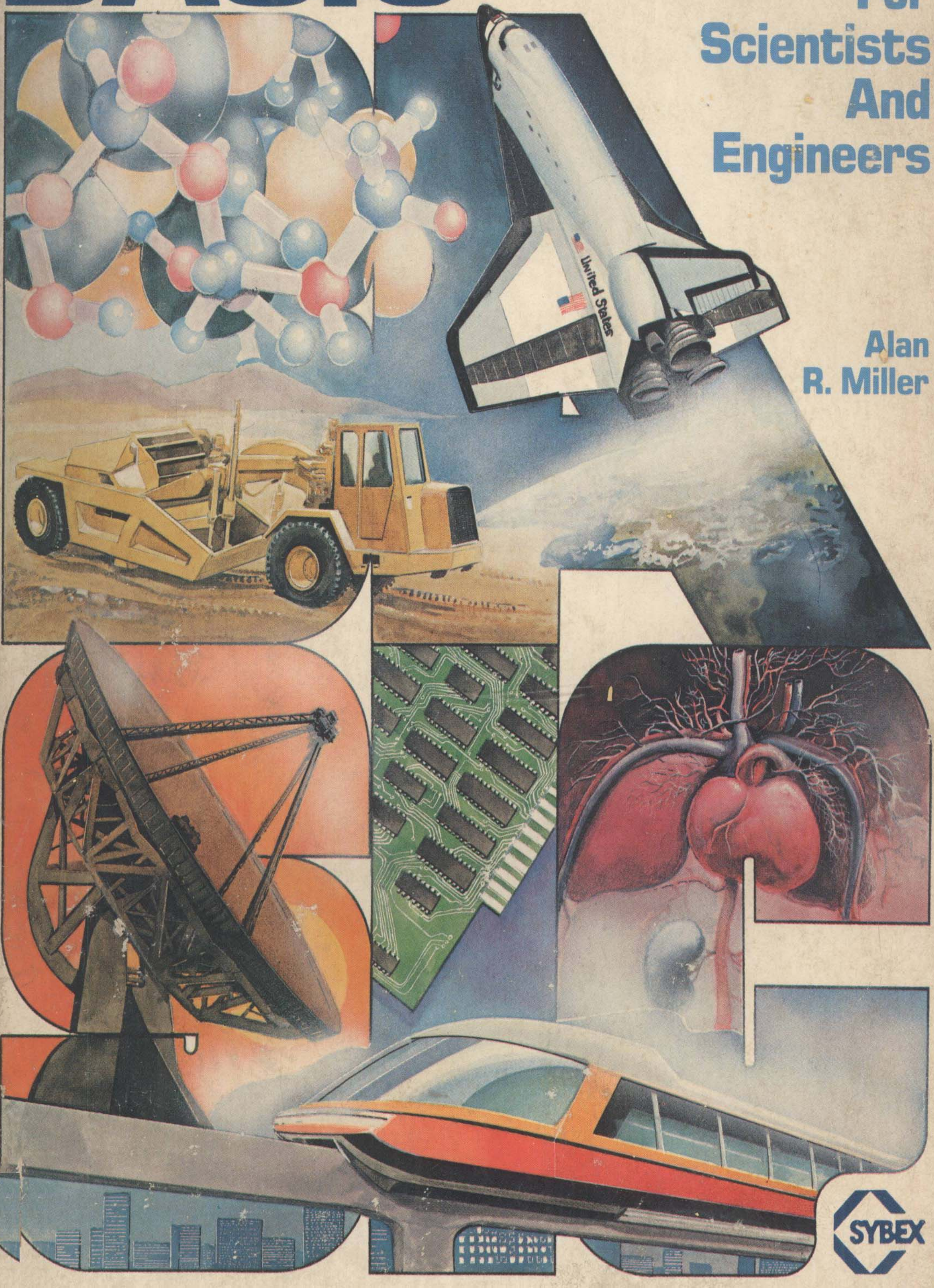
BASIC Programs For Scientists And Engineers

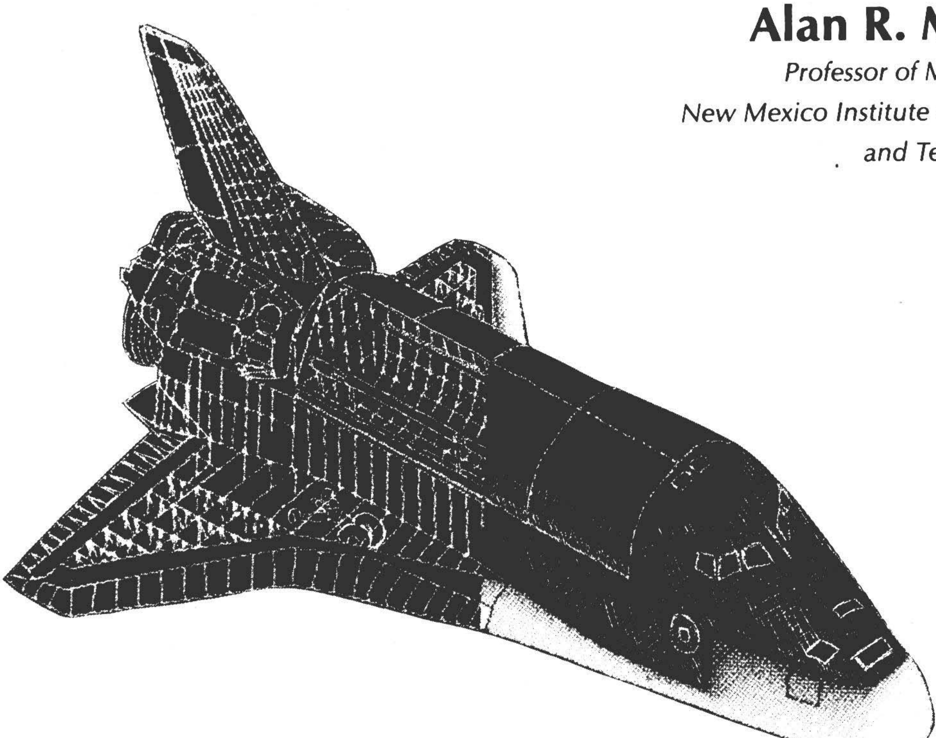
Alan R. Miller



SYBEX

BASIC PROGRAMS for Scientists and Engineers

Alan R. Miller
Professor of Metallurgy
New Mexico Institute of Mining and Technology



Berkeley • Paris • Düsseldorf

CREDITS

Cover Design by Daniel Le Noury
Technical illustrations by J. Trujillo Smith, Jeanne E. Tennant

NOTICES

Apple is a registered trademark of Apple Computer Corporation.
BASIC-80, Microsoft BASIC, and BASCOM are registered trademarks of Microsoft Consumer Products.
CBASIC is a registered trademark of Compiler Systems, Inc.
DEC-20 is a registered trademark of Digital Equipment Corporation.
Lifeboat 2.2 is a registered trademark of Lifeboat Associates.
North Star BASIC is a registered trademark of North Star Computers, Inc.
PET is a registered trademark of Commodore Business Machines.
TRS-80 is a registered trademark of Tandy Corporation.
Xitan BASIC is a registered trademark of Xitan Systems Ltd.
WordStar is a registered trademark of MicroPro International Corporation.
Z80 is a registered trademark of Zilog, Inc.

SYBEX is not affiliated with any manufacturer.

Every effort has been made to supply complete and accurate information. However, Sybex assumes no responsibility for its use, nor for any infringements of patents or other rights of third parties which would result.

Library of Congress Card Number: 81-84003
ISBN 089588-073-3
First Edition 1981
Printed in the United States of America
10 9 8 7 6 5 4 3 2

BASIC PROGRAMS for Scientists and Engineers

Preface

The ideas and material for this book have been developed during my experience teaching sophomore, junior, and senior engineering students over the past 14 years. I have used FORTRAN, BASIC, and Pascal as the computer languages for these courses.

All of the BASIC programs in this book were developed on a Z80 microcomputer. The operating system was the Lifeboat 2.2 version of CP/M. The source programs were developed and executed with Microsoft BASIC-80, Version 5. MicroPro's WordStar was frequently used to modify the ASCII form of the source programs. Most of the programs were also run on other BASICs. These included Microsoft's BASCOM (compiling BASIC), CBASIC, Xitan BASIC, North Star BASIC, and BASIC Plus-4 running on a DEC-20.

The manuscript was created and edited with MicroPro's WordStar running on the same Z80 computer. The BASIC source programs have been incorporated directly into the manuscript from the original source files. Computer printouts shown in the figures were also incorporated magnetically into the manuscript. This was accomplished by altering the CP/M operating system so that console output was written into a block of memory. This block was then saved as a disk file. The final manuscript was submitted to SYBEX in a magnetic form compatible with the photocomposer. Consequently, the manuscript and the BASIC source programs have not been retyped.

I am sincerely grateful for the helpful guidance and suggestions of Rudolph Langer and Douglas Hergert during the development of the manuscript. I would also like to thank Ashok Singh for checking the manuscript, especially the mathematical expressions.

Alan R. Miller
Socorro, New Mexico
August 1981

Introduction

The purpose of this book is twofold: to help the reader develop a proficiency in the use of BASIC, and to provide a library of programs useful for solving problems frequently encountered in science and engineering.

The programs in this book, the second in the SYBEX *Programs for Scientists and Engineers* series, are most valuable to the practicing scientist or engineer. The material is also suitable for a junior- or senior-level engineering course in numerical methods. The reader should have a working knowledge of an applications language such as BASIC, FORTRAN or Pascal. Experience with vector operations and differential and integral calculus will also be helpful.

BASIC is currently the most widely available high-level computer language, especially on microcomputers. Because it is usually implemented as an interpreter, it is a particularly easy language to use: program execution can be interrupted at any point, current values can be printed, and then execution can be resumed.

The lowest level of BASIC is extremely limited. Variable names should be no longer than two characters, and looping control is only available with the **FOR /NEXT** and **GOTO** constructions. Fortunately, however, powerful BASICs are becoming more common. These contain features such as long variable names, the **WHILE/WEND** construction and integer typing of variables. The programs in this book were initially developed using a powerful set of BASIC features, and then simplified to the lowest common denominator. Thus, they can be run on all of the common BASICs. In fact, these programs will run on microcomputers such as the Apple, the TRS-80, and the PET, on Microsoft BASIC and on CBASIC. In addition, they will run as well on larger computers such as the DEC-20 with only minor changes to features such as the random number generator and the multiple statement separator. Users are urged to upgrade the programs to the highest possible level. To help with this conversion, suggested alternate names are given in the source programs, using **REM** lines such as the following:

```
11 REM identifiers
12
13 REM   N1%   NROW%   number of rows
14 REM   N2%   NCOL%   number of columns
15 REM end of identifiers
```

In lines 13 and 14, the two-character variable names used in the program are followed by longer and more descriptive names suitable for less restrictive BASICs, and then by comments identifying the variables.

The line numbers for the programs in this book have been integrated so that there is a minimum of conflict. It should thus be possible to combine any of the different subroutines into a single program. Duplicate line numbers are only utilized for the same kinds of tasks. The assigned blocks of line numbers are detailed below:

1 - 499	Main program
500 - 880	Input routine
800 - 880	Set up matrix
1000 - 1370	Output routine
2000 - 2410	Numerical integration
2500 - 2810	The error function
3000 - 3550	Sorting routine
3700 - 3790	Conversion to upper case
4000 - 4310	Conversion to square matrix
4400 - 4930	Bessel function of the second kind
5000 - 6150	Matrix solution
7000 - 7990	Plot routine
8000 - 8160	Newton's method
8400 - 8430	Function evaluation
8500 - 8780	Bessel function of the first kind
9000 - 9100	Mean and standard deviation
9200 - 9400	Gamma function
9500 - 9580	Gaussian random numbers
9800 - 9840	Normally distributed random numbers
9999	**END** statement

The reader who is primarily interested in the BASIC programs developed in this book will have no trouble locating them; the sections that contain programs or subroutines are clearly labeled. It should be noted, however, that this book is designed to be read from beginning to end. Each chapter discusses and develops tools that will be used again in subsequent chapters. The mathematical algorithms of each program are methodically described before the program itself is implemented, and sample output is supplied for most of the programs. The following brief descriptions summarize the contents of each chapter.

Chapter 1, ***Evaluation of a BASIC Interpreter or Compiler***, identifies weak points in several commercial BASICs, and supplies programs for testing any BASIC. The results will be used to select various constants and operations in later chapters. Also included in Chapter 1 are discussions of the common "**NEXT** without **FOR**" bug and the "10 **GOSUB** 10" problem.

Chapter 2, ***Mean and Standard Deviation***, discusses some simple statistical algorithms and presents a program for implementing them. Routines for generating—and testing—both uniform and Gaussian random numbers are also given.

Chapter 3, ***Vector and Matrix Operations***, summarizes the operations of vector and matrix arithmetic, including dot product, cross product, matrix multiplication and matrix inversion. Two important programs are developed—one for carrying out matrix multiplication, and another for calculating determinants.

Chapter 4, ***Simultaneous Solution of Linear Equations***, presents programs to carry out the algorithms of Cramer's rule, the Gauss elimination method, the Gauss-Jordan elimination method, and the Gauss-Seidel method—all for solving simultaneous equations. In addition, ill conditioning is studied by observing a program that generates Hilbert matrices, and a program is developed for solving equations with complex coefficients.

Chapter 5, ***Development of a Curve-Fitting Program***, is the first of a series of chapters on curve fitting. In a good illustration of top-down program development, a linear least-squares curve-fitting program is written and discussed. The program includes routines to simulate data, plot curves, compute the fitted curve, and supply the correlation coefficient.

Chapter 6, ***Sorting***, describes and compares several BASIC sorting routines, including two bubble sorts, a Shell sort and a nonrecursive quick sort. A sort routine is incorporated into the curve-fitting program of Chapter 5 to enable the program to handle real experimental data.

Chapter 7, ***General Least-Squares Curve Fitting***, extends the curve-fitting program to general polynomial equations, and finds curve fits for three examples: the equations for heat capacity, vapor pressure, and properties of superheated steam.

Chapter 8, ***Solution of Equations by Newton's Method***, presents a series of programs that use Newton's algorithm for finding the roots of an equation. This tool will be used again in Chapter 10 for nonlinear curve fitting.

Chapter 9, ***Numerical Integration***, develops programs for three different integration methods—the trapezoidal rule, Simpson's rule, and the Romberg method. End correction is also discussed. Simpson's rule will be used in Chapter 11 for evaluating the Gaussian error function.

Chapter 10, ***Nonlinear Curve-Fitting Equations***, discusses curve-fitting algorithms for the rational function and the exponential function. Two examples are given—the Clausing factor, and the diffusion equation.

Chapter 11, ***Advanced Applications: The Normal Curve, the Gaussian Error Function, the Gamma Function, and the Bessel Functions***, addresses several advanced topics in programming for mathematical applications. This last chapter summarizes and expands upon a number of the concepts presented earlier in the book.

Each chapter also contains exercises designed to extend the reader's comprehension of the material.

For readers who are approaching BASIC for the first time, a summary of the syntax, standard functions, and reserved words of BASIC is included in the appendices. The real educational experience of this book, however, will be gained by carefully working through the programs themselves.

A Note on Typography

Many readers of programming books believe that typeset programs are more likely to contain errors than photographed programs, that errors may be introduced into the programs during the typesetting process. At SYBEX, we have developed a method of typesetting programs (thus enhancing their legibility) without introducing errors. The author submits tested, executable source code on disks. The programs are then run through a formatting program on our in-house computers. The formatting program establishes conventions for spacing, capitalization, etc., so that the programs will have a uniform appearance, without altering their content. Next, the programs (along with the text) are transmitted electronically to our computerized phototypesetter. At no point are the programs actually retyped, so errors that might have been introduced by retyping have been avoided.

The text of this book has been set in the typeface known as Oracle, the programs are in Futura, and the output has been photographed from the actual line-printer output supplied by the author. Reserved words appear in **boldface**. Mathematical expressions (variables, letter constants) appear in *italics*, except for vectors, which appear in boldface roman. For example:

$$A + Bx + Cx^2 = 0$$

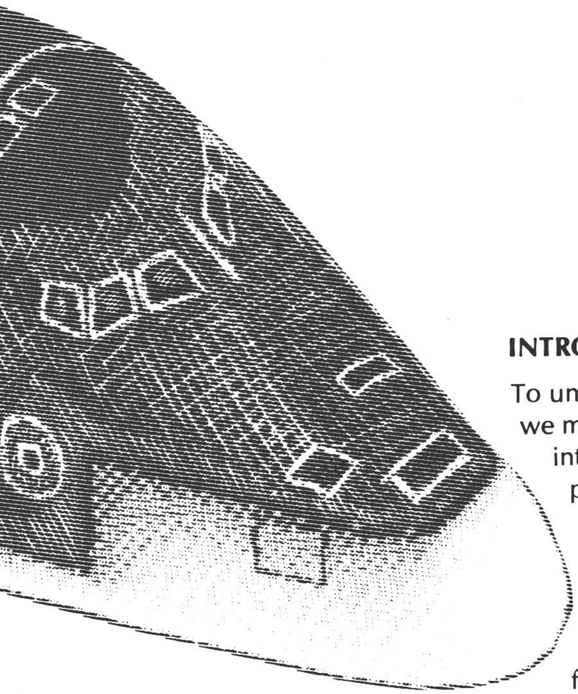
$$\mathbf{v} = [1\ 2\ 3]$$

Throughout the book an effort has been made to distinguish typographically between mathematical values and program structures. Thus, juxtaposed in a single paragraph, the reader may see references to the variable x and the BASIC variable X; the vector **v** and the BASIC array V; the matrix element v_{ij} and the BASIC array element V(I,J).

Contents

CHAPTER 1

Evaluation of a BASIC Interpreter or Compiler



INTRODUCTION

To understand the results of a BASIC program we must be familiar with the limitations of the interpreter or compiler we are using. This is particularly true with scientific application programs such as the ones given in this book. In this first chapter, then, we will present some tools for evaluating the precision and range of BASIC. In fact, the examples given here were derived from several popular BASICs.

PRECISION AND RANGE OF FLOATING-POINT OPERATIONS

Many of the programs in this work are sensitive to the *precision* and *dynamic range* of the BASIC floating-point operations. For example, in one program an algorithm is terminated when a particular term is smaller than a relative tolerance. The formula in this case is:

TERM < SUM * TOL

where TERM is the value of the new term, SUM is the current total, and TOL is an arbitrarily small number known as the *tolerance*.

It is important that the value chosen for the tolerance be within the accuracy of the floating-point operations. Otherwise, the summation step will never terminate. Suppose, for example, that the floating-point operations are performed to a precision of six significant figures. Then, the tolerance must be set to a value larger than 10^{-6}.

The dynamic range of the exponent is a separate matter. Typical binary, floating-point operations are performed with 32 bits of precision. BCD (Binary Coded Decimal) floating-point packages, on the other hand, will usually retain more significant figures and have a greater dynamic range.

We will now present a BASIC program for testing the precision and dynamic range. We will investigate output from several common BASICs to illustrate both mantissa and exponent accuracy.

BASIC PROGRAM: A TEST OF THE FLOATING-POINT OPERATIONS

The program given in Figure 1.1 can be used to determine the precision and the dynamic range of BASIC. Type up the program and execute it. The initial value of X is obtained by dividing 1.0E−4 by 3. Then, successively smaller and smaller values of X are calculated and displayed

```
10  N% = 18
20  X = 1.0E-4 / 3
30  FOR I% = 1 TO N%
40     X = X/10
50     PRINT I%, X,
60     X = X/10
70     PRINT X
80  NEXT I%
90  END
```

Figure 1.1: A Test of the Floating – Point Operations