31

NUMERICAL METHODS

31.1 INTRODUCTION

Chapter 31 is concerned with Numerical Methods. This subject was taught in the past as a means of providing engineers with ways to solve complicated mathematical expressions that they could not solve otherwise. However, with the advent of computers, these solutions have become readily obtainable.

A brief overview of Numerical Methods is given to provide the practicing engineer with some insight into what many of the currently used software packages (MathCad, Mathematica, MatLab, etc.) are actually doing. The authors have not attempted to cover all the topics of Numerical Methods. There are several excellent texts in the literature that deal with this subject matter in more detail.[1,2]

Ordinarily, discussion of the following eight numerical methods would be included in this chapter:

1. Simultaneous linear algebraic equations.
2. Nonlinear algebraic equations.
3. Numerical integration.
4. Numerical differentiation.
5. Ordinary differential equations.
6. Partial differential equations.
7. Regression analysis.
8. Optimization.

Fluid Flow for the Practicing Chemical Engineer. By J. Patrick Abulencia and Louis Theodore

However, because of the breadth of the subject matter, the reader should note that only three numerical methods receive treatment in the chapter. They are the first three topics listed above. The remaining five methods are to be found in the literature.[3,4] It should be noted that the problems section contains fluid flow material dealing with all eight subject topics with solutions available for those who adopt the text for classroom/training purposes.

31.2 EARLY HISTORY

Early in one's career, the engineer/scientist learns how to use equations and mathematical methods to obtain exact answers to a large range of relatively simple problems. Unfortunately, these techniques are often inadequate for solving real-world problems. The reader should note that one rarely needs exact answers in technical practice. Most real-world applications are usually inexact because they have been generated from data or parameters that are measured, and hence represent only approximations. What one is likely to require in a realistic situation is not an exact answer but rather one having reasonable accuracy from an engineering point of view.

The solution to an engineering or scientific problem usually requires an answer to an equation or equations, and the answer(s) may be approximate or exact. Obviously an exact answer is preferred but because of the complexity of some equations, often representing a system or process, exact solutions may not be attainable. For this condition, one may resort to another method that has come to be defined as a numerical method. Unlike the exact solution, which is continuous and in closed form, numerical methods provide an inexact (but reasonably accurate) solution. The numerical method leads to discrete answers that are almost always acceptable.

The numerical methods referred to above provide a step-by-step procedure that ultimately leads to an answer and a solution to a particular problem. The method usually requires a large number of calculations and is therefore ideally suited for digital computation.

High-speed computing equipment has had a tremendous impact on engineering design, scientific computation, and data processing. The ability of computers to handle large quantities of data and to perform mathematical operations at tremendous speeds permits the examination of many more cases and more engineering variables than could possibly be handled on the slide rule—the trademark of engineers of yesteryear. Scientific calculations previously estimated in lifetimes of computation time are currently generated in seconds and, in many instances, microseconds.[5]

A procedure-oriented language (POL) is a way of expressing commands to a computer in a form somewhat similar to such natural languages as English and mathematics. The instructions that make up a program written in a POL are called the source code. Because the computer understands only machine language or object code, a translator program must be run to translate the source code into an object

code. In terms of input, processing, and output, the source code is the input to the translator program, which processes (translates) the code. The output is the object code. It is the object code that is actually executed in order to process data and information.

The first POL to be widely used was FORTRAN, an acronym that was coined from the words "FORmula TRANslation." FORTRAN was designed initially for use on problems of a mathematical nature and it is still used for solving some problems in mathematics, engineering, and science.

PASCAL is a POL designed by Niklaus Wirth in 1968. The motivation behind its design was to provide a language that encouraged the programmer to write programs according to the principles of structured programming. An important aspect of the PASCAL design philosophy is that it is a "small" language. The purpose of this is to provide the programmer with a language that can be easily learned and retained. PASCAL provides a variety of data structuring that enable programmers to easily define new data types. PASCAL is also most commonly used in mathematics, engineering and science.

BASIC is an acronym for Beginner's All-Purpose Symbolic Instruction Code. J. G. Kemeny and T. E. Kurtz developed BASIC in 1967 to give students a simple language for learning programming. BASIC is an interactive language, that is, the programmer sees an error or output as soon as it occurs. The simplicity of BASIC makes it easy to learn and use. Many versions of BASIC have been written since the late 1960s. BASIC can be used effectively for a variety of business and scientific applications.

Two types of translator programs—compilers and interpreters—are used to convert program statements to a machine-readable format. A compiler first translates the entire program to machine language. If any syntax or translation errors are encountered, a complete listing of each error and the incorrect statement is given to the programmer. After the programmer corrects the errors, the program is compiled again. When no errors are detected, the compiled code (object code) can be executed. The machine-language version can then be saved separately so that the compiling step need not be repeated each time the program is executed unless the original program is changed. Compiled programs run much faster than the interpreted ones. An interpreter translates and executes one source code instruction of a program at a time. Each time an instruction is executed, the interpreter uses the key words in the source code to call pre-written machine-language routines that perform the functions specified in the source code. The disadvantage of an interpreter is that the program must be translated each time it is executed.

Today, many powerful commercial mathematical applications are available and widely used in academia and industry. Some of these programs include MathCad, Matlab, Mathematica, etc. These user-friendly programs allow engineers and scientists to perform mathematical calculations without knowing any programming. In addition, new programs, for example, Visual Basic.NET, JAVA, C++, etc., are constantly evolving.

31.3 SIMULTANEOUS LINEAR ALGEBRAIC EQUATIONS

The engineer often encounters problems that not only contain more than two or three simultaneous algebraic equations but also those that are sometimes nonlinear as well. There is therefore, an obvious need for systematic methods of solving simultaneous linear and simultaneous nonlinear equations.[1] This section will address the linear sets of equations; information on nonlinear sets is available in the literature.[6]

Consider the following set of n equations:

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = c_1 \\ &a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = c_2 \\ &\cdots \\ &\cdots \\ &a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = c_n \end{aligned} \tag{31.1}$$

where a is the coefficient of the variable x and c is a constant. The above set is considered to be linear as long as none of the x-terms are nonlinear, for example, x_2^2 or $\ln x_1$. Thus, a linear system requires that all terms in x be linear.

A system of linear algebraic equations may be set in matrix form:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \cdots \\ c_n \end{bmatrix} \tag{31.2}$$

However, it is often more convenient to represent Equation (31.2) in the *augmented matrix* provided in Equation (31.3)

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & c_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & c_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & c_n \end{bmatrix} \tag{31.3}$$

Methods of solution available for solving these linear sets of equations include:

1. Gauss–Jordan reduction.
2. Gauss elimination.
3. Gauss–Seidel.
4. Cramer's rule.
5. Cholesky's methods.

Only the first three methods are discussed in this section.

31.3.1 Gauss–Jordan Reduction

Carnahan and Wilkes[1] solved the following two simultaneous equations using the Gauss–Jordan reduction method

$$3x_1 + 4x_2 = 29 \tag{31.4}$$

$$6x_1 + 10x_2 = 68 \tag{31.5}$$

The four step procedure is provided below.
Step 1. Divide Equation (31.4) through by the coefficient of x_1:

$$x_1 + \frac{4}{3}x_2 = \frac{29}{3} \tag{31.6}$$

Step 2. Subtract a suitable multiple (6, in this case) of Equation (31.6) from Equation (31.5), so that x_1 is "eliminated". Equation (31.6) remains intact so that what remains is:

$$x_1 + \frac{4}{3}x_2 = \frac{29}{3} \tag{31.7}$$

$$2x_2 = 10 \tag{31.8}$$

Step 3. Divide Equation (31.8) by the coefficient of x_2, that is, solve Equation (31.8).

$$x_2 = 5 \tag{31.9}$$

Step 4. Subtract a suitable factor of Equation (31.9) from Equation (31.7) so that x_2 is eliminated. When $(4/3)x_2 = 20/3$ is subtracted from Equation (31.7), one obtains

$$x_1 = 9 \tag{31.10}$$

31.3.2 Gauss Elimination

Gauss elimination is another method used to solve linear sets of equations. This method utilizes the augmented matrix described in Equation (31.3). The goal with Gauss elimination is to rearrange the augmented matrix into a "triangle form" where all the elements below the diagonal are zero. This is accomplished in much the same way as in Gauss–Jordan reduction. The procedure employed follows. Start with the first equation in the set. This is known as the pivot equation and will not change throughout the procedure. Once the matrix is in triangle form, back substitution can be used to solve for the variables. The Gauss elimination algorithm can be found in Figs. 31.1–31.3.

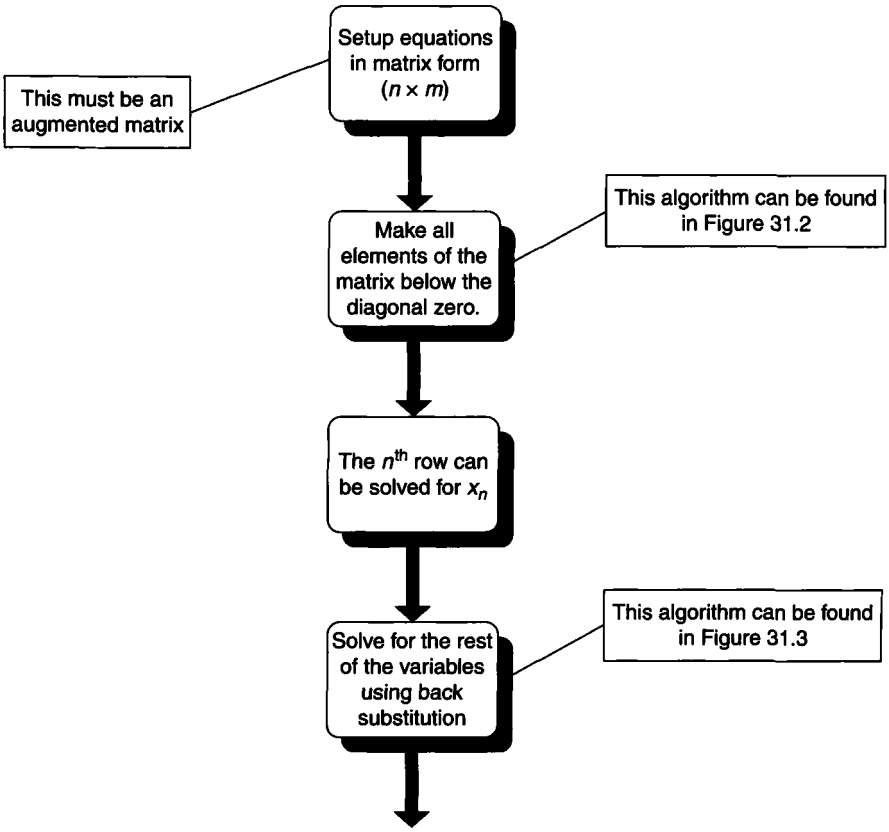

Figure 31.1 Solving a system of linear equations with Gauss elimination.

Illustrative Example 31.1 Solve the following set of linear algebraic equations using Gauss elimination.

$$\begin{aligned} 3x_1 - 2x_2 + 1x_3 &= 7 \\ x_1 + 4x_2 - 2x_3 &= 21 \\ 2x_1 - 3x_2 - 4x_3 &= 9 \end{aligned}$$

Solution First, setup the augmented matrix

$$\begin{bmatrix} 3 & -2 & 1 & 7 \\ 1 & 4 & -2 & 21 \\ 2 & -3 & -4 & 9 \end{bmatrix}$$

In order to convert the first column of elements below the diagonal to be zero, start by setting the first row as the pivot row. For the remaining rows, multiply each element

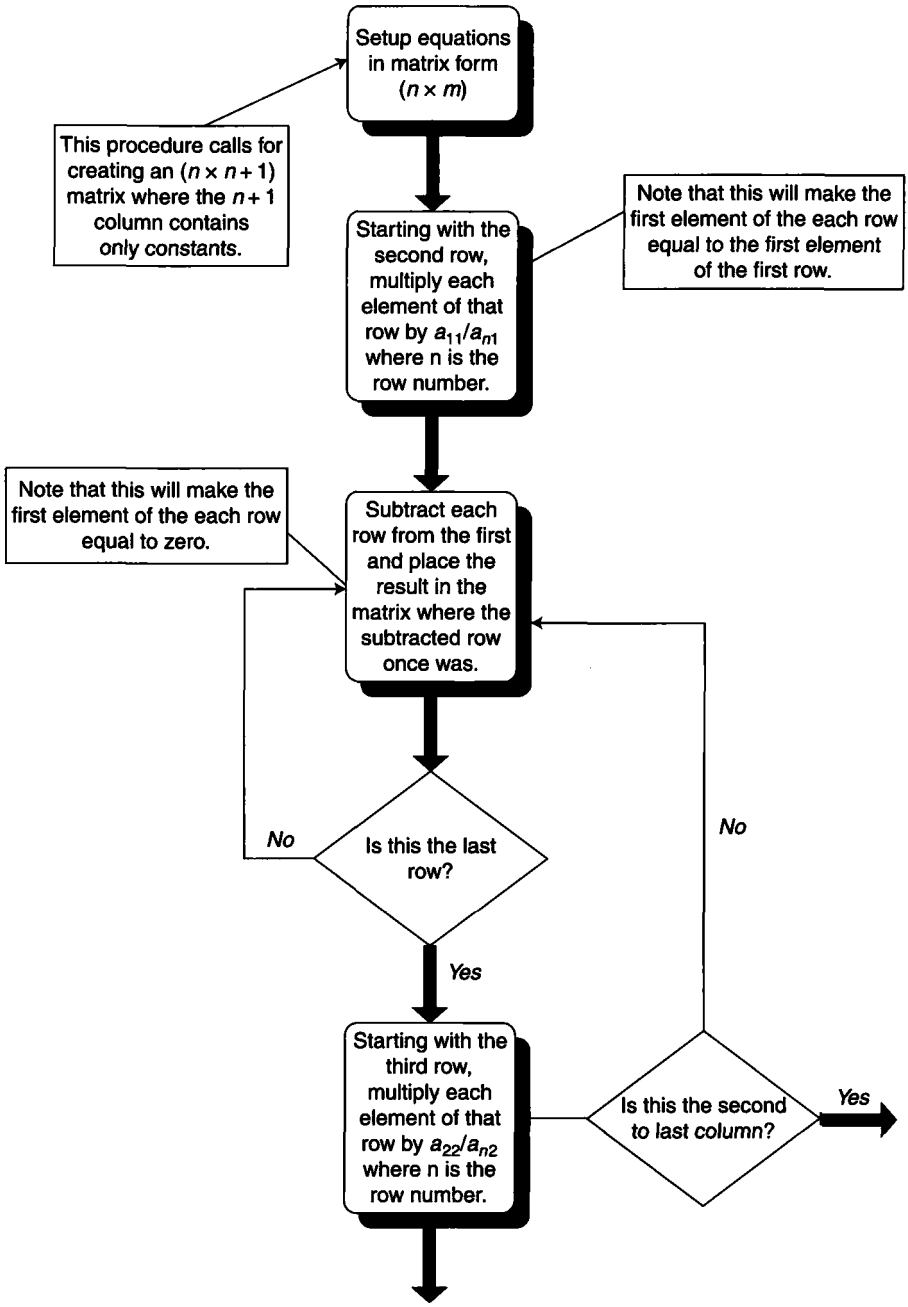

Figure 31.2 Algorithm for Gauss elimination.

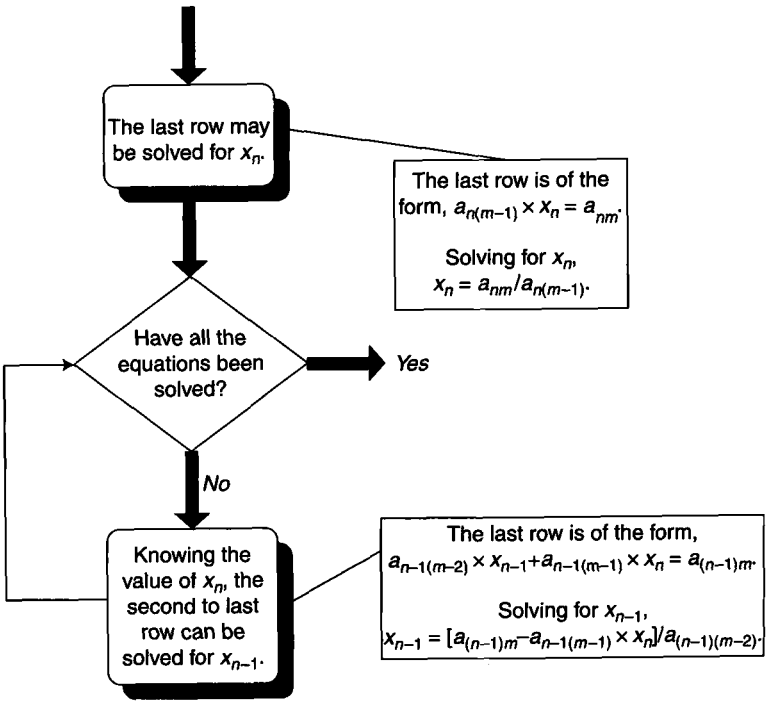

Figure 31.3 Algorithm for Gauss elimination back substitution.

by a_{11}/a_{n1} and subtract each row from the pivot row. Thus:

Row 2:

$$-(3)(1) + 3 = 0$$
$$-(3)(4) - 2 = -14$$
$$-(3)(-2) + 1 = 7$$
$$-(3)(21) + 7 = -56$$

Row 3:

$$-\frac{(3)(2)}{2} + 3 = 0$$
$$-\frac{(3)(-3)}{2} - 2 = 2.5$$
$$-\frac{(3)(-4)}{2} + 1 = 7$$
$$-\frac{(3)(9)}{2} + 7 = -6.5$$

The resulting matrix is:

$$\begin{bmatrix} 3 & -2 & 1 & 7 \\ 0 & -14 & 7 & -56 \\ 0 & 2.5 & 7 & -6.5 \end{bmatrix}$$

Now starting with the second column and the second row, perform the same procedure again, with the second row as the pivot row. Thus:

Row 3:

$$-\frac{(3)(2)}{2} + 3 = 0$$
$$-\frac{(3)(-3)}{2} - 2 = 2.5$$
$$-\frac{(3)(-4)}{2} + 1 = 7$$
$$-\frac{(3)(9)}{2} + 7 = -6.5$$

This produces the following matrix:

$$\begin{bmatrix} 3 & -2 & 1 & 7 \\ 0 & -14 & 7 & -56 \\ 0 & 0 & 46.2 & -92.4 \end{bmatrix}$$

At this point, back substitution may be employed. The following are the results:

$$46.2x_3 = -92.4; \quad x_3 = -2$$
$$-14x_2 + (-2)(7) = -56; \quad x_2 = 3$$
$$3x_1 + (-2)(3) + (1)(-2) = 7; \quad x_1 = 5$$

Gauss elimination is useful for systems that contain fewer than 30 equations. Systems larger than 30 equations become subject to round-off error where numbers are truncated by computers performing the calculations.

31.3.3 Gauss–Seidel

Another approach to solving an equation or series/sets of equations is to make an "informed" or "educated" guess. If the first assumed value(s) does not work, the value is updated. By carefully noting the influence of these guesses on each variable, these answers or correct set of values for a system of equations can be approached. The reader should note that when this type of iterative procedure is employed, a poor guess does not prevent the correct solution from ultimately being obtained.

Ketter and Prawel[2] provide the following example. Consider the equations below

$$\begin{aligned} 4\underline{x_1} + 2x_2 + 0x_3 &= +2 \\ 2x_1 + 10\underline{x_2} + 4x_3 &= +6 \\ 0x_1 + 4x_2 + 5\underline{x_3} &= +5 \end{aligned} \tag{31.11}$$

The reader may choose to assume, as a starting point, $x_1 = x_2 = x_3 = 0$. Solving each equation for the underlined terms (found on the diagonal), one obtains

$$x_1 = 0.50;\ x_2 = 0.60;\ x_3 = 1.00 \tag{31.12}$$

Using these computed values, an updated set of xs can be obtained. Thus,

$$\begin{aligned} x_1 &= \underline{x_1} - \frac{2}{4}x_2 + \frac{0}{4}x_3 = 0.50 - \frac{2}{4}(0.60) + \frac{0}{4}(1.00) - 0.50 = 0.30 \\ x_2 &= \underline{x_2} - \frac{2}{10}x_1 + \frac{4}{10}x_3 = 0.60 - \frac{2}{10}(0.50) + \frac{4}{10}(1.00) = 0.10 \\ x_3 &= \underline{x_3} - \frac{0}{5}x_1 + \frac{4}{5}x_2 = 1.00 - \frac{0}{5}(0.50) + \frac{4}{5}(0.60) = 0.52 \end{aligned} \tag{31.13}$$

The right-hand side of the equations may be viewed as residuals. The procedure is repeated until convergence (the "residuals" approach zero) is obtained. More rapid convergence techniques are available in the literature.[2]

31.4 NONLINEAR ALGEBRAIC EQUATIONS

The subject of the solution to a nonlinear algebraic equation is considered in this section. Although several algorithms are available, the presentation will key on the Newton–Raphson method of evaluating the root(s) of a nonlinear algebraic equation.

The solution to the equation

$$f(x) = 0 \tag{31.14}$$

is obtained by guessing a value for x (x_{old}) that will satisfy the above equation. This value is continuously updated (x_{new}) using the equation

$$x_{\text{new}} = x_{\text{old}} - \frac{f(x_{\text{old}})}{f'(x_{\text{new}})} \tag{31.15}$$

until either little or no change in ($x_{\text{new}} - x_{\text{old}}$) is obtained. One can express this operation graphically (see Fig. 31.4). Noting that

$$f'(x_{\text{old}}) = \frac{df(x)}{dx} \approx \frac{\Delta f(x)}{\Delta x} = \frac{f(x_{\text{old}}) - 0}{x_{\text{old}} - x_{\text{new}}} \tag{31.16}$$

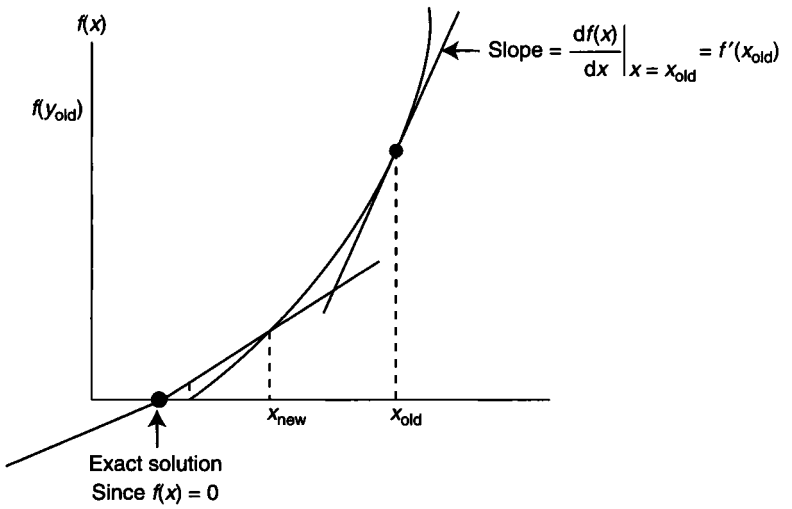

Figure 31.4 Newton–Raphson method.

one may rearrange Equation (31.16) to yield Equation (31.17). The x_{new} then becomes the x_{old} in the next calculation.

This method is also referred to as Newton's Method of Tangents and is a widely used method for improving a first approximation to a root to the aforementioned equation of the form $f(x) = 0$. The above development can be rewritten in subscripted form to (perhaps) better accommodate a computer calculation. Thus

$$f'(x_n) = \frac{f(x_n)}{x_n - x_{n+1}} \tag{31.17}$$

from which

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{31.18}$$

where x_{n+1} is again the improved estimate of x_n, the solution to the equation $f(x) = 0$. The value of the function and the value of the derivative of the function are determined at $x = x_n$, for this procedure, and the new approximation to the root, x_{n+1}, is obtained. The same procedure is repeated, with the new approximation, to obtain a still better approximation of the root. This continues until successive values for the approximate root differ by less than a prescribed small value, ε, which controls the allowable error (or tolerance) in the root. Relative to the previous estimate, ε is given by

$$\varepsilon = \frac{x_{n+1} - x_n}{x_n} \tag{31.19}$$

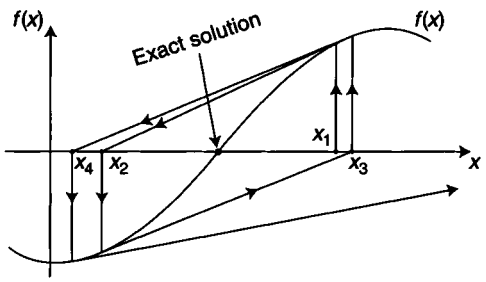

Figure 31.5 Failure of the Newton–Raphson method.

Despite its popularity, the method suffers for two reasons. First, an analytical expression for the derivative, that is, $f'(x_n)$ is required. In addition to the problem of having to compute an analytical derivative value at each iteration, one would expect Newton's method to converge fairly rapidly to a root in the majority of cases. However, as is common with most numerical methods, it may fail occasionally in certain instances. A possible initial oscillation followed by a displacement away from a root is illustrated in Fig. 31.5. Note, however, that the method would have converged (in this case) if the initial guess had been somewhat closer to the exact root. Thus, it can be seen that the initial guess may be critical to the success of the calculation.

Illustrative Example 31.2 The vapor pressure, p', for a new synthetic chemical at a given temperature has been determined to take the form:

$$p' = T^3 - 2T^2 + 2T;\ T = \text{K},\ P' = \text{atm}$$

If $p' = 1$, one may then write

$$f(T) = T^3 - 2T^2 + 2T - 1 = 0$$

where the actual temperature, t (in K), is given by

$$t = 10^3 T$$

Solve the above equation for the actual temperature in K for $p' = 1$. Earlier studies indicate that t is in the 1000 − 1200K range.

Solution Assume an initial temperature t_1. Set $t_1 = 1100$, so that

$$T_1 = (1100)(10^{-3}) = 1.1$$

Obtain the analytical derivative, $f'(T)$

$$f'(T) = 3T^2 - 4T + 2$$

Calculate $f(T_1)$ and $f'(T_1)$

$$f(1.1) = T^3 - 2T^2 + 2T - 1 = (1.1)^3 - 2(1.1)^2 + 2(1.1) - 1 = 0.111$$

$$f'(1.1) = 3T^2 - 4T + 2 = 3(1.1)^2 - 4(1.1) + 2 = 1.23$$

Use the Newton–Raphson method to estimate T_2. Employ Equation (31.15):

$$T_2 = T_1 - \frac{f(T_1)}{f'(T_1)}$$

Substituting,

$$T_2 = 1.1 - \frac{0.111}{1.23} = 1.0098$$

Calculate T_3.

$$f(T_2) = 0.0099$$
$$f'(T_2) = 1.0198$$
$$T_3 = 1.0001$$

Finally, calculate the best estimate (based on two iterations) of t.

$$t = 1000.1\text{K}$$

Other methods that may be employed include:

1. Wegstein's method.
2. False-position.
3. Half-interval.
4. Second-order Newton–Raphson.

Details are available in the literature.[2,6]

Illustrative Example 31.3 The friction factor for smooth tubes can be approximated by

$$f = 0.079\ \text{Re}^{-(1/4)}$$

if $2100 < \text{Re} < 2 \times 10^5$. It can be shown that the average velocity in the system shown in Fig. 31.6, involving the flow of water at 60°F, is given by

$$v = \sqrt{\frac{2180}{213.5\ \text{Re}^{-(1/4)} + 10}}$$

For water at 60°F, Re = 12,168v. Calculate the average velocity, v (ft/s), using the Newton–Raphson method of solution.[7]

Solution Substitute the expression of Reynolds number as a function of velocity into the velocity equation

$$v = \sqrt{\frac{2180}{213.5\ \mathrm{Re}^{-(1/4)} + 10}} = \sqrt{\frac{2180}{213.5(12{,}168v)^{-(1/4)} + 10}}$$
$$= \sqrt{\frac{2180}{[213.5/(12{,}168v)^{(1/4)}] + 10}}$$

Manipulate the above equation into one that is easier to differentiate. Squaring both sides gives

$$v^2 = \frac{2180}{[213.5/(12{,}168v)^{(1/4)}] + 10} = \frac{2180(12{,}168v)^{(1/4)}}{213.5 + 10(12{,}168v)^{(1/4)}}$$

Cross-multiplying leads to

$$213.5v^2 + 10(12{,}168v)^{(1/4)}v^2 - 2180(12{,}168v)^{(1/4)} = 0$$

or

$$f(v) = 213.5v^2 + 105.03v^{2.25} - 22{,}896.08v^{0.25} = 0$$

The analytical derivative of $f(v)$ is

$$f'(v) = 427v + 236.313v^{1.25} - 5724.02v^{-0.75}$$

Make an initial guess of $v = 5$ ft/s and substitute into the above equations:

$$f_1(v) = f_1(5) = -24{,}973.8$$

Applying the initial guess,

$$f_1'(v) = f_1'(5) = 2189.97$$

Calculate the next guess using Equation (31.18).

$$v_2 = v_1 - \frac{f_1(v)}{f_1'(v)} = 5 - \frac{-24{,}973.8}{2189.97} = 16.40\ \mathrm{ft/s}$$

Solve for v_3 etc., until the result converges

$$v_3 = 11.56\,\text{ft/s}$$
$$v_4 = 10.22\,\text{ft/s}$$
$$v_5 = 10.09\,\text{ft/s}$$
$$v_6 = 10.09\,\text{ft/s}$$

The average velocity is therefore 10.09 ft/s.

31.5 NUMERICAL INTEGRATION

Numerous engineering and science problems require the solution of integral equations. In a general sense, the problem is to evaluate the function

$$I = \int_a^b f(x)\,dx \tag{31.20}$$

where I is the value of the integral. There are two key methods employed in their solution: analytical and numerical. If $f(x)$ is a simple function, it may be integrated analytically. For example, if $f(x) = x^2$

$$I = \int_a^b x^2 dx = \frac{1}{3}(b^3 - a^3) \tag{31.21}$$

If, however, $f(x)$ is a function too complex to integrate analytically (e.g., $\ln[\sinh(e^{x^2-2})]$, one may resort to any of the many numerical methods available. Two simple numerical integration methods that are commonly employed are the trapezoidal rule and the Simpon's rule. These are detailed below.

31.5.1 Trapezoidal Rule

In order to use the trapezoidal rule to evaluate the integral with I given by Equation (31.20) as

$$I = \int_a^b f(x)\,dx$$

use the equation

$$I = \frac{h}{2}[y_0 + 2y_1 + 2y_2 + \cdots + 2y_{n-1} + y_n] \tag{31.22}$$

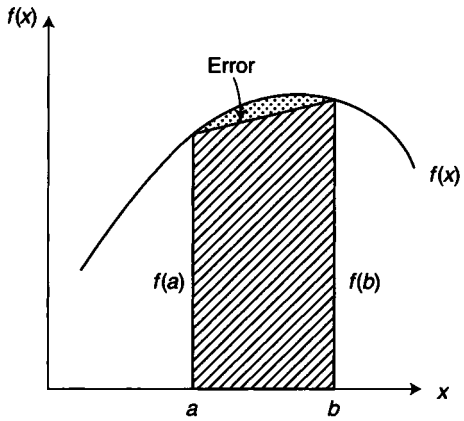

Figure 31.7 Trapezoidal rule error.

where h is the incremental change in x, i.e., Δx, and y_i are the values of $f(x)$ at x_i (i.e., $f(x_i)$). Thus,

$$y_0 = f(x_0) = f(x = a)$$
$$y_n = f(x_n) = f(x = b)$$
$$h = (b - a)/n$$

This method is known as the trapezoidal rule because it approximates the area under the function $f(x)$—which is generally curved—with a 2-point trapezoidal rule calculation. The error associated with this rule is illustrated in Fig. 31.7.

There is an alternative available for improving the accuracy of this calculation. The interval $(a - b)$ can be subdivided into smaller intervals. The trapezoidal rule can be applied repeatedly in turn over each subdivision.

31.5.2 Simpson's Rule

A higher degree interpolating polynomial scheme can be employed for more accurate results. One of the more popular integration approaches is Simpson's rule. For Simpson's 3-point (or one-third) rule, one may use the equation

$$I = \frac{h}{3}[y_a + 4y_{(b+a)/2} + y_b] \tag{31.23}$$

For the general form of Simpson's rule (n is an even integer), the equation is

$$I = \frac{h}{3}[y_0 + 4y_1 + 4y_2 + \cdots + 4y_{n-1} + y_n] \tag{31.24}$$

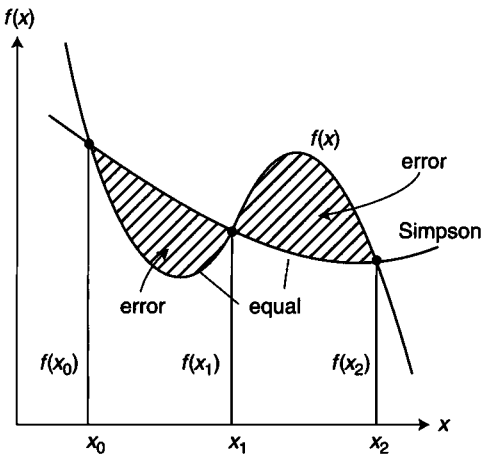

Figure 31.8 Simpson's rule error.

This method also generates an error, although it is usually smaller than that associated with the trapezoidal rule. A diagrammatic representation of the error for a 3-point calculation is provided in Fig. 31.8.

Illustrative Example 31.4 Evaluate the integral below using Simpson's 3-point rule. The term I in this application represents the volume requirement for a tubular flow reactor.

$$I = \int_0^{0.468} \left[\frac{(1-0.4x)^2}{(1-x)(1-0.4x)-1.19x^2}\right] dx$$

Solution Write the 3-point rule. See Equation (31.23).

$$I = \frac{h}{3}[y_a + 4y_{(b+a)/2} + y_b]$$

$$= \frac{h}{3}[f(x=a) + 4f(x=(b+a)/2) + f(x=b)]$$

Evaluate h.

$$h = \frac{0.468}{2} = 0.234$$

Calculate y_a, $y_{(b+a)/2}$ and y_b. For $x = a = 0$,

$$y(0) = \frac{(1-0.4(0))^2}{(1-(0))(1-0.4(0))-1.19(0)^2} = 1$$

Similarly,

$$y(x = 0.234) = 1.548$$
$$y(x = 0.468) = 3.80$$

Finally, calculate the integral I.

$$I = \frac{h}{3}[y_a + 4y_{(b+a)/2} + y_b] = \frac{0.234}{3}[1 + 4(1.548) + 3.80] = 0.857$$

Other numerical integration methods include:

1. Romberg's method.
2. Composite formulas.
3. Gregory's formulas.
4. Taylor's theorem.
5. Method of undetermined coefficients.
6. Richardson's extrapolation.

Details are available in the literature.[3,6,7] Some useful analytical integrals are also provided in mathematical reference handbooks. Note some integrals are indefinite, i.e., the upper and lower limits are not specified.

In closing, the reader should realize that analytical approaches yield closed form and/or exact solutions. Numerical methods provide discrete and/or inexact answers. Thus, the analytical approach should always be attempted first even though numerical methods have become the preferred choice.

REFERENCES

1. B. Carnahan and J. Wilkes, "Digital Computing and Numerical Methods," John Wiley & Sons, Hoboken, NJ, 1973.
2. R. L. Ketter and S. P. Prawler, "Modern Methods of Engineering Computations," McGraw-Hill, New York, NY, 1969.
3. L. Theodore, "Heat Transfer for the Practicing Engineer" (in preparation), John Wiley & Sons, Hoboken, NJ, 2009.
4. L. Theodore, "Mass Transfer for the Practicing Engineer," John Wiley & Sons, Hoboken, NJ, (in preparation).
5. M. Moyle, "Introduction to Computers for Engineers," John Wiley & Sons, Hoboken, NJ, 1967.
6. J. Reynolds, class notes (with permission), Manhattan College, Bronx, NY, 2001.
7. J. Famularo, class notes (with permission), Manhattan College, Bronx, NY, 1981.

NOTE: Additional problems are available for all readers at www.wiley.com. Follow links for this title.