

FORTRAN 90 **AND** **ENGINEERING** **COMPUTATION**

William Schick

Gordon Silverman

FORTAN 90 AND ENGINEERING COMPUTATION

William Schick

Fairleigh Dickinson University, Professor Emeritus

Gordon Silverman

Manhattan College



JOHN WILEY & SONS, INC.

New York

Chichester

Brisbane

Toronto

Singapore

Cover Art by John Jinks

Acquisitions Editor	Charity Robey
Marketing Manager	Susan Elbe
Senior Production Editor	Savoula Amanatidis
Design Coordinator	Laura Nicholls
Text Design	Lynn Rogan
Cover Design	Meryl Levavi/Levavi & Levavi
Manufacturing Manager	Susan Stetzer
Illustration Coordinator	Jaime Perea

This book was typeset in Times Roman by General Graphic Services, and printed and bound by R.R. Donnelley in Willard, Ohio. The cover was printed by Lehigh Press, Inc.

Recognizing the importance of preserving what has been written, it is a policy of John Wiley & Sons, Inc. to have books of enduring value published in the United States printed on acid-free paper, and we exert our best efforts to that end.

Copyright © 1995, by John Wiley & Sons, Inc.

All rights reserved. Published simultaneously in Canada.

Reproduction or translation of any part of this work beyond that permitted by Sections 107 and 108 of the 1976 United States Copyright Act without the permission of the copyright owner is unlawful. Requests for permission or further information should be addressed to the Permissions Department, John Wiley & Sons, Inc.

Library of Congress Cataloging in Publication Data

Schick, William.

Fortran 90 and engineering computation / William Schick, Gordon Silverman.

p. cm.

Includes index.

ISBN 0-471-58512-2 (paper)

1. FORTRAN 90 (Computer program language) I. Silverman, Gordon.

II. Title.

QA76.73.F25S333 1994

005.13'3—dc20

94-30554

CIP

Printed in the United States of America.

10 9 8 7 6 5 4 3 2

FORTRAN 90 AND ENGINEERING COMPUTATION

To our wives, Jessica and Roslyn

Preface



A key objective of this text is to prepare engineers and scientists to function in the highly computerized industrial and scientific environment of the modern world. Problem solving using the digital computer is one of the important skills that students should acquire to attain this objective.

FORTRAN is currently the most widely used high-level programming language taught in technical educational institutions, and it is likely to hold this position for some time. It leads the most frequently taught high-level languages by more than two to one. It is well suited for teaching problem solving. The continued development of new versions of FORTRAN, most recently FORTRAN 90, attests to the recognition of its continuing value in industry. This text is organized around the use of a digital computer for solving engineering problems.

For about two decades, students entering the engineering programs at the university at which the authors were affiliated were introduced to the solving of engineering problems using the FORTRAN language early in their education, usually in the freshman year. For quite a long period, this program of instruction was supported by the text *Fortran for Engineering* by Schick and Merz. Although the present text is a sequel to that successful book, it differs from it not only because FORTRAN 90 is used in this text, but also because of new educational criteria set by the engineering accrediting agency (Accreditation Board for Engineering and Technology, ABET) and changing educational goals of institutions. Although problem solving continues to be important, there now is increased emphasis on engineering design. A relatively early start on teaching engineering design is encouraged. Thus, many of the examples in this book concern topics that are related, directly or indirectly, to the tools of engineering design.

- The authors are aware that the majority of users of this text will be freshmen or sophomores. Therefore, examples that contain design-related or somewhat ad-

vanced mathematical topics are preceded, where feasible, by explanatory discussion. For example:

In Chapter 6, a discussion of electric filter design begins with an explanation of the meaning of the terms *filter*, *signal*, *attenuation*, and *analog*.

In Chapter 7, a program that calculates permutations and combinations is preceded by definitions of these terms and examples of calculation of combinations and of permutations.

In Chapter 9, prior to a program involving diffusion, the terms *independence* and *mutually exclusive* which are used in probability theory are discussed, and examples of their use are given.

- The abundance and variety of example programs are sufficient to permit instructors to choose those best suited for their course. Also, they are of varying difficulty. Thus, the book should be accessible to students with a broad range of mathematical preparation.

Because the users of this text are likely to represent many disciplines, problems have been chosen from several subject areas to facilitate the instructor's ability to select appropriate examples:

- Chemical engineering
- Chemistry
- Civil engineering
- Computer science
- Electrical engineering
- Environmental science
- Industrial engineering
- Mathematics
- Mechanical engineering
- Physics

Some of the examples involve topics *that are applicable to several disciplines*. One such topic is reliability engineering. Another topic is simulation.

- In Chapter 6 reliability engineering is defined and exemplified by a program.
- Simulation, in the context of product testing and model testing, is discussed in Chapter 9. Several examples are given.
- Another topic that applies to several disciplines is signal analysis. This topic is discussed in Chapter 9, and a sufficient number of examples are given to make the topic understandable to beginners.

The programs in this book are written in FORTRAN 90, and its predecessor, FORTRAN 77, is mentioned in only a few instances. The committee that wrote the FORTRAN 90 standard, aware that a very large number of programs had been written in FORTRAN 77, made provision for the fact that in some instances it was not feasible to scrap these programs. Accordingly, the FORTRAN 90 compiler used by the authors is in accordance with the International Standards Organization (ISO) standard, which is tolerant to the use of FORTRAN 77 constructs. This practice will likely continue, at least for a few years.

Although a number of excellent texts use the FORTRAN language, they are not solely concerned with FORTRAN 90. At this writing, this text is the first devoted to engi-

neering computation and design which is exclusively written using FORTRAN 90. All the programs have been tested according to the FORTRAN 90 standard and were compiled using the Numerical Algorithms Group (Nag) compiler (version 1.21).

The organization of the book is as follows:

- Chapters 1 and 2 provide the user with an introduction to organizing problems for computer solution and guide the reader through step-by-step compilation of a simple FORTRAN 90 program.
- Chapters 3 through 7 discuss the principles of FORTRAN 90. After principles are stated and explained, generally two or more programs are given using these principles.
- Chapter 8 discusses and illustrates program testing and debugging. A comprehensive discussion of such topics is rarely given in texts of this type.

After discussing the principles of FORTRAN 90, Chapters 9 and 10 present engineering applications of FORTRAN 90 programming. This material gives the text a reference quality; it shows the students additional use of FORTRAN 90, and it provides an opportunity for the instructor to present challenges to superior students. The applications reflect significant topics that an engineer or a scientist is likely to encounter during his or her professional activities. They are, however, presented at a level suited for undergraduate study. Chapters 9 and 10 deal with:

- Simulation and Monte Carlo methods
- Estimation exemplified by least squares fit to data
- Interpolation and estimation (Lagrange and Taylor)
- Integration and double integration by Simpson's rule
- Introduction to the discrete Fourier transform
- Introduction to computer graphics
- Introduction to queuing
- Introduction to databases
- Introduction to expert systems
- Introduction to solution of differential equations

To the Instructor:

This text may be used to support a one-semester course in engineering computation using FORTRAN 90. To that end, Chapters 1 through 8 would provide the basis for a one-semester course. If appropriate, material can be selected from Chapters 9 and 10.

Alternatively, Chapters 9 and 10 can be used as the nucleus of a second course introducing engineering design in a variety of disciplines.

A solutions manual will be provided for instructors who adopt the text.

The authors are grateful to a number of individuals: Dr. Howard Silver, professor and chair of electrical engineering at Fairleigh Dickinson University, provided invaluable suggestions. Dr. Terrence Akai, assistant dean for computing at the University of Notre Dame, Jerry R. Bayless, associate dean of engineering at the University of Missouri-Rolla, Dr. Joseph Saliba, of the University of Dayton, and Dr. David Edelson of Florida State University all contributed important comments.

Contents



Preface

I COMPUTER AND PROGRAMMING FUNDAMENTALS

1.1	History	1
1.2	Examples of Computer Usage	3
1.3	Computer Fundamentals	8
1.3.1	Hardware	9
1.3.2	Operating System Software	10
1.3.3	Files and File Systems	11
1.3.4	Some Limitations on Calculations	15
1.3.5	Number Systems Within the Computer	16
1.4	Organizing a Problem for Computer Solution	19
	<i>Problems</i>	25

2 FORTRAN 90 PROGRAMMING PROCEDURE 30

2.1	Steps in Program Development	30
2.2	Creating the Source Code	32
2.2.1	A Sample Problem	32
2.2.2	FORTRAN 90 Source Code	35
2.2.3	Using an Editor Program to Create the Source File	39
2.3	Translating the Program: Compiling and Linking	41
2.4	Running and Debugging a Program	44

2.5	The General Format of a FORTRAN 90 Program	46
	<i>Problems</i>	47

3 DATA TYPES AND OPERATIONS

49

3.1	Intrinsic Data Types	49
3.2	Implicit-Type Declaration	49
3.3	Explicit-Type Declaration	50
3.4	Integer and Real Data Types	50
3.4.1	Kind	51
3.5	Input/Output	52
3.6	Echoing Input Data	53
3.7	The Form of a FORTRAN Programs	53
3.7.1	Fixed Source Form	54
3.7.2	Free Source Form	55
3.7.4	Fixed/Free Source Form	56
3.8	Operations	57
3.8.1	Arithmetic Operations	57
3.8.2	Integer Arithmetic	57
3.8.3	Real Arithmetic	57
3.8.4	Exponentiation	58
3.8.5	Arithmetic Expressions	58
3.8.6	Precedence of Arithmetic Operations	58
3.8.7	Program CABLE1 (<i>Civil Engineering</i>)	59
3.8.8	Program Payments (<i>Engineering Economics</i>)	60
3.8.9	Vehicle Braking	61
3.8.10	Program Brake (<i>Civil Engineering</i>)	62
3.8.11	Stress in Highway Pavement	62
3.8.12	Program STRESS (<i>Civil Engineering</i>)	62
3.9	Character Data	63
3.9.1	Assigning a String to Character Variables	63
3.9.2	Concatenation	64
3.9.3	Copying Substrings	64
3.9.4	The Intrinsic Functions Len and Index	64
3.9.5	Example—Operations on Character Variables	65
3.9.6	Collating Sequence	66
3.9.7	Examples	67
3.9.8	Derived Data Type	68
3.10	Complex Data Type	70
3.11	Complex Numbers	70
3.11.1	Complex Constants and Variables	72
3.11.2	Complex Arithmetic Operations	72
3.11.3	Intrinsic Functions Involving Complex Numbers	73
3.12	Mixed Mode	75
3.13	Parameter Statement	75
3.14	Logical Data Type	76
3.15	Types of Logical Operators	76
3.15.1	Relational Operators	76
3.15.2	Relational Expressions	77
3.15.3	Logical Operations	77

3.16	Operator Precedence	77	
3.17	Subscripted Variables	80	
3.17.1	Denoting Subscripts	81	
3.17.2	Dimension Statement	81	
3.17.3	Examples	82	
	<i>Problems</i>	83	
4	HOW TO MAKE THE COMPUTER COMMUNICATE WITH YOU		93
4.1	An Example in Which a Format Specification Is Essential	94	
4.1.1	Using the Computer to Display a Table of Codes	95	
4.2	Data Formats	100	
4.2.1	Edit Descriptors	101	
4.2.2	Examples	102	
4.2.3	Format Descriptor Deviations	113	
4.3	Communicating with Files	113	
4.3.1	Open Communications	116	
4.3.2	File I/O	118	
4.3.3	The INQUIRE Statement	120	
4.3.4	File Positioning Statements	122	
4.3.5	Internal Files	123	
4.3.6	Close	123	
	<i>Problems</i>	124	
5	ITERATIVE PROCESSES		126
5.1	The IF Statement	126	
5.2	The GO TO Statement	127	
5.3	Iterative Processes	127	
5.4	DO Construct	129	
5.5	DO Loops with Loop Control Using a DO Variable	130	
5.6	Restrictions on the DO Statement	131	
5.7	Examples in Which the DO Loop is Used	132	
5.8	Nested DO Loops	148	
5.9	Implied DO Loops	151	
5.10	Multiplication of Matrices	153	
	<i>Problems</i>	157	
6	CONTROL PROCEDURES		164
6.1	The EXIT Statement	164	
6.2	The CYCLE Statement	164	
6.3	IF (<i>logical expression</i>) THEN	165	
6.3.1	DO Loops Without Loop Control	167	
6.3.2	DO WHILE	168	
6.4	The CASE Construct	192	
6.5	Modules, Part I	192	
	<i>Problems</i>	196	

7 AMPLIFYING THE POWER OF A PROGRAM BY USE OF SUBROUTINES 200

7.1	FORTRAN Intrinsic Functions and Statement Functions	201
7.2	The RETURN Statement	202
7.3	The FUNCTION Subprogram	202
7.3.1	The Gamma Function (<i>Mathematics</i>)	203
7.4	The SUBROUTINE Subprogram	205
7.4.1	The INTENT Statement	205
7.4.2	The RESULT Clause	206
7.4.3	Conversion of Octal and Hexadecimal Numbers to Decimal	207
7.4.4	Program CONVERT2 (<i>Computer Science</i>)	207
7.4.5	FIBONACCI Numbers	208
7.4.6	Program FIBONACCI (<i>Mathematics</i>)	209
7.4.7	Recursion	210
7.4.8	Program FIBONN2	210
7.5	The CALL Statement	211
7.6	The EXTERNAL Statement	211
7.7	The COMMON Statement	211
7.7.1	Blank COMMON	211
7.7.2	Labeled COMMON	213
7.7.3	Program COMMONS	214
7.8	MODULES, Part II	216
7.8.1	Program MODPI	218
7.8.2	Highway Traffic Flow	218
7.8.3	Program TRAFFIC (<i>Civil Engineering</i>)	219
7.8.4	Program BUTTERWORTH (<i>Electrical Engineering</i>)	219
7.9	Random Numbers	221
7.9.1	Program RANDOMB	222
7.9.2	Program SORTING	224
7.10	Matrix Exponentiation	226
7.10.1	Program MATRIXPOWER	226
7.11	The CONTAINS Statement	229
7.11.1	Program OPERATE	229
7.11.2	Program CONVERTB	230
7.12	Combinations and Permutations	231
7.12.1	Program SELECT (<i>Mathematics</i>)	233
7.13	Introduction to Filter Design (<i>Electrical Engineering</i>)	234
7.13.1	The DECIBEL	234
7.13.2	Chebyshev Filters	235
7.13.3	Checking the Designs	241
7.13.4	Use of INCLUDE Statement	247
	Problems	249

8 PROGRAM TESTING AND DEBUGGING 259

8.1	Principles	259
Example 8.1	Organization of a Transportation Planning and Research Program	260

Example 8.2	A Case in Which a Syntactical Error Is Revealed At Run Time	261
	Program RUN_TIME_SYNTAX_ERROR	
Example 8.3	A Numerical Bug	262
	Program NUMBER_BUG	
Example 8.4	A Logical Error in Control Structure	264
	Program ASCENDING_SORT	
Example 8.5	Interpreting Program Results in an “Intelligent” Manner	267
	Program CALCULATE_WEIGHT	
Example 8.6	Comparing Results to the “Real World”	269
	Program VISION_SYSTEM	
	Program STREET_TRAFFIC	
Example 8.7	Complete Testing/Debugging of a Program	275
	Program ROBOT	
<i>Problems</i>		284

9 NUMERICAL APPLICATIONS

289

Part I: Simulation

9.1	The Role of Simulation in Design	281
9.2	Histograms	290
9.3	Simulating the Performance of an Electronic Amplifier (<i>Electrical Engineering</i>)	294
9.4	One-Dimensional Random Walk: A Model of Diffusion (<i>Chemistry</i>)	296
9.5	Simulating the Distribution of the Total Resistance of Three Resistors in Series (<i>Electrical Engineering</i>)	301
9.6	Additional Examples of the Use of the Monte Carlo Method	303

Part II: Some Tools for Estimation

Part III: Tools for Interpolation and Estimation

9.7	Lagrange’s Interpolation Formula	310
9.8	Estimation by Taylor Series	316

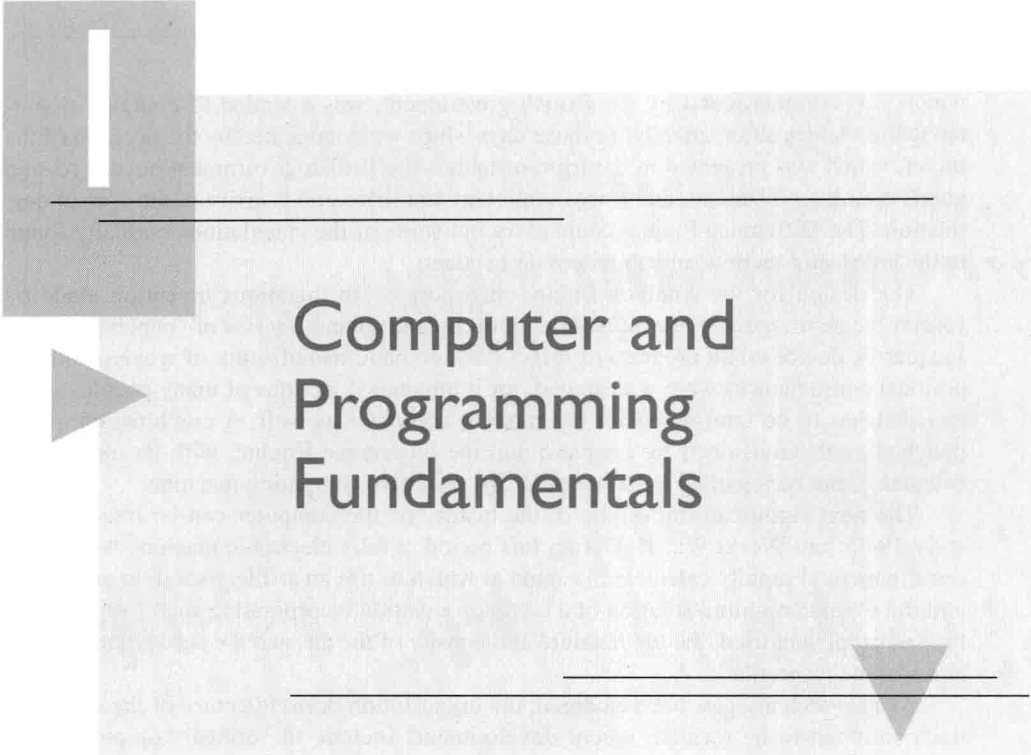
Part IV: Integration

9.9	Integration by Simpson’s Rule	317
9.10	Double Integration	320

Part V: Introduction to the Discrete Fourier Transform (Civil, Electrical, and Mechanical Engineering)

9.11	Analog and Digital Signals	323
9.12	Fourier Series	323
9.13	The Exponential Fourier Series	324
9.14	The Fourier Transform	325
9.15	The Discrete Fourier Transform (DFT)	326
9.16	Generating a Sampled Data Signal	329
9.17	A Subprogram that Computes the DFT and the IDFT	334

9.18	Nonperiodic Signals	336
	<i>Problems</i>	340
10	GRAPHING DATA AND DESIGN APPLICATIONS	346
10.1	Graphic Representation of Data	346
10.2	A Tool for Engineering Planning (Management/Industrial Engineering)	359
10.3	Databases for Maintaining Data	368
10.4	Expert Systems	376
10.5	Differential Equations	383
	<i>Problems</i>	390
	GLOSSARY	393
	APPENDIX A SUMMARY OF MS-DOS COMMANDS	397
	APPENDIX B SUMMARY OF MS-DOS EDIT KEY FUNCTIONS	401
	SOLUTIONS TO SELECTED PROBLEMS	403
	INDEX	421



Computer and Programming Fundamentals

The importance of the digital computer in our daily lives makes it one of the significant developments in the history of humankind. Increasingly, it has become a visible tool in science, engineering design and manufacturing, business, and leisure activities. One of its key attributes is its ability to repeatedly carry out tedious tasks rapidly and without error. Examples of such chores include mathematical calculations, control of other machines, and the management of large quantities of information. In short, it has the potential to improve the quality of our lives, relieve us of tiresome responsibilities, and expand our creative capacity. As with any tool, the user must learn how to employ it properly and explore some of its many applications, both of which are the aims of this text. *This introductory chapter will touch on the computer's history, describe how the machine works, and discuss how to go about organizing a problem for computer solution, including the introduction of the concept of a program.*

1.1 HISTORY

Two aspects of human behavior, commerce and war, have been important factors in the development of the computer. Although events that contributed to its evolution can be traced to ancient times, one of the first clear examples of a machine that resembles the modern computer may be found in the middle of the nineteenth century. Charles Babbage (1792–1871), with the help of Ada, the countess of Lovelace, produced the Difference Engine as well as the design for the Analytic Engine. Although the analytic Engine was never built, its design presaged the modern form of the computer. The Difference Engine,

which was commissioned by the British government, was intended to compute (lunar) navigation tables automatically. In those days, ships were navigated by the position of the moon, which was presented in the form of tables; the British government needed to ship goods over long distances, and it was important that these goods arrive at the correct destination. The Difference Engine could carry out some of the calculations normally found in the arithmetic sections of its modern descendants.

The design for the Analytic Engine incorporated an ingenious invention made by Joseph Jacquard, namely, the automatic loom. By employing a series of “punched cards,” Jacquard’s device could program or direct the automatic manufacture of woven cloth; its political consequences were widespread, for it threatened the jobs of many people, an effect that has to be considered for the modern computer as well. A combination of the punched cards envisioned by Jacquard and the Difference Engine, with its mechanical calculator, can be regarded as the world’s first primitive computing machine.

The next significant milestone in the history of the computer can be traced to the early 1940s and World War II. During this period, a fully electronic machine was developed that could rapidly calculate the angle at which to fire an artillery shell; to accurately aim the weapon required solution of a complex equation incorporating such factors as the type of propellant used, the temperature and density of the air, and the aerodynamic properties of the projectile.

As new technologies have emerged, the organization or *architecture* of the computer has continued to be refined; recent developments include the optical computer, very large-scale integrated circuits (VLSI) architecture, multiprogrammed and multiprocessor systems, structured languages, computer networks, and artificial intelligence systems—automata that make decisions in a way that seeks to imitate human thought processes.

The developers of computers were faced with problems related to the hardware of the machine. Early versions required a system of wires to direct the sequence of operations; the wires had to be changed for each new application, and this required considerable time as well as personnel. Thus, a parallel development in the history of the computer was devoted to ways of simplifying and speeding up this part of the computing operation. Gradually, *stored programs*, rather than wired or switching methods, were developed. Numbers stored within the memory controlled the sequence of operations carried out by the computer. Subsequent developments permitted the programmer to use symbols in place of the numbers. For example, the word (mnemonic) ADD could be used in place of the binary equivalent for the ADD operation (e.g., 010). The resultant sequence of instructions—the program—was translated into the numbers that were then stored within the memory. (The translation program is known as an *assembler*.) Ultimately, statements with characteristics much like those found in English (“English-like”) for each operation or command were used. These also required a translation program to convert such *high-level language* (HLL) statements into computer readable form. (The conversion programs are known as *compilers* or *interpreters*, depending on the translation method.)

One of the earliest programming languages of this type is FORTRAN whose name is a contraction of the words *formula translation*. In the near future we will celebrate the fiftieth anniversary of its origin. Its first formal appearance occurred in 1957, although it was under development by John Backus before that. Despite the development of numerous HLLs, FORTRAN retains viability and can be found in countless applications. Consider, for example, that its formal specification has undergone several revisions; this text will employ its most recent formulation, *FORTRAN 90*, as the vehicle for explaining how scientists and engineers use the computer as a powerful tool in their work.

I.2 EXAMPLES OF COMPUTER USAGE

In this section we present several examples that demonstrate how the computer can be a powerful tool for solving scientific and engineering problems.

EXAMPLE I.1 Automated Analysis of Chemical Substances



A robotic arm, coupled to a computer that controls its actions, can be used to completely automate the analysis of chemical substances. (This idea can be extended to examples such as process control or manufacturing environments that may be hazardous to human health.) A sketch of the complete system is shown in Fig. 1.1. A key element of the design is a robotic, articulated arm.

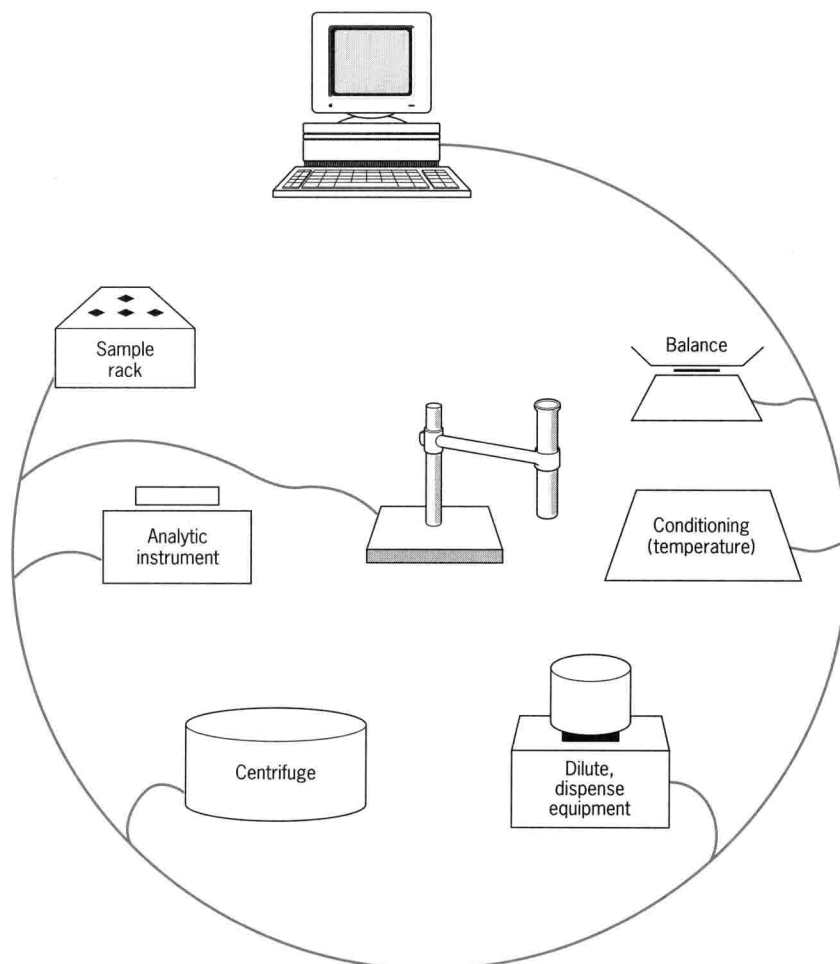


FIGURE I.1 A computer-controlled automated substance analysis system using a robotic arm.