CHAPTER

TWO

ROOTS OF NON-LINEAR EQUATIONS

Contents

2.1 Categories of Non-Linear Equations 13
2.2 Methods to find Roots of Non-Polynomial Equations 14
2.3 Bracketing Methods 14
2.4 Open Methods 17
2.5 Roots of Polynomial Equations 20

Important Note: Dear Students, these lecture notes are "Supplementary", which means, they are not meant to replace your text book, but only provide supporting material to the classroom lectures, in a summarized manner. You should gain enough knowledge from other sources to be in a position to elaborate the topics and material presented here [1]

2.1 Categories of Non-Linear Equations

Do you know this equation:

$$f(x) = ax^2 + bx + c = 0 \tag{2.1}$$

Ofcourse, you know it! You remember it from your school days. It is the famous *Quadratic Equation*. A *Quadratic Equation* has two solutions:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{2.2}$$

These two solutions are known as ***Roots of the Equation*** 2.1, or ***Zeros of the Equation***.

Quadratic Equation 2.1 falls into the broad category of ***Polynomial Equations***. More precisely, it should be called a ***Second Order Polynomial Equation***. Similarly, another famous form of ***Polynomial Equation*** is ***Cubic Equation*** (which is a ***Third Order Polynomial Equation***):

$$f(x) = ax^3 + bx^2 + cx + d = 0 \tag{2.3}$$

So, an ***n^{th}-Order Polynomial Equation*** (*a.k.a* ***n^{th}-Degree Polynomial Equation***) can be generalized as:

$$f(x) = c_1x^n + c_2x^{n-1} + c_3x^{n-2} + \cdots + c_nx + c_{n+1} = 0 \tag{2.4}$$

Hence, as per this definition, a *First Order Polynomial Equation* should take the form:

$$f(x) = ax + b = 0 \tag{2.5}$$

It is commonly known as *Linear Equation*, and written in the form:

$$y = mx + c \tag{2.6}$$

Other than the *Polynomial Equations* there are many other forms of equations. These include equations involving trigonometric functions, logarithmic functions, exponential functions, and many other less common functions. Examples of such equations are:

$$f(x) = e^{-x} - x \tag{2.7}$$

$$f(x) = \sin^2 x + \cos^2 x \tag{2.8}$$

$$f(x) = \ln|x^3| + e^{-\frac{1}{x}} \tag{2.9}$$

In Chapter 1 we derived the equation for terminal velocity of parachutist as:

$$v = \frac{gm}{c}\left[1 - e^{-\frac{c}{m}t}\right] \tag{2.10}$$

Equations 2.7, 2.8, 2.9, 2.10, and similar, fall into the category of ***Non-Polynomial Equations*** (*a.k.a.* ***Transcendental Equations***).

As can be observed easily, solution to *Linear Equation* 2.5, is very obvious (i.e. $x = -c/b$) and does not require any considerable effort. But, any other *Polynomial Equation* (of degree 2, 3, ..., n) and all *Non-Polynomial Equations* demand at least some minimum effort to calculate the roots. We should call these two categories as ***Non-Linear Equations***.

Hence, in the rest of this chapter we will consider the solution to all *Polynomial Equations of degree 2 or higher*, and all *Non-Polynomial Equations*. In other words, in the rest of this chapter, we will solve ***Non-Linear Equations***.

2.2 Methods to find Roots of Non-Polynomial Equations

Several different approaches exist to solve *Non-Polynomial Equations*. Among those, most commonly used methods can be categorized, broadly, into two: (i) Bracketing Methods, and (ii) Open Methods.

Bracketing Methods employ two initial guesses to reach the solution of the equation. These two guesses must bound the root of the equation. In other words, the two guesses must lie on opposite sides of the root of the equation. Once the guesses are correctly suggested, as per this requirement, *Bracketing Methods* will ***SURELY*** converge to the root of the equation.

The other category, ***Open Methods***, do not carry any such requirement, with regards to initial guesses. Hence, with regards to initial guess, *Open Methods* are more flexible than *Bracketing Methods*. But, this flexibility also results in a drawback of *Open Methods*: there is ***NO SURETY*** that the method will converge to the solution. On the contrary, this is not a serious drawback, because it can easily be overcome by restarting the method with another guess. Furthermore, the probability of non-convergence of *Open Methods* to the root is, in general, very low for most of the equations.

Open Methods are also superior to *Brackting Methods* in another aspect; several *Open Methods* need a single guess, as opposed to two guesses in *Bracketing Methods*.

Finally, it is worth mention that, although there exist special faster and/or easier methods to find the roots of *Polynomial Equations* (like *Muller's Method*, which we will study in the end of this chapter), but both *Bracketing Methods* as well as *Open Methods* are equally well-suited to find the roots of *Polynomial Equations* too. Hence, we will apply these methods to all *Non-Linear Equations*.

It is equally worth mention that, both, *Bracketing Methods* as well as *Open Methods*, would (normally) find a single root of an equation. To find multiple roots, initial guesses must be changed, so that the method converges to a different root.

(***Note:*** *Open Methods* can be modified and adopted to find multiple roots, to solve *Systems of Linear Equations* as well as to solve *Systems of Non-Linear Equations*, but, because it is outside our syllabus, we will not study those extended methods in our course.)

2.3 Bracketing Methods

The theory behind *Bracketing Methods* can be summarized in the form of an algorithm as follows. To find a root of equation $f(x) = 0$:

1. *Initial Guess:* Two initial guess roots, lower-bound a and upper-bound b ($a < b$) are suggested, such that $f(a)f(b) < 0$, to ensure that initial guess roots bound the actual solution
2. *Transformation:* Apply some transformation (it will be choice) on a and b to calculate a new value c which lies in-between a and b (i.e. $a < c < b$)
3. *Pick new bounds:* Compare $f(a)f(c)$ and $f(b)f(c)$ to check which product gives a negative value
   - ☛ If $f(a)f(c) < 0$ then the actual solution lies between a and c; hence, discard b and treat c as new upper-bound (i.e. treat c as b for next calculation)
   - ☛ If $f(b)f(c) < 0$ then the actual solution lies between b and c; hence, discard a and treat c as new lower-bound (i.e. treat c as a for next calculation)
4. *Calculate Error:* Calculate the error $\epsilon = b - a$
5. If $\epsilon < tol$ (where tol is the maximum acceptable tolerance)
   - ☛ then treat latest value of c as the approximate solution and stop the method
   - ☛ else repeat the whole process, except the first step *Initial Guess*

All the *Bracketing Methods* follow the same algorithm, except the *Transformation* step, where the method decides how to calculate c from a and b

2.3.1 Bisection Method

Bisection method is the simplest bracketing method to find a root of $f(x) = 0$. It is assumed that $f(x)$ is continuous on an interval $[a, b]$ and has a root there, so that $f(a)$ and $f(b)$ have opposite signs, hence $f(a)f(b) < 0$.

The procedure goes as follows: Locate the midpoint of range $[a, b]$, that is, $c = c_1 = \frac{1}{2}(a + b)$ (this is the *'Transformation'* step in algorithm for *Bracketing Method*), (see Figure 2.1 on page 15). If $f(a)$ and $f(c)$ have opposite signs, the interval $[a, c_1]$ contains the root and will be retained for further analysis. If $f(b)$ and $f(c)$ have opposite signs, we continue with $[c_1, b]$. In Figure 2.1, it so happens that the interval $[c_1, b]$ brackets the root and is retained. Since the right endpoint is unchanged, we update the interval $[a, b]$ by resetting the left endpoint $a = c$. With the reduced new range $[a, b]$ the next mid-point, $c = c_2 = \frac{1}{2}(a + b)$ is calculated. The process is repeated by calculating the mid-points $c_2, c_3, c_4, \cdots$ until the length of the most recent interval $[a, b]$ satisfies the desired accuracy.

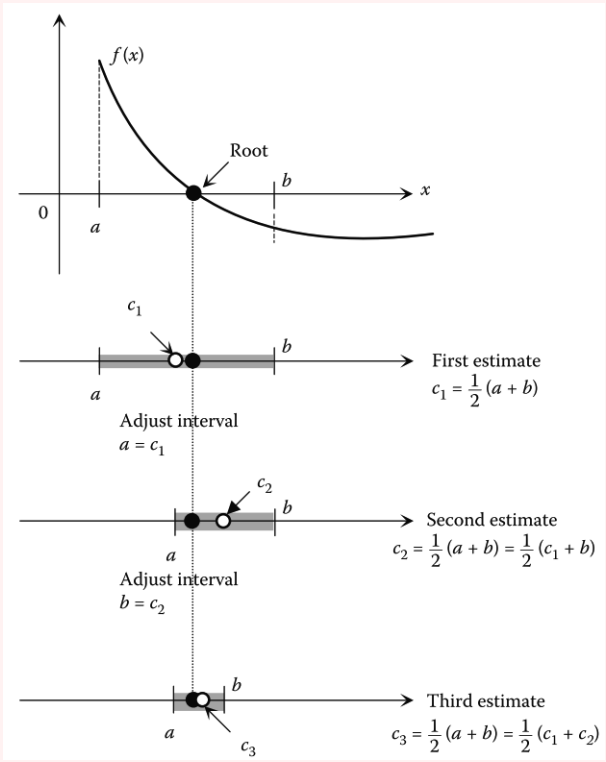

Figure (2.1) – Concept of Bisection Method

Example 2.1 : Solve $x\cos(x) + 1 = 0$ using Bisection Method

Find the root of equation $x\cos(x) = -1$, *within a* tolerance $= 10^{-2}$, *using Bisection method.*

Solution: To find the root, precisely, up to second decimal place, a good initial guess is required. This can be done, easily, by first plotting the graph of the equation (as shown in Figure 2.2 on page 15). Graph shows that function has a root in the interval [-2,4].

Hence, set $a = -2$ and $b = 4$. Calculate the mid-point $c = \dfrac{a+b}{2} = \dfrac{(-2)+(4)}{2} = 1$. Calculate the ordinates for a, b, and c: $f(a) = (-2)\cos(-2) + 1 = 1.8323$, $f(b) = (4)\cos(4) + 1 = -1.6146$, and $f(c) = (1)\cos(1) + 1 = 1.5403$. To determine whether the root lies in the range $[a, c]$ or $[b, c]$ calculate the products $f(a)f(c)$ and $f(b)f(c)$: $f(a)f(c) = 2.8223 > 0$, $f(b)f(c) = -2.4869 < 0$. $f(b)f(c) < 0$ indicates that root lies in the range $[b, c]$ and not in the range $[a, c]$. Hence, discard the range $[a, c]$ by considering the new range as $a = c = 1$ and $b = 4$ i.e. $[a, b] = [1, 4]$. Calculate the error as: $\varepsilon = \dfrac{b-a}{2} = \dfrac{4-(-2)}{2} = 3$. Because the error ($\varepsilon$) is more than tolerance $= 10^{-2}$, repeat the whole process.

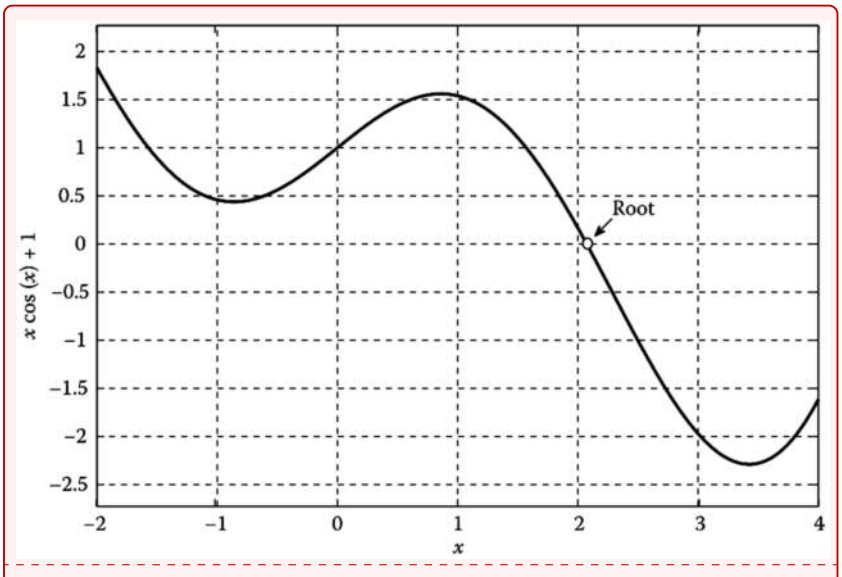

Figure (2.2) – Graph of $x\cos(x) + 1$

Calculate the mid-point $c = \dfrac{a+b}{2} = \dfrac{(1)+(4)}{2} = 2.5$. Calculate the ordinates for a, b, and c: $f(a) = (1)\cos(1) + 1 = 1.5403$, $f(b) = (4)\cos(4) + 1 = -1.6146$, and $f(c) = (2.5)\cos(2.5) + 1 = -1.0029$. To determine whether the root lies in the range $[a, c]$ or $[b, c]$ calculate the products $f(a)f(c)$ and $f(b)f(c)$: $f(a)f(c) = -1.5447 < 0$, $f(b)f(c) = 1.6192 > 0$. $f(a)f(c) < 0$ indicates that root lies in the range $[a, c]$ and not in the range $[b, c]$. Hence, discard the range $[b, c]$ by considering the new range as $a = 1$ and $b = c = 2.5$ i.e. $[a, b] = [1, 2.5]$. Calculate the error as: $\varepsilon = \dfrac{b-a}{2} = \dfrac{4-(-2)}{2} = 3$. Because the error ($\varepsilon$) is more than tolerance $= 10^{-2}$, once again repeat the whole process.

Instead of doing individual calcualtions it is much more intuitive to prepare Table 2.1 (as shown on page 16). In each iteration, calculate the approximate error $\varepsilon = \dfrac{\mathbf{b-a}}{\mathbf{2}}$, and compare it with the given tolerance $= \mathbf{10^{-2}}$. As soon as $\varepsilon <$ tolerance, the procedure stops.

As can be seen from Table 2.1 (on page 16), the procedure attains the value of $c = 2.0723$ after 10 iterations, which has the error $\varepsilon = 0.0059$, which is within our tolerance level of 10^{-2}.

It can also be observed from Table 2.1 (on page 16) that the error gets exactly halved in each iteration. Because of this fact, it is possible to estimate, ahead of procedure, the number of iterations required to achieve a desired precision (i.e. root with in tolerance). That is, in iteration 1, the error is $\varepsilon = \dfrac{b-a}{2}$. In iteration 2, the error is $\varepsilon = \dfrac{b-a}{2^2}$. In iteration 3, the error is $\varepsilon = \dfrac{b-a}{2^3}$. Hence, to generalize, in iteration N, the error is $\varepsilon = \dfrac{b-a}{2^N}$. To stop the procedure, the condition to be satisfied is:

Table (2.1) – Bisection Method: Solve $\mathbf{x}\cos(\mathbf{x}) = -\mathbf{1}$

Iteration	a	b	c	f(a)	f(b)	f(c)	f(a)f(c)	f(b)f(c)	ε
1	-2.0000	4.0000	1.0000	1.8323	-1.6146	1.5403	2.8223	-2.4869	3.0000
2	1.0000	4.0000	2.5000	1.5403	-1.6146	-1.0029	-1.5447	1.6192	1.5000
3	1.0000	2.5000	1.7500	1.5403	-1.0029	0.6881	1.0598	-0.6900	0.7500
4	1.7500	2.5000	2.1250	0.6881	-1.0029	-0.1183	-0.0814	0.1187	0.3750
5	1.7500	2.1250	1.9375	0.6881	-0.1183	0.3053	0.2101	-0.0361	0.1875
6	1.9375	2.1250	2.0313	0.3053	-0.1183	0.0973	0.0297	-0.0115	0.0938
7	2.0313	2.1250	2.0781	0.0973	-0.1183	-0.0096	-0.0009	0.0011	0.0469
8	2.0313	2.0781	2.0547	0.0973	-0.0096	0.0441	0.0043	-0.0004	0.0234
9	2.0547	2.0781	2.0664	0.0441	-0.0096	0.0173	0.0008	-0.0002	0.0117
10	2.0664	2.0781	2.0723	0.0173	-0.0096	0.0038	0.0001	0.0000	0.0059

$$\varepsilon < tol \quad \Rightarrow \quad tol > \varepsilon \quad \Rightarrow \quad tol > \frac{b-a}{2^N}$$

Solving the inequality for N yields:

$$N > \frac{\ln(b-a) - \ln(tol)}{\ln(2)}$$

Hence, if the procedure starts with initial estimates a and b, and the desired tolerance is tol, then the number of iterations, N, after which the desired precision can be achieved by *Bisection Method*, is given by:

$$\mathbf{N} = \left\lceil \frac{\ln(\mathbf{b-a}) - \ln(\mathbf{tol})}{\ln(\mathbf{2})} \right\rceil = \left\lceil \log_{\mathbf{2}} \left(\frac{\mathbf{b-a}}{\mathbf{tol}} \right) \right\rceil$$

In the example, $N = \left\lceil \dfrac{\ln(b-a) - \ln(tol)}{\ln(2)} \right\rceil = \left\lceil \dfrac{\ln(4-(-2)) - \ln(10^{-2})}{\ln(2)} \right\rceil = \lceil 9.23 \rceil = 10$

2.3.2 False-Position Method

The False Position method is another bracketing method to find a root of $f(x) = 0$. Once again, it is assumed that $f(x)$ is continuous on an interval $[a, b]$, and has a root there, so that $f(a)$ and $f(b)$ have opposite signs, or, equally valid to say, $f(a)f(b) < 0$.

The procedure is geometrical in nature, and described as follows. Let $[a_1, b_1] = [a, b]$ be the initial range that brackets the root. Connect points $A : (a_1, f(a_1))$ and $B : (b_1, f(b_1))$ by a straight line, as shown in Figure 2.3 (on page 16). Let c_1 be this line's x-intercept. Then, if $f(a_1)f(c_1) < 0$, then range $[a_1, c_1]$ brackets the root. Otherwise, the root is in the range $[c_1, b_1]$. In Figure 2.3 (on page 16), it just so happens that $[a_1, c_1]$ brackets the root. Hence, treat the range $[a_1, c_1] = [a, b]$ for next iteration and repeat the process by calculating new x-intercept c_2.

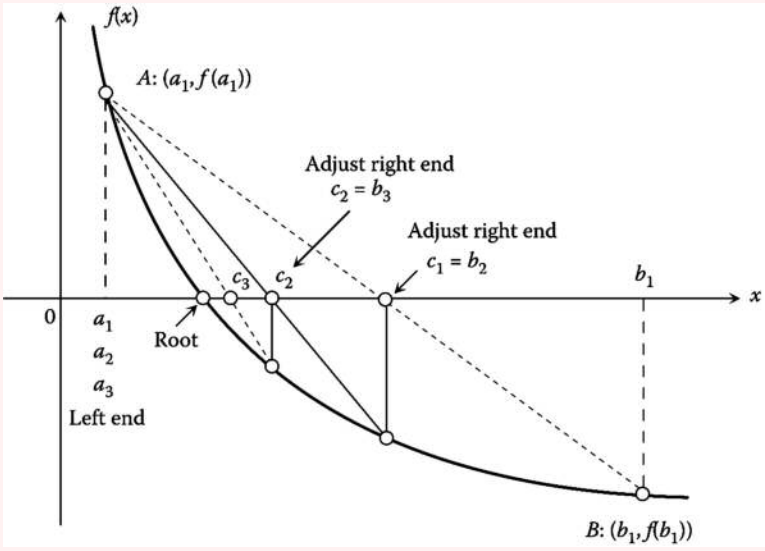

Figure (2.3) – Concept of False Position Method

Continuing this process generates a sequence $c_1, c_2, c_3, \cdots$ that eventually converges to the root.

Analytically, the procedure can be illustrated as follows. The equation of the line connecting points A and B is:

$$\frac{y - f(b_1)}{x - b_1} = \frac{f(a_1) - f(b_1)}{a_1 - b_1}$$

To find the x-intercept, set $y = 0$ and solve for $x = c_1$:

$$\frac{0 - f(b_1)}{c_1 - b_1} = \frac{f(a_1) - f(b_1)}{a_1 - b_1} \quad \Rightarrow \quad c_1 = \frac{a_1 f(b_1) - b_1 f(a_1)}{f(b_1) - f(a_1)}$$

Generalizing this result, the sequence of points that converges to the root is generated via

$$\mathbf{c_n} = \frac{\mathbf{a_n f(b_n) - b_n f(a_n)}}{\mathbf{f(b_n) - f(a_n)}} \qquad \mathbf{n = 1, 2, 3, \cdots} \tag{2.11}$$

Example 2.2 : Solve $x\cos(x) + 1 = 0$ using False Position Method

Find the root of equation $x\cos(x) = -1$, within a tolerance $= 10^{-2}$, using False-Position Method.

Table (2.2) – False Position Method: Solve $x\cos(x) = -1$

Iteration	a	b	c	f(a)	f(b)	f(c)	f(a)f(c)	f(b)f(c)	ε
1	-2.0000	4.0000	1.8323	-1.6146	1.1895	1.4426	2.6434	-2.3293	-
2	1.1895	4.0000	1.4426	-1.6146	2.5157	-1.0389	-1.4987	1.6773	1.3262
3	1.1895	2.5157	1.4426	-1.0389	1.9605	0.2552	0.3681	-0.2651	-0.5552
4	1.9605	2.5157	0.2552	-1.0389	2.0700	0.0091	0.0023	-0.0094	0.1095
5	2.0700	2.5157	0.0091	-1.0389	2.0738	0.0002	0.0000	-0.0002	0.0039

Solution : The procedure is exactly same as that we used in *Bisection Method*, except that calculation of c is different; here, Equation 2.11 is used to calculate c, and ε is calculated as the difference of two consecutive values of c.

As can be seen from Table 2.2 (on page 17), the procedure attains a value of $c = 2.0723$ after 5 iterations, which has an error of $\varepsilon = 0.0039$, and it is within the prescribed tolerance level of 10^{-2}.

2.4 Open Methods

Unlike *Bracketing Methods*, *Open Methods* cannot be summarized in the form of a general algorithm. But, there are certain common aspects which can be observed in all the *Open Methods*.

As opposed to *Bracketing Methods*, the *Open Methods* do not assure for convergence on root. But, still, each of the *Open Methods* demand some pre-conditions for convergence on root. If these conditions are fulfilled then they do converge on root.

Most of the *Open Methods* require only one initial guess (as opposed to two initial guesses required in *Bracketing Methods*), but this initial guess should be close enough to the root, so that the procedure converges on root. As such, here, the flexibility to choose initial guess is more than *Bracketing Methods*. (*Secant Method* is an *Open Method* which requires two initial guess, but, unlike the requirement of *Bracketing Methods* that the two guesses should always fall on opposite sides of the root, *Secant Method* also, like all the other *Open Methods*, does not impose any such condition.)

Lastly, the *Rate of Convergence* of *Open Methods* is usually higher when compared to any of the *Bracketing Methods*. In other words, the total number of iterations required to converge on a root, using *Open Methods*, is much less than required using *Bracketing Methods*.

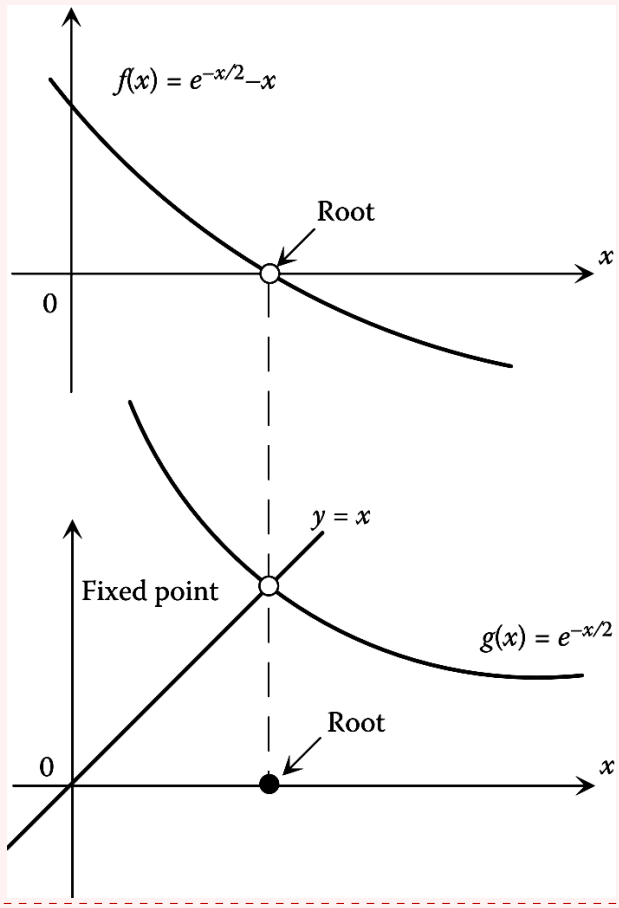

Figure (2.4) – Concept of Fixed-Point Iteration Method

2.4.1 Fixed-Point Iteration Method

The *Fixed-Point Iteration Method* is an *Open Method* to find a root of $f(x) = 0$. The idea is to rewrite $f(x) = 0$ as $x = g(x)$, where $g(x)$ is known as the ***Iteration Function*** or ***Auxiliary Function***. Consequently, a point of intersection of $y = g(x)$ and $y = x$, known as a ***fixed-point*** of $g(x)$, is also a root of $f(x) = 0$.

As an example, consider $e^{-\frac{x}{2}} - x = 0$ and its root, as shown in Figure 2.4 (on page 17). The equation is rewritten as $x = e^{-\frac{x}{2}}$ so that $g(x) = e^{-\frac{x}{2}}$ is the ***Iteration Function***.

It is observed that $g(x)$ has only one *fixed point*, which is the only root of the original equation. It should be noted that for a given equation $f(x) = 0$ there usually exist more than one *Iteration Function*. For instance, $e^{-\frac{x}{2}} - x = 0$ can also be rewritten as $x = -2\ln(x)$ so that $g(x) = -2\ln(x)$.

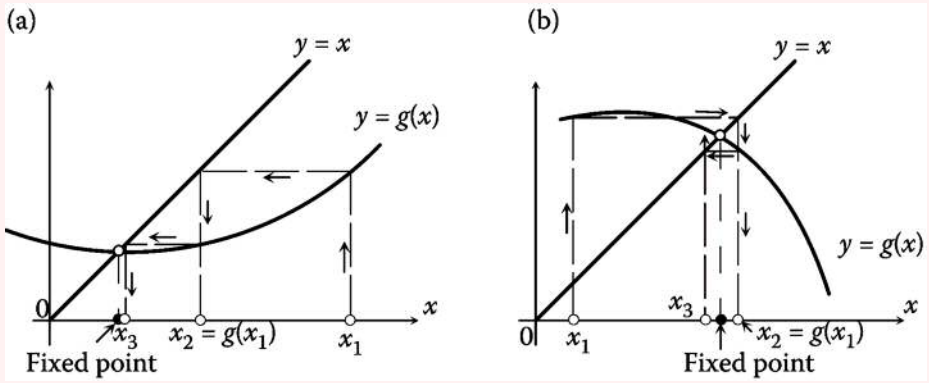

Figure (2.5) – Fixed Point Iteration Method: Types of Convergence (a) Monotone (b) Oscillatory

The *Fixed-Point* of $g(x)$ is found numerically via the ***Fixed-Point Iteration***:

$$\mathbf{x_{n+1} = g(x_n)}, \qquad (2.12)$$

where $n = 1, 2, 3, \cdots$
and x_1 is initial guess

The procedure begins with an initial guess x_1 near the *Fixed-Point*. The next point x_2 is found by evaluating $g(x_1)$. Similarly, x_3 is found by evaluating $g(x_2)$, then x_4, x_5, and so on. This continues until convergence is observed, that is, until two successive points are within a prescribed distance of each other:

$$|\mathbf{x_{n+1} - x_n}| < \mathbf{tol} \qquad (2.13)$$

Two types of convergence can be exhibited by the *Fixed-Point Iteration*: *Monotone* and *Oscillatory*, as illustrated in Figure 2.5 (on page 18). In a *Monotone Convergence*, the elements of the generated sequence converge to the *Fixed-Point* from one side, while in an *Oscillatory Convergence*, the elements bounce from one side of the *Fixed-Point* to the other as they approach it.

Convergence of Fixed-Point Iteration: It can be shown that if $|g'(x)| < 1$ near a *Fixed-Point* of $g(x)$, then convergence is guaranteed. Note that this is a sufficient, and not necessary, condition for convergence.

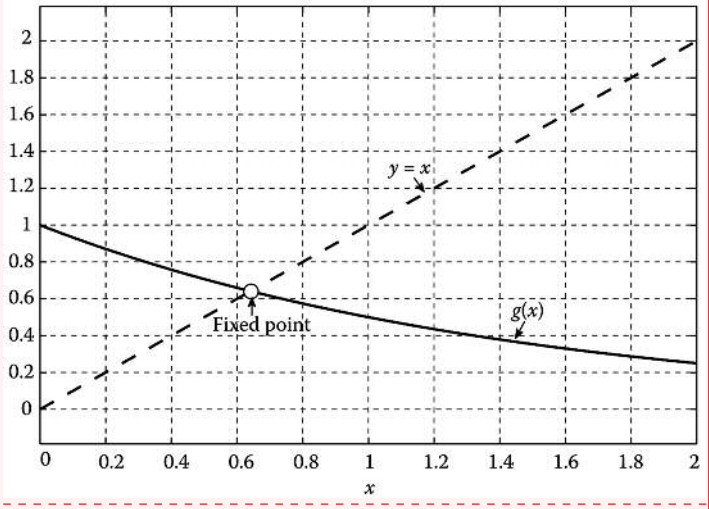

Figure (2.6) – Graph of $x - 2^{-x}$

Example 2.3 : Solve $x - 2^{-x} = 0$ using Fixed Point Iteration Method

Find a root of equation $x - 2^{-x} = 0$, within a tolerance $= 10^{-4}$, using Fixed-Point Iteration Method.

Table (2.3) – Fixed-Point Iteration Method: Solve $x - 2^{-x} = 0$

Iteration	x	g(x)	$\varepsilon = \lvert g(x) - x \rvert$
1	0.0000	1.0000	1.0000
2	1.0000	0.5000	0.5000
3	0.5000	0.7071	0.2071
4	0.7071	0.6125	0.0946
5	0.6125	0.6540	0.0415
6	0.6540	0.6355	0.0185
7	0.6355	0.6437	0.0082
8	0.6437	0.6401	0.0037
9	0.6401	0.6417	0.0016
10	0.6417	0.6410	0.0007
11	0.6410	0.6413	0.0003
12	0.6413	0.6411	0.0001
13	0.6411	0.6412	0.0000

Solution : Rewrite the equation as $x = 2^{-x}$ so that $g(x) = 2^{-x}$. The *Fixed-Point* can be roughly located as in Figure 2.6 (on page 18). To proceed, we can start with an initial guess of $x = 0$, and prepare, as before, a table of iteration as shown in Table 2.3 (on page 18).

The final answer (within the tolerance $= 10^{-4}$) is found as 0.6412, after 13 iterations.

2.4.2 Newton-Raphson Method

The most commonly used open method to solve $f(x) = 0$, where $f'(x) = 0$ is continuous, is *Newton-Raphson Method*.

Consider the graph of $f(x)$ in Figure 2.7 (on page 19). Start with an initial point x_1 and locate the point $(x_1, f(x_1))$ on the curve. Draw the tangent line to the curve at that point, and let its x-intercept be x_2. Locate $(x_2, f(x_2))$, draw the tangent line to the curve there, and let x_3 be its x-intercept. Repeat this until convergence is observed. In general, two consecutive elements x_n and x_{n+1} are related via

$$\mathbf{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}}, \quad \mathbf{n = 1, 2, 3, \cdots} \text{ and } \mathbf{x_1} \text{ is initial guess} \qquad (2.14)$$

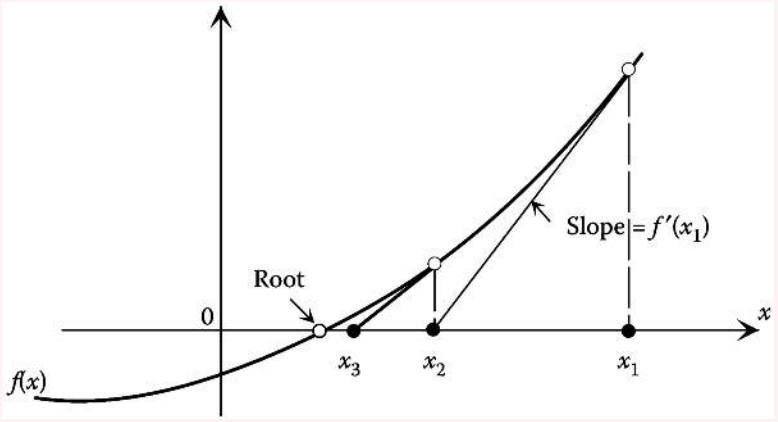

Figure (2.7) – Concept of Newton Raphson Method

The iterations stop when two consecutive elements are sufficiently close to one another, that is,

$$|x_{n+1} - x_n| < \varepsilon \qquad (2.15)$$

where ε is the prescribed tolerance.

Example 2.4 : Solve $x\cos(x) + 1 = 0$ using Newton Raphson Method

Find the first positive root of $x\cos(x) = -1$ using Newton-Raphson Method with a tolerance of 10^{-4}.

Solution : This equation was previously tackled. According to Figure 2.2 (on page 15), the first positive root is located around $x = 2$, thus we start our calculations with an initial guess as $x_1 = 1$. Also, we know $f'(x) = \cos(x) - x\sin(x)$. Hence, we proceed by preparing the Table 2.4 (as shown on page 19). The root of the equation is found to be 2.0739.

Table (2.4) – Newton-Raphson Method: Solve $x\cos(x) + 1 = 0$

Iteration	x_n	$f(x_n)$	$f'(x_n$	x_{n+1}	$\varepsilon = \lvert x_{n+1} - x_n \rvert$
1	1.0000	1.5403	-0.3012	6.1144	5.1144
2	6.1144	7.0275	2.0128	2.6230	3.4914
3	2.6230	-1.2782	-2.1686	2.0336	0.5894
4	2.0336	0.0920	-2.2662	2.0742	0.0406
5	2.0742	-0.0007	-2.2993	2.0739	0.0003
6	2.0739	0.0000	-2.2991	2.0739	0.0000

Example 2.5 : Solve $x^2 - 3x - 7 = 0$ using Newton Raphson Method

Find the roots of $x^2 - 3x - 7 = 0$ using Newton-Raphson method with a tolerance of 10^{-4}.

Solution : Start by plotting the graph for $f(x) = x^2 - 3x - 7$ to find approximate locations of its roots (see Figure 2.8 on page 19). Inspired by Figure 2.8, we prepare two tables (see Table 2.5 and Table 2.6 on page 19), one for each root. Initial guess, for first root, is $x_1 = -2$ and, for second root, is $x_1 = 4$. Accordingly, the two roots found are -1.5414 and 4.5414. (In the tables we employed $f'(x) = 2x - 3$.)

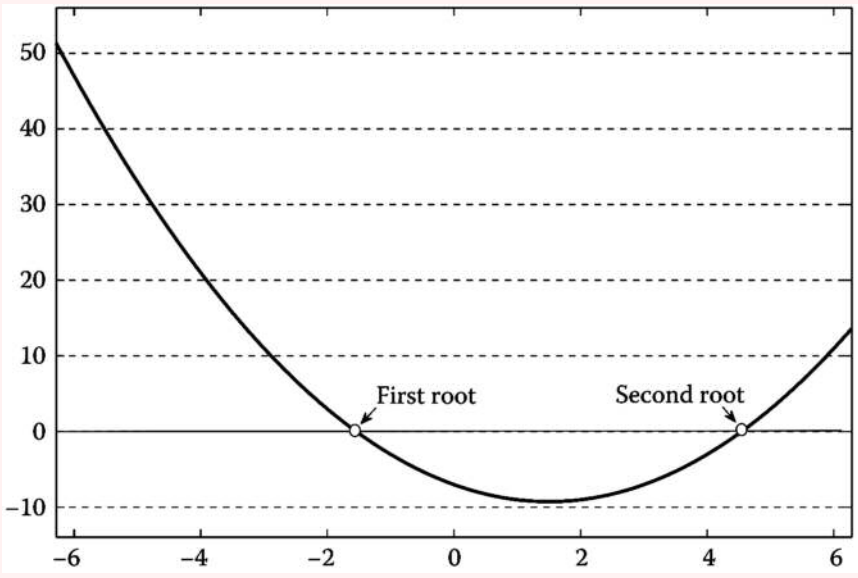

Figure (2.8) – Newton-Raphson Method: Graph of $f(x) \equiv x^2 - 3x - 7 = 0$

Some points with regards to Newton-Raphson Method

Following points are note-worthy as regards *Newton-Raphson Method:*

- When *Newton-Raphson Method* works, it generates a sequence that converges rapidly to the intended root.
- Several factors may cause *Newton-Raphson Method* to fail.
  1. The initial point x_1 is not sufficiently close to the intended root.
  2. At some point in the iterations, $f'(x)$ may be close to or equal to zero.
  3. The iteration halts (usually because of point of discontinuity).
  4. The sequence diverges
- If $f(x)$, $f'(x)$, and $f''(x)$ are continuous, $f'(\text{root}) \neq 0$, and the initial point x_1 is close to the root, then the sequence generated by *Newton-Raphson Method* converges to the root.

Table (2.5) – Newton-Raphson Method: Solve $x^2 - 3x - 7 = 0$: First Root

Iteration	x_n	$f(x_n)$	$f'(x_n$	x_{n+1}	$\varepsilon = \lvert x_{n+1} - x_n \rvert$
1	-2.0000	3.0000	-7.0000	-1.5714	0.4286
2	-1.5714	0.1837	-6.1429	-1.5415	0.0299
3	-1.5415	0.0009	-6.0831	-1.5414	0.0001
4	-1.5414	0.0000	-6.0828	-1.5414	0.0000

Table (2.6) – Newton-Raphson Method: Solve $x^2 - 3x - 7 = 0$: Second Root

Iteration	x_n	$f(x_n)$	$f'(x_n$	x_{n+1}	$\varepsilon = \lvert x_{n+1} - x_n \rvert$
1	4.0000	-3.0000	5.0000	4.6000	0.6000
2	4.6000	0.3600	6.2000	4.5419	0.0581
3	4.5419	0.0034	6.0839	4.5414	0.0006
4	4.5414	0.0000	6.0828	4.5414	0.0000

☛ A downside of *Newton-Raphson Method* is that it requires the expression for $f'(x)$, which can, at times, be difficult.

2.4.3 Secant Method

As mentioned earlier (in Section 2.4.2), at times, finding expression for $f'(x)$ becomes difficult. In such cases, an alternative to *Newton-Raphson Method* is the *Secant Method*.

The secant method is another open method to solve $f(x) = 0$. Consider the graph of $f(x)$ in Figure 2.9 (on page 20). Start with two initial points x_1 and x_2, locate the points $(x_1, f(x_1))$ and $(x_2, f(x_2))$ on the curve, and draw the secant line connecting them. The x-intercept of this secant line is x_3. Next, use x_2 and x_3 to define a secant line and let the x-intercept of this line be x_4. Continue the process until the sequence converges to the root. In general, two consecutive elements x_n and x_{n+1} generated by the secant method are related via

$$\frac{f(x_{n+1}) - f(x_n)}{x_{n+1} - x_n} = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$
$$\Rightarrow \quad x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n), \quad n = 2, 3, 4, \cdots \text{ and } x_1, x_2 \text{ are two initial guess} \tag{2.16}$$

Example 2.6 : Solve $x\cos(x) + 1 = 0$ using Secant Method

Find the first positive root of $x\cos(x) = -1$ using Secant method with a tolerance of 10^{-4}.

Solution : This equation was previously tackled. Using similar procedure followed earlier, we proceed by preparing Table 2.7 (as shown on page 20). Because we know that there is a root near $x = 2$ we can safely start with initial guess of $x_1 = 1$ and $x_2 = 1.5$. Root of the equation is found to be 2.0739.

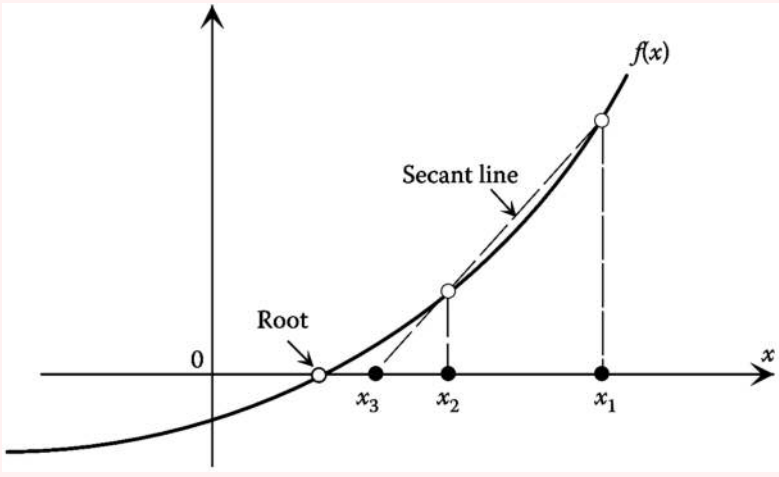

Figure (2.9) – Concept of Secant Method

Table (2.7) – Secant Method: Solve $x\cos(x) + 1 = 0$

Iter	x_{n-1}	x_n	$f(x_{n-1})$	$f(x_n)$	x_{n+1}	$\varepsilon = \lvert x_{n+1} - x_n \rvert$
1	1.00000	1.50000	1.54030	1.10611	2.77374	1.27374
2	1.50000	2.77374	1.10611	-1.58818	2.02292	0.75082
3	2.77374	2.02292	-1.58818	0.11624	2.07412	0.05120
4	2.02292	2.07412	0.11624	-0.00044	2.07393	0.00019
5	2.07412	2.07393	-0.00044	0.00000	2.07393	0.00000

2.5 Roots of Polynomial Equations

The *Bracketing Methods* and *Open Methods* described in former sections were equally suitable to solve *Polynomial Equations* as well as *Transcendental Equations*. But consideration of some aspects of *Polynomial Equations* force us to develop other methods which are more suitable to solve such equations.

The roots of polynomials follow these rules:

☛ For an *n^{th}-order equation*, there exist n roots, out of which some or all may be repeated (i.e., multiple roots with same solution value).
☛ Out of n roots, $m (0 < m < n)$ roots may be real and remaining $n - m$ roots may be complex.
☛ If n is odd then there is at least one real root.
☛ For an equation, if complex roots exist then they always exist as conjugate pairs (i.e., $\lambda \pm \mu i$ where $i = \sqrt{-1}$).

Bracketing Methods as well as *Open Methods* are very viable methods if only real roots exist. However, when complex roots exist *Bracketing Methods* cannot be used because of the obvious problem that the criterion for defining a bracket (that is, sign change) does not translate to complex guesses. Also, the problem of finding good initial guesses complicates both the bracketing and the open methods. Furthermore, the open methods could be susceptible to divergence.

In this section, we will discuss methods which specially excel in finding the roots of *Polynomial Equations*.

2.5.1 Muller's Method

Muller's Method is an inspiration from *Secant Method*. Instead of drawing a *Secant* which passes through two suggested points, *Muller's Method* suggests three points on the function curve, x_0, x_1, and x_2, and passes

a parabola through these three points (see Figure 2.10 on page 21). Definitely, the equation of this parabola is a quadratic equation.

The *Muller's Method* consists of deriving the coefficients of the parabola that goes through the three guess points, x_0, x_1, and x_2. These coefficients can then be substituted into the quadratic formula of the parabola to obtain the point where the parabola intercepts the x-axis – that is, the *estimated root*, or *next guess*, (call it x_3). The equation for parabola passing through a point (x_2, y_2) should take the form:

$$\mathbf{y_2 = a(x - x_2)^2 + b(x - x_2) + c}, \quad \textit{a, b, c are coefficients of the equation} \tag{2.17}$$

The three points, x_0, x_1, and x_2, must satisfy Equation 2.17. Hence,

$$\mathbf{y_0 = a(x_0 - x_2)^2 + b(x_0 - x_2) + c} \tag{2.18}$$
$$\mathbf{y_1 = a(x_1 - x_2)^2 + b(x_1 - x_2) + c} \tag{2.19}$$
$$\mathbf{y_2 = a(x_2 - x_2)^2 + b(x_2 - x_2) + c} \tag{2.20}$$

Equation 2.20 results in:

$$\mathbf{c = y_2} \tag{2.21}$$

Substituting Equation 2.21 in Equations 2.18 and 2.19 yields:

$$\mathbf{y_0 - y_2 = a(x_0 - x_2)^2 + b(x_0 - x_2)} \tag{2.22}$$
$$\mathbf{y_1 - y_2 = a(x_1 - x_2)^2 + b(x_1 - x_2)} \tag{2.23}$$

Equations 2.22 and 2.23 can be solved to give the values of coefficients a and b as:

$$\mathbf{a = +\frac{(x_0 - x_2)(y_1 - y_2) - (x_1 - x_2)(y_0 - y_2)}{(x_0 - x_1)(x_1 - x_2)(x_2 - x_0)}} \tag{2.24}$$
$$\mathbf{b = -\frac{(x_0 - x_2)^2(y_1 - y_2) - (x_1 - x_2)^2(y_0 - y_2)}{(x_0 - x_1)(x_1 - x_2)(x_2 - x_0)}} \tag{2.25}$$

The parabola also passes through the point $(x_3, y_3) = (x_3, 0)$. Substituting this point in the Equation of Parabola 2.17:

$$\mathbf{0 = a(x_3 - x_2)^2 + b(x_3 - x_2) + c} \tag{2.26}$$

As Equation 2.26 is a quadratic equation, it can be solved for $x_3 - x_2$ as:

$$\mathbf{x_3 - x_2 = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}} \tag{2.27}$$

or

$$\mathbf{x_3 = x_2 + \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}} \tag{2.28}$$

Both Equations, 2.27 and 2.28, are useful because they allow to calculate the error:

$$\varepsilon = \left| \frac{\mathbf{x_3 - x_2}}{\mathbf{x_3}} \right| \tag{2.29}$$

The above procedure completes a single iteration. To proceed to next iteration, substitute $x_0 = x_1$, $x_1 = x_2$, and $x_2 = x_3$, and calculate the new values of x_3 and ε. Repeat the process till $\varepsilon > tol$ (or, in other words, stop the process as soon as $\varepsilon < tol$)

A very important point (which may have been overlooked and ignored) with regards to calculation of values of y_0, y_1, and y_2, is that their values are calculated by substituting values of x_0, x_1, and x_2, in the given function $f(x)$, respectively. This is possible because the points (x_0, y_0), (x_1, y_1), and (x_2, y_2), lies simultaneously on the parabola as well as the curve of given function $f(x)$, and hence satisfy equations of both curves (i.e., parabola and given function).

Use of parabola, when compared to secant, results in very rapid convergence, using *Muller's Method*.

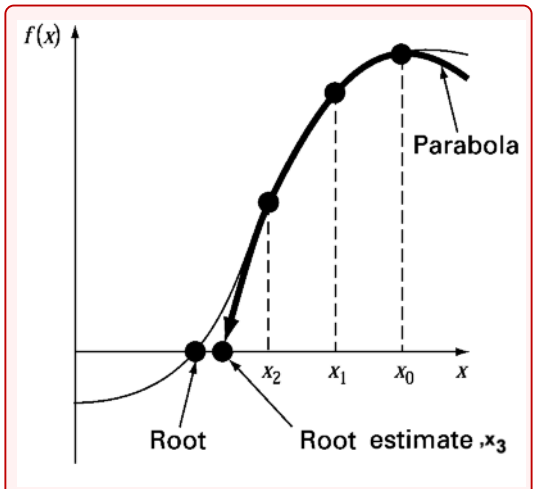

Figure (2.10) – Concept of Muller's Method

Example 2.7 : Solve $x^3 - 13x - 12 = 0$ using Muller's Method

Table (2.8) – Muller's Method: Solve $x^3 - 13x - 12 = 0$

Iteration	x_0	x_1	x_2	y_0	y_1	y_2	a	b	c	d_1	d_2	$x_3 - x_2$	x_3	ε
1	4.50000	5.50000	5.00000	20.62500	82.87500	48.00000	15.00000	62.25000	48.00000	93.79461	30.70539	-1.02351	3.97649	0.25739
2	5.50000	5.00000	3.97649	82.87500	48.00000	-0.81633	14.47649	32.87801	-0.81633	66.46721	-0.71119	0.02456	4.00105	0.00614
3	5.00000	3.97649	4.00105	48.00000	-0.81633	0.03678	12.97754	35.04975	0.03678	70.07226	0.02725	-0.00105	4.00000	0.00026
4	3.97649	4.00105	4.00000	-0.81633	0.03678	0.00002	11.97754	35.00004	0.00002	70.00007	0.00002	0.00000	4.00000	0.00000

Use Muller's Method with initial guesses of $x_0 = 4.5, x_1 = 5.5,$ *and* $x_2 = 5$*, to determine a root of the equation*

$$x^3 - 13x - 12 = 0$$

.

Solution : To solve the equation, prepare Table 2.8 (as shown on page 22), and repeat the process till ε falls sufficiently below a reasonable accuracy (for example, below 10^{-4}). In the table, columns d_1 and d_2 pertain to the denominators of $x_3 - x_2$:

$$\mathbf{d_1 = b + \sqrt{b^2 - 4ac}} \qquad \text{and} \qquad \mathbf{d_2 = b - \sqrt{b^2 - 4ac}} \tag{2.30}$$

While calculating the values of $x_3 - x_2$ and x_3 we always pick up either d_1 or d_2 whichever has the largest absolute value.

Beginning of Chapter ⇑ Table of Contents

CHAPTER
THREE

MATRICES AND SYSTEMS OF EQUATIONS

Contents

3.1 Vectors and Matrices 23
3.2 Elementary Properties of Matrices 24
3.3 Orthogonality and Orthonormality of Vectors and Matrices 27
3.4 Norm of Vectors and Matrices 28
3.5 Linear Equations 30
3.6 Systems of Linear Equations 30
3.7 Numerical Methods to solve Systems of Linear Equations 32
3.8 Numerical Methods to solve Systems of Non-Linear Equations 35

Important Note: Dear Students, these lecture notes are "Supplementary", which means, they are not meant to replace your text book, but only provide supporting material to the classroom lectures, in a summarized manner. You should gain enough knowledge from other sources to be in a position to elaborate the topics and material presented here [1]

3.1 Vectors and Matrices

A ***vector*** $\mathbf{v}$ is an ordered set of n scalars, $v_1, v_2, \ldots, v_n$, written as

$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

More precisely, this is a ***column-vector*** (a.k.a ***column-matrix***). Instead, if the scalars $v_1, v_2, \ldots, v_n$, are written in a row

$$\mathbf{v} = \begin{pmatrix} v_1 & v_2 & \ldots & v_n \end{pmatrix}$$

then it would become a ***row-vector*** (a.k.a ***row-matrix***). The count of scalars in the vector is known as the ***order of vector*** or ***size of vector***. Hence, in the example above, size of vector $\mathbf{v}$ is n. Individual scalars of the vector are known as ***elements*** or ***members*** or ***terms of the vector***. Hence, v_1 is first element of vector $\mathbf{v}$, and v_n is nth element.

When consecutive vectors are arranged sideways or stacked, it makes a ***matrix***:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix} \qquad \text{eg. a } 3 \times 3 \text{ matrix: } \mathbf{P} = \begin{pmatrix} 2 & 3 & 3 \\ -1 & 2 & 4 \\ 5 & 0 & -3 \end{pmatrix}$$

This is a ***Regular Matrix***, where, size of each row of matrix, m, matches with size of any other row, and similarly, sizes of all columns of matrix are same, n. In a matrix, if the size of at least one row (or column) is different than size of another row (or column) of the matrix then such a matrix falls in to the category of ***Sparse Matrices***. *Sparse matrices* have special applications and they are not of interest to us in this course. Hence, we will concentrate on *regular matrices*, and, as such, in this text the term *"matrices"* would refer to *"regular matrices"* only (unless otherwise mentioned explicitly).

The ***size of matrix*** or ***order of matrix*** having m rows and n columns is $m \times n$. In a matrix, $\mathbf{A}$, the element at i^{th} row and j^{th} column is denoted by a_{ij}. (For example, in the above matrix $\mathbf{P}$, element $p_{32} = 0$.) Also, m is the ***first-dimension of matrix*** $\mathbf{A}$ (i.e., the count of rows in the matrix) and n is the ***second-dimension*** (i.e., the count of columns), and as such, $\mathbf{A}$ is a ***two-dimensional (or 2D) matrix***. Higher dimensional matrices, such as 3D matrices, are possible, but, in this course, we will limit ourselves to *2D matrices*.

A 2D matrix of size $n \times n$ is known as a ***Square Matrix***. Consequently, saying "matrix of order n" means same as "matrix of order $n \times n$". A *"non-square matrix"* is known as a ***Rectangular Matrix***. The elements $a_{11}, a_{22}, \ldots, a_{nn}$ in a square matrix of size n form the ***Principal Diagonal*** of matrix. In the display below, *principal diagonal* of matrix $\mathbf{A}$ is shown in bold-characters:

$$\mathbf{A} = \begin{pmatrix} \mathbf{a_{11}} & a_{12} & \ldots & a_{1n} \\ a_{21} & \mathbf{a_{22}} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & \mathbf{a_{mn}} \end{pmatrix}$$

The sum of elements of principal diagonal of a matrix is known as ***Trace*** of matrix.

A ***diagonal matrix*** is a square matrix which has all it's elements as zeros, except that at least one of the elements on it's principal diagonal is non-zero. A *diagonal matrix of order n* which has all the elements (on it's *principal diagonal*) as ones, is called ***Identity Matrix of order n*** denoted by $\mathbf{I_n}$. A ***null-matrix*** (a.k.a ***zero-matrix***) has all it's elements as zeros, and a ***unit matrix*** has all it's elements as ones.

$\begin{pmatrix} a_{11} & 0 & 0 & \ldots & 0 \\ 0 & a_{22} & 0 & \ldots & 0 \\ 0 & 0 & a_{33} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & a_{nn} \end{pmatrix}$	$\mathbf{I_n} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & 1 & 1 & \ldots & 1 \\ 1 & 1 & 1 & \ldots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \ldots & 1 \end{pmatrix}$
Diagonal Matrix	Identity Matrix (of order n)	Null Matrix	Unit Matrix

If O is the $m \times n$ zero matrix and $\mathbf{A}$ is any $m \times n$ matrix, then $A + O = A$. Thus, *Null Matrix* O is the ***Additive Identity*** for matrix addition. Now that the *Additive Identity* for matrix addition is defined, we can observe that the matrix $-A$, which can be obtained as $(-1) \times A$, is the ***Additive Inverse*** of A, in the sense that $A + (-A) = (-A) + A = O$.

A *square matrix of order n* which has all the elements above it's *principal diagonal* as zeros is known as ***Lower Triangular Matrix*** (shown as matrix $\mathbf{L}$ below). Similarly, a *square matrix of order n* which has all the elements below it's *principal diagonal* as zeros is known as ***Upper Triangular Matrix*** (shown as matrix $\mathbf{U}$ below).

$$\mathbf{L} = \begin{pmatrix} a_{11} & 0 & 0 & \ldots & 0 \\ a_{21} & a_{22} & 0 & \ldots & 0 \\ a_{31} & a_{32} & a_{33} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn} \end{pmatrix} \qquad \mathbf{U} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ 0 & a_{22} & a_{23} & \ldots & a_{2n} \\ 0 & 0 & a_{33} & \ldots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & a_{nn} \end{pmatrix}$$

The operation of exchanging rows of matrix with it's columns (and vice-versa) is known as ***Transposition*** of matrix, and the new matrix produced is called ***transpose*** of original matrix. Transpose of matrix $\mathbf{A}$ is denoted by $\mathbf{A^T}$ or $\mathbf{A'}$. Hence

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix} \implies \mathbf{A'} = \begin{pmatrix} a_{11} & a_{21} & \ldots & a_{m1} \\ a_{12} & a_{22} & \ldots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \ldots & a_{mn} \end{pmatrix}$$

3.2 Elementary Properties of Matrices

The ***Scalar Multiple*** αA of matrix A by a real number α is the matrix obtained by multiplying each entry of A by α.

$$\alpha A = \alpha \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} = \begin{pmatrix} \alpha a_{11} & \alpha a_{12} \cdots & & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} \cdots & & \alpha a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ \alpha a_{m1} & \alpha a_{m2} & \cdots & \alpha a_{mn} \end{pmatrix}$$

If $\mathbf{u_1}, \mathbf{u_2}, \cdots, \mathbf{u_m}$ are vectors and if $a_1, a_2, \cdots, a_m$ are scalars, then the vector

$$a_1\mathbf{u_1} + a_2\mathbf{u_2} + \cdots + a_m\mathbf{u_m}$$

is called a ***Linear Combination*** of $\mathbf{u_1}, \mathbf{u_2}, \cdots, \mathbf{u_m}$.

Product of matrix and vector: If A is an $m \times n$ matrix and $\mathbf{x}$ is a $(n \times 1)$ column vector then the product $A\mathbf{x}$ is defined as

$$A\mathbf{x} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{pmatrix}$$

Note that, this product is not commutative, and as such $A\mathbf{x} \neq \mathbf{x}A$. But, this product is distributive, *i.e.* $A(\mathbf{x} + \mathbf{y}) = A\mathbf{x} + A\mathbf{y}$, and, similarly, $(\mathbf{x} + \mathbf{y})A = \mathbf{x}A + \mathbf{y}A$, where $\mathbf{x}$ and $\mathbf{y}$ are column vectors.

Row Echelon Form : A matrix is said to be in *Row Echelon Form* if the first non-zero entry in every row is to the right of the first non-zero entry in all the rows above.

Following matrices are in *Row Echelon Form*:

$$\begin{pmatrix} 2 & -5 & 3 & 9 \\ 0 & 7 & -2 & 4 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}, \quad \begin{pmatrix} 6 & 0 & -3 & 4 & 0 & 8 \\ 0 & 0 & 0 & -2 & 3 & -7 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} -9 & 0 & 0 & 0 \\ 0 & 7 & 0 & 4 \\ 0 & 0 & 3 & 8 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The first non-zero entry of each row is known as the ***Pivot Value*** or ***Corner Value***.

A matrix in *Row Echelon Form* which has all it's *Pivot Values* as 1 is said to be in ***Reduced Row Echelon Form***, or simply ***Reduced Form***. Hence, all the *Identity Matrices* are in *Reduced Row Echelon Form*.

Elementary Row Operations (ERO) : The three EORs that can be performed on any matrix are as follows:

1. ERO_s: *Row Swaps:* interchange two rows
2. ERO_d: *Row Dilations:* multiply a row by a non-zero real scalar
3. ERO_t: *Row Transvections:* add a multiple of another row to a row

It is interesting to note that ***Any matrix can be transformed into an equivalent reduced form by a (not unique) sequence of elementary row operations***. We will denote the reduced form of an $m \times n$ matrix A as A_{red}.

The ***Rank*** of an $m \times n$ matrix A is the number of non-zero rows in A_{red}.

Example 3.1 : Reducing a matrix using Elementary Row Operations

Following example demonstrates transformation of a matrix into it's equivalent reduced form (R_x denotes *Row x*):

$$\begin{pmatrix} 2 & 8 & 7 \\ 4 & 5 & 6 \\ 3 & 2 & 9 \end{pmatrix} \xrightarrow{(ERO_d:)\frac{1}{2}R_1} \begin{pmatrix} 1 & 4 & \frac{7}{2} \\ 4 & 5 & 6 \\ 3 & 2 & 9 \end{pmatrix} \xrightarrow{(ERO_t:)R_2-4R_1, R_3-3R_1} \begin{pmatrix} 1 & 4 & \frac{7}{2} \\ 0 & -11 & -8 \\ 0 & -9 & -\frac{3}{2} \end{pmatrix} \xrightarrow{(ERO_d:)-\frac{1}{11}R_2}$$

$$\begin{pmatrix} 1 & 4 & 7/2 \\ 0 & 1 & 8/11 \\ 0 & -9 & -3/2 \end{pmatrix} \xrightarrow{(ERO_t:)R_3+9R_2} \begin{pmatrix} 1 & 4 & 7/2 \\ 0 & 1 & 8/11 \\ 0 & 0 & 111/22 \end{pmatrix} \xrightarrow{(ERO_d:)\frac{22}{111}R_3} \begin{pmatrix} 1 & 4 & 7/2 \\ 0 & 1 & 8/11 \\ 0 & 0 & 1 \end{pmatrix}$$

Elementary Matrices and Row Operations

It is interesting to note know that *Elementary Row Operations* can be written in the form of matrices. These matrices are known as ***Elementary Matrices***.

To understand *Elementary Matrices* let us write an $m \times n$ matrix A, more elaborately, as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1i} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2i} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ii} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{j1} & a_{j2} & \cdots & a_{ji} & \cdots & a_{jj} & \cdots & a_{jn} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mi} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix}$$

To perform ERO_s (Row Swap) on rows i and j, we can multiply matrix A with the following matrix:

$$E_s = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & a_{ii}=0 & \cdots & a_{ij}=1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & a_{ji}=1 & \cdots & a_{jj}=0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

The *Elementary Matrix* E_s to perform ERO_s on rows i and j is obtained by modifying I_m (the $m \times m$ *Identity Matrix*) such that now $a_{ii} = 0$, $a_{jj} = 0$, $a_{ij} = 1$, and $a_{ji} = 1$. If matrix A is a square matrix of size $n \times n$ then *Elementary Matrix* E_s to perform ERO_s on rows i and j is obtained by modifying the *Identity Matrix* I_n such that now $a_{ii} = 0$, $a_{jj} = 0$, $a_{ij} = 1$, and $a_{ji} = 1$.

Hence, if we perform the *matrix multiplication* E_sA we obtain a new matrix A_1 as:

$$A_1 = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1i} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2i} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{j1} & a_{j2} & \cdots & a_{ji} & \cdots & a_{jj} & \cdots & a_{jn} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ii} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mi} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix}$$

which is a transformation on original matrix A, such that, rows i and j of original matrix A are swapped (or interchanged) to produce the new transformed matrix A_1.

To perform ERO_d (Row Dilation) on row i, with a factor of d, we can multiply matrix A with the matrix E_d, as given below:

$$E_d = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & a_{ii}=d & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

The *Elementary Matrix* E_d to perform ERO_d on row i, with a multiplication factor of d, is obtained by modifying I_m (the $m \times m$ *Identity Matrix*) such that now $a_{ii} = d$. If matrix A is a square matrix of size $n \times n$ then *Elementary Matrix* E_d to perform ERO_d on row i, with a factor of d, is obtained by modifying the *Identity Matrix* I_n such that now $a_{ii} = d$.

Hence, if we perform the *matrix multiplication* E_dA we obtain a new matrix A_2 as:

$$A_2 = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1i} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2i} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ da_{i1} & da_{i2} & \cdots & da_{ii} & \cdots & da_{in} \\ \vdots & \vdots & \cdots & \vdots & & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mi} & \cdots & a_{mn} \end{bmatrix}$$

which is a transformation on original matrix A, such that, row i of original matrix A is dilated with a factor of d to produce the new transformed matrix A_2.

To perform ERO_t (Row Transvection) on row i using row j, with a factor of t, we can multiply matrix A with the matrix E_t, as given below:

astrax111@gmail.com

$$E_t = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & a_{ii}=1 & \cdots & a_{ij}=t & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & a_{ji}=0 & \cdots & a_{jj}=1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

The *Elementary Matrix* E_t to perform ERO_t on row i, using row j, with a multiplication factor of t, is obtained by modifying I_m (the $m \times m$ *Identity Matrix*) such that now $a_{ij} = t$. If matrix A is a square matrix of size $n \times n$ then *Elementary Matrix* E_t to perform ERO_t on row i, using row j, with a factor of t, is obtained by modifying the *Identity Matrix* I_n such that now $a_{ij} = t$.

Hence, if we perform the *matrix multiplication* $E_t A$ we obtain a new matrix A_3 as:

$$A_3 = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1i} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2i} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{j1} & a_{j2} & \cdots & a_{ji} & \cdots & a_{jj} & \cdots & a_{jn} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{i1}+ta_{j1} & a_{i2}+ta_{j2} & \cdots & a_{ii}+ta_{ji} & \cdots & a_{ij}+ta_{jj} & \cdots & a_{in}+ta_{jn} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mi} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix}$$

which is a transformation on original matrix A, such that, row i of original matrix A is transvected with a factor of t, using row j, to produce the new transformed matrix A_3.

Example 3.2 : Reducing a matrix using Elementary Matrices

Following example demonstrates transformation of a matrix into it's equivalent reduced form (R_x denotes *Row x*):

$$\begin{pmatrix} 2 & 8 & 7 \\ 4 & 5 & 6 \\ 3 & 2 & 9 \end{pmatrix}$$

$$\xrightarrow{E_d \text{ on } R_1 \text{ with } d=\frac{1}{2}} \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 8 & 7 \\ 4 & 5 & 6 \\ 3 & 2 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 4 & \frac{7}{2} \\ 4 & 5 & 6 \\ 3 & 2 & 9 \end{pmatrix}$$

$$\xrightarrow[E_t \text{ on } R_3 \text{ using } R_1 \text{ with } t=-3]{E_t \text{ on } R_2 \text{ using } R_1 \text{ with } t=-4} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 4 & \frac{7}{2} \\ 4 & 5 & 6 \\ 3 & 2 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 4 & \frac{7}{2} \\ 0 & -11 & -8 \\ 0 & -9 & -\frac{3}{2} \end{pmatrix}$$

$$\xrightarrow{E_d \text{ on } R_2 \text{ with } d=\frac{1}{-11}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{-11} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 4 & \frac{7}{2} \\ 0 & -11 & -8 \\ 0 & -9 & -\frac{3}{2} \end{pmatrix} = \begin{pmatrix} 1 & 4 & 7/2 \\ 0 & 1 & 8/11 \\ 0 & -9 & -3/2 \end{pmatrix}$$

$$\xrightarrow{E_t \text{ on } R_3 \text{ using } R_2 \text{ with } t=9} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 9 & 1 \end{pmatrix} \begin{pmatrix} 1 & 4 & 7/2 \\ 0 & 1 & 8/11 \\ 0 & -9 & -3/2 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 7/2 \\ 0 & 1 & 8/11 \\ 0 & 0 & 111/22 \end{pmatrix}$$

$$\xrightarrow{E_d \text{ on } R_3 \text{ with } d=\frac{22}{111}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{22}{111} \end{pmatrix} \begin{pmatrix} 1 & 4 & 7/2 \\ 0 & 1 & 8/11 \\ 0 & 0 & 111/22 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 7/2 \\ 0 & 1 & 8/11 \\ 0 & 0 & 1 \end{pmatrix}$$

It is not hard to understand that the above six *elementary matrices* can be combined as a product into a single ***Elementary Reduction Matrix*** as follows:

$$\begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{-11} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 9 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{22}{111} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ -4 & -\frac{1}{11} & 0 \\ -3 & 9 & \frac{22}{111} \end{pmatrix}$$

So, the complete reduction can be performed in a single product as:

$$\begin{pmatrix} \frac{1}{2} & 0 & 0 \\ -4 & -\frac{1}{11} & 0 \\ -3 & 9 & \frac{22}{111} \end{pmatrix} \begin{pmatrix} 2 & 8 & 7 \\ 4 & 5 & 6 \\ 3 & 2 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 7/2 \\ 0 & 1 & 8/11 \\ 0 & 0 & 1 \end{pmatrix}$$

3.3 Orthogonality and Orthonormality of Vectors and Matrices

Orthogonal Vectors : Two vectors x and y are said to be orthogonal if $x \cdot y = 0$. For example, two vectors $\begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 2 \\ 2 \\ 4 \end{pmatrix}$ are *orthogonal*, because their dot product $(1)(2) + (-1)(2) + (0)(4) = 0$. Vector **0** is orthogonal

to every vector.

A set of non-zero vectors $\{\mathbf{v_1}, \mathbf{v_2}, \cdots, \mathbf{v_k}\}$ is said to be ***Mutually Orthogonal*** if $\mathbf{v_i} \cdot \mathbf{v_j} = 0$ for all $i = j$. For example, the standard basis vectors $\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}$ are *mutually orthogonal*.

Unit Vector : A *Unit Vector* is a vector of length 1. If it's length is 1, then the square of it's length is also 1. So, $\mathbf{v}$ is a *unit vector* $\iff \mathbf{v} \cdot \mathbf{v} = 1$. For example, all the standard basis vectors $\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}$ are *unit vectors*.

Normalization of Vector : The process of replacing a vector by a *unit vector* in it's direction is called *Normalization of Vector*. An arbitrary non-zero vector $\mathbf{w}$ can be *normalized* by calculating:

$$\widehat{w} = \frac{1}{\|w\|_2} w$$

For example, if $v = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ then $\widehat{v} = \frac{1}{\|v\|_2} v = \frac{1}{\sqrt{x^2+y^2+z^2}} v$.

Orthonormal Matrices : A matrix Q is *Orthonormal* if $Q^T Q = I$. Few examples of *Orthonormal Matrices* are given below:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \qquad \begin{pmatrix} 1/3 & -2/3 \\ 2/3 & -1/3 \\ 2/3 & 2/3 \end{pmatrix}$$

Orthogonal Matrices : A square orthonormal matrix is known as *Orthogonal Matrix*. Hence, if Q is an *Orthogonal Matrix* then $Q^T Q = I$. But, we known that $Q^{-1} Q = I$. As such, it implies that, for an *Orthogonal Matrix* Q, $Q^{-1} = Q^T$. It also implies that if Q is *orthogonal* then Q^T as well as Q^{-1} are also *orthogonal*. Following are some examples of *Orthogonal Matrices*:

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \qquad \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \qquad \begin{pmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 & -1/2 \\ 1/2 & 1/2 & -1/2 & -1/2 \\ 1/2 & -1/2 & -1/2 & 1/2 \end{pmatrix}$$

The matrix $\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$ is not an *Orthogonal Matrix*. But we can adjust it to make it an *Orthogonal Matrix*: $\frac{1}{\sqrt{2}} \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$. (Such matrices are known as *Matrices with Equi-norm columns*. Observe that each column-vector in this matrix has same l_2-norm.)

3.4 Norm of Vectors and Matrices

Norm of Vector $\mathbf{x}$, denoted as $\|\mathbf{x}\|$, is any real number which satisfies the following properties:

1. $\|\mathbf{x}\| > 0$, if $\mathbf{x} \neq 0$, (*i.e.* at least one element of $\mathbf{x}$ is non-zero)
2. $\|k\mathbf{x}\| = |k|\|\mathbf{x}\|$, for any real scalar $\mathbf{k}$
3. $\|\mathbf{x} + \mathbf{y}\| \leqslant \|\mathbf{x}\| + \|\mathbf{y}\|$, for any two vectors $\mathbf{x}$ and $\mathbf{y}$

As you can see, based on this definition, *Norm of Vector* is just a real number associated with a given vector. As such, there are many possibilities to calculate *Norm of Vector*. But, out of those many, we usually use one of the following three norms: (a) l_1-Norm (b) l_2-Norm (c) l_∞-Norm. For a vector $\mathbf{x}$ of size n, these values are defined as follows, respectively:

- $\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$, where $|x_i|$ denotes the absolute value of x_i
- $\|\mathbf{x}\|_2 = +\sqrt{x^T \cdot x} = +\sqrt{\sum_{i=1}^{n} x_i^2}$, where x^T denotes the transpose of vector $\mathbf{x}$
- $\|\mathbf{x}\|_\infty = \max\limits_{1 \leqslant i \leqslant n} |x_i|$

Example 3.3 : **Norm of vector**

As an example, the three norms for the vector $\mathbf{v} = (1, -2, 3, -4)$ are:

- $l_1 = |1| + |-2| + |3| + |-4| = 10$
- $l_2 = +\sqrt{(1)^2 + (-2)^2 + (3)^2 + (-4)^2} = 5.477$
- $l_\infty = max(|1|, |-2|, |3|, |-4|) = 4$

Norm of Square Matrix A, denoted as $\|A\|$, is any real number which satisfies the following properties:

1. $\|A\| \geqslant 0$
2. $\|A\| = 0$, if and only if $A = 0$, (*i.e.* A is a zero matrix)
3. $\|kA\| = |k|\|A\|$, for any real scalar $\mathbf{k}$
4. $\|A + B\| \leqslant \|A\| + \|B\|$, for any two matrices A and B
5. $\|AB\| \leqslant \|A\|\|B\|$, for any two matrices A and B

Just as in case of vectors, based on this definition, *Norm of Matrix* is just a real number associated with a given matrix. As such, there are many possibilities to calculate *Norm of Matrix*. But, out of those many, we usually use one of the following three norms: (a) l_1-Norm (b) l_2-Norm (c) l_∞-Norm.

For a square matrix A of size $n \times n$, these values are defined as follows, respectively:

- $\|A\|_1 = \max\limits_{1 \leqslant j \leqslant n} (\sum_{i=1}^{n} |a_{ij}|)$, where $|a_{ij}|$ denotes absolute value of a_{ij}, element in i^{th} row j^{th} column of A
- $\|A\|_2 = +\sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2}$
- $\|A\|_\infty = \max\limits_{1 \leqslant i \leqslant n} (\sum_{j=1}^{n} |a_{ij}|)$

Hence, l_1-Norm is "maximum absolute column sum", which means, we sum the absolute values along each column and then take the largest answer. And, l_∞-Norm is "maximum absolute row sum", which means, we sum the absolute values along each row and then take the largest answer.

l_2-Norm is also known as ***Euclidean Norm***, and as such, also denoted as l_E-Norm (or, for a given matrix A as $\|A\|_E$).

Example 3.4 : Norm of square matrix

As an example, the three norms for the matrix $A = \begin{bmatrix} 1 & -2 & 3 \\ -4 & 5 & -6 \\ 7 & -8 & 9 \end{bmatrix}$ are:

- $l_1 = max(|1| + |-4| + |7|, |-2| + |5| + |-8|, |3| + |-6| + |9|) = 18$
- $l_2 = +\sqrt{(1)^2 + (-2)^2 + (3)^2 + (-4)^2 + (5)^2 + (-6)^2 + (7)^2 + (-8)^2 + (9)^2} = 16.882$
- $l_\infty = max(|1| + |-2| + |3|, |-4| + |5| + |-6|, |7| + |-8| + |9|) = 24$

Matrix Inversion

A square matrix whose *rank* is less than the count of rows in the matrix is known as a ***Singular Matrix***. For any *non-singular matrix* M of size n there exist another *non-singular matrix* M^{-1} of size n such that $M^{-1}M = I_n$. We say M^{-1} is the ***Inverse*** of matrix M.

For a given *non-singular matrix* M the corresponding *inverse* M^{-1} can be found by systematic application of *Elementary Row Operations* on M. The procedure can be best illustrated with an example.

Example 3.5 : Matrix Inversion

Let us find the *inverse* of matrix A, as given below:

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 3 & 2 & 4 \\ 1 & -1 & 0 \end{pmatrix}$$

To start, we write:

$$A \mid I$$

Now, we perform a series of *Elementary Row Operations* on both the matrices, simultaneously, on either side of the middle bar, as follows:

$$\begin{pmatrix} -1 & 1 & 1 \\ 3 & 2 & 4 \\ 1 & -1 & 0 \end{pmatrix} \mid \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$\xrightarrow{E_d \text{ on } R_1 \text{ with } d=-1}$

$$\begin{pmatrix} 1 & -1 & -1 \\ 3 & 2 & 4 \\ 1 & -1 & 0 \end{pmatrix} \mid \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$\xrightarrow{\substack{E_t \text{ on } R_2 \text{ using } R_1 \text{ with } t=-3 \\ E_t \text{ on } R_3 \text{ using } R_1 \text{ with } t=-1}}$

$$\begin{pmatrix} 1 & -1 & -1 \\ 0 & 5 & 7 \\ 0 & 0 & 1 \end{pmatrix} \mid \begin{pmatrix} -1 & 0 & 0 \\ 3 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$\xrightarrow{E_d \text{ on } R_2 \text{ with } d=\frac{1}{5}}$

$$\begin{pmatrix} 1 & -1 & -1 \\ 0 & 1 & \frac{7}{5} \\ 0 & 0 & 1 \end{pmatrix} \mid \begin{pmatrix} -1 & 0 & 0 \\ \frac{3}{5} & \frac{1}{5} & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$\xrightarrow{E_t \text{ on } R_1 \text{ using } R_2 \text{ with } t=1}$

$$\begin{pmatrix} 1 & 0 & \frac{2}{5} \\ 0 & 1 & \frac{7}{5} \\ 0 & 0 & 1 \end{pmatrix} \mid \begin{pmatrix} -\frac{2}{5} & \frac{1}{5} & 0 \\ \frac{3}{5} & \frac{1}{5} & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$\xrightarrow{\substack{E_t \text{ on } R_1 \text{ using } R_3 \text{ with } t=-\frac{2}{5} \\ E_t \text{ on } R_2 \text{ using } R_3 \text{ with } t=-\frac{7}{5}}}$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mid \begin{pmatrix} -\frac{4}{5} & \frac{1}{5} & -\frac{2}{5} \\ -\frac{4}{5} & \frac{1}{5} & -\frac{7}{5} \\ 1 & 0 & 1 \end{pmatrix}$$

As can be observed, the aim was to reduce the left matrix (i.e. A) to *Identity Matrix*, I. While reducing the left matrix A to I, the EROs transformed the original I on right side of bar to A^{-1}.

Hence, the inverse of matrix $A = \begin{pmatrix} -1 & 1 & 1 \\ 3 & 2 & 4 \\ 1 & -1 & 0 \end{pmatrix}$ is $A^{-1} = \begin{pmatrix} -\frac{4}{5} & \frac{1}{5} & -\frac{2}{5} \\ -\frac{4}{5} & \frac{1}{5} & -\frac{7}{5} \\ 1 & 0 & 1 \end{pmatrix}$

3.5 Linear Equations

The subject of algebra arose from studying equations. Simplest general form of ***Linear Equation*** is $ax = b$. The letter x is the variable, and a and b are arbitrary constants (*i.e.* fixed numbers).

For example, consider $4x = 3$. The solution is $x = 3/4$. In general, if $a \neq 0$, then $x = b/a$, and this solution is unique. But, if $a = 0$ and $b \neq 0$, then, there is no solution, since the equation says $0 = b$, which, of course, is absurd. And, lastly, for the case where a and b are both 0, every real number x is a solution.

This points out a general property of linear equations. *For any given linear equation, with regards to it's solution, there are only three possibilities, and they are mutually exclusive: (i) Either there is a unique solution (i.e. exactly one), or (ii) There is no solution, or (iii) There are infinitely many solutions.*

Equations such as $z = x^2 + xy^5$ and $z^2 = x + y^4$ represent ***Non-linear Equations***, which, compared to linear equations, are difficult to solve; their theory involves highly sophisticated mathematics.

$ax = b$ is ***monomial*** form of *linear equation*. When more than one variable is involved, *i.e.* $a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$, we call it a ***Polynomial Linear Equation***.

3.6 Systems of Linear Equations

We discussed in above paragraph, a single linear equation. When there are more than one linear equation, all referring to the same set of variables, we call them as a ***System of Linear Equations***.

A general linear system consisting of m equations in n unknowns will look like:

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n = b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n = b_2 \\ \vdots & + & \vdots & + & \cdots & + & \vdots = \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n = b_m \end{array}$$

The case where all the constants b_i are zero is called ***Homogeneous System of Linear Equations***. Otherwise, the system is said to be ***Non-homogeneous System of Linear Equations***.

Hence, a system of linear equations can be represented as $A\mathbf{x} = b$, where A is the ***Coefficient Matrix***, x is the ***Variables Vector***, and b is the ***Constants Vector***, and they are defined as follows:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \text{and} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

The ***Augmented Coefficient Matrix*** is written and defined as:

$$(A|b) = \left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right)$$

3.6.1 Existence and Uniqueness of Solutions to System of Linear Equations

Suppose we have m equations in n variables, then we can write such a system of linear equations in matrix form as $A\mathbf{x} = \mathbf{b}$, where A is the coefficient matrix, x is the variables vector, and b is the constants vector. Clearly, the sizes of A, x and b are $m \times n$, $m \times 1$ and $m \times 1$, respectively. For this system, to have a solution exist, *ranks* of A (the coefficient matrix) and $(A|b)$ (the augmented matrix) must be same. (It should be noted that *rank* cannot exceed the minimum of m and n.)

For example, consider the following system of linear equations:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ -2 \\ 5 \end{pmatrix}$$

The reduced form of these equations is:

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -3 \\ 2 \\ 10 \end{pmatrix}$$

Here, the *rank* of *coefficient matrix* is 2 (recall that *rank* is *number of pivot values in reduced form*), but the *rank* of *augmented matrix* is 3. Hence, this system does not have a solution (which is also evident from the third equation, which is says $0 = 10$, which is, of course, absurd.) Such systems are known as ***Inconsistent Systems***.

On the contrary, ***Consistent Systems*** (where *ranks* of *coefficient matrix* and *augmented matrix* are equal) have *one or more* solutions. For example, consider the following system of linear equations:

$$\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}=\begin{pmatrix}1\\2\\3\end{pmatrix}$$

The reduced form of these equations is:

$$\begin{pmatrix}1&0&-1\\0&1&2\\0&0&0\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}=\begin{pmatrix}-1/3\\2/3\\0\end{pmatrix}$$

Here, the *rank* of *coefficient matrix* as well as *augmented matrix* is 2. Hence, this is a *consistent system* and has *more than one solutions* (actually, infinitely many.) To find the solution, we must express any two variables of the system in terms of third variable. Suppose we decided to express x_2 and x_3 in terms of x_1, then, the solution is $x_2=-2x_1$ and $x_3=x_1+\frac{1}{3},\quad \forall x_1$.

Practically, we are much more interested in systems of linear equations where *coefficient matrix* is of size $n\times n$. When coefficient matrix is a square matrix then it's *rank* can reach a maximum value of n (and, similarly, the *rank* of *augmented matrix* also can reach a maximum value of n). Hence, for a given system of linear equations, if the *rank* of *coefficient matrix* is n then, obviously, the *rank* of *augmented matrix* is also n. Such a matrix falls into the category of ***Non-Singular Matrix*** (Also, *determinant* of *Singular Matrix* is *zero*). ***If the coefficient matrix is non-singular, then, not only solution exists but it will be unique.***

For example, consider the following system of linear equations:

$$\begin{pmatrix}1&8&7\\2&9&6\\3&4&5\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}\begin{pmatrix}5\\3\\1\end{pmatrix}$$

The reduced form of these equations is:

$$\begin{pmatrix}1&8&7\\0&1&8/7\\0&0&48/7\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}\begin{pmatrix}5\\1\\6\end{pmatrix}$$

Here, the *rank* of *coefficient matrix* as well as *augmented matrix* is 3, which is also the *size* of matrix. Hence, this is a *consistent system* and has *unique solution*. To find the solution, we solve the *reduced form* of equations, to get, $x_1=-\frac{9}{8}$, $x_2=0$ and $x_3=\frac{7}{8}$.

Ill-Conditioned Systems of Linear Equations: We have seen that *Singular Matrices* impose *non-existence of solution*. But, equally dangerous are *Ill-conditioned Matrices* (which consequently imply *Ill-conditioned Systems of Linear Equations*) whose determinant is close to zero. Such systems are very sensitive to small changes in values of individual variables of equations, which means, even a small change in value of a single variable will change the determinant of the coefficient matrix drastically, and as such, it becomes very important to check whether a given solution produced by some numerical method is acceptable or not.

To measure this sensitivity we define ***Condition Number,*** κ for a given system of of linear equations (in terms of it's coefficient matrix A) as follows:

$$\kappa(A)=\|A\|\|A^{-1}\|$$

Condition number is always greater than 1 ($\because\ \kappa(A)=\|A\|\|A^{-1}\|\geqslant\|AA^{-1}\|=\|I\|=1$). Values of the condition number close to 1 indicate a ***well-conditioned matrix***, whereas large values indicate an ***ill-conditioned matrix***.

To illustrate the concept of *Ill-conditioning*, let us take the following example of system of linear equations:

$$\begin{pmatrix}1&10^4\\-1&2\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix}=\begin{pmatrix}10^4\\1\end{pmatrix}$$

To make the situation practical, let us assume that our computer has precision for first three significant digits only (and it neglects the fourth digit and above as *round-off* error.)

Now, the reduced form of these equations is:

$$\begin{pmatrix}1&10^4\\0&10^4\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix}=\begin{pmatrix}10^4\\10^4\end{pmatrix}$$

This happened because our computer approximated the calculation $2+10^4\approx10^4$ to the third significant digit (instead of the correct value $2+10^4=10002$ in which precision is involved at fourth significant digit.) Solving the reduced set of equations yield $x_1=0$ and $x_2=1$. For sure, this is a poor approximation of true solution. So, let us see what is the *condition number* for this set of equations:

$$\begin{aligned}\kappa(A)&=\|A\|_1\|A^{-1}\|_1\\&=\left\|\begin{pmatrix}1&10^4\\-1&2\end{pmatrix}\right\|\left\|\frac{1}{2+10^4}\begin{pmatrix}2&-10^4\\1&1\end{pmatrix}\right\|\\&=(2+10^4)\times\tfrac{1}{2+10^4}(1+10^4)\\&=10001\end{aligned}$$

The large value of *condition number* is clearly indicating that the system is *ill-conditioned*.

3.7 Numerical Methods to solve Systems of Linear Equations

As shown in Figure 3.1 (on page 32), *Numerical Methods* to solve *Systems of Linear Equations* are divided into two categories: ***Direct Methods*** and ***Indirect Methods***.

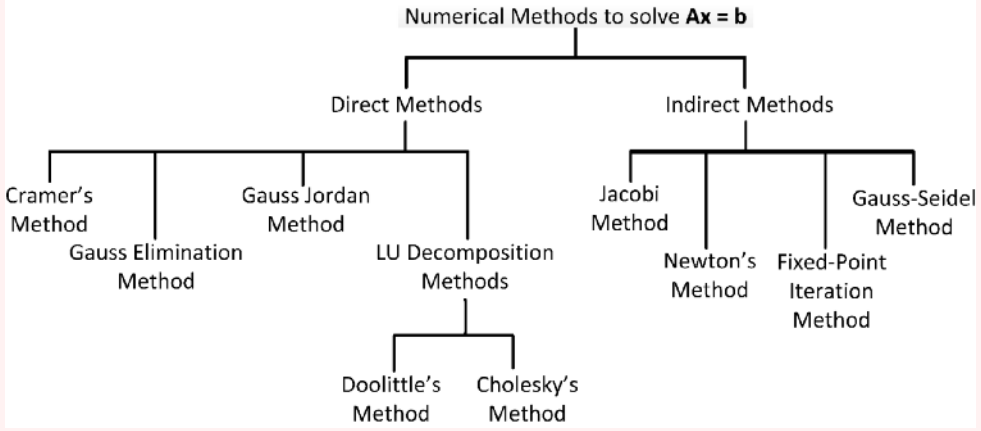

Figure (3.1) – Categorization of Numerical Methods to solve Systems of Equations

A direct method computes the solution by performing a predetermined number of operations. These methods transform the original system into an equivalent system in which the coefficient matrix becomes an upper-triangular matrix, a lower-triangular matrix, or diagonal matrix, thereby making the new system much easier to solve.

Indirect methods use iterations to approximate the solution. The iteration process begins with an initial vector and generates successive approximations that eventually converge to the actual solution. Unlike direct methods, the number of iterations, and thus the number of operations, required for convergence of solution, is not known in advance.

Among *Direct Methods* we will study ***Cramer's Method***, ***Gauss Elimination Method*** and ***Gauss-Jordan Method***. And, from *Indirect Methods* we will cover ***Newton's Method*** and ***Fixed-Point Iteration Method***.

3.7.1 Cramer's Method

Cramer's Method is systematic approach to solving system of linear equations. It can solve only non-singular system of equations.

To use Cramer's Method write the given system of equations as $\mathbf{a_1}x_1 + \mathbf{a_2}x_2 + \cdots + \mathbf{a_n}x_n = \mathbf{b}$, where $x_k(\forall k, \quad 1 \leqslant k \leqslant n)$ are the variables of the equations, and $\mathbf{a_k}(\forall k, \quad 1 \leqslant k \leqslant n)$ are the column-vectors for coefficients of corresponding variable $x_k(\forall k, \quad 1 \leqslant k \leqslant n)$, and $\mathbf{b}$ is the column-vector for constants (RHS of equations). Call matrix A as $[\mathbf{a_1} \quad \mathbf{a_2} \quad \cdots \quad \mathbf{a_n}]$, matrix A_1 as $[\mathbf{b} \quad \mathbf{a_2} \quad \cdots \quad \mathbf{a_n}]$, matrix A_2 as $[\mathbf{a_1} \quad \mathbf{b} \quad \cdots \quad \mathbf{a_n}]$, and so on. To generalize, call matrix A_k as $[\mathbf{a_1} \quad \mathbf{a_2} \quad \cdots \quad \mathbf{a_{k-1}} \quad \mathbf{b} \quad \mathbf{a_{k+1}} \quad \mathbf{a_n}]$, which is the modified matrix A in which the column corresponding to $\mathbf{a_k}$,the coefficient column-vector of x_k, is replaced by constant column-vector $\mathbf{b}$.

The solution to the system of linear equations is given by:

$$x_1 = \frac{|A_1|}{|A|}, \qquad x_2 = \frac{|A_2|}{|A|}, \quad \cdots \quad x_k = \frac{|A_k|}{|A|}, \quad \cdots \quad x_n = \frac{|A_n|}{|A|}$$

where $|A_k|$ denotes the determinant of matrix A_k.

Example 3.6 : Cramer's Method for Two Equations

Solve the following system of equations using Cramer's Method:

$$\begin{aligned} 2x_1 &+ 3x_2 &&= 5 \\ 3x_1 &- 4x_2 &&= -12 \end{aligned}$$

Solution : Write down the matrices and vectors:

$$A = \begin{pmatrix} 2 & 3 \\ 3 & -4 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 5 \\ -12 \end{pmatrix} \quad A_1 = \begin{pmatrix} 5 & 3 \\ -12 & -4 \end{pmatrix} \quad A_2 = \begin{pmatrix} 2 & 5 \\ 3 & -12 \end{pmatrix}$$

Solve the variables:

$$x_1 = \frac{|A_1|}{|A|} = \frac{\begin{vmatrix} 5 & 3 \\ -12 & -4 \end{vmatrix}}{\begin{vmatrix} 2 & 3 \\ 3 & -4 \end{vmatrix}} = \frac{(5)(-4) - (3)(-12)}{(2)(-4) - (3)(3)} = \frac{-20 + 36}{-8 - 9} = -\frac{16}{17}$$

$$x_2 = \frac{|A_2|}{|A|} = \frac{\begin{vmatrix} 2 & 5 \\ 3 & -12 \end{vmatrix}}{\begin{vmatrix} 2 & 3 \\ 3 & -4 \end{vmatrix}} = \frac{(2)(-12)-(5)(3)}{(2)(-4)-(3)(3)} = \frac{-24-15}{-8-9} = \frac{39}{17}$$

Example 3.7 : Cramer's Method for Three Equations

Solve the following system of equations using Cramer's Method:

$$\begin{array}{rcrcrcl} 2x & - & 3y & + & 4z & = & 1 \\ x & + & y & - & z & = & 2 \\ -x & & & + & z & = & 1 \end{array}$$

Solution : Write down the matrices and vectors:

$$A = \begin{pmatrix} 2 & -3 & 4 \\ 1 & 1 & -1 \\ -1 & 0 & 1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

Solve the variables:

$$x = \frac{\begin{vmatrix} 1 & -3 & 4 \\ 2 & 1 & -1 \\ 1 & 0 & 1 \end{vmatrix}}{\begin{vmatrix} 2 & -3 & 4 \\ 1 & 1 & -1 \\ -1 & 0 & 1 \end{vmatrix}} = \frac{(1)(1)(1)+(-3)(-1)(1)+(4)(2)(0)-(4)(1)(1)-(-3)(2)(1)-(1)(-1)(0)}{(2)(1)(1)+(-3)(-1)(-1)+(4)(1)(0)-(4)(1)(-1)-(-3)(1)(1)-(2)(-1)(0)} = 1$$

$$y = \frac{\begin{vmatrix} 2 & 1 & 4 \\ 1 & 2 & -1 \\ -1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} 2 & -3 & 4 \\ 1 & 1 & -1 \\ -1 & 0 & 1 \end{vmatrix}} = \frac{(2)(2)(1)+(1)(-1)(-1)+(4)(1)(1)-(4)(2)(-1)-(1)(1)(1)-(2)(-1)(1)}{(2)(1)(1)+(-3)(-1)(-1)+(4)(1)(0)-(4)(1)(-1)-(-3)(1)(1)-(2)(-1)(0)} = 3$$

$$z = \frac{\begin{vmatrix} 2 & -3 & 1 \\ 1 & 1 & 2 \\ -1 & 0 & 1 \end{vmatrix}}{\begin{vmatrix} 2 & -3 & 4 \\ 1 & 1 & -1 \\ -1 & 0 & 1 \end{vmatrix}} = \frac{(2)(1)(1)+(-3)(2)(-1)+(1)(1)(0)-(1)(1)(-1)-(-3)(1)(1)-(2)(2)(0)}{(2)(1)(1)+(-3)(-1)(-1)+(4)(1)(0)-(4)(1)(-1)-(-3)(1)(1)-(2)(-1)(0)} = 2$$

3.7.2 Gauss Elimination Method

Gauss Elimination Method uses systematic approach to transform the *Augmented Coefficient Matrix* of *System of Linear Equations* into it's equivalent *Reduced Row Echelon form*. This process is known as ***Forward Elimination Process***.

Once in it's *Reduced Form* we can start solving for the equation with single variable, then substitute the variable's value in the equation with two variables and solve for the second variable. By substituting these two variables in the next equation (with three variables) the third variable can be solved. Continuing this process results in resolution of all the variables. This step is known as ***Backward Substitution***.

Hence, solution using *Gauss Elimination Process* involves the two steps : *Forward Elimination process* and *Backward Substitution process*.

Example 3.8 : Gauss Elimination Method for three equations

Solve the following system of equations using *Gauss Elimination Method*:

$$\begin{array}{rcrcrcl} 2x & - & 3y & + & 4z & = & 1 \\ x & + & y & - & z & = & 2 \\ -x & & & + & z & = & 1 \end{array}$$

Solution : Write down the *Augmented Coefficient Matrix* and start *Forward Elimination Process*:

$$\left(\begin{array}{ccc|c} 2 & -3 & 4 & 1 \\ 1 & 1 & -1 & 2 \\ -1 & 0 & 1 & 1 \end{array}\right)$$

$\xrightarrow{E_d \text{ on } R_1 \text{ with } d=\frac{1}{2}}$

$$\left(\begin{array}{ccc|c} 1 & -3/2 & 2 & 1/2 \\ 1 & 1 & -1 & 2 \\ -1 & 0 & 1 & 1 \end{array}\right)$$

$\xrightarrow{\substack{E_t \text{ on } R_2 \text{ using } R_1 \text{ with } t=-1 \\ E_t \text{ on } R_3 \text{ using } R_1 \text{ with } t=1}}$

$$\left(\begin{array}{ccc|c} 1 & -3/2 & 2 & 1/2 \\ 0 & 5/2 & -3 & 3/2 \\ 0 & -3/2 & 3 & 3/2 \end{array}\right)$$

$$\xrightarrow{E_d \text{ on } R_2 \text{ with } d=\frac{2}{5}} \left(\begin{array}{ccc|c} 1 & -3/2 & 2 & 1/2 \\ 0 & 1 & -6/5 & 3/5 \\ 0 & -3/2 & 3 & 3/2 \end{array}\right)$$

$$\xrightarrow{E_t \text{ on } R_3 \text{ using } R_2 \text{ with } t=\frac{3}{2}} \left(\begin{array}{ccc|c} 1 & -3/2 & 2 & 1/2 \\ 0 & 1 & -9/5 & 3/5 \\ 0 & 0 & 6/5 & 12/5 \end{array}\right)$$

$$\xrightarrow{E_d \text{ on } R_3 \text{ with } d=\frac{5}{6}} \left(\begin{array}{ccc|c} 1 & -3/2 & 2 & 1/2 \\ 0 & 1 & -6/5 & 3/5 \\ 0 & 0 & 1 & 2 \end{array}\right)$$

Now start the *Backward Substitution Process:*

$$z = 2$$

$$y - \frac{6}{5}z = \frac{3}{5} \quad \Rightarrow \quad y = \frac{3}{5} - (-\frac{6}{5})(2) = 3$$

$$x - \frac{3}{2}y + 2z = \frac{1}{2} \quad \Rightarrow \quad x = \frac{1}{2} - \big((-\frac{3}{2})(3) + (2)(2)\big) = 1$$

Example 3.9 : Gauss Elimination Method for five equations

Solve the following system of equations using Gauss Elimination Method:

$$\begin{array}{rcrcrcrcr} 2a & - & 5b & + & 4c & - & 9d & = & -7 \\ a & + & b & + & c & + & d & - & e & = & 14 \\ a & + & 2b & + & 3c & + & 4d & - & 10e & = & 36 \\ 8a & - & 5b & - & 3c & & & + & e & = & 10 \\ a & - & 3b & + & 3c & - & 7d & - & e & = & -4 \end{array}$$

Solution : Write down the *Augmented Coefficient Matrix* and start *Forward Elimination Process*:

$$\left(\begin{array}{ccccc|c} 2 & -5 & 4 & -9 & 0 & -7 \\ 1 & 1 & 1 & 1 & -1 & 14 \\ 1 & 2 & 3 & 4 & -10 & 36 \\ 8 & -5 & -3 & 0 & 1 & 10 \\ 1 & -3 & 3 & -7 & -1 & -4 \end{array}\right)$$

$$\xrightarrow{E_d \text{ on } R_1 \text{ with } d=\frac{1}{2}} \left(\begin{array}{ccccc|c} 1 & -5/2 & 2 & -9/2 & 0 & -7/2 \\ 1 & 1 & 1 & 1 & -1 & 14 \\ 1 & 2 & 3 & 4 & -10 & 36 \\ 8 & -5 & -3 & 0 & 1 & 10 \\ 1 & -3 & 3 & -7 & -1 & -4 \end{array}\right)$$

$$\xrightarrow[\substack{E_t \text{ on } R_4 \text{ using } R_1 \text{ with } t=-8 \\ E_t \text{ on } R_5 \text{ using } R_1 \text{ with } t=-1}]{\substack{E_t \text{ on } R_2 \text{ using } R_1 \text{ with } t=-1 \\ E_t \text{ on } R_3 \text{ using } R_1 \text{ with } t=-1}} \left(\begin{array}{ccccc|c} 1 & -5/2 & 2 & -9/2 & 0 & -7/2 \\ 0 & 7/2 & -1 & 11/2 & -1 & 35/2 \\ 0 & 9/2 & 1 & 17/2 & -10 & 79/2 \\ 0 & 15 & -19 & 36 & 1 & 38 \\ 0 & -1/2 & 1 & -5/2 & -1 & -1/2 \end{array}\right)$$

$$\xrightarrow{E_d \text{ on } R_2 \text{ with } d=\frac{2}{7}} \left(\begin{array}{ccccc|c} 1 & -5/2 & 2 & -9/2 & 0 & -7/2 \\ 0 & 1 & -2/7 & 11/7 & -2/7 & 5 \\ 0 & 9/2 & 1 & 17/2 & -10 & 79/2 \\ 0 & 15 & -19 & 36 & 1 & 38 \\ 0 & -1/2 & 1 & -5/2 & -1 & -1/2 \end{array}\right)$$

$$\xrightarrow[\substack{E_t \text{ on } R_4 \text{ using } R_2 \text{ with } t=-15 \\ E_t \text{ on } R_5 \text{ using } R_2 \text{ with } t=\frac{1}{2}}]{E_t \text{ on } R_3 \text{ using } R_2 \text{ with } t=-\frac{9}{2}} \left(\begin{array}{ccccc|c} 1 & -5/2 & 2 & -9/2 & 0 & -7/2 \\ 0 & 1 & -2/7 & 11/7 & -2/7 & 5 \\ 0 & 0 & 16/7 & 10/7 & -61/7 & 17 \\ 0 & 0 & -103/7 & 87/7 & 37/7 & -37 \\ 0 & 0 & 6/7 & -12/7 & -8/7 & 2 \end{array}\right)$$

$$\xrightarrow{E_d \text{ on } R_3 \text{ with } d=\frac{7}{16}} \left(\begin{array}{ccccc|c} 1 & -5/2 & 2 & -9/2 & 0 & -7/2 \\ 0 & 1 & -2/7 & 11/7 & -2/7 & 5 \\ 0 & 0 & 1 & 5/8 & -61/16 & 119/16 \\ 0 & 0 & -103/7 & 87/7 & 37/7 & -37 \\ 0 & 0 & 6/7 & -12/7 & -8/7 & 2 \end{array}\right)$$

$$\xrightarrow[E_t \text{ on } R_5 \text{ using } R_3 \text{ with } t=-\frac{6}{7}]{E_t \text{ on } R_4 \text{ using } R_3 \text{ with } t=\frac{103}{7}} \left(\begin{array}{ccccc|c} 1 & -5/2 & 2 & -9/2 & 0 & -7/2 \\ 0 & 1 & -2/7 & 11/7 & -2/7 & 5 \\ 0 & 0 & 1 & 5/8 & -61/16 & 119/16 \\ 0 & 0 & 0 & 173/8 & -813/16 & 1159/16 \\ 0 & 0 & 0 & -9/4 & 17/8 & -35/8 \end{array}\right)$$

$$\xrightarrow{E_d \text{ on } R_4 \text{ with } d=\frac{8}{173}} \left(\begin{array}{ccccc|c} 1 & -5/2 & 2 & -9/2 & 0 & -7/2 \\ 0 & 1 & -2/7 & 11/7 & -2/7 & 5 \\ 0 & 0 & 1 & 5/8 & -61/16 & 119/16 \\ 0 & 0 & 0 & 1 & -813/346 & 1159/346 \\ 0 & 0 & 0 & -9/4 & 17/8 & -35/8 \end{array}\right)$$

$$\xrightarrow{E_t \text{ on } R_5 \text{ using } R_4 \text{ with } t=\frac{9}{4}} \left(\begin{array}{ccccc|c} 1 & -5/2 & 2 & -9/2 & 0 & -7/2 \\ 0 & 1 & -2/7 & 11/7 & -2/7 & 5 \\ 0 & 0 & 1 & 5/8 & -61/16 & 119/16 \\ 0 & 0 & 0 & 1 & -813/346 & 1159/346 \\ 0 & 0 & 0 & 0 & -547/173 & 547/173 \end{array}\right)$$

$$\xrightarrow{E_d \text{ on } R_5 \text{ with } d=\frac{173}{547}} \left(\begin{array}{ccccc|c} 1 & -5/2 & 2 & -9/2 & 0 & -7/2 \\ 0 & 1 & -2/7 & 11/7 & -2/7 & 5 \\ 0 & 0 & 1 & 5/8 & -61/16 & 119/16 \\ 0 & 0 & 0 & 1 & -813/346 & 1159/346 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{array}\right)$$

Now start the *Backward Substitution Process:*

$$e = -1$$

$$d - \frac{813}{346}e = \frac{1159}{346} \quad \Rightarrow \quad d = \frac{1159}{346} - \left(-\frac{813}{346}\right)(-1) = 2$$

$$c + \frac{5}{8}d - \frac{61}{16}e = \frac{119}{16} \quad \Rightarrow \quad c = \frac{119}{16} - \left[\left(\frac{5}{8}\right)(1) - \left(\frac{61}{16}\right)(-1)\right] = 3$$

$$b - \frac{2}{7}c + \frac{11}{7}d - \frac{2}{7}e = 5 \quad \Rightarrow \quad b = 5 - \left[\left(-\frac{2}{7}\right)(3) + \left(\frac{11}{7}\right)(1) + \left(-\frac{2}{7}\right)(-1)\right] = 4$$

$$a - \frac{5}{2}b + 2c - \frac{9}{2}d = -\frac{7}{2} \quad \Rightarrow \quad a = -\frac{7}{2} - \left[\left(-\frac{5}{2}\right)(4) + (2)(3) + \left(-\frac{9}{2}\right)(1)\right] = 5$$

3.8 Numerical Methods to solve Systems of Non-Linear Equations

Systems of Non-linear Equations can be solved numerically by using either ***Newton's Method*** (for small systems) or the ***Fixed-Point Iteration Method*** (for large systems)

3.8.1 Newton's Method to solve System of Non-Linear Equations

Newton's Method to find roots of equation was discussed in Section 2.4.2 (on page 18). An extension of that technique can be used for solving a system of nonlinear equations. Let us solve first system of two nonlinear equations, followed by a general system of n nonlinear equations.

Solution to System of Two Equations

A system of two (nonlinear) equations in two unknowns can generally be expressed as

$$\begin{aligned} f_1(x,y) &= 0 \\ f_2(x,y) &= 0 \end{aligned}$$

Suppose (x_a, y_a) denotes the actual solution so that $f_1(x_a, y_a) = 0$ and $f_2(x_a, y_a) = 0$. We begin with an initial estimate of solution as (x_e, y_e). If x_e is sufficiently close to x_a, and y_e is sufficiently close to y_a, then, by Taylor's series expansion (ignoring second and higher order terms):

$$\begin{aligned} f_1(x_a,y_a) &= f_1(x_e,y_e) + \left.\frac{\partial f_1}{\partial x}\right|_{(x_e,y_e)} \Delta x + \left.\frac{\partial f_1}{\partial y}\right|_{(x_e,y_e)} \Delta y \\ f_2(x_a,y_a) &= f_2(x_e,y_e) + \left.\frac{\partial f_2}{\partial x}\right|_{(x_e,y_e)} \Delta x + \left.\frac{\partial f_2}{\partial y}\right|_{(x_e,y_e)} \Delta y \end{aligned} \tag{3.1}$$

where $\Delta x = x_a - x_e$ and $\Delta y = y_a - y_e$.

After substituting $f_1(x_a, y_a) = 0$, $f_2(x_a, y_a) = 0$, and rearranging the terms, the above two equations can be written in matrix form as:

$$\left[\begin{array}{cc} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{array}\right]_{(x_e,y_e)} \left\{\begin{array}{c} \Delta x \\ \Delta y \end{array}\right\} = \left\{\begin{array}{c} -f_1 \\ -f_2 \end{array}\right\}_{(x_e,y_e)}$$

Now, system of nonlinear equations is transformed into equivalent system of linear equations! Hence, *Cramer's Method* or *Gauss Elimination Method* can be employed to find the values of Δx and Δy.

This means that we should be able to find the correct solution (x_n, y_n) from the approximate guess (x_e, y_e) by substituting $(x_a, y_a) = (x_e + \Delta x, y_e + \Delta y)$. But, because we did not consider higher order terms while expanding $f_1(x_a, y_a)$ using Taylor's series, we will get only a better approximate guess as $(x_a, y_a) \approxeq (x_e + \Delta x, y_e + \Delta y)$.

But, also, surely $(x_e + \Delta x, y_e + \Delta y)$ is nearer to correct solution (x_a, y_a) as compared to our initial guess (x_e, y_e). This suggests that we can repeat the process iteratively to find even better guesses, and reach a final guess which is close enough to the correct value within an allowed tolerance.

Hence, re-initialize the next guess as $(x_e, y_e) = (x_a, y_a)$ and calculate new (x_a, y_a) using Equation 3.1 (on page 35). In every iteration calculate $\epsilon = \|(\Delta x, \Delta y)\| = +\sqrt{\Delta x^2 + \Delta y^2}$ and repeat till $\epsilon >$ tolerance.

Example 3.10 : Newton's Method to solve system of non-linear equations

Example : Solve the following set of nonlinear equations using *Newton's Method* with a tolerance of 10^{-3}.

$$\begin{aligned} 3.2x^3 + 1.8y^2 + 24.43 &= 0 \\ -2x^2 + 3y^3 &= 5.92 \end{aligned}$$

Solution : Let us name our equations as follows:

$$p = 3.2x^3 + 1.8y^2 + 24.43 \qquad q = 2x^2 - 3y^3 + 5.92$$

Also, let us name the derivatives as follows:

$$a = \frac{\partial p}{\partial x} = 9.6x^2 \qquad b = \frac{\partial p}{\partial y} = 3.6y \qquad c = \frac{\partial q}{\partial x} = 4x \qquad d = \frac{\partial q}{\partial y} = -9y^2$$

So, our aim is to solve the matrix equation :

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}_{(x_e, y_e)} \begin{Bmatrix} \Delta x \\ \Delta y \end{Bmatrix} = \begin{Bmatrix} -p \\ -q \end{Bmatrix}_{(x_e, y_e)}$$

For easier reference let us define following three matrices:

$$C = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad C_x = \begin{bmatrix} -p & b \\ -q & d \end{bmatrix} \qquad C_y = \begin{bmatrix} a & -p \\ c & -q \end{bmatrix}$$

It is critical for convergence of solution to start with an initial guess as close to the actual solution as possible. A good approach for initial guess is to plot the graphs of the functions and guess the initial value approximate to their point of intersection (Any software application, such as MATLAB or Microsoft Excel or FreeMat, will do the job of plotting). In this case, the graphs show that the solution is somewhere near to $(-2, 2)$.

$$p = 3.2x^3 + 1.8y^2 + 24.43 \qquad q = 2x^2 - 3y^3 + 5.92$$

Table (3.1) – Example: Newton's Method to solve system of nonlinear equations

#	x_p	y_p	a	b	c	d	p	q	$\lvert C\rvert$	$\lvert C_x\rvert$	$\lvert C_y\rvert$	Δx	Δy	x_n	y_n	ϵ
1	-2.0000	2.0000	38.4000	7.2000	-8.0000	-36.0000	6.0300	-10.0800	-1324.8000	144.5040	338.8320	-0.1091	-0.2558	-2.1091	1.7442	0.2781
2	-2.1091	1.7442	42.7027	6.2793	-8.4363	-27.3813	0.1148	-1.1035	-1116.2841	-10.0734	48.0896	0.0090	-0.0431	-2.1001	1.7012	0.04402
3	-2.1001	1.7012	42.3361	6.1242	-8.4002	-26.0455	0.0017	-0.0257	-1051.2717	-0.1318	1.2022	0.0001	-0.0011	-2.0999	1.7000	0.0012
4	-2.0999	1.7000	42.3330	6.1201	-8.3997	-26.0105	2.0369E-06	-1.9986E-05	-1049.6958	-6.9336E-05	0.0008	6.6053E-08	-7.8972E-07	-2.0999	1.7000	7.9248E-07

As shown in Table 3.1 (on page 36), we were able to converge to the actual solution as `(-2.0999,1.7000)` in 4 iterations with a tolerance of `0.001`.

Solution to System of n Equations

A system of n (nonlinear) equations in n unknowns can, in general, be expressed as:

$$\begin{aligned} f_1(x_1, x_2, \cdots, x_n) &= 0 \\ f_2(x_1, x_2, \cdots, x_n) &= 0 \\ \vdots &= \vdots \\ f_n(x_1, x_2, \cdots, x_n) &= 0 \end{aligned}$$

Following the same approach as used in the case of two variables, and choosing $(x_{e1}, x_{e2}, \cdots, x_{en})$ as the initial estimate, we arrive at:

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}_{(x_{e1}, x_{e2}, \cdots, x_{en})} \begin{Bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{Bmatrix} = \begin{Bmatrix} -f_1 \\ -f_2 \\ \vdots \\ -f_n \end{Bmatrix}_{(x_{e1}, x_{e2}, \cdots, x_{en})}$$

We can solve this equation to obtain the vector $[\Delta x_k], 1 \leqslant k \leqslant n$, which can be used to find the next estimate as:

$$\begin{Bmatrix} x_{a1} \\ x_{a1} \\ \vdots \\ x_{an} \end{Bmatrix} = \begin{Bmatrix} x_{e1} \\ x_{e1} \\ \vdots \\ x_{en} \end{Bmatrix} + \begin{Bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{Bmatrix}$$

The process can be iterated till $\epsilon = ||(\Delta x_1, \Delta x_2, \cdots, \Delta x_n)|| = +\sqrt{\Delta x_1^2 + \Delta x_2^2 + \cdots + \Delta x_n^2} > tolerance$, every time substituting $(x_{e1}, x_{e2}, \cdots, x_{en}) = (x_{a1}, x_{a2}, \cdots, x_{an})$.

Convergence of Newton's Method

The determinant:

$$J(f_1, f_2, \cdots, f_n) = \begin{vmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{vmatrix}$$

is known as ***Jacobian*** of $(f_1, f_2, \cdots, f_n)$.

Convergence of *Newton's Method* is not guaranteed, but it is expected if these conditions hold:

- $f_1, f_2, \cdots, f_n$ and their partial derivatives are continuous and bounded near the actual solution.
- $J(f_1, f_2, \cdots, f_n) \neq 0$, near the solution.
- The initial solution estimate is sufficiently close to the actual solution.

3.8.2 Fixed-Point Iteration Method to solve System of Non-Linear Equations

The ***Fixed-Point Iteration Method*** to solve a single equation can be extended to handle ***System of Non-Linear Equations***, by rewriting the system as a set of ***Auxiliary Functions*** as follows:

$$\begin{aligned} x_1 &= g_1(x_1, x_2, \cdots, x_n) \\ x_2 &= g_2(x_1, x_2, \cdots, x_n) \\ &\vdots \\ x_n &= g_n(x_1, x_2, \cdots, x_n) \end{aligned}$$

Choose $(x_{p1}, x_{p2}, \cdots, x_{pn})$ as the initial estimate and substitute into the right sides of the *Set of Auxiliary Equations*. The updated estimates are calculated as:

$$\begin{aligned} x_{g1} &= g_1(x_{p1}, x_{p2}, \cdots, x_{pn}) \\ x_{g2} &= g_1(x_{p1}, x_{p2}, \cdots, x_{pn}) \\ &\vdots \\ x_{gn} &= g_1(x_{p1}, x_{p2}, \cdots, x_{pn}) \end{aligned}$$

For the next iteration, these new values are then re-used in the right sides of *Set of Auxiliary Equations* to generate the new updates, and so on. The process continues until convergence is observed (*i.e.* $\epsilon = ||(\Delta x_1, \Delta x_2, \cdots, \Delta x_n)|| \leqslant tolerance$, where $\Delta x_k = x_{gk} - x_{pk}, \forall\ 1 \leqslant k \leqslant n$).

Example 3.11 : Fixed-Point Iteration Method to solve System of Non-Linear Equations

Using the *Fixed-Point Iteration Method*, with a tolerance of `0.001`, solve the following nonlinear system of equations:

$$\begin{aligned} 3.2x^3 + 1.8y^2 + 24.43 &= 0 \\ -2x^2 + 3y^3 &= 5.92 \end{aligned}$$

Use $(-2, 2)$ as your initial estimate.

Solution : First, rewrite the equations as equivalent auxiliary functions as follows (this is only one possibility; other auxiliary functions may be possible):

$$\begin{aligned} x &= -\left(\frac{1.8y^2 + 24.43}{3.2}\right)^{\frac{1}{3}} \\ y &= \left(\frac{2x^2 + 5.92}{3}\right)^{\frac{1}{3}} \end{aligned}$$

or, to be precise:

$$\begin{aligned} x_g &= g_1(x_p, y_p) = -\left(\frac{1.8y_p^2 + 24.43}{3.2}\right)^{\frac{1}{3}} \\ y_g &= g_2(x_p, y_p) = \left(\frac{2x_p^2 + 5.92}{3}\right)^{\frac{1}{3}} \end{aligned}$$

Using these auxiliary functions we iterate to generate Table 3.2 (shown on page 38):

Table (3.2) – Example: Fixed-Point Iteration Method to solve nonlinear equations

#	x_p	y_p	$x_g = g_1(x_p, y_p)$	$y_g = g_2(x_p, y_p)$	ϵ
1	-2.0000	2.0000	-2.1461	1.6679	0.3628
2	-2.1461	1.6679	-2.0953	1.7150	0.0692
3	-2.0953	1.7150	-2.1021	1.6985	0.0178
4	-2.1021	1.6985	-2.0997	1.7007	0.0032
5	-2.0997	1.7007	-2.1000	1.7000	0.0008

As seen in Table 3.2, we were able to converge to the actual solution as `(-2.1000,1.7000)`, in `5` iterations, with a tolerance of `0.001`.

Convergence of Fixed-Point Iteration Method

As was the case of convergence of *Newton's Method*, similarly, convergence of *Fixed-Point Iteration Method* is not guaranteed, but it is expected if these conditions hold:

- Auxiliary functions $g_1, g_2, \cdots, g_n$ and their partial derivatives with respect to $x_1, x_2, \cdots, x_n$ are continuous near the actual solution.
- $$\begin{aligned} \left|\frac{\partial g_1}{\partial x_1}\right| + \left|\frac{\partial g_1}{\partial x_2}\right| + \cdots + \left|\frac{\partial g_1}{\partial x_n}\right| &\leqslant 1 \\ \left|\frac{\partial g_2}{\partial x_1}\right| + \left|\frac{\partial g_2}{\partial x_2}\right| + \cdots + \left|\frac{\partial g_2}{\partial x_n}\right| &\leqslant 1 \\ \vdots + \vdots + \cdots + \vdots &\leqslant 1 \\ \left|\frac{\partial g_n}{\partial x_1}\right| + \left|\frac{\partial g_n}{\partial x_2}\right| + \cdots + \left|\frac{\partial g_n}{\partial x_n}\right| &\leqslant 1 \end{aligned}$$
- The initial estimate $(x_{p1}, x_{p2}, \cdots, x_{pn})$ is sufficiently close to the actual solution $(x_{g1}, x_{g2}, \cdots, x_{gn})$.

Beginning of Chapter ⇑ Table of Contents