**LAB EXPERIMENT # 2:** Root finding using Bisection method and False position method

| | |
|---|---|
| **Name & Roll No.:** Ashraf Al- Khalique, 1801171 | **Grade (20)** |
| **Registration Number:** 351 | |
| **Lab Section:** C-2 | |

## 2.1 Objectives

- To determine roots of an equation in a single variable using ‘Bisection Method’ & ‘False Position method.
- To understand the MATLAB implementation of the ‘Bisection method’ & ‘False Position method’.
- To analyze of results using different initial values and different ranges of error.

## 2.2 Theory

### 2.2.1 Bisection method

Bisection method is one of the many processes to find root of nonlinear equations. It belongs in the bracketing method system since it needs two assumptions on both side of the curve. If the
assumptions are a and b then f(a)*f(b) should be less than zero. Which means sign for both the
multiplicated terms are opposite. This method guarantees to converge.

### 2.2.2 False Position method

False position method is also one of the bracketing methods processes to determine the root of
nonlinear equations. This process is steady and faster than bisection method. Though it needs a
long time to converge like bisection method but it is steady and converges linearly. This process
is also really easy to understand and doesn’t need any complex mathematical calculations.

## 2.3 Apparatus

**2.3.1** MATLAB

## 2.4 Bisection method

### 2.4.1 Algorithm

**<u>Step: 1</u>** Lower ‘xl’ and upper ‘xu’ values are guessed for the root in such a way that ‘f(xl)*f(xu) < 0’

**Step: 2** The approximation of the root is determined by 'xm=(xl+xu)/2'

**Step: 3** The following evaluations are made to determine in which subinterval the root lies between

- If 'f(xl)*f(xu) < 0', the root lies in the lower subinterval (between xl & xm). Therefore, set 'xu = xm' and return to step 2.
- If 'f(xl)*f(xu) > 0', the root lies in the upper subinterval (between xm & xu). Therefore, set xl = xm and return to step 2.
- If 'f(xl)*f(xm) = 0', the root equals xm, terminate the computation.

### 2.4.2 Tolerable Error

Defining the approximation error as

**% Error = (| Xnew - Xold | / Xnew) * 100**

An acceptance level of error says, (E) is input by the user. Then the program is terminated when,

**% Error <= E**

### 2.4.3 Pseudocode

```
Start
Define function f= @x
Input
     xl=input ('Lower value xl= ');
xu=input ('Upper value xu= ');
Emax=input ('Error= ');
Imax=input ('Imax= ');
i=1;

xm=(xl+xu)/2;
E=abs((xu-xm))/xm*100;

if (f(xl)*f(xu)) <0
    while i<Imax
        xold=xm;
        if f(xl)*f(xm)<0
            xu=xm;
        else
            xl=xm;
        end
        xm=(xl+xu)/2;
        E=abs((xm-xold))/xm*100;
        i=i+1;
        if E<=Emax
            break;
        end
```

```
    end

end
Stop
```

### 2.4.4 MATLAB Code

```
clc;
clear all;
close all;
%%
f=@(x) x^3-6*x*x+11*x-1;
%%
xl=input('Lower value xl= ');
xu=input('Upper value xu= ');
Emax=input('Error= ');
Imax=input('Imax= ');
%%
i=1;
xm=(xl+xu)/2;
E = abs((xu-xm))/xm*100;
%%
variables={'Iter','xl','xu','xm','f(xl)','f(xu)','f(xm)','Error'};
M=[i xl xu xm f(xl) f(xu) f(xm) E];
%%
if (f(xl)*f(xu))<0
    while i<Imax
        xold=xm;
        if f(xl)*f(xm)<0
            xu=xm;
        else
            xl=xm;
        end
        xm=(xl+xu)/2;
        E=abs((xm-xold))/xm*100;
```

```
        i=i+1;
        M=[M;i xl xu xm f(xl) f(xu) f(xm) E];
        if E<=Emax
            break;
        end
    end
end
%%
Result=array2table(M);
Result.Properties.VariableNames(1:size(M,2))= variables
fprintf('\n\nroot is: %f\n',xm);
```

### 2.4.5 MATLAB Output

```
D: ▸ Engineering ▸ MathWorks MATLAB R2021a v9.10.0.1602886 - CrackzSoft ▸ bin ▸ win64 ▸
Current Folder
Command Window
Lower value xl= 0
Upper value xu= 0.25
Error= 0.02
Imax= 100

Result =

  14×8 table

 Iter     xl          xu          xm          f(xl)          f(xu)          f(xm)         Error
 ____    ________    ________    ________    ___________    __________    ___________    ________

   1            0        0.25       0.125             -1        1.3906         0.2832         100
   2            0       0.125      0.0625             -1        0.2832       -0.33569         100
   3       0.0625       0.125     0.09375       -0.33569        0.2832       -0.02066      33.333
   4      0.09375       0.125     0.10938       -0.02066        0.2832        0.13266      14.286
   5      0.09375     0.10938     0.10156       -0.02066       0.13266       0.056345      7.6923
   6      0.09375     0.10156    0.097656       -0.02066      0.056345        0.01793           4
   7      0.09375    0.097656    0.095703       -0.02066       0.01793     -0.0013436      2.0408
   8     0.095703    0.097656     0.09668     -0.0013436       0.01793      0.0082985      1.0101
   9     0.095703     0.09668    0.096191     -0.0013436     0.0082985      0.0034788     0.50761
  10     0.095703    0.096191    0.095947     -0.0013436     0.0034788      0.0010679     0.25445
  11     0.095703    0.095947    0.095825     -0.0013436     0.0010679    -0.00013775     0.12739
  12     0.095825    0.095947    0.095886    -0.00013775     0.0010679     0.00046511    0.063654
  13     0.095825    0.095886    0.095856    -0.00013775    0.00046511     0.00016369    0.031837
  14     0.095825    0.095856     0.09584    -0.00013775    0.00016369     1.2971e-05    0.015921


root is: 0.095840
fx >>
```

### 2.5 False Position method

#### 2.5.1 Algorithm

**Step: 1** Lower 'xl' and upper 'xu' values are guessed for the root in such a way that 'f(xl)*f(xu) < 0'

**Step: 2** The approximation of the root is determined by 'xm=xl-((xl-xu)) *f(xl)/(f(xl)-f(xu))'

**Step: 3** The following evaluations are made to determine in which subinterval the root lies between

- If 'f(xl)*f(xm) < 0', the root lies in the lower subinterval (between xl & xm). Therefore, set 'xu = xm' and return to step 2.
- If 'f(xm)*f(xu) > 0', the root lies in the upper subinterval (between xm & xu). Therefore, set xl = xm and return to step 2.
- If 'f(xl)*f(xm) = 0', the root equals xm, terminate the computation.

#### 2.5.2 Tolerable Error

Defining the approximation error as

**% Error = (| Xnew - Xold | / Xnew) * 100**

An acceptance level of error says, (E) is input by the user. Then the program is terminated when,

**% Error <= E**

#### 2.5.3 Pseudocode

```
Start
Define function f= @x
Input
xl=input ('Lower value xl= ');
xu=input ('Upper value xu= ');
Emax=input ('Error= ');
Imax=input ('Imax= ');
i=1;

xm=xl-((xl-xu)) *f(xl)/(f(xl)-f(xu));
E=abs((xu-xm))/xm*100;

if (f(xl)*f(xu)) <0
    while i<Imax
        xold=xm;
        if f(xl)*f(xm)<0
            xu=xm;
        else
            xl=xm;
        end
```

```
        xm=xl-((xl-xu)) *f(xl)/(f(xl)-f(xu));
        E=abs((xm-xold))/xm*100;
        i=i+1;
        if E<=Emax
            break;
        end
    end
end

Stop
```

### 2.5.4 MATLAB Code

```
clc;
clear all;
close all;
%%
f=@(x) x^3-6*x*x+11*x-1;
%%
xl=input('Lower value xl= ');
xu=input('Upper value xu= ');
Emax=input('Error= ');
Imax=input('Imax= ');
%%
i=1;
xm=xl-((xl-xu)) *f(xl)/(f(xl)-f(xu));
E = abs((xu-xm))/xm*100;
%%
variables={'Iter','xl','xu','xm','f(xl)','f(xu)','f(xm)','Erro
r'};
M=[i xl xu xm f(xl) f(xu) f(xm) E];
%%
if (f(xl)*f(xu))<0
    while i<Imax
        xold=xm;
        if f(xl)*f(xm)<0
            xu=xm;
        else
            xl=xm;
        end
        xm=xl-((xl-xu)) *f(xl)/(f(xl)-f(xu));
        E=abs((xm-xold))/xm*100;
        i=i+1;
        M=[M;i xl xu xm f(xl) f(xu) f(xm) E];
        if E<=Emax
            break;
        end
    end
end
%%
```

```
Result=array2table(M);
Result.Properties.VariableNames(1:size(M,2))= variables
fprintf('\n\nroot is: %f\n',xm);
```

### 2.5.5 MATLAB Output

```
D: ▸ Engineering ▸ MathWorks MATLAB R2021a v9.10.0.1602886 - CrackzSoft ▸ bin ▸ win64 ▸
Command Window
Current Folder

Lower value xl= 0
Upper value xu= 0.20
Error= 0.02
Imax= 100

Result =

  4×8 table

    Iter    xl       xu          xm        f(xl)       f(xu)          f(xm)        Error
    ____    __    ________    ________    _____    __________    __________    ________

     1      0          0.2     0.10163     -1           0.968      0.056969        96.8
     2      0      0.10163    0.096149     -1        0.056969     0.0030557      5.6969
     3      0     0.096149    0.095856     -1       0.0030557    0.00016307     0.30557
     4      0     0.095856     0.09584     -1      0.00016307       8.7e-06    0.016307


root is: 0.095840
fx >>
```

## 2.6 Discussion & Analysis

In this mathematical approach, when f(x) is a higher degree polynomial or an expression incorporating transcendental functions, algebraic approaches were insufficient to extract the roots from the solution. In that situation, the roots were discovered using approximation approaches, including the bisection method and the false position method.

There were multiple solutions to the equation. The solution would be determined by the initial interval chosen. For example, suppose the interval was set at [0, 0.25]. The root discovered in this example was 0.095840. If the beginning interval was higher, the number of iterations required to arrive at the identical answer would be greater as well.

Convergence in the Bisection and False Position methods is linear in this case. The only difference is that the Bisection method is more time consuming. While the fake position method is faster than the bisection method.