Chapter Three
Numerical Methods

At the end of this chapter, you should be able to:

- Know the types of error.
- Learn using Newton's and Secant method of solving Algebraic equations.
- Solve numerical integration and differentiation.
- Learn using numerical methods to solve ordinary and partial differential equations.
- Find an approximate value using polynomial interpolation.

3.1 Numerical Methods

To explore complex systems, physicists, engineers, financiers and mathematicians require computational methods since mathematical models are only rarely solvable algebraically. Numerical methods, based upon sound computational mathematics, are the basic algorithms underpinning computer predictions in modern systems science. Such methods include techniques for simple optimisation, interpolation from the known to the unknown, linear algebra underlying systems of equations, ordinary differential equations to simulate systems, and stochastic simulation under random influences. Topics covered are: the mathematical and computational foundations of the numerical approximation and solution of scientific problems; simple optimisation; vectorisation; clustering; polynomial and spline interpolation; pattern recognition; integration and differentiation; solution of large scale systems of linear and nonlinear equations; modelling and solution with sparse equations; explicit schemes to solve ordinary differential equations; random numbers; stochastic system simulation.

3.2 Characteristics of Numerical Methods

1. The solution procedure is iterative, with the accuracy of the solution improving with each iteration.

2. The solution procedure provides only an approximation to the true, but unknown, solution.

3. An initial estimate of the solution may be required.

4. The algorithm is simple and can be easily programmed.

5. The solution procedure may occasionally diverge from rather than converge to the true solution.

3.3 Types of error in Numerical Computation

I. Initial error/Error of the problem:

These are involved in the statement of problem itself. In fact, the statement of a problem generally gives an idealized model and not the exact picture of the actual phenomena. For example, in the calculation of the value of earth's gravitational force g by simple pendulum, the experiment is based upon certain axioms: such as (i) bob is weight less (ii) the motion of the bob is linear, that is, in a straight line; which are not true, in fact.

II. Residual error or truncation error:

This error occurs when mathematical functions like

$$\sin x = x - \frac{X^3}{3!} + \frac{X^5}{5!} \ldots\ldots\ldots\ldots\ldots\infty \text{ and } e^x = 1 + x + \frac{X^3}{2!} + \frac{X^3}{3!} + \ldots\ldots\ldots\ldots\infty,$$

whose infinite series expansion exist, are used in the calculations. Because, in calculating the value of such function for an assigned value of x, only a finite number of terms can be taken, an error get introduced for not considering the remaining terms.

III. Rounding error/ Round-off error:

When the rational numbers like 1/3; 22/7; 5/9; 8/9 etc, whose decimal representation involve infinite number of digits, are involved in our calculations, we are forced to take only a few number of digits from their decimal expression and thus an error named round-off error gets involved. There are universal rules for rounding a number as rounding rules.

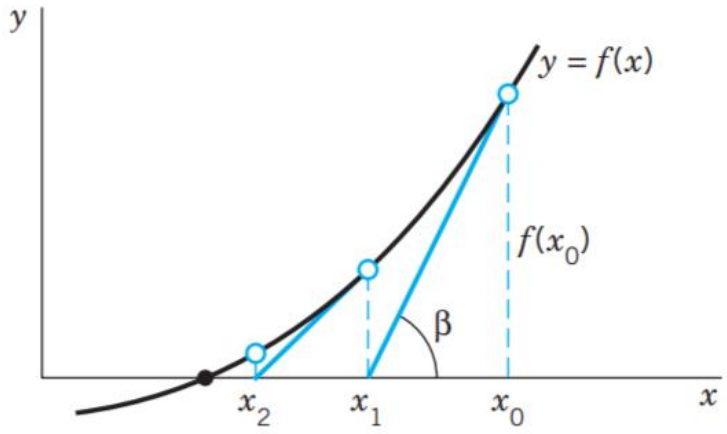
3.4 Numerical Solution of Algebraic Equations

3.4.1 Newton's Method

Newton's method, also known as Newton–Raphson's method, is an iteration method for solving equations where *f* is assumed to have a continuous derivative. The method is commonly used because of its simplicity and great speed.

The underlying idea is that we approximate the graph of *f* by suitable tangents. Using an approximate value obtained from the graph of *f*, we let be the point of intersection of the *x-axis* and the tangent to the curve of *f* at x_0 (see the figure below).

$$\tan\beta = f'(x_0) = \frac{f(x_0)}{x_0 - x_1}, \qquad \text{hence} \qquad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$



The general equation is:

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

Example (1): (**Square Root**) set up a Newton iteration for computing the square root *x* of a given positive number C and apply it to C = 2.

Solution.

Let's take $x = \sqrt{C} \Rightarrow x^2 - C = 0 \Rightarrow f(x) = x^2 - C = 0 \Rightarrow f'(x) = 2x$

The numerical solution starts from $x_0 = 1$, using Eq. 1:

n	x_n	$f(x_n)$	$f'(x_n)$	x_{n+1}	Error*
1	1	-1	2	1.5	0.333
2	1.5	0.25	3	1.416667	0.05882
3	1.416667	0.006944	2.833333	1.414216	0.00173
4	1.414216	6.01E-06	2.828431	1.414214	1.50182**E**-06
5	1.414214	4.51E-12	2.828427	1.414214	1.12764**E**-12
6	1.414214	0	2.828427	**1.414214**	0

Note: Error = $\frac{|x_{n+1} - x_n|}{x_{n+1}}$

The solution is exact to n=6 ($f(x) = 0$ and Error = 0).

Example (2): Find the positive solution of $2\sin x = x$.

Solution. Setting $f(x) = x - 2\sin x \Rightarrow f'(x) = 1 - 2\cos x$

The numerical solution starts from $x_0 = 2$, using Eq. 1:

n	x_n	$f(x_n)$	$f'(x_n)$	x_{n+1}	Error
1	2.00000	0.18141	1.83229	1.90100	0.05208
2	1.90100	0.00904	1.64846	1.89551	0.00289
3	1.89551	0.00003	1.63808	1.89549	0.00001
4	1.89549	0.00000	1.63805	**1.89549**	0.00000

Example (3): Solve the equation $f(x) = x^3 + x - 1 = 0$

Solution. The first derivate of the above equation is:

$$f'(x) = 3x^2 + 1 = 0$$

Let's start the numerical solution from $x_0 = 1$

n	x_n	$f(x_n)$	$f'(x_n)$	x_{n+1}	Error
1	1.00000	1.00000	4.00000	0.75000	0.33333
2	0.75000	0.17188	2.68750	0.68605	0.09322
3	0.68605	0.00894	2.41198	0.68234	0.00543
4	0.68234	0.00003	2.39676	**0.68233**	0.00002
5	0.68233	0.00000	2.39671	0.68233	0.00000

3.4.2 Secant Method

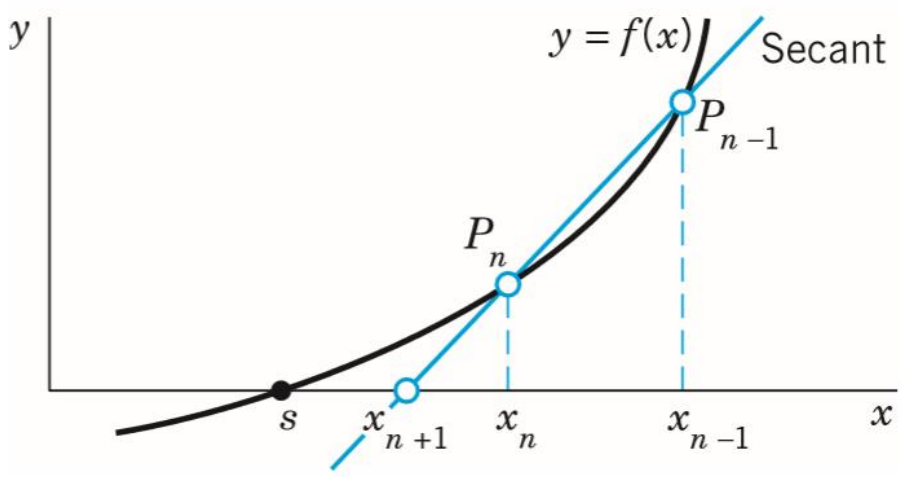
Newton's method is very powerful but has the disadvantage that the derivative may sometimes be a far more difficult expression than *f* itself and its evaluation therefore computationally expensive. This situation suggests the idea of replacing the derivative with the difference quotient

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} .$$

Accordingly, Eq. 1 becomes:

$$x_{n+1} = x_n - f(x_n) \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} .$$

The figure below illustrates the concept of Secant method:



Example (4): Repeat example (2) using Secant method where $x_0 = 2$ and $x_1 = 1.9$:

Solution.

$$f(x) = x - 2\sin x$$

Let's start the numerical solution from $x_0 = 2$ and $x_1 = 1.9$, using Secant formula:

n	x_{n-1}	x_n	$f(x_{n-1})$	$f(x_n)$	x_{n+1}	Error
1	2	1.9	0.181405	0.0074	1.895747	0.002243254
2	1.9	1.895747	0.0074	0.000414	1.895495	0.000132987
3	1.895747	1.895495	0.000414	1.08E-06	**1.895494**	3.46403E-07

NOTE: Only 3 iterations are needed to get a solution in this method in relation to Newton's method.

3.5 Numerical Integration and Differentiation

In many applications, the engineer often encounters integrals that are very difficult or even impossible to solve analytically. For example, the error function:

$$erf(x) = \frac{2}{\pi}\int_0^x e^{-t^2}\,dt$$

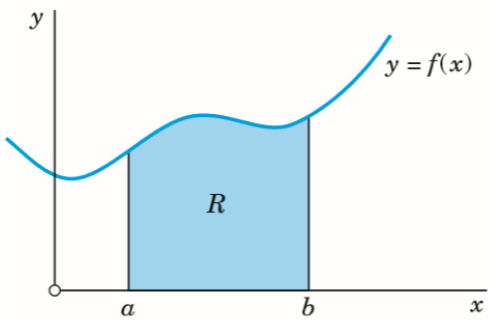
and others cannot be evaluated by the usual methods of calculus. We then need methods from numerical analysis to evaluate such integrals. We also need numerics when the integrand of the integral to be evaluated consists of an empirical function, where we are given some recorded values of that function. Methods that address these kinds of problems are called

methods of numeric integration.

Numeric integration means the numeric evaluation of integrals:

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

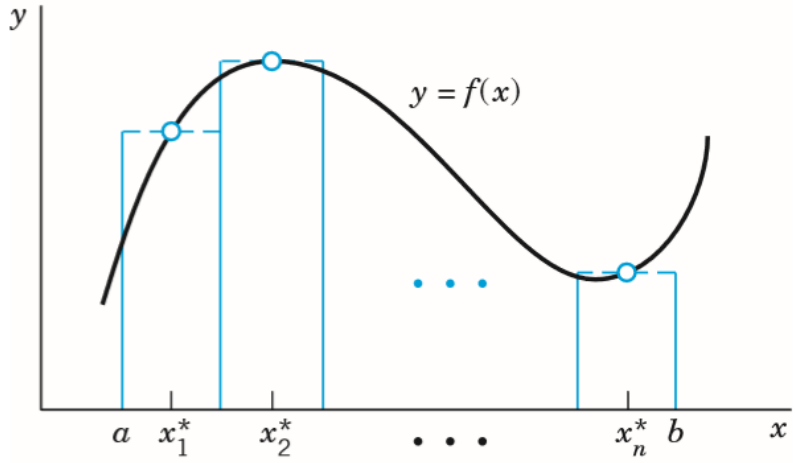
Geometrically, J is the area under the curve of f between a and b (as shown in the figure below), taken with a minus sign where f is negative.



3.5.1 Rectangular Rule:

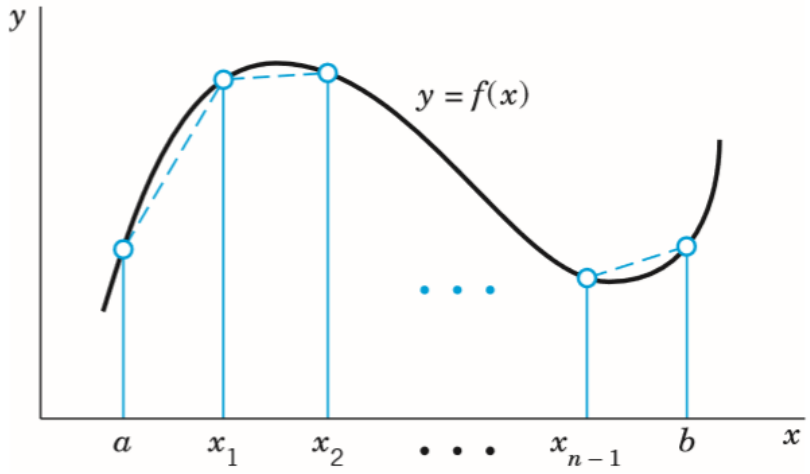
Numeric integration methods are obtained by approximating the integrand f by functions that can easily be integrated. The simplest formula, the rectangular rule, is obtained if we subdivide the interval of integration $a \le x \le b$ into n subintervals of equal length $h = (b-a)/2$ and in each subinterval approximate f by the constant $f(x_j^*)$, the value of f at the midpoint x_j^* of the jth subinterval (as shown in the figure below). Then f is approximated by a step function (piecewise constant function), the n rectangles in the figure have the areas $f(x_j^*)h, \ldots\ldots, f(x_n^*)h$. Thus, the rectangular rule is expressed as:

$$J = \int_a^b f(x)\,dx \approx h[f(x_1^*) + f(x_2^*) + \cdots + f(x_n^*)] \qquad \left(h = \frac{b-a}{n}\right).$$



3.5.2 Trapezoidal Rule:

The concept of this method is illustrated in the figure blow. As one may note, the trapezoidal rule is generally more accurate since the area under curve of f between a and b is approximated by n trapezoids of areas.



$$J = \int_a^b f(x)\,dx \approx h\left[\frac{1}{2}f(a) + f(x_1) + f(x_2) + \cdots + f(x_{n-1}) + \frac{1}{2}f(b)\right]$$

where $h = (b-a)/n$

Example (5): Evaluate

$$J = \int_0^1 e^{-x^2} dx \text{, with } n = 10$$

Using trapezoidal method:

Solution.

$h = (b-a)/n$ = (1-0)/10 = 0.1

Thus, according to trapezoidal rule, *J* value is:

$$J = \int_0^1 e^{-x^2} dx \approx 0.1\left[1/2\ f(0) + f(0.1) + f(0.2) + ... + 1/2\, f(1)\right]$$

To find the values of the sub-functions in the above equation, the following table is constructed:

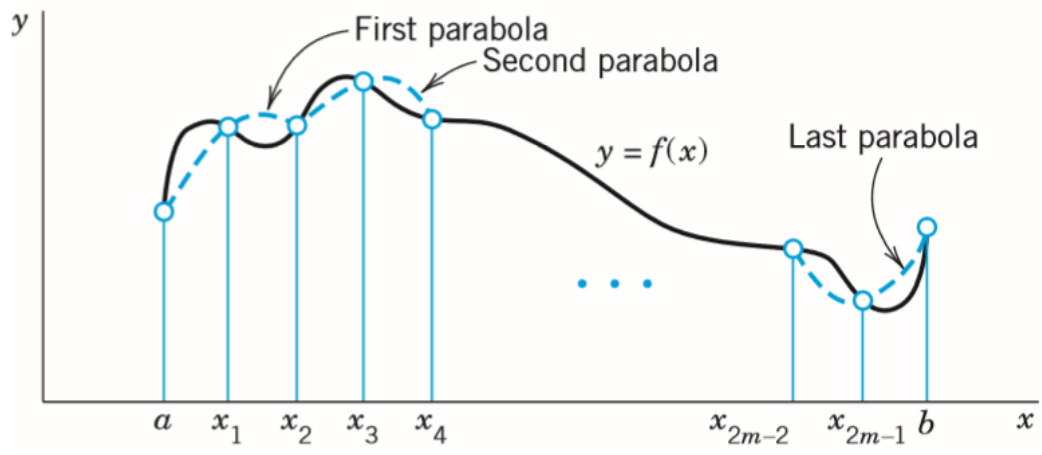
j	x_j	x_j^2	$Exp\ (-x_j^2)$
0	a= 0	0	1
1	0.1	0.01	0.99005
2	0.2	0.04	0.960789
3	0.3	0.09	0.913931
4	0.4	0.16	0.852144
5	0.5	0.25	0.778801
6	0.6	0.36	0.697676
7	0.7	0.49	0.612626
8	0.8	0.64	0.527292
9	0.9	0.81	0.444858
10	b=1	1	0.367879
Sum of the shaded numbers only			**6.778168**

Therefore, now:

$$J = \int_0^1 e^{-x^2}\,dx \approx 0.1\left[1/2*1 + \underbrace{6.77816}_{\text{sum of the values in the table}} + 1/2*0.367879\right] = 0.7462$$

3.5.3 Simpson's Rule:

Piecewise constant approximation of f led to the rectangular rule, piecewise linear approximation to the trapezoidal rule, and piecewise quadratic approximation will lead to Simpson's rule, which is of great practical importance because it is sufficiently accurate for most problems, but still sufficiently simple.



$$\int_a^b f(x)\,dx \approx \frac{h}{3}\,(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 2f_{2m-2} + 4f_{2m-1} + f_{2m}),$$

where $h = (b-a)/2m$

Example (6): Repeat example (5) using Simpson's rule.

$$J = \int_0^1 e^{-x^2} dx \text{, with } n = 2m = 10$$

Thus, according to Simpson's rule, J value is:

$$J = \int_0^1 e^{-x^2} dx \approx \frac{0.1}{3}\left[f(0) + 4f(0.1) + 2f(0.2) + 4f(0.3).... + f(1)\right]$$

To find the values of the sub-functions in the above equation, the following table is constructed:

j	x_j	x_j^2	$Exp\,(-x_j^2)$	$4Exp\,(-x_j^2)$	$2Exp\,(-x_j^2)$
0	a= 0	0	1		
1	0.1	0.01	0.99005	3.960199	
2	0.2	0.04	0.960789		1.921579
3	0.3	0.09	0.913931	3.655725	
4	0.4	0.16	0.852144		1.704288
5	0.5	0.25	0.778801	3.115203	
6	0.6	0.36	0.697676		1.395353
7	0.7	0.49	0.612626	2.450506	
8	0.8	0.64	0.527292		1.054585
9	0.9	0.81	0.444858	1.779432	
10	b=1	1	0.367879		
Sum of the shaded numbers only				**14.96107**	**6.075804**

From the table:

$$J = \int_0^1 e^{-x^2}\,dx \approx \frac{0.1}{3}\left[1 + \underbrace{14.96 + 6.075804}_{\text{sum of the values in the table}} + 0.367879\right] = 0.7468$$

Example (7): Find the value of *J* and check your solution:

$$J = \int_{40}^{100} f(x)\,dx$$

where the values of x and $f(x)$ are given in the table below:

x	40	50	60	70	80	90	100
$f(x)$	44	63	79	91	104	115	128

Solution.

The solution is straightforward. First, from the table $h = 10$

$$J = \int_{40}^{100} f(x)\,dx \approx \frac{10}{3}\left[44 + 4(63 + 91 + 115) + 2(79 + 104) + 128\right] = 5380$$

Checking:

Two methods can be used to check the solution:

1. Plot the data and estimate manually the area under the curve as shown in the figure below:

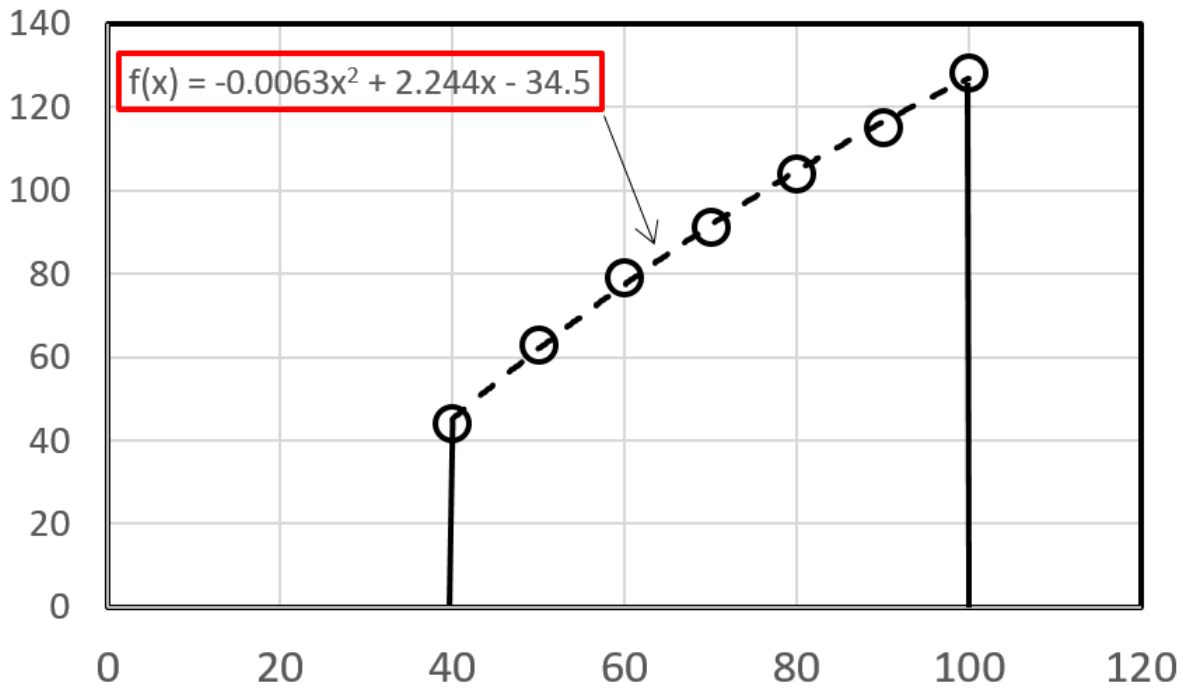
$$J \approx 13.3 * \underbrace{(20 * 20)}_{\text{area of one box}} = 5320$$

2. Plot the data and fit a curve to them as shown in the figure below. This gives the following equation:

$$f(x) = 0.0002x^3 - 0.00471x^2 + 4.9663x - 91.667$$

$$J = \int_{40}^{100} f(x)\,dx = \left[\frac{0.0002}{4} x^4 - \frac{0.0047}{3} x^3 + \frac{4.966}{2} x^2 - 91.667x \right]_{40}^{100}$$

$$J = 5389.2$$



3.6 Numerical Methods for Differential Equations

This section is about numerics for ODEs. We start with first-orer ODEs and discuss, in Sec. 3.6.1, methods for first-order ODEs. The main initial idea is that we can obtain approximations to the solution of such an ODE at points that are a distance *h* apart by using the first two terms of Taylor's formula from calculus. We use these approximations to construct the iteration formula for a method known as Euler's method. While this method is rather unstable and of little practical use, it serves as a pedagogical tool and a starting point toward understanding more sophisticated methods such as the Runge–Kutta method and its variant the Runga–Kutta–Fehlberg (RKF) method, which are popular and useful in practice.

3.6.1 Methods for First-Order ODEs

A. Euler method (or Euler–Cauchy method)

We know that an ODE of the first order is of the form $f(x, y, y') = 0$ and can often be written in the explicit form $y' = f(x, y)$. Euler method to solve the first order differential equation is based on Taylor series:

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2} y''(x) + ...$$

For small *h* the higher powers in the above equation are very small. Dropping all of them gives the crude approximation:

$$\begin{aligned} y(x + h) &\approx y(x) + hy'(x) \\ &= y(x) + hf(x, y) \end{aligned}$$

The general formula for Euler method can be written as:

$$y_{n+1} = y_n + hf(x_n, y_n)$$

Example (8): Apply Euler's method to the following equation, choosing $h = 0.2$ and computing y at $x = 1$. Compare the results with the exact solution.

$$y' = x + y \quad , \quad y(0) = 0$$

Solution:

Here, $f(x, y) = x + y$. Therefore, Euler general formula can be expressed as:

$$y_{n+1} = y_n + 0.2(x_n + y_n) ----(1)$$

To find y (5), we construct the following table:

Table 1

n	x_n	y_n	y_{n+1} from Eq. 1
0	0	0	0
1	0.2	0	0.04
2	0.4	0.04	0.128
3	0.6	0.128	0.274
4	0.8	0.274	0.489
5	1	**0.489**	0.7856

The solution is *y(1)=0.489*

Comparing the results with the exact solution:

The exact solution of $y' = x + y$ is $y(x) = e^x - x - 1$. Therefore, one can compare the results by computing the error:

n	y_n (from Table 1)	$y_{n\,(\text{Excat})}$ $y(x) = e^x - x - 1$	*Error* $\lvert y_{n\ from\ table1} - y_{n\ Excat} \rvert$
0	0	0	0
1	0	0.021403	0.021403
2	0.04	0.091825	0.051825
3	0.128	0.222119	0.094119
4	0.274	0.425541	0.151541
5	**0.489**	0.718282	0.229282

B. Runge–Kutta Methods (RK Methods)

A method of great practical importance and much greater accuracy than that of the Euler method is the classical Runge–Kutta method of fourth order, which we call briefly the Runge–Kutta method.

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where:

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1)$$
$$k_3 = hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_2)$$
$$k_4 = hf(x_n + h, y_n + k_3)$$

Example (9): Apply the Runge–Kutta method to the initial value problem in Example 8, choosing $h = 0.2$ as before, and computing five steps.

Table 2

n	x_n	$k1$	$k2$	$k3$	$k4$	y_{n+1} from RK Eqn
0	0	0	0	0.02	0.022	0.0444
1	0.2	0.0214	0.04428	0.068708	0.071151	0.09851
2	0.4	0.091818	0.098364	0.1282	0.131184	0.1646
3	0.6	0.222106	0.164421	0.200863	0.204508	0.245323
4	0.8	0.425521	0.245104	0.289615	0.294066	0.343917
5	1	**0.718251**	0.34365	0.398015	0.403452	0.464341

The solution is $y(1) = 0.718251$

Comparing the results with the exact solution:

n	y_n (from Table 2)	$y_{n\,(\text{Excat})}$ $y(x) = e^x - x - 1$	*Error* $\lvert y_{n\ from\ table1} - y_{n\ Excat} \rvert$
0	0	0	0
1	0.0214	0.021403	2.76E-06
2	0.091818	0.091825	6.74E-06
3	0.222106	0.222119	1.23E-05
4	0.425521	0.425541	2.01E-05
5	**0.718251**	0.718282	3.07E-05

3.6.2 Methods for Second-Order Equations

Euler method

Methods for single first-order ODEs can be extended to second-order differential equation simply by considering *y* and y' instead of *y* only. Accordingly,

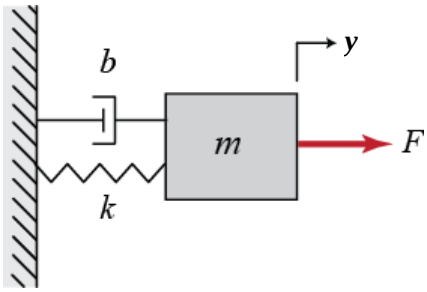
$$y_{n+1} = y_n + hy'$$
$$y'_{n+1} = y'_n + h\ f(x_n, y_n, y'_n)$$

Example (10): Solve the initial value problem for a damped mass–spring system

$$y'' + 2y' + 0.75y = 0, \qquad y(0) = 3, \qquad y'(0) = -2.5$$

by the Euler method for systems with step for *x* from 0 to 1 (where x is time).

Solution:



$$y_{n+1} = y_n + hy' \text{----(1)}$$
$$y'_{n+1} = y'_n + h\ f(x_n, y_n, y'_n) \text{-----(2)}$$

From the differential equation:

$$f(x_n, y_{1,n}, y') = -2y' - 0.75y$$

Using the above equations, one can construct the following table:

n	x_n	y_n	y'	y_{n+1} **(Eq. 1)**	y'_{n+1} **(Eq. 2)**
0	0	3.00	-2.500	2.500	-1.950
1	0.2	2.50	-1.950	2.110	-1.545
2	0.4	2.11	-1.545	1.801	-1.244
3	0.6	1.80	-1.244	1.552	-1.016
4	0.8	1.55	-1.016	1.349	-0.843
5	1	**1.35**	**-0.843**	1.181	-0.708

From the table, the solution is : $y(1) = 1.35$ and $y'(1) = -0.843$

3.7 Interpolation

The interpolation is a process to find an **approximate** value of a function $f(x)$ for an x between different x-values at which the values of $f(x)$ are given.

Many methods can be used to implement an interpolation process. Among those are:

1. Lagrange Interpolation
2. Newton's Divided Difference Interpolation
3. Equal Spacing: Newton's Forward Difference Formula
4. Equal Spacing: Newton's Backward Difference Formula
5. Central Difference

These methods will be discussed and illustrated by way of examples as below.

Lagrange Interpolation

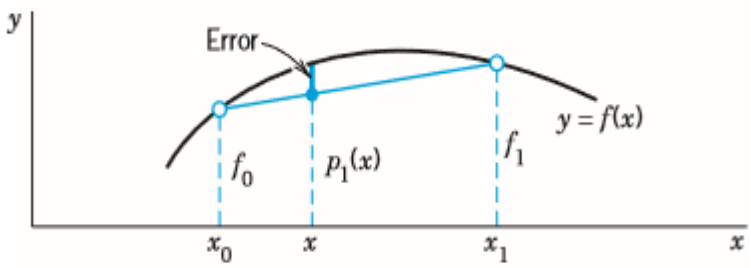
The general formula for this method is:

$$f(x) \approx p_n(x) = \sum_{k=0}^{n} L_k(x) f_k = \sum_{k=0}^{n} \frac{l_k(x)}{l_k(x_k)} f_k$$

where:

$$\begin{aligned} l_0(x) &= (x - x_1)(x - x_2) \cdots (x - x_n), \\ l_k(x) &= (x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n), \qquad 0 < k < n, \\ l_n(x) &= (x - x_0)(x - x_1) \cdots (x - x_{n-1}). \end{aligned}$$

Example (11): (linear Interpolation) If you have two data points **(x_0, f_0)** and **(x_1, f_1)**. Find a linear relationship to determine the value of any point (x, p_1) in between, using Lagrange general formula:



Solution:
From Lagrange general formula:

$$p_1(x) = \sum_0^1 L_k(x) f_k = L_0(x) f_0 + L_1(x) f_1 \quad ----(1)$$

$L_0(x)$ and $L_1(x)$ in Eq.1 can be expressed as:

$$L_0(x) = \frac{l_0(x)}{l_0(x_0)} \quad \text{and} \quad L_1(x) = \frac{l_1(x)}{l_1(x_1)}$$

Hence, Eq. 1 becomes:

$$p_1(x) = \frac{l_0(x)}{l_0(x_0)} f_0 + \frac{l_1(x)}{l_1(x_1)} f_1 \quad ----(2)$$

where

$l_0(x) = (x - x_1)$
$l_0(x_0) = (x_0 - x_1)$
$l_1(x) = (x - x_0)$
$l_1(x_1) = (x_1 - x_0)$

} Substitute in Eq. 2

Thus,

$$p_1(x) = \frac{(x - x_1)}{(x_0 - x_1)} f_0 + \frac{(x - x_0)}{(x_1 - x_0)} f_1$$

Example (12): Compute $\ln 9.2$ from $\ln 9 = 2.1972$ and $\ln 9.5 = 2.2513$, using linear Lagrange interpolation.

Solution:

$x_0 = 9 \rightarrow f_0 = 2.1972$
$x = 9.3 \rightarrow p_1 = ?$
$x_1 = 9.5 \rightarrow f_1 = 2.2513$

As in example 11, from the general Lagrange formula, one can find:

$$p_1(x) = \frac{(x - x_1)}{(x_0 - x_1)} f_0 + \frac{(x - x_0)}{(x_1 - x_0)} f_1$$

$$p_1(9.2) = \frac{(9.2 - 9.5)}{(9 - 9.5)} * 2.1972 + \frac{(9.2 - 9)}{(9.5 - 9)} * 2.2513 \Rightarrow p_1(9.2) = 2.2188$$

Example (13): (Quadratic Interpolation) compute $\ln 9.2$ from $\ln 9 = 2.1972$, $\ln 9.5 = 2.2513$ and $\ln 11 = 2.3979$ using Lagrange interpolation.

Solution:

$$p_1(x) = \sum_0^2 L_k(x) f_k = L_0(x) f_0 + L_1(x) f_1 + L_2(x) f_2$$

$$p_1(x) = \frac{l_0(x)}{l_0(x_0)} f_0 + \frac{l_1(x)}{l_1(x_1)} f_1 + \frac{l_2(x)}{l_2(x_2)} f_2$$

$$p_1(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f_1 + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f_2$$

$$p_1(9.2) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f_1 + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f_2$$

$$p_1(9.2) = \frac{(9.2 - 9.5)(9.2 - 11)}{(9 - 9.5)(9 - 11)} * 2.1972 + \frac{(9.2 - 9)(9.2 - 11)}{(9.5 - 9)(9.5 - 11)} * 2.2513 + \frac{(9.2 - 9)(9.2 - 9.5)}{(11 - 9)(11 - 9.5)} * 2.3979$$

$p_1(9.2) = 2.2192$

Newton's Divided Difference Interpolation

Newton's divided difference interpolation formula is:

$$f(x) \approx f_0 + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] + \cdots + (x - x_0)(x - x_1)\cdots(x - x_{n-1})f[x_0, \cdots, x_n].$$

This method is illustrated in Example 13.

Example (14): Compute ***f*(9.2)** from the values shown in the following table.

x_j	$f_j = f(x_j)$
8.0	2.079442
9.0	2.197225
9.5	2.251292
11.0	2.397895

Solution:

To find $f[x_0, x_1]$, $f[x_0, x_1, x_2]$, $f[x_0, ..., x_n]$ in Newton's interpolation formula, we need to construct a **difference table** as follows:

x_j	$f_j = f(x_j)$	$f[x_j, x_{j+1}]$	$f[x_j, x_{j+1}, x_{j+2}]$	$f[x_j, \cdots, x_{j+3}]$
8.0	2.079442			
		0.117783		
9.0	2.197225		−0.006433	
		0.108134		0.000411
9.5	2.251292		−0.005200	
		0.097735		
11.0	2.397895			

The values in the table are calculated as follows:

First column:

$$f[x_j, x_{j+1}] = \frac{2.197225 - 2.079442}{9 - 8} = 0.117783$$

$$f[x_j, x_{j+1}] = \frac{2.251292 - 2.197225}{9.5 - 9} = 0.108134$$

$$f[x_j, x_{j+1}] = \frac{2.397895 - 2.251292}{11 - 9.5} = 0.097735$$

Second column:

$$f[x_j, x_{j+1}, x_{j+2}] = \frac{0.108134 - 0.117783}{9.5 - 8} = -0.006433$$

$$f[x_j, x_{j+1}, x_{j+2}] = \frac{0.097735 - 0.108134}{11 - 9} = -0.0052$$

Third column:

$$f[x_j, ..., x_{j+3}] = \frac{-0.0052 + 0.006433}{11 - 8} = 0.000411$$

The values needed in Newton's interpolation formula are circled:

$$f(x) \approx p_3(x) = 2.079442 + 0.117783(x - 8.0) - 0.006433(x - 8.0)(x - 9.0) + 0.000411(x - 8.0)(x - 9.0)(x - 9.5).$$

At $x = 9.2$,

$$f(9.2) \approx 2.079442 + 0.141340 - 0.001544 - 0.000030 = 2.219208.$$

Equal Spacing: Newton's Forward Difference Formula

Newton's formula is valid for arbitrarily spaced nodes as they may occur in practice in experiments or observations. However, in many applications the data may be regularly spaced — for instance, in measurements taken at regular intervals of time.

This means: x_0, $x_1 = x_0 + h$, $x_2 = x_1 + 2h$ …etc.

In this case, Newton's Forward Difference Formula can be used to carry out an interpolation process. For a function $f(x)$, this formula can be expressed as:

$$f(x) = f_0 + r\,\Delta f_0 + \frac{r(r-1)}{2!}\Delta^2 f_0 + \ldots\ldots + \frac{r(r-1)\ldots\ldots(r-n-1)}{n!}\Delta^n f_0$$

Note here that ***the first forward difference of f at x_j is defined by***:

$$\Delta f_j = f_{j+1} - f_j,$$

the *second forward difference* of f at x_j by

$$\Delta^2 f_j = \Delta f_{j+1} - \Delta f_j,$$

and, continuing in this way, the **kth forward difference** of f at x_j by

$$\Delta^k f_j = \Delta^{k-1} f_{j+1} - \Delta^{k-1} f_j \qquad (k = 1, 2, \cdots).$$

and, finally, $r = \dfrac{x - x_j}{h}$

Example (15): Compute cosh (0.56) from Newton's forward difference formula and the four values in the following table.

j	x_j	$f_j = \cosh x_j$
0	0.5	1.127626
1	0.6	1.185465
2	0.7	1.255169
3	0.8	1.337435

Solution:

We compute the forward differences as shown in the table below.

j	x_j	$f_j = \cosh x_j$	Δf_j	$\Delta^2 f_j$	$\Delta^3 f_j$
0	0.5	1.127626			
			0.057839		
1	0.6	1.185465		0.011865	
			0.069704		0.000697
2	0.7	1.255169		0.012562	
			0.082266		
3	0.8	1.337435			

The values we need are circled.

In the general formula, we have r = (0.56-0.5)/0.1= 0.6. So that the general formula gives:

$$\cosh 0.56 \approx 1.127626 + 0.6 \cdot 0.057839 + \frac{0.6(-0.4)}{2} \cdot 0.011865 + \frac{0.6(-0.4)(-1.4)}{6} \cdot 0.000697$$

$$= 1.127626 + 0.034703 - 0.001424 + 0.000039$$

$$= 1.160944.$$

Equal Spacing: Newton's Backward Difference Formula

Instead of forward-sloping differences (explained in example 15), we may also employ backward-sloping differences.

Therefore, ***the first backward difference of f at x_j is defined by*** **:**

$$\nabla f_j = f_j - f_{j-1},$$

the *second backward difference* of f at x_j by

$$\nabla^2 f_j = \nabla f_j - \nabla f_{j-1},$$

and, continuing in this way, the ***k*th backward difference** of f at x_j by

$$\nabla^k f_j = \nabla^{k-1} f_j - \nabla^{k-1} f_{j-1} \qquad (k = 1, 2, \cdots).$$

A formula similar to the forward formula but involving backward differences is Newton's backward difference interpolation formula

$$f(x) = f_0 + r\,\nabla f_0 + \frac{r(r-1)}{2!}\nabla^2 f_0 + \ldots\ldots + \frac{r(r-1)\ldots\ldots(r-n-1)}{n!}\nabla^n f_0$$

Example (16): Compute a 7 digits-value of the Bessel function $j_0(x)$ for $x = 1.72$ from the four values in the following table, using (a) Newton's forward formula, (b) Newton's backward formula.

j_{for}	j_{back}	x_j	$J_0(x_j)$
0	−3	1.7	0.3979849
1	−2	1.8	0.3399864
2	−1	1.9	0.2818186
3	0	2.0	0.2238908

Solution:

j_{for}	j_{back}	x_j	$J_0(x_j)$	1st Diff.	2nd Diff.	3rd Diff.
0	−3	1.7	0.3979849			
				−0.0579985		
1	−2	1.8	0.3399864		−0.0001693	
				−0.0581678		0.0004093
2	−1	1.9	0.2818186		0.0002400	
				−0.0579278		
3	0	2.0	0.2238908			

(a) Forward. Here we have $r = (1.72 - 1.70)/0.1 = 0.2$, and j goes from 0 to 3 (see first column). In each column we need the first given number,

$$J_0(1.72) \approx 0.3979849 + 0.2(-0.0579985) + \frac{0.2(-0.8)}{2}(-0.0001693) + \frac{0.2(-0.8)(-1.8)}{6} \cdot 0.0004093$$
$$= 0.3979849 - 0.0115997 + 0.0000135 + 0.0000196 = 0.3864183,$$

which is exact to 6D, the exact 7D-value being 0.3864185.

(b) Backward. Here we use j shown in the second column, and in each column the last number. Since $r = (1.72 - 2.00)/0.1 = -2.8$,

$$J_0(1.72) \approx 0.2238908 - 2.8(-0.0579278) + \frac{-2.8(-1.8)}{2} \cdot 0.0002400 + \frac{-2.8(-1.8)(-0.8)}{6} \cdot 0.0004093$$
$$= 0.2238908 + 0.1621978 + 0.0006048 - 0.0002750$$
$$= 0.3864184. \quad \blacksquare$$