

**Jahangirnagar University**  
**Department of Computer Science and Engineering**

LAB REPORT  
ON  
CSE-206 (NUMERICAL METHODS LAB)

EXPERIMENT No.: 02

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



<b>SUBMITTED BY</b> <b>Sumaita Binte Shorif</b> <b>Class Roll: 357</b> <b>Exam Roll: 191338</b> <b>Submission Date:</b> <b>Sept 06, 2021</b>			<b>SUBMITTED TO</b> <b>Dr. Md. Golam Moazzam</b> <b>Professor</b> <b>Dept. of Computer Science &amp; Engineering</b> <b>Jahangirnagar University</b>
---	--	--	--

## Experiment No: 02

**Name of the Experiment:** Determining the root of a non-linear equation using False Position Method.

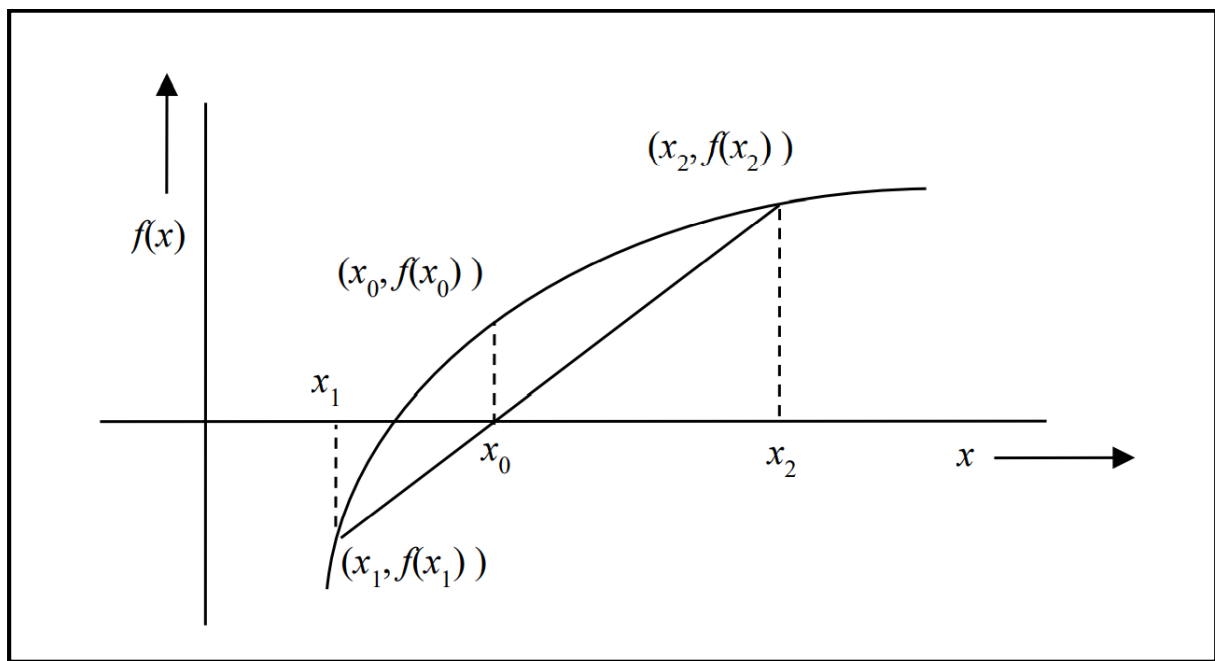
### Objectives:

- ✚ Understanding the process of finding the root of non-linear equation using False Position Method.
- ✚ Executing the implementation of False Position Method.
- ✚ To achieve the accurate root and expected output of a given nonlinear function.
- ✚ To be able to interpret the advantages and disadvantages of the false position method.

### Theory:

The False position method is used to find the roots of a polynomial equation. It separates the interval and subdivides the interval in which the root of the equation lies.

The basic principle of false position method is illustrated in the following figure.



**Fig 01:** Graphical Depiction of False Position Method.

Let us assume that the root lies between  $x_1$  and  $x_2$ .

Let us join the points  $x_1$  and  $x_2$  by a straight line. The point of intersection of this line with the x-axis ( $x_0$ ) gives an improved estimation 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  $f(x_0)$ . This process is repeated with the new values of  $x_1$  and  $x_2$ .

Since this method uses the false position of the root repeatedly, it is called *the 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

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{0 - f(x_1)}{x_0 - x_1}$$

Or,

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

Therefore,

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

This equation is known as the **false position formula**.

### Algorithm:

#### ***False Position Method:***

1. Start
2. Read values of  $x_1$ ,  $x_2$  and  $e$

/\*Here  $x_1$  and  $x_2$  are the two initial guesses  $e$  is the degree of accuracy or the absolute error i.e. the stopping criteria\*/

3. Compute function values  $f(x_1)$  and  $f(x_2)$
4. Check whether the product of  $f(x_1)$  and  $f(x_2)$  is negative or not. If it is positive take another initial guesses. If it is a negative value then go to step 5.
5. Determine:

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

6. Check whether the product of  $f(x_1)$  and  $f(x_0)$  is negative or not. If it is negative, then assign  $x_2 = x_0$ ,  $f(x_2) = f(x_0)$ ; If it is positive, assign  $x_1 = x_0$ ,  $f(x_1) = f(x_0)$ ;
7. Check whether the value of  $f(x_0)$  is greater than  $e$  or not.  
If yes, goto step 5.  
If no, goto step 8.

/\* Here the value  $e=0.000001$  is the desired degree of accuracy, and hence the stopping criteria. \*/

8. Display the root as  $x$ .

## Coding in C:

```
#include<stdio.h>
#include<math.h>
#define EPS 0.000001
#define F(x) (x)*(x) - (x) - 2
int cnt;
float root;

int fal(float a, float b)
{
    float x1, x2, x0, f0, f1, f2;
    x1 = a;
    x2 = b;
    f1 = F(x1);
    f2 = F(x2);
    if(f1 * f2 > 0)
    {
        return 0;
    }
    cnt = 1;
    while(1)
    {
        x0 = x1 - (f1 * (x2 - x1) / (f2 - f1));
        f0 = F(x0);
        if(f1 * f0 < 0)
        {
            x2 = x0;
            f2 = f0;
        }
        else
        {
            x1 = x0;
            f1 = f0;
        }

        if(fabs(f0) < EPS)
        {
            break;
        }
        else
        {
            cnt++;
        }
    }
    root=x0;
    return 1;
}


int main()
{
    int s;
    float a, b;
    printf("\nF(x) = (x)*(x) - (x) - 2\n\n");
    printf("SOLUTION BY FALSE POSITION METHOD\n\n");
    printf("Input starting with values : ");
    scanf("%f %f",&a,&b);
    printf("\n");
    s = fal(a, b);
    if(s == 0)
    {
```

```

        printf("\nStarting points do not bracket any root\n\n");
    }
    else
    {
        printf("\nRoot = %.6f\n",root);
        printf("F(root) = %.6f\n",F(root));
        printf("\nNO. OF ITERATIONS = %d\n",cnt - 1);
    }
    return 0;
}

```

### Output:

 "C:\Users\ASUS\Desktop\FALSE pos.exe"

$F(x) = (x)*(x) - (x) - 2$

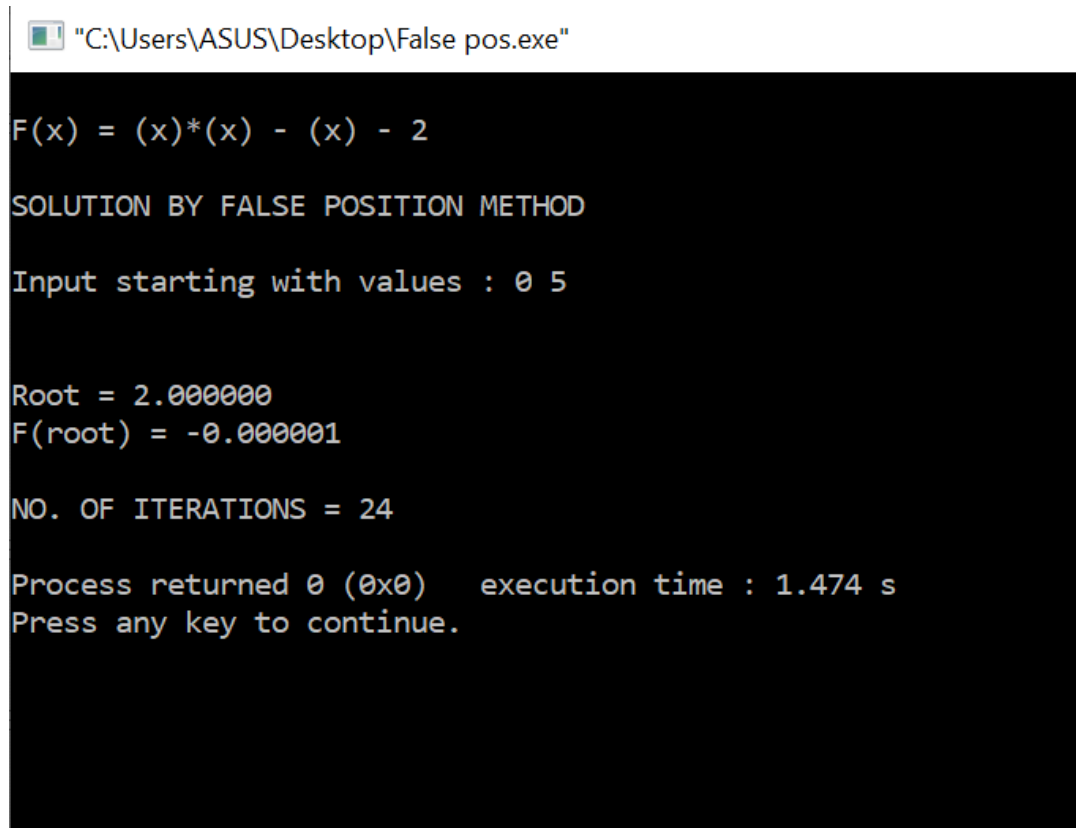
SOLUTION BY FALSE POSITION METHOD

Input starting with values : 8 10

Starting points do not bracket any root

Process returned 0 (0x0) execution time : 12.998 s  
Press any key to continue.

**Fig 02:** Output obtained when the starting points do not bracket any root.



```
"C:\Users\ASUS\Desktop\FALSE pos.exe"

F(x) = (x)*(x) - (x) - 2

SOLUTION BY FALSE POSITION METHOD

Input starting with values : 0 5

Root = 2.000000
F(root) = -0.000001

NO. OF ITERATIONS = 24

Process returned 0 (0x0)   execution time : 1.474 s
Press any key to continue.
```

**Fig 03:** Output Obtained.

### Discussion:

False Position method is linearly convergent as the initial guesses bracket the root. As the convergence rate is very slow, it takes many iterations to find a root. That is, as it is a trial and error method, in some cases it may take large time span to calculate the correct root and thereby slowing down the process. However, it is guaranteed to find root of a nonlinear equation by false position method.

-----