Haven's Light is Our Guide

## Rajshahi University of Engineering & Technology

Department of Computer Science & Engineering

# Lab Report

| | |
|---|---|
| **Course Code:** | CSE 2204 |
| **Course Title:** | Numerical Methods Sessional |
| **Experiment No:** | 01 |
| **Experiment Name:** | Solving Algebric and Transcendental Equations using Bisection and False Position Method and Comparison of their Efficency |

## Date:

December 15, 2023

| | |
|---|---|
| Submitted By: | Submitted To: |
| Name: Md. Abdullah Al Mamun | Shyla Afroge |
| Section: A | Assistant Professor |
| Roll No: 2003028 | Computer Science & Engineering |
| Year: 2nd Year Odd Semester | Rajshahi University of Engineering & Technology |

# Experiment No: 01

## Experiment Name: **Solving Algebric and Transcendental Equations using Bisection and False Position Method and Comparison of their Efficency**

## Theory:

### Bisection Method:

The bisection method is a numerical approach employed for finding the root of a real-valued function within a specified interval. The method begins by selecting an initial interval $[a, b]$ such that the function has opposite signs at the endpoints ($f(a) \cdot f(b) < 0$), adhering to the Intermediate Value Theorem. The process iteratively narrows down the interval by evaluating the function at the midpoint $c = \frac{a+b}{2}$. If $f(c) = 0$, $c$ is the root; otherwise, the interval containing the root is determined based on the signs of $f(a)$ and $f(c)$. This iterative process continues until the interval becomes sufficiently small or a predetermined number of iterations is reached, ensuring convergence. While the bisection method is straightforward and guaranteed to converge, it may converge slowly for certain functions due to its halving of the interval at each step. Nevertheless, its reliability and simplicity make it a widely used numerical technique in various scientific and engineering applications.

### Algorithm:

1. Interval Selection: Choose an initial interval $[a, b]$ such that the function has opposite signs at the endpoints ($f(a) \cdot f(b) < 0$), adhering to the Intermediate Value Theorem.

2. Iterative Process: Divide the interval in half and evaluate the function at the midpoint $c = \frac{a+b}{2}$. If $f(c) = 0$, $c$ is the root. Otherwise, determine the subinterval $[a, c]$ or $[c, b]$ in which the root must lie based on the signs of $f(a)$ and $f(c)$ (or $f(b)$ and $f(c)$).

3. Convergence Criteria: Repeat the process iteratively until the interval becomes sufficiently small or until a predetermined number of iterations is reached. The Bisection method converges to the root because it progressively narrows down the interval containing the root.

4. Accuracy Improvement: The accuracy of the root approximation improves with each iteration, and the method typically converges linearly. The precision can be controlled by specifying a tolerance level or a maximum number of iterations.

### False Position Method:

The False Position method, also known as the Regula Falsi method, is a numerical technique used for approximating the root of a real-valued function within a specified interval. Similar to the bisection method, it relies on the Intermediate Value Theorem, but instead of bisecting the interval, it employs linear interpolation between function values. The method iteratively updates the interval based on the function values at the endpoints, aiming to reduce the width of the interval containing the root. The False Position method can converge faster than the bisection method, but it may encounter convergence issues if the initial interval is poorly chosen or if the function exhibits erratic behavior. Despite these considerations, the method remains a valuable tool for numerical root-finding, particularly when a faster convergence rate is desired.

## Algorithm:

1. Interval Selection: Choose an initial interval $[a, b]$ such that the function has opposite signs at the endpoints ($f(a) \cdot f(b) < 0$), adhering to the Intermediate Value Theorem.

2. Linear Interpolation: Interpolate linearly between function values at the endpoints to find the point $c$ where the line intersects the x-axis. Evaluate $f(c)$.

3. Interval Update: Determine the subinterval $[a, c]$ or $[c, b]$ in which the root must lie based on the signs of $f(a)$ and $f(c)$ (or $f(b)$ and $f(c)$).

4. Convergence Criteria: Repeat the process iteratively until the interval becomes sufficiently small or until a predetermined number of iterations is reached. The False Position method aims to reduce the width of the interval containing the root.

5. Accuracy Improvement: The accuracy of the root approximation improves with each iteration, and the method typically converges faster than the Bisection method. However, it may encounter convergence issues if the initial interval is poorly chosen or if the function exhibits erratic behavior.

## Program:

Listing 1: Bisection Method

```cpp
    double rootByBisection(double a, double b, bool print = false)
    {
        int i = 0;
        double error = abs(a - b), c;

        while (abs(error) > 0.0001)
        {
            if (print)
                cout << i + 1 << "\t|\t" << a << "\t|\t" << b << "\t|\t" <<
                    c << "\t\t|\t" << function(c) << "\t\t|\t" << error <<
                    "\n";

            c = (a + b) / 2;
            if (function(a) * function(c) < 0)
                b = c;
            else
                a = c;

            error = abs(a - b);

            if (i > MAX_ITERATION)
                break;
            i++;
        }
        return c;
    }
```

Listing 2: False Position Method

```cpp
    double rootByFalsePosition(double a, double b, bool print = false)
    {
        int i = 0;
        double c;

        while (abs(function(c)) > 0.0001)
        {
```

```
        if (print)
            cout << i + 1 << "\t|\t" << a << "\t|\t" << b << "\t|\t" <<
                c << "\t\t|\t" << function(c) << "\t\t|\t" <<
                function(c) * 100 << "%\n";

        c = b - function(b) * (b - a) / (function(b) - function(a));

        if (function(a) * function(c) < 0)
            b = c;
        else
            a = c;

        if (i > MAX_ITERATION)
            break;
        i++;
    }
    return c;
}
```

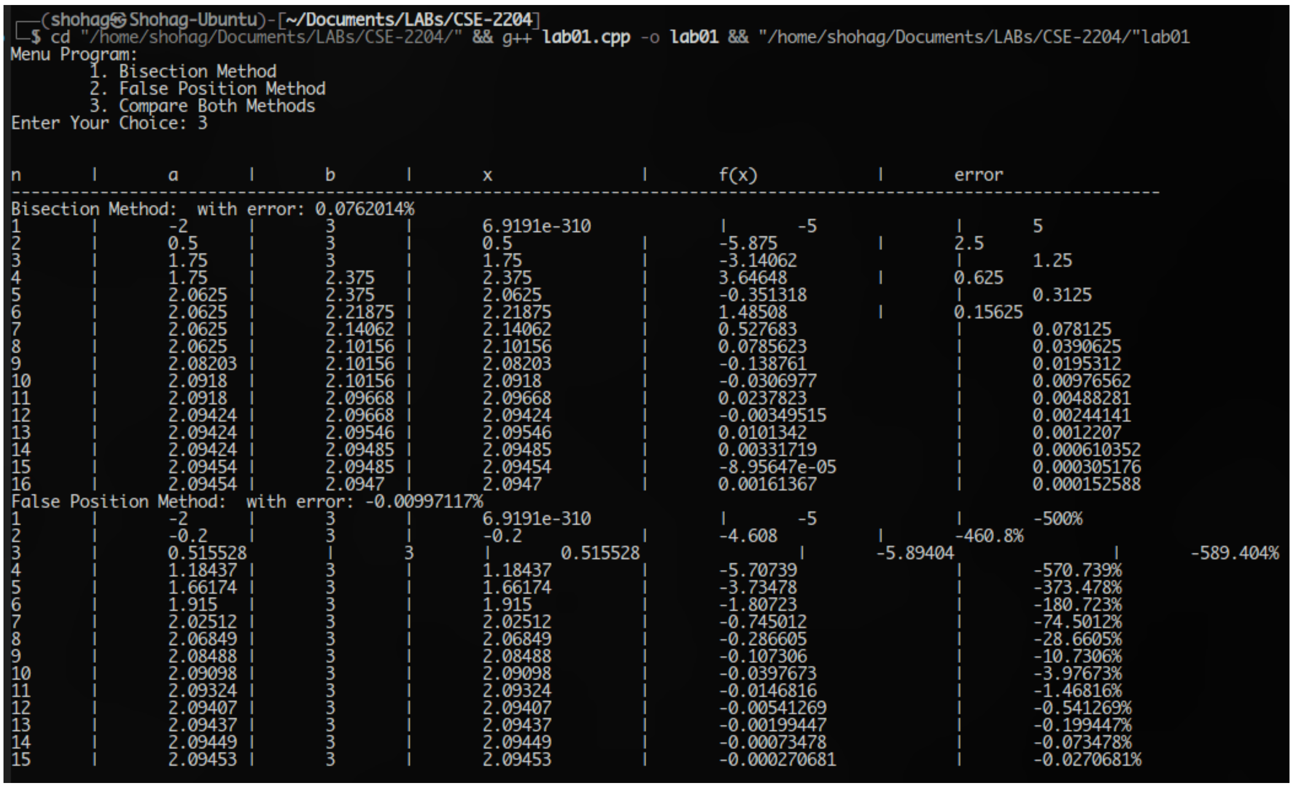
Listing 3: Main Program

```
#include <iostream>
#define MAX_ITERATION 100
using namespace std;

double function(double x)
{
    return x * x - 2 * x - 5;
}

int main()
{
    int choice;
    double a = -1, b = 1;

    while (function(a) * function(b) > 0)
    {
        function(b) > function(a) ? b++ : a--;
    }

    // cout << "a: " << a << "\tb: " << b << "\n";

    cout << "Menu␣Program:␣\n";
    cout << "\t1.␣Bisection␣Method\n";
    cout << "\t2.␣False␣Position␣Method\n";
    cout << "\t3.␣Compare␣Both␣Methods\n";
    cout << "Enter␣Your␣Choice:␣";
    cin >> choice;

    cout << "\n\nn\t|\ta\t|\tb\t|\tx\t\t|\tf(x)\t\t|\terror\n";
    cout <<
       "---------------------------------------------------------\n";

    switch (choice)
    {
    case 1:
        rootByBisection(a, b, true);
        break;
    case 2:
        rootByFalsePosition(a, b, true);
        break;
    case 3:
        cout << "Bisection␣Method:␣" << rootByBisection(a, b) << "␣with␣
            error:␣" << function(rootByBisection(a, b)) * 100 << "%\n";
        cout << "False␣Position␣Method:␣" << rootByFalsePosition(a, b)
            << "␣with␣error:␣" << function(rootByFalsePosition(a, b)) *
```

```
                  100 << "%\n";
              break;
          }
      }
```

## Result:

The Bisection and False Position methods are numerical techniques employed for finding the roots of a real-valued function. In the Bisection method, the interval containing the root is successively halved until a sufficiently accurate root approximation is achieved. While it guarantees convergence, it may be slow for certain functions. On the other hand, the False Position method, also known as the Regula Falsi method, utilizes linear interpolation between function values to approximate the root. Although it typically converges faster than Bisection, it can encounter convergence issues if the initial interval is poorly chosen or if the function exhibits erratic behavior. Both methods have their strengths and limitations, with the choice between them often dependent on the specific characteristics of the function being analyzed.



```
┌──(shohag㉿Shohag-Ubuntu)-[~/Documents/LABs/CSE-2204]
└─$ cd "/home/shohag/Documents/LABs/CSE-2204/" && g++ lab01.cpp -o lab01 && "/home/shohag/Documents/LABs/CSE-2204/"lab01
Menu Program:
        1. Bisection Method
        2. False Position Method
        3. Compare Both Methods
Enter Your Choice: 3

n       |       a       |       b       |       x               |       f(x)            |       error
-------------------------------------------------------------------------------------------------------------------------------------
Bisection Method:  with error: 0.0762014%
1       |       -2      |       3       |       6.9191e-310             |       -5              |       5
2       |       0.5     |       3       |       0.5             |       -5.875          |       2.5
3       |       1.75    |       3       |       1.75            |       -3.14062                |       1.25
4       |       1.75    |       2.375   |       2.375           |       3.64648         |       0.625
5       |       2.0625  |       2.375   |       2.0625          |       -0.351318               |       0.3125
6       |       2.0625  |       2.21875 |       2.21875         |       1.48508         |       0.15625
7       |       2.0625  |       2.14062 |       2.14062         |       0.527683                |       0.078125
8       |       2.0625  |       2.10156 |       2.10156         |       0.0785623               |       0.0390625
9       |       2.08203 |       2.10156 |       2.08203         |       -0.138761               |       0.0195312
10      |       2.0918  |       2.10156 |       2.0918          |       -0.0306977              |       0.00976562
11      |       2.0918  |       2.09668 |       2.09668         |       0.0237823               |       0.00488281
12      |       2.09424 |       2.09668 |       2.09424         |       -0.00349515             |       0.00244141
13      |       2.09424 |       2.09546 |       2.09546         |       0.0101342               |       0.0012207
14      |       2.09424 |       2.09485 |       2.09485         |       0.00331719              |       0.000610352
15      |       2.09454 |       2.09485 |       2.09454         |       -8.95647e-05            |       0.000305176
16      |       2.09454 |       2.0947  |       2.0947          |       0.00161367              |       0.000152588
False Position Method:  with error: -0.00997117%
1       |       -2      |       3       |       6.9191e-310             |       -5              |       -500%
2       |       -0.2    |       3       |       -0.2            |       -4.608          |       -460.8%
3       |       0.515528        |       3       |       0.515528                |       -5.89404                |       -589.404%
4       |       1.18437 |       3       |       1.18437         |       -5.70739                |       -570.739%
5       |       1.66174 |       3       |       1.66174         |       -3.73478                |       -373.478%
6       |       1.915   |       3       |       1.915           |       -1.80723                |       -180.723%
7       |       2.02512 |       3       |       2.02512         |       -0.745012               |       -74.5012%
8       |       2.06849 |       3       |       2.06849         |       -0.286605               |       -28.6605%
9       |       2.08488 |       3       |       2.08488         |       -0.107306               |       -10.7306%
10      |       2.09098 |       3       |       2.09098         |       -0.0397673              |       -3.97673%
11      |       2.09324 |       3       |       2.09324         |       -0.0146816              |       -1.46816%
12      |       2.09407 |       3       |       2.09407         |       -0.00541269             |       -0.541269%
13      |       2.09437 |       3       |       2.09437         |       -0.00199447             |       -0.199447%
14      |       2.09449 |       3       |       2.09449         |       -0.00073478             |       -0.073478%
15      |       2.09453 |       3       |       2.09453         |       -0.000270681            |       -0.0270681%
```

Figure 1: Output of the Program