Numerical Analysis in Engineering

Revised Edition

R.B. Bhat
S. Chakraverty

Alpha Science

Numerical Analysis in Engineering

Revised Edition

Rama B. Bhat
Snehashish Chakraverty

Alpha Science International Ltd.
Oxford, U.K.

Rama B. Bhat
Department of Mechanical Engineering
Faculty of Engineering and Computer Science
Concordia University, Montreal
Quebec, Canada

Snehashish Chakraverty
Computer Centre
Central Building Research Institute
Roorkee, India

Revised Edition 2007

Alpha Science International Ltd.
7200 The Quorum, Oxford Business Park North
Garsington Road, Oxford OX4 2JZ, U.K.

www.alphasci.com

ISBN 978-1-84265-402-6

Printed in India

Numerical Analysis in Engineering

Revised Edition

Preface

Numerical Methods as a subject has not changed very much over the years. The book is not much different from other books available at present in the market. However, our approach in the presentation of the material is simple, providing numerous examples. This will serve as a very good textbook for any undergraduate course on numerical methods. The first nine chapters of this book originally evolved out of the course notes prepared by the first author over a period of several years in Concordia University, starting from 1980 to 1987. Chapter 10 on Partial Differential Equations was added by a colleague, Dr. Gerard Gouw, who also helped in putting the notes together in word processor. The two of us published this version as Course Notes at that time, which was published by Simon and Schuster Custom Publishing, and this has been used in Concordia University for several years.

The second author, Dr. S. Chakraverty, visited Concordia University in 1999 as a Research Associate, and proposed to add some material on the Boundary Characteristic Orthogonal Polynomials and Solution of Boundary Value Problems. The original course notes were extensively revised by both the authors, and the present book evolved.

Undergraduate programs in engineering have several courses that can benefit from a knowledge of the numerical analysis techniques such as circuit analysis, electricity, fluid mechanics, heat transfer, structural analysis, network analysis, mechanical vibrations. Example problems illustrating the methods in many of these areas are provided and discussed.

We acknowledge the helpful comments from our colleagues as well as the students. Thanks are particularly due to Mr. Xiangyu Xie who helped in preparing the manuscript in the final form.

RAMA B. BHAT
SNEHASHISH CHAKRAVERTY

Table of Contents

CHAPTER 1

Solution of Equations for Engineering Design and Analysis

1.1 Introduction

In order to implement any physical systems that engineers conceive in their minds, they should verify whether such physical systems would do their intended operations satisfactorily in the operating environments. This is normally done by developing a mathematical model for the physical system, subjecting them to external loads that they would be subjected to during their operation and ensuring the systems would be stable and perform their intended tasks in the operating environments. The conditions of stable operation under operating environments may be expressed in the form of system of simple equations, system of differential equations or system of integral equations. When it is difficult, cumbersome or sometimes impossible to solve these equations in closed form engineers resort to numerical techniques to solve them.

With the advent of electronic computers, the numerical solutions to engineering problems cast in the form of mathematical equations have become simpler. The accuracy of such numerical solutions also can be improved by systematic applications of the numerical techniques or algorithm, in an iterative form.

1.2 Taylor's Series Expansion of Functions

For satisfactory operation under operating environments, physical systems satisfy equilibrium conditions, which can be expressed in the form of simple equations, differential equations or integral equations. In any of these forms, a function can be expressed in its neighborhood, if we know the function value and their higher derivatives at a given point, using Taylor's series expression about the point.

The Taylor's series is the foundation of numerical methods. Many of the numerical techniques are derived directly from these series.

Taylor's series obtains the function in the form of a polynomial in the neighborhood of the known point. The expansion is given by

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2!}f''(a) + \ldots + \frac{(x-a)^n}{n!}f^{(n)}(a) + \ldots \quad (1.1)$$

If the estimation of the function at a point b which is fairly close to a is desired, it is given by

$$f(b) = f(a) + (b-a)f'(a) + \frac{(b-a)^2}{2!}f''(a) + \ldots + \frac{(b-a)^n}{n!}f^{(n)}(a) + \ldots \quad (1.2)$$

These are infinite series. In practical applications the series can be terminated after a few terms, depending on the distance of point b from a. If b is very close to a, only a few terms will give a good estimation.

The error in Taylor's series for $f(x)$ when the series is terminated after the term containing $(x-a)^n$ is not greater than

$$|f^{(n+1)}|_{\max} \cdot \frac{(|x-a|)^{(n+1)}}{(n+1)!} \quad (1.3)$$

where max denotes the maximum magnitude of the derivative in the interval a to x. However, $f^{(n+1)}$ itself is not known if the function $f(x)$ is not known. This frustrating state of affairs is common in numerical analysis. If the expansion is used to obtain the estimate of the function for very small values of $(x-a)$, then the error also comes down. If the series is truncated after n terms, we can say that $f(x)$ is accurate to $0(x-a)^n$.

Exampel 1.2.1

Determine (a) sinh (0.9) and (b) cosh (0.9) to $0(0.9)^4$.

Solution

$$\sinh(x) = \frac{e^x - e^{-x}}{2}; \qquad \cosh(x) = \frac{e^x + e^{-x}}{2};$$

$$\sinh(0) = 0 \qquad \cosh(0) = 1$$
$$\sinh'(0) = 1 \qquad \cosh'(0) = 0$$
$$\sinh''(0) = 0 \qquad \cosh''(0) = 1$$
$$\sin'''(0) = 1 \qquad \cos'''(0) = 0$$

$$\text{(a) } \sinh(0.9) = \sinh(0) + 0.9\sinh'(0) + \frac{0.9^2}{2!}\sinh''(0) + \frac{0.9^3}{3!}\sinh'''(0)$$

$$= 0 + 0.9 + 0 + \frac{0.9^3}{6!} = 1.0215$$

(exact: 1.0265....)

$$\text{(b) } \cosh(0.9) = \cosh(0) + 0.9\cosh'(0) + \frac{0.9^2}{2!}\cosh''(0) + \frac{0.9^3}{3!}\cosh'''(0)$$

$$= 1 + \frac{0.9^2}{2} = 1.405$$

(exact: 1.4331....)

Example 1.2.2

Consider one more non-zero term in the above expansions.

Solution

(a) $$\sinh(0.9) = 0 + 0.9 + 0 + \frac{0.9^3}{6} + 0 + \frac{0.9^5}{5!} = 1.0264$$

(exact: 1.0265....)

(b) $$\cosh(0.9) = 1 + \frac{0.9^2}{2} + 0 + \frac{0.9^4}{4!} = 1.4320$$

(exact: 1.4331...)

Example 1.2.3

Given $f(1.8) = -1.1664$ and $f'(1.8) = 3.888$. Find out x when $f(x) = 0$.

Solution

$$f(x) = f'(a)(x - a)$$

$$0 = -1.1664 + 3.888(x - 1.8)$$

Hence, $$x = 1.8 + \frac{1.1664}{3.888} = 2.1$$

(The example is based on the function $f(x) = x^4 - 2x^3$, which has a root at $x = 2$)

Example 1.2.4

Given the differential equation $\frac{df}{dx} = 2x$ with $f(1) = 1$ obtain $f(1.2)$.

Solution

$$f(x) = f(1) + f'(1)(x - 1) = f(1) + 2(x - 1) + 2(x - 1)^2/2!$$

$$f(1.2) = 1 + 2(1.2 - 1) + 2(1.2 - 1)^2/2 = 1.44$$

1.3 Digital Computers

There exist basically two types of computers: (1) digital and (2) analog. The analog computer generates a continuous output signal when it is given a continuous input signal. This type of computer is popular in control systems; however, it has little value when number crunching is the main concern as in numerical methods. The digital computer deals with numbers in the form of

finite number of digits. A sequence of simple binary operations, such as on or off, assigned with digits 1 and 0, respectively, can represent numbers to any number of digits of accuracy. The digital computer, like the calculator, can perform arithmetic operations; however, unlike a normal non-programmable calculator, it can perform several operations using data from previous operations if necessary, without help from the user. In other words, the digital computer has the ability to make decisions based on how it is programmed, e.g. if the computer is instructed by the program to compare two numbers A and B, then depending on whether A is less than, greater than, or equal to B, the computer will take one of three entirely different paths for subsequent operations. The computer also has the ability to store previous calculations for future use if they are needed.

The most important characteristic of the digital computer is its high speed. By employing simple routines, calculations that previously took days and weeks, can now be performed in seconds or minutes.

The major contribution which has revolutionized the use of computers is the development of a memory unit. The memory unit is capable of storing a few hundred to several thousand numbers. Commands can also be stored in the unit in a similar fashion.

A computer system consists of five major components: (1) input device, (2) a memory unit, (3) a control unit, (4) an arithmetic logic unit, and (5) output devices. These five components are either combined into one unit or operate as individual units (see Fig. 1.3.1).

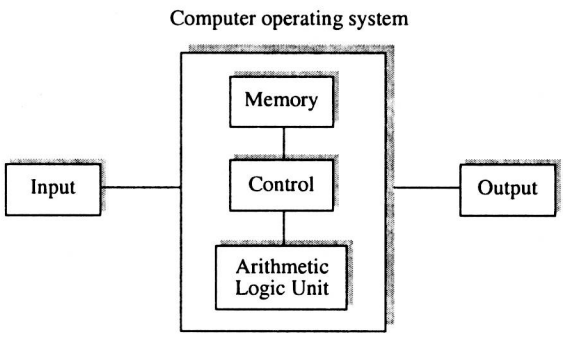


Figure 1.3.1

The input unit serves as a device to feed data and commands into the memory unit. This is achieved by components such as keyboards, punch cards, magnetic disk, or magnetic tape. Magnetic disk is the fastest of the above mentioned.

The memory unit, control unit, and arithmetic logic unit are referred to as the Central Processing Unit or C.P.U. This is the unit which will, through the use of a compiler program, open and close various electronic switches or gates, allowing data to pass from one part of the machine to another. The C.P.U. is also responsible for making logical decisions and performing arithmetic operations.

The output devices, such as printers, plotters, magnetic disks, video screens, or magnetic tape units, serve to display the computed data outside the machine.

The units described so far are known as hardware. Hardware are the actual physical components of the computer, whereas software are the programs or instructions fed to the computer. Notice that software has no physical substance, it is just a set of instructions. This course will deal only with software.

1.4 Number Representation: Floating Point and Fixed Point

When doing calculations on paper, the decimal point can be kept track of quite easily after each calculation. However, when using a computer, where several operations are performed internally before a final result is displayed, there are two techniques in which the decimal point is taken care of.

The first method is the fixed-point method which is seldom used. This is due to the inconvenience of the user having to keep track of decimal places in the input and output. The second and much more accepted method employed in the computer hardware is the floating point method.

In this method the computer represents a number in the following form.

$$\text{Number} = S \times M \times B^k$$

S = sign (+ or –)
M = mantissa, a value which lies between 0.1 and 1
B = base (B = 10 for decimal computers; B = 2 for binary computers)
k = exponent
To illustrate the above formula it is convenient to consider a simple example.

Example 1.4.1
Consider a familiar number such as Young's modulus:

$$E = 30{,}000{,}000 \text{ psi}, N = +0.3 \times 10^8 \text{ psi}$$

Notice that the mantissa is always between 0.1 and 1. Also notice that this would be a decimal computer since the number is represented with base 10.

If an operation is carried out between two numbers in this form, zeros might appear before or after the decimal point. This is taken care of by the computer by normalizing the result.

Example 1.4.2

$$\begin{array}{r} 0.5789875 \times 10^7 \\ \underline{-\,0.5778764 \times 10^7} \\ 0.0011111 \times 10^7 \end{array}$$

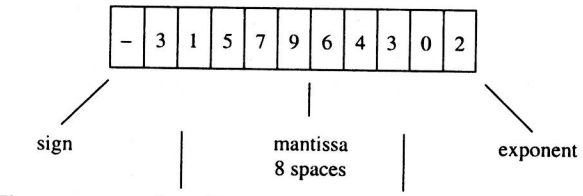
Normalizing this for the next operation would be to put it into the form:

$$0.11111 \times 10^5$$

So far the way in which a number is handled by the decimal based computer has been shown. For simplicity assume that we have a decimal computer with a storage space available for 10 digits plus a sign. In such a case the number:

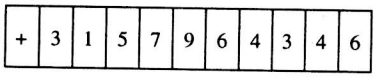
$N = -31.579643$ would be represented by

$N = -0.31579643 \times 10^2$ which would be stored as



The computer handles a negative exponent by adding 50 to it.

$N = 0.31579643 \times 10^{-4}$ this would be stored as



The computer knows that 46 represents an exponent of – 4.

At this point it must be noted that the available amount of storage space is 8 and therefore the precision of the machine is also 8. Suppose the mantissa has 12 digits and the computer precision is only 8 as above, or two numbers with 8 digits are multiplied producing a number 15 digit long. Some questions that would arise are how accurate are the results after 100 such multiplications and how does the computer store the result in its limited memory space.

The machine precision dictates the amount of accuracy of the result.

1.5 Algorithms and Flowcharts

An algorithm is a set of step-by-step logical instructions that are to be followed to reach a goal efficiently. Some examples of an algorithm would be a recipe for cooking a dish or instructions for building a house.

An algorithm is the step-by-step thought process that is used to solve a problem. Algorithms should possess the following qualities:

(1) They should be precise: This means that if an iterative process is being performed that permits a relative error of 0.001, the operation should continue until this condition is satisfied, and not go farther.

(2) They should be finite: The process should terminate after a finite number of steps.

(3) They should be effective: This means that the problem should be solved efficiently. For instance computer time is expensive when running a large problem. If the program is well structured it will be effective.

Algorithms consist of three major parts, some input, a process, and resulting output. A flowchart is a pictorial representation of an algorithm. For example the flowchart representing the algorithm to machine a component is shown below:

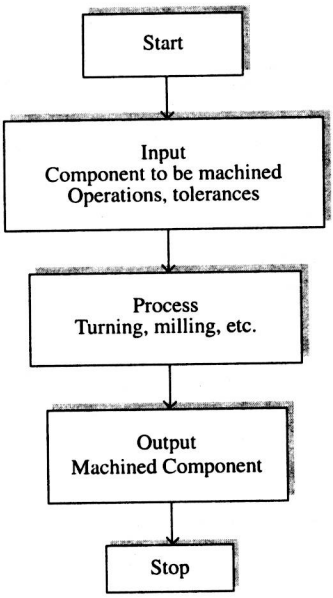


Figure 1.5.1

A logical approach to solving a typical engineering problem in real life is outlined below:

Step 1 Formulate a mathematical model for the given problem. From this model certain mathematical equations can be developed.

Step 2 Select a numerical method suitable for solving the mathematical equations. An algorithm should be developed at this stage.

Step 3 Draw a detailed flowchart. Each block of the flowchart should represent a major section of the program. Make the flow chart as detailed as possible.

Step 4 Write the codes, which is another name for computer program, in a suitable computer language. Each block of the flowchart should be represented by its own section of code. Often subroutines are employed to show the interdependence between two blocks of the flowchart. Subroutines or subprograms are an intricate part of structured programming.

Step 5 Run the program on the digital computer. The first time the program is run, it is bound to have errors, whether it be in the actual logic, or just typing errors (referred to as syntax errors). The process of recognizing and correcting these errors is called debugging. If the program is well structured, with appropriate subroutines used throughout the program, it is usually easy to locate the error in the program. Use of subroutines makes it much easier for someone other than the programmer to understand the program. Subroutines can

also be called at any point in the program so that calculations of the same type can be performed without writing new codes.

Once the general location of the error is found, print statements can be used to indicate what the program is doing incorrectly. This type of procedure is called tracing.

Step 6 After the program is completely debugged, make the final run.

1.6 Error Considerations

There is an inherent form of error in calculators or digital computers because they perform mathematical operations with only a finite number of digits. Whenever such approximation is involved, we would like to know what the extent of approximation is. The difference between the exact value and the approximate value obtained in its place is the error. A typical number representation system in computers is in the normalized decimal form.

$$\pm d_1 d_2 d_3 \ldots d_k \times 10^n$$

$$1 \leq d_1 \leq 9, \qquad 0 \leq d_k \leq 9$$

The fractional part $\pm d_1 d_2 d_3 \ldots d_k$ is called the mantissa and the exponential part is called the characteristic. Different types of computers can handle different values of k and n.

Assume the actual number is of the form

$$y = \pm d_1 d_2 d_3 \ldots d_k \, d_{k+1} \, d_{k+2} \ldots \times 10^n$$

If a given computer can retain the mantissa part up to k decimal places only, it will have to discard the values in decimal places d_{k+1} and above. If the quantities d_{k+1} and above are just dropped off, it is referred to as "chopping". If d_{k+1} is 5 or larger than 5, then it is more meaningful to add a unit value to d_k and then drop off d_{k+1} and above. This latter procedure is called rounding-off. If d_{k+1} is less than 5, then both chopping and rounding procedures will give the same result.

Example 1.6.1

Round off $y = 1/3$ to 6 decimal places in the normalized decimal form.

$$y = 1/3 = 0.3333333 \times 10^0$$

Since $d_7 < 5$ we just drop off d_7 and above. The result is:

$$y = 0.333333 \times 10^0$$

Example 1.7.2

Round off $y = 2/3$ to 6 decimal places in the normalized decimal form.

$$y = 2/3 = 0.6666666 \ldots \times 10^0$$

Since $d_7 > 5$, we add 1 to d_6 and the result is:

$$y = 0.666667 \times 10^0$$

Absolute and Relative Errors

If p is the exact value required and by following a numerical solution method, we obtain an approximation p^* to p, then

$$\text{Absolute error, } \xi_a = |\, p - p^* \,|$$

and

$$\text{Relative error, } \xi_r = \frac{|\, p - p^* \,|}{|\, p \,|}$$

$$\text{Provided } p \neq 0$$

Example 1.6.3

What are the absolute and relative errors involved if $y = 2/3$ is represented in normalized decimal form with 6 digits.

(i) by chopping (also called truncation)
(ii) by rounding off

Solution

(i) $y = 2/3 = 0.666666 \times 10^0$

$$\begin{aligned}\text{Absolute error, } \xi_a &= 2/3 - 0.666666 \times 10^0 \\ &= (0.6666666\cdots - 0.666666) \times 10^0 \\ &= 0.000000666\ldots \times 10^0 \\ &= 0.666\ldots \times 10^{-6}\end{aligned}$$

$$\text{Relative error, } \xi_r = \frac{(0.6666666666\ldots - 0.666666) \times 10^0}{0.66666666\ldots \times 10^0}$$

$$= \frac{0.666666666\ldots \times 10^0}{0.666666666\ldots \times 10^0} = 1 \times 10^{-6}$$

(ii) $y = 2/3 = 0.666667 \times 10^0$

$$\text{Absolute error, } \xi_a = |\, (0.6666666\ldots - 0.666667) \times 10^0 \,|$$

$$= 0.00000033\ldots \times 10^0 = 0.33 \times 10^{-7}$$

$$\text{Relative error, } \xi_r = \frac{|(0.6666666\ldots - 0.6666667) \times 10^0|}{|\, 0.66666666\ldots \times 10^0|}$$

$$= \frac{0.3333333\ldots \times 10^{-6}}{0.6666666\ldots \times 10^0} = \frac{1/3 \times 10^{-6}}{2/3 \times 10^0}$$

$$= 0.5 \times 10^{-6} = 5 \times 10^{-7}$$