Abstract Algebra
A Computational Approach

Charles C. Sims

Abstract Algebra
A Computational Approach

Charles C. Sims
Rutgers University

JOHN WILEY & SONS
ew York Chichester Brisbane Toronto Singapore

Library of Congress Cataloging in Publication Data

Sims, Charles C.
Abstract algebra.

Includes index.
1. Algebra, Abstract--Data processing. I. Title.
QA162.354 1984 512'.02'028542 83-6715
ISBN 0-471-09846-9

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

PREFACE

This book is intended as a text for a one-year introductory course in abstract algebra in which algorithmic questions and computation are stressed. A significant amount of computer usage by students is anticipated. My decision to write the book grew out of my interest in group-theoretic algorithms and my observation that learning the definitions, the theorems, and even the proofs of algebra too often fails to equip students adequately to solve computational algebraic problems. The goals of the book are to:

1 Introduce students to the basic concepts of algebra and to elementary results about them.
2 Present the concept of an algorithm and to discuss certain fundamental algebraic algorithms.
3 Show how computers can be used to solve algebraic problems and to provide a library, *CLASSLIB*, of computer programs with which students can investigate interesting computational questions in algebra.
4 Describe the APL computer language to the extent needed to achieve the other goals.

To help meet these goals, two additional manuals have been prepared, an instructor's manual and a *CLASSLIB* user's manual. There is more material here than can be covered in a one-year course. The instructor's manual contains suggested course outlines, hints on how to use this book, and the answers to selected exercises, including all that involve APL. The *CLASSLIB* user's manual contains detailed information about the library with complete listings of all programs. Generally, students should not need to acquire the user's manual. However, anyone wishing to make extensive use of *CLASSLIB* will probably want to have a copy. Both manuals are available from the publisher. The library can be obtained in machine-readable form from the author.

Computation in algebra is not really new. In some areas, such as number theory, there is a tradition of hand calculation going back hundreds of years. However, the development of the digital computer has inspired new interest in the subject. More and more research effort is being devoted to the

existence and efficiency of algebraic algorithms. In addition, the use of computers to solve problems in algebra is growing steadily. Many algebraic algorithms can be understood by students in an introductory course.

The choice of the computer language to be used was very important. Of the languages normally available at college computer centers, only one is really suited for use in the teaching of algebra to students with little or no prior computing experience. That language is APL. The superiority of APL stems as much from the way the language is implemented as it does from the nature of the language itself. Here are some of the features of APL that make it the natural choice for this book.

1 APL is implemented in an interactive mode.
2 Arrays exist independent of programs.
3 One-line statements can be entered and executed immediately. In effect, beginning students do not have to write programs in the traditional sense.
4 The language contains many powerful primitive operations for manipulating arrays that are very useful in describing algebraic algorithms.

Even with the power of the APL language, most of the algorithms discussed are too complicated to be coded efficiently by beginning programmers. Therefore I decided to provide a library of programs that would allow students to use the algorithms on nontrivial problems while developing their skill in using APL. Students should not need to acquire a separate APL text; the appendices provide an adequate introduction to the language.

At this point, it would be good to mention several things that the book is not. It is not a text in applied algebra, which emphasizes the use of algebraic techniques to solve problems that arise outside of mathematics. Neither is this a book on numerical linear algebra, which deals with the numerical analysis aspects of linear algebra over the real and complex fields. Although the difficulties of performing computer calculations with real and complex numbers are discussed, the emphasis is on exact computation. For this reason, many of the computational exercises in linear algebra involve the fields $GF(p)$, p a prime.

There are two reasons for recommending that this book be used for a one-year course. First, the time required to introduce students to APL is too great to leave sufficient time to cover a reasonable amount of algebra in a one-semester course. In order to be able to understand and reproduce the dialogues in the text, one needs to know the material in Sections 1 to 3 and 5 to 7 of Appendix 1 and Sections 2 and 3 of Appendix 2, as well as certain topics discussed in Sections A1.4 and A2.1. Approximately three weeks are necessary to cover this material, and even more time must be

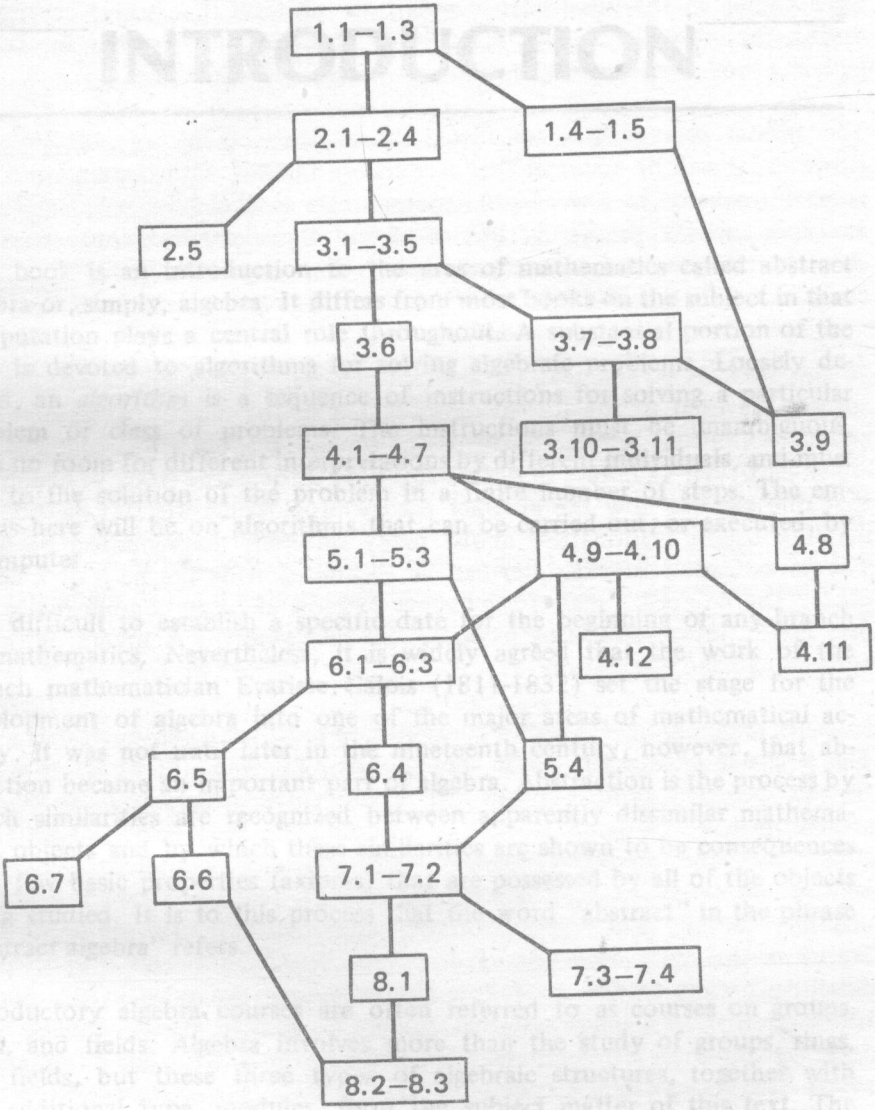
spent if significant original programming is to be required of students. The second reason for suggesting a one-year course is that the most interesting algorithms, at least to me, come in the second half of the course. All of the chapters contain computational topics, but it was my desire to describe the material in Sections 6.5, 7.4, and 8.2, which provided the main motivation for this book. There are several additional topics that I would have liked to include. Some, such as factorization in $\mathbf{Z}[X]$ and a study of the ideals in $\mathbf{Z}[X]$, were omitted for lack of space. Others, such as some of the recent developments in computational group theory, could not be included because they involve concepts that do not fit easily into an introductory algebra course. Galois theory has been left out because practical algorithms for computing Galois groups are too involved to be presented at this level.

In the text, the lemmas, theorems, and corollaries are numbered consecutively within a section. Theorem 3 of Section 4 of Chapter 6 is referred to as Theorem 4.3 in other sections of Chapter 6 and as Theorem 6.4.3 outside of Chapter 6. A similar numbering system is used for examples and for exercises. Names occurring in brackets are references to the bibliography. In the contents, sections marked with an asterisk may be omitted without affecting the logical development. Exercises of greater than average difficulty are also flagged with asterisks. The ends of proofs are marked with the symbol □. A □ at the end of a theorem indicates that the proof of that theorem will be omitted.

Much of the writing of this book was done in college libraries. I am indebted to the library staffs at Rutgers University, Princeton University, Monmouth College, Southern Methodist University, and the Australian National University for the facilities they made available. Many individuals provided assistance throughout the eight years during which this book took shape. I wish to thank James England, Eugene Klotz, and Don Orth for many useful conversations concerning the use of APL in expostion. Michael O'Nan and Hale Trotter provided assistance on various mathematical topics. In particular, some of the material in Section 7.4 is based on a talk by Trotter. Certainly thanks are due to Kenneth Iverson. Without his development of APL, this book, at least in its present form, would not have been possible. Finally, I wish to record my deep gratitude to my wife, Annette, who typed an early version of this text and then typed the entire manuscript into a homegrown word processor. Her assistance made the preparation of the book much easier than it would otherwise have been.

Charles C. Sims

SIGNIFICANT DEPENDENCIES BETWEEN SECTIONS



CONTENTS

INTRODUCTION

This book is an introduction to the area of mathematics called abstract algebra or, simply, algebra. It differs from most books on the subject in that computation plays a central role throughout. A substantial portion of the text is devoted to algorithms for solving algebraic problems. Loosely defined, an *algorithm* is a sequence of instructions for solving a particular problem or class of problems. The instructions must be unambiguous, with no room for different interpretations by different individuals, and must lead to the solution of the problem in a finite number of steps. The emphasis here will be on algorithms that can be carried out, or executed, by a computer.

It is difficult to establish a specific date for the beginning of any branch of mathematics. Nevertheless, it is widely agreed that the work of the French mathematician Evariste Galois (1811-1832) set the stage for the development of algebra into one of the major areas of mathematical activity. It was not until later in the nineteenth century, however, that abstraction became an important part of algebra. Abstraction is the process by which similarities are recognized between apparently dissimilar mathematical objects and by which these similarities are shown to be consequences of a few basic properties (axioms) that are possessed by all of the objects being studied. It is to this process that the word "abstract" in the phrase "abstract algebra" refers.

Introductory algebra courses are often referred to as courses on groups, rings, and fields. Algebra involves more than the study of groups, rings, and fields, but these three types of algebraic structures, together with one additional type, modules, form the subject matter of this text. The term "group" was coined by Galois, but the first formal definition was not given until 1849 and the value of the concept of an "abstract group" was not recognized for nearly 30 years more. The idea of a field is present in Galois' work, but the term was introduced by the German mathematician

Richard Dedekind (1831-1916) and the definition was not standardized until late in the nineteenth century. Although many examples of rings were known in the nineteenth century, the abstract theory was developed during the present century. The term "ring" was formulated by David Hilbert (1862-1943), a very important German mathematician.

The formal prerequisites for the study of abstract algebra are minimal. However, it will be assumed that readers are familiar with certain concepts normally covered in lower-level undergraduate mathematics courses. These concepts include proofs by induction and the elementary properties of sets, the integers, and rational numbers. Some acquaintance with real and complex numbers will also be assumed.

In this text, as in any text on abstract algebra, a considerable amount of space is devoted to the formal development of the subject. As axioms are stated, definitions made, and theorems proved, readers are encouraged to study particular examples in detail. It is only through the study of examples that one can see how the abstract theory provides an efficient method of deriving useful information about many different mathematical objects. The investigation of examples can be facilitated with the help of a computer. The computer makes it possible to look at more complicated and, one hopes, more interesting examples by removing the drudgery of time-consuming hand calculation.

In order to communicate with a computer we must use a computer language. The language chosen for use in this book is APL. The APL language is extremely powerful, which means that complicated calculations can be described with a few symbols. APL also possesses a high degree of internal consistency and, in many ways, is more logical than traditional mathematical notation. No prior knowledge of APL will be assumed. The appendices contain a description of the aspects of APL that are important for using this text. Appendix 1 describes the APL language in sufficient detail to permit readers to follow the computer examples in this book. However, it is very important that readers be able to work out these and other examples on the computer. Appendix 2 contains further information about APL systems that can help readers use their local systems efficiently.

In order to make the best possible use of this book, readers should have access to two APL workspaces that have been specifically created to supplement the text. The workspace *CLASSLIB* contains procedures for carrying out many types of algebraic computations. The workspace *EXAMPLES*

contains arrays that represent various kinds of algebraic objects. The arrays in *EXAMPLES* are used in the computational examples in the text. A more complete description of the contents of these two workspaces can be found in Appendix 3. It is suggested that the naming conventions described in Section A3.1 be read before making extensive use of *CLASSLIB*. All of the computer examples in the text assume that the contents of both *CLASSLIB* and *EXAMPLES* are present in the active workspace.

The algorithms used in most of the procedures in *CLASSLIB* are discussed in the text. As each procedure is introduced, readers should concentrate on learning what it does and on understanding the basic algorithm involved. Once some familiarity with a procedure has been achieved, it is an extremely valuable exercise to write one's own version of the procedure and compare it to the version in the library. I would appreciate being informed about possible improvements to the procedures in *CLASSLIB*.

Readers having some experience with APL may proceed immediately to Chapter 1. Those not familiar with APL should begin by reading through the first seven sections of Appendix 1. It is not necessary to become an expert in the use of APL before starting to learn algebra using this book. Once the fundamentals of the language have been grasped, the study of the real subject matter—abstract algebra—should be begun. The appendices can then be used for reference, as needed.

We will be using two different systems of symbolic notation, APL and traditional, and it is important to be able to recognize which system is being used in a particular expression. APL expressions are printed in a special type font used only for APL. Thus `C←A ZGCD B` is an APL expression. When any other type font is used, as in the statement "let c = gcd(a, b)", traditional mathematical notation is assumed. Some care is required to distinguish between the two commas that occur in this book. The ordinary comma (,) is a mark of punctuation, but the APL comma (,) represents one of two APL operations that are described in Section A1.5. Occasionally, we will borrow certain aspects of APL notation for use with traditional notation. For example, we will sometimes denote the entry in the ith row and jth column of the matrix A by $A[i;j]$, even though A is not an APL array. These borrowings should not present any serious problems.

1
SETS

For nearly 100 years the formal exposition of mathematics has been based on the concept of a set. Readers are no doubt familiar with sets from previous courses in mathematics. In this chapter we will summarize the basic definitions, notation, and operations of set theory. We will also discuss ways of representing sets by APL arrays and techniques for manipulating these arrays to perform set-theoretic operations. The APL index origin, described in Section A1.2, is normally assumed to be 1.

1. SETS

One of the most important ideas in the development of mathematics is the use of the axiomatic method, in which all of the theorems in a particular branch of mathematics are obtained as logical consequences of a few axioms that state the basic properties that are assumed to hold for the objects under study. This approach is probably most familiar in the area of plane geometry. In an axiomatic treatment of plane geometry, no attempt is made to say what points and lines really are. Instead, one writes down axioms such as "through any two distinct points there passes eaactly one line" in an attempt to formalize our intuitive notions about the kinds of pictures we can draw using a straightedge and a very sharp pencil. The axiomatic method is universally agreed to be the proper approach to the study of abstract algebra.

The idea of a set has been found to be of fundamental importance not only in algebra but in most of present-day mathematics. All of the algebraic objects we will study will be sets. It would seem reasonable, therefore, to begin our study of algebra with an axiomatic treatment of sets. This approach seems even more essential when we learn, as we will at the end of this section, that our intuition concerning sets can lead us to logical contradictions. However, we will follow the accepted practice in introductory algebra texts and omit a formal treatment of sets. There are two reasons for this. First, the exposition of axiomatic set theory would delay too long our study of the main subject matter of this book: the basic properties

of algebraic systems such as groups, rings, and fields and the algorithms for solving problems related to them. Second, our intuition leads us astray only when we try to consider sets that are "too big". The sets we will encounter in our study of algebra will be "small enough" that our intuition can be trusted not to get us into trouble.

We define a *set* to be any collection of objects called the *elements* or *members* of the set. A commonly used synonym for "set" is "family". The word "group" should not be used as a synonym for "set" because a group in mathematical terminology is a particular kind of algebraic object that we will study in Chapter 3. If x is an element of the set X, we write $x \in X$ and say that x *belongs to* X or that x *is in* X. If x is not an element of X, we write $x \notin X$. Two sets are equal if they have the same elements. Thus the statement $X = Y$ is equivalent to the following pair of assertions:

1. If $x \in X$, then $x \in Y$.
2. If $y \in Y$, then $y \in X$.

To show that X and Y are not equal, we must exhibit an element of X that is not an element of Y or an element of Y that is not an element of X.

There are two standard ways of describing a set. The first is to list the elements of the set separated by commas and enclosed in braces. Thus

$$S = \{1, 2, 3, 5, 8, 13\}, \qquad U = \{2, 3, 5, 7, 11\},$$
$$A = \{2, 4, 8, 16\}, \qquad B = \{16, 8, 4, 2\},$$
$$C = \{2, 4, 8, 16, 8, 4, 2\}$$

are all sets whose elements are positive integers. Since neither the order in which the elements are listed nor the fact that some elements are repeated has any significance, the sets A, B, and C are all equal.

The easiest sets to represent in APL are finite sets of real numbers. Since the entries of an APL array are real numbers, we may simply define a vector whose components list the elements of the set. Thus, if

```
S←1 2 3 5 8 13          B←16 8 4 2
U←2 3 5 7 11            C←2 4 8 16 8 4 2
A←2 4 8 16
```

then the vectors S, U, A, B, C correspond naturally to the definition of the sets S, U, A, B, C. (To save space, APL dialogues such as the preceding one are printed in two columns. At a terminal, they would appear as one long column.)

The notation $\{a_1, \ldots, a_n\}$ and the use of an APL vector to list the elements of a set both have the drawback that the representation for a particular set is not unique. In general, there is no natural order on the ele-

ments of a set but, if the elements of the set are real numbers, we do get a unique representation if we assume the elements are listed in increasing order and without repetitions. The procedure *SSORT* in *CLASSLIB* produces this standard list of the elements in the set described by an arbitrary vector.

```
      B
16 8 4 2
      SSORT B
2 4 8 16
```

```
      C
2 4 8 16 8 4 2
      SSORT C
2 4 8 16
```

If X is an APL vector, we will often speak of "the set X instead of "the set represented by X". In particular, we will often refer to the set ιN, which can, of course, mean either $\{1,2,\ldots,N\}$ or $\{0,1,\ldots,N-1\}$, depending on the index origin.

Sets whose elements are not real numbers are more difficult to represent in APL. We will have to represent sets of sets of real numbers, sets of polynomials, and many other types of sets. Techniques for doing this will be discussed as the need arises.

The second way to describe a set is to specify a property that characterizes the elements of that set. The statement

$$X = \{x|P(x)\}$$

is read "X is the set of all x such that the property P holds for x." For example, we may define two sets L and M as follows:

$$L = \{x|x \text{ is a positive real number}\},$$

$$M = \{t|t \text{ is an even integer}\}.$$

There are a few sets that will come up so frequently that it is convenient to have special symbols for them. The set of *integers* will be denoted by **Z** and the set of positive integers or *natural numbers* by **N**. The symbols **Q**, **R**, and **C** will stand for the set of *rational numbers*, the set of *real numbers*, and the set of *complex numbers*, respectively.

Suppose we define a set E by

$$E = \{x|x \in \mathbf{R},\ x^2 = -1\}.$$

Since every real number has a nonnegative square, E has no elements. Such a set is said to be *empty*. The following theorem shows, among other things, that the set of all unicorns is equal to the set of all letters in the English alphabet that come after the letter Z.

THEOREM 1. Any two empty sets are equal.

Proof. Let X and Y be empty sets. Since X is empty, we cannot find any element x of X, and so we certainly cannot produce an x in X that is not in Y. Similarly, we cannot find an element of Y that is not in X because Y has no elements. Thus we are forced to conclude that the statement $X \neq Y$ is false, so X and Y must be equal. □

By Theorem 1 we may speak of *the* empty set, since there is only one. It will be denoted by the symbol $\emptyset$. Whenever we define a property that a particular set may or may not have, it is a useful exercise to determine whether the empty set has the property.

A set A is a *subset* of a set B if every element of A is also an element of B. In this case, we also say A is *contained in* B or that B *contains* A. A subset A of B is called a *proper subset* if $A \neq B$, that is, if there is some element of B that is not in A. We will write $A \subseteq B$ when A is a subset of B and $A \subset B$ when A is a proper subset of B. Some authors prefer to write $A \subset B$ where we write $A \subseteq B$. The notation used here has been chosen because it parallels the use of $<$ and $\leq$ to denote inequality of real numbers. The statements $B \supseteq A$ and $B \supset A$ mean $A \subseteq B$ and $A \subset B$, respectively. We have $\emptyset \subseteq A$ and $A \subseteq A$ for any set A.

If the APL vectors A and B list the elements of two subsets A and B of **R**, then the assertion $A \subseteq B$ corresponds to the APL proposition $\wedge/A \in B$. (An APL *proposition* is an APL expression with one entry, which is either 1 or 0. The APL membership operation $\in$ is described in Section A1.5 and the operation $\wedge/$, called "and reduction", is discussed in Section A1.6.) For example,

```
      A←1 3 5
      B←1 2 3 4 5
      C←2 3 5 7
      ∧/A∈B
1
      ∧/A∈C
0
```

Here A is a subset of B but not of C.

For the time being, we will rely on our intuition concerning the term "finite set", which will be defined in Section 3 of this chapter. If X is a finite set, then $|X|$ is the *cardinality* of X, that is, the number of elements of X.

Suppose $A_1, \ldots, A_k$ are sets of real numbers such that $|A_i| = m$ for $1 \leq i \leq k$. We can represent $\{A_1, \ldots, A_k\}$ by a k-by-m matrix A such that the Ith row $A[I;]$ of A lists the elements of A_I. For example, *CLASSLIB* contains a procedure *SSUB* such that `A←K SSUB N` defines A to be a matrix whose rows list the K-element subsets of ιN.