Khulna University of Engineering & Technology, Khulna

Department of Biomedical Engineering

SESSIONAL REPORT

Course No: **BME 2152**

Experiment No : **05**

Name of the Experiment: **Finding the root of polynomial equation using Bisection Method and False Position Method.**

Remarks:

Name: **Md. Fuad Hasan Hamim**

Roll : **2215029**

Year : **2nd**

Term : **1st**

Date of performance : 8 January 2025

Date of submission : 15 January 2025

Objectives:

The main objectives of this experiment are:

- To learn how to find the root of polynomial equation using bisection method in MATLAB.
- To learn how to implement Bisection method and false position method in MATLAB.
- To get familiar with iterative method in MATLAB.

Introduction:

A **polynomial equation** is an equation formed with variables, exponents and coefficients. The highest exponent is the order of the equation. To solve a polynomial equation, we need to take it in standard form (variables and constants on one side and zero on the other side of the equation). Factor it and set each factor to zero. Solve each factor. The solutions are the solutions of the polynomial equation. Example: $2x^5 + x^4 – x – 1=0$, $x^3 - 2x=0$.

Polynomials in MATLAB are represented as row of a vector containing coefficients ordered by descending powers.

For example, the equation $G(x) = 2x^4 + 3x^3 – 4x + 1$ could be represented as g = [2 3 -4 1].

For evaluating Polynomials we use function polyval( ), It evaluates the polynomial g at each point in x.

Similarly, the equation $P(x) = x^4 + 7x^3 + 9$ could be represented as −

p = [1 7 0 0 9];

polyval is a built-in function in MATLAB that allows you to evaluate a polynomial at a specific point. It evaluates the polynomial let's p at the points in x and it returns the corresponding function values in y . The syntax for polyval is as follows:

y = polyval(p,x)

input() is a built-in function in MATLAB to take user defined input in the program. Example:

x= input('Write whatever you want to take as input')

Bisection method:

Bisection Method is one of the simplest, reliable, easy to implement and convergence guarenteed method for finding real root of non-linear equations. It is also known as Binary Search or Half Interval or Chopping Method.

Let us consider a continuous function 'f' which is defined on the closed interval [a, b], is given with f(a) and f(b) of different signs. Then by the intermediate theorem, there exists a point x belonging to (a, b) for which f(x) = 0.

This is a statement of the Intermediate Value Theorem (IVT) for continuous functions. It confirms that if a function is continuous on a closed interval [a,b], and the function's values at the endpoints [a] and [b] have opposite signs, then there exists at least one value c∈(a,b)c / in (a, b)c∈(a,b) such that f(c)=0.

Here, $x_0 = \frac{x_1+x_2}{2}$

Now, there exists the following three conditions:

1. If f(x0) = 0, we have a root at x0.
2. If f(x0)f(x1) <0 there is a root between x0 and x1.
3. If f(x0)f(x2) <0 there is a root between x0 and x2.

Algorithm for Bisection Method:

Step 1: Input Equation

Here, we need to input the equation for which we need to find the roots.

Step 2: Input guess value 1 (x_1) and guess value 2 (x_2)

Step 3: Find $f(x_1)$ and $f(x_2)$

Compute the functional values of two guess value x_1 and x_2

Step 4: If $f(x_1)*f(x_2) > 0$; print: error in finding root

Step 5: If $f(x_1)*f(x_2) <$ eps; print: the root is x_2 and terminate the loop

Step 6: $x_0 = (x_1+x_2)/2$

Here x_0 is midpoint between x_1 & x_2

Step 7: Find $f(x_0)$

Step 8: If $f(x_0) = 0$; print: the root is x0 and terminate the loop

Step 9: If $|(x_2-x_1)/x_2| <$ eps; print: the root is x2 and terminate the loop

Step 10: If $f(x_1)*f(x_0)<0$; $x_2 = x_0$ and $f(x_2)=f(x_0)$

Else $x_1 = x_0$ and $f(x_1) = f(x_0)$

Step 11: Go to step 6 and repeat.

False Position Method:

The False Position Method (also known as the Regula Falsi Method) is a root-finding algorithm that combines features of the bisection method and linear interpolation. It is used to find the root of a function f(x)=0 within a specified interval [a,b] where the function values at the endpoints have opposite signs (f(a)·f(b)<0).

False Position formula :

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

X0 is obtained by applying a correction to x1.

Algorithm for False Position Method:

Step 1: Input Equation

Here, we need to input the equation for which we need to find the roots.

Step 2: Input guess value 1 (x_1) and guess value 2 (x_2)

Step 3: Find $f(x_1)$ and $f(x_2)$

Compute the functional values of two guess value x_1 and x_2

Step 4: If $f(x_1)*f(x_2) > 0$; print: error in finding root

Step 5: If $f(x_1)*f(x_2) < eps$; print: the root is x_2 and terminate the loop

Step 6: $x_0 = x_1 - \frac{f(x_1)(x_2-x_1)}{f(x_2)-f(x_1)}$

Step 7: Find $f(x_0)$

Step 8: If $f(x_0) = 0$; print: the root is x_0 and terminate the loop

Step 9: If $|(x_2-x_1)/x_2| < eps$; print: the root is x_2 and terminate the loop

Step 10: If $f(x_1)*f(x_0)<0$; $x_2 = x_0$ and $f(x_2)=f(x_0)$

Else $x_1 = x_0$ and $f(x_1) = f(x_0)$

Step 11: Go to step 6 and repeat.

Task 1 || Find the roots of the polynomial equation x2-4x-10=0 by using Bisection method.

Code:

```
clc;
clear all;
eps=0.000001;
a=input('Coefficients of equation : ');
x1=input('Guess value1 :');
x2=input('Guess value2 :');
f1=polyval(a,x1)
f2=polyval(a,x2)

if (f1*f2>0)
   fprintf('error in finding roots\n');
else
   for n=1:50;
      x0=(x1+x2)/2;
      f0=polyval(a,x0);
      if f0==0
         fprintf('the root is %g',x0);
         return;
      end

      if (abs((x2-x1)/x2)<eps)
         root=(x1+x2)/2;
         fprintf('the root is %g',root);
         return;
      end

      if (f1*f0<0)
         x2=x0;
         f2=f0;
      else
         x1=x0;
         f1=f0;
      end

   end
end
```

Output:

```
Coefficients of equation :
[1 -4 -10]
Guess value1 :
-2
Guess value2 :
-1
```

```
f1 =

     2

f2 =

    -5

the root is -1.74166
>>
```

Task 2 || Find the roots of the polynomial equation x2-4x-10=0 by using False Position method.

Code

```
%False Position Method %
clc;
clear all;
eps=0.000001;
a=input('Coefficients of equation : ');
x1=input('Guess value1 :');
x2=input('Guess value2 :');
f1=polyval(a,x1)
f2=polyval(a,x2)

if (f1*f2>0)
   fprintf('error in finding roots\n');
else
   for n=1:50;
      x0=x1 - (f1*(x2-x1)/(f2-f1));
      f0=polyval(a,x0);
      if f0==0
         fprintf('the root is %g',x0);
         return;
      end

      if (abs((x2-x1)/x2)<eps)
         root=(x1 - f1*(x2-x1)/(f2-f1));
         fprintf('the root is %g',root);
         return;
      end
```

```
        if (f1*f0<0)
            x2=x0;
            f2=f0;
        else
            x1=x0;
            f1=f0;
        end

    end
end
```

Output

```
Coefficients of equation :
[1 -4 -10]
Guess value1 :
-2
Guess value2 :
-1

f1 =

     2


f2 =

    -5

the root is -1.74166
>>
3

f1 =

    -2


f2 =

     4

the root is 2
>>
```

Conclusion:

In this experiment, we used MATLAB to find the roots of a polynomial equation using the Bisection Method and the False Position Method. Both methods work by narrowing down the interval where the root lies and are easy to implement. The Bisection Method is simple and always finds a root if the function changes sign, but it can be slow. The False Position Method is usually faster because it uses a better estimate based on the function values, but it may get stuck if one endpoint changes very little. This experiment helped us understand how to apply these methods and use MATLAB for solving equations step by step.

References:

[1] Symbolab, "Polynomial Equation Calculator," available at: https://www.symbolab.com/solver/polynomial-equation-calculator.

[2] GeeksforGeeks, "Polynomials in MATLAB," available at: https://www.geeksforgeeks.org/polynomials-in-matlab/.

[3] TutorialsPoint, "MATLAB Polynomials," available at: https://www.tutorialspoint.com/matlab/matlab_polynomials.htm.

[4] GeeksforGeeks, "polyval in MATLAB," available at: https://www.geeksforgeeks.org/polyval-in-matlab/.

[5] CodeSansar, "Bisection Method Algorithm," available at: https://www.codesansar.com/numerical-methods/bisection-method-algorithm.htm.