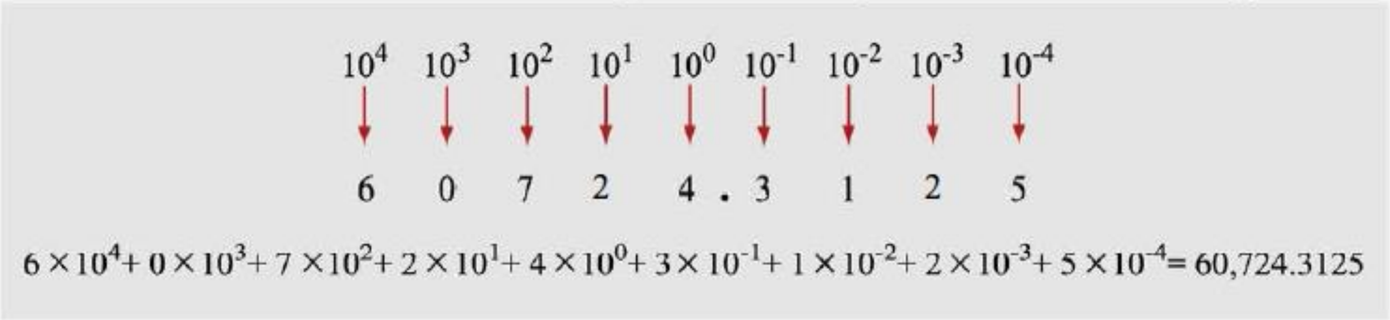
Chapter1: Introduction

1.1 REPRESENTATION OF NUMBERS ON A COMPUTER (Decimal and binary representation)

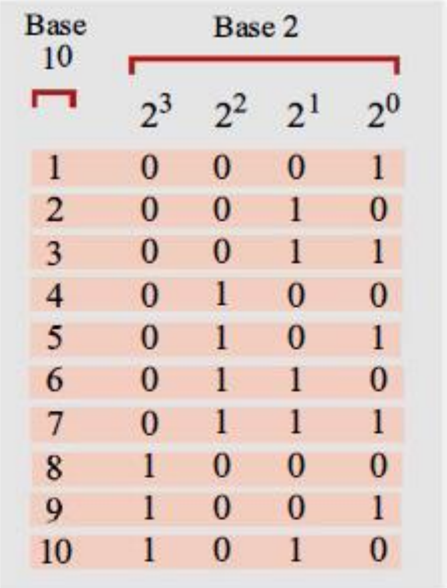
Numbers can be represented in various forms. The familiar decimal system (base 10) uses ten digits 0, 1, ... , 9. A number is written by a sequence of digits that correspond to multiples of powers of 10. As shown in Fig. 1-1, the first digit to the left of the decimal point corresponds to 10^0. The digit next to it on the left corresponds to 10^1 , the next digit to the left to 10^2 , and so on. In the same way, the first digit to the right of the decimal point corresponds to 10^{-1}, the next digit to the right to 10^{-2}, and so on.



$$6\times10^4+0\times10^3+7\times10^2+2\times10^1+4\times10^0+3\times10^{-1}+1\times10^{-2}+2\times10^{-3}+5\times10^{-4}=60{,}724.3125$$

Fig. 1-1: Representation of the number 60,724.3125 in the decimal system (base 10).

In general, however, a number can be represented using other bases. A form that can be easily implemented in computers is the binary (base 2) system. In the binary system, a number is represented by using the two digits 0 and 1. A number is then written as a sequence of zeros and ones that correspond to multiples of powers of 2. The first digit to the left of the decimal point corresponds to 2^0. The digit next to it on the left corresponds to 2^1, the next digit to the left to 2^2, and so on. In the same way, the first digit to the right of the decimal point corresponds to r^1, the next digit to the right to r^2, and so on. The first ten digits 1, 2, 3, . . . , 10 in base 10 and their representation in base 2 are shown in Fig. 1-2. The representation of the number 19.625 in the binary system is shown in Fig. 1-3.



Base 10	Base 2			
	2^3	2^2	2^1	2^0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0

Figure 1-2: Representation of numbers in decimal and binary forms.

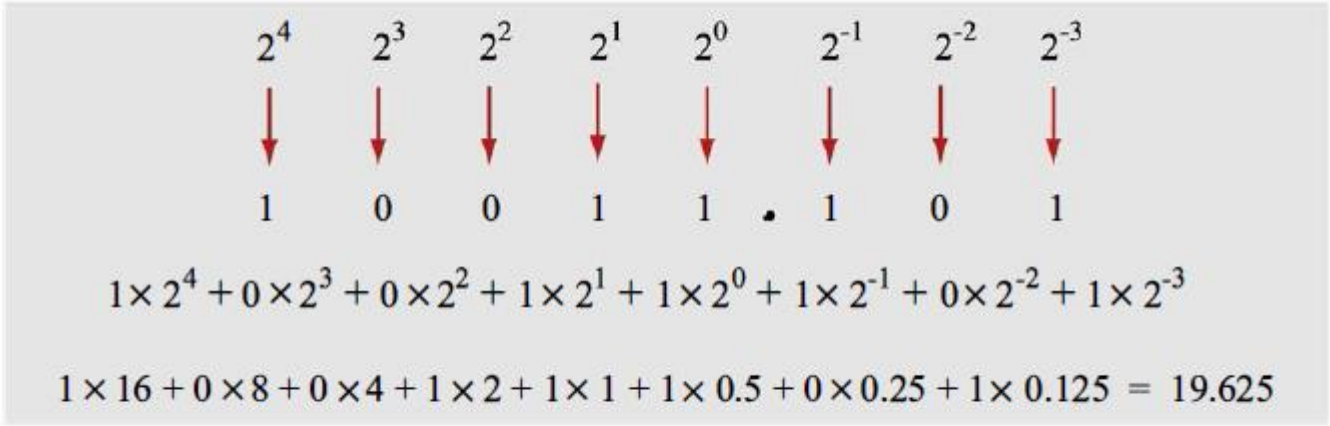

2^4 2^3 2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3}
1 0 0 1 1 . 1 0 1
$1\times 2^4+0\times 2^3+0\times 2^2+1\times 2^1+1\times 2^0+1\times 2^{-1}+0\times 2^{-2}+1\times 2^{-3}$
$1\times 16+0\times 8+0\times 4+1\times 2+1\times 1+1\times 0.5+0\times 0.25+1\times 0.125 = 19.625$

Figure 1-3: Representation of the number 19.625 in the binary system (base 2).

Another example is shown in Fig. 1-4, where the number 60,724.3125 is written in binary form.

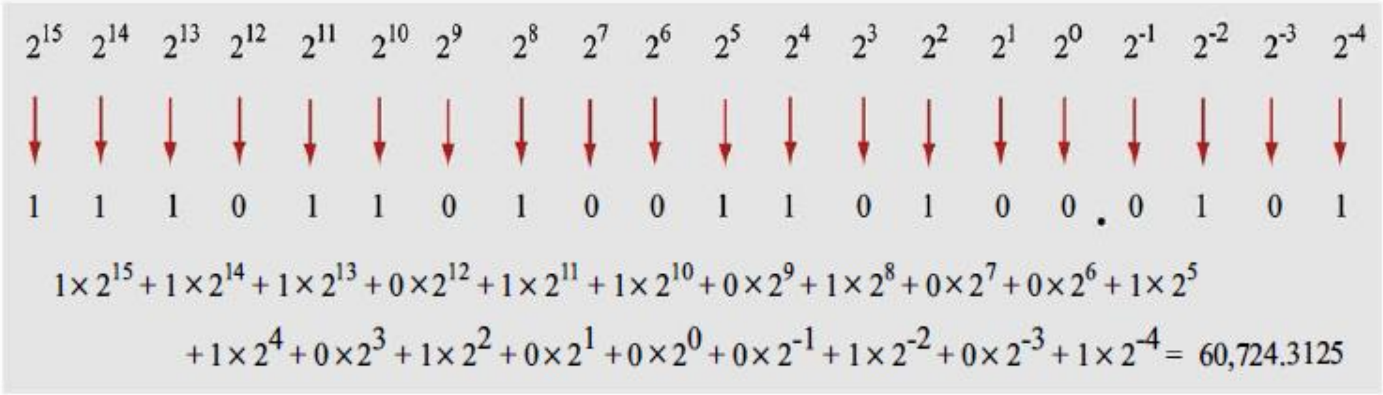

2^{15} 2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^9 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3} 2^{-4}
1 1 1 0 1 1 0 1 0 0 1 1 0 1 0 0 . 0 1 0 1
$1\times 2^{15}+1\times 2^{14}+1\times 2^{13}+0\times 2^{12}+1\times 2^{11}+1\times 2^{10}+0\times 2^9+1\times 2^8+0\times 2^7+0\times 2^6+1\times 2^5$
$+1\times 2^4+0\times 2^3+1\times 2^2+0\times 2^1+0\times 2^0+0\times 2^{-1}+1\times 2^{-2}+0\times 2^{-3}+1\times 2^{-4}= 60{,}724.3125$

Figure 1-4: Representation of the number 60,724.3125 in the binary system (base 2).

Computers store and process numbers in binary (base 2) form:

Each binary digit (one or zero) is called a bit (for binary digit). Binary arithmetic is used by computers because modem transistors can be used as extremely fast switches. Therefore, a network of these may be used to represent strings of numbers with the "1" referring to the switch being in the "on" position and "O" referring to the "off' position. Various operations are then performed on these sequences of ones and zeros.

1.1.2 Floating point representation

To accommodate large and small numbers, real numbers are written in floating point representation. Decimal floating point representation (also called scientific notation) has the form:

$$\text{d.ddddd} \times \text{l}0^P \qquad (1.\ 1)$$

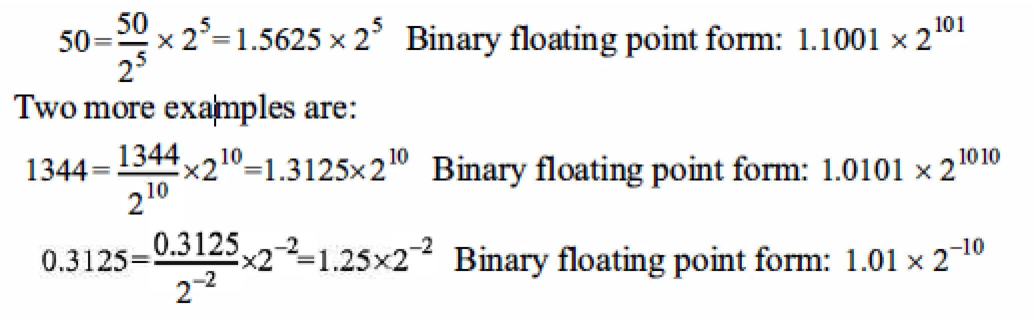
One digit is written to the left of the decimal point, and the rest of the significant digits are written to the right of the decimal point. The number d.dddddd is called the mantissa. Two examples are:

$$6519.23 \text{ written as } 6.51923 \text{ x } 10^3$$
$$0.00000391 \text{ written as } 3.91 \text{ x } 10^{-6}$$

The power of 10, p, represents the number's order of magnitude, provided the preceding number is smaller than 5. Otherwise, the number is said to be of the order of p + 1. Thus, the number 3.91 x 10^{-6} is of the order of 10^{-6}, O(10^{-6}), and the number 6.51923 x 10^3 is of the order of 10^4(written as O(10^4)). Binary floating point representation has the form:

$$1.\ \text{bbbbbb x } 2^{bbb} \quad \text{(b is a decimal digit)} \qquad (1.\ 2)$$

In this form, the mantissa is . bbbbbb , and the power of 2 is called the exponent. Both the mantissa and the exponent are written in a binary form. The form in Eq. (1. 2) is obtained by normalizing the number (when it is written in the decimal form) with respect to the largest power of 2 that is smaller than the number itself. For example, to write the number 50 in binary floating point representation, the number is divided (and multiplied) by 2^5 = 32 (which is the largest power of 2 that is smaller than 50):



$$50=\frac{50}{2^5}\times 2^5=1.5625\times 2^5$$ Binary floating point form: 1.1001×2^{101}

Two more examples are:

$$1344=\frac{1344}{2^{10}}\times 2^{10}=1.3125\times 2^{10}$$ Binary floating point form: 1.0101×2^{1010}

$$0.3125=\frac{0.3125}{2^{-2}}\times 2^{-2}=1.25\times 2^{-2}$$ Binary floating point form: 1.01×2^{-10}

1.2 ERRORS IN NUMERICAL SOLUTIONS

Numerical solutions can be very accurate but in general are not exact. Two kinds of errors are introduced when numerical methods are used for solving a problem. One kind, which was mentioned in the previous section, occurs because of the way that digital computers store numbers and execute numerical operations. These errors are labeled round-off errors. The second kind of errors is introduced by the numerical method that is used for the solution. These errors are labeled truncation errors. Numerical methods use approximations for solving problems. The errors introduced by the approximations are the truncation errors. Together, the two errors constitute the total error of the numerical solution, which is the difference (can be defined in various ways) between the true (exact) solution (which is usually unknown) and the approximate numerical solution. Round-off, truncation, and total errors are discussed in the following three subsections.

1.2.1 Round-Off Errors

Numbers are represented on a computer by a finite number of bits . Consequently, real numbers that have a mantissa longer than the number of bits that are available for representing them have to be shortened. This requirement applies to irrational numbers that have to be represented in a finite form in any system, to finite numbers that are too long, and to finite numbers in decimal form that cannot be represented exactly in binary form. A number can be shortened either by chopping off, or discarding, the extra digits or by rounding. In chopping, the digits in the mantissa beyond the length that can be stored are simply left out. In rounding, the last digit that is stored is rounded.

As a simple illustration, consider the number 2/3. (For simplicity, decimal format is used in the illustration. In the computer, chopping and rounding are done in the binary format.) In decimal form with four significant digits, 2/3 can be written as 0.6666 or as 0.6667. In the former instance, the actual number has been chopped off, whereas in the latter instance, the actual number has been rounded. Either way, such chopping and rounding of real numbers lead to errors in numerical computations, especially when many operations are performed. This type of numerical error (regardless of whether it is due to chopping or rounding) is known as round-off error. Example 1-1 shows the difference between chopping and rounding.

Example 1-1: Round-off errors

Consider the two nearly equal numbers p = 9890.9 and q = 9887. l . Use decimal floating point representation (scientific notation) with three significant digits in the mantissa to calculate the difference between the two numbers, (p - q) . Do the calculation first by using chopping and then by using rounding.

SOLUTION

In decimal floating point representation, the two numbers are:

$$p = 9.8909 \text{ x } 10^3 \text{ and } q = 9.8871 \text{ x } 10^3$$

If only three significant digits are allowed in the mantissa, the numbers have to be shortened. If chopping is used, the numbers become:

$$p = 9.890 \text{ x } 10^3 \text{ and } q = 9.887 \text{ x } 10^3$$

Using these values in the subtraction gives:

$$p- q = 9.890 \text{ x } 10^3 - 9.887 \text{ x } 10^3 = 0.003 \text{ x } 10^3 = 3$$

If rounding is used, the numbers become:

$$p = 9.891 \times 10^3 \text{ and } q = 9.887 \times 10^3 \text{ (q is the same as before)}$$

Using these values in the subtraction gives:

$$p - q = 9.891 \times 10^3 - 9.887 \times 10^3 = 0.004 \times 10^3 = 4$$

The true (exact) difference between the numbers is 3.8. These results show that, in the present problem, rounding gives a value closer to the true answer.

The magnitude of round-off errors depends on the magnitude of the numbers that are involved since, as explained in the previous section, the interval between the numbers that can be represented on a computer depends on their magnitude. Round-off errors are likely to occur when the numbers that are involved in the calculations differ significantly in their magnitude and when two numbers that are nearly identical are subtracted from each other.

For example, consider the quadratic equation:

$$x^2 - 100.000lx + 0.01 = 0 \qquad (1.3)$$

for which the exact solutions are $x_1 = 100$ and $x_2 = 0.0001$. The solutions can be calculated with the quadratic formula:

$$x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a} \text{ and } x_2=\frac{-b-\sqrt{b^2-4ac}}{2a} \qquad (1.4)$$

Using MATLAB (Command Window) to calculate x_1 and x_2 gives:

```
>> format long
>> a = 1; b = -100.0001; c = 0.01;
>> root = sqrt(b^2 - 4*a*c)
root =
        99.999899999999997
>> x1 = (-b + root)/(2*a)
xl =
      100
>> x2 = (-b - root)/(2*a)
x2 =
     1.000000000033197e-004
```

The value that is calculated by MATLAB for x_2 is not exact due to round-off errors. The round-off error occurs in the numerator in the expression for x_2 • Since b is negative, the numerator involves subtraction of two numbers that are nearly equal.

Another example of round-off errors is shown in Example 1-2.

Example 1-2: Round-off errors

Consider the function:

$$f(x) = x(\sqrt{x} - \sqrt{x-1}) \qquad (1.5)$$

(a) Use MATLAB to calculate the value of f(x) for the following three values of x:

$$x = 10,\ x = 1000\ ,\ \text{and } x = 100000\ .$$

(b) Use the decimal format with six significant digits to calculate f(x) for the values of x in part (a). Compare the results with the values in part (a).

SOLUTION

(a)

```
>> format long g
>> x = [10 1000 100000] ;
>> Fx = x.*(sqrt(x) - sqrt(x-1))
Fx =
    1.6227766016838 15.8153431255776 158.114278298171
```

(b) Using decimal format with six significant digits in Eq. (1.5) gives the following values for f(x):

$$f(10) = 10\left(\sqrt{10} - \sqrt{10-1}\right) = 10(3.16228\ - 3)\ =\ 1.62280$$

This value agrees with the value from part (a), when the latter is rounded to six significant digits.

$f(1000) = 1000\left(\sqrt{1000} - \sqrt{1000-1}\right)$ = 1000(31.6228-31.6070) = 15.8

When rounded to six significant digits, the value in part (a) is 15.8153.

$f(100000) = 100000\left(\sqrt{100000} - \sqrt{100000-1}\right)$= 100000(316.228-316.226) = 200

When rounded to six significant digits, the value in part (a) is 158.114.

The results show that the rounding error due to the use of six significant digits increases as x increases and the relative difference between $\sqrt{x}$ and $\sqrt{x-1}$decreases.

1.2.2 Truncation Errors

Truncation errors occur when the numerical methods used for solving a mathematical problem use an approximate mathematical procedure. A simple example is the numerical evaluation of sin(x), which can be done by using Taylor's series expansion :

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots \qquad (1.6)$$

The value of sin $(\frac{\pi}{6})$ can be determined exactly with Eq. (1.6) if an infinite number of terms are used. The value can be approximated by using only a finite number of terms. The difference between the true (exact) value and an approximate value is the truncation error, denoted by E^{TR} . For example, if only the first term is used:

sin $(\frac{\pi}{6})$ = $\frac{\pi}{6}$ = 0.5235988 , E^{TR} = 0.5 - 0.5235988 = -0.0235988

If two terms of the Taylor's series are used:

sin $(\frac{\pi}{6})$ = $\frac{\pi}{6} - \frac{\frac{\pi^3}{6}}{3!}$ = 0.4996742 , E^{TR} = 0.5 - 0.4996742 = 0.0003258

Another example of truncation error that is probably familiar to the reader is the approximate calculation of derivatives. The value of the derivative of a function f(x) at a point x_1 can be approximated by the expression:

$$\frac{df(x)}{dx}|x = x_1 = \frac{f(x_2)-f(x_1)}{x_2-x_1} \qquad (1.7)$$

where x_2 is a point near x_1 • The difference between the value of the true derivative and the value that is calculated with Eq. (1.7) is called a truncation error. The truncation error is dependent on the specific numerical method or algorithm used to solve a problem. Details on truncation errors are discussed as various numerical methods are presented. The truncation error is independent of round-off error; it exists even when the mathematical operations themselves are exact.

1.2.3 Absolute and Relative Errors

If X_E is the exact or true value of a quantity and X_A is its approximate value, then $|X_E - X_A|$ is called the ***absolute error*** E_a. Therefore absolute error:

$$E_a = |X_E - X_A| \qquad (1.8)$$

and ***relative error*** is defined by:

$$E_r = \left|\frac{X_E - X_A}{X_E}\right| \qquad (1.9)$$

provided $X_E \neq\ 0$ or X_E is not too close to zero. The ***percentage relative error*** is:

$$E_p = 100E_r = 100\left|\frac{X_E - X_A}{X_E}\right| \qquad (1.10)$$

Significant digits: The concept of a significant figure, or digit, has been developed to formally define the reliability of a numerical value. The *significant digits* of a number are those that can be used with confidence.

If X_E is the exact or true value and X_A is an approximation to X_E, then X_A is said to approximate X_E to t significant digits if t is the largest non-negative integer for which:

$$\left|\frac{X_E - X_A}{X_E}\right| < 5 \times 10^{-t} \qquad (1.11)$$

Example 1-3:

If $X_E = e$ (base of the natural algorithm = 2.7182818) is approximated by $X_A = 2.71828$, what is the significant number of digits to which X_A approximates X_E?

Solution:

$$\left|\frac{X_E - X_A}{X_E}\right| = \frac{e - 2.71828}{e} \ which\ is < 5 \times 10^{-6}$$

Hence X_A approximates X_E to 6 significant digits.

Example 1-4:

Let the exact or true value = 20/3 and the approximate value = 6.666.

Solution:

The absolute error is 0.000666... = 2/3000.

The relative error is (2/3000)/ (20/3) = 1/10000.

The number of significant digits is 4.

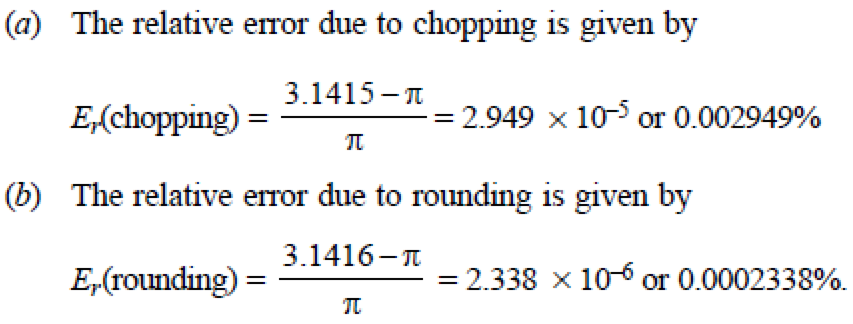
Example 1-5:

Given the number $\pi$is approximated using n = 4 decimal digits.

(a) Determine the relative error due to chopping and express it as a per cent.

(b) Determine the relative error due to rounding and express it as a per cent.

Solution:



(a) The relative error due to chopping is given by

$$E_r(\text{chopping}) = \frac{3.1415 - \pi}{\pi} = 2.949 \times 10^{-5} \text{ or } 0.002949\%$$

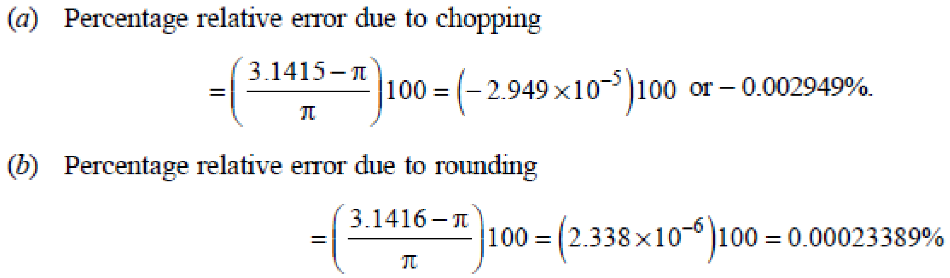
(b) The relative error due to rounding is given by

$$E_r(\text{rounding}) = \frac{3.1416 - \pi}{\pi} = 2.338 \times 10^{-6} \text{ or } 0.0002338\%.$$

Example 1-6:

If the number π= 4 $\tan^{-1}$(1) is approximated using 4 decimal digits, find the percentage relative error due to, (a) chopping (b) rounding.

Solution:



(a) Percentage relative error due to chopping

$$= \left(\frac{3.1415 - \pi}{\pi}\right)100 = \left(-2.949 \times 10^{-5}\right)100 \text{ or } -0.002949\%.$$

(b) Percentage relative error due to rounding

$$= \left(\frac{3.1416 - \pi}{\pi}\right)100 = \left(2.338 \times 10^{-6}\right)100 = 0.00023389\%$$

1.3 PROBLEMS

Problems to be solved by hand

Solve the following problems by hand. When needed, use a calculator or write a MATLAB script file to carry out the calculations.

1. Convert the binary number 1010100 to decimal format.
2. Consider the function $f(x) = \frac{1-\cos x}{\sin x}$.
   a) Use the decimal format with six significant digits (apply rounding at each step) to calculate (using a calculator) f(x) for x = 0.007.
   b) Use MATLAB (format long) to calculate the value of f(x). Consider this to be the true value, and calculate the true relative error, due to rounding, in the value of f(x) that was obtained in part (a).
3. Consider the function $f(x) = \frac{\sqrt{4+x}-2}{x}$.
   a) Use the decimal format with six significant digits (apply rounding at each step) to calculate (using a calculator) f(x) for x = 0.001.
   b) Use MATLAB (format long) to calculate the value of f(x). Consider this to be the true value, and calculate the true relative error, due to rounding, in the value of f(x) that was obtained in part (a).

Chapter2: Solving Nonlinear Equations

2.1 BACKGROUND

Equations need to be solved in all areas of science and engineering. An equation of one variable can be written in the form:

$$f(x) = 0 \qquad (2.1)$$

A solution to the equation (also called a root of the equation) is a numerical value of x that satisfies the equation. Graphically, as shown in Fig. 2-1, the solution is the point where the function *f(x)* crosses or touches the x-axis. An equation might have no solution or can have one or several (possibly many) roots. When the equation is simple, the value of *x* can be determined analytically. This is the case when x can be written explicitly by applying mathematical operations, or when a known formula (such as the formula for solving a quadratic equation) can be used to determine the exact value of x. In many situations, however, it is impossible to determine the root of an equation analytically.

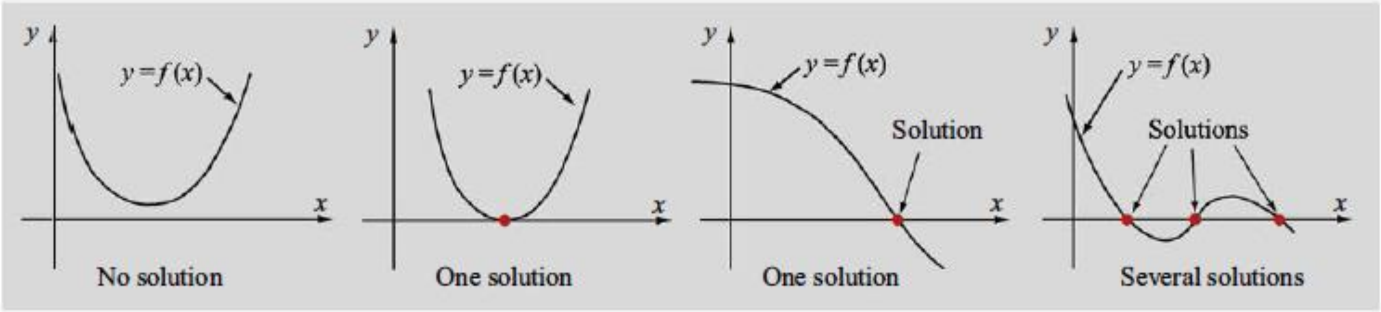


Figure 2-1: Illustration of equations with no, one, or several solutions.

Overview of approaches in solving equations numerically

The process of solving an equation numerically is different from the procedure used to find an analytical solution. An analytical solution is obtained by deriving an expression that has an exact numerical value. A numerical solution is obtained in a process that starts by finding an approximate solution and is followed by a numerical procedure in which a better (more accurate) solution is determined.

An initial numerical solution of an equation *f(x) = 0* can be estimated by plotting *f(x)* versus x and looking for the point where the graph crosses the *x*-axis.

It is also possible to write and execute a computer program that looks for a domain that contains a solution. Such a program looks for a solution by evaluating *f(x)* at different values of x. It starts at one value of x and then changes the value of *x* in small increments. A change in the sign of *f(x)* indicates that there is a root within the last increment. In most cases, when the equation that is solved is related to an application in science or engineering, the range of *x* that includes the solution can be estimated and used in the initial plot of *f(x)*, or for a numerical search of a small domain that contains a solution. When an equation has more than one root, a numerical solution is obtained one root at a time.

The methods used for solving equations numerically can be divided into two groups: bracketing methods and open methods.

In bracketing methods, illustrated in Fig. 2-2, an interval that includes the solution is identified. By definition, the endpoints of the interval are the upper bound and lower bound of the solution. Then, by using a numerical scheme, the size of the interval is successively reduced until the distance between the endpoints is less than the desired accuracy of the solution. In open methods, illustrated in Fig. 2-3, an initial estimate (one point) for the solution is assumed. The value of this initial guess for the solution should be close to the actual solution. Then, by using a numerical scheme, better (more accurate) values for the solution are calculated. Bracketing methods always converge to the solution. Open methods are usually more efficient but sometimes might not yield the solution.

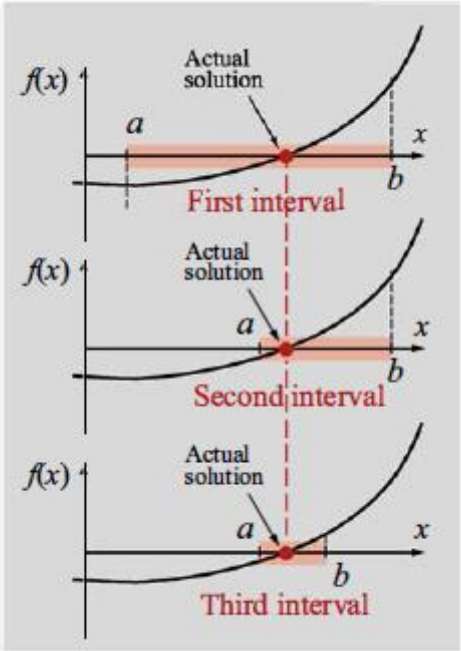


Figure 2-2: Illustration of a bracketing method.

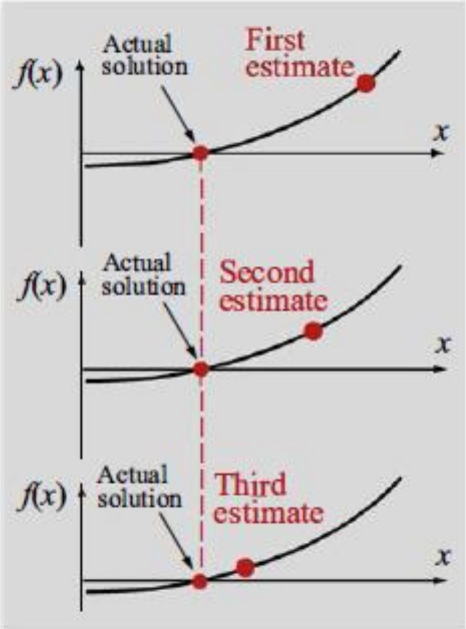


Figure 2-3: Illustration of an open method.

2.2 ESTIMATION OF ERRORS IN NUMERICAL SOLUTIONS

Since numerical solutions are not exact, some criterion has to be applied in order to determine whether an estimated solution is accurate enough. Several measures can be used to estimate the accuracy of an approximate solution. The decision as to which measure to use depends on the application and has to be made by the person solving the equation. Let x_{rs} be the true (exact) solution such that $f(x_{rs}) = 0$, and let x_{Ns} be a numerically approximated solution such that $f(x_{Ns})$ = E (where E is a small number). Four measures that can be considered for estimating the error are:

2.2.1 True error

The true error is the difference between the true solution, X_{rs} and a numerical solution, X_{Ns}:

$$TrueError = X_{rs} - X_{Ns} \qquad (2.2)$$

Unfortunately, however, the true error cannot be calculated because the true solution is generally not known.

2.2.2 Tolerance in *f(x)*:

Instead of considering the error in the solution, it is possible to consider the deviation of $f(x_{Ns})$ from zero (the value of $f(x)$ at x_{rs} is obviously zero). The tolerance in $f(x)$ is defined as the absolute value of the difference between $f(x_{rs})$ and $f(x_{Ns})$:

$$ToleranceInf = |f(x_{rs}) - f(x_{Ns})| = |0 - \varepsilon| = |\varepsilon| \qquad (2.3)$$

The tolerance in $f(x)$ then is the absolute value of the function at x_{Ns}.

2.2.3 Tolerance in the solution:

Tolerance is the maximum amount by which the true solution can deviate from an approximate numerical solution. A tolerance is useful for estimating the error when bracketing methods are used for determining the numerical solution. In this case, if it is known that the solution is within the domain [a, b] , then the numerical solution can be taken as the midpoint between a and b:

$$x_{Ns} = \frac{a+b}{2} \qquad (2.4)$$

plus or minus a tolerance that is equal to half the distance between a and b:

$$\text{Tolerance} = \frac{b-a}{2} \qquad (2.5)$$

2.2.4 Relative error:

If x_{Ns} is an estimated numerical solution, then the True Relative Error is given by:

$$\text{TrueRelativeError} = \left|\frac{x_{rs} - x_{Ns}}{x_{Ns}}\right| \qquad (2.6)$$

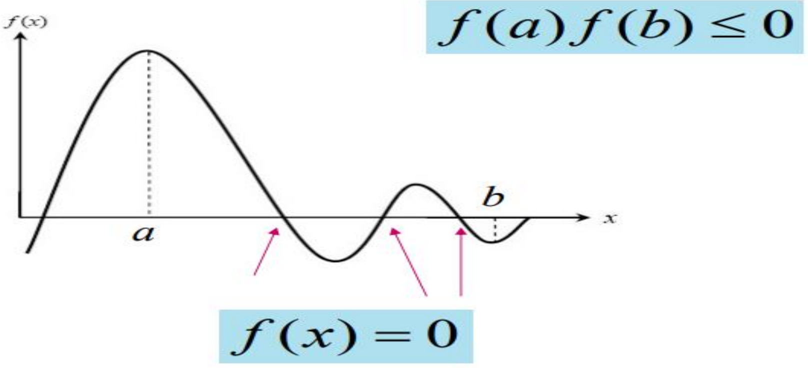
This True Relative Error cannot be calculated since the true solution x_{rs} is not known. Instead, it is possible to calculate an Estimated Relative Error when two numerical estimates for the solution are known. This is the case when numerical solutions are calculated iteratively, wherein each new iteration a more accurate solution is calculated. If $x_{Ns}^{(n)}$ is the estimated numerical solution in the last iteration and $x_{Ns}^{(n-1)}$ is the estimated numerical solution in the preceding iteration, then an Estimated Relative Error can be defined by:

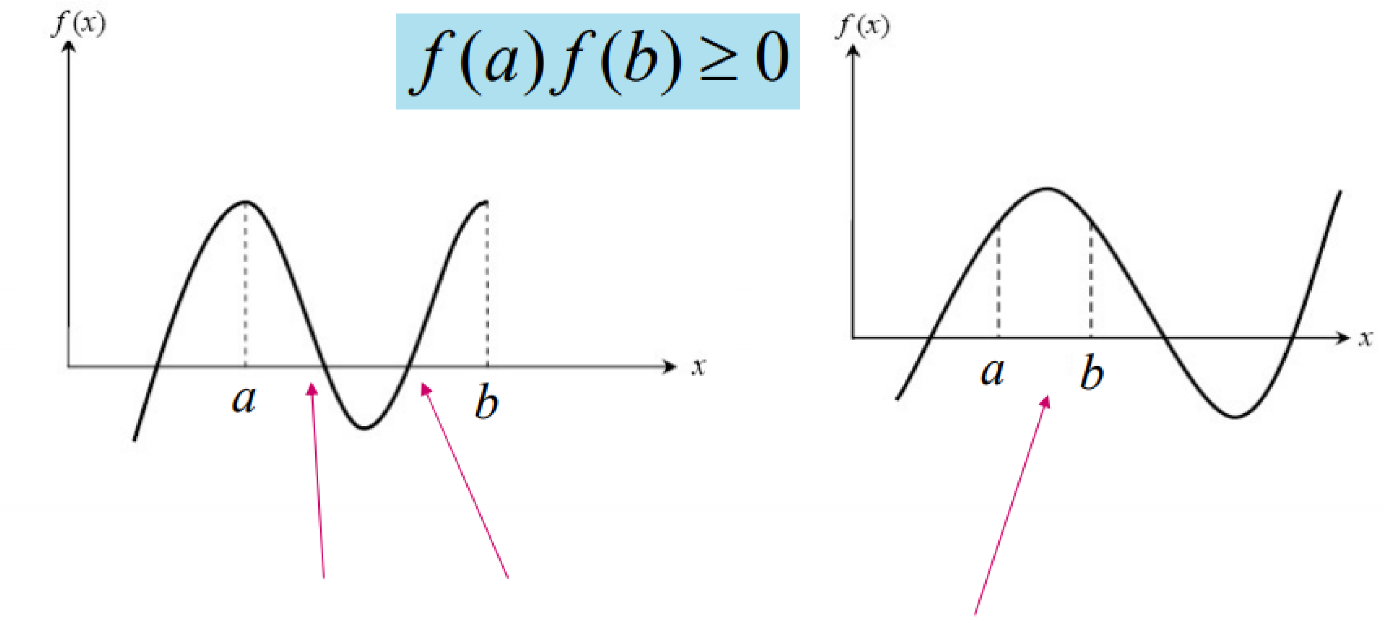
$$\text{Estimated Relative Error} = \left|\frac{x_{Ns}^{(n)} - x_{Ns}^{(n-1)}}{x_{Ns}^{(n-1)}}\right| \qquad (2.7)$$

When the estimated numerical solutions are close to the true solution it is anticipated that the difference $x_{Ns}^{(n)} - x_{Ns}^{(n-1)}$ is small compared to the value of $x_{Ns}^{(n)}$, and the Estimated Relative Error is approximately the same as the True Relative Error.

2.3 Root-finding algorithms

Theorem: If the function $f(x)$ is defined and continuous in the range [a,b] and function changes sign at the ends of the interval that is $f(a)\,f(b) < 0$ then there is at least one single root in the range [a,b].





$$f(a)f(b) \geq 0$$

If the function does not change the sign between two points, there may not be or there may exist roots for this equation between the two points.

Root-finding strategy

- Plot the function (the plot provides an initial guess, and indication of potential problems).
- Isolate single roots in separate intervals (bracketing).
- Select an initial guess.
- Iteratively refine the initial guess with a root-finding algorithm, i.e. generate the sequence :

$$\{x_i\}_{i=0}^{n} : \lim_{n\to\infty}(x_n - \alpha) = 0$$

<u>EXAMPLE 2.1</u>

Find the largest root of $f(x) = x^6 - x - 1 = 0$.

x	-2	-1	0	1	2	3	4
$f(x)$	65	1	-1	-1	61	725	4091

It is obvious that the largest root of this equation is in the interval [1,2].

2.4 BISECTION METHOD

The bisection method is a bracketing method for finding a numerical solution of an equation of the form $f(x) = 0$ when it is known that within a given interval [a, b], $f(x)$ is continuous and the equation has a solution. When this is the case, $f(x)$ will have opposite signs at the endpoints of the interval. As shown in Fig. 2-4, if $f(x)$ is continuous and has a solution between the points x = a and x = b , then either $f(a) > 0$ and $f(b) < 0$ or $f(a) < 0$ and $f(b) > 0$. In other words, if there is a solution between x=a and x = b, then $f(a)f(b) < 0$.

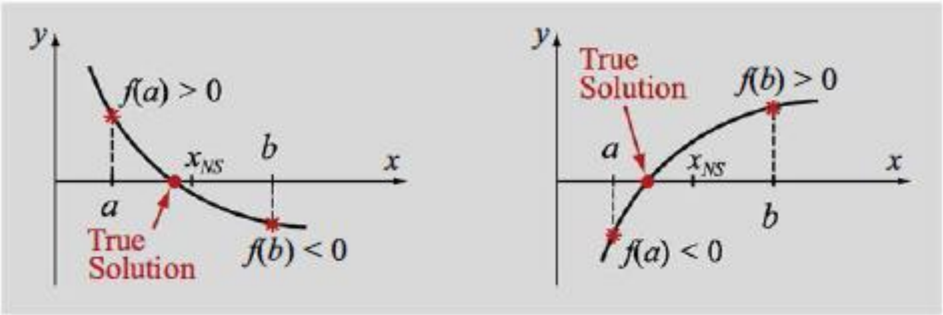


Figure 2-4: Solution of $f(x) = 0$ between x =a and x = b.

Algorithm for the bisection method

1. Choose the first interval by finding points a and b such that a solution exists between them. This means that *f(a)* and *f(b)* have different signs such that *f(a)f(b) < 0*. The points can be determined by examining the plot of f(x) versus x.
2. Calculate the first estimate of the numerical solution x_{Ns1} by:

$$x_{Ns1} = \frac{a+b}{2}$$

3. Determine whether the true solution is between a and x_{NS1} or between x_{Ns1} and b. This is done by checking the sign of the product $f(a) \cdot f(x_{Ns1})$:
If $f(a) \cdot f(x_{Ns1}) < 0$, the true solution is between a and x_{Ns1}.
If $f(a) \cdot f(x_{Ns1}) > 0$, the true solution is between x_{Ns1} and b.
4. Select the subinterval that contains the true solution (a to x_{Ns1}, or x_{Ns1} to b) as the new interval [a, b], and go back to step 2.
Steps 2 through 4 are repeated until a specified tolerance or error bound is attained.

When should the bisection process be stopped?

Ideally, the bisection process should be stopped when the true solution is obtained. This means that the value of x_{Ns} is such that $f(x_{Ns}) = 0$. In reality, as discussed in Section 2.1, this true solution generally cannot be found computationally. In practice, therefore, the process is stopped when the estimated error, according to one of the measures listed in Section 2.2, is smaller than some predetermined value. The choice of termination criteria may depend on the problem that is actually solved.

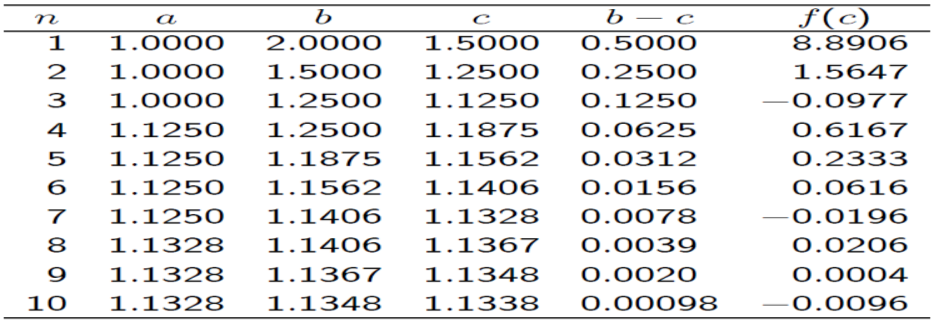
Additional notes on the bisection method

- The method always converges to an answer, provided a root was trapped in the interval [a, b] to begin with.
- The method may fail when the function is tangent to the axis and does not cross the x-axis at f(x) = 0.
- The method converges slowly relative to other methods.

EXAMPLE 2.2

Find the largest root of $f(x) = x^6 - x - 1 = 0$ accurate to within $\in = 0.001$.

Solution With a graph, it is easy to check that $1 < \alpha < 2$. We choose a = 1, b =2; then $f(a) = -1, f(b) = 61$, and the requirement $f(a) f(b) < 0$ is satisfied. The results from Bisect are shown in the table. The entry n indicates the iteration number n.



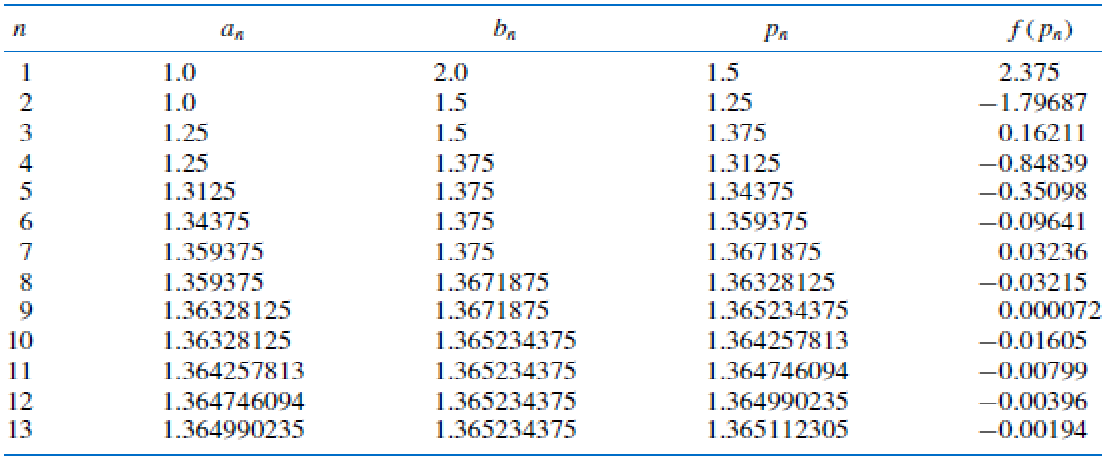
n	a	b	c	$b-c$	$f(c)$
1	1.0000	2.0000	1.5000	0.5000	8.8906
2	1.0000	1.5000	1.2500	0.2500	1.5647
3	1.0000	1.2500	1.1250	0.1250	−0.0977
4	1.1250	1.2500	1.1875	0.0625	0.6167
5	1.1250	1.1875	1.1562	0.0312	0.2333
6	1.1250	1.1562	1.1406	0.0156	0.0616
7	1.1250	1.1406	1.1328	0.0078	−0.0196
8	1.1328	1.1406	1.1367	0.0039	0.0206
9	1.1328	1.1367	1.1348	0.0020	0.0004
10	1.1328	1.1348	1.1338	0.00098	−0.0096

Example 2.3 Show that $f(x) = x^3 + 4x^2 - 10 = 0$ has a root in [1, 2], and use the Bisection method to determine an approximation to the root that is accurate to at least within10^{-4}.

Solution Because $f(1) = -5$ and $f(2) = 14$, the Intermediate Value Theorem ensures that this continuous function has a root in [1, 2].

For the first iteration of the Bisection method we use the fact that at the midpoint of [1,2] we have $f(1.5) = 2.375 > 0$. This indicates that we should select the interval [1,1.5] for our second iteration. Then we find that $f(1.25) = -1.796875$ so our new interval becomes [1.25, 1.5], whose midpoint is 1.375. Continuing in this

manner gives the values in the following table. After 13 iterations, $p_{13} = 1.365112305$ approximates the root p with an error:
$|p - p_{13}| < |b_{14} - a_{14}| = |1.365234375 - 1.365112305| = 0.000122070.$
Since $|a_{14}| < |p|$, we have $|p - p_{13}|/|p| < |b_{14} - a_{14}|/|a_{14}| \leq 9.0 \times 10^{-5}$,



n	a_n	b_n	p_n	$f(p_n)$
1	1.0	2.0	1.5	2.375
2	1.0	1.5	1.25	−1.79687
3	1.25	1.5	1.375	0.16211
4	1.25	1.375	1.3125	−0.84839
5	1.3125	1.375	1.34375	−0.35098
6	1.34375	1.375	1.359375	−0.09641
7	1.359375	1.375	1.3671875	0.03236
8	1.359375	1.3671875	1.36328125	−0.03215
9	1.36328125	1.3671875	1.365234375	0.000072
10	1.36328125	1.365234375	1.364257813	−0.01605
11	1.364257813	1.365234375	1.364746094	−0.00799
12	1.364746094	1.365234375	1.364990235	−0.00396
13	1.364990235	1.365234375	1.365112305	−0.00194

so the approximation is correct to at least within 10^{-4}. The correct value of p to nine decimal places is $p = 1.365230013$. Note that p_9 is closer to p than is the final approximation p_{13}. You might suspect this is true because $|f(p_9)| < |f(p_{13})|$, but we cannot be sure of this unless the true answer is known.

<u>Example 2.4</u> Use the Bisection method to find a root of the equation $x^3 - 4x - 8.95 = 0$ accurate to three decimal places using the Bisection Method.

<u>Solution</u>

Here, $f(x) = x^3 - 4x - 8.95 = 0$
$f(2) = 2^3 - 4(2) - 8.95 = -8.95 < 0$
$f(3) = 3^3 - 4(3) - 8.95 = 6.05 > 0$
Hence, a root lies between 2 and 3.
Hence, we have $a = 2$ and $b = 3$. The results of the algorithm for Bisection method are shown in Table.

n	a	b	x_{S_1}	$b\text{-}x_{S_1}$	$f(x_{S_1})$
0	2	3	2.5	0.5	-3.324999999999999
1	2.5	3	2.75	0.25	0.846875000000001
2	2.5	2.75	2.625	0.125	-1.362109374999999
3	2.625	2.75	2.6875	0.0625	-0.289111328124999
4			2.71875	0.03125	0.270916748046876
5			2.703125	0.015625	-0.011077117919921
6			2.7109375	0.007812	0.129423427581788
7			2.7070312	0.003906	0.059049236774445
8			2.7050781	0.001953	0.023955102264882
9			2.7041016	0.000976	0.006431255675853
10			2.7036133	0.000488	-0.002324864896945
11			2.7038574	0.000244	0.002052711902071
12			2.7037354	0.000122	-0.000136197363826
13			2.7037964	0.000061	0.000958227051843

Hence the root is 2.703 accurate to three decimal places.

2.5 FALS POSITION METHOD

The false position method (also called regula falsi and linear interpolation methods) is a bracketing method for finding a numerical solution of an equation of the form $f(x) = 0$ when it is known that, within a given interval [a, b], $f(x)$ is continuous and the equation has a solution. As illustrated in Fig. 2-5, the solution starts by finding an initial interval $[a_1, b_1]$ that brackets the solution. The values of the function at the endpoints are $f(a_1)$ and $f(b_1)$. The endpoints are then connected by a straight line, and the first estimate of the numerical solution, x_{Ns1}, is the point where the straight line crosses the x-axis. This is in contrast to the bisection method, where the midpoint of the interval was taken as the solution. For the second iteration a new interval, $[a_2, b_2]$ is defined. The new interval is a subsection of the first interval that contains the solution. It is either $[a_1, x_{Ns1}]$ (a_1 is assigned to a_2, and x_{Ns1} to b_2) or $[x_{Ns1}, b_1]$ (x_{Ns1} is assigned to a_2, and b_1 to b_2). The endpoints of the second interval are next connected with a straight line, and the point where this new line crosses the x-axis is the second estimate of the solution, x_{Ns2}. For the third iteration, a new subinterval $[a_3, b_3]$ is selected, and the iterations continue in the same way until the numerical solution is deemed accurate enough.

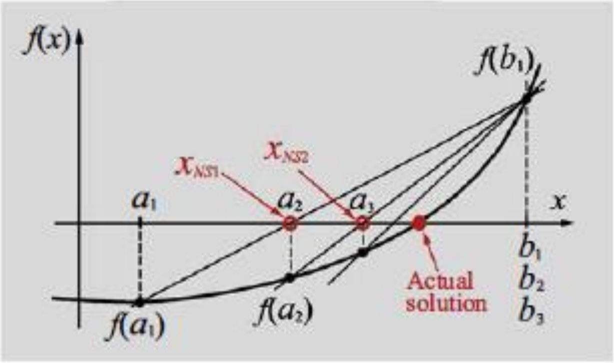


Figure 2-5: False position method

For a given interval [a, b], the equation of a straight line that connects point $(b, f(b))$ to point $(a, f(a))$ is given by:

$$y = \frac{f(b)-f(a)}{b-a}(x-b) + f(a) \qquad (2.8)$$

The point x_{NS} where the line intersects the x-axis is determined by substituting $y=0$ in Eq. (2.8), and solving the equation for x :

$$x_{Ns} = \frac{af(b)-bf(a)}{f(b)-f(a)} \qquad (2.9)$$

The procedure (or algorithm) for finding a solution with the regula falsi method is almost the same as that for the bisection method**.**

Algorithm for the regula falsi method

1. Choose the first interval by finding points a and b such that a solution exists between them. This means that $f(a)$ and $f(b)$ have different signs such that $f(a)f(b) < 0$. The points can be determined by looking at a plot of $f(x)$ versus x.
2. Calculate the first estimate of the numerical solution x_{Ns1} by using Eq. (2.9).
3. Determine whether the actual solution is between a and x_{Ns1} or between x_{Ns1} and b. This is done by checking the sign of the product $f(a) \cdot f(x_{Ns1})$:
If $f(a) \cdot f(x_{Ns1}) < 0$, the solution is between a and $x_{Ns1}\cdot$
If $f(a) \cdot f(x_{Ns1}) > 0$, the solution is between x_{Ns1} and b.
4. Select the subinterval that contains the solution (a to x_{Ns1}, or x_{Ns1} to b) as the new interval [a, b], and go back to step 2.
Steps 2 through 4 are repeated until a specified tolerance or error bound is attained.

When should the iterations be stopped?

The iterations are stopped when the estimated error, according to one of the measures listed in Section 2 .2, is smaller than some predetermined value.

Additional notes on the regula falsi method

• The method always converges to an answer, provided a root is initially trapped in the interval [a, b].

• Frequently, as in the case shown in Fig. 2-5, the function in the interval [a, b] is either concave up or concave down. In this case, one of the endpoints of the interval stays the same in all the iterations, while the other endpoint advances toward the root. In other words, the numerical solution advances toward the root only from one side. The convergence toward the solution could be faster if the other endpoint would also "move" toward the root. Several modifications have been introduced to the regula falsi method that makes the subinterval in successive iterations approach the root from both sides.

Example 2.5

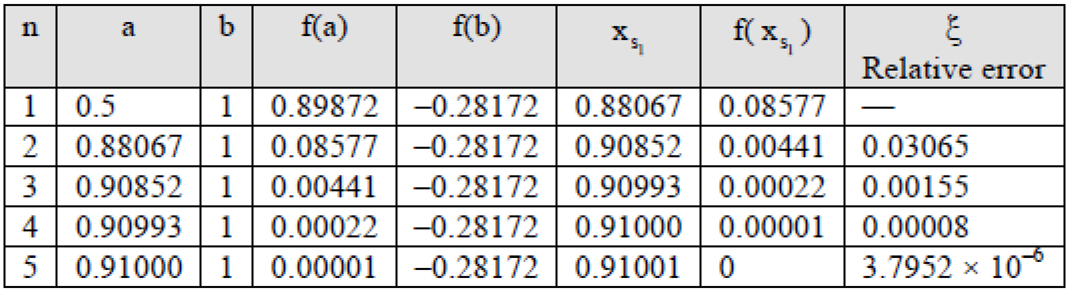
Using the False Position method, find a root of the function $f(x) = e^x - 3x^2$ to an accuracy of 5 digits. The root is known to lie between 0.5 and 1.0.

Solution

We apply the method of False Position with $a = 0.5$ and $b = 1.0$ and equation (2.2) which is:

$$x_s = \frac{a\,f(b) - b\,f(a)}{f(b) - f(a)}$$

The calculations based on the method of False Position are shown in the following table:



n	a	b	f(a)	f(b)	x_{s_1}	$f(x_{s_1})$	ξ Relative error
1	0.5	1	0.89872	–0.28172	0.88067	0.08577	—
2	0.88067	1	0.08577	–0.28172	0.90852	0.00441	0.03065
3	0.90852	1	0.00441	–0.28172	0.90993	0.00022	0.00155
4	0.90993	1	0.00022	–0.28172	0.91000	0.00001	0.00008
5	0.91000	1	0.00001	–0.28172	0.91001	0	3.7952×10^{-6}

The relative error after the fifth step is

$$\left(\frac{0.91001 - 0.91}{0.91001}\right) = 3.7952 \times 10^{-6}.$$

The root is 0.91 accurate to five digits.

Example 2.6

Using the method of False Position, find a real root of the equation $x^4 - 11x + 8 = 0$ accurate to four decimal places.

Solution

Here $f(x) = x^4 - 11x + 8 = 0$
$f(1) = 1^4 - 11(1) + 8 = -2 < 0$
$f(2) = 2^4 - 11(2) + 8 = 2 > 0$
Therefore, a root of $f(x) = 0$ lies between 1 and 2.We apply the method of False Position with a = 1and b =2.
The calculations based on the method of False Position are summarized in the following Table :

n	a	b	f(a)	f(b)	x_{s_1}	$f(x_{s_1})$	ξ
1	1	2	−2	2	1.5	−3.4375	—
2	1.5	2	−3.4375	2	1.81609	−1.9895	0.17405
3	1.81609	2	−1.09895	2	1.88131	−0.16758	3.4666×10^{-2}
4	1.88131	2	−0.16758	2	1.89049	−0.02232	4.85383×10^{-3}
5	1.89049	2	−0.02232	2	1.89169	−0.00292	6.3902×10^{-4}
6	1.89169	2	−0.00292	2	1.89185	−0.00038	8.34227×10^{-5}
7	1.89185	2	−0.00038	2	1.89187	−0.00005	1.08786×10^{-5}

The relative error after the seventh step is

$$\xi = \frac{1.89187 - 1.89185}{1.89187} = 1.08786 \times 10^{-5}$$

Hence, the root is 1.8918 accurate to four decimal places.

2.6 NEWTON'S METHOD

Newton's method (also called the Newton-Raphson method) is a scheme for finding a numerical solution of an equation of the form *f(x) = 0* where *f(x)* is continuous and differentiable and the equation is known to have a solution near a given point. The method is illustrated in Fig. 2.6.

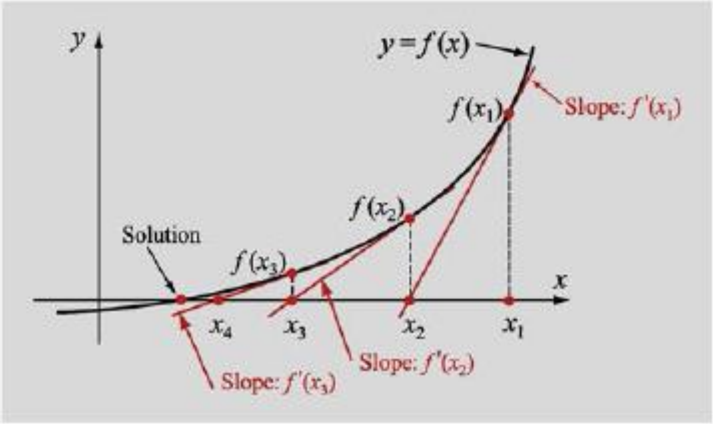


Figure 2-6: Newton's method.

The solution process starts by choosing point x_1 as the first estimate of the solution. The second estimate x_2 is obtained by taking the tangent line to *f(x)* at the point $(x_1, f(x_1))$ and finding the intersection point of the tangent line with the *x*-axis. The next estimate x_3 is the intersection of the tangent line to *f(x)* at the point $(x_2, f(x_2))$ with the *x*-axis, and so on. Mathematically, for the first iteration, the slope, $f'(x_1)$, of the tangent at point $(x_1, f(x_1))$ is given by:

$$f'(x_1) = \frac{f(x_1)-0}{x_1-x_2} \qquad (2.10)$$

Solving Eq. (2.10) for x_2 gives:

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \qquad (2.11)$$

Equation (2 .11) can be generalized for determining the "next" solution x_{i+1} from the present solution x_i:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \qquad (2.12)$$

Equation (2.12) is the general iteration formula for Newton's method. It is called an iteration formula because the solution is found by repeated application of Eq. (2.12) for each successive value of *i*.

Algorithm for Newton's method

1. Choose a point x_i as an initial guess of the solution.
2. For *i* = 1, 2, . . . , until the error is smaller than a specified value, calculate x_{i+1} by using Eq. (2.12).

When are the iterations stopped?

Ideally, the iterations should be stopped when an exact solution is obtained. This means that the value of x is such that $f(x) = 0$. Generally, as discussed in Section 2.1, this exact solution cannot be found computationally. In practice, therefore, the iterations are stopped when an estimated error is smaller than some predetermined value. Tolerance in the solution, as in the bisection method, cannot be calculated since bounds are not known. Two error estimates that are typically used with Newton's method are:

Estimated relative error: The iterations are stopped when the estimated relative error is smaller than a specified value ε:

$$\left|\frac{x_{i+1} - x_i}{x_i}\right| \leq \boldsymbol{\varepsilon}$$

Tolerance in f(x): The iterations are stopped when the absolute value of f(x;) is smaller than some number δ:

$$|f(x_i)| \leq \delta$$

Notes on Newton's method

• The method, when successful, works well and converges fast. When it does not converge, it is usually because the starting point is not close enough to the solution. Convergence problems typically occur when the value of *f '(x)* is close to zero in the vicinity of the solution (where *f(x) = 0*). It is possible to show that Newton's method converges if the function *f(x)* and its first and second derivatives *f '(x)* and *f ''(x)* are all continuous, if *f '(x)* is not zero at the solution, and if the starting value x_1 is near the actual solution. Illustrations of two cases where Newton's method does not converge (i.e., diverges) are shown in Fig. 2-7.

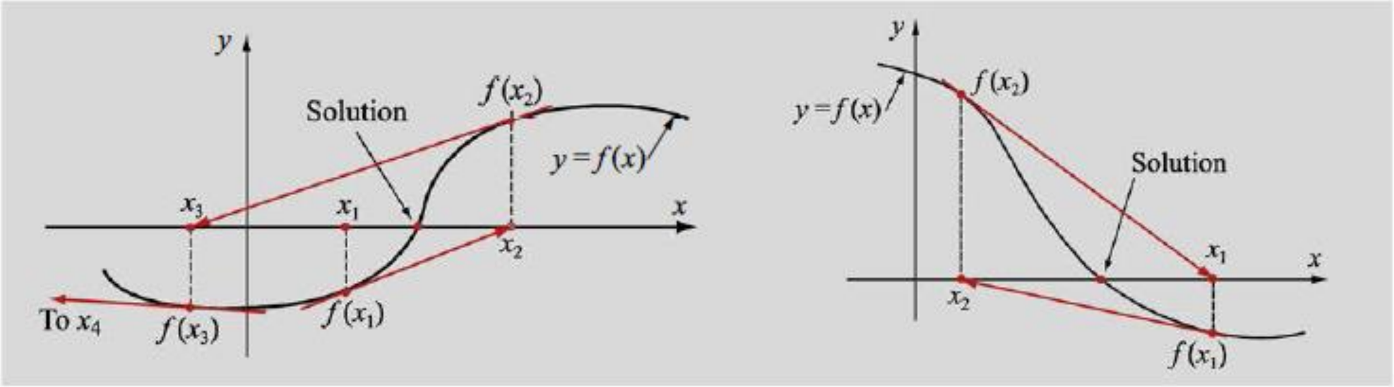


Figure 2-7: Cases where Newton's method diverges.

• A function *f '(x)*, which is the derivative of the function *f(x)*, has to be substituted in the iteration formula, Eq. (2.12). In many cases, it is simple to write the derivative, but sometimes it can be difficult to determine. When an expression for the derivative is not available, it might be possible to determine the slope numerically or to find a solution by using the secant method (Section 2.7), which is somewhat similar to Newton's method but does not require an expression for the derivative.

Example 2.7

Find the solution of the equation *8-4.5(x- sin(x)) = 0* by using Newton's method in the following two ways:

(a) Using a nonprogrammable calculator, calculate the first two iterations on paper using six significant figures.

(b) Use MATLAB with 0.0001 for the maximum relative error and 10 for the maximum number of iterations.

In both parts, use $x = 2$ as the initial guess of the solution.

Solution:

In the present problem, *f(x) = 8 - 4.5(x-sinx)* and *f '(x) = -4.5(1 -cosx)* .

(a) To start the iterations, *f(x)* and *f '(x)* are substituted in Eq. (2.12):

$$x_{i+1} = x_i - \frac{8-4.5(x_i - \sin x_i)}{-4.5(1-\cos x_i)} \qquad (2.13)$$

In the first iteration, $i = 1$ and $x_1 = 2$, and Eq. (2.13) gives:

$$x_2 = 2 - \frac{8-4.5(2-\sin 2)}{-4.5(1-\cos 2)} = 2.485172$$

for the second iteration, $i = 2$ and $x_2 = 2.485172$, and Eq. (2.13) gives:

$$x_3 = 2.485172 - \frac{8-4.5(2.485172-\sin 2.485172)}{-4.5(1-\cos 2.485172)} = 2.430987$$

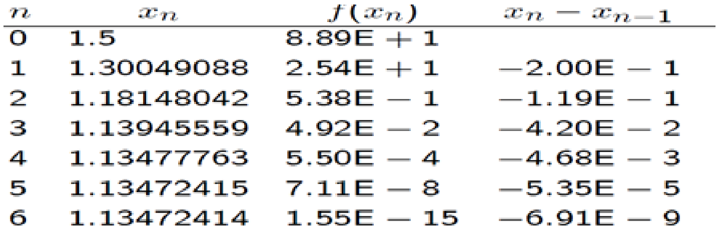
(b) (Exc)

Example2.8 Consider $f(x) \equiv x^6 - x - 1 = 0$ for its positive root α. An initial guess x_0 can be generated from a graph of $y = f(x)$. The iteration is given by:

$$x_{n+1} = x_n - \frac{x_n^6 - x_n - 1}{6x_n^5 - 1} \quad , n \geq 0$$

We use an initial guess of $x_0 = 1.5$.

The column "$x_n - x_{n-1}$" is an estimate of the error $\alpha - x_{n-1}$; justification is given later.



n	x_n	$f(x_n)$	$x_n - x_{n-1}$
0	1.5	8.89E + 1	
1	1.30049088	2.54E + 1	−2.00E − 1
2	1.18148042	5.38E − 1	−1.19E − 1
3	1.13945559	4.92E − 2	−4.20E − 2
4	1.13477763	5.50E − 4	−4.68E − 3
5	1.13472415	7.11E − 8	−5.35E − 5
6	1.13472414	1.55E − 15	−6.91E − 9

As seen from the output, the convergence is very rapid. The iterate x_6 is accurate to the machine precision of around 16 decimal digits. This is the typical behaviour seen with Newton's method for most problems, but not all.

Example 2.9:

Use the Newton-Raphson method to find the real root near 2 of the equation $x^4 - 11x + 8 = 0$ accurate to five decimal places.

Solution:

Here $f(x) = x^4 - 11x + 8$

$f'(x) = 4x^3 - 11$

$x_0 = 2$

and $f(x_0) = f(2) = 2^4 - 11(2) + 8 = 2$

$f'(x_0) = f'(2) = 4(2)^3 - 11 = 21$

Therefore,

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 2 - \frac{2}{21} = 1.90476$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 1.90476 - \frac{(1.90476)^4 - 11(1.90476) + 8}{4(1.90476)^3 - 11} = 1.89209$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 1.89209 - \frac{(1.89209)^4 - 11(1.89209) + 8}{4(1.89209)^3 - 11} = 1.89188$$

$$x_4 = x_3 - \frac{f(x_3)}{f'(x_3)} = 1.89188 - \frac{(1.89188)^4 - 11(1.89188) + 8}{4(1.89188)^3 - 11} = 1.89188$$

Hence the root of the equation is 1.89188.

Example 2.10

Using Newton-Raphson method, find a root of the function $f\,(x) = e^x - 3x^2$ to an accuracy of 5 digits. The root is known to lie between 0.5 and 1.0. Take the starting value of x as $x_0 = 1.0$.

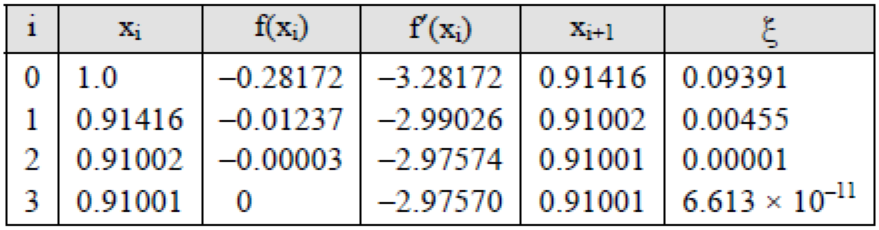
Solution:

Start at $x_0 = 1.0$ and prepare a table as shown in Table 2.8, where $f\,(x) = e^x - 3x^2$ and $f'(x) = e^x - 6x$. The relative error

$$\zeta = \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right|$$

The Newton-Raphson iteration method is given by

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$



i	x_i	$f(x_i)$	$f'(x_i)$	x_{i+1}	ξ
0	1.0	–0.28172	–3.28172	0.91416	0.09391
1	0.91416	–0.01237	–2.99026	0.91002	0.00455
2	0.91002	–0.00003	–2.97574	0.91001	0.00001
3	0.91001	0	–2.97570	0.91001	6.613×10^{-11}

Example 2.11:

Evaluate $\sqrt{29}$ to five decimal places by Newton-Raphson iterative method.

Solution:

Let $x = \sqrt{29}$ then $x^2 - 29 = 0$.

We consider $f\,(x) = x^2 - 29 = 0$ and $f'(x) = 2x$

The Newton-Raphson iteration formula gives

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^2 - 29}{2x_i} = \frac{1}{2}\left(x_i + \frac{29}{x_i}\right) \qquad \text{(E.1)}$$

Now $f\,(5) = 25 - 29 = -4 < 0$ and $f\,(6) = 36 - 29 = 7 > 0$.

Hence, a root of $f\,(x) = 0$ lies between 5 and 6.

Taking $x_0 = 5.3$, Equation (E.1) gives

$$x_1 = \frac{1}{2}\left(5.3 + \frac{29}{5.3}\right) = 5.38585$$

$$x_2 = \frac{1}{2}\left(5.38585 + \frac{29}{5.38585}\right) = 5.38516$$

$$x_3 = \frac{1}{2}\left(5.38516 + \frac{29}{5.38516}\right) = 5.38516$$

Since $x_2 = x_3$ up to five decimal places, $\sqrt{29} = 5.38516$.

2.7 SECANT METHOD

The secant method is a scheme for finding a numerical solution of an equation of the form $f(x) = 0$. The method uses two points in the neighbourhood of the solution to determine a new estimate for the solution (Fig. 2-8). The two points (marked as x_1 and x_2 in the figure) are used to define a straight line (secant line), and the point where the line intersects the x-axis (marked as x_3 in the figure) is the new estimate for the solution. As shown, the two points can be on one side of the solution

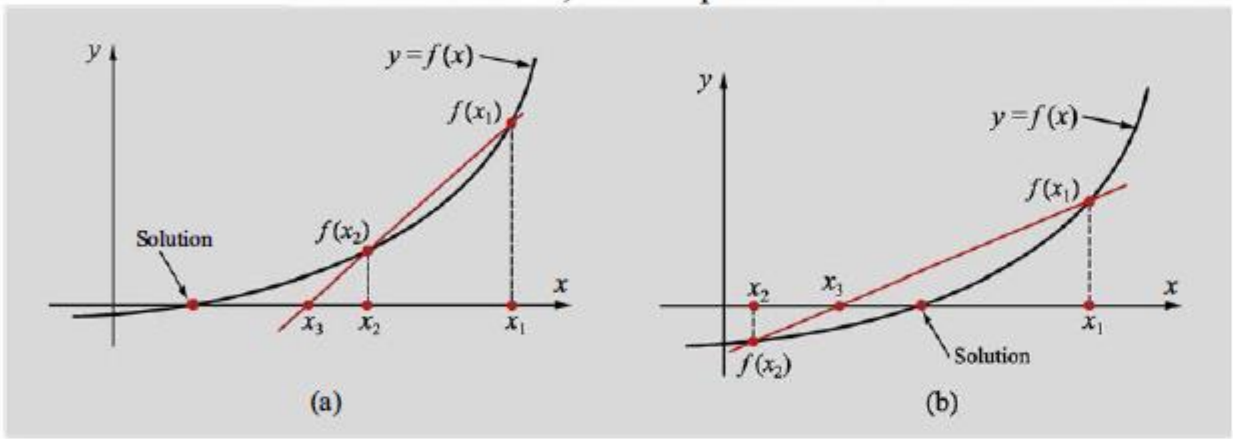


Figure 2-8: The secant method.

The slope of the secant line is given by:

$$\frac{f(x_1)-f(x_2)}{x_1-x_2} = \frac{f(x_2)-0}{x_2-x_3} \qquad (2.14)$$

which can be solved for x_3 :

$$x_3 = x_2 - \frac{f(x_2)(x_1-x_2)}{f(x_1)-f(x_2)} \qquad (2.15)$$

Once point x_3 is determined, it is used together with point x_2 to calculate the next estimate of the solution, x_4. Equation (2.15) can be generalized to an iteration formula in which a new estimate of the solution x_{i+1} is determined from the previous two solutions x_i and x_{i-1}.

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1}-x_i)}{f(x_{i-1})-f(x_i)} \qquad (2.16)$$

Figure 2-9 illustrates the iteration process with the secant method.

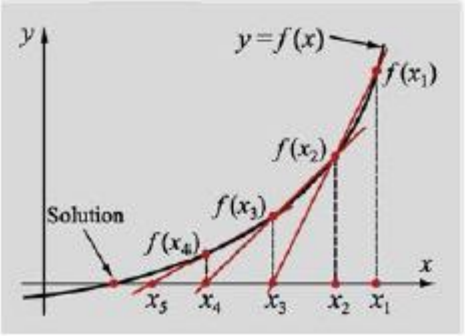


Figure 2-9: Secant method.

<u>Example 2.12</u>

Find a root of the equation $x^3 - 8x - 5 = 0$ using the secant method.

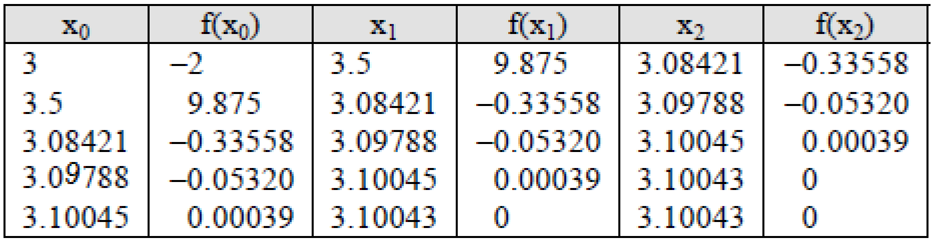
<u>Solution:</u>

$f(x) = x^3 - 8x - 5 = 0$
$f(3) = 3^3 - 8(3) - 5 = -2$
$f(4) = 4^3 - 8(4) - 5 = 27$

Therefore one root lies between 3 and 4. Let the initial approximations be $x_0 = 3$, and $x_1 = 3.5$. Then, x_2 is given by:

$$x_2 = \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)}$$

The calculations are summarized in the above Table



x_0	$f(x_0)$	x_1	$f(x_1)$	x_2	$f(x_2)$
3	–2	3.5	9.875	3.08421	–0.33558
3.5	9.875	3.08421	–0.33558	3.09788	–0.05320
3.08421	–0.33558	3.09788	–0.05320	3.10045	0.00039
3.09788	–0.05320	3.10045	0.00039	3.10043	0
3.10045	0.00039	3.10043	0	3.10043	0

Hence, a root is 3.1004 correct up to five significant figures.

<u>Example 2.13</u>

Determine a root of the equation *sin(x)+ 3 cos(x) – 2 = 0* using the secant method. The initial approximations x_0 and x_1 are 0 and 1.5.

<u>Solution:</u>

The formula for x_2 is given by:

$$x_2 = \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)}$$

The calculations are summarized in the above Table.

x_0	$f(x_0)$	x_1	$f(x_1)$	x_2	$f(x_2)$
0	1	1.5	-0.79029	0.83785149	0.75039082
1.5	-0.79029	0.83785149	0.75039082	1.160351166	0.113995951
0.83785149	0.75039082	1.160351166	0.113995951	1.2181197917	-0.025315908
1.160351166	0.113995951	1.2181197917	-0.025315908	1.2076220119	0.000503735
1.2181197917	-0.025315908	1.2076220119	0.000503735	1.2078268211	0.000002099
1.2076220119	0.000503735	1.2078268211	0.000002099	1.2078276783	-0.000000000
1.2078268211	0.000002099	1.2078276783	-0.000000000		

Hence, a root is 1.2078 correct up to five significant figures.

2.8 FIXED-POINT ITERATION METHOD

Fixed-point iteration is a method for solving an equation of the form *f(x) = 0*. The method is carried out by rewriting the equation in the form:

$$x = g(x) \qquad (2.17)$$

Obviously, when x is the solution of *f(x) = 0*, the left side and the right side of Eq. (2.17) are equal. This is illustrated graphically by plotting y = x and y = g(x), as shown in Fig. 2-10.

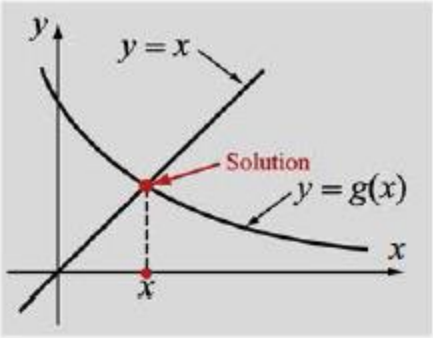


Figure 2-10: Fixed-point iteration method.

The point of intersection of the two plots, called the fixed point, is the solution. The numerical value of the solution is determined by an iterative process. It starts by taking a value of x near the fixed point as the first guess for the solution and substituting it in *g(x)*. The value of *g(x)* that is obtained is the new (second) estimate for the solution. The second value is then substituted back in *g(x)*, which then gives the third estimate of the solution. The iteration formula is thus given by:

$$x_{i+1} = g(x_i) \qquad (2.18)$$

The function g(x) is called ***the iteration function***.

• When the method works, the values of x that are obtained are successive iterations that progressively converge toward the solution. Two such cases are illustrated graphically in Fig. 2-11. The solution process starts by choosing point x_1 on the x-axis and drawing a vertical line that intersects the curve y = g(x) at point $g(x_1)$. Since $x_2 = g(x_1)$, a horizontal line is drawn from point $(x_1, g(x_1))$ toward the line y = x. The intersection point gives the location of x_2 . From x_2 a vertical line is drawn toward the curve y = g(x). The intersection point is now $(x_2, g(x_2))$, and $g(x_2)$ is also the value of x_3. From point $(x_2, g(x_2))$ a horizontal line is drawn again toward y = x, and the intersection point gives the location of x3 . As the process continues the intersection points converge toward the fixed point or the true solution x_{rs}.

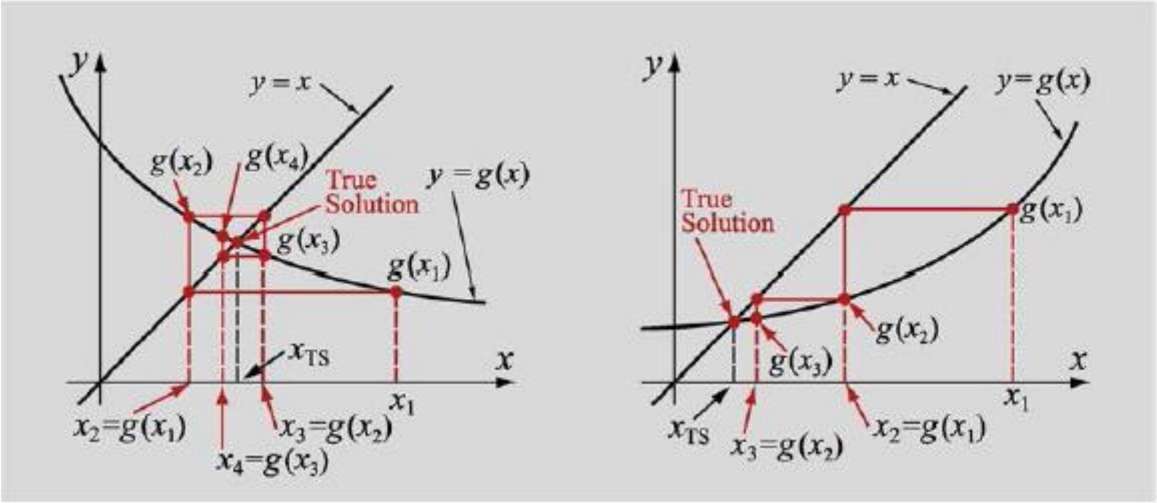


Figure 2-11: Convergence of the fixed-point iteration method.

• It is possible, however, that the iterations will not converge toward the fixed point, but rather diverge away. This is shown in Fig. 2-12. The figure shows that even though the starting point is close to the solution, the subsequent points are moving farther away from the solution.

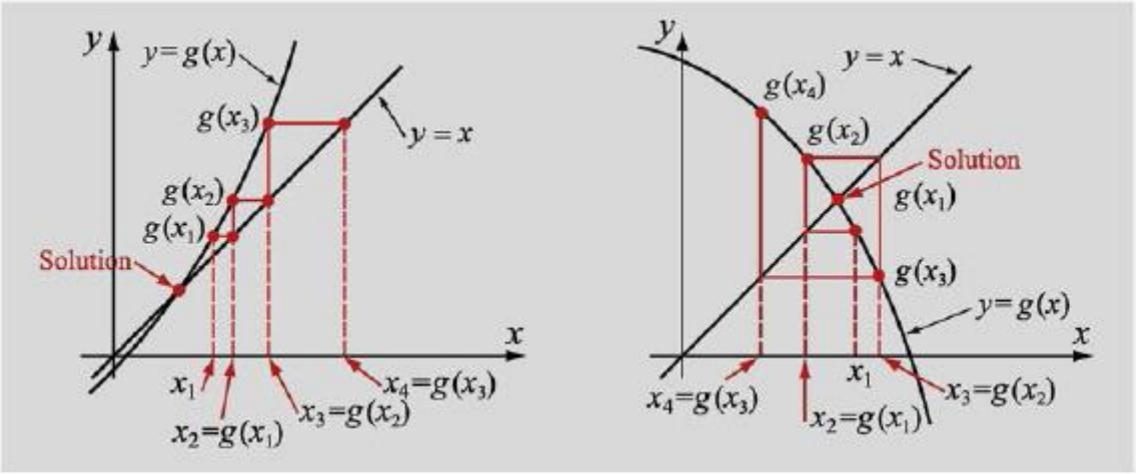


Figure 2-12: Divergence of the fixed-point iteration method.

• Sometimes, the form *f(x) = 0* does not lend itself to deriving an iteration formula of the form *x = g(x)* . In such a case, one can always add and subtract *x* to *f (x)* to obtain *x + f (x) - x = 0*. The last equation can be rewritten in the form that can be used in the fixed-point iteration method:

$$x = x + f(x) = g(x)$$

Choosing the appropriate iteration function g(x)

For a given equation *f(x) = 0*, the iteration function is not unique since it is possible to change the equation into the form *x = g(x)* in different ways. This means that several iteration functions g(x) can be written for the same equation. A *g(x)* that should be used in Eq. (2.18) for the iteration process is one for which the iterations converge toward the solution. There might be more than one form that can be used, or it may be that none of the forms is appropriate so that the fixed-point iteration method cannot be used to solve the equation. In cases where there are multiple solutions, one iteration function may yield one root, while a different function yields other roots. Actually, it is possible to determine ahead of time if the iterations converge or diverge for a specific g(x).

The fixed-point iteration method converges if, in the neighbourhood of the fixed point, the derivative of g(x) has an absolute value that is smaller than 1:

$$\boldsymbol{|g'(x)| < 1} \qquad \textbf{(2.19)}$$

As an example, consider the equation:

$$xe^{0.5x} + l.2x - 5 = 0 \qquad (2.20)$$

A plot of the function $f(x) = xe^{0.5x} + l.2x- 5$ (see Fig. 2-13) shows that the equation has a solution between *x= 1 and x= 2* .

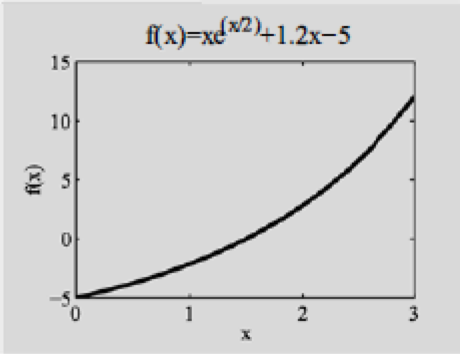


Figure 2-13: A plot of f(x) = xe^{xl2} + l.2x - 5.

Equation (2 .20) can be rewritten in the form *x= g (x)* in different ways. Three possibilities are discussed next.

Case a: $x = \frac{5-xe^{x/2}}{1.2}$

In this case $g(x) = \frac{5-xe^{x/2}}{1.2}$ and $g'(x) = \frac{-(e^{\frac{x}{2}}+0.5xe^{\frac{x}{2}})}{1.2}$

The values of g'(x) at points *x= 1* and *x= 2* , which are in the neighborhood of the solution, are:

$g'(1) = \frac{-(e^{\frac{1}{2}}+0.5(1)e^{\frac{1}{2}})}{1.2}$= -2.0609

$g'(2) = \frac{-(e^{\frac{2}{2}}+0.5(2)e^{\frac{2}{2}})}{1.2}$= -4.5305

Case b: $x = \frac{5}{e^{0.5x}+1.2}$

In this case $g(x) = \frac{5}{e^{0.5x}+1.2}$ and $g'(x) = \frac{-5e^{0.5x}}{2(e^{0.5x}+1.2)^2}$

The value of g'(x) at points *x= 1* and *x= 2* , which are in the neighborhood of the solution, are:

$g'(1) = \frac{-5e^{0.5(1)}}{2(e^{0.5(1)}+1.2)^2}$= -0.5079

$g'(2) = \frac{-5e^{0.51(2)}}{2(e^{0.5(2)}+1.2)^2}$= -0.4426

Case c: $x = \frac{5-1.2x}{e^{0.5x}}$

In this case, $g(x) = \frac{5-1.2x}{e^{0.5x}}$ and $g'(x) = \frac{-3.7+0.6x}{e^{0.5x}}$

The value of g'(x) at points *x= 1* and *x= 2* , which are in the neighborhood of the solution, are:

$g'(1) = \frac{-3.7+0.6(1)}{e^{0.5(1)}}$= -1.8802

$g'(2) = \frac{-3.7+0.6(2)}{e^{0.5(2)}}$= -0.9197

These results show that the iteration function *g(x)* from Case b is the one that should be used since, in this case, $|g'(1)| < 1$and$|g'(2)| < 1$.

Substituting g(x) from Case b in Eq. (2.18) gives:

$$x_{i+1} = \frac{5}{e^{0.5x_i} + 1.2}$$

Starting with x_1 = 1, the first few iterations are:

$x_2 = \frac{5}{e^{0.5(1)}+1.2}$ = 1.755173 , $x_3 = \frac{5}{e^{0.5(1.755173)}+1.2}$ = 1.386928

$x_4 = \frac{5}{e^{0.5(1.386928)}+1.2}$ = 1.56219 , $x_5 = \frac{5}{e^{0.5(1.56219)}+1.2}$ = 1.477601

$x_6 = \frac{5}{e^{0.5(1.477601)}+1.2}$ = 1.518177 , $x_7 = \frac{5}{e^{0.5(1.518177\ )}+1.2}$ = 1.498654

As expected, the values calculated in the iterations are converging toward the actual solution, which is **x = 1.5050**. On the contrary, if the function g(x) from Case a is used in the iteration, the first few iterations are:

$x_2 = \frac{5-e^{1/2}}{1.2}$ = 2.792732

$x_3 = \frac{5-\ 2.792732*e^{\ 2.792732/2}}{1.2}$ = -5.23667

$x_4 = \frac{5+5.23667*e^{\ -5.23667/2}}{1.2}$ = 4.4849

$x_5 = \frac{5-4.4849*e^{\ 4.4849/2}}{1.2}$ = -31.0262

In this case, the iterations give values that diverge from the solution.

When should the iterations be stopped?

The true error (the difference between the true solution and the estimated solution) cannot be calculated since the true solution, in general, is not known. As with Newton's method, the iterations can be stopped either when the relative error or the tolerance in *f(x)* is smaller than some predetermined value.

Example 2.14

Find a real root of $x^3 - 2x - 3 = 0$, correct to three decimal places using the Successive Approximation method.

Solution:

Here $f(x) = x^3 - 2x - 3 = 0$ (E.1)

Also $f(1) = 1^3 - 2(1) - 3 = -4 < 0$

and $f(2) = 2^3 - 2(2) - 3 = 1 > 0$

Therefore, root of Eq.(E.1) lies between 1 and 2. Since $f(1) < f(2)$, we can take the initial approximation $x_0 = 1$. Now, Eq. (E.1) can be rewritten as

$x^3 = 2x + 3$

or $x = (2x + 3)^{1/3} = \varphi(x)$

The successive approximations of the root are given by

$x_1 = \varphi(x_0) = (2x_0 + 3)^{1/3} = [2(1) + 3]^{1/3} = 1.709975947$

$x_2 = \varphi(x_1) = (2x_1 + 3)^{1/3} = [2(1.709975947) + 3]^{1/3} = 1.858562875$

$x_3 = \varphi(x_2) = (2x_2 + 3)^{1/3} = [2(1.858562875) + 3]^{1/3} = 1.88680851$

$x_4 = \varphi(x_3) = (2x_3 + 3)^{1/3} = [2(1.88680851) + 3]^{1/3} = 1.892083126$

$x_5 = \varphi(x_4) = (2x_4 + 3)^{1/3} = [2(1.892083126) + 3]^{1/3} = 1.89306486$

Hence, the real roots of $f(x) = 0$ is 1.893 correct to three decimal places.

Example 2.15

Find a real root of $\cos x - 3x + 5 = 0$. Correct to four decimal places using the fixed point method.

Solution:

Here, we have

$f(x) = \cos x - 3x + 5 = 0$ (E.1)

$f(0) = \cos(0) - 3(0) + 5 = 5 > 0$

$f(\pi/2) = \cos(\pi/2) - 3(\pi/2) + 5 = -3\pi/2 + 5 < 0$

Also $f(0)\, f(\pi/2) < 0$

Hence, a root of $f(x) = 0$ lies between 0 and $\pi/2$.

The given Eq. (E.1) can be written as:

$$x = \frac{1}{3}[5 + \cos x]$$

Here $$\phi(x) = \frac{1}{3}[5 + \cos x] \text{ and } \phi'(x) = -\frac{\sin x}{3}$$

$$|\phi'(x)| = \left|\frac{\sin x}{3}\right| < 1 \text{ in } (0, \pi/2)$$

Hence, the successive approximation method applies.

Let

$$\begin{aligned}
x_0 &= 0\\
x_1 &= \phi(x_0) = \frac{1}{3}\,[5+\cos 0] = 2\\
x_2 &= \phi(x_1) = \frac{1}{3}\,[5+\cos(2)] = 1.52795\\
x_3 &= \phi(x_2) = \frac{1}{3}\,[5+\cos(1.52795)] = 1.68094\\
x_4 &= \phi(x_3) = \frac{1}{3}\,[5+\cos(1.68094)] = 1.63002\\
x_5 &= \phi(x_4) = \frac{1}{3}\,[5+\cos(1.63002)] = 1.64694\\
x_6 &= \phi(x_5) = \frac{1}{3}\,[5+\cos(1.64694)] = 1.64131\\
x_7 &= \phi(x_6) = \frac{1}{3}\,[5+\cos(1.64131)] = 1.64318\\
x_8 &= \phi(x_7) = \frac{1}{3}\,[5+\cos(1.64318)] = 1.64256\\
x_9 &= \phi(x_8) = \frac{1}{3}\,[5+\cos(1.64256)] = 1.64277\\
x_{10} &= \phi(x_9) = \frac{1}{3}\,[5+\cos(1.64277)] = 1.64270
\end{aligned}$$

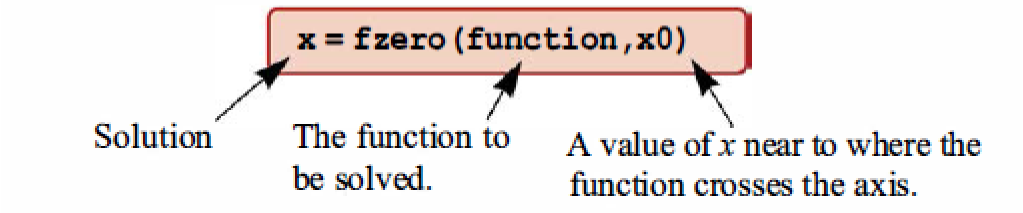
Hence, the root of the equation is 1.6427 correct to four decimal places.

2.9 Use of MATLAB Built-in Functions for Solving NONLINEAR EQUATIONS

MATLAB has two built-in functions for solving equations with one variable. The ***fzero command*** can be used to find a root of any equation, and the ***roots command*** can be used for finding the roots of a polynomial.

2.9.1 The *fzero* Command

The *fzero* command can be used to solve an equation (in the form f(x) = 0) with one variable. The user needs to know approximately where the solution is, or if there are multiple solutions, which one is desired. The form of the command is:



x = fzero(function,x0)
Solution
The function to be solved.
A value of x near to where the function crosses the axis.

- **x** is the solution, which is a scalar. A value of x near to where the function crosses the axis.
- **function** is the function whose root is desired. It can be entered in three different ways:

1. The simplest way is to enter the mathematical expression as a string.
2. The function is first written as a user-defined function, and then the function handle is entered.
3. The function is written as an anonymous function, and then its name (which is the name of the handle) is entered.

- The function has to be written in a standard form. For example, if the function to be solved is xe -x = 0.2, it has to be written as

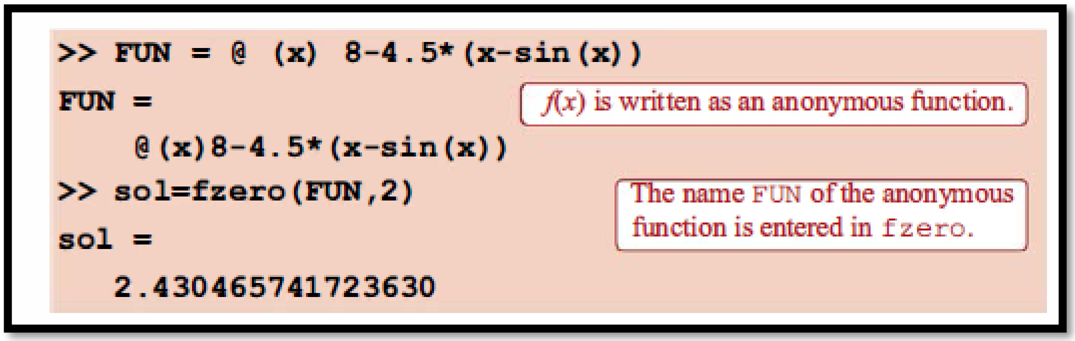
$f(x) = xe^{-x}-0.2 = 0$. If this function is entered into the fzero commands as a string, it is typed as:

*'x*exp (-x) -0. 2'.*

- When a function is entered as a string, it cannot include predefined variables. For example, if the function to be entered is $f(x) = xe^{-x} -0.2$, it is not possible to first define b=0. 2 and then enter:

*'x*exp (-x) -b'.*

• x_0 can be a scalar or a two-element vector. If it is entered as a scalar, it has to be a value of x near the point where the function crosses the x-axis. If x_0 is entered as a vector, the two elements have to be points on opposite sides of the solution. When a function has more than one solution, each solution can be determined separately by using the *fzero* function and entering values for x_0 that are near each of the solutions. Usage of the *fzero* command is illustrated next for solving equation *8 - 4.5(x- sin(x))*. The function *f(x) = 8 - 4.5(x-sin(x))* is first defined as an anonymous function named FUN. Then the name FUN is entered as an input argument in the function *fzero*.



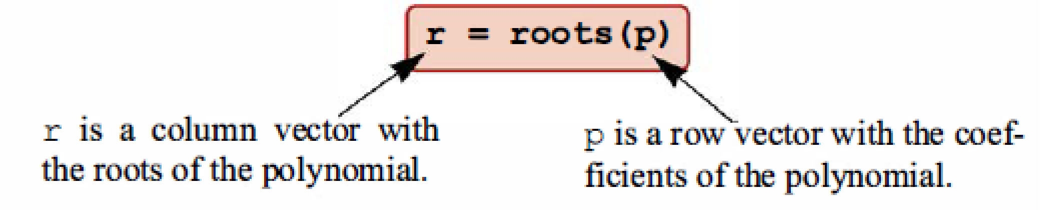
```
>> FUN = @ (x)  8-4.5*(x-sin(x))
FUN =
     @(x)8-4.5*(x-sin(x))
>> sol=fzero(FUN,2)
sol =
   2.430465741723630
```

f(*x*) is written as an anonymous function.

The name FUN of the anonymous function is entered in fzero.

2.9.2 The *roots* Command

The *roots* command can be used to find the roots of a polynomial. The form of the command is:



```
r = roots(p)
```

r is a column vector with the roots of the polynomial.

p is a row vector with the coefficients of the polynomial.

2.10 PROBLEMS

1. Determine the root of $f\,(x) = x – 2e^{-x}$ by:

(a) Using the bisection method. Start with a= 0 and b= 1, and carry out the first three iterations.

(b) Using the secant method. Start with the two points, $x_1 = 0$ and $x_2 = 1$, and carry out the first three iterations.

(c) Using Newton's method. Start at $x_1 = 1$ and carry out the first three iterations.

2. Determine the fourth root of 200 by finding the numerical solution of the equation $x^4\text{-}\ 200 = 0$. Use Newton's method. Start at $x = 8$ and carry out the first five iterations.

3. Determine the positive root of the polynomial $x^3 + 0.6x^2 + 5\ .6\ x\text{-}4.8$.

(a) Plot the polynomial and choose a point near the root for the first estimate of the solution. Using Newton's method, determine the approximate solution in the first four iterations.

(b) From the plot in part (a), choose two points near the root to start the solution process with the secant method. Determine the approximate solution in the first four iterations.

4. The equation x^3 *-x -*e^x *- 2 = 0* has a root between $x = 2$ and $x = 3$.

(a) Write four different iteration functions for solving the equation using the fixed-point iteration method.

(b) Determine which g(x) from part (a) could be used according to the condition in Eq. (2.19).

(c) Carry out the first five iterations using the g(x) determined in part (b), starting with $x = 2$.

5. Use the Bisection method to compute the root of $e^x – 3x = 0$ correct to three decimal places in the interval (1.5, 1.6).

6. Use the Bisection method to find a root of the equation $x^3 – 4x – 9 = 0$ in the interval (2, 3), accurate to four decimal places.

7. Use the method of False Position to find a root correct to three decimal places of the function $x^3 - 4x - 9 = 0$.

8. A root of $f(x) = x^3 - 10x^2 + 5 = 0$ lies close to $x = 0.7$. Determine this root with the Newton-Raphson method to five decimal accuracy.

9. A root of $f(x) = x^3 - x^2 - 5 = 0$ lies in the interval (2, 3). Determine this root with the Newton-Raphson method for four decimal places.

10. Use the fixed point method to find a root of the equation $e^x - 3x = 0$ in the interval
(0, 1) accurate to four decimal places.

11. Use the method of Successive Approximation to determine a solution accurate to within 10^{-2} for $x^4 - 3x^2 - 3 = 0$ on [1, 2]. Use $x_0 = 1$.

Chapter 3: Solving a System of Linear Equations

3.1 BACKGROUND

Systems of linear equations that have to be solved simultaneously arise in problems that include several (possibly many) variables that are dependent on each other. Such problems occur not only in engineering and science but in virtually any discipline (business, statistics, economics, etc.). A system of two (or three) equations with two (or three) unknowns can be solved manually by substitution or other mathematical methods (e.g., Cramer's rule). Solving a system in this way is practically impossible as the number of equations (and unknowns) increases beyond three.

3. 1. 1 Overview of Numerical Methods for Solving a System of Linear Algebraic Equations

The general form of a system of n linear algebraic equations is:

$$\left.\begin{array}{r} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{array}\right\} \qquad (3.1)$$

The matrix form of the equations is shown in Fig. 3-1.

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$$

Figure 3-1: A system of n linear algebraic equations.

Two types of numerical methods, direct and iterative, are used for solving systems of linear algebraic equations. In direct methods, the solution is calculated by performing arithmetic operations with the equations. In iterative methods, an initial approximate solution is assumed and then used in an iterative process for obtaining successively more accurate solutions.

Direct methods

In direct methods, the system of equations that is initially given in the general form, Eqs. (3.1), is manipulated to an equivalent system of equations that can be easily solved. Three systems of equations that can be easily solved are the upper triangular, lower triangular, and diagonal forms. The upper triangular form is shown in Eqs. (3.2),

$$\left.\begin{array}{r} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\ a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\ a_{33}x_3 + \cdots + a_{3n}x_n = b_3 \\ \vdots \\ a_{nn}x_n = b_n \end{array}\right\} \qquad (3.2)$$

and is written in a matrix form for a system of four equations in Fig. 3-2.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Figure 3-2: A system of four equations in upper triangular form.

The system in this form has all zero coefficients below the diagonal and is solved by a procedure called back substitution. It starts with the last equation, which is solved for x_n. The value of x_n is then substituted in the next-to-the-last equation, which is solved for x_{n-1}. The process continues, in the same manner, all the way up to the first equation. In the case of four equations, the solution is given by:

$$x_4 = \frac{b_4}{a_{44}} \quad , \quad x_3 = \frac{b_3 - a_{34}x_4}{a_{33}} \quad , \quad x_2 = \frac{b_2 - (a_{23}x_3 + a_{24}x_4)}{a_{22}}$$

$$\text{and} \quad x_1 = \frac{b_1 - (a_{12}x_2 + a_{13}x_3 + a_{14}x_4)}{a_{11}}$$

For a system of n equations in upper triangular form, general formula for the solution using back substitution is:

$$x_n = \frac{b_n}{a_{nn}} \quad , \quad x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}} \quad , i = n-1, n-2, \dots, 1 \qquad (3.3)$$

In Section 3.2 the upper triangular form and back substitution are used in the Gauss elimination method.

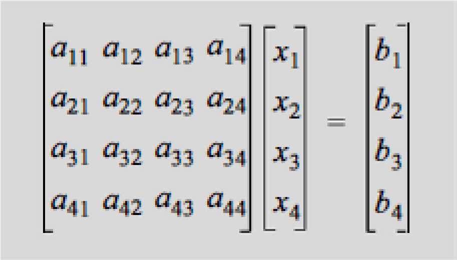
<u>Exc:</u> Write a program for Eq. (3.3).

3.2 GAUSS ELIMINATION METHOD

The Gauss elimination method is a procedure for solving a system of linear equations. In this procedure, a system of equations that are given in a general form is manipulated to be in upper triangular form, which is then solved by using back substitution (see Section 3.1.1). For a set of four equations with four unknowns, the general form is given by:

$$\left.\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 && (3.4a)\\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= b_2 && (3.4b)\\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 &= b_3 && (3.4c)\\
a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 &= b_{41} && (3.4d)
\end{aligned}\right\} \qquad (3.4)$$

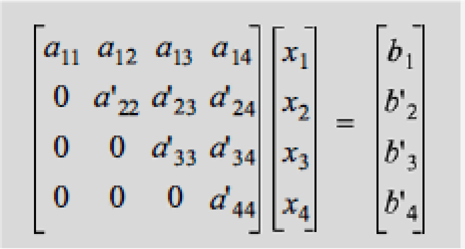
The matrix form of the system is shown in Fig. 3-3.



$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Figure 3-3: Matrix form of a system of four equations.

In the Gauss elimination method, the system of equations is manipulated into an equivalent system of equations that has the form:



$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a'_{33} & a'_{34} \\ 0 & 0 & 0 & a'_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{bmatrix}$$

Figure 3-4: Matrix form of the equivalent system.

In general, various mathematical manipulations can be used for converting a system of equations from the general form displayed in Eqs. (4.10) to the upper triangular form. One, in particular, the Gauss elimination method, is described next. The procedure can be easily programmed in a computer code.

Gauss elimination procedure (forward elimination)

The Gauss elimination procedure is first illustrated for a system of four equations with four unknowns. The starting point is the set of equations that are given by Eqs. (3.4). Converting the system of equations to the upper triangular form is done in steps.

<u>Step 1:</u> In the first step, the first equation is unchanged, and the terms that include the variable x_1 in all the other equations are eliminated. This is done one equation at a time by using the first equation, which is called the ***pivot equation***. The coefficient a_{11} is called the ***pivot coefficient***, or the ***pivot element***. To eliminate the term $a_{i1}x_1$ in Eq. (3.4b), the pivot equation, Eq. (3.4a), is multiplied by $m_{21} = \frac{a_{21}}{a_{11}}$, and then the equation is subtracted from Eq. (3.4b):

$$\begin{array}{rl} & a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = b_2 \\ - & m_{21}(a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4) = m_{21}b_1 \\ \hline & 0 + \underbrace{(a_{22} - m_{21}a_{12})}_{a'_{22}}x_2 + \underbrace{(a_{23} - m_{21}a_{13})}_{a'_{23}}x_3 + \underbrace{(a_{24} - m_{21}a_{14})}_{a'_{24}}x_4 = \underbrace{b_2 - m_{21}b_1}_{b'_2} \end{array}$$

It should be emphasized here that the pivot equation, Eq. (3.4a), itself is not changed. The matrix form of the equations after this operation is shown in Fig. 3-5.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Figure 3-5: Matrix form of the system after eliminating a_{21}.

Next, the term $a_{31}x_1$ in Eq. (3.4c) is eliminated. The pivot equation, Eq. (3.4a), is multiplied by $m_{31} = \frac{a_{31}}{a_{11}}$ and then is subtracted from Eq. (3.4c):

$$\begin{array}{rl} & a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = b_3 \\ - & m_{31}(a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4) = m_{31}b_1 \\ \hline & 0 + \underbrace{(a_{32} - m_{31}a_{12})}_{a'_{32}}x_2 + \underbrace{(a_{33} - m_{31}a_{13})}_{a'_{33}}x_3 + \underbrace{(a_{34} - m_{31}a_{14})}_{a'_{34}}x_4 = \underbrace{b_3 - m_{31}b_1}_{b'_3} \end{array}$$

The matrix form of the equations after this operation is shown in Fig. 3-6.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b'_3 \\ b_4 \end{bmatrix}$$

Figure 3-6: Matrix form of the system after eliminating a_{31}.

Next, the term $a_{41}x_1$ in Eq. (3.4d) is eliminated. The pivot equation, Eq. (3.4a), is multiplied by $m_{31} = \frac{a_{31}}{a_{11}}$ and then is subtracted from Eq. (4.3d):

$$\begin{array}{rl}
- & a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = b_4 \\
 & m_{41}(a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4) = m_{41}b_1 \\
\hline
 & 0 + \underbrace{(a_{42} - m_{41}a_{12})}_{a'_{42}}x_2 + \underbrace{(a_{43} - m_{41}a_{13})}_{a'_{43}}x_3 + \underbrace{(a_{44} - m_{41}a_{14})}_{a'_{44}}x_4 = \underbrace{b_4 - m_{41}b_1}_{b'_4}
\end{array}$$

This is the end of **Step 1.** The system of equations now has the following form:

$$\left.\begin{array}{lr}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1 & (3.5a) \\
0 + a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4 = b'_2 & (3.5b) \\
0 + a'_{32}x_2 + a'_{33}x_3 + a'_{34}x_4 = b'_3 & (3.5c) \\
0 + a'_{42}x_2 + a'_{43}x_3 + a'_{44}x_4 = b'_4 & (3.5d)
\end{array}\right\} \quad (3.5)$$

The matrix form of the equations after this operation is shown in Fig. 3-7. Note that the result of the elimination operation is to reduce the first column entries, except a_{11} (the pivot element), to zero.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{bmatrix}$$

Figure 3-7: Matrix form of the system after eliminating a_{41}.

Step 2: In this step, Eqs. (3.5a) and (3.5b) are not changed, and the terms that include the variable x_2 in Eqs. (3.5c) and (3.5d) are eliminated. In this step, Eq. (3.5b) is the pivot equation, and the coefficient a'_{22} is the pivot coefficient. To eliminate the term $a'_{32}x_2$ in Eq. (3.5c), the pivot equation, Eq. (3.5b), is multiplied by $m_{32} = \frac{a'_{32}}{a'_{22}}$ and then is subtracted from Eq. (3.5c):

$$\begin{array}{rl}
- & a'_{32}x_2 + a'_{33}x_3 + a'_{34}x_4 = b'_3 \\
 & m_{32}(a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4) = m_{32}b'_2 \\
\hline
 & 0 + \underbrace{(a'_{33} - m_{32}a'_{23})}_{a''_{33}}x_3 + \underbrace{(a'_{34} - m_{32}a'_{24})}_{a''_{34}}x_4 = \underbrace{b'_3 - m_{32}b'_2}_{b''_3}
\end{array}$$

The matrix form of the equations after this operation is shown in Fig. 3-8.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b''_3 \\ b'_4 \end{bmatrix}$$

Figure 3-8: Matrix form of the system after eliminating a_{32}•

Next, the term $a'_{42}x_2$ in Eq. (3.5d) is eliminated. The pivot equation, Eq. (3.5b), is multiplied by $m_{42} = \frac{a'_{42}}{a'_{22}}$ and then is subtracted from Eq. (3.5d):

$$\begin{array}{rl} - & a'_{42}x_2 + a'_{43}x_3 + a'_{44}x_4 = b'_4 \\ & m_{42}(a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4) = m_{42}b'_2 \\ \hline & 0 + \underbrace{(a'_{43} - m_{42}a'_{23})}_{a''_{43}}x_3 + \underbrace{(a'_{44} - m_{42}a'_{24})}_{a''_{44}}x_4 = \underbrace{b'_4 - m_{42}b'_2}_{b''_4} \end{array}$$

The matrix form of the equations after this operation is shown in Fig. 3-9.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & 0 & a''_{43} & a''_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b''_3 \\ b''_4 \end{bmatrix}$$

Figure 3-9: Matrix form of the system after eliminating *a_{42}*•

This is the end of **Step 2**. The system of equations now has the following form:

$$\left.\begin{array}{lr} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1 & (3.6a) \\ 0 + a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4 = b'_2 & (3.6b) \\ 0 \;+\; 0 \;+ a''_{33}x_3 + a''_{34}x_4 = b''_3 & (3.6c) \\ 0 \;+\; 0 \;+ a''_{43}x_3 + a''_{44}x_4 = b''_4 & (3.6d) \end{array}\right\} \qquad (3.6)$$

<u>Step 3:</u> In this step, Eqs. (3.6a), (3.6b), and (3.6c) are not changed, and the term that includes the variable x_3 in Eq. (3.6d) is eliminated. In this step, Eq. (3.6c) is the pivot equation, and the coefficient a''_{33} is the pivot coefficient. To eliminate the term $a''_{43}x_3$ in Eq. (3.6d), the pivot equation is multiplied by $m_{43} = \frac{a''_{43}}{a''_{33}}$ and then is subtracted from Eq. (3.6d):

$$\begin{array}{rl} - & a''_{43}x_3 + a''_{44}x_4 = b''_4 \\ & m_{43}(a''_{33}x_3 + a''_{34}x_4) = m_{43}b''_3 \\ \hline & \underbrace{(a''_{44} - m_{43}a''_{34})}_{a'''_{44}}x_4 = \underbrace{b''_4 - m_{43}b''_3}_{b'''_4} \end{array}$$

This is the end of **Step 3**. The system of equations is now in an upper triangular form:

$$\left.\begin{array}{lr} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1 & (3.7a) \\ 0 + a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4 = b'_2 & (3.7b) \\ 0 \;+\; 0 \;+ a''_{33}x_3 + a''_{34}x_4 = b''_3 & (3.7c) \\ 0 \;+\; 0 \;+\; 0 \;+ a'''_{44}x_4 = b'''_4 & (3.7d) \end{array}\right\} \qquad (3.7)$$

The matrix form of the equations is shown in Fig. 3-10. Once transformed to upper triangular form, the equations can be easily solved by using back substitution.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & 0 & 0 & a''_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b''_3 \\ b'''_4 \end{bmatrix}$$

Figure 3-10: Matrix form of the system after eliminating a_{43} •

The three steps of the Gauss elimination process are illustrated together in Fig. 3-11.

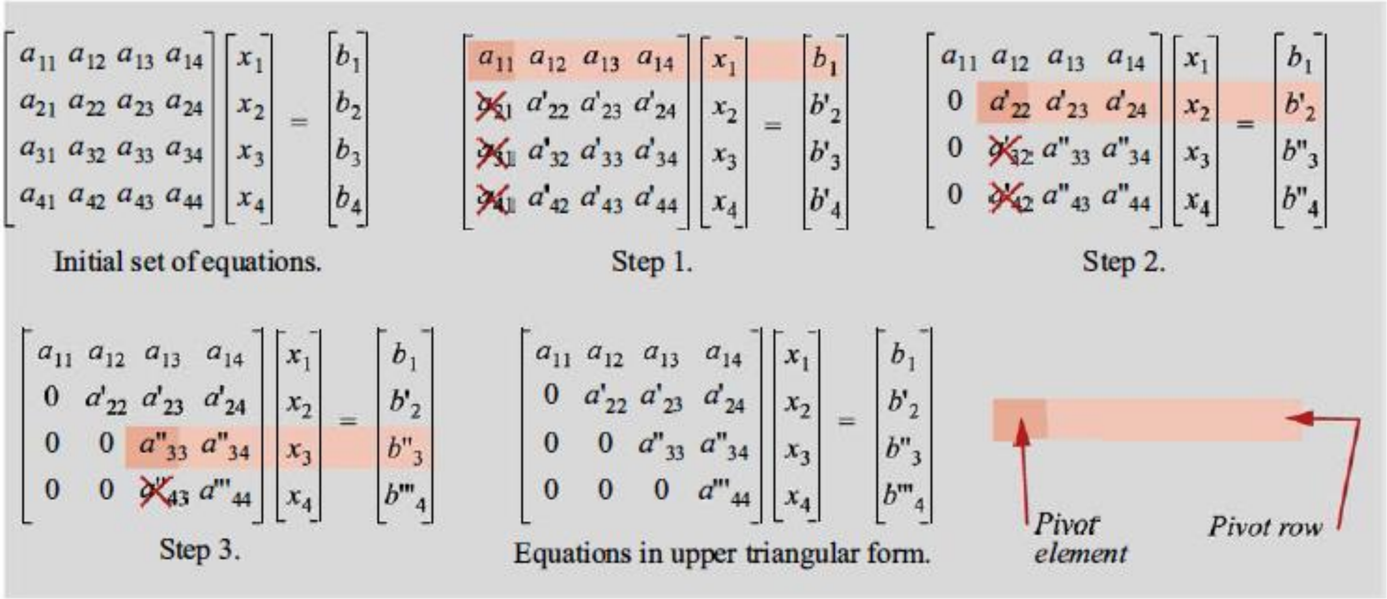

Initial set of equations.
Step 1.
Step 2.
Step 3.
Equations in upper triangular form.
Pivot element
Pivot row

Figure 3-11: Gauss elimination procedure.

Example 3-1: Solve the following system of four equations using the Gauss elimination method.

$$4x_1\text{-}2x_2\text{-}\ 3x_3\text{+}6x_4 = 12$$
$$-6x_1 + 7\ x_2 + 6.5x_3 - 6x_4 = -6.5$$
$$x_1 + 7.5x_2 + 6.25x_3 + 5.5x_4 = 1\ 6$$
$$-12x_1 + 22x_2 + 15.5x_3 - x_4 = 17$$

SOLUTION: The solution follows the steps presented in the previous pages.

Step 1: The first equation is the pivot equation, and 4 is the pivot coefficient.

Multiply the pivot equation by m_{21} = (-6)/ 4 = -1.5 and subtract it from the second equation:

$$\begin{array}{rl} - & -6x_1 + 7x_2 + 6.5x_3 - 6x_4 = -6.5 \\ & (-1.5)(4x_1 - 2x_2 - 3x_3 + 6x_4) = (-6/4)\cdot 12 \\ \hline & 0x_1 + 4x_2 + 2x_3 + 3x_4 = 11.5 \end{array}$$

Multiply the pivot equation by m_{31}= (1I4) =0 .25 and subtract it from the third equation:

$$\begin{array}{rl} - & x_1 + 7.5x_2 + 6.25x_3 + 5.5x_4 = 16 \\ & (0.25)(4x_1 - 2x_2 - 3x_3 + 6x_4) = (1/4)\cdot 12 \\ \hline & 0x_1 + 8x_2 + 7x_3 + 4x_4 = 13 \end{array}$$

Multiply the pivot equation by m_{41}=(-12)/4 = -3 and subtract it from the fourth equation:

$$\begin{array}{rl} - & -12x_1 + 22x_2 + 15.5x_3 - x_4 = 17 \\ & (-3)(4x_1 - 2x_2 - 3x_3 + 6x_4) = -3\cdot 12 \\ \hline & 0x_1 + 16x_2 + 6.5x_3 + 17x_4 = 53 \end{array}$$

At the end of Step 1, the four equations have the form:

$$\begin{aligned} 4x_1 - 2x_2 - 3x_3 + 6x_4 &= 12 \\ 4x_2 + 2x_3 + 3x_4 &= 11.5 \\ 8x_2 + 7x_3 + 4x_4 &= 13 \\ 16x_2 + 6.5x_3 + 17x_4 &= 53 \end{aligned}$$

Step 2: The second equation is the pivot equation, and 4 is the pivot coefficient. Multiply the pivot equation by m_{32} = 8/ 4 = 2 and subtract it from the third equation:

$$\begin{array}{rr} - & 8x_2 + 7x_3 + 4x_4 = 13 \\ & 2(4x_2 + 2x_3 + 3x_4) = 2 \cdot 11.5 \\ \hline & 0x_2 + 3x_3 - 2x_4 = -10 \end{array}$$

Multiply the pivot equation by m_{42} = 16/4 = 4 and subtract it from the fourth equation:

$$\begin{array}{rr} - & 16x_2 + 6.5x_3 + 17x_4 = 53 \\ & 4(4x_2 + 2x_3 + 3x_4) = 4 \cdot 11.5 \\ \hline & 0x_2 - 1.5x_3 + 5x_4 = 7 \end{array}$$

At the end of Step 2, the four equations have the form:

$$\begin{aligned} 4x_1 - 2x_2 - 3x_3 + 6x_4 &= 12 \\ 4x_2 + 2x_3 + 3x_4 &= 11.5 \\ 3x_3 - 2x_4 &= -10 \\ -1.5x_3 + 5x_4 &= 7 \end{aligned}$$

Step 3: The third equation is the pivot equation, and 3 is the pivot coefficient. Multiply the pivot equation by m_{43} = (-1.5)/ 3 = -0.5 and subtract it from the fourth equation:

$$\begin{array}{rr} - & -1.5x_3 + 5x_4 = 7 \\ & -0.5(3x_3 - 2x_4) = -0.5 \cdot -10 \\ \hline & 0x_3 + 4x_4 = 2 \end{array}$$

At the end of Step 3, the four equations have the form:

$$\begin{aligned} 4x_1 - 2x_2 - 3x_3 + 6x_4 &= 12 \\ 4x_2 + 2x_3 + 3x_4 &= 11.5 \\ 3x_3 - 2x_4 &= -10 \\ 4x_4 &= 2 \end{aligned}$$

Once the equations are in this form, the solution can be determined by back substitution. The value of x_4 is determined by solving the fourth equation:

$$x_4 = 2/4 = 0.5$$

Next, x_4 is substituted in the third equation, which is solved for x_3 :

$$x_3 = \frac{-10 + 2x_4}{3} = \frac{-10 + 2(0.5)}{3} = -3$$

Next, x_4 and x_3 are substituted in the second equation, which is solved for x_2:

$$x_2 = \frac{11.5 - 2x_3 - 3x_4}{4} = \frac{11.5 - 2(-3) - 3(0.5)}{4} = 4$$

Lastly, x_4, x_3 and x_2 are substituted in the first equation, which is solved for x_1 :

$$x_1 = \frac{12 + 2x_2 + 3x_3 - 6x_4}{4} = \frac{12 + 2(4) + 3(-3) - 6(0.5)}{4} = 2$$

3.2.1 Potential Difficulties When Applying the Gauss Elimination Method

The pivot element is zero

Since the pivot row is divided by the pivot element, a problem will arise during the execution of the Gauss elimination procedure if the value of the pivot element is equal to zero. As shown in the next section, this situation can be corrected by changing the order of the rows. In a procedure called pivoting, the pivot row that has the zero pivot element is exchanged with another row that has a nonzero pivot element.

The pivot element is small relative to the other terms in the pivot row

Significant errors due to rounding can occur when the pivot element is small relative to other elements in the pivot row. This is illustrated by the following example.

Consider the following system of simultaneous equations for the unknowns x_1 and x_2:

$$0.0003x_1 + 12.34x_2 = 12.343$$
$$0.432l\ x_l + x2 = 5.321 \qquad (3.8)$$

The exact solution of the system is $x_1 = 10$ and $x_2 = 1$. The error due to rounding is illustrated by solving the system using Gaussian elimination on a machine with limited precision so that only four significant figures are retained with rounding. When the first equation of Eqs. (3.8) is entered, the constant on the right-hand side is rounded to 12.34.

The solution starts by using the first equation as the pivot equation and $a_{11}= 0.0003$ as the pivot coefficient. In the first step, the pivot equation is multiplied by $m_{21}= 0.4321/0.0003 = 1440$. With four significant figures and rounding, this operation gives:

$$(1440)(0.0003x_1 + 12.34x_2) = 1440\ (\ 12.34)$$

or:

$$0.4320x_1 + 17770x_2 = 17770$$

The result is next subtracted from the second *equation* in Eqs. (3.8):

$$\begin{array}{rrcl} & 0.4321x_1 + x_2 & = & 5.321 \\ - & 0.4320x_1 + 17770x_2 & = & 17770 \\ \hline & 0.0001x_1 - 17770x_2 & = & -17760 \end{array}$$

After this operation, the system is:

$$0.0003x_1 + 12.34x_2 = 12.34$$
$$0.0001x_1 - 17770x_2 = -17760$$

Note that the a_{21} element is not zero but a very small number. Next, the value of x_2 is calculated from the second equation:

$$x_2 = \frac{-17760}{-17770} = 0.9994$$

Then x_2 is substituted in the first equation, which is solved for x_1:

$$x_1 = \frac{12.34 - 12.34(0.9994)}{0.0003} = \frac{0.01}{0.0003} = 33.33$$

The solution that is obtained for x_1 is obviously incorrect. The incorrect value is obtained because the magnitude of all is small when compared to the magnitude of a_{12}. Consequently, a relatively small error (due to round-off arising from the finite precision of a computing machine) in the value of x_2 can lead to a large error in the value of x_1. The problem can be easily remedied by exchanging the order of the two equations in Eqs. (3.8):

$$0.432l\ x_1 + x_2 = 5.321$$
$$0.0003x_1 + 12.34x_2 = 12.343 \qquad (3.9)$$

Now, as the first equation is used as the pivot equation, the pivot coefficient is $a_{ll}= 0.4321$. In the first step, the pivot equation is multiplied by $m_{21} = 0.0003/0.4321 = 0.0006943$. With four significant figures and rounding this operation gives:

$$(0.0006943)(0.4321x_1 + x_2) = 0.0006943\ (5.321)$$

or:

$$0.0003x_1 + 0.0006943x_2 = 0.003694$$

The result is next subtracted from the second equation in Eqs. (3.9):

$$\begin{array}{rl} - & 0.0003x_1 + 12.34x_2 = 12.34 \\ & \underline{0.0003x_1 + 0.0006943x_2 = 0.003694} \\ & 12.34x_2 = 12.34 \end{array}$$

After this operation, the system is:

$$0.4321x_1 + x_2 = 5.321$$
$$0x1 + 12.34x_2 = 12.34$$

Next, the value of x_2 is calculated from the second equation:

$$x_2 = \frac{12.34}{12.34} = 1$$

Then x_2 is substituted in the first equation that is solved for x_1:

$$x_1 = \frac{5.321 - 1}{0.4321} = 10$$

The solution that is obtained now is the exact solution.

In general, a more accurate solution is obtained when the equations are arranged (and rearranged every time a new pivot equation is used) such that the pivot equation has the largest possible pivot element. This is explained in more detail in the next section.

Round-off errors can also be significant when solving large systems of equations even when all the coefficients in the pivot row are of the same order of magnitude. This can be caused by a large number of operations (multiplication, division, addition, and subtraction) associated with large systems.

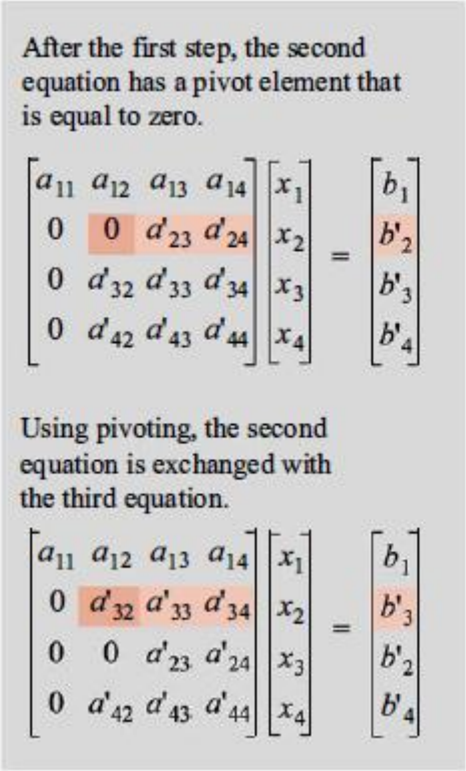
3.3 GAUSS ELIMINATION WITH PIVOTING

In the Gauss elimination procedure, the pivot equation is divided by the pivot coefficient. This, however, cannot be done if the pivot coefficient is zero. For example, for the following system of three equations:

$$0x_1 + 2x_2 + 3x_3 = 46$$
$$4x_1 - 3x_2 + 2x_3 = 16$$
$$2x_1 + 4x_2 - 3x_3 = 12$$

the procedure starts by taking the first equation as the pivot equation and the coefficient of x_1, which is 0, as the pivot coefficient. To eliminate the term $4x_1$ in the second equation, the pivot equation is supposed to be multiplied by 4/0 and then subtracted from the second equation. Obviously, this is not possible when the pivot element is equal to zero. The division by zero can be avoided if the order in which the equations are written is changed such that in the first equation the first coefficient is not zero. For example, in the system above, this can be done by exchanging the first two equations.

In the general Gauss elimination procedure, an equation (or a row) can be used as the pivot equation (pivot row) only if the pivot coefficient (pivot element) is not zero. If the pivot element is zero, the equation (i.e., the row) is exchanged with one of the equations (rows) that are below, which has a nonzero pivot coefficient. This exchange of rows, illustrated in Fig. 3-12, is called pivoting.



After the first step, the second equation has a pivot element that is equal to zero.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & 0 & a'_{23} & a'_{24} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{bmatrix}$$

Using pivoting, the second equation is exchanged with the third equation.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ 0 & 0 & a'_{23} & a'_{24} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_3 \\ b'_2 \\ b'_4 \end{bmatrix}$$

Figure 3-12: Illustration of pivoting.

Additional comments about pivoting

• If during the Gauss elimination procedure a pivot equation has a pivot element that is equal to zero, then if the system of equations that are being solved has a solution, an equation with a nonzero element in the pivot position can always be found.

• The numerical calculations are less prone to error and will have fewer round-off errors if the pivot element has a larger numerical absolute value compared to the other elements in the same row. Consequently, among all the equations that can be exchanged to be the pivot equation, it is better to select the equation whose pivot element has the largest absolute numerical value. Moreover, it is good to employ pivoting for the purpose of having a pivot equation with the pivot element that has the largest absolute numerical value at all times (even when pivoting is not necessary).

3.4 LU DECOMPOSITION METHOD

Background

The Gauss elimination method consists of two parts. The first part is the elimination procedure in which a system of linear equations that is given in a general form, [a][x] = [b], is transformed into an equivalent system of equations [a'][x] = [b'] in which the matrix of coefficients [a'] is upper triangular. In the second part, the equivalent system is solved by using back substitution. The elimination procedure requires many mathematical operations and significantly more computing time than the back substitution calculations. During the elimination procedure, the matrix of coefficients [a] and the vector [b] are both changed. This means that if there is a need to solve systems of equations that have the same left-hand-side terms (same coefficient matrix [a]) but different right-hand-side constants (different vectors [b]), the elimination procedure has to be carried out for each [b] again. Ideally, it would be better if the operations on the matrix of coefficients [a] were dissociated from those on the vector of constants [b] . In this way, the elimination procedure with [a] is done only once and then is used for solving systems of equations with different vectors [b] .

One option for solving various systems of equations [a][x] = [b] that have the same coefficient matrices [a] but different constant vectors [b] is to first calculate the inverse of the matrix [a] . Once the inverse matrix $[a]^{-1}$ is known, the solution can be calculated by: $[x] = [a]^{-1} [b]$.

Calculating the inverse of a matrix, however, requires many mathematical operations, and is computationally inefficient. A more efficient method of solution for this case is the LU decomposition method. In the LU decomposition method, the operations with the matrix [a] are done without using or changing, the vector [b], which is used only in the substitution part of the solution. The LU decomposition method can be used for solving a single system of linear equations, but it is especially advantageous for solving systems that have the same coefficient matrices [a] but different constant vectors [b].

The LU decomposition method

The *LU* decomposition method is a method for solving a system of linear equations *[a] [x] = [b]* . In this method the matrix of coefficients *[a]* is decomposed (factored) into a product of two matrices *[L]* and *[U]*:

$$[a] = [L][U] \qquad (3.10)$$

where the matrix *[L]* is a lower triangular matrix and *[U]* is an upper triangular matrix. With this decomposition, the system of equations to be solved has the form:

$$[L][U][x] = [b] \qquad (3.11)$$

To solve this equation, the product *[U][x]* is defined as:

$$[U][x] = [y] \qquad (3.12)$$

and is substituted in Eq. (3.11) to give:

$$[L][y] = [b] \qquad (3.13)$$

Now, the solution *[x]* is obtained in two steps. First, Eq. (3.13) is solved for *[y]*. Then, the solution *[y]* is substituted in Eq. (3.12), and that equation is solved for *[x]*. Since the matrix *[L]* is a lower triangular matrix, the solution *[y]* in Eq. (3.13) is obtained by using the **forward substitution** method. Once *[y]* is known and is substituted in Eq. (3.12), this equation is solved by using **back substitution**, since *[U]* is an upper triangular matrix. For a given matrix *[a]* several methods can be used to determine the corresponding *[L]* and *[U]*. One of them is related to the Gauss elimination method are described next.

3.4.1 LU Decomposition Using the Gauss Elimination Procedure

When the Gauss elimination procedure is applied to a matrix *[a]*, the elements of the matrices *[L]* and *[U]* are actually calculated. The upper triangular matrix *[U]* is the matrix of coefficients *[a]* that is obtained at the end of the procedure, as shown in Figs. 3-4 and 3- 11. The lower triangular matrix *[L]* is not written explicitly during the procedure, but the elements that make up the matrix are actually calculated along the way. The elements of *[L]* on the diagonal are all **1**, and the elements below the diagonal are the **multipliers** $\boldsymbol{m_{ij}}$ that multiply the pivot equation when it is used to eliminate the elements below the pivot coefficient. For the case of a system of four equations, the matrix of coefficients *[a]* is *(4 x 4)*, and the decomposition has the form:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ m_{21} & 1 & 0 & 0 \\ m_{31} & m_{32} & 1 & 0 \\ m_{41} & m_{42} & m_{43} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & 0 & 0 & a'''_{44} \end{bmatrix}$$

A numerical example illustrating LU decomposition is given next. It uses the information in the solution of Example 3- 1, where a system of four equations is solved by using the Gauss elimination method. The matrix *[a]* can be written from the given set of equations in the problem statement, and the matrix *[U]* can be written from the set of equations at the end of step 3 (page 35). The matrix *[L]* can be written by using the multipliers that are calculated in the solution. The decomposition has the form:

$$\begin{bmatrix} 4 & -2 & -3 & 6 \\ -6 & 7 & 6.5 & -6 \\ 1 & 7.5 & 6.25 & 5.5 \\ -12 & 22 & 15.5 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1.5 & 1 & 0 & 0 \\ 0.25 & 2 & 1 & 0 \\ -3 & 4 & -0.5 & 1 \end{bmatrix} \begin{bmatrix} 4 & -2 & -3 & 6 \\ 0 & 4 & 2 & 3 \\ 0 & 0 & 3 & -2 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

3.5 ITERATIVE METHODS

A system of linear equations can also be solved by using an iterative approach. The process, in principle, is the same as in the fixed-point iteration method used for solving a single nonlinear equation. In an iterative process for solving a system of equations, the equations are written in an explicit form in which each unknown is written in terms of the other unknown. The explicit form for a system of four equations is illustrated in Fig. 3-13.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 &= b_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 &= b_4 \end{aligned}$$

(a)

Writing the equations in an explicit form. $\longrightarrow$

$$\begin{aligned} x_1 &= [b_1 - (a_{12}x_2 + a_{13}x_3 + a_{14}x_4)]/a_{11} \\ x_2 &= [b_2 - (a_{21}x_1 + a_{23}x_3 + a_{24}x_4)]/a_{22} \\ x_3 &= [b_3 - (a_{31}x_1 + a_{32}x_2 + a_{34}x_4)]/a_{33} \\ x_4 &= [b_4 - (a_{21}x_1 + a_{42}x_2 + a_{43}x_3)]/a_{44} \end{aligned}$$

(b)

Figure 3-13: Standard (a) and explicit (b) forms of a system of four equations.

The solution process starts by assuming initial values for the unknowns (first estimated solution). In the first iteration, the first assumed solution is substituted on the right-hand side of the equations, and the new values that are calculated for the unknowns are the second estimated solution. In the second iteration, the second solution is substituted back in the equations to give new values for the unknowns, which are the third estimated solution. The iterations continue in the same manner, and when the method does work, the solutions that are obtained as successive iterations converge toward the actual solution. For a system with n equations, the explicit equations for the *[x$_j$]* unknowns are:

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{j=n} a_{ij}x_j\right), i = 1,2,\ldots,n \qquad (3.14)$$

Condition for convergence

For a system of n equations *[a][x] = [b]*, a sufficient condition for convergence is that in each row of the matrix of coefficients *[a]* the absolute value of the diagonal element is greater than the sum of the absolute values of the off-diagonal elements.

$$|a_{ii}| > \sum_{j=1, j\neq i}^{j=n} |a_{ij}| \qquad (3.15)$$

This condition is sufficient but not necessary for convergence when the iteration method is used. When the condition (3.15) is satisfied, the matrix *[a]* is classified as diagonally dominant, and the iteration process converges toward the solution. The solution, however, might converge even when Eq. (3.15) is not satisfied. Two specific iterative methods for executing the iterations, the Jacobi and Gauss-Seidel methods, are presented next. The difference between the two methods is in the way that the new calculated values of the unknowns are used.

3. 5. 1 Jacobi Iterative Method

In the Jacobi method, an initial (first) value is assumed for each of the unknowns $x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)}$. If no information is available regarding the approximate values of the unknown, the initial value of all the unknowns can be assumed to be zero. The second estimate of the solution $x_1^{(2)}, x_2^{(2)}, \ldots, x_n^{(2)}$ is calculated by substituting the first estimate in the right-hand side ofEqs. (3.14):

$$x_i^{(2)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{j=n} a_{ij}x_j^{(1)}\right), i = 1,2,\dots,n$$

In general, the (k + 1) th estimate of the solution is calculated from the (k) th estimate by:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{j=n} a_{ij}x_j^{(k)}\right), i = 1,2,\dots,n$$

The iterations continue until the differences between the values that are obtained in successive iterations are small. The iterations can be stopped when the absolute value of the estimated relative error of all the unknowns is smaller than some predetermined value:

$$\left|\frac{x_i^{(k+1)} - x_i^{(k)}}{x_i^{(k)}}\right| < \epsilon \ , i = 1,2,\dots,n$$

Example 3.3 Solve the following equations by Jacobi's method.

$$15x + 3y - 2z = 85$$
$$2x + 10y + z = 51$$
$$x - 2y + 8z = 5$$

Solution In the above equations:

$$|15| > |3| + |-2|$$
$$|10| > |2| + |1|$$
$$|8| > |1| + |-2|$$

then Jacobi's method is applicable. We rewrite the given equations as follows:

$$x = \frac{1}{a_1}(d_1 - b_1y - c_1z) \ = \frac{1}{15}(85 - 3y + 2z)$$
$$y = \frac{1}{b_2}(d_2 - a_2x - c_2z) = \frac{1}{10}(51 - 2x - z)$$
$$z = \frac{1}{c_3}(d_3 - a_3x - b_3y) \ = \frac{1}{8}(5 - x + 2y)$$

Let the initial approximations be:

$x^0 = y^0 = z^0 = 0$

Iteration 1:

$$\boxed{x_1} = \frac{d_1}{a_1} = \frac{85}{15} = \frac{17}{3}$$
$$\boxed{y_1} = \frac{d_2}{b_2} = \frac{51}{10}$$
$$\boxed{z_1} = \frac{d_3}{c_3} = \frac{5}{8}$$

Iteration 2:

$$x_2 = \frac{1}{a_1}(d_1 - b_1y_1 - c_1z_1) = \frac{1}{15}\left(85 - 3\times\frac{51}{10} - (-2)\times\frac{5}{8}\right)$$

$$\boxed{x_2} = 4.73$$

$$y_2 = \frac{1}{b_2}(d_2 - a_2x_1 - c_2z_1) = \frac{1}{10}\left(51 - 2\times\frac{17}{3} - 1\times\frac{5}{8}\right)$$

$$\boxed{y_2} = 3.904$$

$$z_2 = \frac{1}{c_3}(d_3 - a_3x_1 - b_3y_1) = \frac{1}{8}\left(5 - 1\times\frac{17}{3} - (-2)\times\frac{51}{10}\right)$$

$$\boxed{z_2} = 1.192$$

Iteration 3:

$$\boxed{x_3} = \frac{1}{15}(85 - 3\times 3.904 + 2\times 1.192) = 5.045$$

$$\boxed{y_3} = \frac{1}{10}(51 - 2\times 4.73 - 1\times 1.192) = 4.035$$

$$\boxed{z_3} = \frac{1}{8}(5 - 1\times 4.173 + 2\times 3.904) = 1.010$$

Iteration 4:

$$\boxed{x_4} = \frac{1}{15}(85 - 3\times 4.035 + 2\times 1.010) = 4.994$$

$$\boxed{y_4} = \frac{1}{10}(51 - 2\times 5.045 - 1\times 1.010) = 3.99$$

$$\boxed{z_4} = \frac{1}{8}(5 - 1\times 5.045 + 2\times 4.035) = 1.003$$

Iteration 5:

$$\boxed{x_5} = \frac{1}{15}(85 - 3\times 3.99 + 2\times 1.003) = 5.002$$

$$\boxed{y_5} = \frac{1}{10}(51 - 2\times 4.994 - 1\times 1.003) = 4.001$$

$$\boxed{z_5} = \frac{1}{8}(5 - 1\times 4.994 + 2\times 3.99) = 0.998$$

Iteration 6:

$$\boxed{x_6} = \frac{1}{15}(85 - 3\times 4.001 + 2\times 0.998) = 5.0$$

$$\boxed{y_6} = \frac{1}{10}(51 - 2\times 5.002 - 1\times 0.998) = 4.0$$

$$\boxed{z_6} = \frac{1}{8}(5 - 1\times 5.002 + 2\times 4.001) = 1.0$$

Iteration 7:

$$\boxed{x_7} = \frac{1}{15}(85 - 3\times 4 + 2\times 1) = 5.0$$

$$\boxed{y_7} = \frac{1}{10}(51 - 2\times 5 - 1\times 1) = 4.0$$

$$\boxed{z_7} = \frac{1}{8}(5 - 1\times 5 + 2\times 4) = 1.0$$

Example 3.4: Use the Jacobi iterative scheme to obtain the solutions of the system of equations correct to three decimal places.

$$x + 2y + z = 0$$
$$3x + y - z = 0$$
$$x - y + 4z = 3$$

Solution

Rearrange the equations in such a way that all the diagonal terms are dominant.

$$3x + y - z = 0$$
$$x + 2y + z = 0$$
$$x - y + 4z = 3$$

Computing for x, y and z we get:

$$x = (z - y)/3$$
$$y = (-x - z)/2$$
$$z = (3 + y - x)/4$$

The iterative equation can be written as:

$$x^{(r+1)} = (z^{(r)} - y^{(r)})/3$$
$$y^{(r+1)} = (-x^{(r)} - z^{(r)})/2$$
$$z^{(r+1)} = (3 - x^{(r)} + y^{(r)})/4$$

The initial vector is not specified in the problem. Hence we choose

$$x^{(0)} = y^{(0)} = z^{(0)} = 1$$

Then, the first iteration gives:

$$x^{(1)} = (z^{(0)} - y^{(0)})/3 \ = (1 - 1)/3 = 0$$
$$y^{(1)} = (-x^{(0)} - z^{(0)})/2 \ = (-1 - 1)/2 = -1.0$$
$$z^{(1)} = (3 - x^{(0)} + y^{(0)})/4 = (3 - 1 + 1)/4 = 0.750$$

similarly, second iteration yields:

$$x^{(2)} = (z^{(1)} - y^{(1)})/3 \ = (0.75 + 1.0)/3 = 0.5833$$
$$y^{(2)} = (-x^{(1)} - z^{(1)})/2 \ = (-0 - 0.75)/2 = -0.3750$$
$$z^{(2)} = (3 - x^{(1)} + y^{(1)})/4 = (3 - 0 - 0)/4 = 0.500$$

Subsequent iterations result in the following:

$x^{(3)} = 0.29167$	$y^{(3)} = -\ 0.34165$	$z^{(3)} = 0.51042$
$x^{(4)} = 0.32986$	$y^{(4)} = -0.40104$	$z^{(4)} = 0.57862$
$x^{(5)} = 0.32595$	$y^{(5)} = -0.45334$	$z^{(5)} = 0.56728$
$x^{(6)} = 0.34021$	$y^{(6)} = -0.44662$	$z^{(6)} = 0.55329$
$x^{(7)} = 0.3333$	$y^{(7)} = -0.44675$	$z^{(7)} = 0.55498$
$x^{(8)} = 0.33391$	$y^{(8)} = -0.44414$	$z^{(8)} = 0.55498$
$x^{(9)} = 0.33304$	$y^{(9)} = -0.44445$	$z^{(9)} = 0.5555$

so to three decimal places the approximate solution:

$$x = 0.333 \ y = -0.444 \ z = 0.555$$

3. 5. 2 Gauss-Seidel Iterative Method

In the Gauss-Seidel method, initial (first) values are assumed for the unknowns x_2, x_3, ..., x_n (all of the unknowns except x_1). If no information is available regarding the approximate value of the unknowns, the initial value of all the unknowns can be assumed to be zero. The first assumed values of the unknowns are substituted in Eq. (3.14) with $i = 1$ to calculate the value of x_1. Next, Eq. (3.14) with $i = 2$ is used for calculating a new value for x_2. This is followed by using Eq. (3.14) with $i = 3$ for calculating a new value for x_3. The process continues until $i = n$, which is the end of the first iteration. Then, the second iteration starts with $i = 1$ where a new value for x_1 is calculated, and so on. In the Gauss-Seidel method, the current values of the unknowns are used for calculating the new value of the next unknown. In other words, as a new value of an unknown is calculated, it is immediately used for the next application of Eq. (3.14). (In the Jacobi method, the values of the unknowns obtained in one iteration are used as a complete set for calculating the new values of the unknowns in the next iteration. The values of the unknowns are not updated in the middle of the iteration.) Applying Eq. (3.14) to the Gauss-Seidel method gives the iteration formula:

$$\left.\begin{array}{c} x_1^{(k+1)} = \frac{1}{a_{11}}\left(b_1 - \sum_{j=1, j\neq i}^{j=n} a_{1j}x_j^{(k)}\right) \\ x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{j=i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{j=n} a_{ij}x_j^{(k)}\right), i = 2, \dots, n-1 \\ x_n^{(k+1)} = \frac{1}{a_{nn}}\left(b_n - \sum_{j=1}^{j=n-1} a_{nj}x_j^{(k+1)}\right) \end{array}\right\} \qquad (3.16)$$

Example 3.5: Solve the following equations by Gauss-Seidal method.

$$8x + 2y - 2z = 8$$
$$x - 8y + 3z = -4$$
$$2x + y + 9z = 12$$

Solution

In the above equations:

$$|8| > |2| + |-2|$$
$$|-8| > |1| + |3|$$
$$|9| > |2| + |1|$$

So, the conditions of convergence are satisfied and we can apply Gauss-Seidal method. Then we rewrite the given equations as follows:

$$x_1 = \frac{1}{a_1}(d_1 - b_1 y^0 - c_1 z^0)$$

$$y_1 = \frac{1}{b_2}(d_2 - a_2 x_1 - c_2 z^0)$$

$$z_1 = \frac{1}{c_3}(d_3 - a_3 x_1 - b_3 y_1)$$

Let the initial approximations be:

$$x0 = y0 = z0 = 0$$

Iteration 1:

$$\boxed{x_1} = \frac{d_1}{a_1} = \frac{8}{8} = 1.0$$

$$\boxed{y_1} = \frac{1}{b_2}(d_2 - a_2 x_1) = \frac{1}{-8}(-4 - 1 \times 1.0) = 0.625$$

$$\boxed{z_1} = \frac{1}{c_3}(d_3 - a_3 x_1 - b_3 y_1) = \frac{1}{9}(12 - 2) = 2 \times 1.0 - 1 \times 0.625) = 1.042$$

Iteration 2:

$$\boxed{x_2} = \frac{1}{a_1}(d_1 - b_1 y_1 - c_1 z_1) = \frac{1}{8}(8 - 2 \times 0.625 - (-2) \times 1.042) = 1.104$$

$$\boxed{y_2} = \frac{1}{b_2}(d_2 - a_2 x_2 - c_2 z_1) = \frac{1}{-8}(-4 - 1 \times 1.104 - 3 \times 1.042) = 1.029$$

$$\boxed{z_2} = \frac{1}{c_3}(d_3 - a_3 x_2 - b_3 y_2) = \frac{1}{9}(12 - 2 \times 1.104 - 1 \times 1.029) = 0.974$$

Iteration 3:

$$\boxed{x_3} = \frac{1}{a_1}(d_1 - b_1 y_2 - c_1 z_2) = \frac{1}{8}(8 - 2 \times 1.029 - (-2) \times 0.974) = 0.986$$

$$\boxed{y_3} = \frac{1}{b_2}(d_2 - a_2 x_3 - c_2 z_2) = \frac{1}{-8}(-4 - 1 \times 0.986 - 3 \times 0.974) = 0.989$$

$$\boxed{z_3} = \frac{1}{c_3}(d_3 - a_3 x_3 - b_3 y_3) = \frac{1}{9}(12 - 2 \times 0.986 - 1 \times 0.989) = 1.004$$

Iteration 4:

$$\boxed{x_4} = \frac{1}{8}(8 - 2 \times 0.989 - (-2) \times 1.004) = 1.004$$

$$\boxed{y_4} = \frac{1}{-8}(-4 - 1 \times 1.004 - 3 \times 1.004) = 1.002$$

$$\boxed{z_4} = \frac{1}{9}(12 - 2 \times 1.004 - 1 \times 1.002) = 0.999$$

Iteration 5:

$$\boxed{x_5} = \frac{1}{8}(8 - 2\times 1.002 - (-2)\times 0.999) = 0.999$$

$$\boxed{y_5} = \frac{1}{-8}(-4 - 1\times 0.999 - 3\times 0.999) = 1.0$$

$$\boxed{z_5} = \frac{1}{9}(12 - 2\times 0.999 - 1\times 1.0) = 1.0$$

Iteration 6:

$$\boxed{x_6} = \frac{1}{8}(8 - 2\times 1 + 2\times 1) = \boxed{1.0}$$

$$\boxed{y_6} = \frac{1}{-8}(-4 - 1\times 1.0 - 3\times 1.0) = \boxed{1.0}$$

$$\boxed{z_6} = \frac{1}{9}(12 - 2\times 1.0 - 1\times 1.0) = \boxed{1.0}$$

Example 3.6: Using the Gauss-Seidal method solve the system of equations correct to three decimal places.

$$x + 2y + z = 0$$
$$3x + y - z = 0$$
$$x - y + 4z = 3$$

Solution

Rearranging the given equations to give dominant diagonal elements, we obtain

$3x + y - z = 0$

$x + 2y + z = 0$

$x - y + 4z = 3$ (E.1)

Equation (E.1) can be rewritten as

$x = (z - y)/3$

$y = -(x + z)/2$

$z = (3 + x + y)/4$ (E.2)

Writing Eq.(E.2) in the form of Gauss-Seidal iterative scheme, we get:

$$x^{(r+1)} = (z^{(r)} - y^{(r)})/3$$
$$y^{(r+1)} = -(x^{(r+1)} - z^{(r)})/2$$
$$z^{(r+1)} = (3 - x^{(r+1)} + y^{(r+1)})/4$$

We start with the initial value

$$x(0) = y(0) = z(0) = 1$$

The iteration scheme gives:

$$x^{(1)} = (z^{(0)} - y^{(0)})/3 = (1 - 1)/3 = 0$$
$$y^{(1)} = (-x^{(1)} - z^{(0)})/2 = (0 - 1)/2 = -0.5$$
$$z^{(1)} = (3 - x^{(1)} + y^{(1)})/4 = (3 \text{ - } 0\text{-}0.5)/4 = 0.625$$

The second iteration gives:

$$x^{(2)} = (z^{(1)} - y^{(1)})/3 = (0.625 + 0.5)/3 = 0.375$$
$$y^{(2)} = (-x^{(2)} - z^{(1)})/2 = (-0.375 - 0.625)/2 = -0.50$$
$$z^{(2)} = (3 - x^{(2)} + y^{(2)})/4 = (3 - 0.375 - 0.5)/4 = 0.53125$$

Subsequent iterations result in:

$$x^{(3)}=0.34375 \qquad y^{(3)}=-0.4375 \qquad z^{(3)}=0.55469$$
$$x^{(4)}=0.33075 \qquad y^{(4)}=-0.44271 \qquad z^{(4)}=0.55664$$
$$x^{(5)}=0.33312 \qquad y^{(5)}=-0.44488 \qquad z^{(5)}=0.5555$$
$$x^{(6)}=0.33346 \qquad y^{(6)}=-0.44448 \qquad z^{(6)}=0.55552$$

Hence, the approximate solution is as follows:

$$x = 0.333,\ y = -0.444,\ z = 0.555$$

3.6 USE OF MATLAB Built IN FUNCTIONS FOR SOLVING A SYSTEM OF LINEAR EQUATIONS

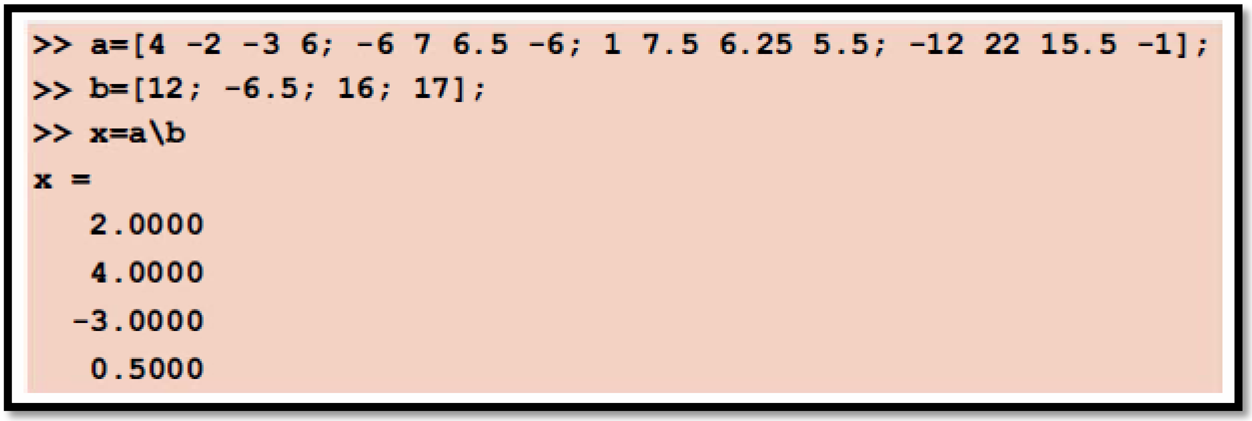
MATLAB has mathematical operations and built-in functions that can be used for solving a system of linear equations and for carrying out other matrix operations that are described in this chapter.

3.6.1 Solving a System of Equations Using MATLAB's Left and Right Division

Left division \ : Left division can be used to solve a system of n equations written in matrix form *[a][x]=[b]*, where *[a]* is the *(n* x *n)* matrix of coefficients, *[x]* is an *(n* x *1)* column vector of the unknowns, and *[b]* is an *(n* x *1)* column vector of constants.

x=a\b

For example, the solution of the system of equations in Example 3-1 is calculated by (Command Window):

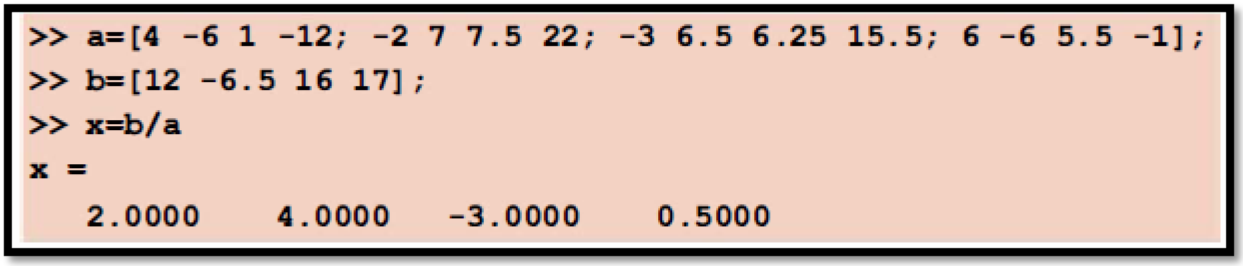


```
>> a=[4 -2 -3 6; -6 7 6.5 -6; 1 7.5 6.25 5.5; -12 22 15.5 -1];
>> b=[12; -6.5; 16; 17];
>> x=a\b
x =
   2.0000
   4.0000
  -3.0000
   0.5000
```

Right division / : Right division is used to solve a system of n equations written in matrix form *[x][a] = [b]*, where *[a]* is the *(n* x *n)* matrix of coefficients, *[x]* is a *(1* x *n)* row vector of the unknowns, and *[b]* is a *(1* x *n)* row vector of constants.

x=b/a

For example, the solution of the system of equations in Example 3-1 is calculated by (Command Window):

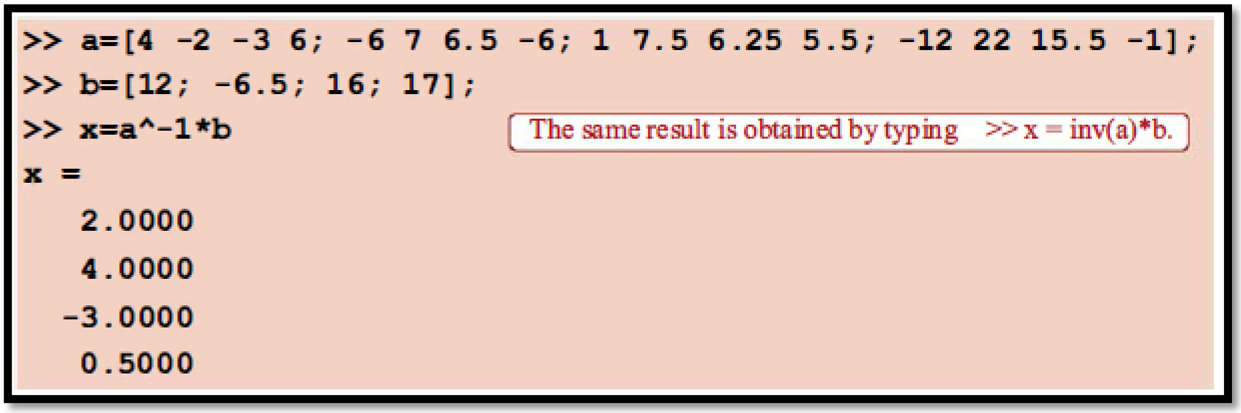


```
>> a=[4 -6 1 -12; -2 7 7.5 22; -3 6.5 6.25 15.5; 6 -6 5.5 -1];
>> b=[12 -6.5 16 17];
>> x=b/a
x =
   2.0000    4.0000   -3.0000    0.5000
```

Notice that the matrix *[a]* used in the right division calculation is the transpose of the matrix used in the left division calculation.

3.6.2 Solving a System of Equations Using MATLAB Inverse Operation

In MATLAB, the inverse of a matrix *[a]* can be calculated either by raising the matrix to the power of -1 or by using the **inv(*a*)** function. Once the inverse is calculated, the solution is obtained by multiplying the vector *[b]* by the inverse. This is demonstrated for the solution of the system in Example 4-1.



```
>> a=[4 -2 -3 6; -6 7 6.5 -6; 1 7.5 6.25 5.5; -12 22 15.5 -1];
>> b=[12; -6.5; 16; 17];
>> x=a^-1*b
x =
   2.0000
   4.0000
  -3.0000
   0.5000
```

The same result is obtained by typing >> x = inv(a)*b.

3.7 Problems

1. Solve the following system of equations using the Gauss elimination method:

$$2x_1 + x_2 - x_3 = 1$$
$$x_1 + 2x_2 + x_3 = 8$$
$$-x_1 + x_2 - x_3 = -5$$

2. Consider the following system of two linear equations:

$$0.0003x_1 + l.566x_2 = l.569$$
$$0.3454x_1 - 2.436x_2 = 1.018$$

(a) Solve the system with the Gauss elimination method using rounding with four significant figures.
(b) Switch the order of the equations, and solve the system with the Gauss elimination method using rounding with four significant figures.

Check the answers by substituting the solution back in the equations.

3. Solve the following set of simultaneous linear equations using the Jacobi's method.
   a. $2x - y + 5z = 15$
      $2x + y + z = 7$
      $x + 3y + z = 10$
   b. $20x + y - 2z = 17$
      $3x + 20y - z = -18$
      $2x - 3y + 20z = 25$

   c. $5x + 2y + z = 12$
      $x + 4y + 2z = 15$
      $x + 2y + 5z = 20$
4. Solve the following system of simultaneous linear equations using the Gauss-Seidal method.
   a. $4x - 3y + 5z = 34$
      $2x - y - z = 6$
      $z + y + 4z = 15$

   b. $2x - y + 5z = 15$
      $2x + y + z = 7$
      $x + 3y + z = 10$

c. $15x + 3y - 2z = 85$
$2x + 10y + z = 51$
$x - 2y + 8z = 5$

5. Determine the LU decomposition of the matrix $a = \begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 1 \\ 6 & -2 & 2 \end{bmatrix}$ using the Gauss elimination procedure.
6. Carry out the first three iterations of the solution of the following system of equations using the Gauss-Seidel iterative method. For the first guess of the solution, take the value of all the unknowns to be zero.

$$8x_1 + 2x_2 + 3x_3 = 51$$
$$2x_1+5x_2+x_3 = 23$$
$$-3x_1+x_2+6x_3 = 20$$

Chapter4: Curve Fitting and Interpolation

Limits processes are the basis of calculus. For example, the derivative

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

is the limit of the difference quotient where both the numerator and the denominator go to zero. A Taylor series illustrates another type of limit process. In this case an infinite number of terms is added together by taking the limit of certain partial sums. An important application is their use to represent the elementary functions: sin(x), cos(x), e^x, ln(x),etc. Table(4.1) gives several of the common Taylor series expansions.

Table(4.1): Taylor Series Expansions for Some Common Function

$Sin(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\ldots$	*for all x*
$Cos(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots$	*for all x*
$e^x=1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots$	*for all x*
$\ln(1+x)=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\ldots$	$-1\le x\le 1$
$\tan^{-1}(x)=x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\ldots$	$-1\le x\le 1$
$(1+x)^p=1+px+\frac{p(p-1)}{2!}x^2+\frac{p(p-1)(p-2)}{3!}x^3+\cdots$	$for\ \lvert x\rvert<1$

We want to learn how a finite sum can be used to obtain a good approximations to an infinite sum. For illustration we shall use the exponential series in table(4.1) to compute the number $e=e^1$. Here we choose $x=1$ and use the series:

$$e^1=1+\frac{1}{1!}+\frac{1^2}{2!}+\cdots+\frac{1^k}{k!}+\cdots$$

Table(4.2): Partial Sums S_n Used to Determine e

n	$s_n = 1 + \frac{1}{1!} + \frac{1^2}{2!} + \cdots + \frac{1^n}{n!} + \cdots$
0	1
1	2
2	2.5
3	2.666 666 666
4	2.708 333 333
5	2.716 666 666
6	2.718 055 555
7	2.718 253 968
8	2.718 278 769
9	2.718 281 525
10	2.718 281 180
11	2.718 281 826
12	2.718 281 182
13	2.718 281828
14	2.718 281 828
15	2.718 281 828

<u>Theorem(4.1)</u>: (Taylor Polynomial Approximation)

Assume that $f\epsilon C^{N+1}[a,b]$and $x_0 \in [a,b]$is a fixed value. If $x \in [a,b]$,then:

$$f(x)=P_N(x)+E_N(x) \qquad (4.1)$$

where $P_N(x)$ is a polynomial that can be used to approximate $f(x)$:

$$f(x) \approx P_N(x) = \sum_{k=0}^{N} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k \qquad (4.2)$$

The error term $E_N(x)$ has the form:

$$E_N(x)=\frac{f^{(N+1)}(c)}{(N+1)!}(x-x_0)^{N+1} \qquad (4.3)$$

for some value c=c(x) that lies between x and x_0.

<u>Example(4.1)</u>: Show why 15 terms are all that are needed to obtain the 13-digit approximation

e=2.718 281 828 459 in table(4.2).

$$P_{15}(x)=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots+\frac{x^{15}}{15!} \qquad (4.4)$$

setting $x=1$ in (4.4) gives the partial sum $S_{15}=P_{15}(1)$

$$E_{15}(x)=\frac{f^{(16)}(c)x^{16}}{16!}$$

Since $x_0=0$ and $x=1$ then $0<c<1$

which implies that $e^c<e^1$

$$|E_{15}(x)|=\left|\frac{f^{(16)}(c)x^{16}}{16!}\right| \le \frac{e^c}{16!} < \frac{3}{16!} < 1.433\,844 \times 10^{-13}$$

<u>Exercises</u>:

1. Let f(x)=sin(x) and apply theorem(4.1)
   a. Use x_0=0 and find $P_5(x)$, $P_7(x)$, and $P_9(x)$.
   b. Show that if |x|≤1 then the approximation

      $$\sin(x) \approx x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\frac{x^9}{9!}$$

      has the error bound $|E_9(x)| < \frac{1}{10!} \le 2.755\,74 \times 10^{-7}$.
   c. Use $x_0=\frac{\pi}{4}$ and find $P_5(x)$, which involves powers of $(x-\frac{\pi}{4})$.
2. (a) Find a Taylor polynomial of degree N=5 for $f(x)=\frac{1}{1+x}$ expanded about x_0=0.

   (b) Find the error term $E_5(x)$ for the polynomial part(a).

4.1 Introduction to Interpolation

We saw how a Taylor polynomial can be used to approximate the function f(x). The information needed to construct the Taylor polynomial is the value of f and its derivatives at x_0. A short coming is that the higher-order derivatives must be known, and often they are either not available or they are hard to compute.

Suppose that the function y=f(x) is known at N+1 points $(x_0,y_0),(x_1,y_1),\ldots,(x_N,y_N)$, where the values x_k are spread out over the interval [a,b] and satisfy. In the construction, only the numerical values x_k and y_k are needed.

$$a \le x_0 < x_1 < \cdots < x_N \le b \quad \text{and} \quad y_k=f(x_k)$$

A polynomial p(x) of degree N will be constructed that passes through these N+1 points.

4.2 Lagrange Approximation

Interpolation means to estimate a missing function value by taking a weighted average of known function values at neighboring points. Linear interpolation uses a line segment that passes through two points. The slope between (x_0,y_0) and (x_1,y_1) is $m=(y_1-y_0)/(x_1-x_0)$, and the point-slope formula for the line $y=m(x-x_0)+y_0$ can be rearranged as:

$$y=P(x)=y_0+(y_1-y_0)\frac{x-x_0}{x_1-x_0} \qquad (4.5)$$

when formula (4.5) is expanded, the result is a polynomial of degree≤1. Evaluation of P(x) at x_0 and x_1, respectively:

$$P(x_0)=y_0+(y_1-y_0)(0)=y_0$$

$$P(x_1)=y_0+(y_1-y_0)(1)=y_1$$

The French mathematician Joseph Louis Lagrange used a slightly different method to find this polynomial. He noticed that it could be written as:

$$y=P_1(x)=y_0\frac{x-x_1}{x_0-x_1}+y_1\frac{x-x_0}{x_1-x_0} \qquad (4.6)$$

Each term on the right side of (4.6) involves a linear factor; hence the sum is a polynomial of degree≤1. The quotient in (4.6) are denoted by

$L_{1,0}(x) = \frac{x-x_1}{x_0-x_1}$ and $L_{1,1}(x) = \frac{x-x_0}{x_1-x_0}$ (4.7)

Computation reveals that $L_{1,0}(x_0) = 1$, $L_{1,0}(x_1) = 0$, $L_{1,1}(x_0) = 0$, and $L_{1,1}(x_1) = 1$ so that the polynomial $P_1(x)$ in (4.6) also passes through the two given pints. The terms $L_{1,0}(x)$ and $L_{1,1}(x)$ are called ***Lagrange coefficient polynomials***. Using this notation, (4.6) can be written in summation form:

$P_1(x)=\sum_{k=0}^{1} y_k L_{1,k}$ (4.8)

Example(4.2): Consider the graph y=f(x)=cos(x) over [0,1.2].

a. Use the nodes $x_0=0$, and $x_1=1.2$ to construct a linear interpolation polynomial $P_1(x)$.
b. Use the nodes $x_0=0.2$, and $x_1=1$ to construct a linear interpolation polynomial$Q_1(x)$.

Using (4.6) with the abscissas $x_0=0$, and $x_1=1.2$ and the ordinates $y_0=\cos(0)=1$ and $y_1=\cos(1.2)=0.362\ 358$

$P_1(x)=1\frac{x-1.2}{0-1.2} + 0.362\ 358\,\frac{x-0}{1.2-0}$

$=-0.833\ 333(x-1.2)+0.301\ 965(x-0)$

When the nodes $x_0=0.2$, and $x_1=1$ with $y_0=\cos(0.2)=0.980\ 067$ and $y_1=\cos(1)=0.540\ 302$ are used, the results is:

$Q_1(x)=0.980\ 067\,\frac{x-1}{0.2-1} + 0.540\ 302\,\frac{x-0.2}{1-0.2}$

$=-1.225\ 083(x-1)+0.675\ 378(x-0.2)$

The generalization of(1.8) is the construction of a polynomial $P_N(x)$ of degree at most N that passes through the N+1 points $(x_0,y_0),(x_1,y_1),\ldots,(x_N,y_N)$ and has the form:

$P_N(x)=\sum_{k=0}^{N} y_k L_{N,k}$ (4.9)

where $L_{N,k}$ is the Lagrange coefficient polynomial based on these nodes

$L_{N,k}=\frac{(x-x_0)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_N)}{(x_k-x_0)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_N)}$ (4.10)

Example(4.3): Consider y=f(x)=cos(x) over [0,1.2]

a. Use the three nodes $x_0=0, x_1=0.6$ and $x_2=1.2$ to construct a quadratic interpolation polynomial $P_2(x)$.
b. Use the four nodes $x_0=0, x_1=0.4, x_2=0.8$ and $x_3=1.2$ to construct a cubic interpolation polynomial $P_3(x)$.

a.

x_i	0	0.6	1.2
$y_i=\cos(x_i)$	1	0.825 336	0.362 358

$$P_2(x)=1\frac{(x-0.6)(x-1.2)}{(0-0.6)(0-1.2)}+0.825\ 336\frac{(x-0)(x-1.2)}{(0.6-0)(0.6-1.2)}$$

$$+0.362\ 358\frac{(x-0)(x-0.6)}{(1.2-0)(1.2-0.6)}$$

$$=1.388\ 889(x-0.6)(x-1.2)-2.292\ 599x(x-1.2)+0.503275x(x-0.6)$$

b.

x_i	0	0.4	0.8	1.2
$y_i=\cos(x_i)$	1	0.921 061	0.696 707	0.362 358

$$P_3(x)=1\frac{(x-0.4)(x-0.8)(x-1.2)}{(0-0.4)(0-0.8)(0-1.2)}+0.921\ 061\frac{(x-0)(x-0.8)(x-1.2)}{(0.4-0)(0.4-0.8)(0.4-1.2)}+0.696\ 707\frac{(x-0)(x-0.4)(x-1.2)}{(0.8-0)(0.8-0.4)(0.8-1.2)}$$

$$+0.362\ 358\frac{(x-0)(x-0.4)(x-0.8)}{(1.2-0)(1.2-0.4)(1.2-0.8)}$$

$$= -2.604\ 167(x-0.4)(x-0.8)(x-1.2)+7.195\ 789x(x-0.8)(x-1.2)$$

$$-5.443\ 021x(x-0.4)(x-1.2)+0.943\ 641x(x-0.4)(x-0.8)$$

Exercises: Find Lagrange polynomials that approximate $f(x)=x^3$.

a. Find the linear interpolation polynomial $P_1(x)$ using the nodes $x_0=-1$ and $x_1=0$
b. Find the quadratic interpolation polynomial $P_2(x)$ using $x_0=-1$, $x_1=0$ and $x_2=1$.

c. Find the cubic interpolation polynomial $P_3(x)$ using x_0=-1, x_1=0 x_2=1 and x_3=2.

d. Find the linear interpolation polynomial $P_1(x)$ using the nodes x_0=1 and x_1=2.

4.2.1 Error Terms and Error Bounds:

Theorem(4.2): (Lagrange Polynomial Approximation)

Assume that $f \in C^{N+1}[a,b]$ and that $x_0, x_1, \dots, x_N \in [a,b]$ are N+1 nodes. If $x \in [a,b]$, then :

$$f(x)=P_N(x)+E_N(x) \qquad (4.11)$$

where $P_N(x)$ is a polynomial that can be used to approximate $f(x)$

$$f(x)=P_N(x)=\sum_{k=0}^{N} f(x_k)L_{N,k} \qquad (4.12)$$

The error term $E_N(x)$ has the form:

$$E_N(X)=\frac{(x-x_0)(x-x_1)\dots(x-x_N)f^{(N+1)}(c)}{(N+1)!} \qquad (4.13)$$

for some value c=c(x) that lies in the interval [a,b].

Theorem (4.3): (Error Bounds for Lagrange Interpolation, Equally Spaced Nodes)

Assume that $f(x)$ is defined on [a,b], which contains equally spaced nodes $x_k=x_0+hk$. Additionally, assume that $f(x)$ and derivatives of f (x), up to order N+1, are continuous and bounded on the special subintervals $[x_0,x_1]$, $[x_0,x_2]$, and $[x_0,x_3]$, respectively; that is:

$$\left|f^{(N+1)}(x)\right| \le M_{N+1} \quad for\ x_0 \le x \le x_N \qquad (4.14)$$

for N=1,2,3. The error terms (4.13) corresponding to the cases N=1,2, and 3 have the following useful bounds on their magnitude:

$$|E_1(x)| \le \frac{h^2M_2}{8} \quad valid\ for\ x \in [x_0,x_1], \qquad (4.15)$$

$$|E_2(x)| \le \frac{h^3M_3}{9\sqrt{3}} \quad valid\ for\ x \in [x_0,x_2], \qquad (4.16)$$

$$|E_3(x)| \le \frac{h^4M_4}{24} \quad valid\ for\ x \in [x_0,x_3], \qquad (4.17)$$

Example(4.4): Consider y=f(x)=cos(x) over [0,1.2]. Use formulas (4.15) through (4.17) and determine the error bounds for the Lagrange polynomial constructed in examples (4.2) and(4.3).

First, determine the bounds M₂, M₃, and M₄ for the derivatives $\left|f^{(2)}(x)\right|, \left|f^{(3)}(x)\right|\ and\ \left|f^{(4)}(x)\right|$, respectively, taken over the interval [0,1.2]:

$$\left|f^{(2)}(x)\right| = |-\cos(x)| \le |-\cos(0)| = 1 = M_2$$

$$\left|f^{(3)}(x)\right| = |\sin(x)| \le |\sin(1.2)| = 0.932\ 039 = M_3$$

$$\left|f^{(4)}(x)\right| = |\cos(x)| \le |\cos(0)| = 1 = M_4$$

For $P_1(x)$ the spacing of the nodes is h=1.2, and its error bound is:

$$|E_1(x)| \le \frac{h^2M_2}{8} \le \frac{(1.2)^2(1)}{8} = 0.180$$

For $P_2(x)$ the spacing of the nodes is h=0.6, and its error bound is:

$$|E_2(x)| \le \frac{h^3M_3}{9\sqrt{3}} \le \frac{(0.6)^3(0.932\ 039)}{9\sqrt{3}} = 0.012\ 915$$

For $P_3(x)$ the spacing of the nodes is h=0.4, and its error bound is:

$$|E_3(x)| \le \frac{h^4M_4}{24} \le \frac{(0.4)^4(1)}{24} = 0.001\ 067$$

Example(4.5): For the data below, obtain the quadratic polynomial and use to estimate f(0.5).

x	1	-1	2
f(x)	0	-2	3

The quadratic Lagrange polynomial are

$$P_2(x)=(0)\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + (-2)\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + 3\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

$$=(0)\frac{(x-(-1))(x-2)}{(1-(-1))(1-2)} + (-2)\frac{(x-1)(x-2)}{(-1-1)(-1-2)} + 3\frac{(x-1)(x-(-1))}{(2-1)(2-(-1))} \quad = \frac{2x^2+3x-5}{3}$$

hence $P_2(0.5)$=-1.

Exercises:

1. Consider the Lagrange coefficient polynomial $L_{2,k}(x)$ that are used for quadratic interpolation at the nodes x_0,x_1, and x_2. Define $g(x)=L_{2,0}(x)+L_{2,1}(x)+L_{2,2}(x)-1$.
   a. Show that g is a polynomial of degree ≤2.
   b. Show that $g(x_k)=0$ for k=0,1,2.
2. Consider the function f(x)=sin(x) on the interval [0,1]. Use theorem(4.3) to determine the step size h so that:
   a. linear Lagrange interpolation has an accuracy of 10^{-6}.
   b. quadratic Lagrange interpolation has an accuracy of 10^{-6}.
   c. cubic Lagrange interpolation has an accuracy of 10^{-6}.

4.3 Divided Difference Interpolation

The Lagrange interpolation polynomial is useful for analysis, but is not the ideal formula for evaluating the polynomial. Here the groundwork is laid for the development of efficient form of the unique interpolating polynomial P_n.

a. by simplifying the construction.
b. by reducing effort required to evaluate the polynomial.

Definition(4.1):

Define $$f[x_i,x_{i+1}]=\frac{f(x_{i+1})-f(x_i)}{x_{i+1}-x_i} \qquad (4.18)$$

is the ***first-order divided difference of f* at *x=x$_i$***

and $$f[x_i,x_{i+1},x_{i+2}]=\frac{f[x_{i+1},x_{i+2}]-f[x_i,x_{i+1}]}{x_{i+2}-x_i} \qquad (4.19)$$

is the ***second-divided difference of f at x=x$_i$.***

and the ***recursive rule for constructing k-order divided differences is***

$$f[x_i,x_{i+1},\dots,x_{i+k}]=\frac{f[x_{i+1},\dots,x_{i+k}]-f[x_i,\dots,x_{i+k-1}]}{x_{i+k}-x_i} \qquad (4.20)$$

and is used to construct the divided differences in table (4.3)

Table(4.3): Divided Differences Table

x_i	$f(x_i)$	$f[x_i,x_{i+1}]$	$f[x_i,x_{i+1},x_{i+2}]$	$f[x_i,x_{i+1},x_{i+2},x_{i+3}]$
x_0	f_0			
		$f[x_0,x_1]$		
x_1	f_1		$f[x_0,x_1,x_2]$	
				$f[x_0,x_1,x_2,x_3]$
		$f[x_1,x_2]$		
x_2	f_2		$f[x_1,x_2,x_3]$	
		$f[x_2,x_3]$		
x_3	f_3			

Theorem(4.4): (Newton Polynomial)

Suppose that $x_0,x_1,\ldots,x_N$ are N+1 distinct numbers in [a,b]. There exists a unique polynomial $P_N(x)$ of degree at most N with the property that:

$$f(x_j)=P_N(x_j) \quad \text{for } j=0,1,\ldots,N$$

The Newton form of this polynomial is:

$$P_N(x)=a_0+a_1(x-x_0)+\ldots+a_N(x-x_0)(x-x_1)\ldots(x-x_{N-1}) \qquad (4.21)$$

where $a_k=f[x_0,x_1,\ldots,x_k]$, for $k=0,1,\ldots,N$.

Example(4.6): Repeating example(4.5) using the polynomial form (4.21) requires a divided difference table.

x_i	f_i	$f[x_i,x_{i+1}]$	$f[x_i,x_{i+1},x_{i+2}]$
1	0		
		1	
-1	-2		$\frac{2}{3}$
		$\frac{5}{3}$	
2	3		

and Newton polynomial is:

$P_2(x)=f[x_0]+f[x_0,x_1](x-x_0)+ f[x_0,x_1,x_1](x-x_0)(x-x_1)$

$=0+(1)(x-1)+(\frac{2}{3})(x-1)(x-(-1))=\frac{2x^2}{3}+x-\frac{5}{3}$

Corollary(4.1): (Newton Approximation)

Assume that $P_N(x)$ is the Newton polynomial given in theorem(4.4) and is used to approximate the function f(x), that is,

$$f(x)=P_N(x)+E_N(x) \qquad (4.22)$$

If f∈ $C^{N+1}[a,b]$, then for each x∈ $[a,b]$there corresponds a number c=c(x) in (a,b), so that the error term has the form

$$E_N(x)=\frac{(x-x_0)(x-x_1)\ldots(x-x_N)f^{(N+1)}(c)}{(N+1)!} \qquad (4.23)$$

Exercises:

1. Compute the divided-difference table for the tabulated function.

x_i	4	5	6	7	8
y_i	2	2.236 07	2.449 49	2.645 75	2.828 43

2. Evaluate the Newton polynomial and find f(3)

x_i	-2	0	1	2	5
$f(x_i)$	-15	1	-3	-7	41

4.4 Equispaced Interpolation:

4.4.1 Difference Operator and Difference Tables:

Differences are similar to divided differences but work with equispaced data. The ***forward difference operator*** Δ is defined by:

$$\Delta^0 f(x)=f(x) \qquad (4.24)$$

$$\Delta f(x)= \Delta^{1}f(x)=f(x+h)-f(x) \qquad (4.25)$$

$$\Delta^{k}f(x)= \Delta(\Delta^{k-1}f(x))= \Delta^{k-1}(\Delta f(x))$$

$$= \Delta^{k-1}f(x+h)- \Delta^{k-1}f(x) \qquad (4.26)$$

The Δ^{k} are conveniently displayed in a difference table(4.4)

Table(4.4): A Table of Forward Differences

x	**f(x)**	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$
$\mathbf{x_0}$	$\mathbf{f(x_0)}$			
		Δf_0		
$\mathbf{x_1}$	$\mathbf{f(x_1)}$		$\Delta^2 f_0$	
		Δf_1		$\Delta^3 f_0$
$\mathbf{x_2}$	$\mathbf{f(x_2)}$		$\Delta^2 f_1$	
		Δf_2		
$\mathbf{x_3}$	$\mathbf{f(x_3)}$			

Example(4.7): The polynomial $P_3(x)=x^3-6x^2+11x-3$ gives rise to the following difference table at x=2, 4, 6, 8, 10.

x	$P_3(x)$	$\Delta P_3(x)$	$\Delta^2 P_3(x)$	$\Delta^3 P_3(x)$
2	3			
		6		
4	9		48	
		54		48
6	63		96	
		150		48
8	213		144	
		294		
10	507			

4.4.2 Backward Difference Operator ∇:

Define $\nabla^0 f(x)=f(x)$ (4.27)

$\nabla f(x)=\nabla^1 f(x)=f(x)-f(x-h)$ (4.28)

$\nabla^k f(x)=\nabla^{k-1} f(x)-\nabla^{k-1} f(x-h)\ ,\quad k\geq 1$ (4.29)

Table(4.5): A Table of Backward Differences

x	f(x)	$\nabla f(x)$	$\nabla^2 f(x)$	$\nabla^3 f(x)$
x_0	y_0			
		∇f_1		
x_1	y_1		$\nabla^2 f_2$	
		∇f_2		$\nabla^3 f_3$
x_2	y_2		$\nabla^2 f_3$	
		∇f_3		
x_3	y_3			

4.4.3 Shift Operator: E

$E^0 f(x)=f(x)$ (4.30)

$Ef(x)=E^1 f(x)=f(x+h)$ (4.31)

$E^{-1} f(x)=f(x-h)$ (4.32)

$E^k f(x)=f(x+kh)=E(E^{k-1} f(x)),\ k=\pm 1, \pm 2,\ldots$ (4.33)

E shifts the data point a number of intervals to the left or right.

There are many relationships between the three difference operators, of which two will be useful for the ensuing discussion:

$$\Delta f(x)=f(x+h)-f(x)=Ef(x)-f(x)=(E-1)f(x)$$

$$\rightarrow \Delta\equiv E-1\ \ ,E\equiv 1+\Delta \qquad (4.34)$$

and $\nabla f(x)=f(x)-f(x-h)=f(x)-E^{-1}f(x)=(1-E^{-1})f(x)$

$$\rightarrow \nabla \equiv 1 - E^{-1} \quad , E \equiv (1 - \nabla)^{-1} \qquad (4.35)$$

4.4.4 Forward Difference Polynomial:

Assume that the nodes $x_0, x_1, \ldots, x_n$ are in ascending order and may be described by an index j and an interval h,

$$x_j = x_0 + jh, \quad j=0,1,\ldots,n \qquad (4.36)$$

j is the number of intervals between the data point x_j and the origin x_0. For a real number t,

$$x = x_0 + th, \; 0 \leq t \leq n \qquad (4.37)$$

If $t \in \{0,1,\ldots,n\}$, x corresponds to a data point. Otherwise x corresponds to a point lying between two adjacent data points.

$$f(x) = f(x_0+th) = E^t f(x_0) = (1+\Delta)^t f(x_0)$$

$$= \left[1 + t\Delta + \frac{t(t-1)}{2!}\Delta^2 + \frac{t(t-1)(t-2)}{3!}\Delta^3 + \cdots\right] f(x_0)$$

then $P_n(x) = f_0 + t\Delta f_0 + \frac{t(t-1)}{2!}\Delta^2 f_0 + \cdots + \frac{t(t-1)(t-2)\ldots(t-n+1)}{n!}\Delta^n f_0$ (4.38)

which is the ***Newton-Gregory forward difference polynomial***.

Example(4.8): Construct a difference table for the function f where f(0.5)=1, f(0.6)=2 and f(0.7)=5, and use quadratic interpolation to estimate f(0.53).

The difference table is:

x	f(x)	Δf(x)	Δ^2f(x)
0.5	$\underline{1}$		
		$\underline{1}$	
0.6	2		$\underline{2}$
		3	
0.7	5		

the quadratic polynomial $P_2(x) = f_0 + t\Delta f_0 + \frac{1}{2}t(t-1)\Delta^2 f_0$

at x=0.53 , $t = \frac{x-x_0}{h}$, h=0.1, we choose $x_0=0.5$

→$t=\frac{0.53-0.5}{0.1}$=0.3

and f(0.53)≈P_2(0.53)=1+0.3(1)+(0.3)(0.3-1)(2)/2!=1+0.3-0.105=1.195

An alternative form of P_n uses the backward difference operator ∇

$$P_n(x)=P_n(x_0+th)=f_0+t\nabla f_0+\frac{t(t+1)}{2!}\nabla^2 f_0+\cdots+\frac{t(t+1)\ldots(t+n-1)}{n!}\nabla^n f_0 \qquad (4.39)$$

Example(4.9): Repeat example(4.8) using the backward formula(4.39) to find f(0.63).

The difference table is identical to that of example(4.8)

x	f(x)	**∇**f(x)	∇^2f(x)
0.5	1		
		1	
0.6	2		$\underline{\underline{2}}$
		$\underline{\underline{3}}$	
0.7	$\underline{\underline{5}}$		

The quadratic polynomial $P_2(x_2+th)=f_2+t\nabla f_2+\frac{1}{2}t(t+1)\,\nabla^2 f_2$

since $t=\frac{x-x_2}{h}=\frac{0.63-0.7}{0.1}$=-0.7 and f(0.63)≈$P_2$(0.63)=5-3*0.7+$\frac{1}{2}$(-0.7)(-0.7+1)(2)=2.69

Exercises:

1. Construct a difference table for the data

x	0	0.2	0.4	0.6	0.8	1
f(x)	0.55	0.82	1.15	1.54	1.99	2.5

and use to find f(0.23) and f(0.995).

4.5 Curve Fitting

4.5.1 Least Squares Approximation:

Let Y_i represent an experimental value, and let y_i be a value from the equation $y_i=ax_i+b$ where x_i is a particular value of the variable assumed free of error. We wish to determine the best values for a and b so that the y's predict the function values that correspond to x-values. Let $e_i=Y_i-y_i$. The least-squares criterion requires that: S=$e_1^2+e_2^2+\cdots+e_N^2$=$\sum_{i=1}^{N} e_i^2=\sum_{i=1}^{N}(Y_i-ax_i-b)^2$ be a minimum. N is the number of x,Y-pairs. We reach the minimum by proper choice of the parameters a and b, so they are the " variables" of the problem. At a minimum for S, the two partial derivatives $^{\partial S}/_{\partial a}$ *and* $^{\partial S}/_{\partial b}$ will be both zero, that is:

$$\frac{\partial S}{\partial a}=0=\sum_{i=1}^{N}2(Y_i-ax_i-b)(-x_i),$$

$$\frac{\partial S}{\partial b}=0=\sum_{i=1}^{N}2(Y_i-ax_i-b)(-1),$$

Dividing each of these equations by -2 and expanding the summation, we get:

$$\left.\begin{array}{r} a\sum x_i^2+b\sum x_i=\sum x_i y_i \\ a\sum x_i+bN=\sum Y_i \end{array}\right\} \qquad (4.40)$$

All the summations in(4.40) are from i=1 to i=N. Solving these equations gives

$$a=\frac{N\sum x_i y_i-\sum x_i\sum y_i}{N\sum x_i^2-(\sum x_i)^2} \qquad (4.41)$$

$$b=\frac{\sum y_i\sum x_i^2-\sum x_i\sum x_i y_i}{N\sum x_i^2-(\sum x_i)^2} \qquad (4.42)$$

<u>Example(4.10):</u> Find the least-squares line for the data point given in the following table:

x	-1	0	1	2	3	4	5	6
y	10	9	7	5	4	3	0	-1

N=8, $\sum x_i$=20, $\sum x_i^2$=92, $\sum y_i$=37, $\sum x_i y_i$=25

from equations(4.41) and (4.42), we get:

a=-1.6071429, b=8.6428571

and y=-1.6071429x+8.6428571

4.5.2 The Power Fit y=Ax^M

Some situations involve f(x)=Ax^M, where M is a Known constant. In this cases there is only one parameter A to be determined.

Theorem(4.5): (Power Fit)

Suppose that {(x_k,y_k)}, k=1,…,N are N points, where the abscissas are distinct. The coefficient A of the least-squares power curve y=Ax^M is given by

$$A={(\sum_{k=1}^{N} x_k^M y_k)}\Big/{(\sum_{k=1}^{N} x_k^{2M})} \qquad (4.43)$$

Example(4.11): Find the constant g in the relation d=$\frac{1}{2}gt^2$ using the following table:

t	0.2	0.4	0.6	0.8	1.0
d	0.196	0.785	1.7665	3.1405	4.9075

Here M=2, N=5, $\sum d_k t_k^2$=7.6868 , $\sum t_k^4$=1.5664

and the coefficient A=7.6868/1.5664=4.9073, so we get g=2A=9.7146.

4.5.3 Data Linearization Method for y=Ce^{Ax}:

Suppose that we are given points (x_1,y_1),…,(x_N,y_N) and want to fit an exponential curve of the form

$$y=Ce^{Ax} \qquad (4.44)$$

The first step is to take the logarithm of both sides:

$$\ln(y)=Ax+\ln(C) \qquad (4.45)$$

Then introduce the change of variables:

$$Y=\ln(y),\ X=x\ ,\ \text{and}\ B=\ln(C) \qquad (4.46)$$

This results in a linear relation between the new variables X and Y

$$Y=AX+B \qquad (4.47)$$

Example(4.12): Use the data linearization method and find the exponential fit y=Ce^{Ax} for the five points (0,1.5), (1,2.5), (2,3.5), (3,5), and (4,7.5).

x	0	1	2	3	4
y	1.5	2.5	3.5	5	7.5

$\sum x_i = 10$, $\sum Y_i = \sum lny_i = 6.19886$, $\sum x_i^2 = 30$,$\sum x_i lnY_i = \sum x_i lny_i = 16.309743$ and N=5

therefore we have a=0.3912023, b=0.457367

then C is obtained with the calculation C=e$^{0.457367}$=1.57991

and y=1.57991e$^{0.3912023x}$

Exercises:

1. Find the least-squares line for the data

x	-6	-2	0	2	6
y	7	5	3	2	0

2. Find the power fits y=Ax^2 and y=Bx^3 for the following data:

x	0.5	0.8	1.1	1.8	4
y	7.1	4.4	3.2	1.9	0.9

3. For the given data find the least-squares curve f(x)=Ce^{Ax}

x	-1	0	1	2	3
y	6.62	3.94	2.17	1.35	0.89

Chapter1: Numerical Differentiation

1.1 Finite Difference Approximation of the Derivative

In finite difference approximations of the derivative, values of the function at different points in the neighborhood of the point $x=a$ are used for estimating the slope. It should be remembered that the function that is being differentiated is prescribed by a set of discrete points. Various finite difference approximation formulas exist. Three such formulas, where the derivative is calculated from the values of two points, are presented in this section.

1.1.1Forward, Backward, and Central Difference Formulas for the First Derivative

The forward, backward, and central finite difference formulas are the simplest finite difference approximations of the derivative. In these approximations, illustrated in Fig. 1-1, the derivative at point x_i is calculated from the values of two points. The derivative is estimated as the value of the slope of the line that connects the two points.

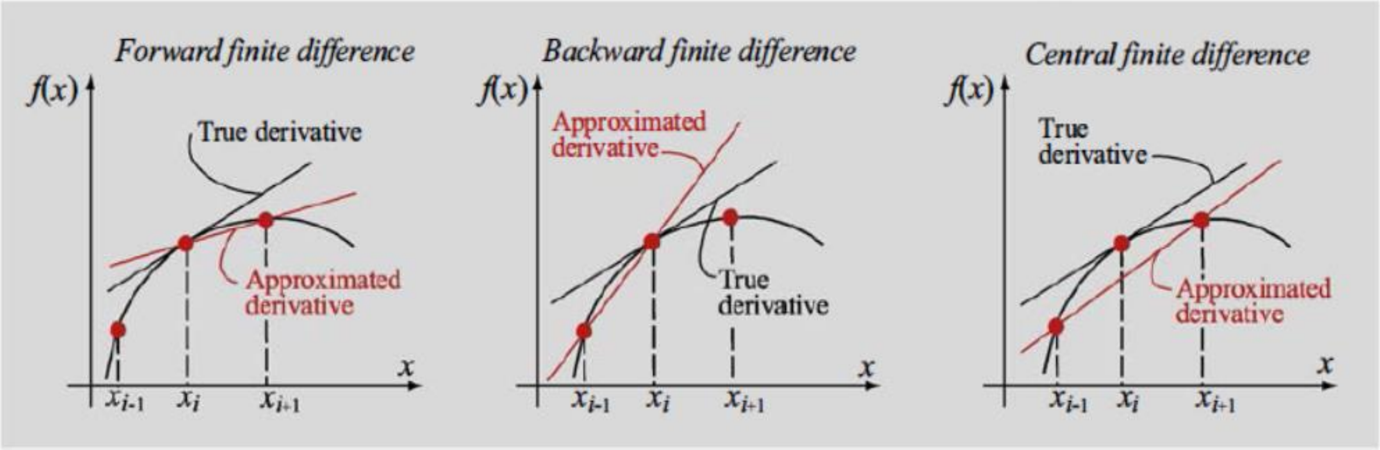


Figure 1-1: Finite difference approximation of derivative.

- **Forward difference** is the slope of the line that connects points $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$:

$$\frac{df}{dx}\Big|_{x=x_i} = \frac{f(x_{i+1})-f(x_i)}{x_{i+1}-x_i} \qquad \textbf{(1.1)}$$

- **Backward difference** is the slope of the line that connects points $(x_{i-1}, f(x_{i-1}))$ and $(x_i, f(x_i))$:

$$\frac{df}{dx}\Big|_{x=x_i} = \frac{f(x_i)-f(x_{i-1})}{x_i-x_{i-1}} \qquad \textbf{(1.2)}$$

- **Central difference** is the slope of the line that connects points $(x_{i-1}, f(x_{i-1}))$ and $(x_{i+1}, f(x_{i+1}))$:

$$\frac{df}{dx}\Big|_{x=x_i} = \frac{f(x_{i+1})-f(x_{i-1})}{x_{i+1}-x_{i-1}} \qquad \textbf{(1.3)}$$

Example 1-1: Comparing numerical and analytical differentiation.

Consider the function $f(x) = x^3$ .Calculate its first derivative at point *x = 3* numerically with the forward, backward, and central finite difference formulas and using:

(a) Points x = 2, x = 3, and x = 4.

(b) Points x = 2.75, x = 3, and x = 3.25.

Compare the results with the exact (analytical) derivative.

SOLUTION

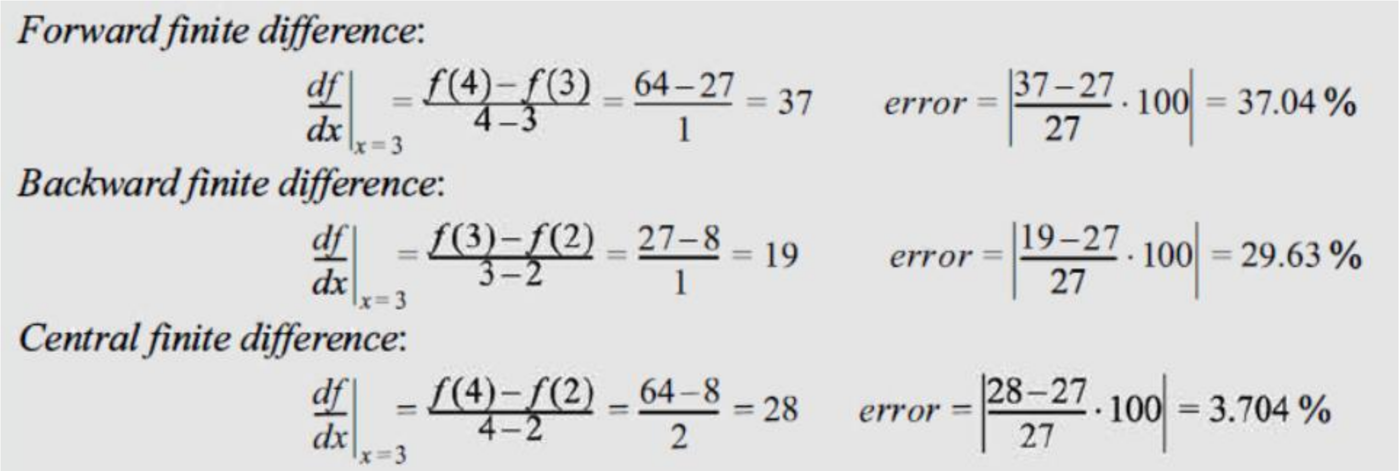
Analytical differentiation: The derivative of the function is$f'(x) = 3x^2$, and the value of the derivative at *x = 3* is $f'(3) = 3(3^2) = 27$.

Numerical differentiation:

(a) The points used for numerical differentiation are:

X	2	3	4
f(x)	8	27	64

Using Eqs. (1.1) through (1.3), the derivatives using the forward, backward, and central finite difference formulas are:



Forward finite difference:

$$\left.\frac{df}{dx}\right|_{x=3} = \frac{f(4)-f(3)}{4-3} = \frac{64-27}{1} = 37 \qquad error = \left|\frac{37-27}{27}\cdot 100\right| = 37.04\,\%$$

Backward finite difference:

$$\left.\frac{df}{dx}\right|_{x=3} = \frac{f(3)-f(2)}{3-2} = \frac{27-8}{1} = 19 \qquad error = \left|\frac{19-27}{27}\cdot 100\right| = 29.63\,\%$$

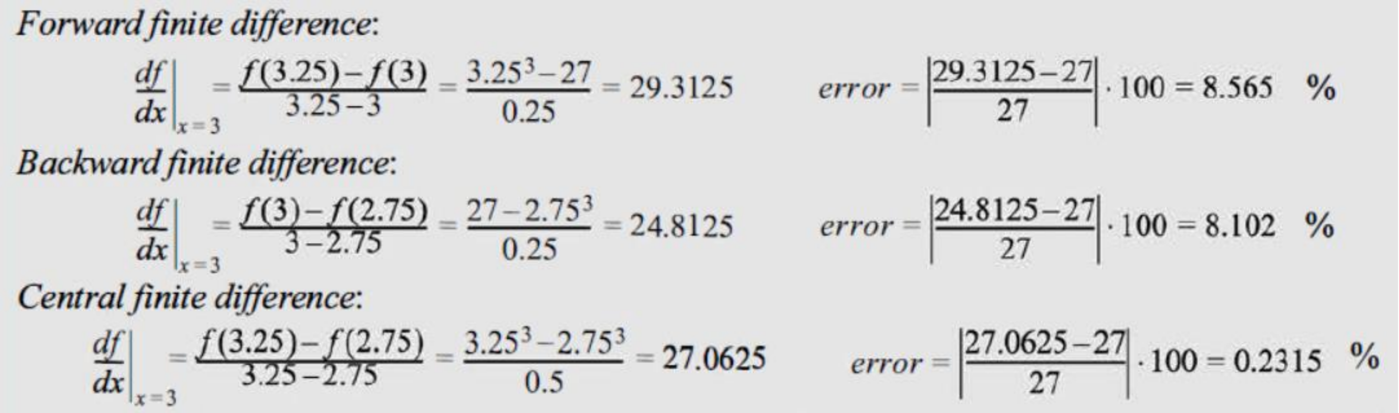
Central finite difference:

$$\left.\frac{df}{dx}\right|_{x=3} = \frac{f(4)-f(2)}{4-2} = \frac{64-8}{2} = 28 \qquad error = \left|\frac{28-27}{27}\cdot 100\right| = 3.704\,\%$$

(b)The points used for numerical differentiation are:

X	2.75	3	3.25
f(x)	2.75^3	3^3	3.25^3

Using Eqs. (1.1) through (1.3), the derivatives using the forward, backward, and central finite difference formulas are:



Forward finite difference:

$$\left.\frac{df}{dx}\right|_{x=3} = \frac{f(3.25)-f(3)}{3.25-3} = \frac{3.25^3-27}{0.25} = 29.3125 \qquad error = \left|\frac{29.3125-27}{27}\right|\cdot 100 = 8.565 \quad \%$$

Backward finite difference:

$$\left.\frac{df}{dx}\right|_{x=3} = \frac{f(3)-f(2.75)}{3-2.75} = \frac{27-2.75^3}{0.25} = 24.8125 \qquad error = \left|\frac{24.8125-27}{27}\right|\cdot 100 = 8.102 \quad \%$$

Central finite difference:

$$\left.\frac{df}{dx}\right|_{x=3} = \frac{f(3.25)-f(2.75)}{3.25-2.75} = \frac{3.25^3-2.75^3}{0.5} = 27.0625 \qquad error = \left|\frac{27.0625-27}{27}\right|\cdot 100 = 0.2315 \quad \%$$

The results show that the central finite difference formula gives a more accurate approximation. This will be discussed further in the next section. In addition, smaller separation between the points gives a significantly more accurate approximation.

1.2 Finite Difference Formulas Using Taylor Series Expansion

The forward, backward, and central difference formulas, as well as many other finite difference formulas for approximating derivatives, can be derived by using Taylor series expansion. The formulas give an estimate of the derivative at a point from the values of points in its neighborhood. The number of points used in the calculation varies with the formula, and the points can be ahead, behind, or on both sides of the point at which the derivative is calculated. One advantage of using Taylor series expansion for deriving the formulas is that it also provides an estimate for the truncation error in the approximation.

1.2.1 Finite Difference Formulas of First Derivative

Several formulas for approximating the first derivative at point x_i based on the values of the points near $x_i$are derived by using the Taylor series expansion. All the formulas derived in this section are for the case where the points are equally spaced.

<u>Two-point forward difference formula for first derivative</u>

The value of a function at point x_{i+1}can be approximated by a Taylor series in terms of the value of the function and its derivatives at pointx_i:

$$\boldsymbol{f}(\boldsymbol{x_{i+1}}) = \boldsymbol{f}(\boldsymbol{x_i}) + \boldsymbol{f}'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(x_i)}{3!}h^3 + \frac{f^{(4)}(x_i)}{4!}h^4 + \cdots \qquad (1.4)$$

where h=$x_{i+1} - x_i$; is the spacing between the points. By using two terms Taylor series expansion with a remainder can be rewritten as:

$$\boldsymbol{f}(\boldsymbol{x_{i+1}}) = \boldsymbol{f}(\boldsymbol{x_i}) + \boldsymbol{f}'(x_i)h + \frac{f''(\xi)}{2!}h^2 \qquad (1.5)$$

where ξ is a value of *x* between x_i and x_{i+1}. Solving Eq. (1.5) for $\boldsymbol{f}'(x_i)$ yields:

$$\boldsymbol{f}'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{f''(\xi)}{2!}h \qquad (1.6)$$

An approximate value of the derivative $\boldsymbol{f}'(x_i)$ can now be calculated if the second term on the right-hand side of Eq. (1.6) is ignored. Ignoring this second term introduces a truncation (discretization) error. Since this term is proportional to h, the truncation error is said to be on the order of h (written as O(h)):

$$\boldsymbol{truncation\ error} = -\frac{f''(\xi)}{2!}h = O(h) \qquad (1.7)$$

Using the notation of Eq. (1.7), the approximated value of the first derivative is:

$$\boldsymbol{f}'(x_i) = \frac{\boldsymbol{f}(\boldsymbol{x_{i+1}}) - \boldsymbol{f}(\boldsymbol{x_i})}{h} - O(h) \qquad (1.8)$$

The approximation in Eq. (1.8) is the same as the forward difference formula in Eq. (1.1).

<u>Two-point backward difference formula for first derivative</u>

The backward difference formula can also be derived by application of Taylor series expansion. The value of the function at point x_{i-1} is approximated by a Taylor series in terms of the value of the function and its derivatives at point x_i:

$$\boldsymbol{f}(\boldsymbol{x_{i-1}}) = \boldsymbol{f}(\boldsymbol{x_i}) - \boldsymbol{f}'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f'''(x_i)}{3!}h^3 + \frac{f^{(4)}(x_i)}{4!}h^4 - \cdots \qquad (1.9)$$

where h=$x_i - x_{i-1}$; is the spacing between the points. By using two terms Taylor series expansion with a remainder can be rewritten as:

$$\boldsymbol{f}(\boldsymbol{x_{i-1}}) = \boldsymbol{f}(\boldsymbol{x_i}) - \boldsymbol{f}'(x_i)h + \frac{f''(\xi)}{2!}h^2 \qquad (1.10)$$

where ξ is a value of x between x_i and x_{i+1}. Solving Eq. (1.10) for $\boldsymbol{f'}(x_i)$ yields:

$$\boldsymbol{f'}(x_i) = \frac{f(x_i)-f(x_{i-1})}{h} + \frac{f''(\xi)}{2!}h \qquad (1.11)$$

An approximate value of the derivative $\boldsymbol{f'}(x_i)$ can now be calculated if the second term on the right-hand side of Eq. (1.11) is ignored. This yileds:

$$\boldsymbol{f'}(x_i) = \frac{f(x_i)-f(x_{i-1})}{h} + O(h) \qquad (1.12)$$

The approximation in Eq. (1.12) is the same as the forward difference formula in Eq. (1.2).

Two-point central difference formula for first derivative

The central difference formula can be derived by using three terms in the Taylor series expansion and a remainder. The value of the function at point x_{i+1} in terms of the value of the function and its derivatives at point x_i is given by:

$$\boldsymbol{f}(\boldsymbol{x_{i+1}}) = \boldsymbol{f}(\boldsymbol{x_i}) + \boldsymbol{f'}(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(\zeta_1)}{3!}h^3 \qquad (1.13)$$

where ζ_1 is a value of x between x_i and x_{i+1}·The value of the function at point x_{i-1} in terms of the value of the function and its derivatives at point x_i is given by:

$$\boldsymbol{f}(\boldsymbol{x_{i-1}}) = \boldsymbol{f}(\boldsymbol{x_i}) - \boldsymbol{f'}(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f'''(\zeta_2)}{3!}h^3 \qquad (1.14)$$

where ζ_2 is a value of x between x_{i-1} and x_i. In the last two equations, the spacing of the intervals is taken to be equal so that h = x_{i+1}-x_i = x_i-x_{i-1}. Subtracting Eq. (1.14) from Eq. (1.13) gives:

$$\boldsymbol{f}(\boldsymbol{x_{i+1}}) - \boldsymbol{f}(\boldsymbol{x_{i-1}}) = \boldsymbol{2f'}(x_i)h + \frac{f'''(\zeta_1)}{3!}h^3 + \frac{f'''(\zeta_2)}{3!}h^3 \qquad (1.15)$$

An estimate for the first derivative is obtained by solving Eq. (1.15) for $\boldsymbol{f'}(x_i)$ while neglecting the remainder terms, which introduces a truncation error, which is of the order of h^2 :

$$\boldsymbol{f'}(x_i) = \frac{f(x_{i+1})-f(x_{i-1})}{2h} + O(h^2) \qquad (1.16)$$

The approximation in Eq. (1.16) is the same as the central difference formula Eq. (1.3) for equally spaced intervals.

1.2.2 Finite Difference Formulas for the Second Derivative

The same approach used in Section 1.2.1 to develop finite difference formulas for the first derivative can be used to develop expressions for higher-order derivatives. In this section, expressions based on central differences, one-sided forward differences, and one-sided backward differences are presented for approximating the second derivative at a point x_i.

Three-point central difference formula for the second derivative

Central difference formulas for the second derivative can be developed using any number of points on either side of the point x_i, where the second derivative is to be evaluated. The formulas are derived by writing the Taylor series expansion with a remainder at points on either side of x_i in terms of the value of the function and its derivatives at point x_i. Then, the equations are combined in such a way that the terms containing the first derivatives are eliminated. For example, for points x_{i+1}, and $x_{i-1}$the four-term Taylor series expansion with a remainder is:

$$\boldsymbol{f(x_{i+1})} = \boldsymbol{f(x_i)} + \boldsymbol{f'}(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(x_i)}{3!}h^3 + \frac{f^{(4)}(\zeta_1)}{4!}h^4 \qquad (1.17)$$

$$\boldsymbol{f(x_{i-1})} = \boldsymbol{f(x_i)} - \boldsymbol{f'}(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f'''(x_i)}{3!}h^3 + \frac{f^{(4)}(\zeta_2)}{4!}h^4 \qquad (1.18)$$

where ζ_1 is a value of x between x_i and x_{i+1}. and ζ_2 is a value of x between x_{i-1} and x_i. Adding Eq. (1.17) and Eq. (1.18) gives:

$$\boldsymbol{f(x_{i+1})} + \boldsymbol{f(x_{i-1})} = \boldsymbol{2f(x_i)} + \boldsymbol{2}\frac{f''(x_i)}{2!}h^2 + + \frac{f^{(4)}(\zeta_1)}{4!}h^4 + \frac{f^{(4)}(\zeta_2)}{4!}h^4 \qquad (1.19)$$

An estimate for the second derivative can be obtained by solving Eq.(1.19) for $\boldsymbol{f''}(x_i)$ while neglecting the remainder terms. This introduces a truncation error of the order of h^2.

$$\boldsymbol{f''}(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} + O(h^2) \qquad (1.20)$$

Example 1-2: Comparing numerical and analytical differentiation.

Consider the function $f(x) = \frac{2^x}{x}$. Calculate the second derivative at x = 2 numerically with the three-point central difference formula using:

(a) Points x = 1.8 , x = 2 , and x = 2.2 .

(b) Points x=l.9, x=2, and x=2.1.

Compare the results with the exact (analytical) derivative.

SOLUTION

Analytical differentiation: The second derivative of the function $f(x) = \frac{2^x}{x}$ is:

$$f'(x) = \frac{2^x - x(\ln 2)2^x}{x^2} = \frac{2^x}{x^2} - \ln 2\frac{2^x}{x}$$

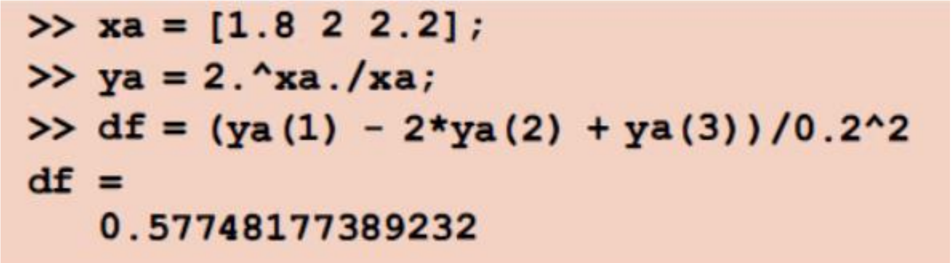
$$f''(x) = \frac{2^x(2x) - x^2(\ln 2)2^x}{x^4} - \ln 2\left(\frac{2^x}{x^2} - \ln 2\frac{2^x}{x}\right)$$

$$\Rightarrow f''(x) = (\ln 2)^2\frac{2^x}{x} - 2(\ln 2)\frac{2^x}{x^2} + 2\frac{2^x}{x^3}$$

and the value of the derivative at x = 2 is f'' (2) = 0.574617 .

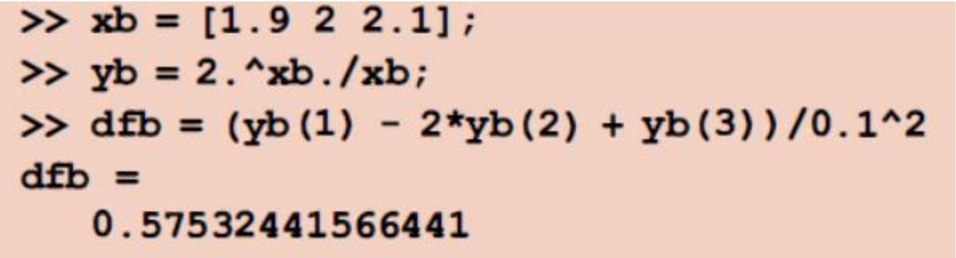
Numerical differentiation

(a) The numerical differentiation is done by substituting the values of the points x = 1.8, x = 2, and x = 2.2 in Eq. (1.20). The operations are done with MATLAB, in the Command Window:

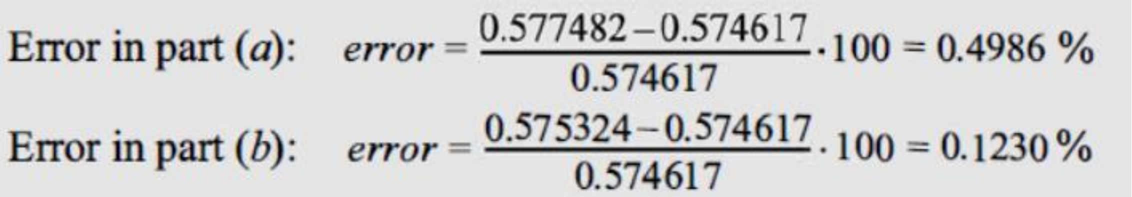


```
>> xa = [1.8 2 2.2];
>> ya = 2.^xa./xa;
>> df = (ya(1) - 2*ya(2) + ya(3))/0.2^2
df =
   0.57748177389232
```

(b) The numerical differentiation is done by substituting the values of the points x = 1.9, x = 2, and x = 2.1 in Eq. (1.20). The operations are done with MATLAB, in the Command Window:



```
>> xb = [1.9 2 2.1];
>> yb = 2.^xb./xb;
>> dfb = (yb(1) - 2*yb(2) + yb(3))/0.1^2
dfb =
   0.57532441566441
```



Error in part (a): $error = \frac{0.577482 - 0.574617}{0.574617} \cdot 100 = 0.4986\ \%$

Error in part (b): $error = \frac{0.575324 - 0.574617}{0.574617} \cdot 100 = 0.1230\ \%$

The results show that the three-point central difference formula gives a quite accurate approximation for the value of the second derivative.

1.3 Summary of Finite Difference Formulas for Numerical Differentiation

Table 3-1 lists difference formulas, of various accuracy, that can be used for numerical evaluation of first, second, third, and fourth derivatives. The formulas can be used when the function that is being differentiated is specified as a set of discrete points with the independent variable **equally spaced**.

Table 1-1: Finite difference formulas.

First Derivative		
Method	**Formula**	**Truncation Error**
Two-point forward difference	$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}$	$O(h)$
Three-point forward difference	$f'(x_i) = \frac{-3f(x_i) + 4f(x_{i+1}) - f(x_{i+2})}{2h}$	$O(h^2)$
Two-point backward difference	$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h}$	$O(h)$
Three-point backward difference	$f'(x_i) = \frac{f(x_{i-2}) - 4f(x_{i-1}) + 3f(x_i)}{2h}$	$O(h^2)$
Two-point central difference	$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$	$O(h^2)$
Four-point central difference	$f'(x_i) = \frac{f(x_{i-2}) - 8f(x_{i-1}) + 8f(x_{i+1}) - f(x_{i+2})}{12h}$	$O(h^4)$
Second Derivative		
Method	**Formula**	**Truncation Error**
Three-point forward difference	$f''(x_i) = \frac{f(x_i) - 2f(x_{i+1}) + f(x_{i+2})}{h^2}$	$o(h)$
Four-point forward difference	$f''(x_i) = \frac{2f(x_i) - 5f(x_{i+1}) + 4f(x_{i+2}) - f(x_{i+3})}{h^2}$	$o(h^2)$

Three-point backward difference	$\boldsymbol{f''(x_i) = \frac{f(x_{i-2}) - 2f(x_{i-1}) + f(x_i)}{h^2}}$	$\boldsymbol{o(h)}$
Four-point backward difference	$\boldsymbol{f''(x_i) = \frac{-f(x_{i-3}) + 4f(x_{i-2}) - 5f(x_{i-1}) + 2f(x_i)}{h^2}}$	$\boldsymbol{o(h^2)}$
Three-point central difference	$\boldsymbol{f''(x_i) = \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1})}{h^2}}$	$\boldsymbol{o(h^2)}$
Five-point central difference	$f''(x_i) = \frac{-f(x_{i-2}) + 16f(x_{i-1}) - 30f(x_i) + 16f(x_{i+1}) - f(x_{i+2})}{12h^2}$	$\boldsymbol{o(h^4)}$

1.4 DIFFERENTIATION FORMULAS USING LAGRANGE POLYNOMIALS

The differentiation formulas can also be derived by using Lagrange polynomials. For the first derivative, the two-point central, three-point forward, and three-point backward difference formulas are obtained by considering three points (x_i, y_i), (x_{i+1}, y_{i+1}) , and (x_{i+2}, y_{i+2}). The polynomial, in Lagrange form, that passes through the points is given by:

$$f(x) = y_i \frac{(x-x_{i+1})(x-x_{i+2})}{(x_i-x_{i+1})(x_i-x_{i+2})} + y_{i+1} \frac{(x-x_i)(x-x_{i+2})}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})} + y_{i+2} \frac{(x-x_i)(x-x_{i+1})}{(x_{i+2}-x_i)(x_{i+2}-x_{i+1})} \quad (1.21)$$

Differentiating Eq.(1.21) gives:

$$f'(x) = y_i \frac{2x-x_{i+1}-x_{i+2}}{(x_i-x_{i+1})(x_i-x_{i+2})} + y_{i+1} \frac{2x-x_i-x_{i+2}}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})} + y_{i+2} \frac{2x-x_i-x_{i+1}}{(x_{i+2}-x_i)(x_{i+2}-x_{i+1})} \quad (1.22)$$

The first derivative at either one of the three points is calculated by substituting the corresponding value of x (x_i, x_{i+1} or x_{i+2}) in Eq. (1.22). This gives the following three formulas for the first derivative at the three points.

$$f'(x_i) = y_i \frac{2x_i-x_{i+1}-x_{i+2}}{(x_i-x_{i+1})(x_i-x_{i+2})} + y_{i+1} \frac{2x_i-x_i-x_{i+2}}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})} + y_{i+2} \frac{2x_i-x_i-x_{i+1}}{(x_{i+2}-x_i)(x_{i+2}-x_{i+1})} \quad (1.23)$$

When the points are equally spaced, Eq. (1.23) reduces to the **three point forward difference formula**:

$$\boldsymbol{f'(x_i) = \frac{-3f(x_i) + 4f(x_{i+1}) - f(x_{i+2})}{2h}}$$

$$f'(x_{i+1}) = y_i \frac{2x_{i+1}-x_{i+1}-x_{i+2}}{(x_i-x_{i+1})(x_i-x_{i+2})} + y_{i+1} \frac{2x_{i+1}-x_i-x_{i+2}}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})} + y_{i+2} \frac{2x_{i+1}-x_i-x_{i+1}}{(x_{i+2}-x_i)(x_{i+2}-x_{i+1})} \quad (1.24)$$

When the points are equally spaced, Eq. (1.24) reduces to the **two point central difference formula**:

$$\boldsymbol{f'(x_{i+1}) = \frac{f(x_{i+2}) - f(x_i)}{2h}}$$

Which is:

$$\boldsymbol{f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}}$$

$$f'(x_{i+2}) = y_i \frac{2x_{i+2}-x_{i+1}-x_{i+2}}{(x_i-x_{i+1})(x_i-x_{i+2})} + y_{i+1} \frac{2x_{i+2}-x_i-x_{i+2}}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})} + y_{i+2} \frac{2x_{i+2}-x_i-x_{i+1}}{(x_{i+2}-x_i)(x_{i+2}-x_{i+1})} \quad (1.25)$$

When the points are equally spaced, Eq. (1.24) reduces to the **three point backward difference formula**:

$$\boldsymbol{f'(x_i) = \frac{f(x_{i-2}) - 4f(x_{i-1}) + 3f(x_i)}{2h}}$$

(1.4) First Derivatives From Interpolating Polynomials:

We begin with a Newton-Gregory forward polynomial:

$$f(x_t) = f_0 + t\Delta f_0 + \frac{t(t-1)}{2!}\Delta^2 f_0 + \frac{t(t-1)(t-2)}{3!}\Delta^3 f_0 + \cdots + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n f_0 + \cdots \qquad (1.26)$$

Differentiating Eq.(1.26) , remembering that f_0 and all the Δ-terms are constants (after all, they are just the numbers from the difference table), we have:

$$f'(x_t) = \frac{d}{dx}[f(x_t)] = \frac{d}{dt}[f(x_t)]\frac{1}{h}$$

$$= \frac{1}{h}\left[\Delta f_0 + \frac{(2t-1)}{2!}\Delta^2 f_0 + \frac{3t^2-6t+2}{3!}\Delta^3 f_0 + \cdots\right] \qquad (1.27)$$

If we let *t=0*, giving us the derivative corresponding to x_0, we have:

$$f'(x_0) = \frac{1}{h}\left[\Delta f_0 - \frac{1}{2}\Delta^2 f_0 + \frac{1}{3}\Delta^3 f_0 - \frac{1}{4}\Delta^4 f_0 \ldots\right] \qquad (1.28)$$

1.5 Use of MATLAB Built-In Functions for Numerical Differentiation

In general, it is recommended that the techniques described in this chapter be used to develop script files that perform the desired differentiation. MATLAB does not have built-in functions that perform numerical differentiation of an arbitrary function or discrete data. There is, however, a built-in function called **diff**, which can be used to perform numerical differentiation, and another built-in function called **polyder**, which determines the derivative of polynomial.

1.5.1 The diff command

The built-in function **diff** calculates the derivative of the functions:

```
>> syms x
>> diff(x^3+2*x^2-1)
ans =
3*x^2 + 4*x
>> diff(x^3+2*x^2-1,2)
ans =
6*x + 4
>> diff(x^3+2*x^2-1,3)
ans =
6
```

1.5.2 The polyder command

The built-in function **polyder** can calculate the derivative of a polynomial (it can also calculate the derivative of a product and quotient of two polynomials).

```
>> p=[4 0 2 5]
p =
    4    0    2    5
>> polyder(p)
ans =
   12    0    2
```

1.6 PROBLEMS

1. Given the following data:

x	1	1.2	1.3	1.4	1.5
f(x)	0.6133	0.7882	0.9716	1.1814	1.4117

Find the first derivative $f'(x)$ at the point *x* = 1.3.
(a) Use the three-point forward difference formula.
(b) Use the three-point backward difference formula.
(c) Use the two-point central difference formula.

2. The following data is given for the stopping distance of a car on a wet road versus the speed at which it begins braking.

v(mi/h)	12.5	25	37.5	50	62.5	75
d(ft)	20	59	118	197	299	420

Calculate the rate of change of the stopping distance at a speed of 62.5 mph using:
(i) the two-point backward difference formula, and (ii) the three-point backward difference formula.

a. Use Lagrange interpolation polynomials to find the finite difference formula for the second derivative at the point $x = x_i$ using the unequally spaced points $x = x_{i+1}$, and $x = x_{i+2}$ What is the second derivative at $x = x_{i+1}$and at $x = x_{i+2}$?

3. Find the first derivative from backward polynomial approximated to the forth difference.
4. Find the second derivative from forward polynomial to the forth difference.
5. Use the data below to estimate the derivative of y at x=1.7:

x	1.3	1.5	1.7	1.9	2.1	2.3	2.5
f(x)	3.669	4.482	5.474	6.686	8.166	9.974	12.182

Chapter2: Numerical Integration

2.1 Introduction to Quadrature:

We now approach the subject of numerical integration. The goal is to approximate the definite integral of f(x) over the interval [a,b] by evaluating f(x) at a finite number of sample points.

Definition(2.1): Suppose that $a=x_0<x_1<\ldots<x_M=b$. A formula of the form:

$$Q[f]=\sum_{k=0}^{M} w_k f(x_k)=w_0 f(x_0)+w_1 f(x_1)+\cdots+w_M f(x_M) \qquad (2.1)$$

With the property that:

$$\int_a^b f(x)dx=Q[f]+E[f] \qquad (2.2)$$

is called a numerical integration or **quadrature** formula. The term *E[f]* is called the **truncation error** for integration. The values $\{x_k\}_{k=0}^{M}$ are called the **quadrature nodes** and $\{w_k\}_{k=0}^{M}$ are called **weights**.

Definition (2.2): The **degree of precision** of a quadrature formula is the positive integer n such that $E[P_i]=0$ for all polynomials $P_i(x)$ of degree $i \le n$, but for which $E[P_{n+1}]\neq 0$ for some polynomial $P_{n+1}(x)$ of degree n+1.

Theorem(2.1): (closed Newton-cotes Quadrature formula)

Assume that $x_k=x_0+kh$ are equally spaced nodes and $f_k=f(x_k)$. The first four closed Newton-Cotes quadrature formulas are

$$\int_{x_0}^{x_1} f(x)dx \approx \frac{h}{2}(f_0+f_1) \qquad (2.3)$$ **(the trapezoidal rule)**

$$\int_{x_0}^{x_2} f(x)dx \approx \frac{h}{3}(f_0+4f_1+f_2) \qquad (2.4)$$ **(Simpson rule)**

$$\int_{x_0}^{x_3} f(x)dx \approx \frac{3h}{8}(f_0+3f_1+3f_2+f_3) \qquad (2.5)$$ **(Simpson's $\frac{3}{8}$ rule)**

$$\int_{x_0}^{x_4} f(x)dx \approx \frac{2h}{45}(7f_0+32f_1+12f_2+32f_3+7f_4) \quad (2.6)$$ **(Boole's rule)**

Corollary(2.1): (Newton-Cotes precision)

Assume that *f(x)* is sufficiently differentiable; then *E[f]* for Newton-Cotes quadrature involves an approximate higher derivative. The trapezoidal rule has degree of precision n=1. If $f \in C^2[a,b]$, then:

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{2}(f_0+f_1) - \frac{h^3}{12}f^{(2)}(c) \qquad (2.7)$$

Simpson's rule has degree of precision *n=3*. If $f \in C^4[a,b]$, then:

$$\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}(f_0+4f_1+f_2) - \frac{h^5}{90}f^{(4)}(c) \qquad (2.8)$$

Simpson's $\frac{3}{8}$ rule has degree of precision *n*=3. If $f \in C^4[a,b]$, then:

$$\int_{x_0}^{x_3} f(x)dx = \frac{3h}{8}(f_0+3f_1+3f_2+f_3) - \frac{3h^5}{80}f^{(4)}(c) \qquad (2.9)$$

Boole's rule has degree of precision *n*=5. If $f \in C^6[a,b]$, then:

$$\int_{x_0}^{x_4} f(x)dx = \frac{2h}{45}(7f_0+32f_1+12f_2+32f_3+7f_4) - \frac{8h^7}{945}f^{(6)}(c) \quad (2.10)$$

Proof of Theorem(2.1): Start with the Lagrange polynomial *P_M(x)* based on $x_0, x_1, \ldots, x_M$ that can be used to approximate *f(x)*:

$$f(x) \approx P_M(x) = \sum_{k=0}^{M} f(x_k) \prod_{\substack{j=0 \\ j\neq k}}^{M} \frac{(x-x_j)}{(x_k-x_j)} \qquad (2.11)$$

An approximate for the integral is obtained by replacing the integrand *f(x)* with the polynomial $P_M(x)$. This is the general method for obtaining a Newton-Cotes integration formula:

$$\int_{x_0}^{x_M} f(x)dx \approx \int_{x_0}^{x_M} P_M(x)dx = \int_{x_0}^{x_M}\left(\sum_{k=0}^{M} f_k \prod_{\substack{j=0\\ j\neq k}}^{M} \frac{(x-x_j)}{(x_k-x_j)}\right) \qquad (2.12)$$

The details for the general proof of the theorem are tedious. We shall give a Simpson's rule, which is the case M=2. This case involves the approximation polynomial

$$P_2(x) = f_0 \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f_1 \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + f_2 \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} \qquad (2.13)$$

Since *f₀*, *f₁* and *f₂* are constant with respect to integration, the relations in (2.12) lead to:

$$\int_{x_0}^{x_2} f(x)dx \approx \int_{x_0}^{x_2} f_0 \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}dx + \int_{x_0}^{x_2} f_1 \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}dx + \int_{x_0}^{x_2} f_2 \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}dx$$

(2.14)

We introduce the change of variable *x=x₀+th* with *dx=hdt* to assist with the evaluation of the integrals in (2.14). The new limits of integration are from *t=0* to *t=2*. The equal spacing of the nodes $x_k=x_0+kh$ leads to $x_k\text{-}x_j=(k\text{-}j)h$ and $x\text{-}x_k=(t\text{-}k)h$, which are used to simplify (2.14), and get:

$$\int_{x_0}^{x_2} f(x)dx \approx f_0 \int_0^2 \frac{h(t-1)h(t-2)}{(-h)(-2h)}hdt + f_1 \int_0^2 \frac{h(t-0)h(t-2)}{(h)(-h)}hdt + f_2 \int_0^2 \frac{h(t-0)h(t-1)}{(2h)(h)}hdt$$

$$= f_0 \frac{h}{2}\int_0^2 (t^2-3t+2)dt + f_1 h \int_0^2 (t^2-2t)dt + f_2 \frac{h}{2}\int_0^2 (t^2-t)dt$$

$$= f_0 \frac{h}{2}\left(\frac{t^3}{3} - \frac{3t^2}{2} + 2t\right)\Big|_{t=0}^{t=2} - f_1 h\left(\frac{t^3}{3} - \frac{2t^2}{2}\right)\Big|_{t=0}^{t=2} + f_2 \frac{h}{2}\left(\frac{t^3}{3} - \frac{t^2}{2}\right)\Big|_{t=0}^{t=2}$$

$$= f_0 \frac{h}{2}\left(\frac{2}{3}\right) - f_1 h\left(\frac{-4}{3}\right) + f_2 \frac{h}{2}\left(\frac{2}{3}\right)$$

$$= \frac{h}{3}(f_0 + 4f_1 + f_2)$$

and the proof is complete.

<u>Example(2.1)</u>: Consider the function f(x)=1+e^{-x}sin(4x), the equally spaced quadrature nodes x_0 =0, x_1 =0.5, x_2 =1, x_3=1.5, x_4 =2 and the corresponding function values f_0 =1, f_1=1.55152, f_2=0.72159, f_3=0.93765 and f_4=1.13390. Apply the various quadrature formulas (2.3) through (2.6).

The step size is h=0.5, and the computations are:

$$\int_0^{0.5} f(x)dx \approx \frac{0.5}{2}(1 + 1.55152) = 0.63788$$

$$\int_0^{1} f(x)dx \approx \frac{0.5}{3}(1 + 4(1.55152) + 0.72159) = 1.32128$$

$$\int_0^{1.5} f(x)dx \approx \frac{3(0.5)}{8}(1 + 3(1.55152) + 3(0.72159) + 0.93765) = 1.64193$$

$$\int_0^{2} f(x)dx \approx \frac{2(0.5)}{45}\big(7(1) + 32(1.55152) + 12(0.72159) + 32(0.93765) + 7(1.1339)\big)$$

$$= 2.29444$$

<u>Examples (2.2)</u>: Consider the integration of the function f(x)=1+e^{-x}sin(4x) over the fixed interval [a,b]=[0,1]. Apply the various formulas (2.3) through (2.6).

For the trapezoidal rule, h=1 and

$$\int_0^1 f(x)dx \approx \frac{1}{2}\big(f(0)+f(1)\big) = \frac{1}{2}(1+0.72159) = 0.86079$$

For Simpson's rule, h=1/2, and we get:

$$\int_0^1 f(x)dx \approx \frac{1/2}{3}\left(f(0)+4f\left(\frac{1}{2}\right)+f(1)\right) = \frac{1}{6}(1+4(1.55152)+0.72159) = 1.32128$$

For Simpson's $\frac{3}{8}$ rule, h=1/3, and we obtain:

$$\int_0^1 f(x)dx \approx \frac{3\left(\frac{1}{3}\right)}{8}\left(f(0)+3f\left(\frac{1}{3}\right)+3f\left(\frac{2}{3}\right)+f(1)\right)$$

$$= \frac{1}{8}(1+3(1.69642)+3(1.23447)+0.72159) = 1.31440$$

For Boole's rule, h=1/4, and the result is:

$$\int_0^1 f(x)dx \approx \frac{2\left(\frac{1}{4}\right)}{45}\left(7f(0)+32f\left(\frac{1}{4}\right)+12f\left(\frac{1}{2}\right)+32f\left(\frac{3}{4}\right)+7f(1)\right)$$

$$= \frac{1}{90}\big(7(1)+32(1.65534)+12(1.55152)+32(1.06666)+7(0.72159)\big)$$

$$=1.30859$$

The true value of the definite integral is:

$$\int_0^1 f(x)dx = 1.308\,250\,604$$

To make a fair comparison of quadrature methods, we must use the same number of function evaluations in each method. Our final example is concerned with comparing integration over a fixed interval [a,b] using exactly five function evaluation $f_k=f(x_k)$, for k=0,1,…,4 for each method. When the trapezoidal rule is applied on the four subintervals [x_0,x_1], [x_1,x_2], [x_2,x_3] and [x_3,x_4], it is called a **composite trapezoidal rule**:

$$\int_{x_0}^{x_4} f(x)dx = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \int_{x_2}^{x_3} f(x)dx + \int_{x_3}^{x_4} f(x)dx$$

$$\approx \frac{h}{2}(f_0 + f_1) + \frac{h}{2}(f_1 + f_2) + \frac{h}{2}(f_2 + f_3) + \frac{h}{2}(f_3 + f_4)$$

$$= \frac{h}{2}(f_0 + 2f_1 + 2f_2 + 2f_3 + f_4) \qquad (2.15)$$

Simpson's rule can also be used in this manner. When Simpson's rule is applied on the two subintervals [x_0,x_2] and [x_2,x_4], it is called a **composite Simpson's rule**:

$$\int_{x_0}^{x_4} f(x)dx = \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx$$

$$\approx \frac{h}{3}(f_0 + 4f_1 + f_2) + \frac{h}{3}(f_2 + 4f_3 + f_4)$$

$$= \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + f_4) \qquad (2.16)$$

Example(2.3): Consider the integration of the function f(x)=1+e^{-x}sin(4x) over [a,b]=[0,1]. Use exactly five function evaluations and compare the results from the composite trapezoidal rule and composite Simpson's rule.

The uniform step size is h=1/4. The composite trapezoidal rule (2.15) produces:

$$\int_0^1 f(x)dx \approx \frac{1/4}{2}\left(f(0) + 2f\left(\frac{1}{4}\right) + 2f\left(\frac{1}{2}\right) + 2f\left(\frac{3}{4}\right) + f(1)\right)$$

$$= \frac{1}{8}(1 + 2(1.65534) + 2(1.55152) + 2(1.06666) + 0.72159)$$

=1.28358

Using the composite Simpson's rule (2.16), we get:

$$\int_0^1 f(x)dx \approx \frac{1/4}{3}\left(f(0)+4f\left(\frac{1}{4}\right)+2f\left(\frac{1}{2}\right)+4f\left(\frac{3}{4}\right)+f(1)\right)$$

$$=\frac{1}{12}(1+4(1.65534)+2(1.55152)+4(1.06666)+0.72159)$$

=1.30938

__Example(2.4):__ Determine the degree of precision of Simpson's $\frac{3}{8}$ rule.

It will suffice to apply Simpson's $\frac{3}{8}$ rule over the interval [0,3] with the five test functions f(x)=1, x, x^2, x^3, and x^4. For the first four functions. Simpson's $\frac{3}{8}$ rule is exact.

$$\int_0^3 1dx=\frac{3}{8}(1+3(1)+3(1)+1)=3$$

$$\int_0^3 xdx=\frac{3}{8}(0+3(1)+3(2)+3)=\frac{9}{2}$$

$\int_0^3 x^2dx=\frac{3}{8}(0+3(1)+3(4)+9)$ =9

$$\int_0^3 x^3dx=\frac{3}{8}(0+3(1)+3(8)+27)=\frac{81}{4}$$

the function f(x)=x^4 is the lowest power of x for which the rule is not exact.

$$\int_0^3 x^4dx=\frac{3}{8}(0+3(1)+3(16)+81)=\frac{99}{2}$$

Therefore, the degree of precision of Simpson's$\frac{3}{8}$ rule is n=3.

Exercises:

1. Consider a general interval [a,b]. Show that Simpson's rule produces exact results for the function f(x)=x^2 and f(x)=x^3, that is

   a. $\int_a^b x^2 dx = \frac{b^3}{3} - \frac{a^3}{3}$ b. $\int_a^b x^3 dx = \frac{b^4}{4} - \frac{a^4}{4}$

2. Integrate the Lagrange interpolation polynomial

   $$P_1(x) = f_0 \frac{(x-x_1)}{(x_0-x_1)} + f_1 \frac{(x-x_0)}{(x_1-x_0)}$$

   over the interval [x_0,x_1] and establish the trapezoidal rule.

3. Determine the degree of precision of the trapezoidal rule.

2.2 Other Ways to Derive Integration Formulas Using Newton Forward Polynomial:

During the integration we will need to change the variable of integration from x to t since our polynomials are expressed in terms of t. Observe that dx=hdt.

$$\int_{x_0}^{x_1} f(x)dx = h \int_{t=0}^{t=1} \left[f_0 + t\Delta f_0 + \frac{t(t-1)}{2!}\Delta^2 f_{0+} \frac{t(t-1)(t-2)}{3!}\Delta^3 f_0 + \cdots \right] dt$$

$$= h \int_0^1 \left[f_0 + t\Delta f_0 + \frac{t^2-t}{2}\Delta^2 f_0 + \frac{t^3-3t^2+2t}{6}\Delta^3 f_0 + \cdots \right] dt$$

$$= h \left[f_0 t + \frac{t^2}{2}\Delta f_0 + \left(\frac{t^3}{6} - \frac{t^2}{4}\right)\Delta^2 f_0 + \left(\frac{t^4}{24} - \frac{t^3}{6} + \frac{t^2}{6}\right)\Delta^3 f_0 + \cdots \right]_{t=0}^{t=1}$$

$$= h \left[f_0 + \frac{1}{2}\Delta f_0 - \frac{1}{12}\Delta^2 f_0 + \frac{1}{24}\Delta^3 f_0 + \cdots \right]$$

using first two terms only, we get:

$$\int_{x_0}^{x_1} f(x)dx = h\left[f_0 + \frac{1}{2}\Delta f_0\right] = h\left[f_0 + \frac{1}{2}(f_1 - f_0)\right] = \frac{h}{2}[f_0 + f_1]$$

Exercise:

Derive Simpson's formula using Newton Forward polynomial.

2.3 Composite Trapezoidal and Simpson's Rule:

Theorem(2.2): (Composite Trapezoidal Rule)

Suppose that the interval [a,b] is subdivided into subinterval $[x_k, x_{k+1}]$ of width h=(b-a)/M by using equally spaced nodes x_k=a+kh, for k=0,1,…,M. The **composite trapezoidal rule for M subintervals** can be expressed in:

$$\int_a^b f(x)dx \approx T(f,h) = \frac{h}{2}[f_0 + 2(f_1 + \cdots + f_{M-1}) + f_M]$$

$$= \frac{h}{2}[f(a) + f(b)] + h\sum_{k=1}^{M-1} f(x_k) \qquad (2.17)$$

Proof: Apply the trapezoidal rule over each subinterval $[x_{k-1}, x_k]$. Use the additive property of the integral for subintervals:

$$\int_a^b f(x)dx = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_{M-1}}^{x_M} f(x)dx$$

$$= \frac{h}{2}[f_0 + f_1] + \frac{h}{2}[f_1 + f_2] + \cdots + \frac{h}{2}[f_{M-1} + f_M]$$

$$= \frac{h}{2}[f_0 + 2(f_1 + f_2 + \cdots + f_{M-1}) + f_M].$$

Example(2.5): Consider $f(x) = 2 + \sin(2\sqrt{x})$. Use the composite trapezoidal rule with 11 sample points to compute an approximation to the integral of f(x) taken over [1,6].

To generate 11 sample points, we use M=10 and h=(6-1)/10=1/2.

x	*1*	*1.5*	*2*	*2.5*	*3*	*3.5*	*4*	*4.5*	*5*	*5.5*	*6*
f(x)	*2.909297*	*2.638157*	*2.308071*	*1.979316*	*1.683052*	*1.4353041*	*1.243197*	*1.108317*	*1.028722*	*1.000241*	*1.017357*

$\int_1^6 f(x)dx = \frac{\frac{1}{2}}{2}[f(1) + 2\big(f(1.5) + f(2) + f(2.5) + f(3) + f(3.5) + f(4) + f(4.5) + f(5) + f(5.5)\big) + f(6)]$=8.193854.

Theorem(2.3): (Composite Simpson Rule)

Suppose that [a,b] is subdivided into 2M subintervals [x_k, x_{k+1}] of equal width with h=(b-a)/(2M) by using x_k=a+kh for k=0,1,…,2M. The **composite Simpson rule for 2M subintervals** can be expressed in:

$$\int_a^b f(x)dx \approx S(f,h) = \frac{h}{3}[f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 2f_{2M-2} + 4f_{2M-1} + f_{2M}]$$

$$= \frac{h}{3}[f(a) + f(b)] + \frac{2h}{3}\sum_{k=1}^{M-1} f(x_{2k}) + \frac{4h}{3}\sum_{k=1}^{M} f(x_{2k-1}) \qquad (2.18)$$

proof: (EXC)

Example(2.6): Consider $f(x) = 2 + \sin(2\sqrt{x})$. Use the composite Simpson rule with 11 sample points to compute an approximation to the integral of f(x) taken over [1,6].

$\int_a^b f(x)dx = \frac{1/2}{3}[f(1) + f(6)] + \frac{1}{3}[f(2) + f(3) + f(4) + f(5)] + \frac{2}{3}[f(1.5) + f(2.5) + f(3.5) + f(4.5) + f(5.5)]$=8.1830155

Error Analysis:

Corollary(2.2): (Trapezoidal Rule: Error Analysis)

Suppose that [a,b] is subdivided into M subintervals [x_k, x_{k+1}] of width h=(b-a)/M. The composite trapezoidal rule:

$$T(f,h) = \frac{h}{2}[f(a) + f(b)] + h\sum_{k=1}^{M-1} f(x_k) \qquad (2.19)$$

is an approximation to the integral:

$$\int_a^b f(x)dx = T(f,h) + E_T(f,h) \qquad (2.20)$$

Furthermore, if $f \in C^2[a,b]$, there exists a value c with a<c<b so that the error term $E_T(f,h)$ has the form:

$$E_T(f,h) = \frac{-(b-a)f^{(2)}(c)h^2}{12} = O(h^2) \qquad (2.21)$$

Proof: We first determine the error term when the rule is applied over $[x_0, x_1]$. Integrating the Lagrange polynomial $P_1(x)$ and its remainder yields:

$$\int_{x_0}^{x_1} f(x)dx = \int_{x_0}^{x_1} P_1(x)dx + \int_{x_0}^{x_1} \frac{(x-x_0)(x-x_1)f^{(2)}(c(x))}{2!}dx \qquad (2.22)$$

The term $(x-x_0)(x-x_1)$ does not change sign on $[x_0, x_1]$, and $f^{(2)}(c(x))$ is continuous. Hence the second Mean value Theorem for integrals implies that there exists a value c_1 so that:

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{2}[f_0+f_1] + f^{(2)}(c_1)\int_{x_0}^{x_1} \frac{(x-x_0)(x-x_1)}{2!}dx \qquad (2.23)$$

Use the change of variable $x=x_0+ht$ in the integral on the right side of (2.23)

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{2}[f_0+f_1] + \frac{f^{(2)}(c_1)}{2}\int_0^1 h(t-0)h(t-1)hdt$$

$$= \frac{h}{2}[f_0+f_1] + \frac{f^{(2)}(c_1)h^3}{2}\int_0^1 (t^2-t)dt$$

$$= \frac{h}{2}[f_0+f_1] - \frac{f^{(2)}(c_1)h^3}{12} \qquad (2.24)$$

Now we are ready to add up the error terms for all of the intervals $[x_k, x_{k+1}]$:

$$\int_a^b f(x)dx = \sum_{k=1}^{M}\int_{x_{k-1}}^{x_k} f(x)dx = \sum_{k=1}^{M}\frac{h}{2}[f(x_{k-1})+f(x_k)] - \frac{h^3}{12}\sum_{k=1}^{M} f^{(2)}(c_k) \quad (2.25)$$

The first sum is the composite trapezoidal rule T(f,h). In the second term, one factor of h is replaced with its equivalent h=(b-a)/M, and the result is:

$$\int_a^b f(x)dx = T(f,h) - \frac{(b-a)h^2}{12}\left(\frac{1}{M}\sum_{k=1}^{M} f^{(2)}(c_k)\right)$$

The term in parentheses can be recognized as an average of values for the second derivative and hence is replaced by $f^{(2)}(c)$. Therefore, we have established that:

$$\int_a^b f(x)dx = T(f,h) - \frac{(b-a)f^{(2)}(c)h^2}{12}$$

and the proof is complete.

__Corollary(2.3)__: (Simpson's rule: Error analysis)

Suppose that [a,b] is subdivided into 2M subintervals $[x_k, x_{k+1}]$ of equal width h=(b-a)/(2M). The composite Simpson rule

$$S(f,h) = \frac{h}{3}(f(a)+f(b)) + \frac{2h}{3}\sum_{k=1}^{M-1} f(x_{2k}) + \frac{4h}{3}\sum_{k=1}^{M} f(x_{2k-1}) \qquad (2.26)$$

is an approximation to the integral:

$$\int_a^b f(x)dx = S(f,h) + E_S(f,h) \qquad (2.27)$$

Furthermore, if $f \in C^4[a,b]$, there exists a value c with a<c<b so that the error term $E_S(f,h)$ has the form:

$$E_S(f,h) = \frac{-(b-a)f^{(4)}(c)h^4}{180} = O(h^4) \qquad (2.28)$$

__Example(2.7)__: Consider $f(x) = \frac{1}{x}$. Investigate the error when the composite trapezoidal rule is used over [1,6] and the number of subintervals is 10.

h=(6-1)/10=0.5, since:

$$E_T(f,h) = \frac{-(b-a)f^{(2)}(c)h^2}{12} = O(h^2)$$

we first compute $f'(x) = \frac{-1}{x^2}$ and $f''(x) = \frac{2}{x^3}$,therefore:

$$f''(1) = 2, f''(2) = \frac{1}{4}, f''(6) = \frac{2}{6^3} = 0.009\,259$$

and hence f''(c)=2 and E_T(f,h)=$\frac{-(6-1)(2)(0.5)^2}{12}=\frac{-2.5}{12}=-0.208\,333$

Example(2.8): Find the number M and the step size h so that the error E_S(f,h) for the Simpson's rule is less than 5×10^{-9} for the approximation $\int_2^7 {dx}/{x}\approx S(f,h)$.

$$f(x)=\frac{1}{x}\xrightarrow{yields}f'(x)=\frac{-1}{x^2}\xrightarrow{yields}f''(x)=\frac{2}{x^3}\xrightarrow{yields}f^{(3)}(x)=\frac{-6}{x^4}\xrightarrow{yields}f^{(4)}(x)=\frac{24}{x^5}$$

the maximum value of $|f^{(4)}(x)|$ taken over [2,7] occurs at the end point x=2 and $f^{(4)}(2)=3/4$, then:

$$|E_S(f,h)|=\frac{|-(b-a)f^{(4)}(c)h^4|}{180}\leq\frac{(7-2)\frac{3}{4}h^4}{180}=\frac{h^4}{48}$$

The step size h and number M satisfy the relation h=5/(2M), and this is used in the above equation to get the relation

$$|E_S(f,h)|\leq\frac{625}{768M^4}\leq5\times10^{-9}$$

$$\xrightarrow{yields}\frac{125}{768}\times10^9\leq M^4\xrightarrow{yields}112.95\leq M$$

since M must be integer, we chose M=113

and the corresponding step size h=5/226=0.022123

Exercises:

1. Approximate the integral $\int_{-1}^{1}\frac{dx}{1+x^2}$using the composite trapezoidal rule with M=10.
2. The length of the curve y=f(x) over the interval $a\leq x\leq b$ is L=$\int_a^b\sqrt{1+(f'(x)^2}$ approximate the length of the function f(x)=x^3 over [0,1] using composite Simpsons rule with M=5.

3. Verify that the trapezoidal rule (M=1, h=1) is exact for polynomials of degree≤1 of the form $f(x)=c_1x+c_0$ over [0,1].
4. Determine the number M and the interval width h so that the composite trapezoidal rule for M subintervals can be used to compute the integral $\int_0^2 xe^{-x}dx$ with an accuracy of 5×10^{-9} .

2.4 Romberg Integration:

The discussion here is based upon the trapezium rule. Let the integration domain [a,b] be divided by three equispaced nodes $x_0=a$, $x_1=(a+b)/2$ and $x_2=b$ at interval of size h. Two successive trapezium estimates using one and two subintervals respectively are:

$$T_1 = \frac{2h}{2}[f(x_0) + f(x_1)] \ \ and \ \ T_2 = \frac{h}{2}[f(x_0) + 2f(x_1) + f(x_2)]$$

On including the truncation error for this estimate we can write:

$$I = T_1 - \frac{(2h)^2}{12} f''(x_0) - G(2h)^4 - \cdots$$

$$I = T_2 - \frac{h^2}{12} f''(x_0) - Gh^4 - \cdots$$

where G is independent of the step size h. Four times the second estimate minus the first estimate gives:

$$I = \tfrac{1}{3}[4T_2 - T_1] + 4Gh^4 + O(h^6) \qquad (2.29)$$

Taken as an estimate to I, the values $(4T_2\text{-}T_1)/3$ has leading error of $O(h^4)$. Expand this estimate:

$$I \approx \frac{1}{3}[4T_2 - T_1] = \frac{1}{3}\left[4\left\{\frac{h}{2}(f_0 + 2f_1 + f_2)\right\} - \frac{2h}{2}(f_0 + f_2)\right]$$

$$= \frac{h}{3}[f_0 + 4f_1 + f_2]$$

Shows it to be the Simpson estimate S_2 using two sub-intervals of size h=(b-a)/2.

This process can be carried out for any two trapezium estimates T_N and T_{2N} to give the more accuracy Simpson's estimate S_{2N}.

Trapezoidal	**Simpson**	
T_1		
T_2	**S_2**	
T_4	**S_4**	**In general S_{2N}=1/3{4T_{2N}-T_N}**
T_8	**S_8**	

In the same way we get:

$$I \approx \frac{1}{15}[16S_4 - S_2] + O(h^6) \qquad (2.30)$$

known as Boole's rule.

Trapezoidal	**Simpson**	**Boole's**	
T_1			
T_2	**S_2**		
T_4	**S_4**	**B_4**	**In general S_{2N}=1/3{4T_{2N}-T_N}**
T_8	**S_8**	**B_8**	**In general B_{4N}=1/15{16S_{4N}-S_{2N}}**

Example(2.9): Estimate the value of $\int_0^1 e^{sinx}dx$ using Romberg integration

N	Trapezium k=1	Simpson k=2	Boole k=3	k=4
1	1.659 888			
2	1.637 517	1.630 060		
4	1.633 211	1.631 776	1.631 891	
8	1.632 201	1.631 864	1.631 869	1.631 869

Exercises:

1. Use Romberg integration to estimate $\int_0^2 x^2 e^{-x^2} dx$ as accurately as possible, working to four decimal places.

Chapter3: Numerical Solution of Ordinary Differential Equations

3.1 Numerical Solution of a First-Order ODE

A numerical solution of a first order ODE formulated as

$$\frac{dy}{dx} = f(x,y) \; with\ the\ initial\ condition\ y(x_1) = y_1 \qquad (3.1)$$

is a set of discrete points that approximate the function y(x). When a differential equation is solved numerically, the problem statement also includes the domain of the solution. For example, a solution is required for values of the independent variable from $x = a$ to $x = b$ (the domain is [a, b]). Depending on the numerical method used to solve the equation, the number of points between a and b at which the solution is obtained can be set in advance, or it can be decided by the method. For example, the domain can be divided into N subintervals of equal width defined by N + 1 values of the independent variable from $x_1 = a$ to $x_{N+1} = b$. The solution consists of values of the dependent variable that are determined at each value of the independent variable. The solution then is a set of points $(x_1, y_1), (x_2, y_2), \ldots , (x_{N+1} , Y_{N+1})$ that define the function y(x) .

3.1.1 Overview of Numerical Methods Used/or Solving a First-Order ODE

Numerical solution is a procedure for calculating an estimate of the exact solution at a set of discrete points. The solution process is incremental, which means that it is determined in steps. It starts at the point where the initial value is given. Then, using the known solution at the first point, a solution is determined at a second nearby point. This is followed by a solution at a third point, and so on.

There are procedures with a single-step and multistep approach. In a **single-step approach**, the solution at the next point, x_{i+1}, is calculated from the already known solution at the present point, x_i. In a **multi-step approach**, the solution at x_{i+1} is calculated from the known solutions at several previous points. The idea is that the value of the function at several previous points can give a better estimate for the trend of the solution.

Also, two types of methods, explicit, and implicit, can be used for calculating the solution at each step. The difference between the methods is in the way that the solution is calculated at each step. Calculating the value of the dependent variable at the next value of the independent variable. In an **explicit formula**, the right-hand side of the equation only has known quantities. In other words, the next unknown value of the dependent variable, y_{i+1}, is calculated by evaluating an expression of the form:

$$y_{i+1} = F(x_i, x_{i+1}, y_i) \qquad (3.2)$$

where x_i, y_i, and x_{i+1} are all known quantities. In **implicit methods**, the equation used for calculating $y_{i+1}$from the known x_i, y_i, and x_{i+1} has the form:

$$y_{i+1} = F(x_i, x_{i+1}, y_{i+1}) \qquad (3.3)$$

Here, the unknown y_{i+1} appears on both sides of the equation.

3.1.2 Errors in Numerical Solution of ODEs

Two types of errors, round-off errors and truncation errors, occur when ODEs are solved numerically. Round-off errors are due to the way that computers carry out calculations. **Truncation errors** are due to the approximate nature of the method used to calculate the solution. Since the numerical solution of a differential equation is calculated in increments (steps), the truncation error at each step of the solution consists of two parts. One, called **local truncation error**, is due to the application of the numerical method in a single step. The second part, called **propagated, or accumulated, truncation error**, is due to the accumulation of local truncation errors from previous steps. Together, the two parts are the **global (total) truncation error** in the solution.

3.1.3 Single-step explicit methods

In a single-step explicit method, illustrated in Fig. 3-1,

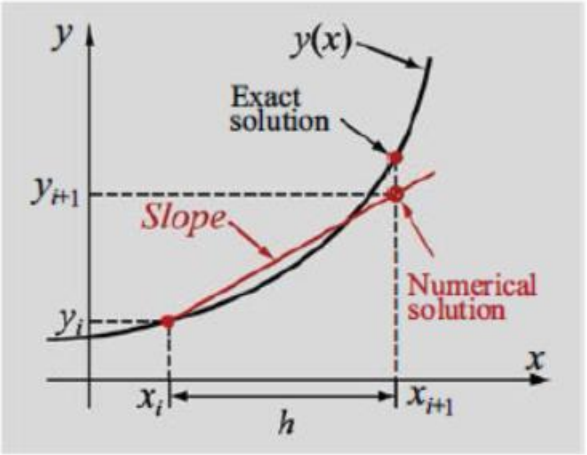


Figure 3-1: Single-step explicit methods.

The approximate numerical solution (x_{i+1}, y_{i+1}) is calculated from the known solution at point (x_i, y_i) by:

$$x_{i+1} = x_i + h \qquad (3.4)$$

$$y_{i+1} = y_i + \text{Slope}\cdot\text{h} \qquad (3.5)$$

where h is the step size, and the Slope is a constant that estimates the value of $\frac{dy}{dx}$ in the interval from $x_i$to x_{i+1}. The numerical solution starts at the point where the initial value is known. This corresponds to *i = 1* and point *(x₁, y₁)*. Then *i* is increased to *i* = 2, and the solution at the next point, *(x_2, y_2)*, is calculated by using Eqs. (3.4) and (3.5). The procedure continues with *i* = 3 and so on until the points cover the whole domain of the solution.

3.2 EULER'S METHODS

Euler's method is the simplest technique for solving a first-order ODE of the form of Eq. (3.1):

$$\frac{dy}{dx} = f(x,y)\ with\ the\ initial\ condition\ y(x_1) = y_1$$

The method can be formulated as an explicit or an implicit method.

3.2.1 Euler's Explicit Method

Euler's explicit method (also called the forward Euler method) is a single-step, numerical technique for solving a first-order ODE. The method uses Eqs. (3.4) and (3.5), where the value of the constant Slope in Eq. (3.5) is the slope of *y(x)* at point $(x_i,\ y_i)$. This slope is actually calculated from the differential equation:

$$Slope = \frac{dy}{dx}|_{x=x_i} = f(x_i, y_i) \qquad (3.6)$$

Euler's method assumes that for a short distance h near (x_i, y_i), the function *y(x)* has a constant slope equal to the slope at (x_i, y_i). With this assumption, the next point of the numerical solution (x_{i+1}, y_{i+1}) is calculated by:

$$\boxed{\begin{aligned} x_{i+1} &= x_i + h \qquad (3.7) \\ y_{i+1} &= y_i + f(x_i, y_i)h \qquad (3.8) \end{aligned}}$$

Equation (3.8) of Euler's method can be derived in several ways. Starting with the given differential equation:

$$\frac{dy}{dx} = f(x, y) \qquad (3.9)$$

An approximate solution of Eq. (3.9) can be obtained either by numerically integrating the equation or by using a finite difference approximation for the derivative.

3.2.1.1 Deriving Euler's method by using finite difference approximation for the derivative

Euler's formula, Eq. (3.8), can be derived by using an approximation for the derivative in the differential equation. The derivative $\frac{dy}{dx}$ in Eq. (3.8) can be approximated with the forward difference formula by evaluating the ODE at the point $x = x_i$:

$$\frac{dy}{dx}|_{x=x_i} \approx \frac{y_{i+1}-y_i}{x_{i+1}-x_i} = f(x_i, y_i) \qquad (3.10)$$

Solving Eq. (3.10) for y_{i+1} gives Eq. (3.8) of Euler's method. (Because the equation can be derived in this way, the method is also known as the **forward Euler method**.)

Example 3-1: Use Euler's explicit method to solve the ODE

$$\frac{dy}{dx} = -1.2y + 7e^{-0.3x}$$

from $x = 0$ to $x = 2.5$ with the initial condition $y = 3$ at $x = 0$.

(a) Solve by hand using $h = 0.5$.

(b) Write a MATLAB program in a script file that solves the equation using $h = 0.5$.

(c) Use the program from part (b) to solve the equation using h = 0.1.

In each part compare the results with the exact (analytical) solution:

$$y(x) = \frac{70}{9}e^{-0.3x} - \frac{43}{9}e^{-1.2x}$$

Solution:

(a) Solution by hand: The first point of the solution is (0, 3), which is the point where the initial condition is given. For the first point $i = 1$. The values of x and y are $x_1 = 0$ and $y_1 = 3$. The rest of the solution is determined by using Eqs. (3.7) and (3.8). In the present problem these equations have the form:

$$x_{i+1} = x_i + h = x_i + 0.5 \qquad (3.11)$$

$$y_{i+1} = y_i + f(x_i, y_i)h = y_i + (-1.2y_i + 7e^{-0.3x_i})0.5 \qquad (3.12)$$

Equations (3.11) and (3.12) are applied five times with i = 1, 2, 3, 4, and 5.

First step: For the first step $i = 1$. Equations (3.11) and (3.12) give:

$$x_2 = x_1 + h = 0 + 0.5 = 0.5$$

$$y_2 = y_1 + (-1.2y_1 + 7e^{-0.3x_1})0.5 = 4.7$$

The second point is (0.5, 4.7).

Second step: For the second step $i = 2$. Equations (3.11) and (3.12) give:

$$x_3 = x_2 + h = 0.5 + 0.5 = 1$$

$$y_3 = y_2 + (-1.2y_2 + 7e^{-0.3x_2})0.5 = 4.8924779$$

The third point is (1, 4.8924779).

Third step: For the third step $i = 3$. Equations (3.11) and (3.12) give:

$$x_4 = x_3 + h = 1 + 0.5 = 1.5$$

$$y_4 = y_3 + (-1.2y_3 + 7e^{-0.3x_3})0.5 = 4.5498549$$

The fourth point is (1.5, 4.5498549).

Fourth step: For the fourth step $i = 4$. Equations (3.11) and (3.12) give:

$$x_5 = x_4 + h = 1.5 + 0.5 = 2$$

$$y_5 = y_4 + (-1.2y_4 + 7e^{-0.3x_4})0.5 = 4.0516405$$

The fifth point is (2, 4.0516405).

Fifth step: For the fourth step $i = 5$. Equations (3.11) and (3.12) give:

$$x_6 = x_5 + h = 2 + 0.5 = 2.5$$

$$y_6 = y_5 + (-1.2y_5 + 7e^{-0.3x_5})0.5 = 3.5414969$$

The sixth point is (2.5, 3.5414969).

The values of the exact and numerical solutions, and the error, which is the difference between the two, are:

i	x_i	y_i numerical	*y*(x_i) *exact*	Error
1	0	3.0000000	3.0000000	0
2	0.5000	4.7000000	4.0722953	0.6277047
3	1.0000	4.8924779	4.3228804	0.5695975
4	1.5000	4.5498549	4.1695687	0.3802862
5	2.0000	4.0516405	3.8351047	0.2165358
6	2.5000	3.5414969	3.4360905	0.1054064

(b) To solve the ODE with MATLAB:

```
function d=euler(f,y1,a,b,n)
h=(b-a)/n;x(1)=a;y(1)=y1;
for k=1:n
    x(k+1)=x(k)+h;
    y(k+1)=y(k)+h*f(x(k),y(k));
end
d=[x' y']
```

3.2.2 Analysis of Truncation Error in Euler's Explicit Method

As mentioned in Section 3.1.2, when ODEs are solved numerically there are two sources of error, round-off and truncation. The round-off errors are due to the way that computers carry out calculations. The truncation error is due to the approximate nature of the method used for calculating the solution in each increment (step). In addition, since the numerical solution of a differential equation is calculated in increments (steps), the truncation error consists of a local truncation error and propagated truncation error. The truncation errors in Euler's explicit method are discussed in this section.

The discussion is divided into two parts. First, the **local truncation error** is analyzed, and then the results are used for determining an estimate of the **global truncation error**.

Definition 3.1: Assume that $\{(x_k,y_k),k=1,\dots,N\}$ is the set of discrete approximations and that $y=y(x)$ is the unique solution to the initial value problem. The ***global discretization error e_k*** is defined by:

$$e_k=y(x_k)-y_k \quad for\ k=1,\dots,N \tag{3.13}$$

The local discretization error $\in_{k+1}$ is defined by:

$$\in_{k+1}=y(x_{k+1})-y_k-h\emptyset(x_k,y_k) \quad for\ k=1,\dots,N-1 \tag{3.14}$$

for some function $\emptyset$ called an increment function.

Theorem 3.1: (Precision of Euler's Method)

Assume that $y(x)$ is the solution to the IVP given in (3.1).If $y(x)\in C^2[t_0,b]$ and $\{(x_k,y_k),k=1,\dots,N\}$ is the sequence of approximations generated by Euler's method, then:

$$|e_k|=|y(x_k)-y_k|=O(h) \tag{3.15}$$

$$|\in_{k+1}|=|y(x_{k+1})-y_k-hf(x_k,y_k)|=O(h^2) \tag{3.16}$$

The error at the end of the interval is called the ***final global error (FGE)***:

$$E(y(b),h)=|y(b)-y_M|=O(h) \tag{3.17}$$

3.2.3 Euler's Implicit Method

The form of Euler's implicit method is the same as the explicit scheme, except, for a short distance, h, near (x_i,y_i) the slope of the function $y(x)$ is taken to be a constant equal to the slope at the endpoint of the interval (x_{i+1},y_{i+1}). With this assumption, the next point of the numerical solution (x_{i+1},y_{i+1}) is calculated by:

$$x_{i+1}=x_i+h \tag{3.18}$$

$$y_{i+1}=y_i+f(x_{i+1},y_{i+1})h \tag{3.19}$$

Now, the unknown y_{i+1} appears on both sides of Eq. (3.19), and unless $f(x_{i+1}, y_{i+1})$depends on y_{i+1} in a simple linear or quadratic form, it is not easy or even possible to solve the equation for y_{i+1} explicitly.

3.3 MODIFIED EULER'S METHOD

The modified Euler method is a single-step, explicit, numerical technique for solving a first-order ODE. The method is a modification of Euler's explicit method. (This method is sometimes called **Heun's method**). As discussed in Section 3.2.1, the main assumption in Euler's explicit method is that in each subinterval (step) the derivative (slope) between points (x_i, y_i) and $(x_{i+1}, y_{i+1})$is constant and equal to the derivative (slope) of *y(x)* at point (x_i, y_i). This assumption is the main source of error. In the modified Euler method the slope used for calculating the value of y_{i+1} is modified to include the effect that the slope changes within the subinterval. The slope used in the modified Euler method is the average of the slope at the beginning of the interval and an estimate of the slope at the end of the interval. The slope at the beginning is given by:

$$\frac{dy}{dx}|_{x=x_i} = f(x_i, y_i) \qquad (3.20)$$

The estimate of the slope at the end of the interval is determined by first calculating an approximate value for y_{i+1} written as y_{i+1}^{Eu}using Euler's explicit method:

$$y_{i+1}^{Eu} = y_i + hf(x_i, y_i) \qquad (3.21)$$

and then estimating the slope at the end of the interval by substituting the point $\left(x_{i+1}, y_{i+1}^{Eu}\right)$ in the equation for $\frac{dy}{dx}$:

$$\frac{dy}{dx}|_{\substack{x=x_{i+1} \\ y=y_{i+1}^{Eu}}} = f(x_{i+1}, y_{i+1}^{Eu}) \qquad (3.22)$$

The modified Euler method is summarized in the following algorithm.

<u>Algorithm for the modified Euler method</u>

1. Given a solution at point (x_i, y_i), calculate the next value of the independent variable:

$$x_{i+1} = x_i + h$$

2. Calculate$f(x_i, y_i)$.
3. Estimate y_{i+1} using Euler's method:

$$y_{i+1}^{Eu} = y_i + hf(x_i, y_i)$$

4. Calculate (x_{i+1}, y_{i+1}^{Eu}) .
5. Calculate the numerical solution at $x = x_{i+1}$:

$$y_{i+1} = y_i + \frac{h}{2}\left[f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{Eu})\right]$$

Example 10-2: Use the modified Euler method to solve the ODE

$$\frac{dy}{dx} = -1.2y + 7e^{-0.3x}$$

from *x=0* to x = 2.5 with the initial condition *y(0) = 3*. Using *h = 0.5*. Compare the results with the exact (analytical) solution:

$y(x) = \frac{70}{9}e^{-0.3x} - \frac{43}{9}e^{-1.2x}$.

Solution:

The first point of the solution is (0, 3), which is the point where the initial condition is given. For the first point *i = 1*. The values of *x* and *y* are $x_1 = 0$ and $y_1 = 3$.

In the present problem these equations have the form:

$$x_{i+1} = x_i + h = x_i + 0.5$$
$$y_{i+1}^{Eu} = y_i + f(x_i, y_i)h = y_i + (-1.2y_i + 7e^{-0.3x_i})0.5$$
$$y_{i+1} = y_i + \frac{h}{2}\left[f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{Eu})\right] = y_i + \frac{0.5}{2}\left[(-1.2y_i + 7e^{-0.3x_i}) + \left(-1.2y_{i+1}^{Eu} + 7e^{-0.3x_{i+1}}\right)\right]$$

First step: For the first step *i = 1*:

$$x_2 = x_1 + h = 0 + 0.5 = 0.5$$
$$y_2^{Eu} = y_1 + (-1.2y_1 + 7e^{-0.3x_1})0.5 = 4.7$$
$$y_i + \frac{0.5}{2}[(-1.2y_1 + 7e^{-0.3x_1}) + (-1.2y_2^{Eu} + 7e^{-0.3x_2})] = 3.946238958743852$$

The second point is (0.5, 3.946238958743852).

The values of the exact and numerical solutions, and the error, which is the difference between the two, are:

i	x_i	y_i numerical	*y*(x_i) *exact*	Error
1	0	3.0000000	3.0000000	0
2	0.5000	3.946238958743852	4.0722953	0.126056374335137
3	1.0000	4.187746065761980	4.3228804	0.135134415959749
4	1.5000	4.063314737957255	4.1695687	0.106253975375624
5	2.0000	3.763482617314995	3.8351047	0.071622108811351
6	2.5000	3.393629530605291	3.4360905	0.042460997400584

Comparing the error values here with those in Example 3-1, where the problem was solved with Euler's explicit method using the same size subintervals, shows that the error with the modified Euler method is much smaller.

3.4 RUNGE-KUTTA METHODS

Runge-Kutta methods are a family of single-step, explicit, numerical techniques for solving a first-order ODE. As was stated in Section 3.1, for a subinterval (step) defined by $[x_i, x_{i+1}]$, where h = x_{i+1}-x_i, the value of $y_{i+1}$is calculated by:

$$y_{i+1} = y_i + slop.h \qquad (3.23)$$

where Slope is a constant. The value of Slope in Eq. (3.23) is obtained by considering the slope at several points within the subinterval. Various types of Runge-Kutta methods are classified according to their order. The order identifies the number of points within the sub interval that are used for determining the value of Slope in Eq. (3.23). Second order Runge-Kutta methods use the slope at two points, third-order methods use three points, and so on. The so-called classical Runge-Kutta method is of fourth order and uses four points. The order of the method is also related to the global truncation error of each method. For example, the second-order Runge-Kutta method is second-order accurate globally; that is, it has a local truncation error of $O(h^3)$ and a global truncation error of $O(h^2)$.

3.4.1 Second-Order Runge-Kutta Methods

The general form of second-order Runge-Kutta methods is:

$$\left.\begin{aligned} y_{i+1} &= y_i + \frac{h}{2}(k_1 + k_2) \\ k_1 &= f(x_i, y_i) \\ k_2 &= f(x_i + h, y_i + k_1 h) \end{aligned}\right\} \tag{3.24}$$

Example 3-3: Solving by hand a first-order ODE using the second-order Runge-Kutta method to solve the ODE

$$\frac{dy}{dx} = -1.2y + 7e^{-0.3x}$$

from $x=0$ to $x = 2.5$ with the initial condition $y(0) = 3$. Using $h = 0.5$. Compare the results with the exact (analytical) solution:

$y(x) = \frac{70}{9}e^{-0.3x} - \frac{43}{9}e^{-1.2x}$.

Solution:

The first point of the solution is (0, 3), which is the point where the initial condition is given. For the first point $i = 1$. The values of x and y are $x_1 = 0$ and $y_1 = 3$.

The rest of the solution is done by steps. In each step the next value of the independent variable is given by:

$$x_{i+1} = x_i + h = x_i + 0.5 \tag{3.25}$$

The value of the dependent variable y_{i+1} is calculated by first calculating k_1 and k_2 using :

$$\left.\begin{aligned} k_1 &= f(x_i, y_i) \\ k_2 &= f(x_i + h, y_i + k_1 h) \end{aligned}\right\} \tag{3.26}$$

and then substituting the k's in :

$$y_{i+1} = y_i + \frac{h}{2}(k_1 + k_2) \tag{3.27}$$

First step: In the first step $i = 1$. Equations (3. 25)-(3. 27) give:

$x_2 = x_1 + 0.5 = 0.5$

$k_1 = f(x_1, y_1)$=$f(0,3) = -1.2(3) + 7e^{-0.3(0)} = 3.4$

$k_2 = f(x_1 + h, y_1 + k_1 h)$= $f\big(0 + 0.5, 3 + 3.4(0.5)\big) = f(0.5, 1.7)$

$= -1.2(1.7) + 7e^{-0.3(0.5)} = 0.384955834975405$

$y_2 = y_1 + \frac{h}{2}(k_1 + k_2)$ =$3 + \frac{0.5}{2}(3.4 + 0.384955834975405) = 3.946238958743852$

Second step: In the first step $i = 2$. Equations (3. 25)-(3. 27) give:

$x_3 = x_2 + 0.5 = 1.0$

$k_1 = f(x_2, y_2)$=$f(0.5, 3.946238958743852)$

$= -1.2(3.946238958743852) + 7e^{-0.3(0.5)} = 1.289469084482783$

$k_2 = f(x_2 + h, y_2 + k_1 h)$

= $f\big(0.5 + 0.5, 3.946238958743852 + 1.289469084482783(0.5)\big)$

=-0.323440656410266

$y_3 = y_2 + \frac{h}{2}(k_1 + k_2)$ =4.187746065761980

Third step:

k1 = 0.160432265857648
k2 = -0.658157577076552
y_4 =4.063314737957255

Fourth step:

k1 = -0.412580624196292
k2 = -0.786747858372744
$y_5 = 3.763482617314995$

Fifth step:

k1 = -0.674497688119808
k2 = -0.804914658719007
$y_6 = 3.393629530605291$

The values of the exact and numerical solutions, and the error, which is the difference between the two, are:

i	x_i	y_i numerical	*y*(x_i) *exact*	Error
1	0	3.0000000	3.0000000	0
2	0.5000	3.946238958743852	4.0722953	0.126056374335137
3	1.0000	4.187746065761980	4.3228804	0.135134415959749
4	1.5000	4.063314737957255	4.1695687	0.106253975375624
5	2.0000	3.763482617314995	3.8351047	0.071622108811351
6	2.5000	3.393629530605291	3.4360905	0.042460997400584

The solution obtained is obviously identical (except for rounding errors) to the solution in example 3-2.

3.4.2 Fourth-Order Runge-Kutta Methods

The general form of classical fourth-order Runge-Kutta method is:

$$\left.\begin{aligned} & y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ & with \\ & \qquad k_1 = f(x_i, y_i) \\ & \qquad k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{hk_1}{2}\right) \\ & \qquad k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{hk_2}{2}\right) \\ & \qquad k_4 = f(x_i + h, y_i + hk_3) \end{aligned}\right\} \qquad (3.28)$$

Example 3-4: Solving by hand a first-order ODE using the fourth-order Runge-Kutta method to solve the ODE

$$\frac{dy}{dx} = -1.2y + 7e^{-0.3x}$$

from *x=0* to *x* = 2.5 with the initial condition *y(0) = 3*. Using *h = 0.5*. Compare the results with the exact (analytical) solution:

$y(x) = \frac{70}{9}e^{-0.3x} - \frac{43}{9}e^{-1.2x}$.

Solution:

First step:

$k_1 = f(x_1, y_1) = f(0,3)$ =3.40
$k_2 = f\left(x_1 + \frac{h}{2}, y_1 + \frac{hk_1}{2}\right)$ =1.874204404299870
$k_3 = f\left(x_1 + \frac{h}{2}, y_1 + \frac{hk_2}{2}\right)$ =2.331943083009909
$k_4 = f(x_1 + h, y_1 + hk_3)$ = 1.025789985169459
$y_2 = y_2 + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$ =4.069840413315752

Second step:

k1 = 1.141147338996503
k2 = 0.363460833637786
k3 = 0.596766785245403
k4 = -0.056141022354118
y_3 =4.320295542849815

Third step:

k1 =0.001372893352247
k2 =-0.373741567888647
k3 =-0.261207229516379
k4 = -0.564233252357536
y_4 =4.167565713365203

Fourth step:

k1 = -0.537681794685830
k2 =-0.698886767064788
k3 =-0.650525275351102
k4 =-0.769082238169397
y_5 =3.833766703557953

Fifth step:

k1 =-0.758838591611358
k2 =-0.808773522533291
k3 =-0.793793043256712
k4 =-0.817678349128413
y_6 =3.435295864197971

The values of the exact and numerical solutions, and the error, which is the difference between the two, are:

i	x_i	y_i numerical	$y(x_i)$ *exact*	Error
1	0	3.000000000000000	3.0000000	0
2	0.5000	4.069840413315752	4.0722953	0.002454919763237
3	1.0000	4.320295542849815	4.3228804	0.002584938871915
4	1.5000	4.167565713365203	4.1695687	0.002002999967676
5	2.0000	3.833766703557953	3.8351047	0.001338022568394
6	2.5000	3.435295864197971	3.4360905	0.000794663807904

3.5 Predictor-Corrector Methods

Predictor-corrector methods refer to a family of schemes for solving ordinary differential equations using two formulae: ***predictor* and *corrector* formula**. In predictor-corrector methods, four prior values are required to find the value of y at x_n. Predictor-corrector methods have the advantage of giving an estimate of error from successive approximations to y_n. The predictor is an explicit formula and is used first to determine an estimate of the solution y_{n+1}. The value y_{n+1} is calculated from the known solution at the previous point (x_n, y_n) using single-step method or several previous points (multi-step methods). If x_n and x_{n+1} are two consecutive mesh points such that :

$$x_{i+1} = x_i + h$$

then in Euler's method, we have:

$$y_{i+1} = y_i + h\, f\,(x_i, y_i),\ i = 0, 1, 2, 3, \ldots \qquad (3.29)$$

Once an estimate of y_{i+1} is found, the corrector is applied. The corrector uses the estimated value of y_{i+1} on the right-hand side of an otherwise implicit formula for computing a new, more accurate value for y_{n+1} on the left-hand side. The modified Euler's method gives as:

$$y_{i+1} = y_i + \frac{h}{2}[f(x_i, y_i) + f(x_{i+1}, y_{i+1})] \qquad (3.30)$$

The value of y_{i+1} is first estimated by Eq.(3.29) and then utilized in the right-hand side of Eq.(3.30) resulting in a better approximation of y_{i+1}. The value of y_{i+1} thus obtained is again substituted in Eq.(3.30) to find a still better approximation of y_{i+1}. This procedure is repeated until two consecutive iterated values of y_{i+1} are very close. Here, the corrector equation (3.30) which is an implicit equation is being used in an *explicit* manner since no solution of a non-linear equation is required.

In addition, the application of corrector can be repeated several times such that the new value of y_{i+1} is substituted back on the right-hand side of the corrector formula to obtain a more refined value for y_{i+1}. The technique of refining an initially crude estimate of y_{i+1} by means of a more accurate formula is known as ***predictor-corrector* method**. Equation (2.29) is called the ***predictor*** and Eq. (3.30) is called the ***corrector*** of y_{n+1}.

Example 3.5: Use the PC method on (2, 3) with h = 0.1 for the initial value problem

$$\frac{dy}{dx} = -xy^2,\ y(2) = 1.$$

Exact solution is $y(x) = \dfrac{2}{x^2 - 2}$.

Solution:

First, we use Euler method:

$y_1 = y_0 + hf(x_0, y_0)=1+0.1(-2(1)^2)=0.8$

Then, we use modified Euler:

$y_1 = y_0 + \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1)]=1+0.1/2*[-2*1^2+(-2.1)*(0.8)^2]=0.8328$

Containing in the same manner, we obtain:

x_i	y_i	$Y(x_i)$
2	1.000000000000000	1.000000000000000
2.1	0.832800000000000	0.829875518672199
2.2	0.708036878443888	0.704225352112676
2.3	0.611802381778826	0.607902735562310
2.4	0.535592749372665	0.531914893617021
2.5	0.473938067466517	0.470588235294118
2.6	0.423170282558423	0.420168067226891
2.7	0.380742913556783	0.378071833648393
2.8	0.344835715939071	0.342465753424658
2.9	0.314114751637895	0.312012480499220
3.0	0.287581256501905	0.285714285714286

Example 3.6: Approximate the *y* value at *x* = 0.4 of the following differential equation:

$$\frac{dy}{dx}=\frac{1}{2}y, y(0)=1 \text{ and } 0\le x\le 1.$$

using the PC method with h=0.1.

Solution:

x_i	y_i
0	1.000000000000000
0.1	1.051250000000000
0.2	1.105126562500000
0.3	1.161764298828125
0.4	1.221304719143066

3.6 Higher-Order Differential Equations:

Higher-order differential equations involve the higher derivatives x''(t), x'''(t), and so on. They arise in mathematical models for problems in physics and engineering. By solving for the second derivative, we can write a second-order initial value problem in the form:

$$x''(t)=f(t,x(t),x'(t)) \text{ with } x(t_0)=x_0 \text{ and } x'(t_0)=y_0 \qquad (3.31)$$

The second-order differential equation can be reformulated as a system of two first-order equations if we use the substitution:

$$x'(t)=y(t) \qquad (3.32)$$

Then $x''(t)=y'(t)$ and the differential equation in (3.31) becomes a system:

$$\frac{dx}{dt} = y$$

$$\frac{dy}{dt} = f(t,x,y) \qquad with \begin{cases} x(t_0) = x_0 \\ y(t_0) = y_0 \end{cases} \tag{3.33}$$

A numerical procedure such as Rung-Kutta method can be used to solve (3.33) and will generate two sequences $\{x_k\}$ and $\{y_k\}$. The first sequence is the numerical solution to (3.31).

Now, consider RK2 for the system of two differential equation :

x'(t)=f(t,x,y)

y'(t)=g(t,x,y)

as follows:

$x_{k+1}=x_k+1/2(k_1+k_2)$, $y_{k+1}=y_k+1/2(p_1+p_2)$

where $k_1=hf(t_k,x_k,y_k)$, $p_1=hg(t_k,x_k,y_k)$

and $k_2=hf(t_k+h,x_k+k_1,y_k+p_1)$, $p_2=hg(t_k+h,x_k+k_1,y_k+p_1)$.

Example 3.7: Consider the second-order IVP

x"(t)+4x'(t)+5x(t)=0 with x(0)=3 and x'(0)=-5

(a) Write down the equivalent system of two first-order equation.
(b) Use The RK2 method to solve the reformulated problem over [0,1] using M=5.
(c) Compare the numerical solution with the true solution $x(t)=3e^{-2t}\cos(t)+e^{-2t}\sin(t)$.

First assume x'(t)=y(t) then x"(t)=y'(t) and we have:

x'(t)=y(t)

y'(t)=-4y(t)-5x(t) with x(0)=3 and y(0)=-5, then h=(1-0)/5=0.2

t_k	x_k	$x(t_k)$
0	3	3
0.2		
0.4		
0.6		
0.8		
1		

Exercises:

Solve the system x'=3x-y, y'=4x-y with x(0)=0.2 and y(0)=0.5 using RK2 with h=0.5 in [0,1].

3.7 Boundary Value Problems:

Another type of differential equation has the form:

$$x''=f(t,x,x') \quad \text{for } a\leq t\leq b \qquad (3.34)$$

with the boundary conditions

$$x(a)=\alpha \text{ and } x(b)=\beta \qquad (3.35)$$

This is called ***a boundary value problem (BVP)***.

Finite-difference Method:

Methods involving difference quotient approximations for derivatives can be used for solving second-order BVP. Consider the linear equation:

$$x''=p(t)x'(t)+q(t)x(t)+r(t) \qquad (3.36)$$

over [a,b] with x(a)= α and x(b)= β. Form a partition of [a,b] using the points $a=t_0<t_1<\ldots<t_N=b$, where h=(b-a)/N and $t_j=a+jh$ for j=0,1,…N. The central-difference formulas discussed in chapter two are used to approximate the derivatives:

$$x'(t_j)=\frac{x(t_{j+1})-x(t_{j-1})}{2h}+O(h^2) \qquad (3.37)$$

$$x''(t_j)=\frac{x(t_{j+1})-2x(t_j)-x(t_{j-1})}{h^2}+O(h^2) \qquad (3.38)$$

To start derivation, we replace each term $x(t_j)$ on the right side of (3.37) and (3.38) with x_j and the resulting equations are substituted into (3.36), to obtain the relation:

$$\frac{x_{j+1}-2x_j+x_{j-1}}{h^2}=p_j\left(\frac{x_{j+1}-x_{j-1}}{2h}\right)+q_jx_j+r_j \qquad (3.39)$$

which is used to compute numerical approximation to the differential equation(3.36). This is carried out by multiplying each side of (3.39) by h^2 and then collecting terms involving x_{j-1}, x_j and x_{j+1} and arranging them in a system of linear equations:

$$\left(\frac{-h}{2}p_j - 1\right)x_{j-1} + \left(2 + h^2 q_j\right)x_j + \left(\frac{h}{2}p_j - 1\right)x_{j+1} = -h^2 r_j \qquad (3.40)$$

for j=1,2,…,N-1, where $x_0 = \alpha\ and\ x_N = \beta$.

Example 3.8 Solve the boundary value problem

$$x''(t) = \frac{2t}{1+t^2}x'(t) - \frac{2}{1+t^2}x(t) + 1$$

with x(0)=1.25 and x(4)=-0.95 over the interval [0,4] with h=1.

since h=1 we get N=4 and t_0=0, t_1=1, t_2=2, t_3=3 and t_4=4

In the same way:

$$\frac{x_{j+1} - 2x_j + x_{j-1}}{h^2} = \frac{2t_j}{1+t_j^2}\left(\frac{x_{j+1} - x_{j-1}}{2h}\right) - \frac{2}{1+t_j^2}x_j + 1$$

then, we get:

$$\left(-\frac{h}{2}\frac{2t_j}{1+t_j^2} - 1\right)x_{j-1} + \left(2 - \frac{2h^2}{1+t_j^2}\right)x_j + \left(\frac{h}{2}\frac{2t_j}{1+t_j^2} - 1\right)x_{j+1} = -h^2$$

$$\left(-\frac{ht_j}{1+t_j^2} - 1\right)x_{j-1} + \left(2 - \frac{2h^2}{1+t_j^2}\right)x_j + \left(\frac{ht_j}{1+t_j^2} - 1\right)x_{j+1} = -h^2$$

for j=1,2,3 and x_0=1.25, x_4=-0.95

so for j=1, we get

$$\left(-\frac{ht_1}{1+t_1^2} - 1\right)x_0 + \left(2 - \frac{2h^2}{1+t_1^2}\right)x_1 + \left(\frac{ht_1}{1+t_1^2} - 1\right)x_2 = -h^2$$

for j=2

$$\left(-\frac{ht_2}{1+t_2^2} - 1\right)x_1 + \left(2 - \frac{2h^2}{1+t_2^2}\right)x_2 + \left(\frac{ht_2}{1+t_2^2} - 1\right)x_3 = -h^2$$

and for j=3

$$\left(-\frac{ht_3}{1+t_3^2} - 1\right)x_2 + \left(2 - \frac{2h^2}{1+t_3^2}\right)x_3 + \left(\frac{ht_3}{1+t_3^2} - 1\right)x_4 = -h^2$$

therefore, we hence the algebraic system of three equations

$$\left.\begin{array}{c}\left(2-\frac{2}{1+1}\right)x_1+\left(\frac{1}{1+1}-1\right)x_2=-1-\left(-\frac{1}{1+1}-1\right)(1.25)\\ \left(-\frac{2}{1+4}-1\right)x_1+\left(2-\frac{2}{1+4}\right)x_2+\left(\frac{2}{1+4}-1\right)x_3=-1\\ \left(-\frac{3}{1+9}-1\right)x_2+\left(2-\frac{2}{1+9}\right)x_3=-1-\left(\frac{3}{1+9}-1\right)(-0.95)\end{array}\right\}$$

$$\left.\begin{array}{c}x_1-\frac{1}{2}x_2=-1+\frac{3}{2}(1.25)\\ -\frac{7}{5}x_1+\frac{8}{5}x_2-\frac{3}{5}x_3=-1\\ -\frac{13}{10}x_2+\frac{18}{10}x_3=-1+\frac{7}{10}(-0.95)\end{array}\right\}$$

then after solving this system, we obtain:

x_1=0.52143, x_2-0.70714and x_3=-1.4357

Problems:

1. Consider the following first-order ODE:

$$\frac{dy}{dx}=x^2/y\ from\ x=0\ to\ x=2.1\ with\ y(0)=2$$

(a) Solve with Euler's explicit method using h = 0.7.
(b) Solve with the modified Euler method using h = 0.7.
(c) Solve with the classical fourth-order Runge-Kutta method using h = 0.7.

The analytical solution of the ODE is $y=\sqrt{\frac{2x^3}{3}+4}$. In each part, calculate the error between the true solution and the numerical solution at the points where the numerical solution is determined.

2. Write the following second-order ODE as a system of two first-order ODEs:

$$\frac{d^2y}{dt^2}+5\left(\frac{dy}{dt}\right)^2-6y+e^{sint}=0$$

3. Consider the following second-order ODE:

$$\frac{d^2y}{dx^2}+x\frac{dy}{dx}+y=2xy\quad for\ 0\le x\le 1\ ,with\ y(0)=1\ and\ y(1)=1$$

Using the difference formulas for approximating the derivatives, discretize the ODE (rewrite the equation in a form suitable for solution with the finite difference method).

Chapter 4: Numerical Solution of Partial Differential Equations

4.1 Classification of Partial Differential Equations:

A partial differential equation (PDE) is an equation that involves an unknown function (the dependent variable) and some of its partial derivatives for two or more independent variables. The classification of PDEs is important for the numerical solution you choose. Consider the general, second-order, linear partial differential equation in two variables :

$$A(x, y)U_{xx} + 2B(x, y)U_{xy} + C(x, y)U_{yy} = F(x,y ,U_x, U_y, U) \quad (4.1)$$

4.1.1 Elliptic

$$AC > B^2$$

For example, Laplace's equation:

$$U_{xx} + U_{yy} = 0$$

$A = C = 1, B = 0$

4.1.2 Hyperbolic

$$AC < B^2$$

For example, the 1-D wave equation:

$$U_{xx} = \frac{1}{c^2}U_{tt}$$

A = 1, C = $\frac{1}{c^2}$, B = 0

4.1.3 Parabolic

$$AC = B^2$$

For example, the heat or diffusion Equation

$$U_t = Uxx$$

$A = 1;B = C = 0$

4.2 Finite Difference Solution of Partial Differential Equations:

4.2.1 Parabolic Equations

Consider the boundary-initial value problem (BIVP):

$$\left.\begin{array}{c} \boldsymbol{u_{xx} = \frac{1}{c}u_t \ , u = u(x,t), 0 < x < 1 \, , t > 0} \\ \boldsymbol{u(0,t) = u(1,t) = 0 \quad (boundary\ conditions)} \\ \boldsymbol{u(x,0) = f(x)\ (initial\ condition)} \end{array}\right\} \qquad (4.2)$$

Where c is a constant, this problem represents transient heat conduction in a rod with the ends held at zero temperature and an initial temperature profile *f(x).*

To solve this problem numerically, we discretize *x* and *t* such that:

$$x_i = i * h \ , i = 0,1,2, \dots$$
$$t_j = jk \ , j = 0,1,2, ..$$

4.2.1.1 Explicit Finite Difference Method

Let u_{ij} be the numerical approximation to $u(x_i \ , t_j)$. We approximate u_t with the finite forward difference:

$$u_t \approx \frac{u_{i,j+1} - u_{i,j}}{k} \qquad (4.3)$$

and u_{xx} with the central finite difference:

$$u_{xx} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} \qquad (4.4)$$

The finite difference approximation to the PDE is then:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} = \frac{\mathbf{1}}{\boldsymbol{c}} \frac{u_{i,j+1} - u_{i,j}}{k} \qquad (4.5)$$

Define the parameter *r* as

$$r = \frac{ck}{h^2}$$

in which case Eq. 4.5 becomes:

$$r\left(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}\right) = \left(u_{i,j+1} - u_{i,j}\right)$$

therefore,

$$u_{i,j+1} = ru_{i+1,j} + (1 - 2r)u_{i,j} + ru_{i-1,j} \qquad (4.6)$$

The domain of the problem and the mesh are illustrated in Fig. 4.1.

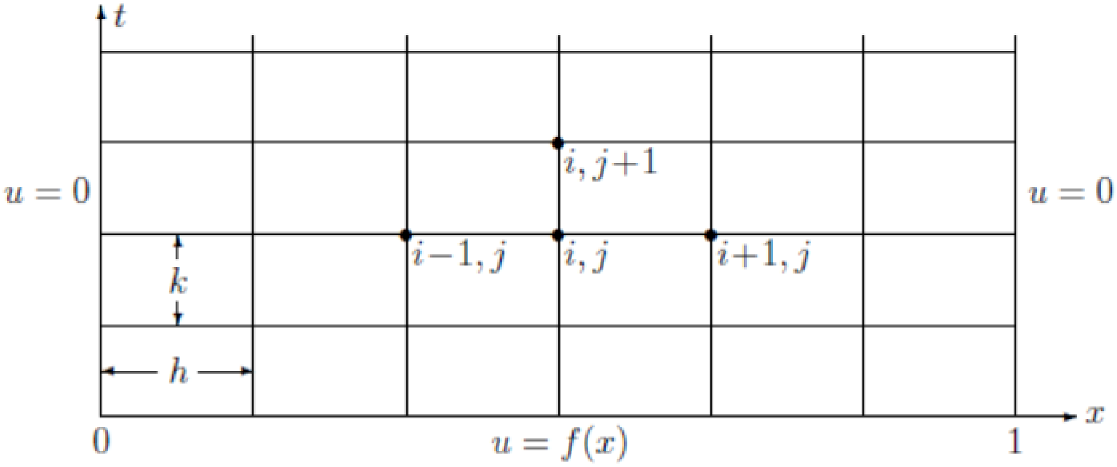


Figure 4.1: Mesh for 1-D Heat Equation.

Eq. 4.6 is a recursive relationship giving u in a given row (time) in terms of three consecutive values of u in the row below (one time step earlier). This equation is an explicit formula since one unknown value can be found directly in terms of several other known values.

We can write out the matrix system of equations we will solve numerically for the temperature *u*. Suppose we use five grid points $x_0 \ , \ x_1 \ , x_2 \ , x_3 \ and \ x_4$.

Now, for *i=1* eq. (4.6) becomes:

$$u_{1,j+1} = ru_{2,j} + (1-2r)u_{1,j} + ru_{0,j}$$

and for *i=2* eq. (4.6) becomes:

$$u_{2,j+1} = ru_{3,j} + (1-2r)u_{2,j} + ru_{1,j}$$

and for *i=3* eq. (4.6) becomes:

$$u_{3,j+1} = ru_{4,j} + (1-2r)u_{3,j} + ru_{2,j}$$

Using boundary condition in Eq. (4.2), we get:

$$u_{1,j+1} = ru_{2,j} + (1-2r)u_{1,j}$$
$$u_{2,j+1} = ru_{3,j} + (1-2r)u_{2,j} + ru_{1,j}$$
$$u_{3,j+1} = (1-2r)u_{3,j} + ru_{2,j}$$

The equation above in matrix form becomes:

$$\begin{bmatrix} u_{1,j+1} \\ u_{2,j+1} \\ u_{3,j+1} \end{bmatrix} = \begin{bmatrix} 1-2r & r & 0 \\ r & 1-2r & r \\ 0 & r & 1-2r \end{bmatrix} \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ u_{3,j} \end{bmatrix} \qquad (4.7)$$

where

$$r = \frac{ck}{h^2}$$

Now, for the system of eq's (4.7), substitute j=0,1,2:

for j=0

$$\begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \end{bmatrix} = \begin{bmatrix} 1-2r & r & 0 \\ r & 1-2r & r \\ 0 & r & 1-2r \end{bmatrix} \begin{bmatrix} u_{1,0} \\ u_{2,0} \\ u_{3,0} \end{bmatrix}$$

where $u_{k,0} = u(x_k, 0) = f(x_k)$ (by using initial condition)

for j=1

$$\begin{bmatrix} u_{1,2} \\ u_{2,2} \\ u_{3,2} \end{bmatrix} = \begin{bmatrix} 1-2r & r & 0 \\ r & 1-2r & r \\ 0 & r & 1-2r \end{bmatrix} \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \end{bmatrix}$$

for j=2

$$\begin{bmatrix} u_{1,3} \\ u_{2,3} \\ u_{3,3} \end{bmatrix} = \begin{bmatrix} 1-2r & r & 0 \\ r & 1-2r & r \\ 0 & r & 1-2r \end{bmatrix} \begin{bmatrix} u_{1,2} \\ u_{2,2} \\ u_{3,2} \end{bmatrix}$$

Chapter 5: Numerical Solution of Integral Equations

5.1 Classification of Integral Equations:

An integral equation is an equation in which the unknown function u(x) appears under an integral sign. The most general linear integral equation in u(x) can be presented as:

$$h(x)u(x) = f(x) + \int_a^{b(x)} k(x,t)u(t)dt \qquad (5.1)$$

where k(x,t) is a function of two variables called the **kernel** of the integral equation.

This equation is called a ***Volterra integral equation*** when b(x)=x,

$$h(x)u(x) = f(x) + \int_a^{x} k(x,t)u(t)dt \qquad (5.2)$$

when h(x)=0 it is called a ***Volterra equation of the first kind***,

$$-f(x) = \int_a^{x} k(x,t)u(t)dt \qquad (5.3)$$

and is called a ***Volterra equation of the second kind*** when h(x)=1,

$$u(x) = f(x) + \int_a^{x} k(x,t)u(t)dt \qquad \ldots(5.4)$$

The integral equation (5.1) is called a ***Fredholm integral equation*** when b(x)=b, where b constant,

$$h(x)u(x) = f(x) + \int_a^{b} k(x,t)u(t)dt \qquad \ldots(5.5)$$

It is also called a ***Fredholm equation of the first and second kinds*** when h(x)=0 and h(x)=1, respectively:

$$-f(x) = \int_a^{b} k(x,t)u(t)dt \qquad \ldots(5.6)$$

$$u(x) = f(x) + \int_a^{b} k(x,t)u(t)dt \qquad \ldots(5.7)$$

5.2 Numerical Solution of Volterra Integral Equations:

Let us consider the Volterra equation of the second kind:

$$u(x) = f(x) + \int_a^{x} k(x,t)u(t)dt$$

we will subdivide the interval of integration (a,x) into n equal subintervals of width h=(x_n-a)/n, n≥1, where x_n is the end point we choose for x, we shall set t_0=a and t_j=a+jh. Note that the particular value u(x_0)=f(a), so if we use the trapezoidal rule with n subintervals to approximate the integral in the Volterra integral equation of the second kind (5.4), we have:

$$\int_a^x \boldsymbol{k(x,t)u(t)dt} \approx \frac{h}{2}\begin{bmatrix} \boldsymbol{k(x,t_0)u(t_0) + 2k(x,t_1)u(t_1) + \cdots + 2k(x,t_{n-1})u(t_{n-1})} \\ \boldsymbol{+k(x,t_n)u(t_n)} \end{bmatrix}$$

(5.8)

and the integral equation (5.4) is then approximated by the sum:

$$u(x) = f(x) + \frac{h}{2}\left[k(x,t_0)u(t_0) + 2\sum_{j=1}^{n-1} k(x,t_j)u(t_j) + k(x,t_n)u(t_n)\right] \qquad (5.9)$$

If we consider n+1 sample values of *u(x), u(x_i),i=0,1,…,n,* equation (5.9) will become a set of n+1 equations in *u(x_i)* (or *u_i*)[note that *u(x_0)=f(x_0)* since the integral in (5.4) vanishes for *x=x_0=a*].

$$\left.\begin{array}{c} u_0 = f_0 \\ u_i = f_i + \frac{h}{2}\left[k_{i0}u_0 + 2\sum_{j=1}^{m-1} k_{i,j}u_j + k_{i,m}u_m\right], \\ i = 1,2,\dots,n, k_{ij} = k(x_i,t_j),\ \ j \le i \end{array}\right\} \qquad (5.10)$$

which are n+1 equations in *u_i*, the approximation to the solution *u(x)* of (5.4) at *x_i=a+ih* for *i=0,1,…,n*.

Example 5.1: Use trapezoidal method to find an approximate values to the solution for the following Volterra integral equation $u(x) = x - \int_0^x (x-t)u(t)dt$ at x=0,1,2,3,and 4.

Here, f(x)=x, k(x,t)=t-x for t≤x=4 and is zero for t>x=4, and a=0 with u(0)=0. We also have n=4 and hence h=(4-0)/4=1. So using (5.10) to obtain:

u_0=f_0=0

u_1=f_1+$\frac{h}{2}[k_{10}u_0 + k_{11}u_1]$=1+$\frac{1}{2}[(0-1)(0) + (1-1)u_1]$=1

u_2=f_2+$\frac{h}{2}[k_{20}u_0 + 2k_{21}u_1 + k_{22}u_2]$

$= 2 + \frac{1}{2}[(0-2)(0) + 2(1-2)(1) + (2-2)u_2]$=1 u3=f3+$\frac{h}{2}[k_{30}u_0+2k_{31}u_1+2k_{32}u_2+k_{33}u_3]=3+\frac{1}{2}[(0-3)(0)+2(1-3)(1)+2(2-3)(1)+(3-3)u_3]=3+\frac{1}{2}[-4-2]=0$

$u_4=f_4+\frac{h}{2}[k_{40}u_0+2k_{41}u_1+2k_{42}u_2+2k_{43}u_3+k_{44}u_4]=4+\frac{1}{2}[(0-4)(0)+2(1-4)(1)+2(2-4)(1)+2(3-4)(0)+(4-4)u_4]=4+\frac{1}{2}[-6-4]=-1$

x_k	0	1	2	3	4
u_k	0	1	1	0	-1

5.3 Numerical Solution of Fredholm Integral Equations:

Let us consider the Fredholm equation of the second kind:

$$u(x)=f(x)+\int_a^b k(x,t)u(t)dt \qquad (5.11)$$

we will subdivide the interval of integration (a,b) into n equal subintervals of width h=(b-a)/n, n≥1, we shall set t_0=a,t_n=b and t_j=a+jh. Note that the particular value , so if we use the trapezoidal rule with n subintervals to approximate the integral in the Fredholm integral equation of the second kind (5.11), we have:

$$\boldsymbol{\int_a^b k(x,t)u(t)dt \approx \frac{h}{2}\begin{bmatrix} k(x,t_0)u(t_0)+2k(x,t_1)u(t_1)+\cdots+2k(x,t_{n-1})u(t_{n-1}) \\ +k(x,t_n)u(t_n) \end{bmatrix}}$$

(5.12)

and the integral equation (5.11) is then approximated by the sum:

$$u(x)=f(x)+\frac{h}{2}\left[k(x,t_0)u(t_0)+2\sum_{j=1}^{n-1}k(x,t_j)u(t_j)+k(x,t_n)u(t_n)\right] \qquad (5.13)$$

If we consider n+1 sample values of *u(x), u(x*$_i$*),i=0,1,…,n*, equation (5.13) will become a set of n+1 equations in *u(x*$_i$*)* (or *u*$_i$).

$$\left.\begin{array}{l} u_i=f_i+\frac{h}{2}\left[k_{i0}u_0+2\sum_{j=1}^{m-1}k_{i,j}u_j+k_{i,m}u_m\right], \\ \qquad i=1,2,\dots,n, k_{ij}=k(x_i,t_j),\ \ j\le i \end{array}\right\} \qquad (5.14)$$

which are n+1 equations in *u*$_i$, the approximation to the solution *u(x)* of (5.11) at *x*$_i$*=a+ih* for *i=0,1,…,n*.

Example 5.2*:* Use trapezoidal method to find an approximate values to the solution for the integral equation u(x)=$x^2+\frac{1}{4}-\frac{1}{3}x+\int_0^1(x-t)u(t)dt$ with h=0.25 notice that the real solution is u(x)=x^2

We have f(x)= $x^2+\frac{1}{4}-\frac{1}{3}x$ and k(x,t)=x-t.

Since h=0.25, we have $x_0=t_0=0, x_1=t_1=0.25, x_2=t_2=0.5$, $x_3=t_3=0.75$ and $x_4=t_4=1$

for i=0,1,2,3 and 4, we have:

$$u_0=f_0+\frac{h}{2}[k_{00}u_0+2k_{01}u_1+2k_{02}u_2+2k_{03}u_3+k_{04}u_4]$$

$$u_1=f_1+\frac{h}{2}[k_{10}u_0+2k_{11}u_1+2k_{12}u_2+2k_{13}u_3+k_{14}u_4]$$

$$u_2=f_2+\frac{h}{2}[k_{20}u_0+2k_{21}u_1+2k_{22}u_2+2k_{23}u_3+k_{24}u_4]$$

$$u_3=f_3+\frac{h}{2}[k_{30}u_0+2k_{31}u_1+2k_{32}u_2+2k_{33}u_3+k_{34}u_4]$$

$$u_4=f_4+\frac{h}{2}[k_{40}u_0+2k_{41}u_1+2k_{42}u_2+2k_{43}u_3+k_{44}u_4]$$

therefore, we hence:

$$u_0=0.25+\frac{0.25}{2}[(0-0)u_0+2(0-0.25)u_1+2(0-0.5)u_2+2(0-0.75)u_3+(0-1)u_4]$$

$$u_1=0.22917+\frac{0.25}{2}[(0.25-0)u_0+2(0.25-0.25)u_1+2(0.25-0.5)u_2+2(0.25-0.75)u_3+(0.25-1)u_4]$$

$$u_2=0.33333+\frac{0.25}{2}[(0.5-0)u_0+2(0.5-0.25)u_1+2(0.5-0.5)u_2+2(0.5-0.75)u_3+(0.5-1)u_4]$$

$$u_3 = 0.5625 + \frac{0.25}{2}[(0.75-0)u_0 + 2(0.75-0.25)u_1 + 2(0.75-0.5)u_2 + 2(0.75-0.75)u_3 + (0.75-1)u_4]$$

$$u_4 = 0.91667 + \frac{0.25}{2}[(1-0)u_0 + 2(1-0.25)u_1 + 2(1-0.5)u_2 + 2(1-0.75)u_3 + (1-1)u_4]$$

then,

$8u_0 + 0.5u_1 + u_2 + 1.5u_3 + u_4 = 2$

$-0.25u_0+8u_1+0.5u_2+u_3+0.75u_4=1.8333$

$-0.5u_0-0.5u_1+8u_2+0.5u_3+0.5u_4=2.6667$

$-0.75u_0-u_1-0.5u_2+8u_3+0.25u_4=4.5$

$-u_0-1.5u_1-u_2-0.5u_3+8u_4=7.3333$

solving this system, we get:

$u=[-0.010417 \quad 0.052083 \quad 0.23958 \quad 0.55208 \quad 0.98958]^T$

x_k	u_k	$u(x_k)$
0	-0.010417	0
0.25	0.052083	0.0625
0.5	0.23958	0.25
0.75	0.55208	0.5625
1	0.98958	1

Exercise:

1. Use trapezoidal method to find an approximate values to the solution for the integral equation u(x)=$x-\frac{x^3}{3}+\int_0^x tu(t)dt$, x∈[0,1], with h=0.25.(note that u(x)=x)
2. Use trapezoidal method to find an approximate values to the solution for the integral equation u(x)=$e^x - xe^1 + x + \int_0^1 xu(t)dt$, with h=0.5 (note that u(x)=e^x).

Chapter 6: Eigenvalues and Eigenvectors

Definition 6.1: If A is an n×n real matrix, then its n eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ are the real and complex roots of the characteristic polynomial

$$p(\lambda) = \det(a - \lambda I) \qquad (6.1)$$

Definition 6.2: If λ is an eigenvalue of A and the nonzero vector V has the property that

$$AV = \lambda V \qquad (6.2)$$

then V is called an eigenvector of A corresponding to the eigenvalue λ.

Example 6.1: Find the eigenvalues λ_j for the matrix

$$A=\begin{bmatrix} 3 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 3 \end{bmatrix}$$

The characteristic equation det(A- λI)=0 is

$$\begin{vmatrix} 3-\lambda & -1 & 0 \\ -1 & 2-\lambda & -1 \\ 0 & -1 & 3-\lambda \end{vmatrix} = -\lambda^3 + 8\lambda^2 - 19\lambda + 12 = 0$$

which can be written as $-(\lambda-1)(\lambda-3)(\lambda-4)=0$

Therefore, the eigenvalues are $\lambda_1=1$, $\lambda_2=3$ and $\lambda_3=4$.

Power Method:

Definition 6.3: If λ_1 is an eigenvalue of A that is larger in absolute value than any other eigenvalue, it is called the dominant eigenvalue.

Definition 6.4: An eigenvector V is said to be normalized if the coordinate of largest magnitude is equal to unity. (i.e. the largest coordinate in the vector V is the number 1).

It is easy to normalize an eigenvector $[v_1\ v_2\ \dots\ v_n]^T$, by forming a new vector $V=(1/c)\,[v_1\ v_2\ \dots\ v_n]^T$, where $c=v_j$ and $v_j=\max\limits_{1\le i\le n}\{|v_i|\}$.

Suppose that the matrix A has a dominant eigenvalues λ and that there is a unique normalized eigenvector V that corresponds to λ. This eigenpair λ, V can be found by the following iterative procedure called ***power method***. Start with the vector

$$X_0=[1 \;\; 1 \;\; \ldots \;\; 1]^T \qquad (6.3)$$

Generate the sequence $\{X_k\}$ recursively, using

$$Y_k=AX_k \qquad (6.4)$$

$$X_{k+1}=\frac{1}{c_{k+1}}Y_k \qquad (6.5)$$

where c_{k+1} is the coordinate of Y_k of largest magnitude. The sequences $\{X_k\}$and $\{c_k\}$ will converge to V and λ, respectively:

$$\lim_{k\to\infty} X_k = V \qquad \mathit{and} \qquad \lim_{k\to\infty} c_k = \lambda \qquad (6.6)$$

Example 6.2*:* Use the power method to find the dominant eigenvalue and eigenvector for the matrix

$$A=\begin{bmatrix} 0 & 11 & -5 \\ -2 & 17 & -7 \\ -4 & 26 & -10 \end{bmatrix}$$

Start $X_0=[1 \;\; 1 \;\; 1]^T$ and use the formulas in (6.4) and (6.5) to generate the sequence of vectors $\{X_k\}$ and constants $\{c_k\}$. The first iteration produces

$$\begin{bmatrix} 0 & 11 & -5 \\ -2 & 17 & -7 \\ -4 & 26 & -10 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \\ 12 \end{bmatrix} = 12\begin{bmatrix} \frac{1}{2} \\ \frac{2}{3} \\ 1 \end{bmatrix} = c_1X_1$$

The second iteration produces

$$\begin{bmatrix} 0 & 11 & -5 \\ -2 & 17 & -7 \\ -4 & 26 & -10 \end{bmatrix}\begin{bmatrix} \frac{1}{2} \\ \frac{2}{3} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{7}{3} \\ \frac{10}{3} \\ \frac{16}{3} \end{bmatrix} = \frac{16}{3}\begin{bmatrix} \frac{7}{16} \\ \frac{5}{8} \\ 1 \end{bmatrix} = c_2X_2$$

Iteration generate the sequence {X$_k$} (where X$_k$ is a normalized vector):

$$12\begin{bmatrix}\frac{1}{2}\\ \frac{2}{3}\\ 1\end{bmatrix},\frac{16}{3}\begin{bmatrix}\frac{7}{16}\\ \frac{5}{8}\\ 1\end{bmatrix},\frac{9}{2}\begin{bmatrix}\frac{5}{12}\\ \frac{11}{18}\\ 1\end{bmatrix},\frac{38}{9}\begin{bmatrix}\frac{31}{76}\\ \frac{23}{38}\\ 1\end{bmatrix},\frac{78}{19}\begin{bmatrix}\frac{21}{52}\\ \frac{47}{78}\\ 1\end{bmatrix}$$

the sequence of vectors converges to V=$\begin{bmatrix}\frac{2}{3} & \frac{3}{5} & 1\end{bmatrix}^T$, and the sequence of constants converges to λ=4.

Exercises:

Find the dominant eigenpair of the following matrices:

$$A=\begin{bmatrix}7 & 6 & -3\\ -12 & -20 & 24\\ -6 & -12 & 16\end{bmatrix}\quad ,B=\begin{bmatrix}-14 & -30 & 42\\ 24 & 49 & -66\\ 12 & 24 & -32\end{bmatrix}$$

(do two iteration).