

Preface

Computer algebra is the field of mathematics and computer science that is concerned with the development, implementation, and application of algorithms that manipulate and analyze mathematical expressions. This book and the companion text, *Computer Algebra and Symbolic Computation: Mathematical Methods*, are an introduction to the subject that addresses both its practical and theoretical aspects. This book, which addresses the practical side, is concerned with the formulation of algorithms that solve symbolic mathematical problems, and with the implementation of these algorithms in terms of the operations and control structures available in computer algebra programming languages. *Mathematical Methods*, which addresses more theoretical issues, is concerned with the basic mathematical and algorithmic concepts that are the foundation of the subject. Both books serve as a bridge between texts and manuals that show how to use computer algebra software and graduate level texts that describe algorithms at the forefront of the field.

These books have been in various stages of development for over 15 years. They are based on the class notes for a two-quarter course sequence in computer algebra that has been offered at the University of Denver every other year for the past 16 years. The first course, which is the basis for *Elementary Algorithms*, attracts primarily undergraduate students and a few graduate students from mathematics, computer science, and engineering. The second course, which is the basis for *Mathematical Methods*, attracts primarily graduate students in both mathematics and computer science. The course is cross-listed under both mathematics and computer science.

Prerequisites

The target audience for these books includes students and professionals from mathematics, computer science, and other technical fields who would like to know about computer algebra and its applications.

In the spirit of an introductory text, we have tried to minimize the prerequisites. The mathematical prerequisites include the usual two year freshman–sophomore sequence of courses (calculus through multivariable calculus, elementary linear algebra, and applied ordinary differential equations). In addition, an introductory course in discrete mathematics is recommended because mathematical induction is used as a proof technique throughout. Topics from elementary number theory and abstract algebra are introduced as needed.

On the computer science side, we assume that the reader has had some experience with a computer programming language such as Fortran, Pascal, C, C++, or Java. Although these languages are not used in these books, the skills in problem solving and algorithm development obtained in a beginning programming course are essential. One programming technique that is especially important in computer algebra is recursion. Although many students will have seen recursion in a conventional programming course, the topic is described in Chapter 5 of *Elementary Algorithms* from a computer algebra perspective.

Realistically speaking, while these prerequisites suffice in a formal sense for both books, in a practical sense there are some sections as the texts progress where greater mathematical and computational sophistication is required. Although the mathematical development in these sections can be challenging for students with the minimum prerequisites, the algorithms are accessible, and these sections provide a transition to more advanced treatments of the subject.

Organization and Content

Broadly speaking, these books are intended to serve two (complementary) purposes:

- *To provide a systematic approach to the algorithmic formulation and implementation of mathematical operations in a computer algebra programming language.*

Algorithmic methods in traditional mathematics are usually not presented with the precision found in numerical mathematics or conventional computer programming. For example, the algorithm for the expansion of products and powers of polynomials is usually given informally instead of with (recursive) procedures that can be expressed as a computer program.

The material in *Elementary Algorithms* is concerned with the algorithmic formulation of solutions to elementary symbolic mathematical problems. The viewpoint is that mathematical expressions, represented as expression trees, are the data objects of computer algebra programs, and by using a few primitive operations that analyze and construct expressions, we can implement many elementary operations from algebra, trigonometry, calculus, and differential equations. For example, algorithms are given for the analysis and manipulation of polynomials and rational expressions, the manipulation of exponential and trigonometric functions, differentiation, elementary integration, and the solution of first order differential equations. Most of the material in this book is not found in either mathematics textbooks or in other, more advanced computer algebra textbooks.

- *To describe some of the mathematical concepts and algorithmic techniques utilized by modern computer algebra software.*

For the past 35 years, the research in computer algebra has been concerned with the development of effective and efficient algorithms for many mathematical operations including polynomial greatest common divisor (gcd) computation, polynomial factorization, polynomial decomposition, the solution of systems of linear equations and multivariate polynomial equations, indefinite integration, and the solution of differential equations. Although algorithms for some of these problems have been known since the nineteenth century, for efficiency reasons they are not suitable as general purpose algorithms for computer algebra software. The classical algorithms are important, however, because they are much simpler and provide a context to motivate the basic algebraic ideas and the need for more efficient approaches.

The material in *Mathematical Methods* is an introduction to the mathematical techniques and algorithmic methods of computer algebra. Although the material in this book is more difficult and requires greater mathematical sophistication, the approach and selection of topics is designed so that it is accessible and interesting to the intended audience. Algorithms are given for basic integer and rational number operations, automatic (or default) simplification of algebraic expressions, greatest common divisor calculation for single and multivariate polynomials, resultant computation, polynomial decomposition, polynomial simplification with Gröbner bases, and polynomial factorization.

Topic Selection

The author of an introductory text about a rapidly changing field is faced with a difficult decision about which topics and algorithms to include in the work. This decision is constrained by the background of the audience, the mathematical difficulty of the material and, of course, by space limitations. In addition, we believe that an introductory text should really be an introduction to the subject that describes some of the important issues in the field but should not try to be comprehensive or include all refinements of a particular topic or algorithm. This viewpoint has guided the selection of topics, choice of algorithms, and level of mathematical rigor.

For example, polynomial gcd computation is an important topic in *Mathematical Methods* that plays an essential role in modern computer algebra software. We describe classical Euclidean algorithms for both single and multivariate polynomials with rational number coefficients and a Euclidean algorithm for single variable polynomials with simple algebraic number coefficients. It is well known, however, that for efficiency reasons, these algorithms are not suitable as general purpose algorithms in a computer algebra system. For this reason, we describe the more advanced subresultant gcd algorithm for multivariate polynomials but omit the mathematical justification, which is quite involved and far outside the scope and spirit of these books.

One topic that is not discussed is the asymptotic complexity of the time and space requirements of algorithms. Complexity analysis for computer algebra, which is often quite involved, uses techniques from algorithm analysis, probability theory, discrete mathematics, the theory of computation, and other areas that are well beyond the background of the intended audience. Of course, it is impossible to ignore efficiency considerations entirely and, when appropriate, we indicate (usually by example) some of the issues that arise. A course based on *Mathematical Methods* is an ideal prerequisite for a graduate level course that includes the complexity analysis of algorithms along with recent developments in the field¹.

Chapter Summaries

A more detailed description of the material covered in these books is given in the following chapter summaries.

¹A graduate level course could be based on one of the following books: Akritas [2], Geddes, Czapor, and Labahn [39], Mignotte [66], Mignotte and Ștefănescu [67], Mishra [68], von zur Gathen and Gerhard [96], Winkler [101], Yap [105], or Zippel [108].

Elementary Algorithms

Chapter 1: Introduction to Computer Algebra. This chapter is an introduction to the field of computer algebra. It illustrates both the possibilities and limitations for computer symbolic computation through dialogues with a number of commercial computer algebra systems.

Chapter 2: Elementary Concepts of Computer Algebra. This chapter introduces an algorithmic language called *mathematical pseudo-language* (or simply MPL) that is used throughout the books to describe the concepts, examples, and algorithms of computer algebra. MPL is a simple language that can be easily translated into the structures and operations available in modern computer algebra languages. This chapter also includes a general description of the evaluation process in computer algebra software (including automatic simplification), and a case study which includes an MPL program that obtains the change of form of quadratic expressions under rotation of coordinates.

Chapter 3: Recursive Structure of Mathematical Expressions. This chapter is concerned with the internal tree structure of mathematical expressions. Both the conventional structure (before evaluation) and the simplified structure (after evaluation and automatic simplification) are described. The structure of automatically simplified expressions is important because all algorithms assume that the input data is in this form. Four primitive MPL operators (*Kind*, *Operand*, *Number_of_operands*, and *Construct*) that analyze and construct mathematical expressions are introduced. The chapter also includes a description of four MPL operators (*Free_of*, *Substitute*, *Sequential_substitute*, and *Concurrent_substitute*) which depend only on the tree structure of an expression.

Chapter 4: Elementary Mathematical Algorithms. In this chapter we describe the basic programming structures in MPL and use these structures to describe a number of elementary algorithms. The chapter includes a case study which describes an algorithm that solves a class of first order ordinary differential equations using the separation of variables technique and the method of exact equations with integrating factors.

Chapter 5: Recursive Algorithms. This chapter describes recursion as a programming technique in computer algebra and gives a number of examples that illustrate its advantages and limitations. It includes a case study that describes an elementary integration algorithm which finds the antiderivatives for a limited class of functions using the linear properties of the integral and the substitution method. Extensions of the algorithm to include the elementary rational function integration, some trigonometric integrals, elementary integration by parts, and one algebraic function form are described in the exercises.

Chapter 6: Structure of Polynomials and Rational Expressions. This chapter is concerned with the algorithms that analyze and manipulate polynomials and rational expressions. It includes computational definitions for various classes of polynomials and rational expressions that are based on the internal tree structure of expressions. Algorithms based on the primitive operations introduced in Chapter 3 are given for degree and coefficient computation, coefficient collection, expansion, and rationalization of algebraic expressions.

Chapter 7: Exponential and Trigonometric Transformations. This chapter is concerned with algorithms that manipulate exponential and trigonometric functions. It includes algorithms for exponential expansion and reduction, trigonometric expansion and reduction, and a simplification algorithm that can verify a large class of trigonometric identities.

Mathematical Methods

Chapter 1: Background Concepts. This chapter is a summary of the background material from *Elementary Algorithms* that provides a framework for the mathematical and computational discussions in the book. It includes a description of the mathematical pseudo-language (MPL), a brief discussion of the tree structure and polynomial structure of algebraic expressions, and a summary of the basic mathematical operators that appear in our algorithms.

Chapter 2: Integers, Rational Numbers, and Fields. This chapter is concerned with the numerical objects that arise in computer algebra, including integers, rational numbers, and algebraic numbers. It includes Euclid's algorithm for the greatest common divisor of two integers, the extended Euclidean algorithm, the Chinese remainder algorithm, and a simplification algorithm that transforms an involved arithmetic expression with integers and fractions to a rational number in standard form. In addition, it introduces the concept of a field which describes in a general way the properties of number systems that arise in computer algebra.

Chapter 3: Automatic Simplification. Automatic simplification is defined as the collection of algebraic and trigonometric simplification transformations that are applied to an expression as part of the evaluation process. In this chapter we take an in-depth look at the algebraic component of this process, give a precise definition of an automatically simplified expression, and describe an (involved) algorithm that transforms mathematical expressions to automatically simplified form. Although automatic simplification is essential for the operation of computer algebra software, this is the only detailed treatment of the topic in the textbook literature.

Chapter 4: Single Variable Polynomials. This chapter is concerned with algorithms for single variable polynomials with coefficients in a field. All algorithms in this chapter are ultimately based on polynomial division. It includes algorithms for polynomial division and expansion, Euclid's algorithm for greatest common divisor computation, the extended Euclidean algorithm, and a polynomial version of the Chinese remainder algorithm. In addition, the basic polynomial division and gcd algorithms are used to give algorithms for numerical computations in elementary algebraic number fields. These algorithms are then used to develop division and gcd algorithms for polynomials with algebraic number coefficients. The chapter concludes with an algorithm for partial fraction expansion that is based on the extended Euclidean algorithm.

Chapter 5: Polynomial Decomposition. Polynomial decomposition is a process that determines if a polynomial can be represented as a composition of lower degree polynomials. In this chapter we discuss some theoretical aspects of the decomposition problem and give an algorithm based on polynomial factorization that either finds a decomposition or determines that no decomposition exists.

Chapter 6: Multivariate Polynomials. This chapter generalizes the division and gcd algorithms to multivariate polynomials with coefficients in an integral domain. It includes algorithms for three polynomial division operations (recursive division, monomial-based division, and pseudo-division); polynomial expansion (including an application to the algebraic substitution problem); and the primitive and subresultant algorithms for gcd computation.

Chapter 7: The Resultant. This chapter introduces the resultant of two polynomials, which is defined as the determinant of a matrix whose entries depend on the coefficients of the polynomials. We describe a Euclidean algorithm and a subresultant algorithm for resultant computation and use the resultant to find polynomial relations for explicit algebraic numbers.

Chapter 8: Polynomial Simplification with Side Relations. This chapter includes an introduction to Gröbner basis computation with an application to the polynomial simplification problem. To simplify the presentation, we assume that polynomials have rational number coefficients and use the lexicographical ordering scheme for monomials.

Chapter 9: Polynomial Factorization. The goal of this chapter is the description of a basic version of a modern factorization algorithm for single variable polynomials in $\mathbf{Q}[x]$. It includes square-free factorization algorithms in $\mathbf{Q}[x]$ and $\mathbf{Z}_p[x]$, Kronecker's classical factorization algorithm for $\mathbf{Z}[x]$, Berlekamp's algorithm for factorization in $\mathbf{Z}_p[x]$, and a basic version of the Hensel lifting algorithm.

Computer Algebra Software and Programs

We use a procedure style of programming that corresponds most closely to the programming structures and style of the Maple, Mathematica, and MuPAD systems and, to a lesser degree, to the Macsyma and Reduce systems. In addition, some algorithms are described by transformation rules that translate to the pattern matching languages in the Mathematica and Maple systems. Unfortunately, the programming style used here does not translate easily to the structures in the Axiom system.

The dialogues and algorithms in these books have been implemented in the Maple 7.0, Mathematica 4.1, and MuPAD Pro (Version 2.0) systems. The dialogues and programs are found on a CD included with the books. In each book, available dialogues and programs are indicated by the word “Implementation” followed by a system name Maple, Mathematica, or MuPAD. System dialogues are in a notebook format (mws in Maple, nb in Mathematica, and mnb in MuPAD), and procedures are in text (ASCII) format (for examples, see the dialogue in Figure 1.1 on page 3 and the procedure in Figure 4.15 on page 148). In some examples, the dialogue display of a computer algebra system given in the text has been modified so that it fits on the printed page.

Electronic Version of the Book

These books have been processed in the $\text{\LaTeX} 2_{\epsilon}$ system with the *hyperref* package, which allows hypertext links to chapter numbers, section numbers, displayed (and numbered) formulas, theorems, examples, figures, footnotes, exercises, the table of contents, the index, the bibliography, and web sites. An electronic version of the book (as well as additional reference files) in the portable document format (PDF), which is displayed with the Adobe Acrobat software, is included on the CD.

Acknowledgements

I am grateful to the many students and colleagues who read and helped debug preliminary versions of this book. Their advice, encouragement, suggestions, criticisms, and corrections have greatly improved the style and structure of the book. Thanks to Norman Bleistein, Andrew Burt, Alex Champion, the late Jack Cohen, Robert Coombe, George Donovan, Bill Dorn, Richard Fateman, Clayton Ferner, Carl Gibbons, Herb Greenberg, Jillane Hutchings, Lan Lin, Zhiming Li, Gouping Liu, John Magruder, Jocelyn Marbeau, Stanly Steinberg, Joyce Stivers, Sandhya Vinjamuri, and Diane Wagner.

I am grateful to Gwen Diaz and Alex Champion for their help with the L^AT_EX document preparation; Britta Wienand, who read most of the text and translated many of the programs to the MuPAD language; Aditya Nagrath, who created some of the figures; and Michael Wester who translated many of the programs to the Mathematica, MuPAD, and Macsyma languages. Thanks to Camie Bates, who read the entire manuscript and made numerous suggestions that improved the exposition and notation, and helped clarify confusing sections of the book. Her careful reading discovered numerous typographical, grammatical, and mathematical errors.

I also acknowledge the long-term support and encouragement of my home institution, the University of Denver. During the writing of the book, I was awarded two sabbatical leaves to develop this material.

Special thanks to my family for encouragement and support: my late parents Elbert and Judith Cohen, Daniel Cohen, Fannye Cohen, and Louis and Elizabeth Oberdorfer.

Finally, I would like to thank my wife, Kathryn, who as long as she can remember, has lived through draft after draft of this book, and who with patience, love, and support has helped make this book possible.

Joel S. Cohen
Denver, Colorado
September, 2001

