<u>Name of the experiment</u> : Study of Bisection method .

<u>Objective</u> :

i) To know how to write algorithm for Bisection method .

ii) To know how to solve problem by programming .

<u>Theory for Bisection method</u> :

The bisection method is one of the bracketing methods for finding roots of an equation.
For a given a function *f(x)*, guess an interval which might contain a root and perform a number of iterations, where, in each iteration the interval containing the root is get halved.
The bisection method is based on the intermediate value theorem for continuous functions .

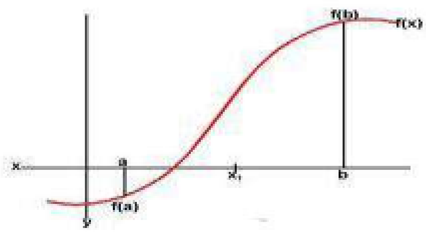


Fig : Graph for bisection method .

Let , $f(x) = 0$

If f(x) is continuous at $a \le x \le b$
f(a) , f(b) are opposite sign , then there exists at least one root between a & b .
For definiteness , let f(a) be negative & f(b) be positive . Then the root lies between a & b and let approximate value be given by

$$X_1=\frac{a+b}{2}\text{.................}(1)$$

If $f(x_1)=0$,we conclude that x_1 is a root of the equation $f(x)=0$.

Otherwise , the root lies between x_1 and b , or between x_1 and a depending on whether f (x_1) is negative or positive . We designate this new interval as [a1 ,b1] whose lenth is I b – a I /2 . As before, this is bisected at x_1 and the new interval will be exactly half the length of the previous one . We repeat this process until the latest interval is as small as desired , say e .

Algorithm for Bisection method :

1.Readx_o,x_1,e

2.$y_o \leftarrow f(x_o)$

3.$y_1 \leftarrow f(x_1)$

4.$i \leftarrow 0$

5.if(sign(y_o)=sign(y_1)) then begin to write ‘starting values unsuitable’

Write x_o,x_1,y_0,y_1

Stop end

6.While $|(x_1-x_0)/x_1|>e$

do begin

7.$x_2 \leftarrow (x_0+x_1)/2$

8.$y_2 \leftarrow f(x_2)$

9.$i \leftarrow i+1$

10.if(sign(y_o)=sign(y_2)) then $x_0 \leftarrow x_2$ else $x_1 \leftarrow x_2$

end

11.Write ’solution converges to a root’

12.Write ‘Number of iterations= ’,i

13.Write x_2,y_2

14.Stop

Program for Bisection method:

Example : Solve the equation by bisection method .

$$f(x)=x^3-x-1=0$$

Program :

```
clc;
clear all;
fx=input('Enter the function ,F(x) = ','s');
f=eval(['@(x)',fx]) ;
a=input('Enter a=');
b=input('Enter b=');
v=b;
while(f(b)<0)
  b=a;
  a=v;
  break;
end
s=1;
fprintf('N\t    \ta\t\t       b\t\t       x\t\t     f(x)\t\t\t     Error\n');
for k=1:100;
  it(k)=abs(k);
  x(k)=(a+b)/2;
  c=f(x(k));
  fprintf('%g       %f       %f       %f       %f       %f\n',k,a,b,x(k),c,s);
  if c>0
    b=x(k);
  else
    a=x(k);
  end
```

```
  x(k+1)=(a+b)/2;
  s=((abs(x(k+1)-x(k)))/abs(x(k+1)))*100;

  if s<=.01
     break;
  end
end
fprintf('\n\n Hence the root is %f ',x(k));
```

Output :

Enter the function ,F(x) = x^3-x-1

Enter a=1

Enter b=2

N	a	b	x	f(x)	Error
1	1.000000	2.000000	1.500000	0.875000	1.000000
2	1.000000	1.500000	1.250000	-0.296875	20.000000
3	1.250000	1.500000	1.375000	0.224609	9.090909
4	1.250000	1.375000	1.312500	-0.051514	4.761905
5	1.312500	1.375000	1.343750	0.082611	2.325581
6	1.312500	1.343750	1.328125	0.014576	1.176471
7	1.312500	1.328125	1.320313	-0.018711	0.591716
8	1.320313	1.328125	1.324219	-0.002128	0.294985
9	1.324219	1.328125	1.326172	0.006209	0.147275
10	1.324219	1.326172	1.325195	0.002037	0.073692
11	1.324219	1.325195	1.324707	-0.000047	0.036860
12	1.324707	1.325195	1.324951	0.000995	0.018426

Hence the root is 1.324951 >>

Name of the experiment : Study of False-position method .

Objective :

i) To know how to write algorithm for false-position method .

ii) To know how to solve problem by programming .

Theory for False-position method :

This method is also based on the intermediate value theorem . In this method we choose two points a and b such that f(a) and f(b) are of opposite sign (i.e. , $f(a)f(b) < 0$) . Then , intermediate value theorem suggests that a zero of f(x) lies in between a and b , if f(x) is a continuous function .

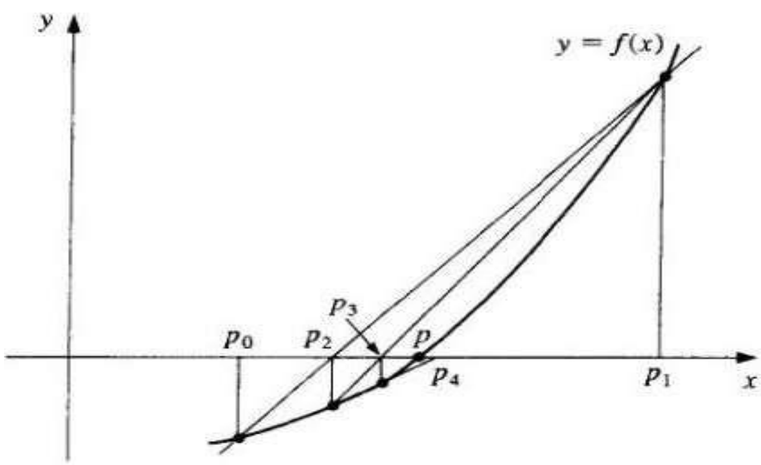


Fig : Method of false position

Now , the equation of the chord joining the two points [a, f(a)] and [b , f(b)] is given by

$$\frac{y-f(a)}{x-a} = \frac{f(b)-f(a)}{b-a} \quad \ldots\ldots\ldots\ldots\ldots \text{i}$$

The method consists in replacing the part of the curve between the points [a, f(a)] and [b , f(b)] by means of the cord joining these points , and taking the points of intersectin of the cord with the x-axis as an approximation to the root . The point of intersection in the present case is obtained by putting y=0 . Thus we obtain

$$X_1 = a - \frac{f(a)}{f(b)-f(a)}\ (b-a)$$

$$= \frac{a\ f(b)-bf(a)}{f(b)-f(a)} \quad \text{...................ii}$$

Algorithm for False-position method :

1) Read x_0 ,x_1,e , n
2) $f_0 \leftarrow f(x_0)$
3) $f_1 \leftarrow f(x_1)$
4) for i=1 to n
5) $X_2 \leftarrow (x_0 f_1 - x_1 f_1)/(f_1 - f_0)$
6) $f_2 \leftarrow f(x_2)$
7) if $| f_2 | \leq$ e then
8) begin Write ' convergent solution ' , x_2 , f_2
9) stop end
10) if sign $(f_2) \neq$ sign (f_0)
11) then begin $x_1 \leftarrow x_2$
12) $f_1 \leftarrow f_2$ end
13) else begin $x_0 \leftarrow x_2$
14) $f_0 \leftarrow f_2$ end

Endfor

15) write 'Does not converge in n iterations '

16) write x_2 , f_2

17) Stop

Program for False-position method:

Example : Solve the equation by False-position method .

$$f(x)=x^3-x-4=0$$

Program :

```
clc;
clear all;
fx=input('Enter the function ,F(x) = ','s');
f=eval(['@(x)',fx]) ;
a=input('Enter a=');
b=input('Enter b=');
s=1;
fprintf('N\t     \ta\t\t      b\t\t          x\t\t        f(x)\t\t\t    Error\n');
for k=1:100;
   x(k)=a-(f(a)*(b-a))/(f(b)-f(a));
   c=f(x(k));
   fprintf('%g       %f       %f        %f        %f        %f\n',k,a,b,x(k),c,s);
   if c>0
      b=x(k);
   else
      a=x(k);
   end
   x(k+1)=a-(f(a)*(b-a))/(f(b)-f(a));
   s=((abs(x(k+1)-x(k)))/abs(x(k+1)))*100;
   if s<=.01
      break;
   end
```

```
end

fprintf('\n\nThe root is =%f',x(k));
```

Output :

Enter the function ,F(x) = x^3-x-4

Enter a=1

Enter b=2

N	a	b	x	f(x)	Error
1	1.000000	2.000000	1.666667	-1.037037	1.000000
2	1.666667	2.000000	1.780488	-0.136098	6.392694
3	1.780488	2.000000	1.794474	-0.016025	0.779384
4	1.794474	2.000000	1.796107	-0.001862	0.090957
5	1.796107	2.000000	1.796297	-0.000216	0.010559

The root is =1.796297>>

Name of the experiment : Study of Newton-Raphson method .

Objective :

i) To know how to write algorithm for Newton-Raphson method .

ii) To know how to solve problem by programming .

Theory for Newton-Raphson method :

Consider f(x) =0 , where f(x) has continuous derivative f ′(x) .

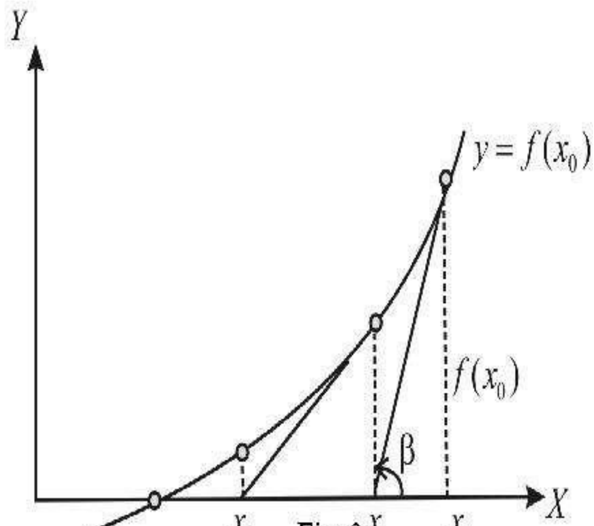


Fig. 3

From the figure we can say that at $x = a$, $y = f(a) = 0$; which means that a is a solution to the equation $f(x)=0$. In order to find the value of a , we start with any arbitrary point x_0. From figure we see that , the tangent to the curve f(x) at (x_0 , f(x_0)) (with slope f ' (x_0)) touches the x-axis at x_1.

Now , tan B = f '(x_0) = (f (x_0) – f (x_1))/ (x_0 - x_1) i

As f (x_1) = 0 , the above simplifies to

$$x_1 = x_0 - f(x_0)/f'(x_0)$$

In the second step , we compute

$$x_2 = x_1 - f(x_1)/f'(x_1)$$

In the third step we compute

$$x_3 = x_2 - f(x_2)/f'(x_2)$$

And so on . More generally , we write x_{n+1} , f (x_n) and f ' (x_n) for n = 1 , 2 ,3 .... By means of Newton-Raphson formula

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

<u>Algorithm for Newton-Raphson method</u> :

1) Read x_0, epslon, delta , n
2) for i=1 to n

3) $f_0 \leftarrow f(x_0)$

4) $f'_0 \leftarrow f'(x_0)$

5) if I f' I $\leq$ delta then go to 11

6) $x_1 \leftarrow x_0-(f_0/f'_0)$

7) if I $(x_1-x_0)/x_1$ I < epsilon then go to 13

8) $x_0 \leftarrow x_1$

Endfor

9) write 'Does not converge in n iterations' , f_0 ,f'_0, x_0 ,x_1

10) stop

11) write ' Slope too small ' x_0 ,f_0, f'_0, i

12) Stop

13) write ' convergent solution ' , x_1 ,$f(x_1)$, i

14) stop

Program for Newton-Raphson method:

Example : Solve the equation by Newton-Raphson method .

$$f(x)=x^3-5x+3=0$$

Program :

```
clc;
clear all;
fx=input('Enter the function ,F(x) = ','s');
f=eval(['@(x)',fx]) ;
fx=input('Enter the function ,F"(x) = ','s');
f1=eval(['@(x)',fx]) ;
a=input('Enter a = ');
s=1;
fprintf('N\t     \tX(i)\t\t      X\t\t       f(x)\t\t      Error\n');
for k=1:1:100
   x(k)=a-(f(a)/f1(a));
```

```
  fprintf('%g        %f       %f        %f        %f\n',k,a,x(k),f(x(k)),s);
  a=x(k);
  x(k+1)=a-(f(a)/f1(a));
  s=((abs(x(k+1)-x(k)))/abs(x(k+1)))*100;
  if s<=.0001
     break;
  end
end
fprintf('\n\nThe root is =%f',x(k));
```

Output :

```
Enter the function ,F(x) = x^3-5*x+3
Enter the function ,F"(x) =  3*x^2-5
Enter a = 1
```

N	X(i)	X	f(x)	Error
1	1.000000	0.500000	0.625000	1.000000
2	0.500000	0.647059	0.035620	22.727273
3	0.647059	0.656573	0.000177	1.449035
4	0.656573	0.656620	0.000000	0.007254

The root is =0.656620>>

<u>Name of the experiment</u> : Study of Secant method .

<u>Objective</u> :

i) To know how to write algorithm for Secant method .

ii) To know how to solve problem by programming .

<u>Theory for Secant method</u> :

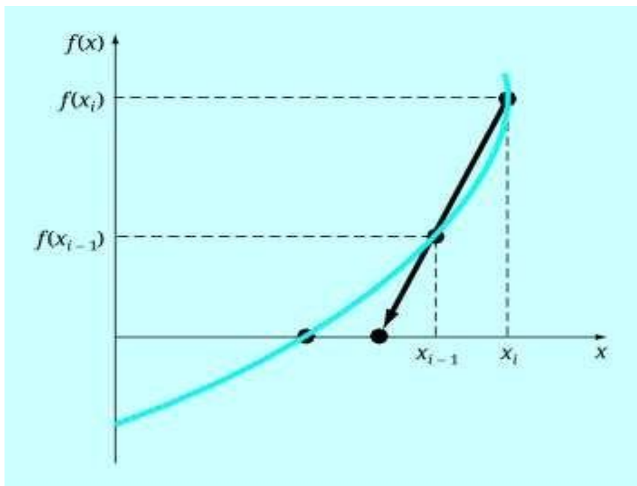


Fig : Graph for Secant Method .

In the secant method , the derivative at X_i is approximated by the formula

$$f'(X_i) = (f(X_i) - f(X_{i-1}))/(X_i - X_{i-1})$$

Which can be written as

$$f'_i = (f_i - f_{i-1})/(X_i - X_{i-1})$$

Where $f_I = f(X_I)$.

Hence , the Newton-Raphson formula becomes

$$X_{i+1} = X_I - (f_I(X_i - X_{i-1}))/(f_i - f_{i-1})$$

$$= (X_I f_{i-1} - X_{i-1} f_i)/(f_{i-1} - f_i)$$

It should be noted that this formula requires two initial approximations to the root .

Algorithm for Secant method :

1) Read x_0 ,x_1,e, delta , n
2) $f_0 \leftarrow f(x_0)$
3) $f_1 \leftarrow f(x_1)$
4) for i=1 to n
5) if $| f_1-f_0 |$ < delta then go to 15
6) $X_2 \leftarrow (x_0f_1-x_1f_1)/(f_1-f_0)$
7) $f_2 \leftarrow f(x_2)$
8) if $| f_2 | < e$ then goto 17
9) $f_0 \leftarrow f_1$
10) $f_1 \leftarrow f_2$
11) $x_0 \leftarrow x_1$
12) $x_1 \leftarrow x_2$

Endfor

13) write 'Does not converge ' , x_0 ,x_1,f_0,f_1
14) stop
15) write ' Slope too small ',i ,f_0, f_1, x_0,x_1
16) Stop
17) **write ' convergent solution ' , i,x_2 ,f_2**
18) **stop**

Program for Secant method method:

Example : Solve the equation by Secant method .

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

Program :

```
clc;
```

```
clear all;
fx=input('Enter the function ,F(x) = ','s');
f=eval(['@(x)',fx]) ;
a=input('Enter a=');
b=input('Enter b=');
x(1)=a;
x(2)=b;
s=1;
fprintf('N\t\t x(i-1)\t\t x(i)\t\t  x(i+1)\t\t f(x)\t\t   Error\n');
for k=3:103;
   it(k)=abs(k-2);
   x(k)=x(k-1)-(f(x(k-1))*(x(k-1)-x(k-2)))/(f(x(k-1))-f(x(k-2)));
   c=f(x(k));
   fprintf('%g      %f     %f      %f      %f      %f\n\n',it(k),x(k-2),x(k-1),x(k),c,s);
   s=((abs(x(k)-x(k-1)))/abs(x(k)))*100;
   if s<=.001
      break;
   end
end
fprintf('\n\nThe root is =%f',x(k));
```

Output :

Enter the function ,F(x) = x^3-2*x-5

Enter a=2

Enter b=3

N	x(i-1)	x(i)	x(i+1)	f(x)	Error
1	2.000000	3.000000	2.058824	-0.390800	1.000000
2	3.000000	2.058824	2.081264	-0.147204	45.714286
3	2.058824	2.081264	2.094824	0.003044	1.078197
4	2.081264	2.094824	2.094549	-0.000023	0.647333
5	2.094824	2.094549	2.094551	-0.000000	0.013116

The root is =2.094551>>

Name of the experiment : Study of the Gauss-elimination method .

Objective :

i) To know how to write algorithm for Gauss-elimination method .

ii) To know how to solve problem by programming .

Theory for Gauss-elimination method :

The approach is designed to solve a general set of n equations :

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 + \dots\dots + a_{1n} x_n = b_1 \quad \dots\dots\text{i}$$

$$a_{21} x_1 + a_{22} x_2 + a_{23} x_3 + \dots\dots + a_{2n} x_n = b_2 \quad \dots\dots\text{ii}$$

$$\cdot \qquad \cdot \qquad \cdot$$

$$\cdot \qquad \cdot \qquad \cdot$$

$$a_{n1} x_1 + a_{n2} x_2 + a_{n3} x_3 + \dots\dots + a_{nn} x_n = b_n \quad \dots\dots\text{iii}$$

The first phase is design to reduce the set of equations to an upper triangular system. The initial step will be to eliminate the first unknown, x_1 , from the second through the nth equations . To do this , multiply Equation (i) by a_{21}/ a_{11} to give

$$a_{21} x_1 + (a_{21}/a_{11}) a_{12} x_2 + \dots + (a_{21}/a_{11}) a_{1n} x_n = (a_{21}/a_{11}) b_1 \quad \dots\dots\dots\dots\text{iv}$$

Now , this equation can be subtracted from equation (ii) to give

$$(a_{22} - a_{21}/a_{11}\, a_{12}) x_2 + \dots + (a_{2n} - a_{21}/a_{11}\, a_{1n})\; x_n = b_2 - a_{21}/a_{11}\, b_1$$

Or $\quad a'_{22} x_2 + \dots + a'_{2n} x_n = b'_2$

Where the prime indicates that the elements have been changed from their original values .

Repeating the procedure for the remaining equations results in the following modified system :

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 + \ldots\ldots\ldots + a_{1n} x_n = b_1 \quad \ldots\ldots\ldots v$$

$$a'_{22} x_2 + a'_{23} x_3 + \ldots\ldots + a'_{2n} x_n = b'_2 \quad \ldots\ldots\ldots vi$$

$$a'_{32} x_2 + a'_{33} x_3 + \ldots\ldots + a'_{3n} x_n = b'_3 \quad \ldots\ldots\ldots vii$$

$$\vdots \qquad \qquad \vdots$$

$$a'_{n2} x_2 + a'_{n3} x_3 + \ldots\ldots + a'_{nn} x_n = b'_n \quad \ldots\ldots\ldots viii$$

Now repeat the above to eliminate the second unknown from equation (vii) through (viii) . To do this multiply equation (vi) by a'_{32}/a'_{22} and subtract the result from equation (vii) . Perform a similar elimination for the remaining equations to yield

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 + \ldots\ldots\ldots + a_{1n} x_n = b_1$$

$$a'_{22} x_2 + a'_{23} x_3 + \ldots\ldots + a'_{2n} x_n = b'_2$$

$$a''_{33} x_3 + \ldots\ldots + a''_{3n} x_n = b''_3$$

$$\vdots \qquad \qquad \vdots$$

$$a''_{n3} x_3 + \ldots\ldots + a''_{nn} x_n = b''_n$$

The procedure can be continued using the remaining pivot equations . The final multiplication in the sequence is to use the (n-1) th equation to eliminate the x_{n-1} term from the nth equation . At the point , the system will have been transformed to an upper triangular system :

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 + \ldots\ldots\ldots + a_{1n} x_n = b_1 \quad \ldots\ldots\text{ix}$$

$$a'_{22} x_2 + a'_{23} x_3 + \ \ldots\ldots + a'_{2n} x_n = b'_2 \quad \ldots\ldots\text{x}$$

$$. \qquad a''_{33} x_3 + \ \ldots\ldots + a''_{3n} x_n = b''_3 \quad \ldots\ldots\ \text{xi}$$

$$. \qquad\qquad . \qquad\qquad\qquad .$$

$$. \qquad\qquad . \qquad\qquad\qquad .$$

$$. \qquad\qquad . \qquad\qquad + a^{(n-1)}_{nn} x_n = b^{(n-1)}_{n} \quad \ldots\text{xii}$$

Algorithm for Gauss-elimination method :

1) Start
2) Declare the variables and read the order of the matrix n .
3) Take the coefficients of the linear equation as:

    Do for k=1 to n

    Do for j=1 to n+1

    Read a[k][j]

    End for j

    End for k

4) Do for k=1 to n-1

    Do for i=k+1 to n

    Do for j=k+1 to n+1

    a[i][j]=[i][j] – a[i][k]/a[k][k]*a[k][j]

    End for j

    End for i

    End for k

5) Compute x[n]=a[n][n+1]/a[n][n]
6) Do for k=n-1 to 1

    Sum=0

    Do for j=k+1 to n

Sum=sum + a[k][j]*x[j]

End for j

X[k]=1/a[k][k]*(a[k][n+1]-sum)

End for k

7) Display the result x[k]
8) Stop

Program for gauss-elimination method:

Use Gauss-elimination to solve

$$2\,x_1 + 3\,x_2 + 1\,x_3 = 9 \qquad \ldots\ldots\ldots\ldots\ldots\ldots\text{i}$$

$$1\,x_1 + 2\,x_2 + 3\,x_3 = 6 \qquad \ldots\ldots\ldots\ldots\ldots\ldots\text{ii}$$

$$3\,x_1 + 1\,x_2 + 2\,x_3 = 8 \qquad \ldots\ldots\ldots\ldots\ldots\ldots\text{iii}$$

Program :

```
clc;
clear all;
a=input('Enter matrix A = ');
b=input('Enter matrix B = ');
[m,n]=size(a);
for k=1:m-1
  for i=k+1:m
      fact=a(i,k)/a(k,k);
        for j=1:n
            a(i,j)=a(i,j)-a(k,j)*fact;
        end
        b(i,1)=b(i,1)-b(k,1)*fact;
  end
end
x(m)=b(m,1)/a(m,n);
```

```
for i=m-1:-1:1
   sum=0;
   for j=i+1:n
      sum=sum+a(i,j)*x(j);
   end
   x(i)=(b(i,1)-sum)/a(i,i);
end
disp('After forward elimination the matrix [A B] :');
disp([a b]); %%Showes a &b in matrix form
fprintf('\nThe Required solution : ');
for i=1:n
fprintf('\nx(%d) = %f',i,x(i));
end
```

Output :

```
Enter matrix A = [2 3 1;1 2 3;3 1 2]
Enter matrix B = [9;6;8]
After forward elimination the matrix [A B] :
    2.0000    3.0000    1.0000    9.0000
         0    0.5000    2.5000    1.5000
         0         0   18.0000    5.0000
The Required solution :
x(1) = 1.944444
x(2) = 1.611111
x(3) = 0.277778>>
```

Name of the experiment : Study of the Linear Regression method .

Objective :

i) To know how to write algorithm for Linear Regression method .

ii) To know how to solve problem by programming .

Theory for Linear Regression method :

The simplest example of a least-squares approximation is fitting a straight line to a set of paired observations : (x_1, y_1), (x_2, y_2),….. (x_n, y_n). The mathematical expression for the straight line is

$$y = a_0 + a_1 x + e \qquad \text{………………..i}$$

where a_0 and a_1 are coefficients representing the intercept and the slope , respectively, and e is the error or residual, between the model and the observations, which can be rearranging equation (i) as

$$e = y - a_0 - a_1 x$$

Thus the error, or residual, is the discrepancy between the true value of y and the approximate value, $a_0 + a_1 x$, predicted by the linear equation .

Therefore, another logical criterion might be to minimize the of the absolute value of the discrepancies, as in

$$\sum_{i=1}^{n} |e_i| = \sum_{i=1}^{n} |y_i - a_0 - a_1 x_i|$$

$$S_r = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_{i.measured} - y_{i.model})^2$$

$$= \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2 \quad \text{......................ii}$$

To determine values for a_0 and a_1, equation (ii) is differentiated with respect to each coefficient :

$$\frac{\partial Sr}{\partial ao} = -2 \sum (y_i - a_0 - a_1 x_i)$$

$$\frac{\partial Sr}{\partial a1} = -2 \sum [(y_i - a_0 - a_1 x_i) x_i]$$

Setting these derivatives equal to zero will result in a minimum S_r .

If this is done , the equations can be expressed as

$$0 = \sum y_i - \sum a_0 - \sum a_1 x_i$$

$$0 = \sum y_i x_i - \sum a_0 x_i - \sum a_1 x_i^2$$

Now realizing that $\sum a_0 = na_0$, we can express the equations as a set of two simultaneous linear equations with two unknowns (a_0 and a_1) :

$$na_0 + (\sum x_i) a_1 = \sum y_i \quad \text{................iii}$$

$$(\sum x_i) a_0 + (\sum x_i^2) a_1 = \sum x_i y_i \quad \text{...............iv}$$

These are called the normal equations . They can be solved simultaneously

$$a_1 = \frac{n \sum xiyi - \sum xi \sum yi}{n \sum xi^2 - (\sum xi)^2} \quad \text{.........................v}$$

This result can be used in conjunction with equation (iii) to solve for

$$A_0 = \acute{y} - a_1 x' \quad \text{..........................vi}$$

Where $\acute{y}$ and x' are the means of y and x , respectively .

Algorithm for Linear Regression method :

1) Read n
2) sum x ← 0
3) sum xsq ← 0
4) sum y ← 0
5) sum xy ← 0
6) for i = 1 to n do
7) Read x , y
8) sum x ← sum x + x
9) sum xsq ← sum xsq + x^2
10) sum y ← sum y + y
11) sum xy ← sum xy + x *y
    endfor
12) denom ← n x sum xsq – sum x * sum x
13) a_0 ← (sum y x sum xsq – sum x * sum xy)/denom
14) a_1 ← (n x sum xy – sum x * sum y)/denom
15) Write a_1 , a_0
16) Stop

Program for Linear Regression method :

```
Clc;
clear all;
n=input('inter the value of n=');
x0=0;
xsq0=0;
y0=0;
xy0=0;
for i=1:n
   x=input('enter the value of x=');
   y=input('enter the value of y=');
```

```
   x0=x0+x;

   xsq0=xsq0+(x*x);

   y0=y0+y;

   xy0=xy0+(x*y);

end

d=n*xsq0-x0*x0;

a=(y0*xsq0-x0*xy0)/d;

b=(n*xy0-x0*y0)/d;

a
b
 Printf('y=%g x + %g ',b,a);
```

Output :

```
Enter the value of n = 7
Enter the value of x = 1
Enter the value of y = 2
 Enter the  value of x = 2
Enter the value of y = 5
Enter the value of x = 4
Enter the value of y = 7
 Enter the value of x = 5
Enter the value of y = 10
Enter the value of x = 6
Enter the value of y = 12
Enter the value of x = 8
Enter the value of y = 15
Enter the value of x = 9
Enter the value of y = 19
   a =0.0962
   b =1.9808
   y=1.9808 x + 0.0962
```

<u>Name of the experiment</u> : Study of the Trapezoidal Rule .

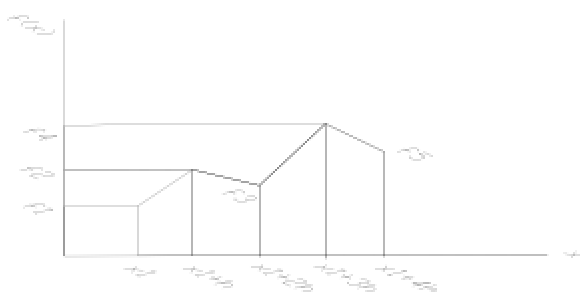
<u>Objective</u> :

i) To know how to write algorithm for Trapezoidal rule .

ii) To know how to solve problem by programming .

<u>Theory for Trapezoidal Rule</u> :

Formulae for numerical integration called quadrature are also based on fitting a polynomial through a specified set of points and integrating this approximating function . Assume that the values of a function f (x) are given at $x_1 + h$, $x_1 + 2h$ .... $x_1 + nh$ and that it is required to find the integral of f(x) between x_1 and $x_1 + nh$. The simplest technique to use would be to fit straight lines through f(x_1) , f ($x_1 + nh$) ..... and determine the area under this approximating function as shown in below figure .



The equation of the straight line between x_i and $x_i + h$ is given by

$$f(x) = f(x_i) + \frac{f(Xi+1) - f(Xi)}{h}(x - x_i)$$

$$S = \int_{X1}^{Xn} f(x)\, dx$$

$$= \sum_{i=1}^{n} \cdot \int_{Xi}^{Xi+h} . [f_i + \frac{\Delta fi}{h} (x - x_i)]\, dx$$

$$= \sum_{i=1}^{n} \cdot h/2(f_i + f_{i+1}) \quad \ldots\ldots\ldots\ldots\ldots\ldots.i$$

The integration formula of equation is known as Trapezoidal rule . The integral may be rewritten as

$$S = (f_1/2 + f_2 + f_3 + f_4 + \ \ldots\ldots \ + f_{n+1}/2)\, h$$

Algorithm for Trapezoidal Rule :

1) Read n , h
2) for i = 1 to n+1

   Read f_i endfor
3) sum ← ($f_1 + f_{n+1}$)/2
4) for j = 2 to n do
5) sum ← sum + f_j

   Endfor
6) Integral ← h x sum
7) Write Integral
8) Stop

Program for Trapezoidal Rule :

Table of x vs f(x) is given below . Integrate the function using Trapezoidal Rule

x (angle radians)	0.25	0.26	0.27	0.28	0.29
f(x)=sin x	0.2474	0.2571	0.2667	0.2764	0.2860

Program :

```
clc

clear all

n=input('Enter the value n= ');

h=input('Enter the value h= ');
```

```
for i=1:n+1
   f(i)=input('Enter the value f(i)= ');
end
sum=(f(1)+f(n+1))/2;
for j=2:n
   sum=sum+f(j);
end
Integral=h*sum;
Integral
```

Output :

```
Enter the value n= 4
Enter the value h= .01
Enter the value f(i)= .2474
Enter the value f(i)= .2571
Enter the value f(i)= .2667
Enter the value f(i)= .2764
Enter the value f(i)= .2860
Integral =  0.0107>>
```

Name of the experiment : Study of the simpson's Rule .

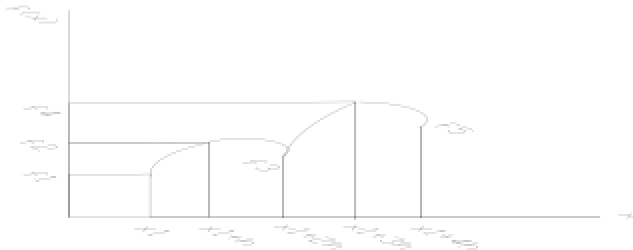
Objective :

i) To know how to write algorithm for simpson's rule .

ii) To know how to solve problem by programming .

Theory for simpson's Rule :

Simpson's rule is a popular numerical integration technique . It is based on approximating the function f(x) by fitting quadratics through sets of three points . This is illustrated graphically in figure .



Using a quadratic interpolation polynomial we get

$$S = \int_{X1}^{X1+nh} f(x)\, dx$$

$$= \sum_{i=1,3,..n-2} \cdot \int_{Xi}^{Xi+2h} . \left[f_i + \frac{\Delta fi}{h}(x - x_i) \right.$$

$$\left. + \Delta^2 f_i/2h^2 (x - x_i) (x - x_i - h) \right] dx \quad .....i$$

Integrating the above expression we obtain

$$S = \sum_{i=1,3,..n-2} . h/3(f_i + 4 f_{i+1} + f_{i+2}) \quad ....ii$$

$$= h/3(f_1 + 4 f_2 + 2f_3 + 4f_4 + \ ...... \ + f_{n+1}) \quad ....iii$$

It using the above formula it is implied that f is tabulated at an odd number of points .

<u>Algorithm for simpson's Rule</u> :

Remarks : Assume function tabulated at odd number of points (n+1)

1) Read n , h
2) for i = 1 to n+1

   Read f_i endfor
3) sum ← ($f_1 + f_{n+1}$)
4) for i = 2 to n in steps of 2 do
5) sum ← sum + 4 x f_i

   endfor

6) for i = 3 to n-1 in steps of 2 do

7)) sum ← sum + 2 x f_i

endfor

8) Integral ← h x sum/3

9) Write Integral

10) Stop

Program for simpson's Rule :

Table of x vs f(x) is given below . Integrate the function using Simpson's Rule

x (angle radians)	0.25	0.26	0.27	0.28	0.29
f(x)=sin x	0.2474	0.2571	0.2667	0.2764	0.2860

Program :

```
clc
clear all
n=input('Enter the value of n= ');
h=input('Enter the value of h= ');
for i=1:n+1
  f(i)=input('Enter the value of f(i)= ');
end
  sum=(f(1)+f(n+1));
  for i=2:n
    sum=sum+4*f(i);
  end
  for i=3:n-1
    sum=sum+2*f(i)
  end
  Integral=h*(sum/3);
```

Integral

Output :

Enter the value of n= 4

Enter the value of h= .01

Enter the value of f(i)= .2474

Enter the value of f(i)= .2571

Enter the value of f(i)= .2667

Enter the value of f(i)= .2764

Enter the value of f(i)= .2860

sum = 4.2676

Integral = 0.0142

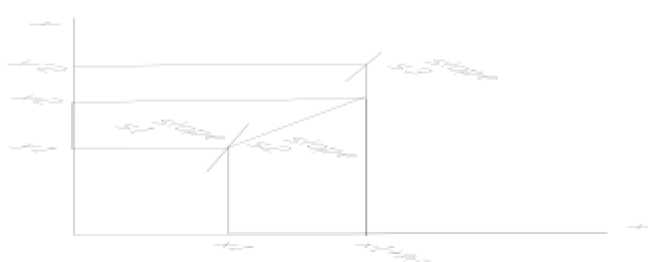
Name of the experiment : Study of the Runge-Kutta method .

Objective :

i) To know how to write algorithm for Runge-Kutta method .

ii) To know how to solve problem by programming .

Theory for Runge-Kutta method :



Consider the following geometric method of extrapolating the y(x) curve to obtain the solution to the differential equation

$$\frac{dy}{dx} = f(x, y)$$

$$y(x_1) = y_1$$

Draw a straight line from (x_1 , y_1) with a slope

$$s_1 = f(x_1, y_1)$$

Let it cut the vertical line through $x_1 + h$ at (x_1+h , y'_2) . Determine the slope $\frac{dy}{dx}$ of the solution curve y(x) at (x_1+h , y'_2) . This is given by

$$S_2 = f(x_2, y'_2)$$

($x_2 = x_1+h$) . Now draw a straight line from (x_1 , y_1) with a slope ($s_1 + s_2$)/2 . The point y_2 where this straight line cuts the vertical line at x_1+h is the approximating solution of the differential equation at x_1+h. Thus we have :

$$y_2 = y_1 + (s_1 + s_2)h/2 \quad \text{..................i}$$

In general the (i+1)th point is obtained from the i-th point using the formula

$$Y_{i+1} = y_i + (s_i + s_{i+1})h/2 \quad \text{............ii}$$

Where $s_i = f(x_i, y_i)$ and

$$s_{i+1} = f(x_{i+1}, y_i + s_i h)$$

This method is called a second order Runge-Kutta method (also Henu's method) .

Algorithm for Runge-Kutta method :

1) Read function f (x , y)

2) Read x_1 , y_1 , h , x_f

3) While $x_1 \leq x_f$ do

Begin

4) Write x_1 , y_1

5) $s_1 \leftarrow f(x_1, y_1)$

6) $x_2 \leftarrow x_1 + h$

7) $y_2 \leftarrow y_1 + hs_1$

8) $s_2 \leftarrow f(x_2, y_2)$

9) $y_2 \leftarrow y_1 + h \times (s_1 + s_2)/2$

10) $x_1 \leftarrow x_2$

11) $y_1 \leftarrow y_2$

end

12) Stop

Program for Runge-Kutta method :

$$\frac{dy}{dx} + xy = 0 \ , \ x = 0 \ , \ y(0) = 1$$

Solve the equation using Runge-Kutta method .

Program :

```
clc
clear all
fx=input('Enter the function ,dx/dy = ','s');
f=eval(['@(x,y)',fx]) ;
x1=input('initial value of x1= ');
y1=input('initial value of y1= ');
xp=input('input x at which y is required xp= ');
h=input('input step size h= ');
n=(xp-x1)/h;
for i=1:n
   m1=f(x1,y1);
   x2=x1+h;
   y2=y1+h*m1;
   m2=f(x2,y2);
   y2=y1+((h/2)*(m1+m2));
```

```
   x1=x2;
   y1=y2;
end
fprintf('At x=%g the value of y(%g)=%f',xp,xp,y2);
```

Output :

```
Enter the function ,dx/dy = -x*y
initial value of x1= 0
initial value of y1= 1
input x at which y is required xp= .25
input step size h= .05
At x=0.25 the value of y(0.25)=0.969230>>
```