshop Chair
B. F. Caviness, Coordinator, National Science Foundation and University of Delaware[1]

Subpanel on Applications

André Deprit, National Bureau of Standards,[2] Chair
John Fitch, Department of Computer Science, University of Bath, United Kingdom
Gerald Guralnik, Department of Physics, Brown University
Michael Levine, Pittsburgh Supercomputer Center and Physics Department, Carnegie-Mellon University
James McIver, Department of Chemistry, SUNY Buffalo
Anil Nerode, Mathematical Sciences Institute, Cornell University

Subpanel on Software and Systems Design

Richard Jenks, IBM Watson Research Center, Chair
Richard Fateman, Division of Computer Science, University of California–Berkeley
Gaston Gonnet, Computer Science Department, University of Waterloo
Alan Perlis, Department of Computer Science, Yale University

Subpanel on Algorithms and Theory

Moss Sweedler, Department of Mathematics, Cornell University, Chair
Shreeram Abhyankar, Departments of Mathematics and Industrial Engineering, Purdue University
Bruno Buchberger, Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria
David Chudnovsky, Department of Mathematics, Columbia University
William F. Schelter, Department of Mathematics, University of Texas
Peter Weinberger, AT&T Bell Laboratories

[1] Given affiliations were the ones in effect at the time of the workshop.

[2] The name of the NBS has been changed to the National Institute of Standards and Technology.

Subpanel on Symbolic/Numeric Interfaces

Morven Gentleman, Division of Electrical Engineering, National Research Council of Canada, Chair

Richard Askey, Department of Mathematics, University of Wisconsin

John Rice, Department of Computer Science, Purdue University

Phil Smith, IMSL, Houston

Paul Wang, Department of Mathematical Sciences, Kent State University

Preface

At the meeting of the Advisory Committee for the Division of Computer and Computation Research of the National Science Foundation on December 3–4, 1987, "It was noted that the symbolic and algebraic computation area currently receives few proposals and that the potential of the area is largely unrealized. A proposal to appoint a panel to develop an initiative in this area was endorsed."[1]

The Mathematical Sciences Institute at Cornell University indicated an interest in organizing a workshop. Since it had considerable experience and expertise in organizing workshops and much interest in the subject matter, the National Science Foundation encouraged the submittal of a proposal that led to its organizing the event. This report is based on the issues raised, the solutions suggested, and the actions recommended at the workshop that was held in Washington, DC, on April 29–30, 1988.

The purpose of the workshop was to provide NSF and other research funding agencies with a deeper understanding of the current status of research in this area, the potential of future research, and the expected payoff of increased funding for symbolic and algebraic computation. In attendance were a coordinator and 22 participants from a wide variety of backgrounds in symbolic computation research, in the underlying mathematical theory, and in various application fields. In addition, several observers from federal agencies were present (see Appendix A). The workshop began with a general, roundtable discussion of the main issues. Then the participants divided into four subpanels—applications, systems and software, algorithms and theory, and numeric/symbolic interfaces—to discuss issues in these areas in more detail.

The report has been assembled and written by the editors from material provided by the workshop participants and others, especially reports provided by the subpanel chairs: André Deprit, Morven Gentleman, Richard Jenks, and Moss Sweedler. The editors acknowledge and express our appreciation to the following additional colleagues who helped in various ways

[1] From the minutes of the meeting.

with the preparation of the report. Their contributions were manifold, including written contributions on some specialized topics, various items of information, and constructive criticism.

Kamal Abdali	Dennis Arnon
Guy Cherry	James Davenport
David Elliott	Hans van Hulzen
Moayyed Hussain	Erich Kaltofen
Arthur Norman	Richard Pavelle
Richard Petti	David Saunders
Charles Sims	Michael Singer
Stephen Wolfram	Richard Zippel

We thank Wilson Kone for his able administration of the workshop, Susan Allmendinger for her valuable editorial assistance in preparing the report for publication, and the referees for their suggestions, especially the ones that lead to a better organization of Chapter 5.

Drafts of the report were reviewed by the participants and others. Many of the resulting suggestions have been incorporated into the final report, but any remaining errors, omissions, or other failings are the sole responsibility of the editors.

Contents

Executive Summary[2]

Computing is dramatically affecting the way that modern science and engineering are carried out. To date, symbolic computation has played a limited, but important, role in the twentieth century advancement of science and engineering. This report documents that role, explores the potential of this mode of computation, and recommends ways to harness its large and mostly untapped potential.

Symbolic computation is the science and technology that aims to automate a wide range of the computation involved in mathematical problem solving. It emphasizes discrete computation on symbols representing mathematical objects. The symbols represent not only numbers like the integers and rationals, but also other mathematical objects like polynomials, rational and trigonometric functions, algebraic numbers, groups, ideals, and tensors. Typically, the computations are exact, in contrast to most numeric calculations where computations use approximate floating point arithmetic. Often though, exact and approximate calculations are used together in important ways, for example, in computing exactly the first n terms of a series that is an approximation to the solution of a differential equation. The truncated series can then be evaluated at a particular point using floating point arithmetic to approximate the numerical solution of the differential equation.

Notable results have been achieved in symbolic computation over the last two decades. Algorithms have been discovered for integration in finite terms and for computing closed form solutions of differential equations. Fast algorithms have been devised for factoring polynomials and computing greatest common divisors. Powerful interactive systems for doing symbolic computation have been designed and built. The software has improved the productivity of scientists and engineers; it has made possible the solution of problems that were previously intractable. However, only the surface has been scratched. We are still confronted with a wide spectrum of challenging

[2]The format of the Executive Summary is borrowed from the Executive Summary of *Future Directions in Control Theory—A Mathematical Perspective*, Report of the Panel on Future Directions in Control Theory, Wendell H. Fleming (Chair), Society for Industrial and Applied Mathematics, Philadelphia, 1989.

problems whose solution will have a crucial influence on our technological problem solving ability.

Uses for symbolic computation range over the entire scope of mathematics and its applications, that is, essentially all science and engineering. There are three modes of use: (1) computations that could be carried out by hand, but can be done more productively and accurately by a symbolic computation system, (2) computations that are beyond hand calculation but can be done more or less routinely by machine, and (3) calculations that require substantial effort to complete even when using a computer. In addition, there are important applications that are out of reach of present computational methods and hardware. This report contains short descriptions of sample applications in a variety of fields.

Findings

Symbolic computation is a field of accomplishment, a field of promise, and a field of contrasts. It faces educational, technological, research, and communications challenges that arise from its diversity, richness, wide applicability, and immaturity. The field is at a turning point of possibilities brought forth by improvements in computer hardware, new algorithms, and new software. A central challenge is to increase the number of applications and users, which will bring symbolic computation more into the main stream of science and engineering.

The prerequisites for more users are:

- Better software platforms that include the key ingredients of improved user interfaces and well integrated symbolic and numeric capabilities
- Better documentation and other educational materials for users
- More effective methods and algorithms for solving important scientific and engineering problems
- The increased availability of cheap, high-performance, large-memory computers that are capable of serving as adequate hardware platforms for symbolic computation systems

Other key findings are:

- Symbolic computation is a part of a key technology, namely, scientific and engineering computation, that is becoming increasingly important to science, technology, and society
- Symbolic computation enhances scientific and engineering productivity

- Symbolic computation has made important progress in developing software and in discovering new algorithms since the mid-1960s
- The opportunities for substantial progress in symbolic computation are significant

There are many aspects to this multi-faceted field.

- Mathematics and computer science are the primary disciplines that contribute to basic research in this area. Applications occur throughout science and engineering. The interplay between fundamental results and technology is an important dynamic of the field.
- A broad range of mathematics is relevant to symbolic computation research and vice-versa. To date algebra, algebraic geometry, logic, group theory,number theory, combinatorics, complex variables and analysis, among others, have played important roles in fundamental research on symbolic computation. Conversely, symbolic computation is a potential research tool for all areas of mathematics.
- In computer science complex data structures, object-oriented programming, and other advanced programming tools are important. Design of user interfaces is particularly crucial. Both heuristic and algorithmic methods are necessary for successful applications. There are important interactions with scientific text processing, graphics, and numerical computing.

Symbolic computation software reflects many of the successes and problems of the field.

- Symbolic computation software is typically large, sophisticated, and error prone. All the problems associated with the design and engineering of large software systems are present here.
- Implementation of new results is lagging; many new algorithms that have been discovered over the past decade have been implemented in few, if any, systems.
- Software development is lagging behind new hardware technology, especially in the use of new display hardware and in the use of new architectures, including supercomputers.
- More modular, reusable, high-quality, library-style software needs to be developed. The lack of such publicly available software inhibits researchers from building on the work of others and impedes the development of special systems for applications.

The theory and algorithms base has been substantially improved in recent years, but there is much work still to be done.

- The algorithm base contains gaping holes in fundamental areas such as symbolic linear algebra, symbolic approximations, and complex variables.
- Little research has been done on parallel symbolic algorithms.
- More research is needed on large scale and special purpose algorithms.

Educational matters are particularly crucial at this time.

- Nonspecialists have little knowledge about the capabilities and limitations of symbolic computation.
- The pool of human resources for research in symbolic computation is small.
- Educating new researchers and attracting new users are keys to the future development of the field of symbolic computation. In the United States, education in this area is almost nonexistent. Currently, established curricula do not exist for graduate students who wish to work in this area. At most American institutions with graduate mathematics and computer science programs, graduate students have no opportunity to study this area, no faculty are doing research in the area, and few courses are directly relevant.

Recommendations

From 1965 to 1980 a small but dynamic group of researchers in the United States provided the early foundations for the field of symbolic computation. Out of this work came important advances in algorithms and software. Most of the important worldwide software developments were done in North America. Important examples include ALTRAN, MACSYMA, MU-MATH, REDUCE, SAC-I & II, SCRATCHPAD, and SMP. Since 1980 fundamental research and software development have leveled off.[1] DERIVE,[2] MAPLE,[3] and

[1] Important contributions have been made since 1980 in both algorithms and systems, but the research activity in the United States has leveled off or even declined.

[2] DERIVE *User Manual*, 3rd ed., Soft Warehouse, Inc., Honolulu (1989).

[3] B. W. Char et al., MAPLE *User's Guide*, 4th ed., WATCOM Publications Ltd., Waterloo, Ontario (1985).

MATHEMATICA[4] are three of the few, if not the only, important new symbolic computation systems to appear since 1980. Now other countries are beginning to exploit the advances that have been made. The United States can continue to lead in this area, can leverage its previous investments, and reap the considerable potential benefits of this field by reinvigorating its research efforts. The following recommendations are designed to accomplish these objectives.

The most important recommendations are concerned with increasing the number of applications and users of symbolic computation. Other recommendations will help to build a critical mass of researchers in this area. Improvements to the software platform are also needed to accelerate and to speed applications. These recommendations should be given priority.

Many of the recommendations are couched in terms of funding initiatives, but much can be accomplished by the academic and industrial sectors independent of any new funding. Universities can immediately begin to teach more in this area and in other ways begin to focus on improving the flow of information about symbolic computation. Industry can continue its fledgling steps to take software needs in this area seriously. However, to make substantial progress will require serious government action. Government funding for upstream research developments, university efforts in research and in increasing the human resource pool, and both coupled with commercial efficiencies and focus on downstream implementation should be an effective outline for progress.

New initiatives are needed by the funding agencies to increase significantly the levels of research on and applications of symbolic computation. Reacting to the normal flow of proposals is likely to be insufficient since a critical mass has not been obtained in this field. The following actions are recommended.

Priority Recommendations

To the Funding Agencies

- Substantially improve the software platform for research and applications by:
  - Funding research on high-quality, reusable user interfaces
  - Funding software acquisition, development, and maintenance needed to capitalize the instrumentation for symbolic computation research and applications

[4] S. Wolfram, MATHEMATICA—*A System for Doing Mathematics by Computer*, Addison-Wesley Publishing Co., Redwood City, California (1988).

  - Funding establishment of high-quality libraries of symbolic algorithms and methods
  - Funding research on interface protocols between various software packages and systems
  - Encouraging joint university and industry research
  - Supporting summer sessions and special years to support tool building and experimental aspects of the field

- Stimulate developments at the interface of symbolic and numeric computation by:

  - Funding research in defining the interface and on algorithms that employ both symbolic and numeric methods
  - Funding course development that incorporates symbolic and numeric computing
  - Funding workshops to attack a particular problem using symbolic and numeric methods

- Improve the basic mathematics and algorithms underlying symbolic computation by:

  - Accelerating research on symbolic methods that are especially relevant to applications such as approximation methods and methods for solving ordinary and partial differential equations
  - Funding further research on algorithms for fundamental computations in areas like symbolic linear algebra, nonlinear algebra, and complex variables
  - Funding research on symbolic algorithms to take advantage of new computer architectures, including supercomputers
  - Encouraging more research on ways to deal with complexity problems in symbolic computations such as the use of probabilistic algorithms

- Address education for users and new researchers by:

  - Funding research on incorporating symbolic computation in mathematics, science, and engineering curricula
  - Supporting the development of educational materials on symbolic computation

To the Universities

- Provide adequate computing facilities for symbolic computation, including appropriate software.
- Include more material relevant to symbolic computation in university curricula. This can be done in a variety of ways:
  - Introduce symbolic computation as a tool into existing courses, especially ones covering aspects of applied mathematics.
  - Introduce new courses that contain material on the various aspects of symbolic computation, including applications.
  - Put more emphasis on constructive techniques in current mathematics courses, especially courses in algebra, algebraic geometry, and number theory.
  - Use more examples from symbolic computation in computer science courses dealing with software engineering, graphics, and algorithms. An increased emphasis on ideas from category theory and universal algebra in software design would be particularly applicable to symbolic computation software.
  - Develop special education and research programs in which symbolic computation plays a key role. A curriculum that combines mathematics and computation provides a good fundamental education for future scientists and engineers. One such curriculum is given in Appendix B.

To the Professional Community

- Produce more and better educational materials. A lack of textbooks is an especially pressing problem. These needs have been cited on several occasions and improvements are slowly occurring, but continued efforts are still needed.

Additional Recommendations

To the Professional Community

- Increase efforts to improve and increase interactions among the various components of the scientific computation community. More interactions are desirable between mathematicians and computer scientists interested in symbolic computation, between researchers in numerical and symbolic computing, and between software builders and users.

- Resolve the conflict over the commercial requirements to protect products and the academic requirements to exchange information freely. A solution to this problem may be found through the use of proprietary software kernels with publicly available source code built on top. For this model to serve science successfully, the mathematical algorithms must be in the public domain. Several producers of symbolic software have already adopted this model.

To the Commercial Sector

- Continue the initial steps to take symbolic computation software seriously.

To the Funding Agencies

Address the interdisciplinary nature of the field by:

- Establishing mini-centers for symbolic and algebraic computation
  - that span science and engineering with a users point of view
  - that cut-across computer science and mathematics from a system development point of view
- Funding workshops and special years devoted to symbolic computation and its applications
- Supporting summer session programs with a regional or national scope on the use and development of symbolic computation systems

Chapter 1

Introduction

Symbolic computation, while implemented on computers only since 1953, has a long history as an important tool in scientific development.

> Exploitation of algebra and computation were prominent aspects of the revolutionary science of the 16th century. Architects of that new science were Paracelsus (d. 1541) in chemistry, Nicolause Copernicus (d. 1543) in astronomy, Andreas Vesalius (d. 1564) in anatomical medicine, and Girolamo Cardano (d. 1576) in algebra.[1]

Several generations later in Europe's "century of genius," from 1620 to 1720, we find that:

> Europe's turbulent "century of genius" embraced the visual arts, literature, and thought, as well as mathematics. For some scholars this coexistence of high creativity in culture and high disorder in society is paradoxical; to others it indicates the tenacity of Europe's republic of letters, expanded patronage from wealthy bourgeoisie, and the robust vitality of the age. ... Mathematics too had enjoyed spectacular development since the mid-sixteenth century. Algebra and arithmetical computations particularly had flourished.[2]
>
> Leibniz's studies leading to the calculus began under the guidance of Huygens in Paris between 1673 and 1676. His work rests on at least three basic components. Leibniz sought a *characteristica generalis*, a symbolic language, for translating mathematical methods and statements into algorithms and formulas.[3]

Very early in the development of the technology for automatic computation it was realized that computers could do symbolic as well as numeric

[1] J. E. Brown, The Medieval-Renaissance-Reformation Periods in Europe—Introduction, in R. Calinger, *Classics of Mathematics*, Moore Publishing Co., Oak Park, Illinois (1982), p. 219.

[2] R. Calinger, op. cit., pp. 267–273.

[3] Ibid., p. 279.

computation. In 1842 L. F. Menabrea summarized the remarkable ideas of Charles Babbage for an "Analytical Engine." Menabrea's paper was translated into English and extensively annotated by the Countess of Lovelace (daughter of Lord Byron) and published in *Taylor's Scientific Memoirs*.[4]

Unlike many of her contemporaries, Byron thoroughly understood and appreciated Babbage's machine. She was also a scientific visionary in her own right, as can be seen when she wrote:

> Many persons who are not conversant with mathematical studies imagine that because the business of the engine [Babbage's Analytical Engine] is to give its results in *numerical notation*, the *nature of its processes* must consequently be *arithmetical* and *numerical* rather than *algebraical* and *analytical*. This is an error. The engine can arrange and combine its numerical quantities exactly as if they were *letters* or any other *general* symbols; and in fact it might bring out its results in algebraical *notation* were provisions made accordingly.[5]

In 1953, after the invention of the electronic computer, the Countess' vision was first realized in two master's theses, one by J. F. Nolan of the Massachusetts Institute of Technology[6] and another by H. G. Kahrimanian at Temple University.[7]

By the mid-1960s research on symbolic computation had begun in earnest at several places, primarily in the United States, as is documented by the proceedings of an early conference.[8] However, after an initial flurry of activity, research on symbolic and algebraic computation has faltered in the United States. The considerable potential of the powerful new techniques has hardly been realized.

This report investigates why this is the case and its focus is therefore:

- to describe the current state of symbolic computation
- to indicate its impact on science and engineering
- to assess the potential future impact

[4] L. F. Menabrea, Sketch of the Analytical Engine Invented by Charles Babbage. With Notes upon the Memoir by the Translator, Ada Augusta, Countess of Lovelace [From the *Bibliothèque Universelle de Genève*, Oct 1842, No. 82.], Source: P. Morrison and E. Morrison (eds.), *Charles Babbage and His Calculating Engines—Selected Writings by Charles Babbage and Others*, Dover Publications, Inc., New York, 1961.

[5] Ibid.

[6] J. F. Nolan, *Analytical Differentiation on a Digital Computer*, SM Thesis, Massachusetts Institute of Technology, May 1953.

[7] H. G. Kahrimanian, *Analytical Differentiation by a Digital Computer*, MA Thesis, Temple University, May 1953.

[8] R. W. Floyd (ed.), Proceedings of the ACM symposium on symbolic and algebraic manipulation, *Comm. ACM* **9** (1966), pp. 547–643.

- to determine what is needed to advance the field significantly

It has been generally recognized that mathematical computing, of which symbolic computing is a part, is a basic mode for research and development in science and engineering. Several reports have focused on the importance of mathematical computing to the well-being of the nation. To date these reports have concentrated on the important areas of numeric and high-performance computing. The emergence of symbolic computation is making possible the fruitful and natural use of a full-range of both symbolic and numeric computation in mathematical, scientific, and engineering computation.

Recent national reports

1. *Report of the Panel on Large Scale Computing in Science and Engineering.* Peter Lax, Chairman, Sponsored by the U.S. Department of Defense and the National Science Foundation, in cooperation with the Department of Energy and the National Aeronautics and Space Administration, Washington, DC, December 26, 1982.

2. *A National Computing Environment for Academic Research.* Marcel Bardon and Kent Curtis, NSF Working Group on Computers for Research. National Science Foundation, July 1983.

3. *Renewing U.S. Mathematics—Critical Resource for the Future*, Report of the Ad Hoc Committee on Resources for the Mathematical Sciences, The Commission on Physical Sciences, Mathematics, and Resources, National Research Council, National Academy Press, Washington, DC, 1984.

4. *A Report of the Panel on Future Directions in Computational Mathematics, Algorithms, and Scientific Software.* Werner C. Rheinboldt, Chairman. SIAM, Philadelphia, 1985.

5. *A National Computing Initiative—The Agenda for Leadership.* Report of the Panel on Research Issues in Large-Scale Computational Science and Engineering. Harold J. Raveché, Duncan H. Lawrie, and Alvin M. Despain. SIAM, Philadelphia, 1987.

6. *Research and Development Strategy for High Performance Computing.* Executive Office of the President, Office of Science and Technology Policy, November 20, 1987.

Chapter 2

Current Status

Since World War II, science and technology have made spectacular advances in field after field, but among the most spectacular have been the advances in computing. Computing is a part of a wave of fundamental scientific and technological advances that are unique to our history. Symbolic computation is a part of these spectacular advances. Some of the assessments of symbolic computation that have appeared in the popular scientific press underline the fascination with this computational mode. Pavelle, Rothstein, and Fitch in *Scientific American* note that "Of all the tasks to which the computer can be applied none is more daunting than the manipulation of complex mathematical expressions."[1] *Science News*, in an article on computer calculus, began

> "Awesome . . . invaluable . . . unbelievable . . . ." These are the assessments by normally taciturn research scientists of symbolic computer algebra, a group of programs that allows computers to carry out theoretical (rather than merely numerical) calculations. These programs do in a few brief minutes virtually all mathematics that most engineers and scientists know; their ability to slog through theoretical solutions to large systems of equations has already led to advances in [the study of] gravitation and high energy physics.[2]

An article in *Nature*[3] begins, "The notion that computers are mere number crunchers is fast being exorcised by one small group of people—the computer algebraists. In the past twenty years, several classes of problems which require analytical and not numerical solutions have been tackled successfully."

[1] R. Pavelle, M. Rothstein, and J. Fitch, Computer algebra, *Scientific American* **245** (1981), pp. 136–152.

[2] L. A. Steen, Computer calculus, *Science News* **119** (1981), pp. 250–251.

[3] Algebra made mechanical, *Nature* **290** (1981), pp. 198–200.

An article in the *New York Times*[4] about MATHEMATICA says, "The importance of the program cannot be overlooked or overestimated, however, because it so fundamentally alters the mechanics of mathematics. It calculates and graphically displays, in either two or three dimensions, virtually any formula or expression you can concoct."

Hyperbole aside, what is this computational paradigm that evoked such statements? Symbolic computation is the science and technology that aims to automate a wide range of the processes involved in mathematical problem solving. More particularly, symbolic computation is mathematical computation in which the emphasis is on discrete computation with symbols representing mathematical objects. The symbols include the usual representation systems for numbers like the integers and rationals. But it also includes polynomials, rational and trigonometric functions, algebraic numbers, groups, ideals, and tensors, among others. In fact, it is possible, in theory, to carry out any heuristic or algorithmic computational method. Typically, the computations are carried out exactly. In contrast, most numeric calculations are carried out using approximate floating point arithmetic. Often though, exact and approximate calculations are used together in important ways, such as computing exactly the first n terms of a series that is an approximation to the solution of a differential equation. The truncated series can then be evaluated at a particular point using floating point arithmetic to get an approximation to the numerical solution of the differential equation.

A notable feature and difficulty of symbolic computation research, development, and applications is the diversity of knowledge and skills required. With respect to development of software systems, all the well-known engineering problems that occur in building large-scale computer software systems are compounded by the complexity and scope of the underlying mathematics. Good symbolic computation systems can be developed only by teams with considerable software engineering skills and broad mathematical knowledge. In algorithm research, a diversity of computer science and mathematical issues arises. For example, techniques from classical analysis are used in studying the computational complexity question of bounding the size of expressions that arise in computation. Algebra and algebraic geometry are important tools in algorithm research, even in areas like integration in finite terms and differential equations. Applications of symbolic computation range over all of mathematics,[5] science, and engineering.

Numeric computation automates the last step of mathematical problem solving. It has had substantial effect on the efficiency and modes of scientific

[4] Liberating the 'prose' of math from its grammar, *New York Times*, July 19, 1988, p. 21.

[5] Both pure and applied.

and engineering inquiry. Symbolic computation aims at the automation of the steps of mathematical problem solving that precede evaluating numerical models and that, to a large extent, are still the domain of human problem solvers. Used together, symbolic and numeric computation enhance each other.

Many terms have been used for symbolic computation in the literature, including *computer algebra, symbolic and algebraic computation, symbolic mathematical computation, seminumerical computation, symbolic and algebraic*[6] *manipulation*, and *formula manipulation*. The terms *computational group theory* and *computational number theory* denote important subfields of symbolic computation.

Symbolic and algebraic computation shares intellectual interests with several other areas, including numeric computation, automatic theorem proving and checking, automatic program verification, computational geometry, and programming language design.

Constructive methods in mathematics have a long tradition, going back to Euclid's algorithm and earlier. Early examples of symbolic computation can be found in Isaac Newton's *Universal Arithmetic* (1728), in which rules for manipulating universal mathematical expressions, that is, formulas containing symbolic indeterminates, and algorithms for solving equations built with these expressions are systematically discussed. Before the mid-nineteenth century the work involved in mathematical problem solving was much different from that of today. In particular, computation played a much larger role, but by the end of the nineteenth century mathematics had changed substantially; it had become more abstract and more qualitative. Mathematicians increasingly concerned themselves with questions about the structure of mathematical systems. Many mathematical problems were too difficult to be solved with the computational tools that were available.

With the existence of computers and with advances in mathematics that have led to better algorithms, the solution of problems via computation has once again become a profitable way to do science and engineering. The first uses of computers to do symbolic computation occurred in the 1950s, but the full use of symbolic computation still awaits the exploitation of recently available hardware with fast processors and, more importantly for symbolic computation, with large and cheap memories.

[6]The term *algebra* is used both in its meaning as a branch of mathematics that includes many interactions with symbolic computation and in its more generic sense as a mathematical system with a set of symbols satisfying given rules and relations. It is in this latter sense that algebraic computation is often used as a synonym for symbolic computation. But the reader should not conclude that symbolic computation is only for computation in algebra (in the sense of a subfield of mathematics). Symbolic computation is useful in all areas of mathematics and has applications to many other fields.

With the advent of time-sharing and virtual memory operating systems in the mid- to late 1960s, research on symbolic computation flourished at a few research sites. This research on new algorithms, on the design and engineering of software, and on the use of symbolic computation software gradually lead to the development of an international research community sponsoring regular symposia and workshops. The research roots were in computer science and mathematics, and they drew on some of the deepest aspects of both disciplines as well as those of the application areas like theoretical physics. Thus, symbolic computation lies at a boundary among several fields, a position that gives it much depth and richness and, from a research perspective, makes it a difficult and challenging area. From an administrative and support perspective, the interdisciplinary nature presents difficult problems.

Notable results have been achieved in symbolic computation over the last two decades. Algorithms have been discovered for integration in finite terms and for computing closed form solutions of differential equations; fast algorithms have been devised for polynomial factorization and greatest common divisor computations, and powerful interactive systems have been designed and built for carrying out symbolic computation. The software has improved the productivity of scientists and engineers; it has made it possible to solve previously unsolvable problems. However, only the surface has been scratched. We are still confronted with a wide spectrum of challenging problems whose solution will have a crucial influence on our technological problem-solving potential.

Mathematics is a basis of technological progress, and technological progress is a key for international competitiveness. Automating an important part of the mathematical problem-solving process is a key technology for a nation that wishes to control, structure, and accelerate technological progress. The automation of the solution of mathematical problems is a powerful lever by which human productivity and expertise can be amplified many times.

Symbolic computation research is accelerating in other countries at the same time that it is leveling off in the United States. Of the 1130 sites and individuals who have obtained at least one copy of REDUCE from licensed distributors during the past three years, 40% were in Japan, 30% in Europe, 27% in the United States and Canada, and 3% in the rest of the world. In the twelve months from June 1987 to June 1988, Symbolics, Inc., sold over 600 new licenses and software update contracts. About 30% of those sold were to users outside the United States. Also, in the last three years the Austrian government helped set up an institute at the Johannes Kepler University in Linz that is entirely devoted to symbolic computation. There

is no comparable facility in the United States.

Applications of symbolic systems do seem to be increasing in this country. At the General Electric Corporate Research and Development Center in Schenectady, New York, symbolic computation is used routinely for such tasks as designing coils for their nuclear magnetic resonance imaging systems, for designing antenna arrays, and for inverse problems in tomography. At the GE Microelectronics Center in the Research Triangle Park, North Carolina, symbolic computation systems are used to do integrations that arise in circuit simulations. At the General Motors Research Laboratory in Warren, Michigan, and at the GM Systems Engineering Center in Troy, Michigan, symbolic computation is being used in several general-purpose programs to derive mathematical models, to do Jacobian sensitivity analysis, and to do analyses for control applications. In addition, symbolic computation systems are used in calculator mode to increase productivity and to ensure correctness of mathematical computations. At the Sandia National Laboratory in Livermore, California, symbolic computation is used for combustion stability analysis and for nonlinear spectroscopy calculations.

Assessment of symbolic computation is bedeviled by the half-full, half-empty phenomenon: when to be optimistic about how far symbolic computation has progressed and when to be pessimistic about how much needs to be done for solving important problems. But given the current low levels of federal support for research in this area, it is hardly surprising that more has not been accomplished.

2.1 Software and Systems

One of the biggest successes in symbolic computation research has been the development of software systems of substantial sophistication and scope. It was the software systems that evoked the previously quoted enthusiastic admiration of the scientific press.

The term *symbolic computation software* refers to a full range of software from the implementation of a single algorithm to the implementation of large integrated systems, from special-purpose application systems to large general-purpose systems. Such software often includes graphics, elaborate user interfaces, and numeric software as well as algorithms for symbolic computation.

The main intellectual focus for symbolic computation software is the design issue. Some issues are specific to scientific computation, such as how does one design software that

- advances the ability of scientists and engineers to solve complex mathematical problems efficiently and effectively,

- takes advantage of the natural abstraction and structure in the domain of mathematics and thus has clean semantics that avoids the complicated use of flags and other *ad hoc* devices,
- integrates with symbolic computation other modes of scientific and mathematical computation such as graphics and numerical computing,
- is well designed for a range of equipment from workstations to supercomputers while taking advantage of new trends in architectures such as parallelism,
- is general in the sense of providing a wide range of facilities and mathematical algorithms but is still effective for specific applications?

Other issues are common to all large software systems. For example, how does one design symbolic computation software that

- is reliable and maintainable,
- is appropriately standardized,
- has a good user interface appropriate for the target group of users,
- is reusable in both general- and special-purpose systems,
- takes advantage of new trends in software methodologies, for example, object-oriented and logic programming?

In addition to the intellectual issues, there are issues of practical production: symbolic computation software is often large and exceedingly complex, requiring many person-years of effort to implement, document, maintain, and upgrade. A real challenge for symbolic computation is to find ways to modularize the software development process so that it can be done in smaller parts. Then the parts can be shared and used in many contexts and will be easier to maintain and upgrade.

The use of symbolic systems has grown substantially in recent years. This growth can largely be attributed to improvements in hardware, in which numbers of computational cycles and bytes of memory grow by a factor of two per year. At the same time, advances in hardware for color graphics, desktop publishing, and storage media such as compact disks will likely have a profound effect not only on the cost but also on the effectiveness and expectations of symbolic computation systems over the coming decade. Corresponding improvements in algorithms have made previously intractable algebraic computations a reality. Software for indefinite integration of elementary functions, for example, has become a tool that rivals or exceeds the

usefulness of tables of integrals. Improved understanding and development of Gröbner basis techniques, integer factorization, and techniques for analytic approximation have opened up new application areas of symbolic computation never before realizable. Nevertheless, numerous problems continue to stunt the potential growth and use of symbolic computation.

Reusability and interaction with other software. Symbolic computation systems generally do not interact with other software systems. Most are stand-alone systems with their own interactive language used to access their facilities. Systems often do not provide adequate access to the commonly used numeric libraries such as IMSL and NAG, but there is some recent, interesting progress in this area. Furthermore, most systems cannot be used as a symbolic engine by other software. Symbolic computation systems need further development of interface software that enables them to interact with other software, including editors, numeric languages and libraries, text processors, graphics, and artificial intelligence programs.

There have been several efforts along these lines. GENTRAN,[7] implemented in both REDUCE and MACSYMA, facilitates the generation of FORTRAN programs from symbolic results generated by a symbolic computation system. The notebook concept in MATHEMATICA ties text processing closely to computation. J. J. Uhl has noted some of the possibilities of the latter combination.

> With the introduction of MATHEMATICA notebooks as live electronic texts, I believe MATHEMATICA will revolutionize undergraduate mathematics. When this happens, undergraduate mathematics in America will reach Lynn Arthur Steen's goal of becoming more like real mathematics both in the industrial workplace and in academic research.[8]

In addition, symbolic computation systems should be able to interact with one another. For example, it would be convenient to do group theoretic computations in CAYLEY and polynomial computation in another system. A design of a computational environment that would allow communication between several independent software programs is needed to enhance the use of computer algebra. Robust, high-quality "libraries" of symbolic computation algorithms that can be used in many applications are needed. Some work on generic interface issues has been done.[9]

---

[7] B. L. Gates, A numerical code generation system for REDUCE, in *Proc. 1986 Symp. on Symbolic and Algebraic Computation.*, ACM (1986), pp. 94–99.

[8] J. J. Uhl, MATHEMATICA and me, *Notices AMS* **35** (1988), pp. 1345–1347.

[9] J. Purtilo, Applications of a software interconnection system in mathematical problem solving environments, in *Proc. 1986 Symp. on Symbolic and Algebraic Computation*, ACM (1986), pp. 16–23; D. Arnon, R. Beach, K. McIsaac, and C. Waldspurger, CaminoReal:

Improved coverage of mathematics. Most systems provide a good deal of symbolic computation through college-level calculus. Beyond that the knowledge and algorithms are spotty. In particular, facilities for applied mathematics are relatively minimal. An important need is for additional symbolic approximation methods that apply to a wide range of applied problems. Using a symbolic computation system, a student should be able to formulate many computational problems found in a college-level applied mathematics textbook.

A related problem is the implementation of state-of-the-art algorithms. In recent years important new algorithms have been discovered that are not routinely available in the current software. This is basically a person-power problem—there are just not enough persons to do the time-consuming work to implement these complex methods.

State-of-the-art displays. Well-displayed input and output are other critical issues for the effective use of symbolic computation systems. It is important that the input and output of symbolic systems appear on the screen in textbook quality. More effective ways of handling large expressions that are often generated in the course of symbolic computation are needed. The systems should provide visual means for extracting or replacing parts of expressions, moving terms around, and effecting transpositions, cancellations, and other manipulations. Further research and development like that on IRIS[10] and MATHSCRIBE[11] should be encouraged. In addition, it is important for data to be presented effectively in numeric, symbolic, or graphic form. The user should be able to transform the expression in a way to achieve some specially appropriate format. The appropriate graphic representation may be a curve, a surface, a contour plot, a movie, or any combination of these.

Understandability. Symbolic computation systems must deal with many subtle issues in mathematical notation. Sometimes mathematical notation is ambiguous; other times the same or similar notation is used for different concepts in various branches of mathematics. A symbolic computation system must deal with the ambiguity and different meanings. Currently, this is most often done in an *ad hoc* fashion that makes the systems cumbersome. The study of ways to provide symbolic computation systems

---

an interactive mathematical notebook, in *Document Manipulation and Typography* (Proc. Intl. Conf. on Electronic Publishing, Document Manipulation, and Typography [EP88], Nice, France, April 20–22, 1988), Cambridge University Press, New York (1988), pp. 1–18.

[10]B. L. Leong, Iris: design of a user interface program for symbolic algebra, in *Proc. 1986 Symp. on Symbolic and Algebraic Computation*, ACM (1986), pp. 1–6.

[11]C. J. Smith and N. M. Soiffer, MathScribe: a user interface for computer algebra systems, in *Proc. 1986 Symp. on Symbolic and Algebraic Computation*, ACM (1986), pp. 7–12.

with clean semantics that make them more understandable and easier to use is a research topic that deserves much more attention. Important progress has been made on some of these problems through powerful abstractions that have been built into some recent systems.[12] The best known of these is SCRATCHPAD.[13]

Simplification of expressions is a central issue in symbolic computation. It too is handled mostly in an *ad hoc* fashion by present software, and that adds to the difficulty of using and understanding current systems. Symbolic forms must be convertible to "understandable" alternative forms that may involve factorization, common subexpression elimination, or collection of coefficients, for example. Some of the simplification should be done automatically by the system, and some should be under user control. More research is needed on these difficult problems.

Extensibility. Systems should be easily extendible by the scientist or engineer so that new facilities can be added to the existing system. Here it is desirable that the user be able to build easily on existing facilities. It is also important that the systems allow users to alter existing facilities and to do this without a performance penalty.

Large computations and the use of supercomputers. Unlike numeric computation, it is difficult to estimate the time and space requirements for symbolic computation. The success of a symbolic computation depends critically on the size of intermediate expressions generated during the computation. Carefully designed monitoring aids for software and hardware would be useful in measuring space, time, and progress toward a solution. If a satisfactory answer has not been obtained, a user may wish to know whether

- there are fundamental inadequacies in the approach; for example, no algorithm is known or known algorithms have not been programmed,
- a heuristically determined limit has been exceeded in some kind of search, or
- some resource limit (for example, stack overflow, memory) has been encountered and why.

Supercomputers have not been used extensively for symbolic computation. But there are applications that could benefit from a combination of

[12] J. Foderaro, *The Design of a Language for Algebraic Computation Systems*, Ph.D. Thesis, University of California–Berkeley, 1983; S. K. Abdali, G. W. Cherry, and N. Soiffer, An object oriented approach to algebra system design, in *Proc. 1986 Symp. on Symbolic and Algebraic Computation.*, ACM (1986), pp. 24–30.

[13] R. D. Jenks, A primer: 11 keys to new SCRATCHPAD, *Proc. EUROSAM '84, Lect. Notes Comp. Sci.* **174** (1984), pp. 123–147.

supercomputers and symbolic computation, as has been demonstrated by a recent work on a Cray at the Konrad-Zuse-Zentrum für Informationstechnik in West Berlin.[14]

Reliability and maintainability. Another major problem is the maintenance of software so that it is robust and reliable. Immense software structures are needed to solve some of the most simply stated mathematical problems, for example, computation of an indefinite integral. The question of how to organize a large software system that can accumulate, over time, the machinery necessary to provide facilities for a far-reaching and general mathematical system is a difficult one.

2.2 Algorithms and Theory

Just as symbolic computation would not exist as we know it today without the significant accomplishments in designing and building software systems, the same can be said about accomplishments in algorithms and theory. Many of the significant computations that can be done today would be impossible without the important advances that have been made in algorithms, especially the progress on faster algorithms for fundamental, ubiquitous operations like computing greatest common divisors of polynomials and factorization.

Other recent advances in algorithms and theory span a wide range, including fast algorithms for operations like integer factorization, computational methods in group theory, nonlinear algebra, the computation of closed-form solutions in integral calculus, and others. In some areas, algorithm development dwindled in the late nineteenth and early twentieth centuries because the algorithms had reached the limit of what could be done by hand. The advent of the computer and symbolic computation has spurred algorithm development for two reasons: additional horsepower provided by computers makes more complex algorithms feasible, and additional applications being run on computers make algorithm development necessary.

Much of science and engineering is put in terms of equations. Often we use linear approximation rather than higher-degree equations because of the difficulty in dealing with nonlinear equations. In the mid-sixties Buchberger in his dissertation[15] presented a method, based on what he later called

[14] M. Melenk, H. M. Möller, and W. Neun, On Gröbner bases computation on a supercomputer using REDUCE, Preprint SC 88–2 (Jan. 1988), Konrad-Zuse-Zentrum für Informationstechnik—Berlin.

[15] B. Buchberger *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*, Ph.D. Thesis, University of Innsbruck, December, 1965.

Gröbner bases, for effectively dealing with multivariate polynomial equations of arbitrary degree. This work received little attention from the mathematical community. In the mid-seventies, when the computer algebra community discovered Buchberger's work, his method became the basis of a powerful set of tools for working with higher-degree polynomial equations. A recent bibliography on Gröbner basis–related work contains almost 300 items, and there continues to be much research activity based on Buchberger's work.[16]

Buchberger's methods are but one approach to working with nonlinear equations. In the mid-seventies George Collins developed a method for finding exact real solutions to equations.[17]

In the areas of integration in finite terms and finding closed form solutions to differential equations, fundamental progress has been made on long-standing problems. Today, indefinite integration of elementary functions with elementary integrals[18] is conveniently done by symbolic computation systems even though many of the latest algorithm improvements are yet to be implemented. A decision procedure exists for solving n-th order homogeneous linear ordinary differential equations,[19] and an implementation of a different algorithm[20] that works only for second order equations has proved effective in practice. A method and an implementation also exist for solving first order nonlinear equations (or equivalently for solving two-dimensional autonomous systems).[21] Progress has also been made on symbolic methods for computing series solutions to differential equations.[22]

Another example of algorithm development concerns factoring polynomials. While superficially the problem seems related to the computationally difficult problem of factoring integers—Isaac Newton[23] based his polynomial

[16] B. Buchberger, Gröbner bases: an algorithmic method in polynomial ideal theory, in N. K. Bose (ed.), *Multidimensional Systems Theory*, D. Reidel Publishing Co., Hingham, Massachusetts (1985), pp. 184–232.

[17] D. S. Arnon, G. E. Collins, and S. McCallum, Cylindrical algebraic decomposition I: the basic algorithm. *SIAM J. Comput.* **13** (1984), pp. 865–877.

[18] R. H. Risch, The problem of integration in finite terms, *Trans. Amer. Math. Soc.* **139** (1969), pp. 167–189.

[19] M. F. Singer, Liouvillian solutions of nth order homogeneous linear differential equations, *Amer. J. Math.* **103** (1981), pp. 661–682.

[20] J. J. Kovacic, An algorithm for solving second order linear homogeneous differential equations, *J. Symbolic Comp.* **2** (1986), pp. 3–43.

[21] M. J. Prelle and M. F. Singer, Elementary first integrals of differential equations, *Trans. Amer. Math. Soc.* **279** (1983), pp. 215–229.

[22] J. Della Dora, Cl. di Crescenzo, and E. Tournier, An algorithm to obtain formal solutions of a linear homogeneous differential equation at an irregular singular point, *Computer Algebra—EUROCAM '82, Lect. Notes Comp. Sci.* **144** (1982), pp. 273–280.

[23] *Arithmetica Universalis*, 2nd ed., London (1728). Reprinted in D. T. Whiteside (ed.), *The Mathematical Works of Isaac Newton*, vol. 2, Johnson Reprint Corp., New York (1967).

algorithm on factoring their integral values and interpolating all possible factor combinations—the algorithms invented within the past twenty-five years have made it possible to factor large multivariate polynomials over many coefficient domains.

Modern research begins with the algorithms by E. Berlekamp[24] for factoring a polynomial in a single variable over a finite field. Berlekamp's contributions are twofold. First, the 1967 algorithm for small coefficient fields is the first indication that polynomial factoring is computationally simpler than integer factoring. Over the finite field with two elements, one can factor a polynomial of degree n, represented by Berlekamp by an n-digit integer, in cubic time in n, but to-date the corresponding integer factoring problem requires exponential time in n. Berlekamp's 1970 paper concerns large coefficient fields. In order to obtain a similar running time, Berlekamp introduced the use of random elements into his method. His algorithm is probably the first instance of a randomized solution to a problem whose deterministic best solution is exponential. It should be noted that, despite the inefficient deterministic algorithms known, by randomization one factors polynomials over large finite fields routinely on computer algebra systems, perhaps without the users even realizing that the method is of a probabilistic nature.

Shortly after Berlekamp resolved the problem of factoring polynomials over finite fields, H. Zassenhaus,[25] following a direction in van der Waerden's 1936 book *Moderne Algebra*,[26] employed the so-called Hensel lemma of p-adic number arithmetic to "lift" factors of an integral polynomial that are first computed modulo a suitable prime by Berlekamp's algorithm to full-fledged integral factors. Hensel lifting, as the process is now called, is a generic approach[27] to reconstruct factors from their modular images. Unlike interpolation, which requires several images, Hensel lifting requires only a single image.

Several packages are in existence that realize the Berlekamp-Zassenhaus scheme, for example, the MACSYMA, MAPLE, REDUCE, SAC/2, and MATHEMATICA polynomial factorization codes. This class of algorithms suffers from two major shortcomings. First, the algorithms exhibit exponential running time behavior on certain special inputs. This turns out to be a serious problem, since another algorithm, the Kronecker algorithm for factoring polynomials over an algebraic extension of the rational numbers[28] relies ex-

[24]E. R. Berlekamp, Factoring polynomials over finite fields, *Bell Systems J.* **46** (1967), pp. 1853–1859; ____, Factoring polynomials over large finite fields, *Math. Comput.* **24** (1970), pp. 713–735.

[25]H. Zassenhaus, On Hensel factorization I, *J. Number Theory* **1** (1969), pp. 291–311.

[26]B. L. van der Waerden, *Modern Algebra*, F. Ungar Publishing Co., New York (1953).

[27]D. R. Musser, Multivariate polynomial factorization, *J. ACM* **22** (1975), pp. 291–308.

[28]B. Trager, Algebraic factoring and rational function integration, in *Proc. 1976 Symp.*

actly on the factorization of such bad polynomials. Secondly, for multivariate polynomials, the algorithms work with a dense encoding of the polynomials and experience fill-in problems similar to that of Gaussian elimination when applied to a sparse matrix.

Putting the problem of multivariate polynomial factorization into the class of problems solvable in polynomial time is a lasting accomplishment. There are essentially two key ingredients to the solution. The first comes from the geometry of numbers and is an ingenious diophantine optimization algorithm, the so-called lattice reduction algorithm of Lenstra, Lenstra, and Lovász, that is used in conjunction with the Berlekamp-Zassenhaus method.[29] The second is a family of effective Hilbert irreducibility theorems due to Kaltofen[30] that guarantee that the lifting process will start out with good modular factors. Other researchers, among them A. L. Chistov, J. von zur Gathen, D. Yu. Grigoryev, S. Landau, A. Schönhage, and P. Weinberger, have also made important contributions to this problem.

The investigations into sparsity preserving multivariate polynomial factorization were begun in 1979 by R. Zippel.[31] This is the second subject where randomization is a crucial tool to make the algorithms work efficiently. Perhaps the key contribution to the resolution of the problem is not a clever algorithm but the realization that the sparse representation of multivariate polynomials is not a natural one. In fact, this can be deduced from the way we represent multivariate polynomials in mathematical notation, be it as a Toeplitz determinant or a product-of-summations formula. There are two models of representation that have been suggested, one is Strassen's straight-line program representation,[32] and one is the representation by a "black-box" program for computing values of the polynomial when given values for the variables. It is certainly surprising that for the polynomial factorization problem both representations are stable; that is, given a multivariate polynomial in straight-line or a black-box representation, one can efficiently compute a straight-line program[33] or a black box,[34] respectively,

---

on Symbolic and Algebraic Computation, ACM (1976), pp. 219–228.

[29]A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, Factoring with rational coefficients, *Math. Ann* **261** (1982), pp. 515–534.

[30]E. Kaltofen, A polynomial reduction from multivariate to bivariate integral polynomial factorization, in *Proc. 14th Ann. ACM Symp. Theory Comput.*, ACM (1982), pp. 261–263.

[31]R. E. Zippel, Newton iteration and the sparse Hensel algorithm, in *Proc. 1981 Symp. on Symbolic and Algebraic Computation*, ACM (1981), pp. 68–72.

[32]V. Strassen, Berechnung und Programm I, *Acta Inf.* **1** (1972), pp. 320–335.

[33]E. Kaltofen, Uniform closure properties of p-computable functions, in *Proc. 18th Ann. ACM Symp. Theory Comput.* ACM (1986), pp. 330–337.

[34]E. Kaltofen and B. Trager, Computing with polynomials given by black boxes for their evaluations: greatest common divisors, factorization, separation of numerators and denominators, in *Proc. 29th IEEE Symp. Foundations Comput. Sci.* IEEE (1988), pp.

that represents all irreducible factors of the multivariate polynomial. Sparse factors can be retrieved efficiently from these straight-line programs or black boxes by several of the recently discovered sparse multivariate interpolation algorithms.[35] It is important to observe that the black-box representation has broken a major barrier in computer algebra, namely that of intermediate expression size growth. The black-box programs for the factors, for example, have more or less constant size—they accomplish the evaluation of the factors by making within a loop calls to the black box of the input polynomial. In the future, this representation can perhaps be adopted to other problems in computer algebra that currently suffer from large intermediate expression size, such as the Gröbner basis problem.

Recent work on polynomial factorization has focused on the problem of representing algebraic coefficient fields[36] and on the problem of factoring polynomials, especially the bivariate defining equations of algebraic curves, over the real or complex numbers.[37]

As demonstrated in the preceding discussion on polynomial factorization, an important idea in algorithm design is the use of probabilistic algorithms. Probabilistic algorithms suffer from certain likelihoods of failure, because they may occasionally either give incorrect answers (the Monte Carlo variety) or fail to produce an answer in polynomial time (the Las Vegas variety). For several problems, researchers have discovered probabilistic algorithms that are much more efficient than any possible deterministic algorithms. Another notable example in symbolic computation is Schwartz's probabilistic algorithm for verifying polynomial identities.[38] The use of probabilistic algorithms in symbolic computation deserves to be more intensely studied, especially for problems with intrinsically high deterministic complexity.

Another recent major consideration in algorithm development is the problem of taking advantage of new parallel computer architectures. For many problem classes, the algorithms that are best suited for sequential ma-

---

296–305.

[35]R. E. Zippel, Interpolating polynomials from their values, *J. Symbolic Comp.*, (in press).

[36]J. A. Abbott, Recovery of algebraic numbers from their p-adic approximations, in *Proc. 1989 Intl. Symp. Symbolic Algebraic Comput.*, ACM (1989), pp. 112–120.

[37]A. Schönhage, The fundamental theorem of algebra in terms of computational complexity, Technical Report, University Tübingen, 1982; D. Duval, Absolute factorization of polynomials: a geometric approach, Technical Report **103** (1988), University of Grenoble; E. Kaltofen, Computing the irreducible real factors and components of an algebraic curve, *Proc. 5th ACM Symp. Comput. Geometry*, ACM (1989), pp. 79–87; C. Bajaj, J. Canny, T. Garrity, and J. Warren, Factoring rational polynomials over the complexes, in *Proc. 1989 Intl. Symp. Symbolic Algebraic Comput.*, ACM (1989), pp. 81–90.

[38]J. T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, *JACM* **27** (1980), pp. 701–717.

chines are not necessarily the best ones when converted into algorithms for parallel machines. In fact, the available and soon-to-be available parallel architectures are of such wide variety that the optimum use of each may warrant the development of quite different algorithms.

These are just some examples of recent advances in algorithm development. Additional algorithmic advances and seminal problems are surveyed in Kaltofen's article.[39] Other overviews have been given by Winkler[40] and Caviness.[41] However, despite the important and deep progress that has been made on new symbolic computation algorithms, nothing more than a substantial beginning has been accomplished. A systematic investigation of better algorithms is needed in many fundamental areas, including linear algebra, nonlinear algebra, vector calculus, ordinary differential equations—especially systems, partial differential equations, and complex function theory. Research is also needed on algorithms that take advantage of special characteristics of important applied problems.

Significant advances in computational power depend on the development of computationally oriented mathematics leading to new algorithms. That this happens, in algebra, is illustrated by Buchberger's ideal theoretic techniques, as contrasted with Kronecker's ideal theoretic techniques. A new generation of computational scientists with intense mathematical and computational training related to symbolic computation and working in computationally oriented research programs is required.

2.3 Numeric and Symbolic Computation

Daniel Lazard, writing in the preface to the new book *Computer Algebra—Systems and Algorithms for Algebraic Computation*,[42] notes a curious anomaly about the term *scientific computation* as it has come to be used in the last thirty years as a result of the way computers have been used to do computation.

> This concept of "scientific calculation" conceals an ambiguity, which it is important to note: before computers appeared on the scene, a calculation usually consisted of a mixture of numerical calculation and

[39] E. Kaltofen, Computer algebra algorithms, *Ann. Rev. Comput. Sci.* **2** (1987), pp. 91–118.

[40] F. Winkler, Computer algebra, in *Ency. of Physical Science and Technology* **3**, Academic Press (1987), pp. 330–356.

[41] B. F. Caviness, Computer algebra: past and future, *J. Symbolic Comp.* **2** (1986), pp. 217–236.

[42] J. H. Davenport, Y. Siret, and E. Tournier, *Computer Algebra—Systems and Algorithms for Algebraic Computation*, Academic Press, London (1988), pp. vi-vii.

> what we shall call "algebraic calculation", that is calculation by mathematical formulae. The only example of purely numerical calculation seems to have been the feats of calculating prodigies such as Inaudi: the authors of tables, especially of logarithms, did indeed carry out enormous numerical calculations, but these were preceded by a restatement of the algebraic formulae and methods which were essential if the work was to be within the bounds of what is humanly possible. For example the famous large calculations of the 19th century include a large proportion of formula manipulation. The best known is certainly Le Verrier's calculation of the orbit of Neptune, which started from the disturbances of the orbit of Uranus and led to the discovery of Neptune. The most impressive calculation with pencil and paper is also in astronomy: Delaunay took 10 years to calculate the orbit of the moon, and another 10 years to check it. The result is not numerical, because it consists for the most part of a formula that by itself occupies all the 128 pages of Chapter 4 of his book.
>
> The ambiguity mentioned above is the following: when computers came on the scene, numerical calculation was made very much easier and it became commonplace to do enormous calculations, which in some cases made it possible to avoid laborious algebraic manipulations. The result was that, for the public at large and even for most scientists, numerical calculation and scientific calculation have become synonymous.

The emergence of symbolic computation software is once again making possible the fruitful and natural symbiosis, noted by Lazard, between symbolic and numeric computation. As was historically true, the principal focus is on restating algebraic formulas and methods to particularize them to the problem at hand, so that the eventual numeric computation can be cheaper, more accurate, and less sensitive to errors. Hearn estimates that over 50% of the REDUCE usage on mainframes is spent producing FORTRAN code. Symbolic computation is thus an important tool for scientists and engineers seeking numerical answers. It can be an even more important tool for the numerical analyst, that is, someone whose primary interest is in the design and analysis of numeric methods. The analysis and comparison of numeric methods often require richer mathematics than does mere derivation.

Beyond derivation and analysis of numeric methods, there are other ways in which the combination of symbolic and numeric computation can be effective. Many problems involve specifying a model in algebraic form with undetermined coefficients that are fit statistically to observed data, but subsequent symbolic manipulation of the model is needed. Numerical exploration of complicated closed form solutions to problems can investigate questions that are not tractable symbolically, or can expose behavior that can then be established formally. On the other hand, symbolic exploration of complicated expressions can establish properties such as nonnegativity,

monotonicity, relative maxima, or asymptotes that may be interesting for understanding the qualitative numerical behavior of the expression or may be important for choosing the preferred numeric algorithm to use for a problem involving that expression.

Unfortunately, some potential benefits of the combination of symbolic and numeric computation are not yet realizable because symbolic computation software does not yet provide the mathematics needed to support numerical analysis. The indeterminates in symbolic systems have most commonly represented rational or real values, whereas the algebra for numeric methods often requires indeterminates that represent complex numbers, or matrices in N-space (where N is symbolic), or even functions. Manipulation of inequalities or of operator algebra is just beginning to become available. Commonly used knowledge, such as the standard matrix factorizations, is not yet built into the symbolic computation systems. These shortcomings can be circumvented for specific applications, but the impression of awkwardness and user-unfriendliness is left. The symbolic computation systems are not yet considered to be of production quality by the numeric computation community.

Another barrier to greater use of symbolic systems by the numeric computation community concerns what it means to solve a problem. Although there has been some symbolic work with approximation, principally using power series, by and large the objective of symbolic computation has been to obtain exact answers. By contrast, exact answers are often worthless for the problems addressed by numeric computation, since they may not exist, or they may be too complicated to be useful or even insightful. Often, the mathematical problem to be solved is only an approximation to the real physical problem. Thus, approximation is essential to addressing the problems of interest to the numeric computation community. Many types of approximation are used, including linearization of functions, approximation of operators, and expansion in many different bases from Fourier series to Bessel functions to B-splines to Sobolov bases, with error term representations from difference formulations to integral formulations to Rolle's theorem to formulations as poles of functions of a complex variable. Approximation in this rich setting could exploit symbolic computation, but because of the clash in objectives, there is as yet no support for it.

Hamming's famous quotation,[43] "The purpose of computing is insight, not numbers," can be paraphrased as, "The purpose of computing is insight, not formulas," for symbolic computation. The assertion is not strictly true, for we have seen production of executable code as one important benefit.

[43]Richard W. Hamming, *Numerical Methods for Scientists and Engineers*, McGraw-Hill, New York (1962), p. v.

Nevertheless, it is largely the case that human understanding of input to and output from scientific computation, both symbolic and numeric, is essential. That implies that mathematical notation and graphic representations are essential for both input and output.

2.4 Education

Research and education have always been closely linked. This is especially true for symbolic computation. To have better symbolic computation research we need better education and vice versa. In the area of symbolic computation, both research and education face the difficulty that symbolic computation does not fall neatly within one discipline. Symbolic computation lies at the frontier of several fields, notably computer science and mathematics. The discipline of computer science has grown out of mathematics and electrical engineering. For much of the last twenty years, the computer science community has looked inward, and rightly so, to understand and shape the nature of the discipline. Much important work has been accomplished. Now there are important links to be reestablished. As noted by Steen,[44] computing has had fundamental influences on mathematics. Now is an opportune time to reunite aspects of computer science and mathematics and to bring symbolic and numeric computation together in a new synergism. To accomplish this, it is necessary to face the task of educating the current and future scientific community.

Three concerns regarding symbolic computation and education are:

- education about symbolic computation—what is available and how to use it
- development of more researchers in symbolic computation
- use of symbolic computation as a teaching tool

A common problem in all three areas is a lack of human resources.

A key to the future development of the field of symbolic computation is the education and development of current and future researchers. The education of potential researchers in the area of symbolic computation is essentially nonexistent in the United States. Currently, there is no generally established curriculum in the United States for graduate students who wish to work in this area. At most American institutions with graduate mathematics and computer science programs, graduate students have no opportunity to study this area, no faculty are doing research in the area,

[44]The quote appears later in this section.

and there are few directly relevant courses. Furthermore, there are few, if any, postdoctoral positions for scientists and engineers interested in symbolic computation. Postdoctoral positions are important training vehicles for interdisciplinary fields like this one.

To capitalize fully on the potential of symbolic computation in science, engineering, and education, better and different tools are needed. An important job for future symbolic computation research and software development is to provide these tools. However, the lack of human resources to do the needed research makes this a difficult problem to address.

The most fully developed educational program on symbolic computation is at the Johannes Kepler University in Linz, Austria. A description of this program appears in Appendix B. Ten years ago there were several small, but dynamic symbolic computation educational programs at universities in the United States. MIT, Wisconsin, Utah, Rensselaer, and Berkeley produced a steady supply of symbolic computation researchers. On a worldwide basis, over the past ten years, the trend in symbolic computation education and the production of symbolic computation researchers has been one of decline in the United States and growth in such countries as Austria, Canada, France, Italy, Japan, and the United Kingdom.

There is a lack of awareness about what present-day symbolic manipulation systems can do. This was highlighted at the April, 1988 workshop on symbolic and algebraic computation when attendees lamented that there was no current system having a certain capability, only to be told that some system other than the one they were currently using did have that facility. If sophisticated symbolic computation users are unaware of general facilities, the problems for the casual and neophyte users must be legion. Most people doing research in mathematics, science, and engineering are unfamiliar with even the best-known symbolic computation systems. It would be difficult for them to develop or fully appreciate a curriculum involving symbolic computation. A reason for this state of affairs is that few educational materials, especially textbooks, are available in this area. A list of the few such books is given in Appendix C. The development of more educational materials for symbolic computation would help increase awareness substantially. The recent, more aggressive marketing of symbolic computation systems should also help with the awareness problem.

Symbolic computation is destined to have a dramatic impact on mathematical and all technological education. The use of computers in undergraduate education is actively being deliberated. Inevitably, symbolic computation will become an integral part of scientific and engineering courses. Symbolic computation will not only shape teaching methodology; it will shape course content as well.

In his retiring presidential address to the Mathematical Association of America on January 8, 1988,[45] Steen explored the important relationship between computation, especially symbolic computation, and the teaching of mathematics. In so doing he exhorted his listeners to

> think, as many did at the NRC colloquium on Calculus for a New Century, about the contrast between the five thousand exercises in typical calculus books that mostly ask students to imitate calculator buttons, and the discovery potential in symbolic computer systems ....

Steen goes on to observe:

> Computers influence mathematicians not only by providing new tools for research and teaching, but also by posing deep questions about central issues in our discipline. Now that calculators manipulate symbols and calculate answers,
> - What—if not arithmetic—should be the core of elementary school mathematics?
> - What—if not manipulation—should be the core of high school algebra?
> - What—if not calculation—should be the core of calculus?
> - What—if not calculus—should be the core of college mathematics?
>
> At the same time that computers force attention on issues that are deeply rooted in unexamined tradition, mathematical research has transformed the nature of mathematics, opening up new options for what might be considered central and what derivative among the concepts of mathematics.
>
> We need to find new threads of continuity with which to weave a mathematics curriculum for the twenty-first century. Finding appropriate central themes poses an immense challenge for the best minds among us, researchers and teachers alike. It gives common purpose to our diverse expertise, and sets a common agenda for those in research, those in college teaching, and those in school mathematics.

The question is no longer whether symbolic computation ought to be used in scientific and engineering courses but rather how it can be used most effectively. The current role of symbolic computation in education is minimal. Several reasons for this exist, the most obvious ones being the lack of adequate facilities at many institutions and the lack of educational materials. However, with cost of computer equipment coming down and the attitude of university administrations toward the access of equipment changing, this is not the obstacle that it used to be. The development of educational materials and textbooks should be encouraged.

Some institutions are already experimenting with symbolic computation systems for educational purposes. The University of Waterloo has conducted

[45] L. A. Steen, Celebrating mathematics, *Amer. Math. Monthly* **95** (1988), pp. 414–427.

one of the most extensive experiments.[46] At St. Olaf College, where the use of computer algebra systems began in undergraduate mathematics several years ago, Paul Zorn stated that computer algebra systems

> offer important new possibilities for mathematical teaching and learning. They offer, for the first time, a kit of powerful mathematical tools—*graphical, algebraic, and numerical*—at acceptable cost in time and distraction. Using such tools, we and our students can represent and manipulate mathematical ideas more efficiently, effectively, and flexibly than before. At a minimum, MATHEMATICA and other systems offer computational leverage. At best—and I believe we can expect this—MATHEMATICA can help us foster genuinely deeper understanding of mathematical ideas.[47]

2.5 Funding for Research

In the past the Department of Energy, the Army Research Office, the National Science Foundation, and the Systems Development Foundation have provided much of the funding for symbolic computation research activities. For example, in the late 1960s and early 1970s the Department of Energy provided much of the support for the development of MACSYMA. In fiscal year 1987 the National Science Foundation created the Numeric and Symbolic Computation Program, which had as one of its primary objectives to support symbolic computation research. However, mainly because of a lack of proposals, the funds, as can be seen from Table 2.1, devoted to symbolic computation research have been minimal. The lack of proposals is another indication of the need for more researchers in this field.

Table 2.1: **Funding for Symbolic Computation from the NSF Program for Numeric and Symbolic Computation**

Fiscal Year	1984	1985	1986	1987	1988
Amount	\$14K	\$252K	\$73K	\$254K	\$359K

To provide an estimate of current funding by federal sources for symbolic computation research, an informal survey of federal agencies was carried out. Agencies were asked to identify 1988 fiscal year expenditures for research on symbolic algorithms, symbolic software, and applications. The estimates obtained are given in Table 2.2.

[46] B. W. Char et al., Computer algebra in the undergraduate mathematics classroom, in *Proc. 1986 Symp. on Symbolic and Algebraic Computation*, ACM (1986), pp. 135–140.

[47] Paul Zorn, MATHEMATICA in undergraduate mathematics, *Notices AMS* **35** (1988), pp. 1347–1349.

Table 2.2: **Funding for Symbolic Computation Research in Fiscal Year 1988**

Division of Electrical, Communications & Systems Engineering (Program in Systems Theory & Operations Research), NSF	$50,000
Division of Emerging Engineering Technologies (Program in Computational Engineering), NSF	423,255
Division of Computer and Computation Research, NSF	560,665
Division of Mathematical Sciences, NSF	547,019
Office of Undergraduate Science, Engineering & Mathematics Education (Program on Course & Curriculum), NSF	68,436
Air Force Office of Scientific Research	0
Army Research Office	250,000
Department of Energy	0
National Security Agency (computational number theory)	200,000
Office of Naval Research (research in related areas)	300,000
Total	**$2,399,375**

A few companies in the United States are carrying out commercial research and development in symbolic computation. The largest group is at the IBM Research Center in Yorktown Heights, New York. Research and development is also being done at Franz, Inc., in Berkeley, California; at the Soft Warehouse in Honolulu; at Symbolics, Inc., in Cambridge, Massachusetts; at Tektronix Computer Research Laboratory in Beaverton, Oregon; at Wolfram Research in Champaign, Illinois; and at Xerox PARC in Palo Alto, California.

A tabulation of the total expenditures by these companies for research and development efforts in symbolic computation is not available, but based on the number of persons involved, it is almost surely less than $5 million per year, but is increasing.

2.6 Current Problems

In a discussion of the current status of symbolic and algebraic computation, several problems become apparent. In broad terms, these can be categorized as research, education, and technical problems.

Research. Many research problems need attention, but a serious problem facing the advancement of this area in the United States is the lack of

human resources. There are few researchers working in this area and few programs to educate new persons in the field. This is a chicken and egg problem. It is difficult to develop a body of researchers in symbolic and algebraic computation without adequate undergraduate and graduate education in the area. It is equally difficult to implement an effective educational program in symbolic computation without the critical mass of researchers to develop the curriculum and teach the courses.

Symbolic computation faces the special problems within academia and within potential funding agencies that confront all interdisciplinary research. The interdisciplinary nature of symbolic and algebraic computation is an impediment to advancing research, training people, and promoting applications of this technology. Symbolic computation research involves, at the minimum, a knowledge of mathematics and computer science and often involves other sciences and engineering. Consequently, there is much to assimilate to reach the frontiers of research and much material on which to remain current. At present, computer science researchers in symbolic and algebraic computation would benefit from a better understanding of the relevant progress in mathematics. On the other hand, too few academic mathematicians in the United States have become involved in symbolic computation research. This has both hurt the research effort in the United States and delayed the introduction of symbolic computation into mathematical education.

Education. There are three facets to the educational problems in symbolic computation:

- educating the scientific and engineering communities about the capabilities and use of symbolic computation systems
- educating potential researchers in this field
- using symbolic computation in general scientific education

Knowledge of what is possible with symbolic computation and the distinctions between the various symbolic packages is not widespread. One person who regularly gives talks about symbolic computation to the science and engineering communities estimates that 80% to 90% of the attendees at his talks have never heard of symbolic computation. Furthermore, those who do use symbolic computation systems have significant difficulty using them effectively. A better awareness of these systems and how to use them in the scientific and engineering communities is needed.

The education of potential symbolic computation researchers in the United States is essentially nonexistent. At present, there is no generally established curriculum in this country for graduate students who wish to work in this area. At the overwhelming majority of American institutions with graduate

mathematics and computer science programs, it would be difficult to write a Ph.D. thesis in symbolic and algebraic computation because of a lack of relevant courses and knowledgeable advisers. A graduate student wishing to pursue such studies has only a few choices of institutions and departments.

It is well accepted that symbolic computation systems will eventually become a standard part of the curriculum in calculus and other undergraduate education courses. The integration of symbolic computation into these courses, however, is hampered by the lack of awareness in the scientific community.

Technical problems. Numerous technical problems impede the accessibility of symbolic computation systems to the community. The systems are not necessarily user-friendly, and the documentation is often unintelligible. Symbolic and algebraic computation systems are too difficult to use.

A multitude of symbolic computation systems is currently available. However, they do not interact well, if at all. Reusable software components are desperately needed so that new software developments do not have to start from scratch. Better interfaces between symbolic computation software and other software must be devised, especially for numeric software. More software is needed to implement various approximation methods in symbolic computation to complement exact, closed form solutions and to make it possible to solve a wider range of applications problems.

In summary, the current software has many positive and negative attributes. A significant improvement of the software platform could substantially accelerate research on, and applications of, symbolic computation.

Technology transfer. The development and application of effective symbolic computation methods and facilities involve a broad range of activities from fundamental research to commercial software development. The roles of universities, funding agencies, and private enterprise must be worked out for the mutual benefit of all. Some issues affecting the commercialization of this technology are:

- Economies of scale are formidable since entry and enhancement costs are large compared to the size of the market.

- The adoption of symbolic mathematics software technology in the low and middle segments of the market is limited more by shortages of resources for commercialization of technology that is already available than by the need for new system design and algorithms.

- Efforts to increase ease of use through on-line and paper documentation and through improved user interfaces primarily require development resources and user-driven direction rather than more research effort.

- Some improved algorithms, for example, for integration, solving differential equations, and factoring, developed in the 1980s have not been implemented in many systems because resources have been focused on porting, debugging, and improving ease of use.

A better dialogue between academic research and commercial efforts will expedite the transfer of symbolic computation technology.

Funding. The level of funding for research in this area is low. The funding level has remained low at a time when funding for scientific and engineering computation has been increasing rapidly. Reasons for this are not completely clear, but perhaps the increase in funding for numeric computation has masked the low levels of funding for symbolic computation. Furthermore, the small number of researchers in symbolic computation has not put much demand on funding agencies.

Chapter 3

Applications in Science and Engineering

There are at least three characteristic ways in which symbolic computation systems are used: (1) to do computations that could be carried out by hand, but can be done more productively and accurately by a symbolic computation system, (2) to do computations that are beyond hand calculation but can be done more or less routinely by machine, and (3) to do calculations that require substantial effort to perform even when using a computer.

There are four principal advantages according to MacCallum.[1]

> (i) it is exact. In particular most computer algebra systems provide exact arithmetic on indefinitely large integers, and many provide arbitrarily high precision floating point arithmetic. The calculations are not usually of the sort requiring error analysis or studies of convergence and stability.
> (ii) it removes the tediousness of lengthy routine calculations. Two well-known examples are the calculation of the orbit of the moon, which took Delaunay 20 years, and has more recently [been] done by machine, including writing the software from scratch, in about 9 months, and the calculation of the curvature for the Bondi metric in general relativity, which originally took about 6 months, but was done by ALAM in 1967 in 4 minutes and can be done now in as little as 8 seconds plus printing time ....
> (iii) it is accurate. In both the examples cited in the previous paragraph, the machine version identified corrections needed in the hand calculations.
> (iv) (perhaps the most interesting) it enables previously unthinkable calculations to be attempted. This may focus attention on aspects that

[1] M. A. H. MacCallum, Algebraic computing in relativity, in J. M. Centella (ed.), *Proc. of a Workshop on Dynamical Spacetimes and Numerical Relativity*, Cambridge University Press, New York (1986), pp. 411–445.

> cause a fundamental re-appraisal of the analytic approach. The resulting paper may in the end not even mention the brute force calculations which suggested the more elegant final analysis (for some examples see d'Inverno 1976[2]).

Although symbolic computation is widely applicable, MacCallum noted that many applications are never reported in the literature. Odlyzko[3] reinforced this when he wrote:

> Some of the most interesting applications of symbolic mathematics are in mathematics itself. Areas of both pure and applied mathematics, including coding theory, cryptography, probability theory, analysis, combinatorics, and number theory, have all gained from the availability of the new symbolic manipulation tools. These tools have been used to prove a number of results directly. Their main application, however, has been to obtain insight into behavior of various mathematical objects, which then led to conventional proofs being constructed. ... My experience with symbolic mathematics goes back over ten years. Most of this work was with MACSYMA, although at various points I or my collaborators have used other systems such as ALTRAN, MAPLE, and SMP. These systems were used in many fields of mathematics ... . Still, these extremely varied examples do not cover the full range of applications that have been made, and are a reflection of my research interests. ... I see the main role of symbolic algebra systems as that of helping to formulate hypotheses, search for examples and counterexamples, and in general explore ramifications of mathematical models. *In other words, the main role of these systems is to obtain mathematical insight. Once that insight is obtained, one can then go on and construct canonical mathematical proofs, in which there might not even be any traces of the use of computer algebra* [editors' emphasis].

This is the nature of symbolic computation—it is a tool that increases scientific and engineering productivity.

---

[2]R. A. d'Inverno, Algebraic computing in general relativity, *Gen. Rel. and Grav.* **6** (1975), pp. 567–593.

[3]A. M. Odlyzko, Applications of symbolic mathematics to mathematics, in R. Pavelle (ed.), *Applications of Computer Algebra*, Kluwer Academic Press, Boston (1985), pp. 95–111.

A sampling of some of the problems to which symbolic computation has been applied demonstrates the scope and importance of this mode of computation. Areas to which symbolic computation has been applied include:

- Physics—astrophysics, quantum electro- and chromo-dynamics, general relativity, optics, plasma physics, celestial mechanics, fluid mechanics, quantum mechanical perturbation theory, calculation of dissociation energies for various atoms

- Chemistry—molecular electronic structure calculations, kinetic energy operators for molecule based internal coordinates, solutions to first order rate equations, steady state concentrations for complex reacting systems, quantum mechanics of molecular pseudorotation

- Mathematics—numerical analysis, coding theory, cryptography, probability theory, analysis, combinatorics, number theory, real closed fields, group theory, geometry, topology, commutative algebra, algebraic geometry, proof of the Macdonald-Morris conjecture, understanding of Ramanujan's mock theta functions

- Chemical engineering—column flow rate parameters in absorption packed beds and in chemical reactors

- Nuclear magnetic resonance—expansion of Wigner rotation matrix term, examination of magic angle spinning expressions, computation of expressions for the angular dependence of slow molecular motion in solids, the calculation of the antisymmetric part of the chemical shift tensor, computation of closed expressions for nuclear magnetic resonance line shapes

- Engineering—electrical network analysis, turbine design, ship hull design, hydrodynamic lubrication, helicopter rotor design, control theory, image and signal processing, antenna design

Descriptions of a few applications illustrate the breadth and relevance of symbolic computation.

3.1 High Energy Physics

This is an area in which symbolic computation systems have been routinely and extensively used since their early days, and probably more application papers have been published on this topic than any other. High-energy physics research has also provided an impetus for system development, since

several systems for such calculations have evolved into general-purpose systems that are now widely used by engineers, mathematicians, and physicists, besides prompting the development of a number of special-purpose systems that are still in use today. As early as 1970, a computer algebra calculation of the Lamb shift in hydrogen[4] was considered one of the outstanding physics results of that year.

In this field more than any other, computer algebra is an indispensable tool for researchers. It is difficult, however, to describe the exact nature of such calculations in simple terms. In essence, one computes the outcome of a physical process in quantum electrodynamics or quantum chromodynamics by means of a perturbation theory expressed in diagrammatic form. These diagrams, normally called Feynman diagrams after their inventor, can be transformed into an algebraic expression by applying the so-called Feynman rules. This process is fairly straightforward, although there are systems available for doing parts of this automatically. The algebraic expression that results is a multidimensional integral that is almost impossible to evaluate numerically because of the instabilities involved. However, if some of the integrations can be done symbolically, the numerical calculation of the remaining integrals is much more manageable. The key problem is to do as many of these integrals as possible analytically by computer. Since they arise from the multidimensional integration of rational functions, they are sufficiently well structured that special-purpose packages can be written for this purpose. The sheer size of many of these calculations boggles the mind. At times, several million terms are manipulated in such calculations, whereas the final result is often quite small.

Modern mathematical methods in quantum field theories rely on algebraic techniques from algebraic topology and algebraic geometry, particularly in superstring and supersymmetry. Applications of computer algebra already appear in this area: see, for example, identities on elliptic curve characteristic classes derived using computer algebra[5] (Witten-Landweber-Stong conjecture).

3.2 Celestial Mechanics

An elementary exercise like solving the main problem in the theory of artificial satellites requires only basic operations in a well-structured set of func-

[4] T. W. Appelquist and S. J. Brodsky, The order α^2 electrodynamic corrections to the Lamb shift, *Phys. Rev. Letters* **24** (1970), pp. 562–565.

[5] D. and G. Chudnovsky, Elliptic formal groups over Z and Fp, *Applications to Number Theory, Computer Science and Topology, Lect. Notes Math.* **1326** (1988), pp. 11–54; ____, Elliptic modular functions and elliptic genra, *Topology* **27** (1988), pp. 163–170.

tions that are in part polynomials in several variables and in part trigonometric sums in several arguments. But these operations must be repeated many times; moreover, as the calculation proceeds, the number of terms in the intermediate results and the size of their coefficients grow quasi-exponentially. Astronomical calculations have been done on specially tailored systems that can handle the complexity of these calculations. This has been the trend ever since the late 1960s, when the grand schemes of Delaunay or Hill-Brown were automated to solve the main problem of lunar theory. Perhaps the reason is that from the beginning specialists in celestial mechanics felt that calculation automation would not work for them unless the traditional schemes meant to speed up hand calculations were replaced by algorithms specially designed for computers. Most significant in that regard has been the substitution of explicit Lie transformations for the implicit algorithms elaborated by Poincaré. As the software available grew in sophistication, specialists in celestial mechanics kept refining their *ad hoc* programs for manipulating what they call Poisson or d'Alembert series. This approach has produced results that can be described unreservedly as sensational. It is now possible to generate in less than half an hour on major computers the semianalytical solution of Hill-Brown for the main problem of lunar theory; furthermore, for the first time in the history of celestial mechanics, a complete semianalytical solution for the full theory, including the planetary perturbations, has been produced to meet the precision of current observations. The same has been done for all the planets in the solar system. The time is fast approaching when the traditional task of producing astronomical almanacs will be entirely automated on any type of computer—small or large, on board a satellite, or in the middle of an almanac office. The next job is to automate the production of FORTRAN codes from analytical solutions in the theory of artificial satellites. A preliminary model has already been realized, but further progress has been hampered by difficulties caused by small divisors and resonances.

André Deprit was awarded the James Craig Watson medal by the National Academy of Sciences "for his resolution of the problem of lunar motion around the earth through his adaption of modern computing machinery to algebraic rather than arithmetic operations. ... He has made it possible to correct the theory differentially whenever small changes in the initial conditions become necessary, without having to repeat the laborious analysis."[6]

Now that the classical problems are well within their grasp, astronomers have turned lately to awesome tasks like establishing the history of the eccentricity of the earth's orbit and of the inclination of the equator over the ecliptic for fifty to a hundred million years in the past and possibly for even

[6]Taken from the award documents, National Academy of Sciences, April 23, 1972.

longer. Besides helping geologists in relating periods in the ice ages with periods in the earth's insolation, as conjectured by Milankovich, it would establish astronomical time scales over geological periods. It is not altogether obvious that these problems have reached computational maturity. What is certain, however, is that astronomers will not succeed unless they identify the places where their private symbolic computation codes lend themselves to massively parallel computing.

3.3 Group Theory

The mathematical concept of a group was introduced by Galois in the 1830s. In the intervening 150 years, group theory has found application in many fields, including mathematics itself, physics, and chemistry. The classification of the finite simple groups is a major achievement of twentieth-century mathematics. A famous paper by Feit and Thompson started a series of events that led to the final classification: any finite simple group is an alternating group, is a finite version of a simple Lie group, or is one of 26 exceptional groups. The latter, the 26 sporadic groups, do not fit naturally into any of the infinite families of simple groups. The construction of several of these sporadic groups involved extensive machine computation. For example, the Lyons sporadic group was first constructed[7] as a permutation group on 8,835,156 points and more recently was found[8] as a group of 111-by-111 matrices over the field of five elements.

The finite simple groups are the building blocks from which all finite groups are constructed. To use the classification of simple groups to obtain general theorems about finite groups, it is necessary to have a great deal of information about the simple groups. Much has been learned, but much remains to be done. The character table of a group describes the representations of the group by complex matrices. Conway et al.[9] give the character tables of all the sporadic groups along with the tables of some of the smaller members of the infinite families. Most of these character tables were obtained using machine computation. Moreover, the computer often provides the best means for deducing additional properties of the groups from their character tables.

When the characteristic of the field divides the order of the group, information about the matrix representations of the simple groups is not nearly

[7] C. C. Sims, The existence and uniqueness of Lyons' group, *Finite Groups '72*. North-Holland (1972), pp. 138–141.

[8] W. Meyer, W. Neutsch, and R. Parker, The minimal 5-representation of Lyons' sporadic group, *Math. Ann.* **272** (1985), pp. 29–39.

[9] J. H. Conway, R. S. Curtis, S. P. Norton, R. A. Parker, and R. A. Wilson, *Atlas of Finite Groups*. Clarendon Press, Oxford (1985).

as complete as it is for characteristic zero. The computer will undoubtedly play a major role in exploring these modular representations. A basic tool here is the "Meat-Axe,"[10] an algorithm for deciding whether a matrix group is irreducible.

Another use of computers in group theory has been in the study of Burnside groups. If r and n are positive integers, then the Burnside group $B(r,n)$ is the most general group generated by r elements that has the property that the n-th power of every element in the group is the identity. The Burnside groups are known to be infinite when r is at least 2 and n is divisible by a large odd number. However, the groups are finite for some small values of n, and it is not known whether they are finite or infinite when n is a large power of 2.

If $n = 4$, then the Burnside groups are finite for all r, but the proof of this general result gives extremely crude upper bounds for the order of $B(r,4)$. It is not hard to show that the orders are 2^2 and 2^{12} when r is 1 and 2, respectively. The orders $|B(3,4)| = 2^{69}$ and $|B(4,4)| = 2^{422}$ were obtained[11] using implementations of the procedure known as the Nilpotent Quotient Algorithm (NQA). Recently, with an improved version of the NQA, M. F. Newman has obtained the upper bound 2^{2728} for $|B(5,4)|$ and has conjectured that this is the exact order. The Canberra NQA software has also been used[12] to show that the number of groups of order 256 is 56,092.

3.4 Chemistry

Although relatively new to chemistry, symbolic computation is rapidly becoming a standard tool in a wide variety of research areas. A typical common application involves predicting exactly how a molecule behaves in certain circumstances, such as when undergoing a chemical reaction. Complex mathematical molecular models, either ab initio or empirical, are necessary for this. In the more recently developed models, the required formulas and optimized computer code were obtained using symbolic computation. This is an important, if straightforward, application of symbolic computation in chemistry. It allows otherwise precluded chemical problems, such as those

[10] R. A. Parker, The computer calculation of modular characters (the Meat-Axe), in M. D. Atkinson (ed.), *Computational Group Theory*, Academic Press, London (1984), pp. 267–274.

[11] A. J. Bayes, J. Kautsky, and J. W. Wamsley, Computation in nilpotent groups (application), *Proc. Second Intl. Conf. Theory of Groups* (Canberra, 1973). *Lect. Notes Math.* **372** (1974), pp. 82–89; W. A. Alford, G. Havas, and M. F. Newman, Groups of exponent four. *Notices Amer. Math. Soc.* **22** (1975), p. A.301.

[12] E. A. O'Brien, *The Groups of Order Dividing 256.* Ph.D. Thesis, Australian National University, 1988.

appearing in the design of drugs, to be accurately studied. The motivation for greatly increased use of symbolic computation in molecular modeling is thus high.[13]

Symbolic computation is also used in some of the important macroscopic problems in chemistry. The concentrations of certain chemicals in living systems often vary in time according to first order rate laws. A formal solution to this problem is useful for obtaining and analyzing experimental rate constants. Symbolic computation allows solutions to much more complicated systems of this type than has been possible previously.[14]

Another recent and important application in macroscopic chemistry is the determination of the steady state or equilibrium concentrations of species in a complex reacting system.[15] These are solutions to systems of (usually low order) polynomials. Recently, systems of up to fifty species have been solved on a Cray X-MP. The Buchberger algorithm is used to transform the system to a Gröbner basis from which the solutions are directly obtained.

3.5 Numeric Computing

The mathematical library included with BSD 4.3 Unix is of considerable significance, not only because Berkeley Unix is so widely used, but also because this high-quality library has been widely copied to provide mathematical libraries on other systems. The function approximations used in this library are particularly accurate, having been carefully chosen to take into account IEEE standard arithmetic as well as the arithmetic of the VAX. The coefficients are optimal, given that they must be represented in the finite precision of the floating point arithmetic.

Kahan, MacDonald, and Tang achieved this by solving a series of nonstandard optimal approximation problems. Each problem of the series required a combination of symbolic and numeric steps. The problems involved choosing one coefficient whose optimal value had been determined in an earlier problem, adjusting that coefficient to an appropriate representable value, and solving the now nonstandard problem where the value of that coefficient is now fixed. Remes' algorithm used to solve each problem required the derivatives of the error to be accurately determined. The only practical

[13]M. McCourt and J. McIver, Symbolic algebra software, a useful tool for code developers, *QCPE Bulletin* **7** (1987), p. 69; P. Knowles and N. Handy, Projected unrestricted Moller-Plesset second-order energies, *J. Chem. Phys.* **88** (1988), p. 6991.

[14]W. Kreye, P. Batra, and G. Skinner, Analytic solutions to sets of first-order rate equations with up to six rate constants using a symbolic computer language SMP and application to biochemical kinetics, *J. Comp. Chem.* **9** (1988), p. 674.

[15]H. Melenk, H. M. Möller, and W. Neun, Symbolic solution of large stationary chemical kinetics problems, *Impact of Computing in Sci. and Eng.*, **1** (1989), pp. 138–167.

method to do this was to differentiate symbolically the error function, producing FORTRAN code that could be executed to yield numeric values. This code was then used in the numeric iteration to find the optimal coefficient values for that problem.

3.6 Robotics

The inverse robot kinematics problem is the problem of determining, for a given robot and a desired gripper position and orientation, the angles of the rotational joints and the translations of the prismatic joints that cause the gripper to assume the desired position and orientation. Mathematically, this problem reduces to the solution of systems of multivariate algebraic equations. The exact structure of these systems depends on the type of robot considered.

In the book of Paul,[16] analytic solutions of the algebraic systems for the inverse kinematics of the most important robot types are derived. Roughly, these solutions are derived by some skillful substitutions and transformations that are successful in the robot examples but, of course, cannot yield a general methodology for the analytic solutions of systems of algebraic equations. Using Paul's book, an engineer who wants to study the inverse kinematics problem for a specific robot has to look up the various types of algebraic systems shown in this book, choose an appropriate one, try to find appropriate substitutions that make the problem a special case of one solved in the book, and apply the steps of the transformations until an analytic solution formula for the robot is reached.

Hintenaus[17] has implemented the implied procedures in Paul's book as a symbolic computation program. For this program the only input needed is the traditional Denavit-Hartenberg representation of the robot. The program produces the system of kinematic equations, attempts to find substitutions by which the system can be handled as a special case of one treated by Paul, and applies the appropriate analytic solution method.

Symbolic computation has also been used in the simulation of the dynamics of a robot system. Hirschberg and Schramm[18] use the special-purpose program NEWEUL to generate symbolically the equations of motion, linearized with respect to a nominal motion. These equations are coded in

[16] R. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, MIT-Press, Cambridge, Massachusetts (1981).

[17] P. Hintenaus, An inverse kinematics system in MACSYMA, Tech. Report No. 87–18, Research Institute for Symbolic Computation, University of Linz, Austria (1987).

[18] W. Hirschberg and D. Schramm, Application of NEWEUL in robot dynamics, *J. of Symbolic Comp.* **7** (1989), pp. 199–204.

FORTRAN and are therefore ready for further processing in a simulation program.

3.7 Geometric Modeling

As R. N. Goldman[19] has noted:

> Geometric modeling is rapidly becoming an important tool in modern industrial design and manufacture. Computer models are replacing physical models. They are cheaper to construct, easier to change, and simpler to analyze. Computer simulations save both time and money, and computer analyses of geometric models lead to better and cheaper products. Stress analysis, interference checking, process planning, N.C. verification, mass properties calculations, and a host of other applications either are already being performed today or will be performed in the very near future directly from computer models. Architectural engineering, electrical engineering, industrial engineering, and mechanical engineering will all be revolutionized by this new technology.

In geometric modeling, the *implicitization problem* is the problem of finding an implicit algebraic equation for a surface or curve from a given parametric representation. This problem is important because problems for geometric objects may be of very different degrees of difficulty for different representations. For example, the decision about whether a given point is inside or outside a geometric object is easy for implicitly given objects but difficult for objects given by a parametric representation.

For many years, the implicitization problem has been deemed difficult if not algorithmically unsolvable in the general case. However, using the method of Gröbner bases and symbolic computation, it can be solved in total generality. Although the problem of computing Gröbner bases has a high intrinsic computational complexity, experience shows that parametric representations occurring in practical examples are well amenable by the preceding general method. This method has been proposed by Arnon and Sederberg.[20] The general method has been proved correct by Buchberger.[21]

---

[19] R. N. Goldman, The role of surfaces in solid modeling, in G. E. Farin (ed.), *Geometric Modeling: Algorithms and New Trends*, SIAM, Philadelphia (1987), pp. 69–90.

[20] D. Arnon and T. W. Sederberg, Implicit equations for a parametric surface by Gröbner basis, *Proc. 1984* MACSYMA *User's Conference*, Schenectady, New York, pp. 431–436.

[21] B. Buchberger, Applications of Gröbner bases in nonlinear computational geometry, *IMA Volumes in Mathematics and Its Applications* **14**, Springer-Verlag, New York (1988).

3.8 Mathematical Biology

The somatic cells of higher animals have a nucleus containing the DNA in a distinct compartment within the cytoplasm of the cell. Such cells are called eucaryotic cells. A model of a eucaryotic cell viewed as two compartments where reactions and diffusion occur according to the basic theory of feedback repression of Jacob and Monod is considered in a paper of Busenberg and Mahaffy.[22]

Previous models of this type considered the eucaryotic cell as two or more well-mixed compartments. Hence, a fundamental question is whether the bifurcations that help interpret the observed phenomenon of epigenic oscillations can be explained via the simpler, well-mixed compartment models when the diffusion rates in the cytoplasm are very high. This question was addressed by deriving a characteristic equation using a symbolic manipulation program and then doing detailed asymptotic analysis on this equation. The results confirmed the use of well-mixed compartment models when there are no significant reactions on the cell wall membrane and showed that when such reactions are important, well-mixed compartment models are not adequate approximations. These results cannot be proved by any numeric scheme, and the derivation of the characteristic equations was so involved that it could not be done by hand. Hence, the use of a symbolic system was necessary for this important aspect of the project.

3.9 Radar Design

Engineers design radar systems from a limited number of hardware elements that are combined to produce various systems that meet desired specifications. Although it is physically possible to construct and test each proposed system in a design, it is economically not feasible, and thus mathematical models are used to evaluate the effectiveness of a proposed design. A primary design criterion for a radar system is the precision with which it detects an object and its velocity. This can be determined analytically by studying the so-called ambiguity function of the system that measures the correlation between the received signal and an ideal response signal. Ideally, the design engineers would like to have an easy way of analyzing the effects that various parameters have on the ambiguity function of the system they are designing. For this type of work, it is best to have a fast way of generating analytical expressions for the ambiguity function in terms of the basic system parameters. The Pomona Division of General Dynamics Corporation sponsored a

[22]S. Busenberg and J. Mahaffy, Interaction of spatial diffusion and delays in models of genetic control by repression, *J. Math. Biol.* **22** (1985), pp. 313–333.

Mathematics Clinic project at Harvey Mudd College during the academic year 1984–1985 to develop an automatic computer method of generating the analytical expressions for the ambiguity functions of various radar systems. The project used a symbolic manipulation program to solve this problem. The results of this project are described in a report by di Franco et al.[23] This work required the use of a computer algebra program and could not be done by means of numeric computations.

3.10 Signal Processing and Coding

This topic concerns the processing of digital data for the purpose of transmission or storage, through a physical medium referred to as the channel. The channels are usually specified by their Fourier spectral characteristic obtained as the Fourier transform of an impulse response; hence, they are assumed to be linear (superposition principle holds). Furthermore, they are noisy; that is, the data are perturbed by a disturbance modeled as a random process. Signal processing amounts to matching the data to these channel characteristics. The mathematical techniques required involve the computation of orthogonal bases derived from the channel characteristics. Symbolic computation is a tool that can easily provide a variety of such families and can furthermore be used to solve the linear algebra problems involved.

Another channel characterization is given in terms of linear and nonlinear difference equations and is used to model "memory" or dependence on past signals. Because the matching of data to these channels involves the solution of recursions, continued fraction algorithms that are easily carried out using a symbolic computation system are used. The data must be stored or transmitted with a specific reliability. This requires algebraic redundancy or the use of error correcting codes. These are signal processors that work in finite field arithmetic. Computer algebra has been used to develop a variety of low-complexity discrete Fourier transforms and short convolution algorithms.[24] These algorithms are parts of many digital signal processing and fast Fourier transform packages and are embedded in silicon for digital signal processing chips. Their design requires solution of linear congruences

[23] R. di Franco et al., Radar design using symbolic manipulation software, Harvey Mudd College Mathematics Clinic report to General Dynamics, Pomona Division, June 1985, Harvey Mudd College, Claremont, California.

[24] S. Winograd, Algebraic constructions for algorithms, in *Proc. 1981 Symp. on Symbolic and Algebraic Computation*, ACM (1981), pp. 141–145; J. Cooley, Automated generation of optimized convolution algorithms, in D. V. Chudnovsky and R. D. Jenks (eds.), *Computer Algebra*, M. Dekker, New York (1989), pp. 183–198; L. Auslander, J. Cooley, and A. Silberger, On the use of SCRATCHPAD in the construction of convolution algorithms, ibid., pp. 151–182.

in a finite field. Padé approximations and continued fractions are the relevant tools. All these operations can be supplied by computer algebra and are used by researchers working in this area.

In this context algebraic geometry is useful for the construction of codes on algebraic curves. These constructions require machine models of curves, desingularization algorithms, divisor construction, and ultimately interpolation algorithms in fields of algebraic functions in one or more variables. Symbolic computation *is the tool* for these constructions. Algebraic curves are used in computer algebra systems to construct algebraico-geometric and Goppa codes[25] and to develop polynomial multiplication algorithms with low complexity.[26]

3.11 Control Theory

The problem of maneuvering flexible space structures by feedback control of certain elements of the stiffness matrix is examined in a paper by Moon and Rand.[27] The method is a mechanical analogue to nature's management of animal structural configuration, namely, active control of internal stress of muscles. In a man-made structure this can be done by applying a self-equilibrated internal stress state through the use of cables or hydraulic actuators. The tension in cables can be controlled by direct current servomotors and gear reducers. From elementary structural theory it is known that the initial stress state can change the elastic stiffness matrix. The authors propose to use feedback to control elements of the stiffness matrix by controlling the internal stress. The method leads to nonlinear control equations. Nonlinear analysis using center manifold theory and normal form theory determines criteria on the nonlinear control gains for stable or unstable operation. The analysis was made possible by the use of computer algebra.

3.12 Geostatistics

Kriging is an established tool in geostatistics used to estimate the average grade of an in-ground mineral. Kriging consists of computing the best linear

[25]M. Hassner, W. Burge, and S. Watt, Construction of algebraic ECC on the elliptic Riemann surface, SCRATCHPAD *Newsletter* **2** (1987), pp. 5–8.

[26]D. and G. Chudnovsky, Algebraic complexities and algebraic curves over finite fields, *Proc. Natl. Acad. Sci.* **84** (1987), pp. 1739–1743.

[27]F. C. Moon and R. H. Rand, Parametric stiffness control of flexible structures, in *Proc. of the Workshop on Identification and Control of Flexible Space Structures*, Vol II, pp. 329–342, Jet Propulsion Laboratory Publication 85–29, California Institute of Technology, Pasadena (1985).

unbiased estimator of a random function associated with the grade of the mineral over a specified block. Typically, the local estimation in a mining deposit can involve the Kriging of 10,000 blocks, each of them being estimated by using 10 to more than 20 neighboring data points. A model is chosen to best fit the experimental variogram obtained from the data. The number of models used for the variogram is limited (< 10), and the geometries of the blocks are often the same. Therefore, solving the Kriging system involves integrating the same functions on domains that differ only by their dimensions. This makes the finding of closed formulas for these integrals plausible and potentially useful.

Closed formulas have already been found for integrals of the Kriging system for the linear and spherical models where the block is a rectangle. The result for the spherical model can be found in a paper by Guibal.[28] These results have been used in BLUEPAK, a large geostatistical system. J. H. and J-P. Marbeau[29] derive closed formulas for these integrals using the spherical and linear models, where the block is chosen to be a parallelopiped with rectangular edges.

The traditional means of solving a Kriging system is by numerical integration. A comparison of the numeric approach and the closed form approach is dependent on the number of discretization points. For a small number of points, the numeric approach was faster. However, the closed form solution proved to be faster for larger numbers of points. As would be expected, the accuracy for the numeric approach improves as the number of points increases. The closed form approach is accurate for small numbers of points. Unfortunately, the closed formula approach involves matrix inversion, and consequently, as the number of points increases, there is a problem with insufficient memory to compute the necessary inverse matrix.

As noted in *Geostatistics*,[30] the specific applications of symbolic computation presented by Marbeau will not change immediately the practice of geostatistics. However, this work is considered to be an important pioneering effort into territory that will have a dramatic influence in the future.

---

[28] D. Guibal, Les fonctions auxiliaries a deux dimensions pour le schema spherique, Centre de Geostatistique, Fontainebleau, France (1973).

[29] J. H. Marbeau, *Towards Symbolic Kriging with the Help of* MACSYMA, M.S. Thesis, University of Denver, June 1987; J. H. Marbeau and J-P. E. Marbeau, Formal computation of integrals for 3-d kriging, *Geostatistics* **2** (1989), pp. 773–784.

[30] *Geostatistics* **1** (1987), p. 6, R. Barnes (ed.) North American Council on Geostatistics, Minneapolis.

3.13 Algebraic Geometry

A relationship between the geometry of a projective curve and the algebra of its defining ideal (the defining equations) has been studied by Eisenbud, Koh, and Stillman.[31] The defining equations are given by the two-by-two minors of a matrix of linear forms for projective curves satisfying suitable conditions. Their theorem extends a theorem of Castelnuovo.[32] It is a result in the spirit of classical algebraic geometry that was missed by the classical geometers. The new research was strongly influenced by use of the computer algebra system MACAULAY. Eisenbud, Koh, and Stillman were led to formulate and prove their main theorem after studying examples produced by MACAULAY.[33]

3.14 Number Theory

The remarkable identities left by S. Ramanujan[34] have been subject to extensive developments using computer algebra.[35] In particular, the mock theta functions are now much better understood,[36] and stunning interactions between classical additive number theory and algebraic number theory have been discovered.[37] In addition, a new nonalgebraic proof of Gauss' famous theorem that each integer is a sum of at most three triangular numbers has been found.[38] All these developments would have been impossible without the intuition gained from the construction of Bailey pairs and Bailey chains using computer algebra.[39]

Number theory has always been a prime target of applications of com-

[31]D. Eisenbud, J. Koh, and M. Stillman, Determinantal equations for curves of high degree, *Amer. J. Math.* (in press).

[32]G. Castelnuovo, Sui multipli di una serie lineare di gruppi di punti appartenente ad una curva algebrica, *Rend. Circ. Mat. Palermo* **7** (1893), pp. 89–110.

[33]D. Bayer and M. Stillman, The design of MACAULAY: a system for computing in algebraic geometry and commutative algebra, in *Proc. 1986 Symp. on Symbolic and Algebraic Computation*, ACM (1986), pp. 157–162.

[34]S. Ramanujan, *The Lost Notebook and Other Unpublished Papers*, Narosa Publishing House, New Delhi (1988).

[35]G. E. Andrews, The fifth and seventh order mock theta functions, *Trans. Amer. Math Soc.* **293** (1986), pp. 113–134.

[36]D. R. Hickerson, A proof of the mock theta conjectures, *Inventiones Math.* **94** (1988), pp. 639–660.

[37]G. E. Andrews, F. J. Dyson, and D. R. Hickerson, Partitions and indefinite quadratic forms, *Inventiones Math.* **91** (1988), pp. 391–407; H. Cohen, q-identities for Maass waveforms, *Inventiones Math.* **91** (1988), pp. 409–422.

[38]G. E. Andrews, $E\Upsilon PHKA!$ $num = \triangle + \triangle + \triangle$, *J. Number Th.* **23** (1986), pp. 285–293.

[39]Ibid.

puter algebra. A large volume of outstanding calculations probe the way to conjectures and, eventually, solutions to classical problems, for example, the pioneering Birch-Swinnerton-Dyer computations and their extensions.[40] Number-theoretic techniques developed in computer algebra have been applied to diophantine problems.[41] Solution of diophantine and differential equations is one target of computer algebra computations.[42] One example is the help of symbolic differential calculus in the solution of Kolchin's problem.[43]

3.15 Applied Mathematics

The problem considered in a paper by Dangelmayr and Guckenheimer[44] is the determination of phase portraits that occur in a specific four-parameter family of vector fields on the plane. Computer algebra is used to do two things. First, systems of algebraic equations that determine the parameters at which various types of bifurcations of equilibrium solutions occur are computed. Second, using an averaging argument that involves the calculation and reduction of elliptic integrals, the location of periodic orbits and their bifurcations is determined. The algebra, including the reduction of elliptic integrals, is routine but lengthy and tedious.

In a recent paper[45] Guckenheimer, Rand, and Schlomiuk report on a calculation that is related to one of Hilbert's famous unsolved problems, the 16th, about limit cycles for polynomial vector fields. It addresses the question of how many limit cycles can collapse simultaneously into a homoclinic orbit of a divergence free quadratic vector field. The calculation uses the same type of techniques as the Dangelmayr and Guckenheimer paper, but the crux of the argument depends on the symbolic computation of the integral of a rational function of degree 18. The result of this computation is a large expression, requiring thousands of bytes of storage, that would be impossible to compute by other means. More elliptic integral calculations

[40]B. Birch and H. P. S. Swinnerton-Dyer, Numerical tables on elliptic curves, *Lect. Notes Math.* **476** (1975), pp. 81–144.

[41]J. H. Davenport, Y. Siret, and E. Tournier, *Computer Algebra—Systems and Algorithms for Algebraic Computation*, Academic Press, London (1988).

[42]D. and G. Chudnovsky, Padé and rational approximations to systems of functions and their arithmetic applications, *Lect. Notes Math.* **1052** (1984), pp. 37–84.

[43]D. and G. Chudnovsky, The Wronskian formalism for linear differential equations and Padé approximations, *Advances in Math.* **52** (1984), pp. 111–138.

[44]G. Dangelmayr and J. Guckenheimer, On a four parameter family of planar vector fields, *Arch. Rat. Mech. Anal.* **97** (1987), pp. 321–352.

[45]J. Guckenheimer, R. H. Rand, and D. Schlomiuk, Degenerate homoclinic cycles in perturbations of quadratic Hamiltonian systems, Cornell University MSI Technical Report No. '87–45.

appear in another paper by Armbruster, Guckenheimer, and Holmes[46] for a bifurcation problem with a higher dimensional phase space. The calculations could have been performed by hand, but computer algebra was used and helped speed the process.

A paper by Guckenheimer and Johnson[47] gives a systematic approach to locally measuring the deviation of iterates of a one-dimensional mapping from a quadratic or a linear function. The arguments involve a complicated construction of a sequence of "box maps" and estimates on their distortion and relative sizes. Throughout the argument there are calculations involving compositions of quadratic functions and fractional linear transformations. Both the details of the final proof and the development of its strategy were greatly aided by the systematic use of computer algebra.

Computer algebra has played a significant role in extending the work on exactly solved models in statistical mechanics. Building on Baxter's original solution of the hard hexagon model,[48] Andrews, Baxter, and Forrester[49] produced an infinite family of solved models. The main mathematical technique was based on empirical methods implemented in computer algebra. Subsequently, computer algebra played an even more powerful role in extending these models.[50]

In another area, F. A. Grünbaum and his students at Berkeley have worked on determining explicit solutions of the Kadomtsev-Petviashvili equation and related "soliton" completely integrable nonlinear partial differential equations of mathematical physics. The period from 1965 to 1975 was an important one in the area of nonlinear partial differential equations. First, a number of equations like the Korteweg-deVries, the nonlinear Schroedinger, and the Toda lattice were found to be completely integrable by using inverse scattering techniques. Second, some new equations, most notably the KP equation (after the Soviet physicists Kadomtsev and Petviashvili), were proposed as a model for plasma waves and shallow water wave propagation in two dimensions. Then the KP equation was found to be a "universal equation" resulting from a delicate balance of nonlinearity, dispersion, and some weak transversal dependence. Third, a nonlinear superposition principle for "cnoidal waves" was uncovered in the Soviet Union. This work brought to

[46]D. Armbruster, J. Guckenheimer, and P. J. Holmes, Heteroclinic and modulated travelling waves in systems with O(2) symmetry, *Physica* **D29** (1988), pp. 257–282.

[47]J. Guckenheimer and S. Johnson, Distortion of S-unimodal maps, authors' preprint.

[48]R. J. Baxter, Hard hexagons: exact solutions, *J. Phys. A.* **13** (1980), pp. L61–L70.

[49]G. E. Andrews, R. J. Baxter, and P. J. Forrester, Eight-vertex SOS model and generalized Rogers-Ramanujan type identities, *J. Statist. Phys.* **35** (1984), pp. 193–266.

[50]G. E. Andrews and R. J. Baxter, Lattice gas generalization of the hard hexagon model. III. q-trinomial coefficients, *J. Statist. Phys.* **47** (1987), pp. 297–330; G. E. Andrews and R. J. Baxter, SCRATCHPAD explorations for elliptic theta functions, in *Proc. of Computers and Mathematics Conf.* 1986, (in press).

the fore the theory of compact Riemann surfaces, making it possible to exhibit some explicit solutions in terms of the Riemann theta function of such an algebraic curve. These were the only ones known before 1987. Fourth, by exploiting formal pseudodifferential operators and previous work of Hirota in Japan, the Kyoto group of M. Sato showed that the KP hierarchy of equations is the way to understand scores of nonlinear partial differential equations that can be put into Lax-Zakharov-Shabat form. This led to further connections, in particular with Kac-Moody Lie algebras. Fifth, in an attempt to produce new classes of explicit solutions of the KP equation, S. Novikov and I. Krichever in the Soviet Union brought in the powerful tools of algebraic geometry in the form of vector bundles over algebraic curves. These solutions could reveal new phenomena and, like the "pure N soliton solutions," bring new life into this field. However, their methods have never been implemented, and no new solutions have emerged out of this rather "high brow" approach.

Recently, Grünbaum embarked on an effort to reinterpret the program of Novikov and Krichever in terms of "down to earth analysis" and a use of symbolic computation. Grünbaum has now produced the first new solutions corresponding to "rank two bundles" over a curve of genus one. This much already makes clear the power of the new approach.

3.16 Other Surveys

More extensive surveys on applications of symbolic computation can be found elsewhere. Calmet and van Hulzen[51] discuss applications in biology, chemistry, physics, mathematics, and computer science. They also refer to several other survey papers. MacCallum[52] gives a more specialized review of the use of symbolic computation in relativity. However, the number of applications of symbolic computation appears to be increasing substantially so that survey papers published as recently as the early 1980s are likely to be out of date. An indication of this is the fact that the 1986 version of the *Bibliography of Publications Referencing* MACSYMA lists more than 600 publications over a wide range of scientific disciplines, but even this extensive listing is incomplete. A recent book[53] contains papers on applications in chaotic systems, fluid dynamics, nonlinear control systems, and robotics.

[51] J. Calmet and J. A. van Hulzen, Computer algebra applications, in B. Buchberger, G. E. Collins, and R. Loos (eds.), *Computer Algebra—Symbolic and Algebraic Computation*, 2nd ed. Springer-Verlag, Vienna (1982), pp. 245–258.

[52] M. A. H. MacCallum, op. cit.

[53] R. Grossman (ed.), Symbolic Computation: Applications to Scientific Computing, *Frontiers in Applied Mathematics* **5**, Soc. for Industrial and Applied Mathematics, Philadelphia (1989).

Chapter 4

Future Directions

Speculation about future directions usually proceeds from relatively safe extrapolations from the recent past or from more risky leaps from various other less reliable trends and possibilities. Predicting the future in symbolic computation is especially difficult because its future will be shaped by advances in computer science, computing technology, mathematics, and a growing commercial market. Computing is a fast-changing field that has helped to reshape the face of science and engineering in the last half of this century. Mathematics is a mature and distinguished discipline with a history of unanticipated advances of great practical importance. The commercial sector is aggressively enlarging the market for symbolic computation products, bringing them to more scientists, engineers, and students. A richer mixture of these dynamics can lead to an environment explosive with possibilities. The sketches of applications in Chapter 3 indicate the breadth of possibilities. In this section selective scenarios of the future are described.

There are reasons to be concerned about the future of this field, as well as reasons to be optimistic. In the United States, applications are still benefiting from research and development that is often fifteen years old and from times when federal support was stimulating a small but growing and productive research community. On the other hand, every NeXT computer will come with a copy of MATHEMATICA. In addition, MATHEMATICA will be marketed by Ardent Computer Corporation, IBM, Silicon Graphics, Stellar Computer, Inc., and other companies. This may indicate that computer companies are beginning to take symbolic computation software seriously. Furthermore, there is an awakening in the scientific community to the potential of this form of computation that bodes well for the future.

4.1 Computing Technology

Advances are being made in computer hardware that hold the potential for a considerable impact on symbolic computation. By the mid-1990s every serious scientist should have access to a personal workstation with a processor capable of 20 to 50 million instructions per second, 16 to 32 megabytes of memory, at least 1 gigabyte of random access secondary storage, and a connection to high-resolution graphics and printing capabilities. Future symbolic computation systems must take into account these hardware trends.

Symbolic computation systems as such will expand into scientific and engineering computational environments that incorporate symbolic capabilities. Numeric, graphic, and text processing will be important components of the new environments. These components will interact in novel ways, for example, like the notebook concept in MATHEMATICA that combines text and "live" code that can be executed to illustrate and extend the text. The software design will further exploit the natural structures and abstractions found in the underlying mathematical domains as is exemplified by the recent research[1] using object-oriented styles of programming. This will be a part of software design that puts more stress on modularity and reusability. These general mathematical computation systems will become increasingly important day-to-day tools of scientists, engineers, and students.

To solve especially difficult problems, special-purpose systems to support various applications research will increase in importance, often using supercomputers, with the emphasis being on the science or engineering that was made possible and not on the tool itself. The rate of development of special-purpose systems will be substantially affected by success, or lack thereof, in developing more modular, reusable software.

New algorithm design will be driven by new architecture and system designs. If custom chip design and manufacture become as easy as making photographic copies, special chips for important kernel components of symbolic computation may be justified. There will be a continuing need for more cross-fertilization between those using symbolic computation and those designing the tools.

Just as every researcher will have powerful personal machines by the mid-1990s, every high school student will have access to a personal computer with at least one megabyte of memory, 100 megabytes of secondary storage, and a

[1] S. K. Abdali, G. W. Cherry, and N. Soiffer, An object oriented approach to algebra system design, in *Proc. 1986 Symp. on Symbolic and Algebraic Computation*, ACM (1986), pp. 24–30; J. Foderaro, The Design of a Language for Algebraic Computation Systems, Ph.D. Thesis, University of California–Berkeley, 1983; R. D. Jenks, A primer: 11 keys to new SCRATCHPAD, *Proc. EUROSAM '84. Lect. Notes Comp. Sci.* **174** (1984), pp. 123–147.

bit-mapped display. This presents a huge potential for symbolic computation as a part of a larger computer-aided instruction environment.

4.2 Nonlinearity

Recently, much progress has been made in handling nonlinear problems, problems that are often of major importance in a wide range of applications. As the David report[2] pointed out,

> In chemistry and biology ... reaction-diffusion mechanism study ... has involved the nonlinear generation of wave patterns, pulses, and shock fronts which are phenomenologically new and require new modes of analysis. In geophysical-physical sciences, analytical approximation to atmospheric, oceanic, and elastic wave motions has produced new interpretations with which to forecast weather and predict earthquakes.
>
> In all these fields, considerable interest focuses on the new, nonlinear phenomena associated with strong force and energy interactions, discrete-continuous interactions, or the more subtle low-energy nonlinearities of the biological world, phenomena which will dominate much of the mathematics of science from now on. We already see this in the fascination with solitons, chaos, and bifurcation and singularity theories.

Computing has been an important tool in studies involving nonlinearity. In turn, the study of nonlinear problems strains even the largest of current computers and the best of current algorithms.

Recently, important progress, based on the work of Buchberger and Collins with contributions by many others, has been made on new algorithms for dealing with nonlinear algebraic problems. This progress has been brought about by a partnership between symbolic computation and algebraic geometry. The work on nonlinearity in symbolic computation has been of a different nature than that discussed in the David report, but this fledgling effort has already brought important gains and seems destined to be an important research area in symbolic computation as well as other areas for the foreseeable future.

[2] *Renewing U.S. Mathematics—Critical Resource for the Future*, Report of the Ad Hoc Committee on Resources for the Mathematical Sciences, E. E. David, Jr., Chairman, National Academy Press, Washington (1984).

4.3 Breaking the Deterministic Complexity Barriers

A central problem in all computing, and especially in symbolic computation, is that as the size of the computational problem grows the computational resources, that is, time and space, needed to solve the problem often grow much faster. For some fundamental problems it has been determined that any possible deterministic algorithm will require time and/or space that is exponential in the size of the problem. The implication of these deterministic complexity results is that the solution of problems above a certain size will never be possible with conventional computational methods. The computational complexity barriers clearly delineate the difficulties in solving certain problems by computational methods, indicate paths of research that are unlikely to be successful, and help guide research on overcoming the difficulties.

One of the most promising developments for breaking through these deterministic complexity barriers has been the astonishing discovery that computations that rely on chance often can be far more effective than following any possible predetermined algorithm. The classic example is the Monte Carlo method for numerical integration. More recently, randomness has been shown to be a powerful tool for algebraic problems.

One of the main probabilistic results to date in symbolic computation is an algorithm for showing that a multivariate polynomial presented by a computation (that is, a straight-line program or even a black box) is nonzero. This result, given by Schwartz,[3] has a host of applications. Among them are the effective versions of the Hilbert Irreducibility Theorem[4] that are crucial for the fast sparse polynomial factorization algorithms, randomizations in algebra such as testing a Tutte matrix for a matching,[5] solving a sparse linear system over finite fields,[6] and work on parallelizing the Smith normal form construction.[7] This is an area in which results have just begun to

[3]J. T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, *JACM* **27** (1980), pp. 701–717.

[4]J. Heintz and M. Sieveking, Absolute primality of polynomials is decidable in random polynomial-time in the number of variables, *Proc. ICALP '81, Lect. Notes Comput. Sci.* **115** (1981), pp. 16–28; J. von zur Gathen, Irreducibility of multivariate polynomials, *J. Comput. Syst. Sci.* **31** (1985), pp. 225–264; E. Kaltofen, Effective Hilbert irreducibility, *Inform. Control* **66** (1985), pp. 123–137.

[5]L. Lovász, On determinants, matchings, and random algorithms, in L. Budach (ed.), *Fundamentals of Computing Theory*, Akademia Verlag, Berlin (1979).

[6]D. Wiedmann, Solving sparse linear equations over finite fields, *IEEE Trans. Inform. Theory* **IT-32** (1986), pp. 54–62.

[7]E. Kaltofen, M. S. Krishnamoorthy, and B. D. Saunders, Fast parallel computation of Hermite and Smith forms of polynomial matrices, *SIAM J. Alg. Discrete Math.* **8** (1987),

appear. Many important new algorithms using randomization are likely in the future.

Another method for skirting the complexity barriers is to identify important subproblems that, by avoiding the full generality of a given problem, become amenable to more effective solutions. This has been done, for example, by Melenk, Möller, and Neun[8] in their work on large stationary chemical kinetics problems. They modified the Buchberger algorithm for computing Gröbner bases of nonlinear algebraic equations to exploit the special structure of equations derived from the nonlinear ordinary differential equations for reaction systems. Consequently, they were able to handle larger systems of equations.

pp. 683–690.

[8]M. Melenk, H. M. Möller, and W. Neun, On Gröbner bases computation on a supercomputer using REDUCE, Preprint SC 88–2 (Jan. 1988), Konrad-Zuse-Zentrum für Informationstechnik–Berlin.

Chapter 5

Findings and Recommendations

To realize the potential contributions of symbolic computation, individuals, academia, industry, and government must work together. We must continue the dialogue as we search for solutions and answers to the challenges set forth in this report.

5.1 Findings

Symbolic computation is a field of accomplishment, a field of promise, and a field of contrasts. It faces educational, technological, research, and communications challenges that arise from its diversity, richness, wide applicability, and immaturity. We have demonstrated that symbolic computation has wide applicability and substantially under-utilized potential. The field is at a turning point of possibilities brought forth by improvements in computer hardware, new algorithms, and new software. But there are still important impediments. Better algorithms are needed, especially for applications. The study and use of symbolic computation is not ubiquitous; there are too few centers of activity. Better user interfaces are needed. The separation between symbolic and numeric computation is too large. A central challenge is to increase the number of users, which will bring symbolic computation more into the main stream of science and engineering. Many of the problems of this field would be ameliorated by an order of magnitude increase in the number of users of symbolic computation. More users would create an increased pool of scientists and engineers knowledgable about symbolic computation. This would help with the human resource problems and their feedback would serve as a forcing function for more action on all fronts. This is depicted schematically in Figure 5.1.

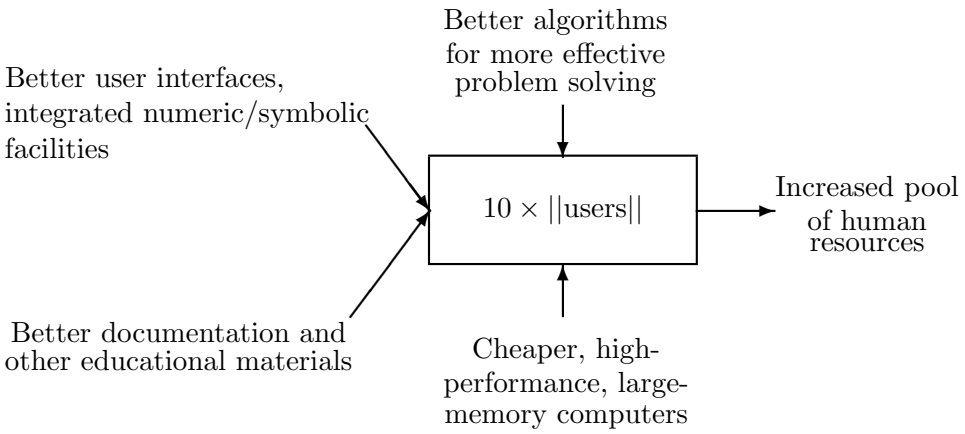

Figure 5.1: **A Key Issue**

The prerequisites for more users are:

- Better software platforms that include the key ingredients of improved user interfaces and well integrated symbolic and numeric capabilities
- Better documentation and other educational materials for users
- More effective methods and algorithms for solving important scientific and engineering problems
- The increased availability of cheap, high-performance, large-memory computers that are capable of serving as adequate hardware platforms for symbolic computation systems

Other key findings are:

- Symbolic computation is a part of a key technology, namely, scientific and engineering computation, that is becoming increasingly important to science, technology, and society
- Symbolic computation enhances scientific and engineering productivity
- Symbolic computation has made important progress in developing software and in discovering new algorithms since the mid-1960s

- The opportunities for substantial progress in symbolic computation are significant

There are many aspects to this multi-faceted field.

- Mathematics and computer science are the main fields that contribute to basic research. Applications occur throughout science and engineering. The interplay between fundamental results and technology is an important aspect of the field.

- A broad range of mathematics is relevant to symbolic computation research and vice-versa. To date algebra, algebraic geometry, logic, group theory,number theory, combinatorics, complex variables and analysis, among others, have played important roles in fundamental research on symbolic computation. Conversely, symbolic computation is a potential research tool for all areas of mathematics.

- In computer science complex data Structures, object-oriented programming, and other advanced programming tools are important. User interfaces are particularly important. Both heuristic and algorithmic methods are necessary for successful applications. There are important interactions with scientific text processing, graphics, and numerical computing.

Symbolic computation software reflects many of the successes and problems of the field.

- Symbolic computation software is typically large, sophisticated, and error prone. All the problems associated with the design and engineering of large software systems are present here.

- Implementation of new results is lagging; many new algorithms that have been discovered over the past decade have been implemented in few, if any, systems.

- Software development is lagging behind new hardware technology, especially in the use of new display hardware and in the use of new architectures, including supercomputers.

- More modular, reusable, high-quality, library-style software needs to be developed. The lack of such publicly available software inhibits researchers from building on the work of others and impedes the development of special systems for applications.

The theory and algorithms base has been substantially improved in recent years, but there is much still to be done.

- The algorithm base contains gaping holes in fundamental areas such as symbolic linear algebra, symbolic approximations, and complex variables.
- Little research has been done on parallel symbolic algorithms.
- More research is needed on large scale and special purpose algorithms.

Educational matters are particularly crucial at this time.

- Nonspecialists have little knowledge about the capabilities and limitations of symbolic computation.
- The pool of human resources for research in symbolic computation is small.
- Educating new researchers and attracting new users are keys to the future development of the field of symbolic computation. In the United States, education in this area is almost nonexistent. Currently, established curricula do not exist for graduate students who wish to work in this area. At most American institutions with graduate mathematics and computer science programs, graduate students have no opportunity to study this area, no faculty are doing research in the area, and few courses are directly relevant.

5.2 Recommendations

The most important recommendations are concerned with increasing the number of applications and users of symbolic computation. Other recommendations will help to build a critical mass of researchers in this area. Improvements to the software platform are also needed to accelerate and to speed applications. These recommendations should be given priority.

Many of the recommendations are couched in terms of funding initiatives, but much can be accomplished by the academic and industrial sectors independent of any new funding. Universities can immediately begin to teach more in this area and in other ways begin to focus on improving the flow of information about symbolic computation. Industry can continue its fledgling steps to take software needs in this area seriously. However, to make substantial progress will require serious government action. Government funding for upstream research developments, university efforts in research and in increasing the human resource pool, and both coupled with commercial efficiencies and focus on downstream implementation should be an effective outline for progress.

New initiatives are needed by the funding agencies to increase significantly the levels of research on and applications of symbolic computation. Reacting to the normal flow of proposals is likely to be insufficient since a critical mass has not been obtained in this field. The following actions are recommended.

Priority Recommendations

To the Funding Agencies

- Substantially improve the software platform for research and applications by:
  - Funding research on high-quality, reusable user interfaces
  - Funding software acquisition, development, and maintenance needed to capitalize the instrumentation for symbolic computation research and applications
  - Funding establishment of high-quality libraries of symbolic algorithms and methods
  - Funding research on interface protocols between various software packages and systems
  - Encouraging joint university and industry research
  - Supporting summer sessions and special years to support tool building and experimental aspects of the field
- Stimulate developments at the interface of symbolic and numeric computation by:
  - Funding research in defining the interface and on algorithms that employ both symbolic and numeric methods
  - Funding course development that incorporates symbolic and numeric computing
  - Funding workshops to attack a particular problem using symbolic and numeric methods
- Improve the basic mathematics and algorithms underlying symbolic computation by:
  - Accelerating research on symbolic methods that are especially relevant to applications such as approximation methods and methods for solving ordinary and partial differential equations

  - Funding further research on algorithms for fundamental computations in areas like symbolic linear algebra, nonlinear algebra, and complex variables
  - Funding research on symbolic algorithms to take advantage of new computer architectures, including supercomputers
  - Encouraging more research on ways to deal with complexity problems in symbolic computations such as the use of probabilistic algorithms

- Address education for users and new researchers by:

  - Funding research on incorporating symbolic computation in mathematics, science, and engineering curricula
  - Supporting the development of educational materials on symbolic computation

To the Universities

- Provide adequate computing facilities for symbolic computation, including appropriate software.

- Include more material relevant to symbolic computation in university curricula. This can be done in a variety of ways:

  - Introduce symbolic computation as a tool into existing courses, especially ones covering aspects of applied mathematics.
  - Introduce new courses that contain material on the various aspects of symbolic computation, including applications.
  - Put more emphasis on constructive techniques in current mathematics courses, especially courses in algebra, algebraic geometry, and number theory.
  - Use more examples from symbolic computation in computer science courses dealing with software engineering, graphics, and algorithms. An increased emphasis on ideas from category theory and universal algebra in software design would be particularly applicable to symbolic computation software.
  - Develop special education and research programs in which symbolic computation plays a key role. A curriculum that combines mathematics and computation provides a good fundamental education for future scientists and engineers. One such curriculum is given in Appendix B.

To the Professional Community

- Produce more and better educational materials. A lack of textbooks is an especially pressing problem. These needs have been cited on several occasions and improvements are slowly occurring, but continued efforts are still needed.

Additional Recommendations

Additional important recommendations are:

To the Professional Community

- Increase efforts to improve and increase interactions among the various components of the scientific computation community. More interactions are desirable between mathematicians and computer scientists interested in symbolic computation, between researchers in numerical and symbolic computing, and between software builders and users.
- Resolve the conflict over the commercial requirements to protect products and the academic requirements to exchange information freely. A solution to this problem may be found through the use of proprietary software kernels with publicly available source code built on top. For this model to serve science successfully, the mathematical algorithms must be in the public domain. Several producers of symbolic software have already adopted this model.

To the Commercial Sector

- Continue the initial steps to take symbolic computation software seriously.

To the Funding Agencies

Address the interdisciplinary nature of the field by:

- Establishing mini-centers for symbolic and algebraic computation
  - that span science and engineering with a users of point of view
  - that cut-across computer science and mathematics from a system development point of view
- Funding workshops and special years devoted to symbolic computation and its applications

- Supporting summer session programs with a regional or national scope on the use and development of symbolic computation systems

Mini-centers

Why are SAC mini-centers needed? By its very nature symbolic and algebraic computation is interdisciplinary and is best done by teams of researchers representing a variety of expertise. In the United States a relatively high proportion of symbolic computation research takes place in industry since, among other reasons, this is where there is a critical mass of investigators. In academia, symbolic computation researchers tend to be isolated in number and segregated by departments. The natural home for this type of research would be somewhere in mathematics, computer science, and engineering. Since it does not fit easily into the usual university structure, the difficulties are further compounded.

The demographics of researchers contribute in many ways to the problems in symbolic computation in the United States. The lack of critical mass at most universities makes the scenario of a cadre of researchers with diverse backgrounds impossible. In addition, this lack of concentration contributes to the difficulties in educating and training people in symbolic computation. It is difficult to have a graduate program or many undergraduate courses based on the interests of one or two people.

Many of the problems that have been identified could be addressed by the formation of research groups in symbolic and algebraic computation. They will be referred to as SAC centers. A SAC center would be composed of at least five researchers representing several aspects of symbolic computation. This would provide the necessary critical mass. In addition, there would be an active visitor program both to supplement the basic research staff and to disseminate information. These centers would also have the responsibility for education on many levels as well as serve as a repository for information on symbolic computation.

An active visitor program serves several purposes. Visitors from industry would provide a link with the activities and needs in this sector. Often this information is not readily accessible since publication in industry is not required to the extent that it is in academia. Visitors from academia and industry serve the usual purpose of dissemination of information and increased opportunity for joint research. At the present stage of research on symbolic computation in the United States, researchers tend to be isolated. An active visitor program would provide contact for isolated researchers, helping to maintain their interest and activity. There should also be some provision for visitors who wish to learn about what exists and what can be done with symbolic computation.

SAC centers could act as central consultants and repositories for information about symbolic computation systems. These services would be effectively advertised. In addition, they could offer seminars and short courses on symbolic and algebraic computation.

SAC centers would be active in symbolic and algebraic computation education on many levels. They would provide undergraduate, graduate, and research-level courses and seminars. They would develop symbolic computation courses to be taught at other institutions and conduct workshops to teach educators from both high school and college.

The preceding goals are ambitious and yet appear to be attainable if the SAC centers are properly organized. The consensus at the workshop was that several smaller centers would be more effective than a single monolithic center. They would be more accessible to the community and would provide a better atmosphere for interaction.

Appendix A

Workshop Participants and Observers

A group of scientists representing a broad range of interests was invited to the workshop as participants. Programs officers from a variety of government funding agencies were invited to attend and participate as they saw fit. They are listed in the following as observers.

A.1 Participants

Anthony C. Hearn, RAND Corporation, Workshop Chair
B. F. Caviness, Coordinator, National Science Foundation and University of Delaware[1]

Subpanel on Applications

André Deprit, National Bureau of Standards,[2] Chair
John Fitch, Department of Computer Science, University of Bath, United Kingdom
Gerald Guralnik, Department of Physics, Brown University
Michael Levine, Pittsburgh Supercomputer Center and Physics Department, Carnegie-Mellon University
James McIver, Department of Chemistry, SUNY Buffalo
Anil Nerode, Mathematical Sciences Institute, Cornell University

Subpanel on Software and Systems Design

Richard Jenks, IBM Watson Research Center, Chair

[1] Given affiliations were the ones in effect at the time of the workshop.

[2] The name of the NBS has been changed to the National Institute of Standards and Technology.

Richard Fateman, Division of Computer Science, University of California–Berkeley
Gaston Gonnet, Computer Science Department, University of Waterloo
Alan Perlis, Department of Computer Science, Yale University

Subpanel on Algorithms and Theory

Moss Sweedler, Department of Mathematics, Cornell University, Chair
Shreeram Abhyankar, Departments of Mathematics and Industrial Engineering, Purdue University
Bruno Buchberger, Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria
David Chudnovsky, Department of Mathematics, Columbia University
William F. Schelter, Department of Mathematics, University of Texas
Peter Weinberger, AT&T Bell Laboratories

Subpanel on Symbolic/Numeric Interfaces

Morven Gentleman, Division of Electrical Engineering, National Research Council of Canada, Chair
Richard Askey, Department of Mathematics, University of Wisconsin
John Rice, Department of Computer Science, Purdue University
Phil Smith, IMSL, Houston
Paul Wang, Department of Mathematical Sciences, Kent State University

A steering committee consisting of B. F. Caviness, André Deprit, Morven Gentleman, Anthony C. Hearn, Richard Jenks, Anil Nerode, and Moss Sweedler was in charge of the organization of the workshop. The steering committee was expertly assisted on administrative matters by Wilson Kone from the Mathematical Sciences Institute at Cornell University.

A.2 Observers

William Adams, Program Director, Algebra and Number Theory, NSF
Peter Bates, Program Director, Applied Mathematics, NSF
Ann Boyle, Program Director, Algebra and Number Theory, NSF

Charles Brownstein, Acting Assistant Director, Directorate of Computer and Information Sciences and Engineering, NSF

Jagdish Chandra, Director, Mathematical Sciences Division, Army Research Office

Raymond Chin, Program Director, Computational Mathematics, NSF

Melvyn Ciment, Acting Division Director, Advanced Scientific Computing, NSF

David Elliott, Program Director, Systems Theory and Operations Research, NSF

Peter Freeman, Division Director, Computer and Computation Research, NSF

Harry Hedges, Program Director, Institutional Infrastructure Program, NSF

Richard Hirsh, Program Director, New Technologies/Computational Science and Engineering, NSF

William Lakin, Program Director, Applied Mathematics, NSF

George Lea, Program Director, Computational Engineering, NSF

Nathaniel Macon, Program Director, Software Systems, NSF

Andrzej Manitius, Deputy Division Director, Mathematical Sciences, NSF

Louise Raphael, Program Director, Course and Curriculum Program, NSF

John V. Ryff, Program Director, Classical Analysis, NSF

Judith Sunley, Division Director, Mathematical Sciences, NSF

Alvin Thaler, Program Director, CISE Instrumentation, NSF

Ralph Wachter, Program Director, Software Research, Office of Naval Research

Elbert Walker, Program Director, Special Projects in Mathematical Sciences, NSF

Yechezkel Zalcstein, Program Director, Computer Systems Architecture, NSF

Appendix B

A Sample Curriculum for Education in Symbolic Computation

The following discussion describes the way that one institution has set up an extensive educational program in symbolic computation. RISC-LINZ (Research Institute for Symbolic Computation at the Johannes Kepler University in Linz, Austria, Europe) is an institute that specializes in research, graduate education, and industrial cooperation in symbolic computation.

RISC is embedded into the two departments of mathematics and computer science, which together have a faculty of about 40 professors (13 of them full professors). However, RISC-LINZ organizationally is independent of these two departments. In the spring of 1988, seven faculty members were exclusively working for RISC-LINZ. In the fall of 1988, this number increased to nine (one full professor, a second full professorship for RISC-LINZ has been created and a search for suitable candidates started).

At RISC-LINZ, symbolic computation is understood in the broad sense defined in the *Journal of Symbolic Computation*; that is, it encompasses algorithmic problem solving in all symbolic object domains. Notably, it encompasses computational logic, computer algebra, computer analysis, computer geometry, and computer-aided programming. Also, all three aspects of symbolic computation, namely, theoretical foundations of symbolic algorithms, symbolic computation software, and applications, are pursued at RISC-LINZ.

Research topics at RISC-LINZ are determined by and vary with the faculty. At present, there is a definite emphasis in algebraic and geometric algorithms.

RISC-LINZ graduate education is organized as a special curriculum in

symbolic computation. Roughly, it includes 30 courses (each 2 credit hours) that cover the whole range of symbolic computation and its aspects.

- Preparatory courses
  - Preparation I (training in proving and algorithmic problem solving)
  - Preparation II (introduction to scientific work in symbolic computation)
  - Preparation III (survey on symbolic computation)
- Core courses
  - Computational Logic I (resolution theorem proving)
  - Computational Logic II (special techniques in resolution theorem proving)
  - Computational Logic III (decision algorithms for special logical theories)
  - Computer Algebra I (algorithms in basic algebraic domains)
  - Computer Algebra II (advanced topics, for example, algorithmic polynomial ideal theory)
  - Computer Analysis (analytic integration and related topics)
  - Computational Geometry I (combinatorial geometrical algorithms)
  - Computational Geometry II (algebraic algorithms in geometry)
  - Computational Geometry III (geometric modeling)
  - Computer-Aided Program Verification
  - Computer-Aided Program Synthesis
- Foundation courses
  - Logic I (syntax and semantics of predicate logic)
  - Logic II (model theory of predicate logic)
  - Logic III (limitations of the algorithmization of logics)
  - Algorithm Theory I (design and analysis of algorithms)
  - Algorithm Theory II (fundamental concepts of computability)
  - Algorithm Theory III (complexity classes)
  - Semantics I (lambda-calculus and related systems)

  - Semantics II (universal algebra and abstract data types)
  - Semantics III (methods for describing semantics)

- Software courses

  - SC Software I (functional programming, LISP)
  - SC Software II (logical programming, PROLOG)
  - SC Software III (computer algebra systems)
  - SC Software IV (systems for theorem proving and automatic programming)

- Application courses

  - Expert Systems
  - CAD/CAM Systems
  - Robot Programming (kinematics and dynamics)
  - Others (according to the special interests of the permanent and visiting faculty)

- Special courses and seminars

  Chosen according to the special interests of faculty and visiting faculty

Before participating in the courses of the special symbolic computation curriculum, students having a bachelor's degree in mathematics are supposed to pass certain core courses of the computer science curriculum (assembler, compilers, several high-level programming languages, operating systems, software technology). Conversely, students having a bachelor's degree in computer science are supposed to pass certain courses of the mathematics curriculum (numerical mathematics, algebra). It is important to note that both students of computer science and students of mathematics are admitted to the RISC-LINZ special curriculum of symbolic computation and that the explicit goal of this curriculum is to produce students who master the mathematical and computer science aspects of symbolic computation equally well.

Both the diploma (that is, master's degree) and the Ph.D. degree can be obtained in the frame of the RISC-LINZ special curriculum. Diploma and Ph.D. theses typically are written in the frame of research projects. Some of them are motivated by problems arising from cooperation with industrial companies. Typically, also, theses consist of a mathematical part and a computer science part.

After the bachelor's degree, usually three years are needed for the diploma degree and another three years for the Ph.D. The sequence of courses in the special curriculum is repeated every three semesters (1 1/2 years) such that students can enter the curriculum at virtually any time. In the spring of 1988, nine students were working on Ph.D. theses, ten students were working on a diploma thesis, and another 15 to 20 students were doing course work in preparation for the diploma thesis.

Appendix C

Textbooks and Other Instructional Materials for Symbolic Computation

A. G. Akritas, *Elements of Computer Algebra with Applications*, John Wiley & Sons, New York (1989).

M. D. Atkinson, Computational group theory, *Proceedings of the London Mathematical Society Symposium on Computational Group Theory*, Academic Press, San Diego (1984).

B. Buchberger, G. E. Collins, and R. G. K. Loos (eds.), *Computer Algebra—Symbolic and Algebraic Computation*, 2nd ed., Springer-Verlag, Vienna (1983).

B. W. Char, K. O. Geddes, G. H. Gonnet, and S. M. Watt, First leaves: a tutorial introduction to MAPLE, in MAPLE *User's Guide*, WATCOM Publications Ltd., Waterloo, Ontario (1985).

J. H. Davenport, Y. Siret, and E. Tournier, *Computer Algebra—Systems and Algorithms for Algebraic Computation*, Academic Press, San Diego (1988).

J. C. Howard, *Practical Applications of Symbolic Computations*, IPC Science and Technology Press, Guildford, England (1979).

D. E. Knuth, *The Art of Computer Programming, Vol. 2: Semi-Numerical Algorithms*, 2nd ed., Addison-Wesley, Reading, Massachusetts (1981).

J. D. Lipson, *Algebra and Algebraic Computing*, Addison-Wesley, Reading, Massachusetts (1981).

MACSYMA *User's Guide*, Symbolics, Inc., Cambridge, Massachusetts (1987).

M. Mignotte, *Mathématiques pour le calcul formel*, Presses Universitaires de France, Paris (1989).

R. Pavelle (ed.), *Applications of Computer Algebra*, Kluwer Academic Publishers, Boston (1985).

R. H. Rand, *Computer Algebra in Applied Mathematics: An Introduction to* MACSYMA, Pitman Publishing Inc., Marshfield, Massachusetts (1984).

R. H. Rand and D. Armbruster, Perturbation methods, bifurcation theory and computer algebra, *Applied Mathematical Sciences* **65**, Springer-Verlag, New York (1987).

G. Rayna, REDUCE—*Software for Algebraic Computation*, Springer-Verlag, New York (1987).

J. R. Rice, Mathematical aspects of scientific software, *IMA Volumes in Mathematics and Its Applications* **14**, Springer-Verlag, New York (1988).

C. C. Sims, *Abstract Algebra: A Computational Approach*, John Wiley & Sons, New York (1984).

D. Stauffer, F. W. Hehl, V. Winkelmann, and J. G. Zabolitzky, *Computer Simulation and Computer Algebra—Lectures for Beginners*, Springer-Verlag, New York (1988).

S. Wolfram, MATHEMATICA—*A System for Doing Mathematics by Computer*, Addison-Wesley, Redwood City, California (1988).

C. Wooff and D. Hodgkinson, muMATH*: A Microcomputer Algebra System*, Academic Press, San Diego (1987).

H.G. Zimmer, Computational problems, methods, and results in algebraic number theory, *Lect. Notes Math.* **268**, Springer-Verlag, New York (1972).

Appendix D

The Scholarly Community

This appendix presents an overview of some professional activities in symbolic computation.

D.1 Professional Societies

Since the late 1960s many research activities in symbolic computation have been coordinated by ACM SIGSAM (the Association for Computing Machinery Special Interest Group on Symbolic and Algebraic Manipulation). SIGSAM publishes the quarterly *SIGSAM Bulletin* and in 1988 had approximately 1300 members. In Europe there are a number of national professional groups whose main interest is symbolic computation. These groups have been loosely coordinated by the umbrella organization SAME (Symbolic and Algebraic Manipulation in Europe). In Japan, the Research Institute for Mathematical Sciences at Kyoto University has organized a series of meetings on symbolic computation, and in the Soviet Union, the Joint Institute for Nuclear Research in Dubna has also organized a series of such meetings.

These meetings have evolved into the annual ISSAC (International Symposium on Symbolic and Algebraic Computation) meeting. Recent meetings in this series have attracted 150 to 200 attendees in North America and approximately 400 when held in Europe. The 1988 meeting was held in Rome; the 1989 meeting in Portland, Oregon, and the 1990 meeting in Japan.

In the United States, SIAM (the Society for Industrial and Applied Mathematics) has had a series of special sessions on this subject at its regularly scheduled meetings, as have other societies like AMS (American Mathematical Society), AAAS (American Association for the Advancement of Science), ACS (American Chemical Society), and APS (American Physical Society). The computational group theory community has organized several meetings

devoted to their particular interests. Richard Jenks and David Chudnovsky have organized a highly successful series of meetings entitled *Computers and Mathematics* that have brought together a broad spectrum of persons. Symbolic computation has been the unifying theme.

D.2 Publications

The proceedings of the various international meetings on symbolic computation have been primarily published by Springer-Verlag in the series entitled *Lecture Notes in Computer Science* and by ACM. The primary journal in symbolic computation is the *Journal of Symbolic Computation*, which was started in 1985. It has been quite successful in its short existence, expanding from its initial four issues a year to six in 1987 and to twelve issues in 1989. Other important outlets for papers in this area are the SIAM *Journal on Computing* and the ACM *Transactions on Mathematical Software*. Many other journals have also published papers in this area.

D.3 Places with Educational, Research, or Software Development Activities in Symbolic Computation

The following is a partial list of organizations that have significant research, educational, or software development activities in symbolic computation:[1]

```
Academia Sinica, Beijing, China
Australian National University
Cambridge University, UK
Columbia University
Concordia University, Canada
Cornell University
Franz, Inc., Berkeley, California
General Electric Corporate Research and Development Center
Gesellschaft fuer Mathematik und Datenverarbeitung, Bonn, FRG
IBM T. J. Watson Research Center
The Institute of Physical and Chemical Research, Saitama, Japan
Johannes Kepler University, Austria
Joint Institute for Nuclear Research, Dubna, USSR
Kent State University
Massachusetts Institute of Technology
```

[1]This list has been compiled by an informal poll.

Moscow State University, Moscow, USSR
National Bureau of Standards
New York University
North Carolina State University
Ohio State University
Polish Academy of Sciences, Warsaw
Purdue University
Queen Mary College, University of London
The RAND Corporation
Rensselaer Polytechnic Institute
Rutgers University
The Soft Warehouse, Honolulu, Hawaii
Southern Methodist University
State University of New York---Albany
Steklov Institute of Mathematics, Leningrad, USSR
Swiss Federal Institute of Technology---Zurich
Symbolics, Inc., Cambridge, Massachusetts
Technische Hochschule Aachen, FRG
Tektronix Computer Research Laboratory
Twente University of Technology, Enschede, The Netherlands
Universitaet des Saarlandes, FRG
University of Bath, UK
University of California at Berkeley
University of Delaware
University of Denver
University of Duesseldorf, FRG
University of Essen, FRG
University of Genova, Italy
University of Grenoble, France
University of Hawaii
University of Illinois, Champaign-Urbana
University of Illinois, Chicago
University of Karlsruhe, FRG
University of Leipzig, GDR
University of New Mexico
University of Paris VI, France
University of Saskatchewan, Canada
University of St. Andrews, Scotland
University of Stockholm, Sweden
University of Strasbourg, France
University of Sydney, Australia

University of Texas---Austin
University of Toronto
University of Tuebingen, FRG
University of Warwick, UK
University of Waterloo, Canada
Wolfram Research Inc., Champaign, Illinois
Xerox Palo Alto Research Center