Experiment No.: 01

Name of the Experiment:

Determining the root of a non-linear equation using Bisection Method.

Objectives:

- Getting introduced with Bisection Method.
- Determining the roots of non-linear equations in C.
- Determining the roots of non-linear equations in Microsoft Excel.
- Making comparison of experimental results in C and in Microsoft Excel.

Theory:

The Bisection Method is one of the simplest and most reliable of iterative method for the solution of non-linear equations. This method is also known as binary chopping or half interval method. It relies on the fact that if $f(x)$ is real and continuous in the interval $a < x < b$, and $f(a)$ and $f(b)$ are of opposite signs, that is

$$f(a) * f(b) < 0$$

Then there is at least one real root in the interval between a and b . That is,

$$x_0 = (x_1 + x_2)/2$$

Now there exist following three conditions:

1. If $f(x_0) = 0$, we have a root at x_0 .
2. If $f(x_0)f(x_1) < 0$, there is a root between x_0 and x_1
3. If $f(x_0)f(x_2) < 0$, there is a root between x_0 and x_2

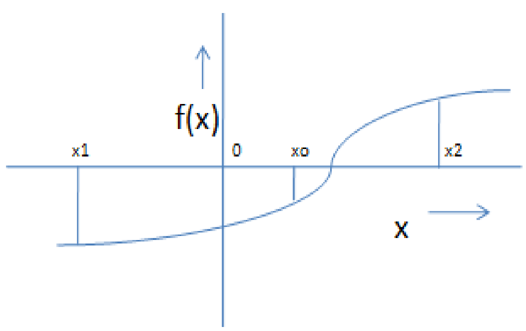


Figure: Illustration of Bisection Method

Algorithm for Bisection Method:

1. Decide initial values for x_1 and x_2 and stopping criterion, E.
2. Computing $f1 = f(x_1)$ and $f2 = f(x_2)$
3. If $f1 * f2 > 0, x_1$ and x_2 do not bracket any root and go to step 7.
4. Compute $x_0 = (x_1 + x_2)/2$ and compute $f_0 = f(x_0)$

5. If $f_1 * f_0 < 0$ then

   set $x_2 = x_0$

   else

   set $x_1 = x_0$

   set $f_1 = f_0$

6. If absolute value of $(x_2 - x_1)/x_2$ is less than error E, then

   root=$(x_1 + x_2)/2$

   write the value of root,

   go to step 7

   else

   go to step 4

7. Stop.

C code of Bisection Method:

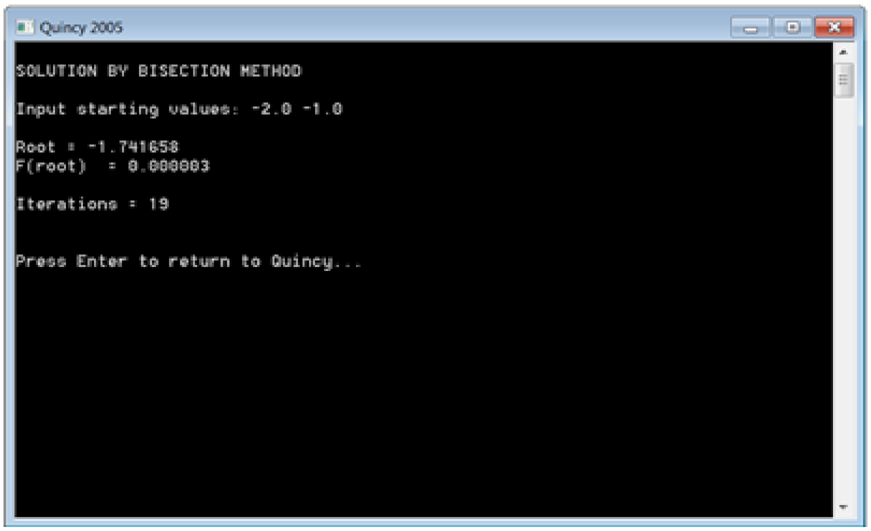
```
/* Write a C program to find out a real root of the following non-linear equation using Bisection
method:
            x^2 - 4x - 10 = 0
    Done by: Kamelia Zaman Moon, Class Roll: 299
    Date: 21/11/2018
*/
#include<stdio.h>
#include<math.h>
#define EPS 0.000001
#define F(x) (x)*(x)-4*(x)-10
int main()
{
   int s,count;
   float a,b,root;
   printf("Solution by Bisection method\n Input starting values\n");
   scanf("%f %f",&a,&b);
   bim(&a,&b,&root,&s,&count);
   if(s==0)
      printf("Starting points do not bracket any roots\n Check whether they bracket Even
roots\n");
   else
      printf("Root= %f\n F(root)= %f\n Iterations= %d\n",root,F(root),count);
}
```

```
bim(float *a, float *b, float *root, int *s, int *count)
{
    float x1,x2,x0,f1,f2,f0;
    x1=*a;
    x2=*b;
    f1=F(x1);
    f2=F(x2);
    if(f1*f2 >0)
    {
        *s=0;
        return;
    }
    else
    {
        *count=0;
begin:
        x0=(x1+x2)/2.0;
        f0=F(x0);
        if(f0==0)
        {
            *s=1;
            *root=x0;
            return;
        }
        if(f1*f2 <0)
            x2=x0;
        else
        {
            x1=x0;
            f1=f0;
        }
        if(fabs((x2-x1)/x2)<EPS)
        {
            *s=1;
            *root=(x1+x2)/2.0;
            return;
        }
        else
        {
            *count=*count+1;
```

```
            goto begin;
        }
    }
}
```

Output:

Solution by Bisection method
Input starting values: -2.0 -1.0
Root = -1.741658
F(root) = 0.000003
Iterations = 19



Bisection Method in Microsoft Excel:

Experiment Name: Find the root of the following equation using Bisection Method:

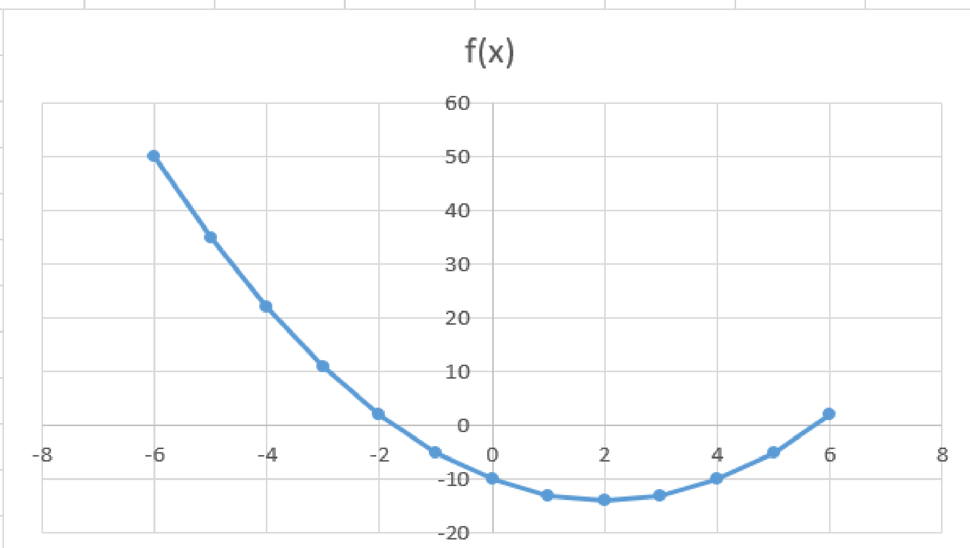
$$f(x) = x^2 - 4x - 10$$

Therefore, range of X= $\sqrt{((a_{n-1}/a_n)^\wedge 2) - (2 * (a_{n-2}/a_n))}$
= 6

$$f(x) = x^2 - 4x - 10$$

x	f(x)	x1	x2	x0	f(x0)	f(x1)	f(x2)	f(x0)f(x1)	f(x0)f(x2)
-6	50	-2	-1	-1.5	-1.75	2	-5	-3.5	8.75
-5	35	-2	-1.5	-1.75	0.0625	2	-1.75	0.125	-0.109375
-4	22	-1.75	-1.5	-1.625	-0.859375	0.0625	-1.75	-0.05371094	1.50390625
-3	11	-1.75	-1.625	-1.6875	-0.4023438	0.0625	-0.859375	-0.02514648	0.34576416
-2	2	-1.75	-1.6875	-1.71875	-0.1708984	0.0625	-0.4023438	-0.01068115	0.06875992
-1	-5	-1.75	-1.7188	-1.734375	-0.0544434	0.0625	-0.1708984	-0.00340271	0.00930429
0	-10	-1.75	-1.7344	-1.742188	0.0039673	0.0625	-0.0544434	0.000247955	-0.00021599
1	-13	-1.7422	-1.7344	-1.738281	-0.0252533	0.0039673	-0.0544434	-0.00010019	0.00137487
2	-14	-1.7422	-1.7383	-1.740234	-0.0106468	0.0039673	-0.0252533	-4.2239E-05	0.00026887
3	-13	-1.7422	-1.7402	-1.741211	-0.0033407	0.0039673	-0.0106468	-1.3254E-05	3.5568E-05
4	-10	-1.7422	-1.7412	-1.741699	0.000313	0.0039673	-0.0033407	1.24193E-06	-1.0458E-06
5	-5	-1.7417	-1.7412	-1.741455	-0.0015139	0.000313	-0.0033407	-4.7392E-07	5.0575E-06
6	2	-1.7417	-1.7415	-1.741577	-0.0006004	0.000313	-0.0015139	-1.8796E-07	9.0901E-07

Sheet1



Result:

After 1st iteration the root is -1.5
After 2nd iteration the root is -1.75
After 3rd iteration the root is -1.625
After 5th iteration the root is -1.71875

After 10th iteration the root is -1.74121
After 15th iteration the root is -1.74167
Approximately the root is -1.74166

Discussion:

The root is not totally accurate. The root has been taken when the interval between x1 and x2 is equal to 1.91E-06. After 20th iteration the difference is 1.91E-06. This is the error of this calculation. The amount of error is too little that it can be avoided. So, -1.74166 can be considered as the root of the equation $x^2 - 4x - 10 = 0$.

Experiment No.: 02

Name of the Experiment:

Determining the root of a non-linear equation using False-Position Method.

Objectives:

- Getting introduced with False-Position Method.
- Determining the roots of non-linear equations in C.
- Determining the roots of non-linear equations in Microsoft Excel.
- Making comparison of experimental results in C and in Microsoft Excel.

Theory:

In Bisection Method, the interval between x_1 and x_2 is divided into who equal halves, irrespective of location of the root. It may be possible that the root is closer to one end than the other as shown below. The root is closer to x_1. Joining the points x_1 and x_2 by a straight line, the point of intersection of this line with the x-axis gives an improved estimate of the root and is called the false position of the root. This point then replaces one of the initial guesses that has a function value of the same sign as $f(x_0)$. The process is repeated with the new values of x_1 and x_2.

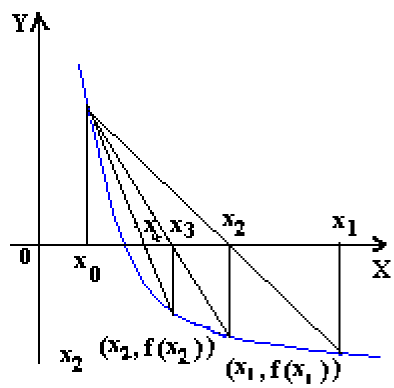


Figure: Illustration of False-Position Method

We know that equation of the line joining the points $(x_1, f(x_1))$ and $(x_2, f(x_2))$ is given by

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

Since the line intersects the x-axis at x_0, when $x = x_0, y = 0$, we have

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

Then, we have

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

This equation is known as the False-Position Formula.

Algorithm for False-Position Method:

Having calculated the first approximate to the root, the process is repeated for the new interval, as done in the bisection method, using algorithm below

1. Decide initial values for x_1 and x_2 and stopping criterion, E.
2. Computing $f1 = f(x_1)$ and $f2 = f(x_2)$
3. If $f1 * f2 > 0, x_1$ and x_2 do not bracket any root and go to step 6.
4. Let $x_0 = x_1 - f(x_1) * \frac{(x_2 - x_1)}{f(x_2) - f(x_1)}$
5. If $(x_0) * f(x_1) < 0$
   set $x_2 = x_0$
   otherwise
   set $x_1 = x_0$
6. Stop

C code of False-Position Method:

/* Write a C program to find out a real root of the following non-linear equation using False-Position method:

$x^2 - x - 2 = 0$

Done by: Kamelia Zaman Moon, Class Roll: 299

Date: 21/11/2018

*/

```
#include<stdio.h>
#include<math.h>
#define EPS 0.000001
#define F(x) (x)*(x)-(x)-2
int main()
{
   int s,count;
   float a,b,root;
   printf("Solution by False-position method\n Input starting values\n");
   scanf("%f %f",&a,&b);
   fal(&a,&b,&s,&root,&count);
   if(s==0)
```

```
        printf("Starting points do not bracket any root\n");
    else
        printf("Root=%f\n F(root)=%f\n Iterations=%d\n",root,F(root),count);
}
fal(float *a, float *b, int *s, float *root, int *count)
{
    float x1,x2,x0,f1,f2,f0;
    x1=*a;
    x2=*b;
    f1=F(x1);
    f2=F(x2);
    if(f1*f2 >0)
    {
        *s=0;
        return;
    }
    else
        printf("   \n        x1          x2\n");
    *count=1;
begin:
    x0= x1-f1*(x2-x1)/(f2-f1);
    f0=F(x0);
    if(f1*f2<0)
    {
        x2=x0;
        f2=f0;
    }
    else
    {
        x1=x0;
        f1=f0;
    }
    printf("%5d %15.6f %15.6f\n",*count,x1,x2);
    if(fabs(x2-x1)/x2<EPS)
    {
        *s=1;
        *root=(x1+x2)/2.0;
        return;
    }
    else
```

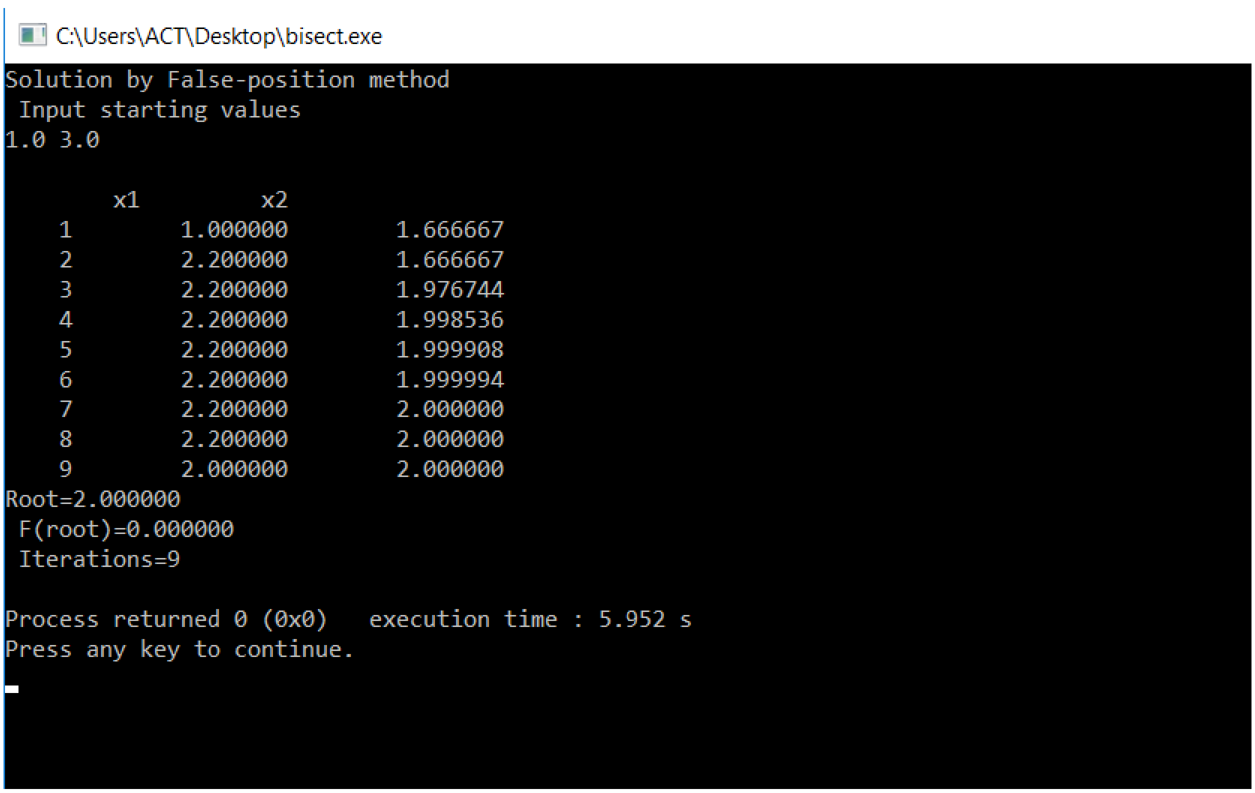
```
    {
        *count=*count+1;
        goto begin;
    }
}
```

Output:

Solution by False-position method
Input starting values -2.0 -1.0

	x1	x2
1	1.000000	1.666667
2	2.200000	1.666667
3	2.200000	1.976744
4	2.200000	1.998536
5	2.200000	1.999908
6	2.200000	1.999994
7	2.200000	2.000000
8	2.200000	2.000000
9	2.000000	2.000000

Root=2.000000
F(root)=0.000000
Iterations=9



```
C:\Users\ACT\Desktop\bisect.exe
Solution by False-position method
 Input starting values
1.0 3.0

        x1         x2
    1        1.000000        1.666667
    2        2.200000        1.666667
    3        2.200000        1.976744
    4        2.200000        1.998536
    5        2.200000        1.999908
    6        2.200000        1.999994
    7        2.200000        2.000000
    8        2.200000        2.000000
    9        2.000000        2.000000
Root=2.000000
 F(root)=0.000000
 Iterations=9

Process returned 0 (0x0)   execution time : 5.952 s
Press any key to continue.
```

False-Position Method in Microsoft Excel:

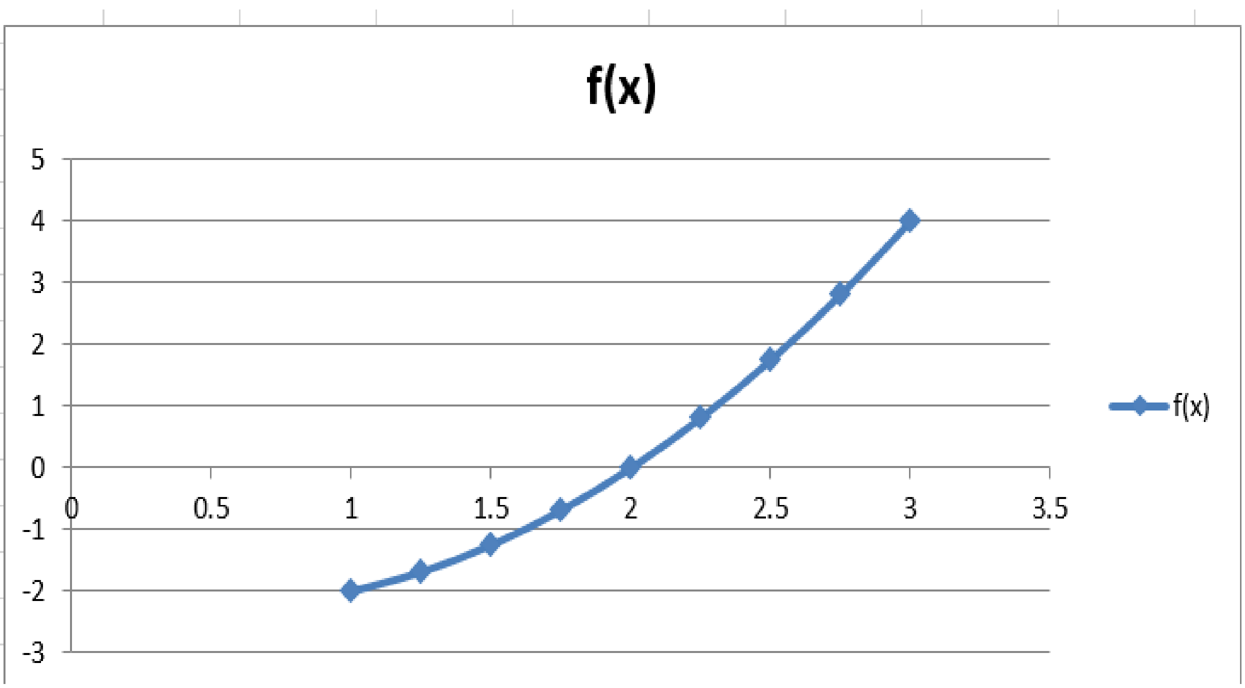
Experiment Name: Find the root of the following equation using False-Position Method:

$$f(x) = x^2 - x - 2$$

Given range of x is $1 < x < 3$

$$f(x) = x^2 - x - 2$$

x	f(x)	x1	x2	x0	f(x1)	f(x2)	f(x0)	f(x1)f(x0)	f(x2)f(x0)
1	-2	1	3	1.666667	-2	4	-0.888888889	1.777777778	-3.55556
1.25	-1.6875	1.666667	3	1.909091	-0.88889	4	-0.26446281	0.235078053	-1.05785
1.5	-1.25	1.909091	3	1.976744	-0.26446	4	-0.069226609	0.018307864	-0.27691
1.75	-0.6875	1.976744	3	1.994152	-0.06923	4	-0.017509661	0.001212134	-0.07004
2	0	1.994152	3	1.998536	-0.01751	4	-0.004390243	7.68717E-05	-0.01756
2.25	0.8125	1.998536	3	1.999634	-0.00439	4	-0.001098365	4.82209E-06	-0.00439
2.5	1.75	1.999634	3	1.999908	-0.0011	4	-0.000274641	3.01656E-07	-0.0011
2.75	2.8125	1.999908	3	1.999977	-0.00027	4	-6.86635E-05	1.88578E-08	-0.00027
3	4	1.999977	3	1.999994	-6.9E-05	4	-1.71661E-05	1.17868E-09	-6.9E-05



Result:

After 1st iteration the root is 1.6667
After 2nd iteration the root is 1.909
After 3rd iteration the root is 1.978
After 5th iteration the root is 1.998
After 6th iteration the root is 1.9996
After 8th iteration the root is 1.9999
Approximately the root is 2.0000

Discussion:

The root is not totally accurate. The root has been taken when the interval between x1 and x2 is equal to 1.91E-06. After 6th iteration the difference is 1.91E-06. This is the error of this calculation. The amount of error is too little that it can be avoided. So, 2.000 can be considered as the root of the equation $x^2 - x - 2 = 0$.

Experiment No.: 03

Name of the Experiment:

Determining the root of a non-linear equation using Newton-Raphson Method.

Objectives:

- Getting introduced with Newton-Raphson Method.
- Determining the roots of non-linear equations in C.
- Determining the roots of non-linear equations in Microsoft Excel.
- Making comparison of experimental results in C and in Microsoft Excel.

Theory:

Let us assume that x_1 is an approximate root of $f(x) = 0$ in the graph of $f(x)$ as shown below. There is a tangent drawn in the figure. The point of intersection of this tangent with the x-axis gives the second approximation of the root. Let the point of intersection be x_2. The slope of the tangent is given by

$$\tan \propto = \frac{f(x_1)}{x_1 - x_2} = f'(x_1)$$

where $f'(x_1)$ is the slope of $f(x)$ at $x = x_1$. Solving for x_2, we obtain

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

This is called the Newton-Raphson formula.

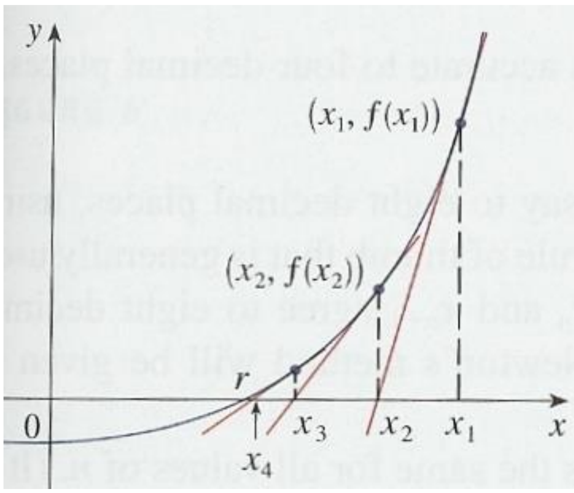


Figure: Illustration of Newton-Raphson Method

The next approximation would be

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

In general,

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

This method of successive approximation is called Newton-Raphson method.

Algorithm for Newton-Raphson Method:

1. Assign an initial value of x, say x_0.
2. Evaluate $f(x_0)$ and $f'(x_0)$
3. Find the improved estimate of x_0
   $x_1 = x_0 - \frac{f(x_0)}{f\prime(x_0)}$
4. Check for accuracy of the latest estimate.
   Compare relative error to a predefined value E. If $\left|\frac{x_1 - x_0}{x_1}\right| \leq$ E stop; otherwise continue.
5. Replace x_0 by x_1 and repeat steps 3 and 4.

C code of Newton-Raphson Method:

```
/* Write a C program to find out a real root of the following non-linear equation using Newton-
Raphson method:
            x^2 − 3x + 2 = 0
    Done by: Kamelia Zaman Moon, Class Roll: 299
    Date: 21/11/2018
*/
#include<stdio.h>

#include<math.h>

#define EPS 0.000001

#define F(x) (x)*(x)-3*(x)+2

#define FD(x) 2*(x)-3

int main()

{

   int count;

   float x0,xn,fx,fdx;

   printf("Input initial value of x\n");
```

```
    scanf("%f",&x0);
    printf("Solution by Newton-Raphson method\n");
    count=1;
begin:
    fx=F(x0);
    fdx=FD(x0);
    xn=x0-fx/fdx;
    if(fabs((xn-x0)/xn)<EPS)
        printf("Root=%f\n Function value=%f\n Number of iterations=%d\n",xn,F(xn),count);
    else
    {
        x0=xn;
        count=count+1;
        goto begin;
    }
}
```

Output:

Input initial value of x: 0

Solution by Newton-Raphson method

Root=1.000000

Function value=0.000000

Number of iterations=6

```
C:\Users\ACT\Desktop\bisect.exe
Input initial value of x
0
Solution by Newton-Raphson method
Root=1.000000
 Function value=0.000000
 Number of iterations=6

Process returned 0 (0x0)   execution time : 2.126 s
Press any key to continue.
```

Newton-Raphson Method in Microsoft Excel:

Experiment Name: Find the root of the following equation using Newton-Raphson Method:

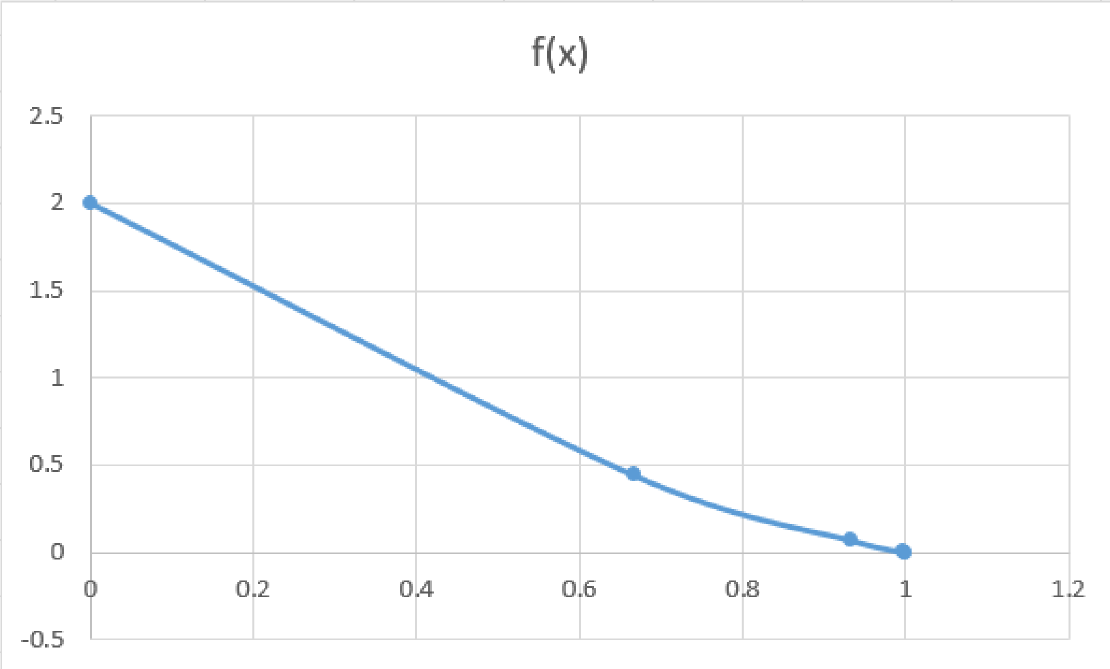
$$f(x) = x^2 - 3x + 2$$

In the vicinity of x=0

$$f(x) = x^2 - 3x + 2$$

$$f'(x) = 2x - 3$$

x	f(x)	f'(x)
0	2	-3
0.6667	0.4444	-1.6667
0.9333	0.0711	-1.1333
0.9961	0.0039	-1.0078
1	2E-05	-1
1	2E-10	-1



Result:

After 1st iteration the root is 0
After 2nd iteration the root is 0.6667
After 3rd iteration the root is 0.9333
After 4th iteration the root is 0.9961
After 5th iteration the root is 1
After 6th iteration the root is 1
Approximately the root is 1

Discussion:

The root is not totally accurate. The root has been taken when the initial value is 0. After 6th iteration the difference between 5th and 6th iteration is 0. The amount of error is too little that it can be avoided. Now we stop our work. So, 1.000 can be considered as the root of the equation $x^2 - 3x + 2 = 0$.

Experiment No.: 04

Name of the Experiment:

To fit a curve using Least Square Method.

Objectives:

- Getting introduced with Least Square Method.
- To fit a curve in C.
- To fit a curve in Microsoft Excel.
- Making comparison of experimental results in C and in Microsoft Excel.

Theory:

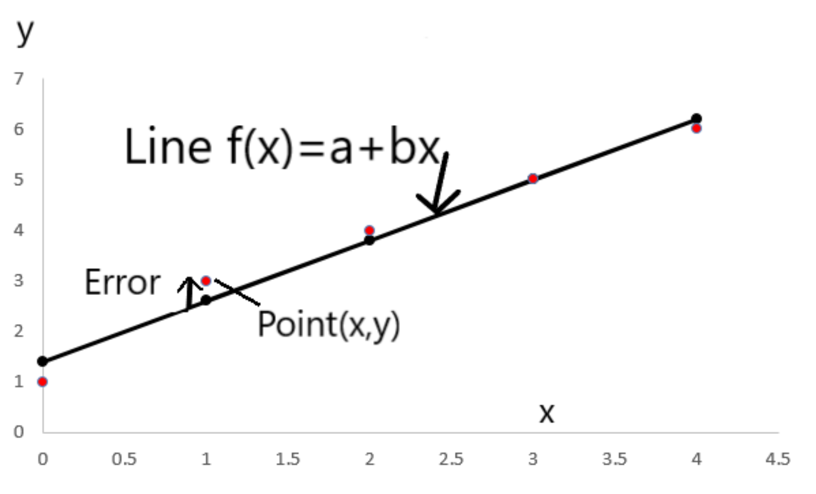
Fitting a straight line is the simplest approach of regression analysis. Let us consider the mathematical equation for a straight line

$$y = a + bx = f(x)$$

to describe the data. We know that a is the intercept of the line and b its slope. Consider a point (x_i, y_i) as shown in the figure below. The vertical distance of this point from the line

$f(x) = a + bx$ is the error q_i. Then,

$$q_i = y_i - f(x_i)$$
$$= y_i - a - bx_i$$



Let the sum of squares of individual errors be expressed as

$$Q = \sum_{i=1}^{n} q_i^2 = \sum_{i=1}^{n} [y_i - f(x_i)]^2$$
$$= \sum_{i=1}^{n} (y_i - a - bx_i)^2$$

In the method of least squares, we choose a and b such that Q is minimum. Since Q depends on a and b, a necessary condition for Q to be minimum is

$$\frac{\partial Q}{\partial a} = 0 \quad \text{and} \quad \frac{\partial Q}{\partial b} = 0$$

Then $$\frac{\partial Q}{\partial a} = -2\sum_{i=1}^{n}(y_i - a - bx_i) = 0$$

$$\frac{\partial Q}{\partial b} = -2\sum_{i=1}^{n} x_i(y_i - a - bx_i) = 0$$

Thus $$\sum y_i = na + b\sum x_i$$

$$\sum x_i y_i = a\sum x_i + b\sum {x_i}^2$$

These are called normal equations. Solving for a and b, we get

$$b = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum {x_i}^2 - (\sum x_i)^2}$$

$$a = \frac{\sum y_i}{n} - b\frac{\sum x_i}{n} = \bar{y} - b\bar{x}$$

where $\bar{x}$ and $\bar{y}$ are the averages of *x* values and *y* values, respectively.

Algorithm for Least Square Method:

1. Read data values.
2. Compute sum of powers and products
   $$\sum x_i, \sum y_i, \sum {x_i}^2, \sum x_i y_i$$
3. Check whether the denominator of the equation for *b* is zero.
4. Compute *b* and *a*.
5. Print out the equation.
6. Interpolate data, if required.

C code of Least Square Method:

/* Write a C program to fit a straight line to the following sets of data using Least Square method:

x	1	2	3	4	5
y	3	4	5	6	8

Done by: Kamelia Zaman Moon, Class Roll: 299
Date: 21/11/2018

```
*/
#include<stdio.h>
#include<math.h>
#define EPS 0.000001
int main()
{
    int i,n;
    float x[10],y[10],sumx,sumy,sumxx,sumxy,xmean,ymean,denom,a,b;
    printf("Input number of data points,n\n");
    scanf("%d",&n);
    printf("Input x and y values\n One set on each line\n");
    for(i=1; i<=n; i++)
        scanf("%f %f",&x[i],&y[i]);
    sumx=0.0;
    sumy=0.0;
    sumxx=0.0;
    sumxy=0.0;
    for(i=1; i<=n; i++)
    {
        sumx=sumx+x[i];
        sumy=sumy+y[i];
        sumxx=sumxx+x[i]*x[i];
        sumxy=sumxy+x[i]*y[i];
    }
    xmean=sumx/n;
    ymean=sumy/n;
    denom=n*sumxx-sumx*sumx;
    if(fabs(denom)>EPS)
    {
```

```
        b=(n*sumxy-sumx*sumy)/denom;
        a=ymean-b*xmean;
        printf("Linear Regression Line y=a+bx\n The coefficients are:\n a=%f\n b=%f\n",a,b);
    }
    else
        printf("No solution\n");
    return 0;
}
```

Output:

```
Input number of data points,n
5
Input x and y values
 One set on each line
1 3
2 4
3 5
4 6
5 8
Linear Regression Line y=a+bx
 The coefficients are:
 a=1.600000
 b=1.200000
```

"F:\Numerical Lab\C\Least Square.exe"

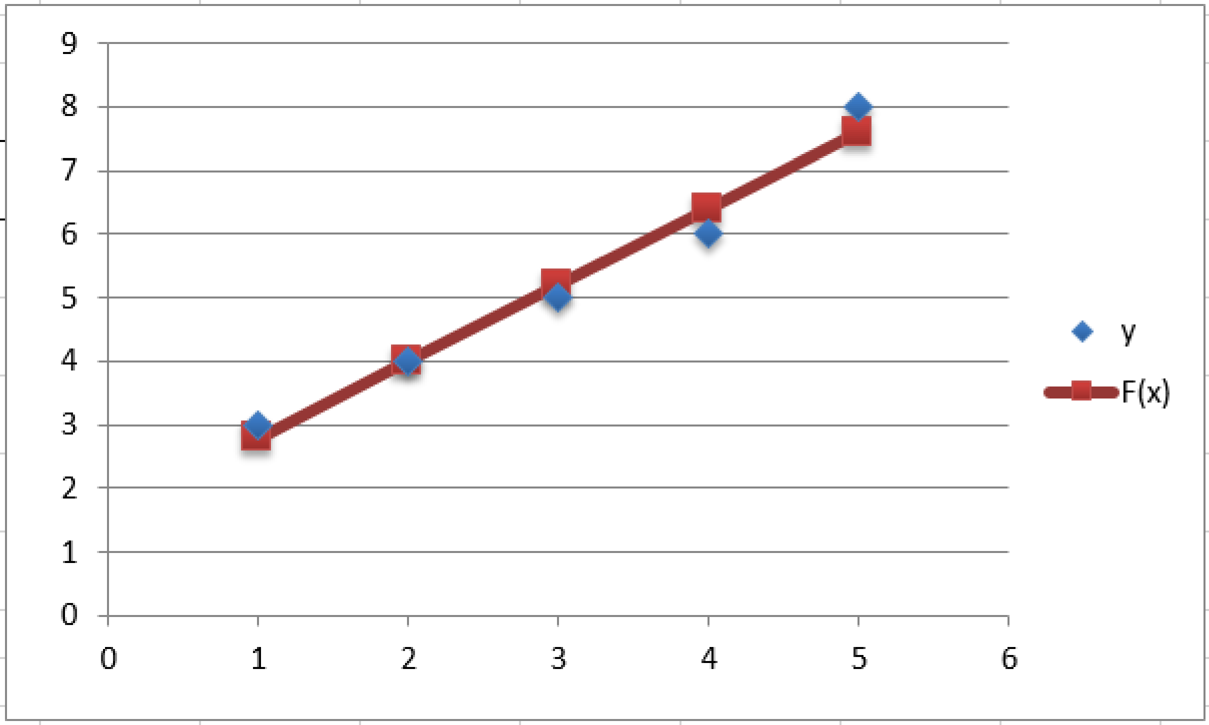
```
Input number of data points,n
5
Input x and y values
 One set on each line
1 3
2 4
3 5
4 6
5 8
Linear Regression Line y=a+bx
 The coefficients are:
 a=1.600000
 b=1.200000

Process returned 0 (0x0)   execution time : 33.343 s
Press any key to continue.
```

Least Square Method in Microsoft Excel:

Experiment Name: To fit a straight line using Least Square Method:

y=a+bx

	x	y	x*x	x*y	n	a	b	f(x)
	1	3	1	3				2.8
	2	4	4	8				4
	3	5	9	15	5	1.6	1.2	5.2
	4	6	16	24				6.4
	5	8	25	40				7.6
sum	15	26	55	90				19.6



Result:

Here, a=1.60

b=1.20

Therefore, the linear equation is

y=1.6+1.2x

Discussion:

The value of a and b are obtained from the calculation. Putting these values in the general equation $y = a + bx$ we get the required equation of straight line. From this equation we can plot a straight line on the graph.