COMPUTER APPLICATIONS IN FINITE MATHEMATICS AND CALCULUS

SECOND EDITION

CAROLYN L. MEITLER
MICHAEL R. ZIEGLER

COMPUTER APPLICATIONS IN FINITE MATHEMATICS AND CALCULUS

Second Edition

CAROLYN L. MEITLER
Marquette University

MICHAEL R. ZIEGLER
Marquette University

Dellen Publishing Company
San Francisco and Santa Clara, California

PERMISSIONS: Dellen Publishing Company
400 Pacific Avenue
San Francisco, California 94133

ORDERS: Macmillan Publishing Company
Front and Brown Streets
Riverside, New Jersey 08075

Collier Macmillan Canada, Inc.

Library of Congress Cataloging in Publication Data

ISBN 0-02-431490-0

Printing: 1 2 3 4 5 6 7 8 Year: 4 5 6 7 8 9 0

PREFACE

The purpose of this book is to present the computer as a tool that can be used as part of the problem-solving process within the context of a college mathematics course. The solutions of many problems involve calculations that are best performed on a computer. The programs in this book can be used to solve such problems, eliminating the need to restrict one's attention to oversimplified problems that can be solved easily by hand. Furthermore, since the level of computer expertise expected of college graduates is steadily increasing, the inclusion of the material in this book in a college mathematics course will give students additional hands-on experience which will be valuable in their future professional activities.

The topics presented in this text have been selected from those covered in the following series of college mathematics texts by Raymond A. Barnett and Michael R. Ziegler:

College Mathematics for Management, Life, and Social Sciences

Finite Mathematics for Management, Life, and Social Sciences

Calculus for Management, Life, and Social Sciences

Applied Calculus for Business and Economics, Life Sciences, and Social Sciences

Applied Mathematics for Business and Economics, Life Sciences, and Social Sciences

Definitions of important terminology, examples, and exercises are included for each topic. Thus, this book can be used in conjunction with any of the books in the above series or any other textbook that deals with finite mathematics, introductory calculus, or both.

Thirty-seven BASIC programs are presented in this book in eighteen chapters. At least one example and sample execution are included for each program. These provide sufficient information so that students with no previous computer experience can use the programs with a minimum of instruction. The first two chapters contain a brief introduction to the use of BASIC programs on a computer and some very simple programming concepts. Chapters 3 through 11 cover topics that are usually found in a finite mathematics course and Chapters 12 through 18 deal with topics that are related

to introductory calculus. The Appendix contains a summary of the features of the BASIC language used in these programs. Answers to the odd-numbered exercises are included in the back of the book. These answers include the output generated by the program used in the exercise.

The programs are written in a structured format with complete variable definitions and many comments. Each program has been kept as simple as possible, with legibility always considered to be more important than efficiency. No advanced programming concepts are used. Level 0 BASIC, as dexcribed in the *Conduit BASIC Guide* (Conduit, The University of Iowa), is used throughout in order to make the programs as portable as possible. A VAX-11/780 computer was used to write all the programs and produce all the computer output that appears in this book. Whenever possible, the output from programs was formatted to fit in a 40-column display; however, some programs do produce output that cannot conveniently be printed in 40 columns. These programs will have to be modified by the user whose display is limited to 40 columns or fewer. It may also be necessary to make other minor changes in some of the programs, such as changing the exponentiation symbol from ** to ^.

It is assumed that the instructor will provide the students with information concerning loading, executing, and storing programs, and other system dependent operations. In addition, students will have to learn how to change data statements and function definitions, and to write arithmetic expressions in BASIC. This material is covered at the appropriate points in the book. Students are not assumed to have had any previous programming experience, and no attempt is made to teach computer programming. There are a few exercises that involve making changes in programs in the book which can be assigned to students who are familiar with BASIC programming.

All the programs are available on diskettes for APPLE II computers and IBM personal computers. One of these diskettes can be obtained from the publisher without charge by any institution that adopts this book or any of the books in the series that was listed earlier in this preface. Instructors and students have permission to make additional copies of these programs for instructional purposes.

There are several ways that the material in this book can be incorporated into a mathematics course, depending on the amount of time available and the availability of computing equipment. Instructors can use the programs to produce material for classroom use without requiring the students to use the computer at all. For example, students can be asked to interpret the results of a computer solution to a linear programming problem supplied to them by the instructor. Or students can be instructed to execute programs prepared by the instructor to illustrate and reinforce the concepts discussed in the classroom. Markov chains and Newton's method are two good examples of topics whose presentation can be enhanced by just having students execute programs which already contain the necessary data statements and function definitions. Finally, if students are taught how to enter new data statements and function definitions, then they can use the computer to solve problems from their textbook or from this book and to experiment with problems of their own construction.

We would like to express our appreciation to Don Dellen and his staff for their support of this project, our colleagues at Marquette University for class testing much of the material in this book, Glenn Saito for his assistance in preparing the microcomputer diskettes, and Jo Ann Vine for her excellent production of the book.

CAROLYN L. MEITLER
MICHAEL R. ZIEGLER

CONTENTS

PROGRAMS

CHAPTER 1 INTRODUCTION TO COMPUTERS

This chapter provides a brief introduction to the structure of a computer and the use of an operating system. Variables, numbers, and algebraic expressions are also discussed.

The computer is a powerful tool for solving mathematical problems because of its ability to store large amounts of information and to perform many calculations in a short period of time. Both the instructions for the operations the computer is to perform and the information to be used in these operations can be stored in the computer. The set of instructions is called a *program*. The program can be changed, stored, and used repeatedly, so that a single computer can solve a wide variety of problems. The instructions must be written in one of several special languages, called *programming languages*, which the computer can understand. One of the most popular programming languages is the BASIC language. This the language used throughout this book.

STATEMENTS AND SYSTEM COMMANDS

In order to use the computer to solve problems, you must be able to give instructions to the computer. The instructions which you will giving the computer can be divided into two categories—*statements* and *system commands*. *Statements* are instructions which direct the computer to perform an operation, such as adding two numbers, printing a number, etc. A *program* is a collection of these statements identified by line numbers. This book presents a collection of programs which will help you solve certain mathematical problems that would be extremely difficult and time-consuming to do by hand. A description of those statements from the BASIC language which we will be using is in the Appendix.

System commands instruct the computer to perform certain actions with entire programs. In order to understand how to use these commands, it is helpful to view a computer as a system of interconnected components. To begin with, the central processing unit, or *CPU*, is the heart of any computer. This is the component that actually carries out the instructions given to the computer. The instructions for the program being executed and the data being used by that program are both

stored in a second component, generally called the *main memory*. All computers have a CPU and a main memory. The remainder of the components can be quite varied. Typically, there is an *input device*, such as a typewriter-like keyboard, and an *output device*, such as a TV screen or a printer. Finally, since the instructions that are stored in the main memory may be destroyed when another program is entered or the computer is turned off, it is necessary to have some means of permanently storing programs so that they can be used again. Magnetic tapes, magnetic disks, and punched cards are the most common *auxiliary storage devices*. For convenience in our discussions, we will assume that a magnetic disk is being used for auxiliiay storage. A graphic representation of the process we have just discussed is shown in Figure 1.1.

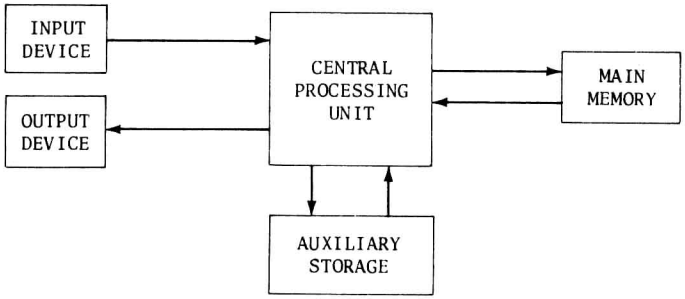

Figure 1.1

System commands instruct the computer to perform certain actions involving one or more of these components. The actual form of the command will vary from one computer to another, but the fundamental operations which must be performed are the same on every computer. In order to effectively use a computer system, you must be able to:

1. Begin the session with the computer.
2. Load a program from a disk or enter a new program from the keyboard.
3. Display the program which is currently stored in the main memory.
4. Make changes in the program which is currently stored in the main memory.
5. Execute the program which is currently stored in the main memory.
6. Save the current contents of the main memory on a disk.
7. End the session with the computer.

EXAMPLE OF USING SYSTEM COMMANDS

Since the procedure for beginning a session depends on the type of computer being used, we will assume that we have started the session. Suppose that we wish to perform the operations listed

above on a program called SAMPLE which is already stored on the disk.

Step 1 LOAD SAMPLE (ret)

This instructs the computer to locate the program called SAMPLE on the magnetic disk used for auxiliary storage and to place it in the main memory. (ret) represents the return key. You press the return key to tell the computer that you have finished entering a line.

Step 2 LIST (ret)

This instructs the computer to print the contents of the portion of main memory where instructions are stored. The computer prints

```
100  REM  **********
110  REM  * SAMPLE *
120  REM  **********
130  PRINT "THIS IS A SAMPLE PROGRAM EXECUTED BY"
140  PRINT "YOUR NAME GOES HERE"
150  END
```

SAMPLE uses three statements from the BASIC language, REM, PRINT, and END. REM is short for REMARK and indicates that this line contains information for the person reading the program. The computer ignores any line which begins with the word REM. PRINT instructs the computer to print whatever is enclosed within quotation marks. We will see other ways to use the PRINT statement later. END is the last statement in any program.

Step 3 RUN (ret)

This instructs the computer to execute the program which is currently in its main memory. In this example, the computer would print:

```
THIS IS A SAMPLE PROGRAM EXECUTED BY
YOUR NAME GOES HERE
```

Obviously, the phrase YOUR NAME GOES HERE is not what we want to be printed. Let's change the program so that an actual name will be printed.

Step 4 140 PRINT "ALEXANDER AARDVARK" (ret)

This instructs the computer to replace the original line 140 which is currently in the main memory, with the line that has been entered as shown above. It is extremely important for you to understand that only the current contents of the main memory are changed by this command. The original version of SAMPLE that is stored on the disk has not been changed. This type of instruction can also be used to add or to delete lines in the program in the main memory. To add a new line, use a line number which has not been used before to insert the statement in the proper position in the program.

Step 5 145 PRINT "GOODBYE FOR NOW"

This instructs the computer to store this instruction in the main memory between lines 140 and 150. To delete a line, type the number of the line followed by (ret).

Step 6 100 (ret)

This instructs the computer to erase the instruction that is stored in line 100. Now use the LIST command to see the changes we have made.

Step 7 LIST (ret)

The computer prints

```
110  REM  * SAMPLE *
120  REM  **********
130  PRINT "THIS IS A SAMPLE PROGRAM EXECUTED BY"
140  PRINT "ALEXANDER AARDVARK"
145  PRINT "GOODBYE FOR NOW"
150  END
```

When we compare this with the listing printed in Step 2, we can see that line 100 is gone, that line 140 has been changed, and that line 145 has been added. We have changed *only* the contents of the main memory. We have not altered the original version of SAMPLE which is still stored on the disk.

Step 8 RUN (ret)

This instructs the computer to execute the current contents as listed in Step 7. The computer prints

```
THIS IS A SAMPLE PROGRAM EXECUTED BY
ALEXANDER AARDVARK
GOODBYE FOR NOW
```

If we wish to store the program that is currently in the main memory, we must choose a name for it. Using the same name—that is, SAMPLE—may destroy the original version, so we will choose a different name.

Step 9 SAVE NUSAMP (ret)

This instructs the computer to store the current contents of the main memory on the magnetic disk under the name NUSAMP. Step 9 is essential if you want to keep a permanent copy of your program. A very common and frustrating oversight is to enter a program, or to make extensive changes in a program, and then forget to save the program before ending the session with the computer.

This example does not include all of the BASIC system commands. You may want to obtain a list of the system commands for your computer to see if any of the commands on your computer differ from the ones used in this example. You may also want to learn the system commands that will delete a program from the auxiliary storage, type a list of all the programs that you have

stored on auxiliary storage, or prepare the main memory to accept a new program that you wish to enter.

STORING AND RETRIEVING DATA FROM MAIN MEMORY

We mentioned earlier that both the program and the data used by the program can be stored in the computer's main memory. We will now see how to store and retrieve data in a program. We will consider only numbers, such as the coordinates of a point or the coefficients of a linear equation. The portion of the main memory where data is stored can be visualized as a collection of boxes. Each box has an address giving its location in the memory and each box can hold one number. We do not have to worry about the actual location of these boxes in the computer's memory. Instead, we can use symbolic names for these locations and let the computer worry about the actual locations. As an example, consider this program.

```
100  LET X = 10
110  PRINT X
120  END
```

Line 100 instructs the computer to store the value 10 in a memory location whose address is X. We do not know where that location is, but the computer does. Line 110 instructs the computer to go to the location whose address is X and to print the value of the number that is stored there. It is very important to understand that the symbol X represents the location of a number that is stored in the computer's memory, *not* the value of the number. Let us look at another example.

```
100  LET X = 10
110  PRINT "FIRST X =", X
120  LET X = X + 5
130  PRINT "NOW X =", X
140  END
```

Line 100 is the same as in the previous example. Line 110 has been modified to print the phrase "FIRST X =" before printing the value that is stored in the location whose address is X. Line 120 requires very close examination. The computer will perform the following actions when it executes this line.

1. Locate the memory position whose address is X.
2. Obtain the value of the number that is stored there (in this case, 10).
3. Add 5 to this value, obtaining a new value of 15.
4. Store the new value in the memory location whose address is X.

Notice that the previous value of 10 is destroyed by this action. A memory box can hold only one number at a time. Line 130 instructs the computer to print the phrase "NOW X =" followed by the value currently stored in the location with

address X. Executing this program produces the following output:

```
FIRST X = 10
NOW X =  15
```

As this example points out, the same location can hold different values during the course of a program. For this reason, a symbol such as X in the program given above is called a variable. To be precise, a *variable* is a symbol that represents the address of a location in the memory of the computer where the value of a number can be stored. A value can be stored in that location by using the variable in a LET statement.

It is unfortunate and somewhat confusing that the term "variable" has a slightly different meaning in mathematics. In algebra, you learned that a variable is a symbol which represents the value of a number. In computing, a variable is a symbol which represents the location of the value of a number. These two interpretations are not equivalent. As we have previously explained, the statement "LET X = X + 5" makes perfectly good sense to a computer. It does not make sense as an algebraic equation. In computing, this statement means "assign to the location whose address is X the value that is obtained by adding 5 to the value currently stored in the location whose address is X." This is often shortened to "let X be assigned the value X plus 5."

So far we have used only the letter X to represent a variable. In BASIC you can use any of the letters of the alphabet or any of the letters followed by a single digit to designate variables. Thus, A, MØ, and Z9 are all valid variable names, while AA, MO, P#, and Z19 are all invalid variable names. (Ø is used to distinguish the number zero from the letter "oh.")

NUMBERS IN BASIC

Notice that, in addition to the variable X, the preceding program used the numbers 5 and 10. The rules for using numbers in BASIC are fairly simple. A minus sign goes in front of a negative number. The plus sign may go in front of a positive number, but it is not required. If a decimal is used, there must be at least one digit before the decimal point and at least one digit after the decimal point. Commas are not used within a number. In BASIC, commas are used to separate numbers in a list. The computer will interpret the number 1,900,400 as a list of three numbers: the number 1, followed by the number 900, followed by the number 400. It will not read this as the number one million, nine hundred thousand, four hundred. The total number of digits that can be used in a number depends on the computer you are using, but almost all will accept numbers with seven or eight digits. The computer will also accept numbers that have been written in scientific or exponential notation. The exponential form

for 1900400 is 1.9004E+6 and is interpreted as 1.9004×10^6. The exponential form for -0.000215 is -2.15E-4 and is interpreted as -2.15×10^{-4}. The computer will automatically use exponential notation when it is printing very large or very small numbers.

EXPRESSIONS IN BASIC

In order to solve mathematical problems on a computer, it is necessary to know how to write arithmetic expressions in the BASIC language. The symbols for addition and subtraction are the usual ones, + and -.

Example. The linear expression $x - y + 5$ is written as X - Y + 5 in BASIC.

The symbol for multiplication is * and must be used to indicate each multiplication operation.

Example. The linear expression $2x + 4y - 7$ is written as 2*X + 4*Y - 7 in BASIC.

Division is denoted by /. Parentheses must be used if the divisor has more than one term in it.

Example. $\frac{1}{x} + \frac{1}{y}$ is written as 1/X + 1/Y, but $\frac{1}{x + y}$ is written as 1/(X + Y).

Exponentiation is denoted by ^, **, or ↑, depending on the keyboard of the computer being used. We will use ** throughout this book.

Example. The quadratic expression $2x^2 + 3x - 5$ would be written as 2*X**2 + 3*X - 5.

The expression $\sqrt[3]{x + 2}$ would be written as (X + 2)**(0.333333) or as (X + 2)**(1/3).

Parentheses are also used to avoid writing expressions with two possible interpretations.

Example. The expression 8/4/2 could be interpreted as

$$\frac{\frac{8}{4}}{2} = \frac{8}{2} = 4, \text{ or as } \frac{8}{\frac{4}{2}} = \frac{2}{2} = 1.$$

In fact, the computer would choose the second interpretation, but the problem can be avoided by using parentheses in the following ways.

If you want $\frac{\frac{8}{4}}{2}$, write 8/(4/2), and if you want $\frac{8}{\frac{4}{2}}$, write (8/4)/2.

CHAPTER 1 EXERCISES

1. In order to familiarize yourself with the system commands for your computer, repeat each of the steps listed in the text for program SAMPLE. Remember, some of the system commands on your computer may differ from those used here.

2. Given that X = 2, Y = 3, and Z = 4, find the value of each of the following BASIC expressions.

(a)	`X + Y`	(f)	`Z - (X - Y)`
(b)	`X + (Y/2)`	(g)	`(Z/X)*Y`
(c)	`X + (Y*Z)`	(h)	`(Z*Y)/(X**2)`
(d)	`(X + Y)*Z`	(i)	`(X + Y)*X + Y`
(e)	`Z - X - Y`	(j)	`(Y**X)**Z`

3. In each of the following program fragments, determine the values stored in X, Y, and Z after all the statements have been executed.

(a)
```
100  LET X = 2
110  LET Y = X
120  LET Z = X + Y
```

(b)
```
500  LET X = 3
520  LET Y = X**2 - 1
535  LET X = X + Y
560  LET Z = X + 2*Y
```

(c)
```
1000  LET X = 4
1005  LET Y = (X/2)*3
1006  LET Z = X + Y
1009  LET X = Z - X + Y
1013  LET Y = X*Y*Z
```

4. Write the following algebraic expressions as BASIC expressions.

(a)	x^3	(f)	$(a + 4)^2$
(b)	$\sqrt{y}$	(g)	$pr + pq$
(c)	$3z^3 - 4z^2 + 2z - 5$	(h)	$\frac{bc}{d}$
(d)	$\sqrt{u^2 + v^2}$	(i)	$\frac{b}{cd}$
(e)	$\frac{2x + 3}{4x + 5}$	(j)	x^{x^2+1}

5. Write each of the following decimals in BASIC exponential form.

(a)	25.637	(d)	-0.0000001
(b)	-35674.57	(e)	621342000000
(c)	0.0042567	(f)	2.78183

6. Write each of the following numbers in ordinary decimal form.

(a)	3.21E+04	(c)	9.9999E-01
(b)	-2.1356E+06	(d)	-9.876E-03