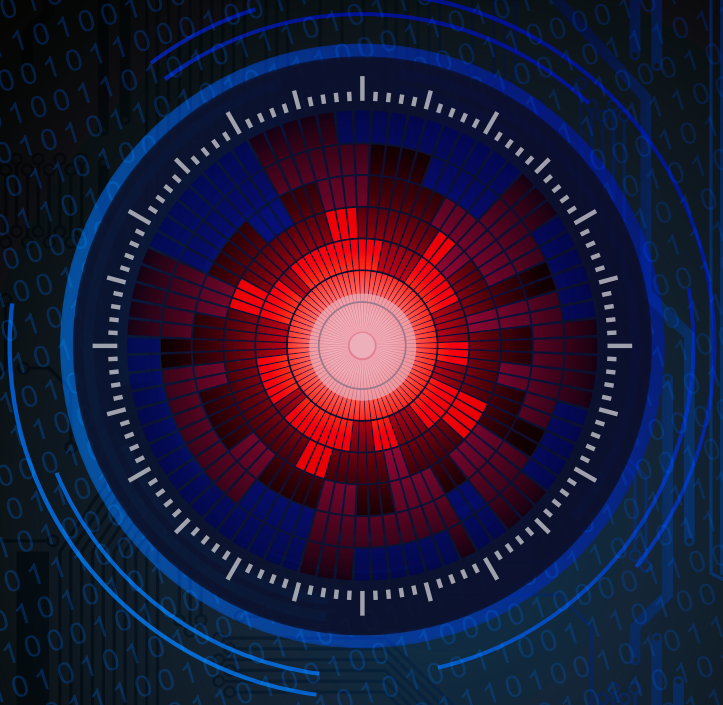
Computational and Physical Processes in Mechanics and Thermal Sciences

COMPUTER METHODS FOR ENGINEERING WITH MATLAB® APPLICATIONS

SECOND EDITION

YOGESH JALURIA



CRC Press
Taylor & Francis Group

Computational and Physical Processes
in Mechanics and Thermal Sciences

COMPUTER METHODS FOR ENGINEERING WITH MATLAB® APPLICATIONS

SECOND EDITION

Series in Computational and Physical Processes in Mechanics and Thermal Sciences

A Series of Reference Books and Textbooks

Series Editors

W. J. Minkowycz

Mechanical and Industrial Engineering
University of Illinois at Chicago
Chicago, Illinois

E. M. Sparrow

Mechanical Engineering
University of Minnesota, Twin Cities
Minneapolis, Minnesota

Computer Methods for Engineering with MATLAB® Applications, Second Edition, *Yogesh Jaluria*

Numerical Heat Transfer and Fluid Flow, *Suhas V. Patankar*

Heat Conduction Using Green's Functions, Second Edition, *Kevin D. Cole, James V. Beck, A. Haji-Sheikh, and Bahman Litkouhi*

Numerical Heat Transfer, *T.M. Shih*

Finite Element Analysis in Heat Transfer, *Gianni Comini, Stefano Del Guidice, and Carlo Nonino*

Computer Methods for Engineers, *Yogesh Jaluria*

Computational Fluid Mechanics and Heat Transfer, Second Edition, *John C. Tannehill, Dale A. Anderson, and Richard H. Pletcher*

Computational Grids, *Graham F. Casey*

Modern Computational Methods, *Herbert A. Konig*

The Intermediate Finite Element Method: Fluid Flow and Heat Transfer Applications, *Juan C. Henrich and Darrell W. Pepper*

Modeling and Dynamics of Regenerative Heat Transfer, *A. John Willmott*

Computational Heat Transfer, Second Edition, *Yogesh Jaluria and Kenneth Torrance*

The Finite Element Method: Basic Concepts and Applications, Second Edition, *Darrell W. Pepper and Juan C. Heinrich*

Computational Methods in Heat and Mass Transfer, *Pradip Majumdar*

Computational and Physical Processes
in Mechanics and Thermal Sciences

COMPUTER METHODS FOR ENGINEERING WITH MATLAB® APPLICATIONS

SECOND EDITION

YOGESH JALURIA

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20121009

International Standard Book Number-13: 978-1-4398-9727-0 (eBook - PDF)

Contents

Preface to the Second Edition

Computer methods continue to be critical in the analysis, simulation, design and optimization of engineering processes and systems. Computational approaches are needed to solve the complex mathematical equations that typically arise in engineering problems, for correlating experimental data, and for obtaining numerical results that are used for improving existing processes and developing new ones. The second edition follows the basic ideas, discussions, approaches, and presentation employed in the first edition. The focus is clearly on engineering processes and systems and on the equations that characterize and describe these. Computer methods that are employed to solve these equations and the nature and validity of the numerical results obtained are discussed for a variety of problems. The main thrust is on the discussion of the various numerical methods that are available for a given problem, on the presentation of the basic aspects of the methods, discussing their applicability, efficiency and behavior, and then applying these to typical problems chosen from various engineering disciplines.

Besides discussing the solution of different types of mathematical equations, a large number of engineering examples and problems were chosen to present the choice of the method, development of the numerical algorithm and use of the computer to solve the problem. A systematic approach is followed to obtain physically realistic, valid and accurate results through numerical modeling. Examples from many different engineering areas are employed to explain the various elements involved in the numerical solution and to make the presentation relevant and interesting. Similarly, a large number of solved examples and exercises are included to supplement the discussion and to illustrate the ideas and methods presented in the text. The book continues the thinking that the basic purpose of the computational approach is to provide physical insight and to obtain inputs for analysis and design of practical systems. Thus, the solution methodology is linked to both the computer and to the fundamental nature of the problem to allow the student to appreciate the basic aspects of the numerical approach.

The book is appropriate as a textbook for engineering undergraduate courses on computer methods at the sophomore or junior levels. Because the background of students at the sophomore level may not be sufficient for some of the topics covered, such as partial differential equations, a few such topics may be avoided for sophomore students and may be included in the junior or senior courses. The book is also appropriate as a reference on computational methods for various other basic and applied undergraduate courses in mechanical engineering and in other engineering disciplines. The book will also be useful as a reference for engineers who are interested in using computer methods for analysis, simulation, design, or data analysis.

The second edition is a substantially revised and updated version of the earlier book. Recent advances in available computational facilities, both in software and in hardware, are included. In several places, the presentation has been simplified and clarified to make it easier to follow. Certainly, the main difference from the first edition is the extensive use of MATLAB®, instead of a high-level programming language like Fortran, for numerical modeling. This is done in view of the current trend in engineering education where MATLAB has emerged as the dominant environment for the numerical solution of basic mathematical equations. Much of the discussion on computer solution is thus directed at MATLAB and a large number of MATLAB commands and programs are given in the text, as well as in the Appendix, in order to facilitate the presentation as well as to provide ready access to MATLAB programs for solving exercises given in the text and other similar problems. In many cases, the programs are focused on the example or problem being considered, in order to encourage the readers to develop their own computer programs for specific problems. However, the programs can be easily modified for different circumstances and parameters. Available MATLAB functions and commands are frequently employed to generate results that can be used for comparisons with the results obtained from more detailed and versatile programs. Fortran has not been abandoned because of its continued importance in engineering and the existence of substantial software in Fortran for many complex problems. Several important Fortran programs are included in the Appendix to illustrate the ease with which one could go from one computational environment or language to another and to allow those interested in Fortran to use these for their specific problems. Additional exercises and examples are included in all the chapters. References have been added on new topics included in the book and references in the first edition have been updated.

The methods, discussions, and computer programs presented in this textbook are the result of many years of teaching computer methods to engineering undergraduate students, in required as well as elective courses. The inputs from many colleagues and graduate students, as well as undergraduate students, who took the courses from me, have been valuable in selecting the topics, the depth of coverage, the computer programs presented here and many other aspects related to computer methods for solving engineering problems. Inputs from those who have used the first edition in their courses, particularly from Professor Wally Minkowycz, have been particularly valuable. The support and assistance provided by the editorial staff of Taylor & Francis, particularly by Jessica Vakili and Jonathan Plant, have been valuable in the development of the second edition.

The book would never have been completed without the strong support and encouragement of my wife, Anuradha. Our children, Ankur, Aseem, and Pratik, as well as Pratik's wife Leslie and son Vyan, have also been sources of inspiration and encouragement for me and have contributed in their own way to my efforts over the years. I greatly appreciate the patience and understanding of my family that made it possible for me to spend extensive periods of time on the book.

MATLAB® is a registered trademark of The MathWorks, Inc. For product information, please contact:

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098 USA
Tel: 508 647 7000
Fax: 508-647-7001
E-mail: info@mathworks.com
Web: www.mathworks.com

Preface to the First Edition

The use of computational methods in the analysis and simulation of engineering processes and systems has grown tremendously over recent years. Increasing national and international competition has made it imperative to improve existing facilities and to develop new ones for a wide variety of applications. Because of the constraints imposed on detailed experimentation needed for design and optimization of systems, due to excessive time, manpower, and financial requirements, computer simulation is extensively employed to obtain the desired information. Analytical methods are generally very restrictive in their applicability to practical problems, and numerical methods are usually necessary. In addition to the growing need for numerical solutions to engineering problems, we have also seen substantial improvements in the computational facilities available, both in software and in hardware, over the last decade. All of these changes have made it more important than ever for engineers and engineering students to develop expertise in numerical methods and to use them for solving problems of practical interest.

In recognition of the growing importance of computer methods in engineering, many courses in engineering curricula now include the numerical solution of engineering problems on the basis of numerical analysis taught earlier at the sophomore or junior level. Generally, engineering students are first exposed to the computational procedure through a course on programming, frequently employing Fortran as the programming language. Numerical methods are then taught at a later stage to introduce the basic concepts of numerical analysis and to allow the students to numerically solve important mathematical problems such as integration, matrix inversion, root solving, and solution of differential equations. However, since the basic purpose of the computational approach is to provide physical insight and to obtain valuable information for the analysis and design of practical systems, such courses have been integrated into the engineering curricula at most universities. This implies that the solution methodology is coupled with the computer on one hand and with the physical or chemical nature of the problem on the other. The numerical procedure, as well as the results, are considered in terms of actual problems to permit the student to develop a physical feel for the numerical approach to engineering problems.

Traditionally, numerical analysis courses have been mathematically oriented. Although this orientation brings in some very important and fundamental aspects of numerical analysis, it lacks in the application of the methodology to actual problems. It is extremely important to integrate the basic understanding of the methods with their actual use on the computer. Unless the students learn to choose and implement a computational scheme on the computers available, they will not develop a satisfactory appreciation or understanding of the numerical technique. In addition, recent advances in computational facilities, such as structured programming, interactive computer usage, and graphics output, must be introduced so that the most efficient procedure is

adopted for a given problem. The incorporation of problems derived from various engineering disciplines aids in this learning process and also makes it interesting and enjoyable. In addition, it reinforces the important point that the physical or chemical background of the given problem forms an important element in the selection of the method and in the evaluation of the accuracy of the results obtained.

This book, directed at computer methods for engineering, integrates the treatment of numerical analysis with the physical background of the problems being solved and with the implementation of the methods on available computers, employing several recent advances in this field. Although a large number of books are available on numerical analysis, not many satisfactorily discuss the implementation of the methodology on the computer, and even fewer discuss the implications of the physical nature of the problem in the numerical solution. This book recognizes the need for a satisfactory incorporation of these concepts into the mathematical treatment of numerical analysis. It couples numerical methods for a variety of mathematical problems with the use of these methods for the solution of engineering problems on the computer.

Numerical methods for important mathematical operations, such as integration, differentiation, root solving, and solution of algebraic systems, are discussed in detail. The solution of differential equations, both ordinary and partial, is presented. Curve fitting, which is an important consideration in engineering problems, is also discussed. A large number of problems from basic sciences and various engineering disciplines are chosen to illustrate the use of these methods. The problems chosen are relatively simple so that they can easily be understood by students at the sophomore/junior level. However, in several cases, the basic background of the problem is outlined so as to bring the important points into proper focus. The importance of the physical or chemical background of the problem in the selection of the method, the choice of numerical parameters, the estimation of the accuracy of the results, and the overall validity of the results is discussed. The book mainly uses Fortran 77 to demonstrate the implementation of the numerical methods on the computer, because of the overwhelming importance of this language in engineering applications. However, a few programs in *Basic* are also given to bring out the similarities between the two languages and the ease with which one may switch from one to the other. A discussion of other languages and important aspects in computational procedure is included. A large number of examples, with the corresponding programs, are given. The programs are written specifically for these examples, so that the students must develop their own programs for the large number of problems given at the end of the chapters. Several important features that are currently employed in computational procedure are demonstrated in these programs. Recent trends in this area are outlined, and their significance for engineering applications is discussed. The students are strongly encouraged in every way to develop their own computer programs, since this is an essential ingredient for learning computer methods.

Most of the material covered in this book has been employed by the author for courses at the sophomore and junior levels. Since the background of students at the sophomore level may not be sufficient for some of the topics covered, such as partial differential equations, this particular topic and a few sections marked with an asterisk may be avoided by sophomore students. The book can also be used at the senior level, if such a course is included in the curriculum at this level. The material included is

quite adequate for a one-semester course. However, the best time to teach this course is probably at the junior level, so that the students can fully understand the material and then use it in courses taught at higher levels. The book is also appropriate for professional engineers in various disciplines and as a reference for courses that employ computational methods as an important element in the presentation. The book considers problems from diverse engineering applications, and the treatment is at a level appropriate for engineering students of all disciplines.

I owe tremendous gratitude to several colleagues and students who have contributed to my understanding and enjoyment of computational methods for engineering applications. First, I would like to thank Dr. Frank Kreith, who suggested that I write this book and contributed several very valuable suggestions on the presentation. I would also like to acknowledge several stimulating and interesting discussions on the subject with Professors Dave Briggs and Abdel Zebib. Professor Samuel Temkin provided me with tremendous support and encouragement. Dr. M. V. Karwe helped with the numerical solution of some problems. Also of considerable value was the support provided by the staff of Allyn and Bacon, Inc., particularly by Ray Short. The manuscript and its several versions were typed with great patience and competence by Diane Belford and Lynn Ruggiero.

I would like to dedicate this book to my parents, who have always encouraged, supported, and inspired me to strive for the best I could achieve. The greatest contributions to this effort have been the encouragement and support of my wife, Anuradha, and of our children, Pratik, Aseem, and Ankur, who had to bear long hours that kept me away, working on this book, with patience and understanding.

The author extends special thanks to the following reviewers whose contributions have enriched the text: Professor Clayton Crowe, Washington State University; Professor Rodney W. Douglass, University of Nebraska; Professor S. V. Patankar, University of Minnesota; Dr. James F. Welty, U.S. Department of Energy.

Author

Yogesh Jaluria is currently a board of governors professor at Rutgers, the State University of New Jersey, New Brunswick, New Jersey, and the chairman of the Mechanical and Aerospace Engineering Department. He received his BS from the Indian Institute of Technology (IIT), Delhi, India, and his MS and PhD in mechanical engineering from Cornell University. He worked at Bell Labs and at IIT, Kanpur, before joining Rutgers University in 1980.

Professor Jaluria has contributed more than 450 technical articles, including over 170 in archival journals and 16 chapters in books. He has two patents in materials processing and is the author/coauthor of seven books. He is also editor/coeditor of 13 conference proceedings, two books, and three special issues of archival journals. Professor Jaluria received the prestigious 2007 Kern Award from the American Society of Chemical Engineers (AIChE), the 2003 Robert Henry Thurston Lecture Award from the American Society of Mechanical Engineers (ASME), and the 2002 Max Jakob Memorial Award, the highest international recognition for eminent achievement in the field of heat transfer, from ASME and the AIChE.

In 2001, he was named a board of governors professor of mechanical and aerospace engineering at Rutgers University. He received the 2000 Freeman Scholar Award for work on fluid flow in materials processing, the 1999 Worcester Reed Warner Medal for extensive contributions to the engineering literature, and the 1995 Heat Transfer Memorial Award for significant research contributions to the science of heat transfer, all from ASME. He served as the chair of the Heat Transfer Division of ASME during 2002–2003. He was the editor of the ASME *Journal of Heat Transfer*, the preeminent publication in this field, during 2005–2010, and is on the editorial boards of several international journals.

1 Introduction

1.1 INTRODUCTORY REMARKS

Over the past three decades, there has been a tremendous increase in the use of computers for engineering problems. This increase has been mainly due to the growing need to optimize systems and processes in order to raise productivity and reduce costs. With increasing worldwide competition, it has become necessary to modernize existing engineering facilities and develop new ones through analysis and design. Consequently, we have seen a considerable improvement in engineering systems, particularly those related to electronic circuitry, materials processing, biotechnology, transportation, and energy generation. The concern with safety, including homeland security, and with our environment has also led to detailed investigations of existing engineering processes and to substantial improvements in many of these to reduce the impact on our environment and to make their use safer.

Because of the complexities involved in most engineering applications, analytical methods based on mathematical techniques are usually unable to provide a solution to the equations that characterize their behavior, and computational methods are needed to obtain quantitative information on physical quantities of interest. Even though analytical solutions are obtained in a few simple cases, the form of the solution itself may be quite involved, since the results are frequently expressed as a series or in terms of integrals and complex functions. In such cases, the computer is needed to extract the desired information from the analytical solution. Also, the problem may have to be solved several times with different sets of data, making it advantageous to use the computer rather than analytical methods.

There has also been a phenomenal increase in the availability of computers over the recent years. With the advent of microcomputers, such as personal computers (PCs), computational facilities have become widely available. The computational power available has also increased dramatically in individual, single-processor, machines, or *serial* computers, as well as in linked multiple machines or processors that result in a *parallel* computing cluster. There is every indication that these trends will continue, making computers even more accessible and powerful. Although most practical engineering problems still require larger and faster computers (such as supercomputers, minicomputers, or parallel computing systems), microcomputers do allow the solution of many common problems and are also useful in testing numerical procedures that may subsequently be employed on larger or parallel machines. The availability of a wide variety of microprocessors has also substantially affected the control and operation of systems through automation and expanded the reach of computational software.

Along with the revolution in computer hardware, there has inevitably been one in the available software as well, making the use of computers for scientific and engineering

problems easier than ever. Thus, for a wide range of problems, the programs available in the computer library, commercially available software, or user-friendly computational environment may be used effectively. However, it is generally necessary to understand the basic techniques involved in order to modify the program for satisfactory application to a given problem. In industrial systems, the use of commercially available programs is particularly important, since the processes are often quite involved and interest lies in obtaining the needed information as rapidly as possible. For simpler problems, such as those related to individual physical and chemical processes that constitute the overall system, it is often easier and more desirable to personally write the computer program or use an appropriate computational environment, rather than use a commercially available code written specifically for a given problem. Therefore, it is important to understand computational methods relevant to engineering applications and to use them in physical problems that are of interest to various disciplines.

Computer-aided design, simulation-based design and optimization, and computer-aided manufacturing are important areas that have grown substantially in the very recent past. These areas have arisen from the need to optimize on the one hand and the growing availability of the computers on the other. They are interdisciplinary in nature, particularly simulation-based design, which is of interest in such diverse fields as electronic systems and structural design. The basic approach in this case is to numerically solve the governing equations, choose physical parameters to simulate existing processes and systems, and finally vary these parameters to optimize the design for existing and future systems. Several other similar applications of computer methods have arisen in recent years, making it imperative to link the computational approach to the physical or chemical aspects of the problem under consideration.

In view of the growth of computer usage and availability in the recent years, it is surprising that much of the mathematical background underlying numerical analysis and computer logic has been available for several centuries. Binary logic operations, which use 2 as the base, instead of 10 employed in the decimal system, and which form the basis for most present digital computing, have been known and used for quite some time. Francis Bacon used binary codes in the early seventeenth century to transmit secret messages. In 1804, Joseph Marie Jacquard used punched cards with binary codes and logic to operate looms. A mathematical theory for binary logic was developed by George Boole during the nineteenth century. Similarly, adding machines and mechanical calculators were developed centuries ago, such as the one developed by Blaise Pascal in the seventeenth century. Charles Babbage designed the first automatic digital computer in 1833, with several features similar to those of modern computers. However, this machine was never constructed.

Modern digital computers were developed largely after World War II. A high-speed electronic digital computer was developed during the period from 1945 to 1952 under the direction of John von Neumann at the Institute for Advanced Study in Princeton, New Jersey. Binary digits, which can be represented by the opening or closing of a switch, were stored electrostatically in cathode-ray tubes. Several thousand vacuum tubes were used for computer memory, which had to be again stored about a thousand times per second due to the decay of electrostatic charge. Much of the logic behind this machine has persisted in modern computers. The major advancement has been in

electronic hardware, particularly in the development of transistors, integrated circuits, microelectronics, and now nanoscale devices and systems. As a result, there has been a considerable reduction in size and cost of electronic digital computers and also a substantial increase in their capability, speed, and reliability. The availability of PCs has brought computational techniques within easy access for a wide variety of problems, both for students and for professional engineers. Therefore, the coming years may be expected to improve the available computational facilities even further through the advancement in both computer software and hardware. It is also evident that PCs, with an interface with larger machines or with other machines in a parallel computing environment for more complicated problems, will continue to grow in availability and usage. Thus, it is important to learn the computational techniques relevant to engineering problems on the basis of the currently available computational facilities, while considering expected future trends as well.

Several important and useful features have been incorporated in the modern computer systems. Among the most important of these is an *interactive* use of the computer, rather than the previously common *batch* operation mode. Frequently, an interpretive compiler is used so that each program statement entered into the computer is screened for syntax errors and a message issued if any error has been committed. The interactive mode allows one to enter variables and make changes in the program, as the need arises after each run of the program. The execution may also be stopped to make modifications and then continued. Therefore, the interactive mode is very well suited for the initial stages of program development, when the testing and debugging of the program is being done, and for obtaining the trends for a wide range of input parameters. For instance, if the values of x at which a nonlinear equation $f(x) = 0$ is satisfied (known as roots of the equation) are to be determined, the interactive mode may be used very effectively to obtain the general behavior of $f(x)$ over the range of interest in x. Various values of x may be entered and the corresponding values of $f(x)$ obtained. A graph of $f(x)$ versus x may easily be plotted using available software. The information obtained may then be used to select the method for finding the roots and also to obtain suitable initial guesses for the roots. Figure 1.1 shows a few examples where the plot of $f(x)$ versus x would be particularly useful in root finding.

The batch operation mode involves feeding the complete job into the computer and then running it with no interaction with the operator until the job is executed. This mode is appropriate for obtaining the numerical results for different parametric values after the program has been developed and debugged, particularly for large programs. Other important features available with present computer systems are graphics facilities, which plot the computed results, and interfacing between various computers, which allows program development to be carried out on small computers in the interactive mode. Once the program has been completed, debugged, and tested, the numerical code may be transmitted to a larger computer or to a parallel computer system, which would generally be more efficient for computing and will have greater storage capability, and run in the batch mode to obtain the desired computed results. Of course, with the increasing computational power and storage capacity of individual machines and workstations, code developments, as well as extensive computational runs, are often carried out on the same unit.

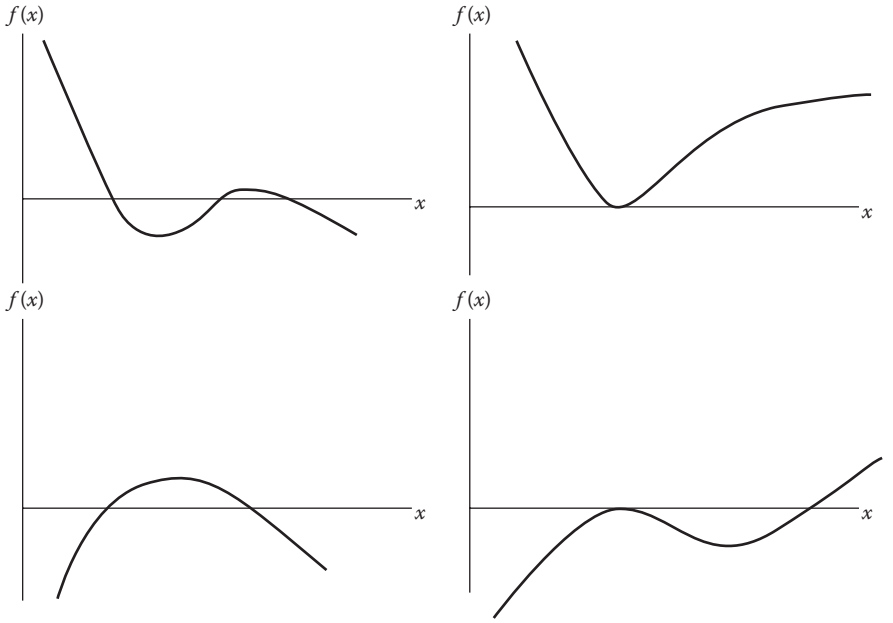


FIGURE 1.1 Some examples of the plotting of the function $f(x)$ versus x to determine the approximate values of the roots of the equation $f(x) = 0$.

1.2 NUMERICAL SOLUTION

The development of a computational procedure, or *algorithm*, to solve a given problem requires knowledge of both the available numerical methods and the methodology to interface with the computer. Since several methods are generally available for a given application, it is important to understand the applicability and advantages of each method compared to those of the other methods. For instance, a system of linear equations may be solved by a wide variety of methods, including direct methods, which give a solution in a definite number of steps, and iterative methods, which involve a repeated solution of the equations until a chosen convergence criterion is satisfied. The choice of the method for a given problem depends mainly on the nature and number of the equations. Direct methods are suitable for smaller systems and iterative methods for large sets of equations. Also, if the same system of equations must be solved several times with different constants on the right-hand side of the equality sign, methods based on matrix inversion are often preferable since the different solutions may be obtained easily once the coefficient matrix has been inverted. Similarly, in curve fitting, the method to be adopted is strongly dependent on the nature and form of the given data. If the data have been provided at uniform intervals of the independent variable, certain specialized methods may be used, taking advantage of the uniform distribution of data.

Sometimes, several methods are applicable for a given problem, and the selection of the method becomes a matter of personal choice. The previous experience with a

particular method may be an important consideration in its selection. Also, the availability of certain programs in the computer library may make it advantageous to choose a given method. Many specialized methods have been developed for specific applications. Such methods are often limited in their applicability, although they may be the most efficient ones when applied to the problem for which they are particularly suited. For instance, certain methods for finding the roots of an algebraic equation are applicable only to polynomial equations and are popular choices for this application. They cannot be used for other types of algebraic equations, say, transcendental equations that involve transcendental functions such as exponential, logarithm, and trigonometric functions. Similarly, direct methods for solving systems of equations apply only for linear equations. Iterative methods are generally necessary for a system of nonlinear equations.

It is evident from the preceding discussion that the selection of the most appropriate numerical method for a given problem is an important consideration and is generally based on the nature of the problem. Once the method has been selected, one proceeds to implement it on the computer. The program is written in a programming language or in the computational environment available on the computer system to be employed. Although Fortran, with its many versions like Fortran 77, Fortran 90, Fortran 2003, and Fortran 2008, has been used extensively in engineering applications on most minicomputers and mainframe systems, *Basic*, *C*, *C++*, and other languages developed in recent years have often been used on PCs. MATLAB® is probably the most commonly used computational software being used today on both PCs and servers to solve mathematical problems that arise in engineering and scientific applications. Most of the numerical solutions discussed in this book, therefore, employ MATLAB.

The computer program written in the chosen programming language is converted into machine language by the computer. This process, known as compilation of the program, is achieved by using the relevant software, termed the *compiler*, available on the computer. An *operating system* is used for the control of the program and the computer resources. The editing of the program, for making changes and corrections, is done with the help of the editing system available on the given computer. The compilation, editing, and execution of the program are governed by the operating system of the computer and therefore vary with the machine. Similarly, the *job control language*, which interfaces the programmer with the computer, depends on the computer system. For those who may not be familiar with the terms mentioned here, Chapter 2 outlines the basic features of a computer system.

The interpretation of the numerical results obtained is also an important consideration, since it relates to the accuracy and the correctness of the numerical solution. The computational scheme may be employed to yield results for a wide range of input variables, so that the results may be considered in terms of the physical or chemical nature of the problem being investigated. If possible, a comparison is made with available analytical results in order to determine the accuracy of the computed results. The *verification* and *validation* of the numerical scheme involve ensuring that the results obtained are accurate and valid. These are particularly important if a commercially available computer program or one available in the public domain is being employed to solve a given problem. It is also important to

determine the range of governing parameters over which the scheme can be used to yield accurate numerical results. These considerations are discussed in the following sections. Once the accuracy and validity of the results have been verified, the desired results may be obtained in a tabulated or graphical form.

1.3 IMPORTANCE OF ANALYTICAL RESULTS

As mentioned earlier, the equations that arise in most engineering problems are too complicated to be solved analytically, and computational techniques must be used to obtain the numerical values needed. Analytical solutions are often obtained only in very simplified circumstances. Also, as indicated before, analytical results are frequently given in terms of convergent series, integrals, and complicated functions, such as transcendental functions, Bessel functions, and so on. In engineering, we are largely interested in numerical values corresponding to given input data, and the computer is frequently needed to obtain the desired numerical information from a given analytical solution. However, analytical results, whenever available, are extremely important in evaluating the accuracy of the numerical scheme and in validating the model. Similarly, analytical results may be used to study the convergence characteristics of the numerical method and to decide if the correct solution has been obtained.

As an example, let us consider the solution of the differential equation that governs the variation, with time t, of the charge q of a capacitor in an electrical circuit that also contains a voltage source and a resistance. If the initial charge in the capacitor is Q and the voltage input, resistance, and capacitance are denoted by E, R, and C, respectively, the governing equation is obtained as follows (Young et al., 2000):

$$R\frac{\mathrm{d}q}{\mathrm{d}t} + \frac{q}{C} = E \tag{1.1}$$

If R, C, and E are constants, the preceding equation may be solved mathematically to obtain

$$q = Q\mathrm{e}^{-t/RC} + EC(1 - \mathrm{e}^{-t/RC}) = EC + (Q - EC)\mathrm{e}^{-t/RC} \tag{1.2}$$

The physical problem and the analytical solution are sketched in Figure 1.2. The charge q decreases from the initial value of Q to a steady-state value of EC, if $EC < Q$. Similarly, q increases to a steady charge of EC, if $EC > Q$.

Several other physical problems are governed by equations similar to Equation 1.1. The temperature T(t) of a small, heated metal block being cooled by a stream of air, the moisture content of a wet body drying in air, and the pressure of gas in a container with an opening are often governed by equations of the same form as Equation 1.1. However, in actual practice, the parameters, such as R, C, and E, may be the nonlinear functions of the charge or voltage and may, in some cases, also vary independently with time. For instance, nonlinear conductors, such as vacuum tubes, do not obey Ohm's law, and heat and mass transfer processes operating at the surface of a given

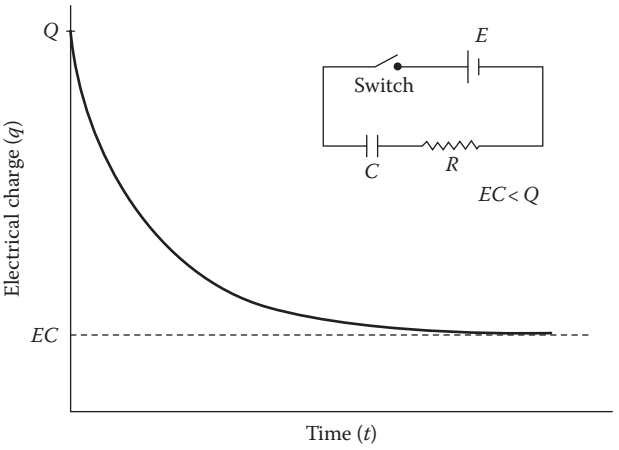


FIGURE 1.2 Variation with time t of the charge q in a capacitor, which is originally at charge Q, due to the closing of the switch in the electrical circuit shown.

object generally depend on the temperature, concentration, and pressure, making the differential equation nonlinear. The governing equation may, in general, be written as $d\phi/dt = -H(\phi, t)\phi + B$, where ϕ is the dependent variable, $H(\phi, t)$ is a functional parameter, and B is a constant. If q is replaced by ϕ in Equation 1.1, then $H(\phi, t) = 1/RC$ and $B = E/R$. This equation is linear in ϕ, or q, since H and B are constants, resulting in only the first power of ϕ to appear in the equation.

If H is not a constant but a function of ϕ as $H(\phi, t)$, an analytical solution is often not obtained because of the nonlinear expression $-H(\phi, t)\ \phi$ that arises on the right-hand side of the differential equation. In such circumstances, a numerical solution of the differential equation may be obtained by choosing a time step Δt and advancing time to compute ϕ as a function of time, starting with the given initial condition. This computation is done until an insignificant change is observed in $\phi(t)$ from one time level to the next, thereby indicating that the temperature has reached steady state, given by $d\phi/dt = 0$. However, since an analytical solution is available for the simplified circumstance of Equation 1.1, the numerical scheme should first be used to solve the problem with H taken as a constant and the computed results compared with the analytical solution. This comparison will allow determination of the anticipated accuracy of the numerical results and will also check the correctness of the procedure. Such a comparison is particularly valuable in complicated problems where an error in the numerical scheme may go undetected. Fortunately, many physical and chemical problems can be formulated in terms of idealized circumstances, which lead to simplified equations that can be solved analytically. Chapter 8 discusses several methods for solving ordinary differential equations (ODEs) and demonstrates again the importance of available analytical results.

Similarly, in numerical differentiation and integration, the computational scheme may be tested by employing simple functions whose derivatives and integrals can be obtained analytically. In radiative heat transfer, for instance, integration over the

wavelength λ of the radiation is frequently needed to determine the total energy lost or gained, Q, per unit area, at a surface. The expression for Q is

$$Q = \int_0^\infty f(\lambda)\,d\lambda \tag{1.3}$$

where $f(\lambda)$ is known as the monochromatic emissive power and is often a fairly complicated function of the wavelength λ, generally obtained from a curve fit of experimental measurements. However, the radiation from a blackbody, which is an idealized circumstance, is well known in physics and is given by Planck's law, which expresses $f(\lambda)$ as

$$f(\lambda) = \frac{c_1}{\lambda^5\left[\exp(c_2/\lambda T) - 1\right]} \tag{1.4}$$

where T is the surface temperature on the Kelvin scale and c_1, c_2 are the known constants. Figure 1.3 shows the variation of $f(\lambda)$ with λ for the ideal surface of a

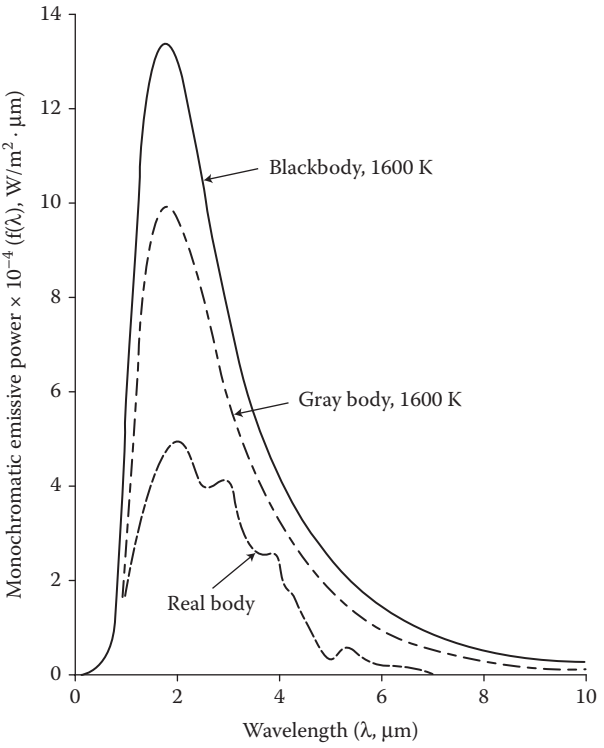


FIGURE 1.3 Variation of the emissive power $f(\lambda)$ with the wavelength λ for thermal radiation by a blackbody, a gray body, and a real surface.

blackbody, for a real or practical surface, and for a gray body for which $f(\lambda)$ is a constant fraction of that for a blackbody at all λ.

For a blackbody, the integral in Equation 1.3 has been evaluated analytically and is given by

$$Q = \sigma T^4 \tag{1.5}$$

where σ is known as the Stefan–Boltzmann constant and whose numerical value is given in the literature as 5.67×10^{-8} W/m^2 K^4. Therefore, the computational scheme developed for numerically determining Q for a wide variety of engineering surfaces, and thus different $f(\lambda)$, may first be applied to blackbody radiation and the results compared with the analytical solution given by Equation 1.5 to determine the accuracy and validity of the numerical method.

The numerical solution of large systems of linear or nonlinear equations is often needed in engineering problems. Since small sets of equations, typically three or four equations, can be solved analytically, the numerical procedure for solving systems of simultaneous algebraic equations may be employed for a small number of equations and the numerical results compared with the analytical values, to determine the accuracy and correctness of the numerical solution.

In numerical methods based on iteration, a convergence criterion ε is employed to decide when to terminate the iteration. Generally, the convergence criterion is applied to a physical variable in the problem, and computation is stopped when the change from one iteration to the next is less than the chosen value of ε. A relationship between ε and the accuracy of the numerical results may be obtained by a comparison of the computed values with the analytical solution that may be available for a simplified circumstance. This information can then be employed in the choice of the convergence criterion. If analytical results are not available, an extensive testing of the numerical procedure, over wide ranges of the initial guess, convergence criterion, and time step Δt, for example, in the problem given by Equation 1.1, must be carried out to ensure that the numerical results are essentially independent of the values chosen and that the desired accuracy level has been achieved. Figure 1.4 sketches typical computed iterative and converged solutions to the ODEs that govern a particular flow circumstance. The questions related to iterative convergence and to the choice of the numerical parameters, such as ε and Δt, are extremely important and are discussed in detail in Chapter 2.

1.4 PHYSICAL CONSIDERATIONS

The physical or basic considerations that give rise to a given mathematical expression or equation can often be used very effectively in selecting the numerical method, in choosing an acceptable solution from the several that may be obtained, and in testing the method for accuracy and correctness. In most engineering problems, the basic nature of the desired solution is known, along with the range in which it lies. Let us consider, for example, the free fall of a body of arbitrary shape in air. A terminal velocity is attained due to the balancing of the gravitational force by the frictional drag force (Halliday et al., 2004). Depending on the size and shape of the body, an

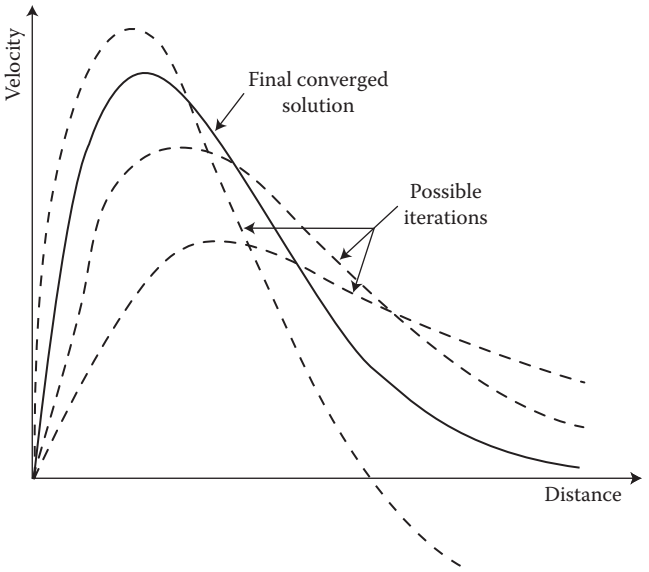


FIGURE 1.4 Typical iterations, leading to a converged result, in the numerical solution of ODEs that determine the velocity profile in a flow.

expression for drag may be obtained from considerations of air flow around the body. For a flat plate, a commonly employed expression for the frictional force is $(AV^{13/7} - BV)$, where V is the speed at which the plate is moving in stationary air and A and B are constants that depend on the length of the plate and the properties of air at the given temperature. Then, if m is the mass of the plate and g the magnitude of gravitational acceleration, the terminal velocity is the root of the equation

$$AV^{13/7} - BV = mg \tag{1.6}$$

From a physical consideration of the problem, we know that the terminal velocity must have a unique, positive value. The range in which the value lies may also be estimated from the available results for other bodies, for example, the sphere. A similar equation is obtained for bodies of other shapes and sizes. In many cases, the expression for drag is obtained from a curve fit of experimental results and is given as a fairly complicated function of the velocity V. A solution of the resulting force balance equation will then yield the terminal velocity for the given body. The method for solving the above equation may be selected knowing that the root is real, distinct, and positive. As discussed in Chapter 5, the secant method and the Newton–Raphson method are two efficient computation schemes that may be employed for this problem. If a method that determines all possible roots of the equation is used, the physical considerations are employed in choosing the correct solution. Since the solution is expected to be unique, the other roots must be complex numbers, negative or beyond the expected range of values.

The physical background of the mathematical problem being solved numerically is particularly important in the solution of nonlinear equations, such as the polynomial equation, Equation 1.6, or transcendental equations. Some examples of the latter are as follows:

$$\tan x = \frac{B}{x} \tag{1.7}$$

$$\log x + 2x^2 = 4 \tag{1.8}$$

$$e^x + x^2 - 2x = 2 \tag{1.9}$$

Nonlinear equations arise very frequently in engineering problems, such as those related to fluid flow, heat transfer, chemical reactions, and dynamics of bodies. The problems encountered may involve finding the roots of a given nonlinear equation or solving a system of nonlinear equations. Since the characteristics of nonlinear equations are generally much more complicated than those of linear equations and since several solutions are feasible, the physical aspects of the problem are used in the development of the computational procedure and in deciding which solutions are acceptable. Even for solving a system of linear equations by iterative methods, physical considerations are often important in obtaining the starting values. Linear and nonlinear equations are also frequently obtained in the numerical solution of partial differential equations (PDEs). The physical nature of the quantities to be computed is usually employed in the choice of the method, the initial guess, the grid to *discretize* a computational region, the desired accuracy level, and the convergence criterion for the termination of the numerical scheme. Since analytical solutions are rarely available, the numerical results obtained are generally considered in terms of the fundamental nature of the problem in order to determine the validity of the numerical scheme.

Curve fitting is another area in which the physical or basic considerations underlying the given problem are of particular importance in developing the computational scheme. Numerical methods are generally used to obtain the best fit to a given set of data. In such cases, it is important to know the expected trends on the basis of the physical aspects of the problem, so that the best fit obtained is a true representative of the process involved.

Consider, for example, the mean daily ambient air temperature at a given location. We wish to obtain a mathematical expression from the 365 data points that represent the measurements of the average daily temperature over a year. We could obtain a 364th-order polynomial from the given data. However, to do so would involve a substantial computational effort, both in obtaining the polynomial and in the subsequent usage of the polynomial in relevant problems. Moreover, the air temperatures fluctuate due to environmental disturbances. Consequently, we are interested in obtaining an expression that represents a best fit to the data and also characterizes the variation over the year. Since we know that the variation is periodic, with a time period of 365 days, we may try to fit the measurements to a

sinusoidal variation. Examples of some of the distributions that may be employed are as follows:

$$T_a = A \sin [\omega(t - a)] \tag{1.10}$$

$$T_a = A \sin \omega t + B \cos \omega t \tag{1.11}$$

$$T_a = A \sin \omega t + b \sin 2\omega t \tag{1.12}$$

where T_a is the ambient temperature; ω is the frequency, given as $2\pi/365$; t is the time in days; and A, B, and a are constants to be determined numerically from a best fit. The first equation is frequently used, with fairly satisfactory results. Figure 1.5 shows the resulting curve fit qualitatively. Similar considerations are employed in obtaining empirical correlations from experimental data and for representing material property data, such as those of interest in thermodynamics, by a best fit.

Numerical simulation of engineering systems is important in design and optimization. It involves the mathematical modeling of components and physical or chemical processes that comprise the given problem to simplify the problem, followed by a numerical solution of the governing equations obtained. The input parameters, initially chosen on the basis of available data, are varied until a close agreement between the physical system and the numerical model is obtained. Once an existing system or process has been numerically simulated, the effects of variations in design on the performance of the system may be studied numerically, leading to optimization. At various stages in such a study, the physical or chemical aspects of the problem are employed. In fact, the comparison between the numerical model and

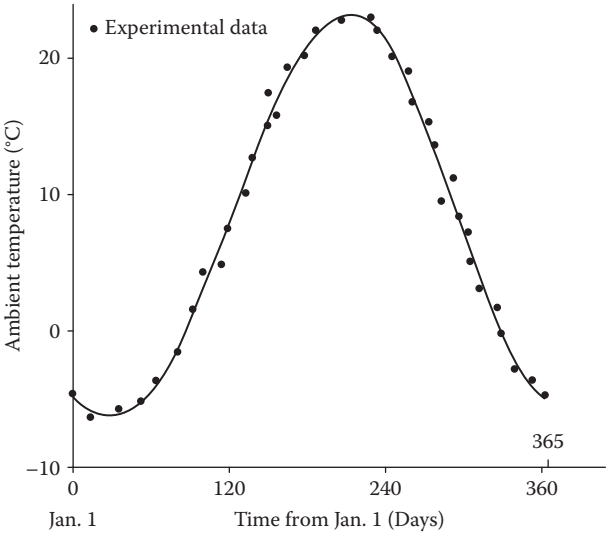


FIGURE 1.5 Sketch of the best-fit curve to the experimental data on the ambient temperature variation over the year at a given location.

the actual system forms the basis for the development of the numerical scheme and for the study of the numerical results obtained.

Therefore, in the presentation of numerical methods for engineering problems, actual problems need to be considered, in order to demonstrate the importance of the physical background of the problem in the selection of the method and in determining if the numerical results are accurate and valid. The general features of the various methods are important and must also be studied in detail. However, some of the important aspects can be best understood in terms of the underlying physical considerations. Therefore, simple examples from several areas of engineering interest are employed in this book.

1.5 APPLICATION OF COMPUTER METHODS TO ENGINEERING PROBLEMS

Computational techniques are used in engineering for a wide variety of applications. Several examples of problems that are generally solved on the computer have been given in the preceding discussion. Numerical methods for engineering application may best be considered in terms of the various mathematical problems that commonly arise in engineering. Computer methods for the solution of these problems may then be considered, using examples of mathematical expressions and equations from various engineering disciplines. This approach would allow a consideration of the various methods that may be employed for obtaining the numerical solution of a particular mathematical problem, say, integration, while employing examples from engineering to bring out the importance of physical considerations in obtaining accurate and valid results. This book employs this approach to present and discuss computer methods for engineering.

Various types of mathematical equations are encountered in engineering applications, such as linear and nonlinear algebraic equations and ordinary and PDEs. Frequently, systems of equations, which are linked with each other through the unknown variables, are obtained. PDEs arise in areas such as heat transfer, fluid mechanics, elasticity, electrostatics, and combustion. These equations are usually solved by *finite-difference* or *finite-element* methods, which convert the problem into a system of algebraic equations by applying the PDEs at a finite number of grid points or integrating them over finite regions. ODEs are also sometimes solved by these methods. Therefore, the solution of a system of algebraic equations is very important in engineering applications, and many methods have been developed to solve the different types of equations that are frequently encountered. Sets of algebraic equations are also directly obtained in many physical problems, such as those of interest in thermodynamics, economics, vibrations, structural analysis, and electrical networks. Although linear systems are particularly important, many engineering problems result in systems of nonlinear equations, which must be solved iteratively to obtain the solution. However, in most cases, nonlinear systems are formulated so that the methods for linear equations may be employed iteratively to converge to the desired solution.

In many engineering problems, the roots of a nonlinear algebraic equation, transcendental or polynomial, are to be determined. Such problems arise, for instance,

in the determination of the temperature of a body from an energy balance, the terminal velocity of a body falling under gravity, the density of a gas from its equation of state, and vibration frequencies from the characteristic equation of a given system. Again, various methods are available, some of which are applicable only to polynomial equations, while others may be used for finding the real or complex roots of other types of equations. Depending on the nature of the problem, the appropriate method may be selected. If not much prior information is available on the nature and approximate magnitude of the roots, the general behavior of the function $f(x)$ that constitutes the given equation, $f(x) = 0$, where x is the unknown, may be investigated numerically. The numerical method for the solution may then be chosen on the basis of the information obtained on the variation of $f(x)$ with x.

ODEs are important in several areas of engineering interest, such as heat and mass transfer, dynamics, fluid flow, chemical reactions, electrical circuit analysis, and elasticity. In some cases, PDEs can be transformed into ODEs. Frequently, several ODEs that are coupled through the unknowns are to be solved simultaneously. The solution procedure depends on the nature of the problem, particularly on the order of the equation, that is, the highest-order derivative in the equation, and the boundary conditions. For instance, the following second-order ordinary differential is obtained for a resonant electrical circuit:

$$A\frac{\mathrm{d}^2V}{\mathrm{d}t^2} + B\frac{\mathrm{d}V}{\mathrm{d}t} + V = 0 \tag{1.13}$$

where V is the voltage across a capacitor, A and B are constants that depend on the resistance, inductance, and capacitance in the circuit, and t is time. If the initial conditions are given as

$$V = V_0 \quad \text{and} \quad \frac{\mathrm{d}V}{\mathrm{d}t} = 0 \ \text{ at } t = 0 \tag{1.14}$$

we have an *initial-value* problem, in which the integration of the equation may be started at the given time $t = 0$ and incremented to larger time to obtain the solution. If one of the conditions is given at a different time, a *boundary-value* problem is obtained, in which a correction scheme is needed to satisfy the given conditions. Similarly, the boundary conditions may be given at two different spatial locations, or two different values of the independent variable. Then, iteration is generally employed to converge to the solution.

Besides algebraic and differential equations, several other mathematical problems arise in engineering. Numerical differentiation and integration are needed in many cases, often as part of a more complicated problem. Numerical integration over time is needed, for instance, in determining the total energy lost or gained by an object, such as at the surface of a lake. Similarly, integration of velocity across a cross section of a channel gives the total volume flow rate in the channel. Numerical differentiation is needed, for example, in the determination of the acceleration of a particle from the measured variation of its velocity with time. Rate processes are important in engineering, and numerical differentiation is frequently employed for obtaining the rates of change of various physical quantities. Numerical techniques are also

needed in interpolation and extrapolation, employing curve fitting of given data. In some cases, an exact fit which yields the exact value at the given data points is appropriate. However, more frequently, a best fit of the data is employed so that the general features of the results may be represented by a correlating equation, without forcing the curve to pass through each data point, as seen earlier in Figure 1.5. Software for graphics can be employed advantageously with the computer solution of engineering problems to present the numerical results.

In summary, a consideration of numerical methods for engineering application involves a wide variety of mathematical problems, as outlined here. It is important to understand the advantages and limitations of a particular method for solving a given problem. The numerical procedure and the results obtained must also be related to the physical or basic background of the problem in order to ensure the validity of the computational scheme and to choose an acceptable solution. Similarly, a comparison between the numerical and analytical results must be made, whenever possible, to check the accuracy of the results obtained. The development of the numerical scheme for a given problem may be discussed in several ways. A practical approach is to take the mathematical problem arising from the actual circumstance, present the computer program, and discuss the numerical results in terms of the physical aspects of the problem and available analytical results. It is this approach that is followed here. The computational software chosen is MATLAB, which is presently the most widely used computational environment for the application of computer methods to engineering problems. However, other languages and software may also be employed by suitably modifying the given programs, as discussed in Chapter 2. Of particular importance in the use of numerical techniques for solving engineering problems is the need to check the computational scheme for accuracy and to correctly interpret the numerical results obtained. In this book, these and other aspects mentioned earlier will be considered in terms of various examples taken from several engineering disciplines, including aeronautical, chemical, civil, electrical, industrial, and mechanical engineering.

1.6 OUTLINE AND SCOPE OF THE BOOK

1.6.1 Basic Features

This book presents the mathematical background as well as the application of computational techniques to problems of engineering interest. The material is developed by the derivation of the formulas for each method, followed by a discussion of the accuracy, computational effort, storage requirements, and range of applicability of the method. For each problem area considered, for example, root solving, several methods are discussed, emphasizing the ones that are most extensively employed. A comparison between various methods applicable for a particular type of mathematical problem is made, in order to indicate the advantages and disadvantages of a given method. Of particular interest in such a comparison are the associated errors, ease in programming, computing time and storage needed, and flexibility in the application to a wide variety of problems. The circumstances under which a given method would be the preferred one are outlined. This consideration is an important one, since several methods are frequently available for problems that arise in engineering applications

and the choice of the most appropriate method is highly desirable, in order to minimize the computing resources needed and to obtain the required accuracy level.

Following a detailed discussion of the mathematical background and the derivation of the relevant formulas for each numerical method, the computational procedure for applying the technique is discussed. The important considerations underlying the development of the numerical scheme are discussed, along with the difficulties that may be encountered. Appropriate MATLAB commands and schemes are outlined, whenever appropriate, or reference is made to programs in Appendix B to illustrate the numerical solution. Finally, examples based on actual engineering or mathematical problems are given, for most of the methods considered, and the computer program is outlined. Again, the important features of the program are discussed and the numerical results obtained are presented and discussed. The emphasis is on presenting the basic algorithm of the method in terms of its application to an actual physical, chemical, or mathematical problem. Although the program is discussed as part of the example and is, therefore, geared to the solution of the specific problem considered, a few modifications in the program can easily be made to use it for the solution of other problems of similar nature. This approach of writing a problem-oriented computer program presents the program simply as a sample and encourages the reader to write his or her own program on the basis of the information given, making the program as efficient as possible and employing ongoing improvements in available computational facilities. General programs that can be used for a wide range of problems are also presented in many cases.

1.6.2 Computer Programs

Many useful features are incorporated in the computer programs given in the book. Both interactive and batch operation modes are utilized. In the former case, the input data are fed and the results are obtained interactively by the operator. This makes an interactive use of the computer preferable for short computer runs and for program development. The batch mode, in which the entire program is entered with the input data and the computer gives the results after the complete run, is preferred for large runs and complicated programs, after the program has been developed, tested, and debugged. Although most programs are written for the MATLAB environment, several programs are also given in Fortran, in order to indicate the similarities and differences between these and to demonstrate the ease with which the basic logic of the program can be employed in a different language or environment. Also, Fortran continues to be an important programming language for engineering problems. Subroutines or function files are useful in developing complicated programs and are employed wherever appropriate. In some cases, the outputs are stored in data files for future analysis or plotting and, in others, these are printed or plotted as soon as the computational runs are completed.

1.6.3 Examples and Problems

The examples and problems considered in this book are derived from topics of interest in the major engineering disciplines and in the basic sciences. The physical

or basic background of the problem is outlined in order to enable the reader to follow the relevance of these considerations in the choice and testing of a particular numerical technique. Also, a selection of problems that arise in practical circumstances makes the discussion interesting and relevant to engineering applications. As discussed earlier in this chapter, numerical solutions must be considered not only in terms of the basic nature of the given problem but also in terms of any analytical solutions available, even if these are for very simple situations. These aspects are stressed in evaluating the numerical results for accuracy and validity. In solving problems of engineering interest, the available information on the given system or process must form the basis for the development of the numerical scheme and for the verification of the results obtained.

Both the problems and the examples tend to expand on the material covered, so that they contribute to an increased understanding of the discussion given in the text. Several new physical phenomena are also introduced in the problems to indicate the application of the methods presented to a much wider spectrum of engineering processes. Although the emphasis is, obviously, on the numerical solution, several problems are also directed at the mathematical background, particularly at the errors involved and the mathematical formulation for a numerical solution. In addition, many problems can be solved on a calculator in order to study a given numerical scheme.

Much of the material presented in this book has been used in courses taught at the sophomore and junior levels in engineering. A few of the topics covered may be somewhat advanced for sophomore students. Similarly, the physical background of the problems may not be familiar to some of the readers. Consequently, a brief discussion of the important aspects of the problem or example under consideration is included. In some cases, reference is also made to books that can be consulted for a more detailed coverage of the topic. A background in programming, such as a freshman-level, one-semester course, is assumed, although some of the important aspects are covered in Chapters 2 and 3 for completeness.

1.6.4 A Preview

The presentation of the numerical techniques for engineering application starts with Chapter 2 on the basic considerations in computer methods. This chapter outlines the important elements in computational procedure, including program development, numerical errors, accuracy, convergence, and other basic aspects. Although some of the discussion will be quite familiar to those experienced in computer programming, many of the aspects considered in this chapter are important in obtaining an accurate and valid solution to a problem of engineering interest. This chapter also outlines the current trends in computational methods and facilities, with respect to both the software development and the growing capability of computer systems.

A brief review of MATLAB is presented in Chapter 3 in order to discuss the main features of this computational environment. Commonly used commands and the basic procedures to develop a program in MATLAB are outlined. Standard software that can be used advantageously to solve mathematical problems, such as matrix

inversion, root solving for polynomial equations, solution of a system of linear equations, and obtaining a best fit from given data, is presented and discussed. Since plotting of data is easily done in MATLAB, some simple plotting methods are presented. This chapter serves to give a brief discussion of programming in MATLAB, while referring to more extensive presentations in other books, and also outlines the terminology and nomenclature to be used in later chapters

The *Taylor* series, which forms an important element in the estimation of numerical truncation errors (TEs), is presented in Chapter 4, along with the numerical approximation of derivatives. Several methods for differentiation are presented, and many of the results presented here are employed in later chapters. Methods for finding the roots of nonlinear algebraic equations are discussed in Chapter 5. Several methods, which are based on the sign change, at the root, of the function $f(x)$ in the given equation $f(x) = 0$, are first considered. Efficient methods such as the secant and Newton's methods, which converge very rapidly, although they may also diverge in certain cases, are discussed in detail. Specialized methods for equations in which $f(x)$ is a polynomial are also discussed. Finally, a comparison between the various available methods is made.

The solution of simultaneous linear or nonlinear algebraic equations is an important problem in engineering applications and forms the subject of Chapter 6. Direct as well as iterative numerical methods are discussed, the latter being the inevitable approach for most nonlinear equations. Eigenvalue problems are also considered and the available methods outlined. Numerical methods for curve fitting of data are presented in Chapter 7, considering both the exact fit as well as the best-fit approach. Various techniques for interpolation are discussed, emphasizing popular methods such as Lagrange and Newton's interpolating polynomials. The least-squares method for a best fit is discussed in detail, and various forms of the function for curve fitting are considered.

Numerical integration forms the subject of Chapter 8, and several important methods, such as the trapezoidal and Simpson's rules, Romberg integration, and Gaussian quadrature, are discussed. The advantages of each method, its limitations, and the conditions under which it is preferred are considered in some detail. The associated errors and the resulting accuracy are also discussed. The numerical integration of improper integrals, whose limits of integration may be infinite or the integrand may become singular over the range of integration, is also presented.

The solution of differential equations is an important subject in engineering. Because of the complexity of typical engineering problems, numerical methods are generally needed. ODEs are considered in Chapter 9 and PDEs in Chapter 10. Both self-starting methods, such as Euler's and Runge–Kutta methods, and multistep methods, such as predictor–corrector methods, are considered for ODEs. Also, the associated errors, accuracy, stability, and convergence of these methods are considered, along with their efficiency in terms of the computational effort required. Several types of equations, including initial-value, boundary-value, and systems of equations, are considered and the relevant numerical techniques are presented. Again, a critical comparison between the various methods is made in order to guide the choice of the most suitable scheme for a given problem. Finite-difference methods, derived from the numerical approximation of derivatives given in Chapter 4, are also outlined for ODEs.

PDEs are included in this book largely for junior- and senior-level students and also for professional engineers. With the introductory background presented, the material could also be used for less advanced students. The material covered in Chapter 10 considers mainly linear equations of parabolic, elliptic, and hyperbolic type. The basic nature of the equations is discussed in detail, and important numerical methods for their solution are presented. The questions of accuracy, convergence, and stability are again considered. Finite difference methods are largely considered, with a brief introduction to finite element methods, since the former is easier to understand and can be developed on the basis of the material presented in Chapter 4. The methods for treating different types of boundary conditions are also outlined.

In all the topics considered here, a large number of examples and problems are given, so as to provide a strong physical and numerical base for the computational study of engineering problems. Since the best way to learn numerical methods is by applying the techniques available to different problems and developing one's own computer code, almost all the examples and many of the exercises demand the development of the relevant program and its use for obtaining the desired numerical results. Although a calculator may be used in several cases to study the computational steps in a given method, the readers are strongly encouraged to write computer programs for the problems given, using the discussions, formulas, and examples given in the text.

As mentioned earlier, this book is largely directed at the use of the MATLAB computing environment for solving engineering problems. However, many Fortran programs are also included in deference to the continued importance of this programming language in engineering. Extensive expertise and software exist in Fortran and it continues to be widely used, particularly for complex problems. However, the student or the reader can easily focus entirely on MATLAB, if desired, or a chosen mixture of the two computing software may be employed for instruction.

2 Basic Considerations in Computer Methods

2.1 INTRODUCTION

In the numerical solution of engineering problems, there are several important aspects that need to be considered in order to ensure the validity of the chosen approach for a given problem and the accuracy of the results obtained. The computational procedure involves a consideration of the methods available for solving the given problem, the appropriate programming language, the computational environment and software being employed, the computer and its operating system, and so forth, before proceeding to the development of the numerical scheme, or algorithm, and the corresponding program. Since these considerations are fundamental to most computer methods, this chapter discusses the general approach to the development of the computational scheme. Also considered are the interfacing with available computer software and the verification and validation of the numerical results by a comparison with available analytical and experimental results, as discussed in Chapter 1.

The consideration of numerical errors and the accuracy of the results is important in the numerical solution of any given problem. The various types of errors that arise in the computational approach are discussed, along with methods that may be employed for reducing the error. The accuracy of the solution may often be estimated by comparing the numerical results with those from the analytical solution for simpler problems, since the analytical solution of the given problem is presumably not available. Frequently, satisfactory analytical results are not available for comparison. In such cases, the numerical scheme itself is first employed to check the accuracy of the numerical results by ensuring that numerical parameters, such as the chosen time step and grid size, do not significantly affect the results. This process is often known as *verification* of the numerical method. Also, the basic nature of the problem being solved can often be employed as a check on the validity of the numerical scheme and the correctness of the results obtained. The accuracy of the numerical results can frequently be evaluated by substituting the solution obtained back into the algebraic or PDE being solved to determine how closely it satisfies the equation. Several other similar procedures are generally employed to check the accuracy of the numerical solution.

Consider, for example, the dynamics of a moving body whose displacement x is governed by the ODE $\mathrm{d}x/\mathrm{d}t = F(x, t)$, where t is time and $F(x, t)$ is a given function. We may assume that the analytical solution is not available, since if it were, there would be no need to solve the problem numerically. However, the numerical scheme

may be employed to solve a simpler equation, say, $dx/dt = -ax + b$, where a and b are constants. The mathematical solution to this equation can be obtained as $x = ce^{-at} + b/a$, where c is a constant to be determined by applying the initial condition, that is, by using the given value x_0 of the displacement at time $t = 0$ or at any other specified time; see Figure 2.1. The accuracy of the numerical method may be estimated by comparing the numerical solution for this simple problem with the analytical solution. For a more complicated function $F(x, t)$, the following considerations may be used. The physical nature of the problem demands that the displacement be real and positive. Also, it would often be known whether it is periodic or whether it must increase, or decrease, with time. This information may be employed to select the correct solution in case multiple solutions arise and also to check the validity of the numerical scheme. Once the numerical solution $x(t)$ is obtained, numerical differentiation may be used to determine dx/dt for a few selected values of t. These may then be employed to check if the numerical values of x do indeed satisfy the equation $dx/dt = F(x, t)$ to the desired accuracy level. Finally, the step size Δt employed in the numerical scheme must be reduced until a further reduction in Δt does not significantly affect the numerical results. Of course, if any experimental results are available on the given problem, these may be effectively used for evaluating the accuracy of the numerical results.

The numerical methods for the solution of several problems are based on an iterative approach, in which the solution is gradually improved, starting with an initial, guessed value until the change in the solution from one step to the next becomes less than a chosen small quantity, known as the *convergence criterion* or parameter. In such cases, the convergence of the iterative procedure is an important consideration, and it is necessary to determine the conditions under which the scheme may diverge. If a particular method diverges for a given problem, the problem can sometimes be reformulated

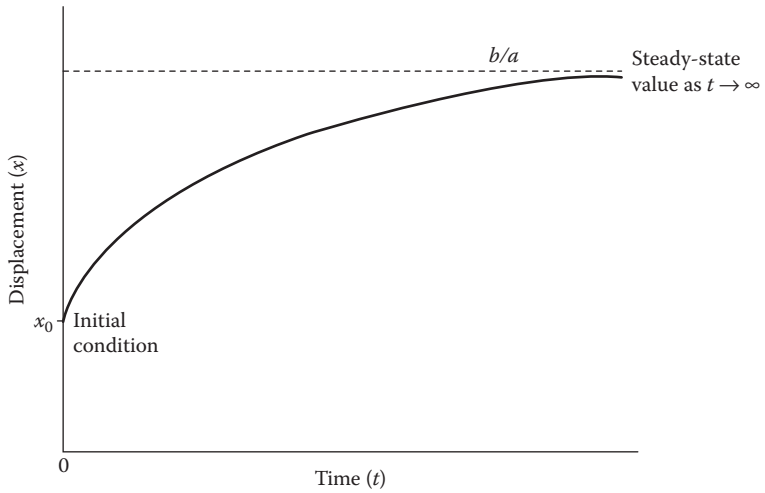


FIGURE 2.1 Sketch of the analytical solution of the differential equation $dx/dt = -ax + b$, where a and b are constants and $x = x_0$ at $t = 0$.

so that the scheme converges. Otherwise, a different method must be employed. *Numerical stability* is another important consideration that guides the selection of the method and of the grid, or step, size in the numerical scheme. Again, it is necessary to determine when numerical instability might arise and to take steps to avoid it.

This chapter discusses many of these considerations which are basic to most numerical methods. The general approach to the development of a numerical scheme is outlined, indicating various important aspects that need to be taken into account. The concepts of error, accuracy, iteration, convergence, and stability are discussed in general terms, by taking examples from various topics, such as root solving, numerical differentiation and integration, curve fitting, and solution of algebraic and differential equations, considered in greater detail in later chapters. The discussion in this chapter forms the basis for the development, application, verification, and validation of the numerical procedures for these and other topics of interest in engineering applications.

2.2 COMPUTATIONAL PROCEDURE

The general approach to the development and application of the computational procedure for solving a given problem is discussed in this section, indicating the important aspects that generally need to be considered for an efficient and accurate scheme. Although some of the considerations outlined here may not be applicable to a particular circumstance, it is important to recognize the important steps that lead to a successful numerical method. Most of the items included here are fairly straightforward and are quite familiar to those who have done a significant amount of numerical work. However, the systematic approach given here is helpful, particularly for those who are relatively less experienced in computer methods, in investigating the relevant aspects that determine the efficiency, accuracy, and validity of the numerical procedure. It is assumed that the mathematical formulation of the given physical or engineering problem has been completed and that an analytical solution is not easily obtainable, so that it has been decided to solve the problem numerically.

2.2.1 Method Selection

Frequently, several methods are available for the numerical solution of a given mathematical problem. The selection of the method to be employed, from among the several applicable methods, is an important consideration and is generally based on many relevant criteria, such as the following:

1. Accuracy
2. Efficiency
3. Numerical stability
4. Programming simplicity
5. Versatility
6. Computer storage requirements
7. Interfacing with available software
8. Previous experience with a given method

The accuracy of a given method is an important consideration in its selection for solving a particular problem. The evaluation of the accuracy of a method may be based on a comparison of the numerical results with available analytical results, as outlined in the preceding section, on an estimation of the associated numerical errors, or on various methods for checking the correctness of the numerical solution, such as substitution of the numerical results back into the equation being solved to determine the accuracy to which the numerical solution satisfies it. All these aspects, particularly the numerical errors that arise in computational methods, are discussed in detail later in this chapter.

The *efficiency* of a given method is generally based on the total number of arithmetic operations needed for solving the given problem. This is reduced to the number of arithmetic operations needed per computational step if the number of steps is fixed. One could also solve a given problem with different methods and determine the computational or central processing unit (CPU) time needed in each case, as obtained from the computer. However, the number of arithmetic operations, which include addition, subtraction, multiplication, and division, can often be determined by noting down the various mathematical manipulations performed, per step, in a given numerical scheme. If a particular method involves a smaller number of total arithmetic operations needed to solve the given problem, than another method, then it is more efficient. A higher efficiency of the method also implies shorter computer time and, thus, lower computational cost. For instance, matrix inversion methods for solving systems of linear equations, though convenient and widely used, are generally less efficient than other direct methods, as seen in a later chapter.

Numerical instability refers to the unbounded growth of numerical errors as computation proceeds. It is of particular concern in the solution of differential equations and, if present, can lead to an erroneous and unacceptable numerical solution. Therefore, it is important to determine the stability characteristics of the various methods that are applicable to a given problem. Frequently, the numerical scheme may be conditionally stable; that is, it may be stable within certain constraints that often limit the grid or step size. In the solution of parabolic PDEs, for instance, the explicit schemes, which are generally simpler to use, often restrict the step size to small values, making these schemes inefficient. Then the implicit methods, which usually do not have such constraints resulting from stability considerations, are preferred. Thus, the numerical instability of the method is an important consideration in its selection.

As listed before, several other considerations also play an important role in the selection of the method. These include simplicity in programming, versatility of the method, computer storage needed, and interfacing with available software. In engineering applications, the simplicity and versatility of the method are very important, since interest often lies in solving a wide variety of problems with the least amount of effort. This is particularly true for the design and optimization of systems that often involved a diversity of components and equations. Frequently, some sacrifice is made with respect to accuracy and efficiency in order to select a simpler and more versatile method. An example of this is the *Runge–Kutta* method, for solving ODEs. This method is often chosen over *predictor–corrector* methods, which are more efficient than the former but are also more complicated to program.

The computer storage requirements of the method are generally important in the simulation of large systems that are of interest in engineering applications. For

example, the *Jacobi* method for solving a system of linear algebraic equations involves the storage of the matrices of the unknowns at two iterative steps, the present and the previous one, whereas the *Gauss–Seidel* method requires the storage of only the latest values. Thus, the latter method requires only about half the storage needed by the first method. It is also more efficient on conventional single-processor computers and is preferred.

The interfacing of the numerical method with the computer software is particularly important when available programs are being employed. For instance, if a matrix inversion program is available, methods based on the inverse of the matrix for solving a system of linear equations may be chosen. This is particularly true for MATLAB®, which has excellent matrix inversion software built into the system. Similarly, prior personal experience with a given method would be an important consideration in its selection.

2.2.2 Programming Language

After the numerical method for the solution of the given problem has been selected, the next step is the development of the computer program or code that allows one to interface with the computer system. However, before proceeding with the code development, one must select the programming language and the computer system to be used and become fully conversant with the selections made. The programming languages, often termed *high-level languages*, allow one to write the step-by-step instructions, or algorithm, for the computer in a form that is quite similar to ordinary English and algebra. The computer itself interprets and executes statements only in the machine language, and a compiler is employed by the computer to achieve the translation from the programming language to the machine language. The machine language program is then stored, providing direct access for immediate or later execution.

Several high-level programming languages have been developed over the years. In the past, the most widely used among these, for engineering and science, was Fortran, which stands for *formula translation*. It was originally developed by IBM in the 1950s for scientific and engineering applications and is now available in many versions, such as Fortran 77, Fortran 90, Fortran 95, and Fortran 2003. It is still commonly used and remains one of the important languages for high-performance scientific computing and for benchmarking and ranking the world's fastest supercomputers, partly because of extensive existing programs for a wide array of engineering problems. Fortran 90 and beyond are also well suited for use on parallel machines. Most Fortran programs are structured so that control flows from top to bottom, rather than one in which control is transferred from one point in the program to another in a seemingly random fashion. The structured system makes development as well as debugging relatively easy. Similarly, other important features, such as *object-oriented programming* that uses *objects*, which include information on the relevant data, methods, and their use to design the computer programs, have also been incorporated in recent versions. Several Fortran programs are given in this book to present the algorithm and the logic of the method, as well as to show the similarities with and differences from the MATLAB environment and to provide information for those who are well versed in this programming language. Many books are available on programming in Fortran and may be referenced for details on

the language. See, for instance, the books by Metcalf, Reid, and Cohen (2004), Chapman (2007), and Chivers and Sleightholme (2009).

There are several other programming languages that have been employed for solving problems in science and engineering. These include *Basic*, *Pascal*, *C*, *Lisp*, and others. Among these, *Basic*, which stands for *beginner's all-purpose symbolic instruction code*, was also a widely used language, particularly on PCs, since it is generally simpler to use than Fortran and is well suited for small programs. However, it is not as versatile as Fortran and is often inconvenient for large, complex programs. Many improved versions of *Basic* have been developed in recent years, and many of the constraints that existed in the earlier versions have been eliminated. A useful version is *Visual Basic*, which is a relatively easy to learn and use programming language, because of its graphical development features and derivation from *Basic*.

Similarly, other programming languages have their special advantages and limitations. An important language is *C*, which is a general-purpose programming language developed in the last two decades. It is a relatively low-level language, implying that it is closer to assembly language than high-level languages such as Fortran. As a result, it is more difficult to move the program from one computer system to a different one. However, the language was designed to encourage machine-independent programming, allowing *C* programs to be compiled for a very wide variety of computer platforms and operating systems with little or no change to its source code. The language has several advantageous features in control flow and data structures because of which it is one of the most popular programming languages and is widely used on many different software platforms. *C* has greatly influenced many other popular programming languages, most notably *C++*, which originally began as an extension to *C*. For details on the *C* and *C++* languages, the books by Kernighan and Ritchie (1988), Kochan (2004), Prata (2005), King (2008), and Stroustrup (2000, 2009), among many other available books, may be consulted.

Several other programming languages have gained considerable importance in the last few years. Among these are languages that allow symbolic manipulation, that is, languages in which words, sentences, and expressions can be employed for programming. *Lisp*, which takes its name from *list programming*, is one such language that is important in the development of intelligence in computers. Similarly, *Prolog* and *Smalltalk* are languages used in generating artificial intelligence in engineering systems. For details on these languages, several references are available. See, for instance, the books by Winston and Horn (1989), Clocksin (2003), Clocksin and Mellish (2004), and Lalonde (2008).

Recent years have seen a tremendous growth in computational software, including programming languages and computational environments, making it convenient and efficient to carry out the numerical solution of the wide range of problems encountered in engineering applications. Some of these that may be mentioned are MATLAB, *Mathematica*, *SciLab*, *Maple*, GNU *Octave*, *R programming language*, and *Perl Data Language*. The more computationally intensive aspects in the software are often based on some variation of Fortran or *C*. The main computational environment used in this book is MATLAB and Chapter 3 is devoted to a brief discussion on the programming and implementation in this environment.

The computer program, written in a high-level language such as Fortran or *C++*, is implemented on the computer by means of an interpreter or a compiler. An interpreter examines each line of the program and checks it for the rules of the language before it is executed. The interpreted approach is very valuable during program development, since error messages are given as soon as a statement is entered. However, it is very slow in the execution of the program. A compiler, on the other hand, organizes the entire program into a set of machine instructions and locations, and several compilers are available. The compiler is often written for a given computer system and is generally a completely separate process undertaken before the program is run. Once the machine code has been produced by the compiler, the compiled program is stored and the program may be executed with a separate command. A single command that compiles and executes the program may also be used. The use of a compiler thus reduces the computer time for a given problem. Various compilers have their particular advantages and characteristics. For instance, Unix and Linux are particularly good at providing diagnostic error messages and are widely used.

From the above brief discussion of the various programming languages widely employed for engineering problems, it is obvious that the trend has been toward structured programming and interactive use of the computer, through an interpreter, which responds almost immediately, or an interactive compiler. Substantial improvements and modifications continue to be made in the available languages to simplify programming and to increase the versatility and capability of the language. Although it is difficult to keep up with all the advancements in the high-level languages, available interpreters and compilers and computational software, it is important to determine what is available on a given computer.

In general, an interactive use of the computer is preferable during program development, since the parameters of the problem may be entered by the operator at the terminal. The program may be compiled and executed to obtain the output as the program continues to execute. If the results are unacceptable, the execution may be stopped at any stage, and the input parameters varied and execution resumed. In the batch operation mode, the input parameters are part of the program, and the execution of the program must be completed before any changes can be made. Thus, at the initial stages of program development, interactive computer usage is particularly valuable. Once the program has been satisfactorily developed, detailed numerical results are best obtained by the batch operation mode on the computer.

Example 2.1

Compute the sum S of the series

$$S = 1 + x + x^2 + x^3 + \cdots + x^n + \cdots \tag{2.1}$$

where x is a variable whose value is to be entered into the program interactively. In order for the series to be convergent, $|x| < 1$. This series represents the binomial expansion of $1/(1 - x)$, which therefore gives the exact value SX of the series. Compare the exact and computed values of S to determine the numerical error. Discuss the dependence of the sum S on the number of terms n taken in the series.

SOLUTION

The value of x is to be entered and terms in Equation 2.1 are to be added sequentially. The basic considerations relevant to convergence are discussed in detail later in this chapter. However, it will suffice to mention here that each term in the series, given in Equation 2.1, is larger than the next term, for $|x| < 1$. Thus, the contribution of each additional term to the sum decreases as n is increased. This relationship is used as a check on the convergence, since it is not possible to take an infinite number of terms and since it is desirable to have the least number of terms that give S within an acceptable error. If SN represents the nth term and S the sum of the series up to and including this term, then the condition $SN/S < \varepsilon$, where ε is a chosen small quantity, such as 10^{-6}, which implies that the contribution of the nth term to the sum S is less than 10^{-4}%, can be employed to check the convergence and to terminate the computation if this condition is satisfied. The percentage error E is then given by $E = 100\ [(SX - S)/SX]$.

The preceding description of the procedure to solve the problem may be written in terms of the following steps:

1. Set the initial value of the sum S as zero.
2. Set the initial value of the term n as zero.
3. Enter the value of x.
4. Add the next term $SN = x^n$ to the sum S.
5. Check if the convergence criterion $SN/S < \varepsilon$ is satisfied.
6. If the convergence criterion is satisfied, stop and print the results on n, S, and E.
7. If the convergence criterion is not satisfied, advance n by 1 and go back to step 4.
8. Continue till convergence criterion is satisfied or a given maximum value of n is reached.

A fairly simple computer program can be written to follow these steps, as discussed below and shown in Figure 2.2 in Fortran 77. This program is presented to show the logic and the various steps involved and for those who are familiar with the language.

The program would then yield the number of terms needed for the preceding convergence criterion to be satisfied, the computed sum S of the series, and the percentage error E. Figure 2.3 presents the typical results obtained from this program. Here E is given in a format of the form 0.1E–04, or 0.1×10^{-4}, in order to check against the convergence criterion of $SX/S < 10^{-6}$. Clearly, the error is a function of ε, which may be chosen to keep the error within an acceptable value. Also, note that the number of terms needed increases with the value of x. This result is expected, since convergence is slower at the larger value of x, as discussed in most textbooks on advanced calculus; see, for instance, Larson et al. (2005) and Stewart (2007).

This is an interesting problem, which shows the effect of truncating a series after a certain number of terms and the use of a convergence criterion. The analytical result of the summation of the infinite series is known and can be used as a check on accuracy.

```
C    PROGRAM SERIES SUMMATION
C
C    HERE S IS THE SUM OF THE SERIES UP TO AND INCLUDING THE NTH
C    TERM, SN IS THE NTH TERM, SX IS THE EXACT VALUE OF THE
C    FUNCTION F(X)=1.0/(1.0-X), WHICH IS REPRESENTED BY THE
C    SERIES, AND ER IS THE ERROR.
C
C
C    ENTER INPUT QUANTITIES
C
         IMPLICIT REAL (A-H,O-Z)
         DO 5 I=1,5
         PRINT *, 'ENTER THE VALUE OF X'
         READ *, X
         N=0
         S=0.0
C
C    SUM THE SERIES
C
   1     SN=X**N
         S=S+SN
C
C    CONVERGENCE CHECK
C
         IF ((SN/S) .GT. 1E-06)THEN
         N=N+1
         GO TO 1
         ELSE
   6     WRITE (1,2)X
   2     FORMAT(2X, 'X=', F6.3)
         WRITE(1,7)N
   7     FORMAT(2X, 'THE REQUIRED NUMBER OF TERMS=',I5)
         WRITE(1,3)S
   3     FORMAT(2X, 'THE SUM OF THE SERIES=', F12.6)
C
C    COMPUTE THE ANALYTICAL VALUE OF THE SUM AND THE ERROR
C
         SX=1.0/(1.0-X)
         ER=((SX-S)/SX)*100.0
         WRITE(1,4)ER
   4     FORMAT(2X, 'THE ERROR=', E10.5,'PERCENT' /)
         END IF
   5     CONTINUE
         STOP
         END
```

FIGURE 2.2 Computer program in Fortran for the summation of the series given in Example 2.1.

```
 ENTER THE VALUE OF X
0.1
  X = 0.100
  THE REQUIRED NUMBER OF TERMS = 7
  THE SUM OF THE SERIES = 1.111111
  THE ERROR = .10729E-04PERCENT

 ENTER THE VALUE OF X
0.3
  X = 0.300
  THE REQUIRED NUMBER OF TERMS = 13
  THE SUM OF THE SERIES = 1.428571
  THE ERROR = .25034E-04PERCENT

 ENTER THE VALUE OF X
0.5
  X = 0.500
  THE REQUIRED NUMBER OF TERMS = 20
  THE SUM OF THE SERIES = 1.999998
  THE ERROR = .95367E-04PERCENT

 ENTER THE VALUE OF X
0.7
  X = 0.700
  THE REQUIRED NUMBER OF TERMS = 37
  THE SUM OF THE SERIES = 3.333328
  THE ERROR = .17166E-03PERCENT

 ENTER THE VALUE OF X
0.9
  X = 0.900
  THE REQUIRED NUMBER OF TERMS = 111
  THE SUM OF THE SERIES = 9.999912
  THE ERROR = .85831E-03PERCENT
```

FIGURE 2.3 Results from the program in Fortran for Example 2.1.

2.2.3 Computer System

The next consideration in the numerical solution of a given problem pertains to the computer system. Frequently, several systems, ranging from PCs or workstations to minicomputers and mainframe computers, are available to engineers. Supercomputers may also be accessible for large-scale simulations of engineering systems. If several computers are available, the selection of the most appropriate one for a given problem is important. Once this selection has been made, or if only one computer system is available, one proceeds to obtain detailed information on the various elements of the system, such as the languages available, the operating system, the software available on the system, the input/output facilities, the memory/storage constraints, and the job control language, so as to implement the computer program being developed on the system.

As mentioned earlier, there are two main steps in the numerical solution of an engineering problem. The first involves the development of the computer code, and the second involves repeated execution of the program for a wide variety of input conditions and governing parameters to generate the numerical data needed for, say, the design and analysis of a given engineering system such as a furnace, a boiler, electronic equipment, a robot, a mechanical structure, or a chemical reactor. The computer requirements are usually quite different for these two steps. Code development involves frequent changes in the program and is thus best suited to an interactive use of the computer, preferably with an interpreter. The operating system, examples of which are Microsoft Windows, UNIX, and LINUX, controls the interaction with the computer, particularly the editor, and is an important component in the process. A screen editor, such as word processing programs and EMACS, which is available on many personal and minicomputers, allows one to make changes in the program very rapidly by moving the cursor to the desired location and making the needed modification. A line editor, on the other hand, allows changes to be made line by line, or in a collection of lines, and is much slower. The speed of the CPU, which finally runs the program, is not a very important consideration during code development. Similarly, the output facilities are not as important as at the second stage when computational results are being obtained, in tabular or graphical form.

Thus, during the development of the computer program, a good screen editor, which allows frequent changes and corrections in the program, is desirable. Also, the interpreter or compiler should provide adequate error diagnostics. PCs, workstations, and several minicomputers are particularly suited to code development because of the availability of most of the desirable features mentioned above.

Once the computer program has been developed, the desired numerical results for wide ranges of the governing parameters are obtained by repeatedly running the program with minor changes to enter the appropriate parametric values. Clearly, a rapid execution, with good output facilities, particularly graphics, is desirable at this stage. The editor and error diagnostics are not important. Also, an interactive use of the computer is not necessary. Thus, a batch execution of the developed program on a mainframe computer, or on a supercomputer, is the best method, particularly for large, computationally intensive programs. The program is loaded, compiled, and linked with computer memory before execution, which then proceeds rapidly.

2.2.4 Program Development

2.2.4.1 Algorithm

After the selection and the consideration of the important aspects of the method of solution, the programming language, and the computer system, one proceeds to the development of the computer program. However, before the program can be written, a step-by-step procedure, known as an *algorithm*, must be developed.

STEP 1. Start the calculation.
2. Input the limits x_1 and x_2 on x and the definition of the function $f(x)$.
3. Select the numerical parameters: Step size Δx and the convergence parameter ε.
4. Initialize: Take $x_i = x_1$.
5. Calculate the first derivative $f'(x_i)$
6. Check whether the magnitude of the derivative is within ε.
7. If $|f'(x_i)| > \varepsilon$, then advance x_i by Δx and check whether $x_i < x_2$. If $|f'(x_i)| < \varepsilon$, then go to Step 10.
8. Stop the calculation if $x_i > x_2$.
9. Calculate $f'(x_i)$ and again compare its magnitude with ε. Continue with Step 7 if $|f'(x_i)| > \varepsilon$.
10. If $|f'(x_i)| < \varepsilon$ then calculate the second derivative $f''(x_i)$.
11. If $f''(x_i)$ is positive or zero, advance x_i by Δx. Go to Step 8.
12. If $f''(x_i)$ is negative, a maximum is indicated.
13. Print the required results: x_i and $f(x_i)$.
14. Stop the calculation.

FIGURE 2.4 Representation of the algorithm for determining the value and location of the maximum of a given function $f(x)$ as a sequence of steps to be followed by the computer.

The method of solution is generally expressed in terms of the mathematical formulas involved in the computation. However, the computer must be programmed to follow a definite, logical, step-by-step procedure to perform the desired computation. The algorithm may be written as a sequence of steps to be followed. More frequently, the algorithm is represented graphically by means of a *flow chart*, which shows the steps in the form of a block diagram. Generally, a flow chart is used to outline the computational procedure, without giving the details of the actual computational steps, which are eventually entered into the actual program. Thus, a flow chart serves to indicate the logical sequence of programming steps and is frequently drawn before the program is developed.

The flow chart follows an accepted collection of symbols to represent input/output, decision, terminal, and computation. For example, let us consider the determination of the maximum of a function $f(x)$. In the optimization of engineering systems, one is frequently concerned with maximization or minimization of functions, under specified constraints. Let us assume that it is known that the given function $f(x)$ has a maximum in the range $x_1 < x < x_2$, where x is the independent variable. We know from mathematics that at the maximum, df/dx is zero and d^2f/dx^2 must be negative. Employing these characteristics of a maximum, one may write the algorithm as a sequence of steps, shown in Figure 2.4, or represented by a flow chart, shown in Figure 2.5.

For this problem, the computational procedure involves entering x_1 and x_2, advancing x with a chosen step size Δx, and computing the derivative df/dx. If the derivative is close enough to zero, as indicated by a chosen small quantity ε, a maximum or a minimum is obtained. Then the second derivative d^2f/dx^2 is computed. A maximum is obtained if d^2f/dx^2 is negative. In this case, the computation is

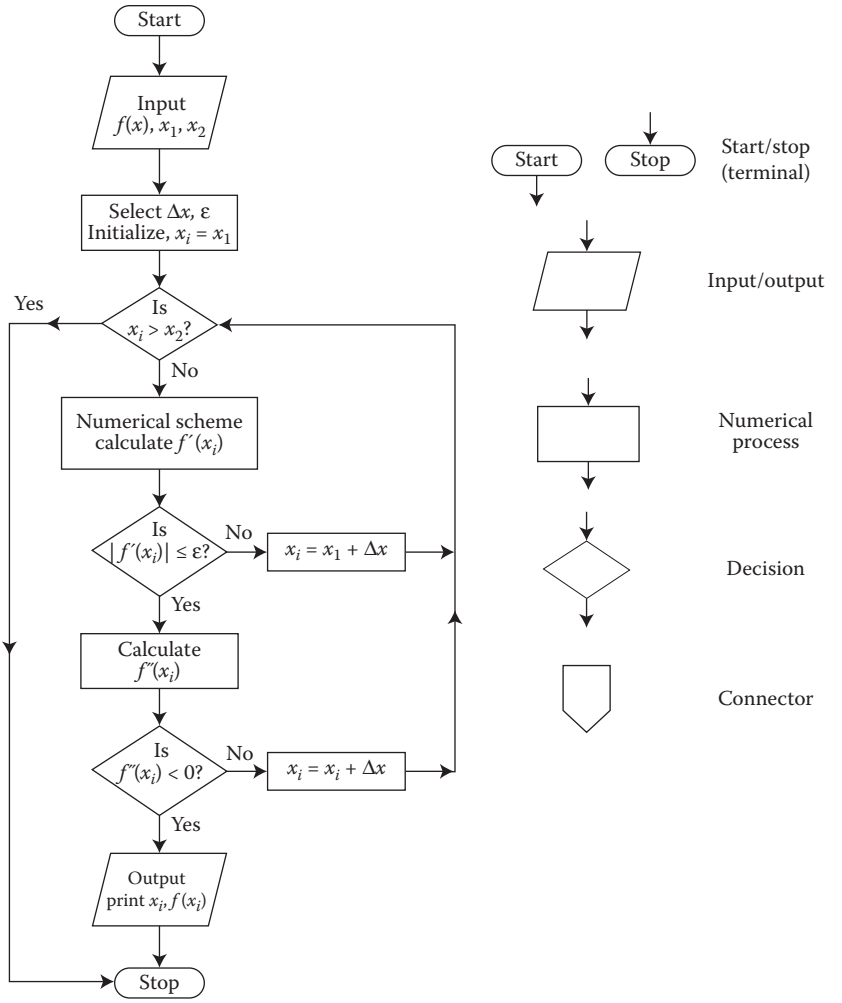


FIGURE 2.5 Flow chart representation of the algorithm outlined in Figure 2.4.

terminated and the output printed. However, if d^2f/dx^2 is positive, a minimum is indicated. A value of zero indicates a saddle or inflexion point. Then, the computation of df/dx is again carried out by advancing x until a maximum is obtained or until the upper limit on x (i.e., $x = x_2$) is attained. If a maximum is not obtained in the given domain and if $f(x)$ is known to have a maximum in the region, a larger value of ε may be selected and the procedure repeated. In fact, both ε and Δx must be varied to ensure that the location of the maximum is essentially independent of the values chosen.

As shown in Figures 2.4 and 2.5, a flow chart is a more convenient representation of an algorithm. The various symbols used for indicating the type or nature of a given step

are also shown in Figure 2.5. The flow chart is a useful tool as long as it is used to give an outline of the overall process and not the detailed representations of individual steps. The numbered sequence of steps, given in Figure 2.4, can also be used instead, depending on the personal preference of the programmer. However, with experience, one could form a mental picture of the various steps in the algorithm, particularly for relatively simple problems, and proceed directly to computer programming.

2.2.4.2 Available Programs

Along with improvements in computer systems in recent years, with respect to memory and computational speed, there has been an explosive growth in software as well. A question, which is frequently asked these days, is if there is a need to develop numerical codes when many general purpose and specialized codes are easily available in the public and commercial domains. General-purpose programs such as *Fidap*, *Fluent*, *Phoenics*, *Nekton*, and *Ansys* are commercially available and can easily be used to study a wide variety of engineering problems. Software such as *Maple*, *MathCAD*, and MATLAB can be used for obtaining analytical and numerical solutions to a variety of mathematical problems such as differential equations, integration, root solving, and algebraic equations. Similarly, specialized codes such as *Polyflow* for polymer processing can be employed for specific problems and applications. In the public domain, several codes are available free of cost. These include programs for solving systems of linear equations, for solving ODEs, for inverting matrices, for curve fitting, and for providing graphical outputs of the computational results.

Commercially available software is generally expensive and usually does not provide the source code so that it is difficult to make changes in the code for a specific problem. In many cases, information on the algorithm, accuracy, discretization, convergence characteristics, range of applicability, and other important aspects associated with the software is not available in adequate detail. Despite the claims made with respect to the wide variety of problems a given software is capable of solving, one must judge each program very carefully and choose the one most suitable for a given application, keeping its cost, versatility, accuracy, and other features in mind. However, the general-purpose programs are finding wide use in industry, usually with specific changes made in the software to address the requirements of the given industry.

Computer programs in the public domain do not have many of these concerns and can often be adapted to a given computer system and linked with other software to solve a given problem. Thus, a program for solving a system of linear equations by cyclic reduction, fast Fourier transforms, or matrix decomposition may be used as part of the overall computer code to simulate an engineering problem. Certainly, software packages for producing graphical outputs are extensively used with the computational scheme generating the results. This approach of developing the core software and linking it with codes available in the public domain is a particularly attractive approach and is widely used.

Besides the easy availability of a wide variety of computer codes in the public and commercial domains that have led to considerable improvements and simplifications in numerical model development for engineering processes, several other advancements have occurred in recent years. These are expected to continue to

have a significant impact on computational methods. Certainly, the most important development is that of parallel machines which employ several processors, instead of the single CPU used in traditional serial or sequential computing machines. As outlined in Section 2.2.5, multiple processors allow concurrent calculations to be carried out, resulting in a considerable speed up of the process. Similarly, considerable progress has been made in graphical representations of the results, employing color plots, contour plots, particle trajectories, two- and three-dimensional graphs, and vector field graphs, among other useful and interesting features.

The need to use supercomputers to solve complicated problems, such as those involving three-dimensional transport and turbulent flow, has led to improvements in computational techniques through vectorization of the variables, so that rather than treating each quantity in an array as a scalar the whole array is treated as a vector. Improvements in the user–computer interface, using languages such as *Visual Basic*, have also resulted in considerable ease in entering the relevant data such as geometry, operating conditions, and material characteristics. Information storage and retrieval, linking with the knowledge base on a given process or material, often using artificial intelligence techniques, and other new features in computer systems and software have had a considerable impact on traditional programming. It is expected that such advancement will continue in the future, resulting in valuable and desirable changes in the field of computational methods as well.

2.2.4.3 Validation

The final stage in the development of the computer program for solving a given problem is verification or validation of the numerical scheme. As discussed in Section 1.3, validation is done by a comparison of the numerical results with available analytical solutions and experimental results. However, the analytical solution of the problem being solved numerically is obviously not available, at least in a convenient form, making a numerical solution necessary. Therefore, the numerical scheme is generally validated by a comparison with the analytical solution available for simpler problems. For example, the algorithm shown in Figure 2.4 may be used with a simple analytic function whose maximum can easily be determined mathematically. Thus, a function such as $f(x) = 5 + 4x - 3x^3$, which can easily be shown to have a maximum at $x = 2/3$, may be chosen for the testing of the numerical scheme. The numerically obtained value may be compared with the analytical one to verify that the scheme is performing satisfactorily. Other, more complicated expressions may also be employed, if the corresponding analytical results are known, for the validation of the computer program. Similarly, experimental results are generally not available on the problem being solved. However, experimental data on similar systems or problems may be available. These data can then be used to validate the numerical solution.

2.2.5 Serial versus Parallel Computing

In this book, it is generally assumed that at a given instant only one computational step is being carried out on the computer. This assumption applies to most commonly used computers, such as PCs and minicomputers, for engineering calculations. The computational procedure in which the required calculations are performed sequentially, with

each step being undertaken by the machine after the previous one is over, is known as *serial* or *sequential* computing. Thus, a single CPU is involved in the computation. However, in recent years, computers with multiple processors that allow concurrent calculations have been developed. Generally termed *parallel computers*, these machines represent the new generation of computing and have become important in the numerical simulation of complicated processes and systems.

In order to fully utilize these machines with multiple processing units, one must write the algorithm so as to employ the feature of parallel computing. Thus, statements must be given to direct various calculation steps to different units. Algorithms in which different steps are independent of each other are ideally suited for parallel computing, since each calculation step can easily be assigned to a given processor. Algorithms that involve strongly coupled steps cannot be solved very efficiently with parallel computing. Besides the calculation for each step, the processors need to communicate with each other at various stages in order to solve the overall problem. Thus, parallel computing involves developing algorithms that allow concurrent calculations and message passing between processors for greater efficiency. Depending on the problem and the algorithm, a considerable speed up of the computation can be obtained for a system consisting of n processors, a value approaching n indicating an excellent utilization of the parallel computing environment. Even though the assumption here is serial or sequential computing, the implications for parallel computing will be given at many places in the book. For details on parallel computing, see Grama et al. (2003) and Scott et al. (2005).

Example 2.2

A firm needs to borrow \$50,000 to undertake improvements in its existing facilities. For the repayment of the loan, the firm wishes to pay only \$1000 each month, beginning at the end of the first month after taking the loan, toward the principal and the interest. Considering possible interest rates as 8%, 10%, and 12%, determine the time required to pay off the loan for these three cases. Calculate the time required and the *future worth* (FW), or the value on the day the repayment is completed, of the money paid toward the loan. Also, determine the amount by which the final payment must be reduced to pay off the loan exactly.

SOLUTION

Let x denote the percent interest rate, so that an annual compounding yields an interest of x on \$100. Then the annual interest on each dollar is $x/100$, denoted by x_1. Therefore, the FW of an amount P after n years is $P(1 + x_1)^n$, due to this interest which is compounded annually. Similarly, the *present worth* (PW), or the value today, of an amount R paid at the end of n years is $R/(1 + x_1)^n$. The concepts of PW and FW are very important in economic analysis; see, for instance, Stoecker (1989). First, we need to consider the PW of a series of uniform annual amounts R, paid at the end of each year starting at the end of the first year. If n is the total number of years, the PW of such a series of amounts is

$$\text{PW} = \frac{R}{(1 + x_1)^1} + \frac{R}{(1 + x_1)^2} + \frac{R}{(1 + x_1)^3} + \cdots + \frac{R}{(1 + x_1)^n} \tag{2.2}$$

The series can be summed up to give

$$PW = R\frac{(1 + x_1)^n - 1}{x_1(1 + x_1)^n} \tag{2.3}$$

where $x_1 = x/100$ (since x is given as a percent).

Equation 2.2 follows from the fact that the PW of an amount P paid at the end of n years is given by $PW = P/(1 + x_1)^n$ and from the consideration of each lump-sum annual payment to yield the given series. Now, if we consider monthly payments, the total number of payments become m, where $m = 12n$, and the interest rate becomes x_m, where $x_m = x/(12 \times 100)$. Thus,

$$PW = R\frac{(1 + x_m)^m - 1}{x_m(1 + x_m)^m} \tag{2.4}$$

The FW of this series of amounts is obtained by simply multiplying the PW by $(1 + x_m)^m$. Therefore,

$$FW = R\frac{(1 + x_m)^m - 1}{x_m} \tag{2.5}$$

Now, R is given as \$1000 and x as 8%, 10%, or 12%. We wish to compute the time, in months m, needed to repay the loan, and the FW of the total payment. The PW is \$50,000. Thus, m is to be computed from Equation 2.4, and the FW may then be obtained from Equation 2.5. The determination of m from Equation 2.4 is a root-solving problem, which will be presented in Chapter 5. Here, we shall use a very simple approach, since root-solving methods have not been discussed yet. For a given value of x_m, the value of m may be increased in steps of 1, starting with $m = 1$, and the PW computed from Equation 2.4, until the value of \$50,000 is reached. The computation stops when PW exceeds this amount, since a fixed payment of \$1000 is made each month. In practice, the monthly payment is adjusted to an appropriate value close to \$1000, so that the loan is paid off exactly.

Figure 2.6 shows the algorithm to be employed, in terms of a flow chart. The computational scheme is very simple for this problem and is based on a comparison between the PW of \$50,000 and the sum of the series in Equation 2.4, employing an increasing number of terms m. Once the latter exceeds the PW, the loan is paid off and the number of months needed is printed. Also, the FW, on the date when the loan is paid off, of the total payment made is computed from Equation 2.5. The PW of the total payment exceeds \$50,000, and the last payment may be reduced to avoid this excess payment or the monthly payments may be adjusted, as mentioned above. The FW of the loan is \$50,000 $(1 + x_m)^m$, and if this amount is subtracted from the computed FW of the payments, we obtain the amount by which the final payment may be reduced to pay off the loan exactly.

A computer program may easily be developed on the basis of this algorithm. Figure 2.7 presents a Fortran 77 program to give the logic and the various steps indicated in the algorithm.

Figure 2.8 presents the numerical results obtained from such a program. The inputs are entered and the print out gives the results, along with the input parameters to ensure that the correct values are being employed in the calculations. As seen here, the number of months needed to repay the loan increases with

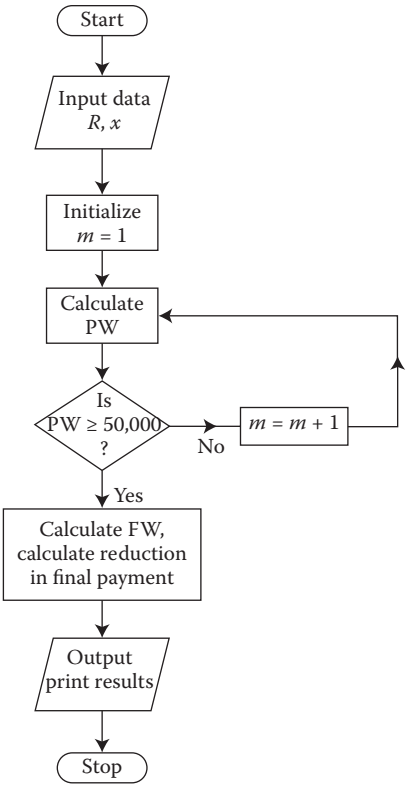


FIGURE 2.6 Flow chart for the problem in economics considered in Example 2.2.

the interest rate, as expected. Also, the FW increases. Note also that, since the monthly payment is kept constant, the total payment is more than the loan. To determine the amount needed to repay the loan exactly, subtract the FW of the loan from the FW of the total payment. This amount is the overpayment and is subtracted from the last month's payment of $1000 to obtain the reduction in the final payment if the loan is to be paid off exactly.

2.3 NUMERICAL ERRORS AND ACCURACY

A very important consideration in the solution of a given mathematical, chemical, physical, or engineering problem by computational methods is the accuracy of the numerical results obtained. The true measure of inaccuracy, or error, in the numerical solution is the difference between the numerical and the exact, or analytical, results. However, the analytical solution of the given problem is presumably not available, making it necessary to solve it numerically. Thus, alternative methods for estimating the errors involved and the accuracy of the numerical solution are needed. The dependence of the errors on the various parameters associated with the numerical procedure must also be determined, so that the accuracy of the solution may be improved by varying these parameters.