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

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+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 \leq 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 = (x1+xu)/2;
E=abs((xu-xm))/xm*100;
if (f(x1)*f(xu)) <0
    while i<Imax</pre>
        xold=xm;
        if f(x1)*f(xm)<0
             xu=xm;
        else
             x1=xm;
        end
        xm = (x1+xu)/2;
        E=abs((xm-xold))/xm*100;
        i=i+1;
        if E<=Emax
             break;
        end
```

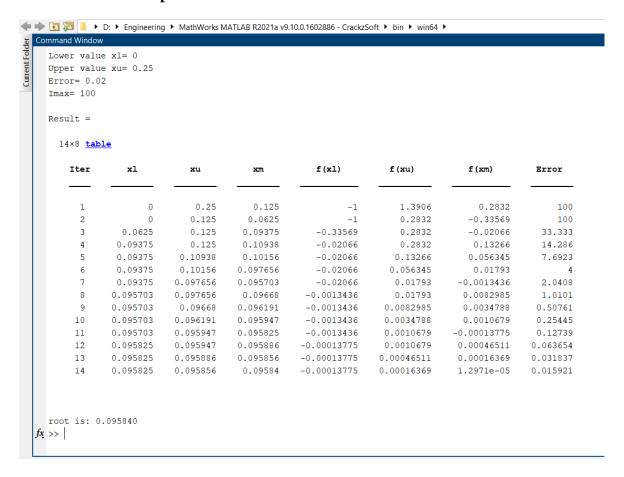
end

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=(x1+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(x1)*f(xu))<0
    while i<Imax
        xold=xm;
        if f(x1)*f(xm)<0</pre>
            xu=xm;
        else
            x1=xm;
        end
        xm=(x1+xu)/2;
        E=abs((xm-xold))/xm*100;
```

2.4.5 MATLAB Output



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))'

<u>Step: 3</u> 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 x = 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**

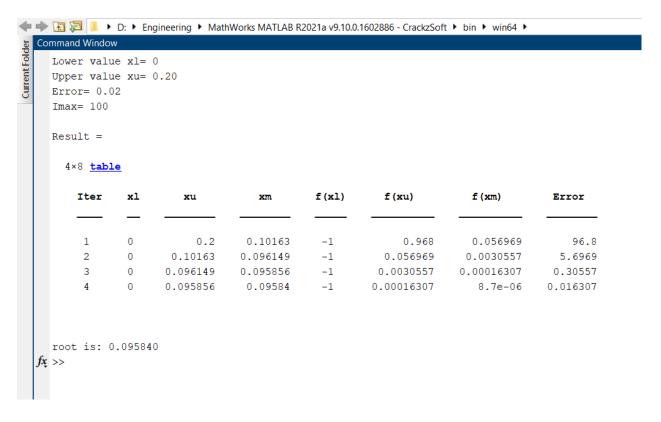
2.5.3 Pseudocode

```
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(x1)*f(xm)<0
            xu=xm;
        else
            x1=xm;
        end
```

```
xm=xl-((xl-xu)) *f(xl)/(f(xl)-f(xu));
        E=abs((xm-xold))/xm*100;
        i=i+1;
        if E<=Emax</pre>
            break;
        end
    end
end
Stop
2.5.4 MATLAB Code
clc;
clear all;
close all;
f=0(x) x^3-6*x*x+11*x-1;
응응
xl=input('Lower value xl= ');
xu=input('Upper value xu= ');
Emax=input('Error= ');
Imax=input('Imax= ');
응응
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
M=[i xl xu xm f(xl) f(xu) f(xm) E];
응응
if (f(x1)*f(xu))<0
    while i<Imax</pre>
        xold=xm;
        if f(x1) * f(xm) < 0
             xu=xm;
        else
             x1=xm;
        end
        xm=xl-((xl-xu)) *f(xl)/(f(xl)-f(xu));
        E=abs((xm-xold))/xm*100;
        M=[M;i xl xu xm f(xl) f(xu) f(xm) E];
        if E<=Emax</pre>
            break;
        end
    end
end
응응
```

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

2.5.5 MATLAB Output



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.