Heaven's Light is Our Guide

Rajshahi University of Engineering and Technology

Course Code

ECE 2214

Course Title

Numerical Methods and Discrete Mathematics Sessional

Experiment Date: November 11, 2023,
Submission Date: November 18, 2023

Lab Report 7, 8 & 9: Implementing root finding methods; Bisection, False Position & Newton-Raphson in MATLAB

Submitted to
Md. Omaer Faruq Goni
Lecturer
Dept of ECE, Ruet

Submitted by
Md. Tajim An Noor
Roll: 2010025

Finding root of nonlinear equation using Bisection Method.

Introduction

Bisection Method

Bisection method is based on the fact that if $f(x)$ is real and continuous function, and for two initial guesses a and b brackets the root such that: $f(a) \times f(b) < 0$ then there exists at least one root between a and b.

Root is obtained in Bisection method by successive halving the interval i.e. If a and b are two guesses then we compute new approximated root as:

$$c = \frac{(a+b)}{2}$$

Now we have following three different cases:

- If $f(c) = 0$ then the root is c.
- If $f(a) \times f(b) < 0$ then root lies between a and c.
- If $f(a) \times f(c) > 0$ then root lies between b and c.

And then the process is repeated until we find the root within desired accuracy.[1]

Tools Used

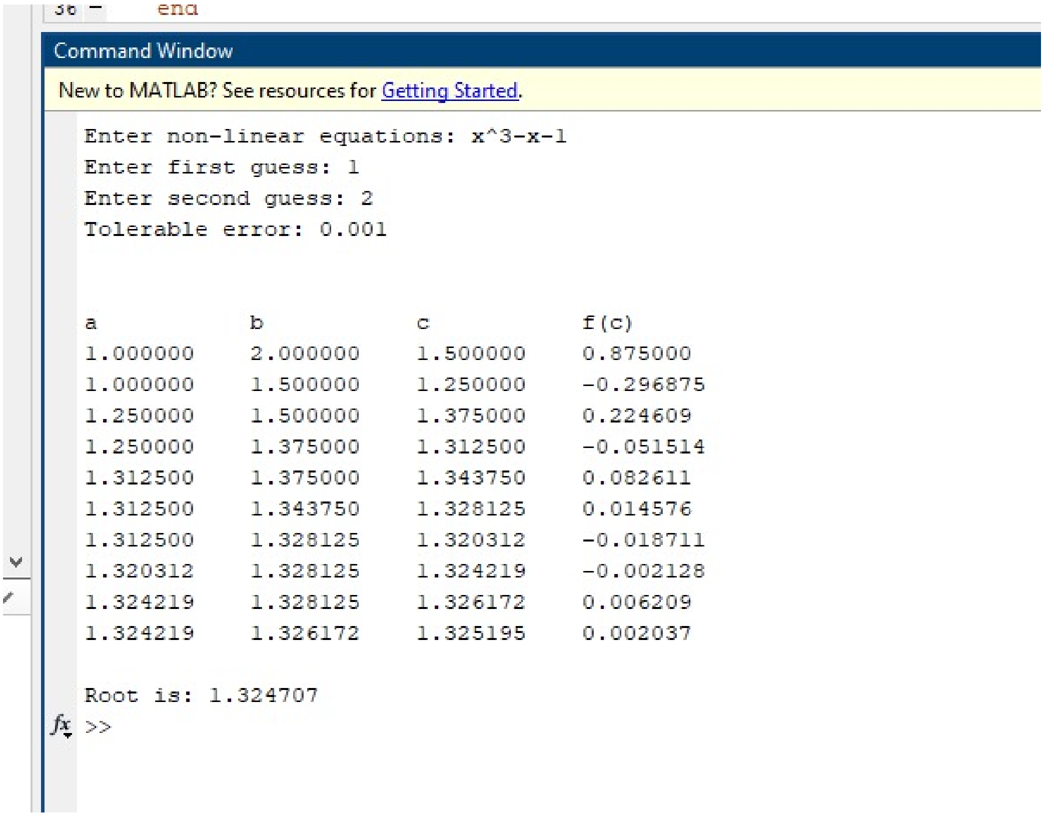
- MATLAB R2021a - for writing and running code.
- MacTeX -LaTeXcompiler.
- VS Code with LaTeXworkshop extension as a text editor.

Process

Code for Bisection:

```
% Clearing Screen
clc

% Setting x as symbolic variable, in every string, x will be considered as a variable
syms x;

% Input Section
eqn = input('Enter non-linear equations: '); %input as normal string.
a = input('Enter first guess: ');
b = input('Enter second guess: ');
e = input('Tolerable error: '); %error margin

% Finding Functional Value
fa = eval(subs(eqn,x,a));
fb = eval(subs(eqn,x,b));

% Implementing Bisection Method
if fa*fb > 0
    disp('Initial values does not create bracket around the root');
else
    c = (a+b)/2;
    fc = eval(subs(eqn,x,c));
    fprintf('\n\na\t\t\tb\t\t\tc\t\t\tf(c)\n');
    while abs(fc)>e
        fprintf('%f\t%f\t%f\t%f\n',a,b,c,fc);
        if fa*fc< 0
            b =c;
        else
            a =c;
        end
        c = (a+b)/2;
        fc = eval(subs(eqn,x,c));
    end
    fprintf('\nRoot is: %f\n', c);
end
```

Output



```
Enter non-linear equations: x^3-x-1
Enter first guess: 1
Enter second guess: 2
Tolerable error: 0.001


a            b            c            f(c)
1.000000     2.000000     1.500000     0.875000
1.000000     1.500000     1.250000     -0.296875
1.250000     1.500000     1.375000     0.224609
1.250000     1.375000     1.312500     -0.051514
1.312500     1.375000     1.343750     0.082611
1.312500     1.343750     1.328125     0.014576
1.312500     1.328125     1.320312     -0.018711
1.320312     1.328125     1.324219     -0.002128
1.324219     1.328125     1.326172     0.006209
1.324219     1.326172     1.325195     0.002037

Root is: 1.324707
>>
```

Figure 1: Bisection method

Finding root of nonlinear equation using False Position Method.

Introduction

False Position Method

False position method is based on the fact that if $f(x)$ is real and continuous function, and for two initial guesses a and b brackets the root such that: $f(a) \times f(b) < 0$ then there exists at least one root between a and b.

If a and b are two guesses then we compute new approximated root as:

$$c = a - \frac{f(a) \times (b - a)}{f(b) - f(a)}$$

Now we have following three different cases:

- If $f(c) = 0$ then the root is c.
- If $f(a) \times f(b) < 0$ then root lies between a and c.
- If $f(a) \times f(c) > 0$ then root lies between b and c.

And then the process is repeated until we find the root within desired accuracy.[2]

Tools Used

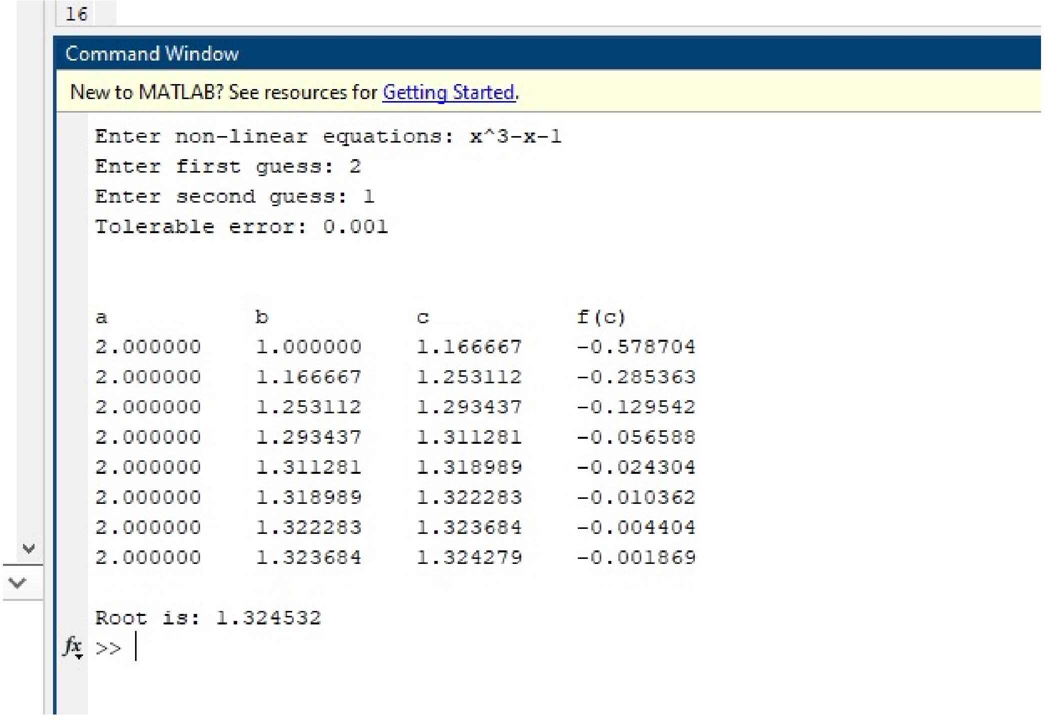
- MATLAB R2021a - for writing and running code.
- MacTeX -LaTeXcompiler.
- VS Code with LaTeXworkshop extension as a text editor.

Process

Code for False Position:

```matlab
% Clearing Screen
clc
% Setting x as symbolic variable
syms x;

% Input Section
eqn = input('Enter non-linear equations: ');
a = input('Enter first guess: ');
b = input('Enter second guess: ');
e = input('Tolerable error: ');

% Finding Functional Value
fa = eval(subs(eqn,x,a));
fb = eval(subs(eqn,x,b));

% Implementing False Position Method
if fa*fb > 0
    disp('Given initial values do not bracket the root.');
else
    c = a - (a-b) * fa/(fa-fb);
    fc = eval(subs(eqn,x,c));
    fprintf('\n\na\t\t\tb\t\t\tc\t\t\tf(c)\n');
    while abs(fc)>e
        fprintf('%f\t%f\t%f\t%f\n',a,b,c,fc);
        if fa*fc< 0
            b =c;
            fb = eval(subs(eqn,x,b));
        else
            a =c;
            fa = eval(subs(eqn,x,a));
        end
        c = a - (a-b) * fa/(fa-fb);
        fc = eval(subs(eqn,x,c));
    end
    fprintf('\nRoot is: %f\n', c);
end
```

Output



```
Enter non-linear equations: x^3-x-1
Enter first guess: 2
Enter second guess: 1
Tolerable error: 0.001


a           b           c           f(c)
2.000000    1.000000    1.166667    -0.578704
2.000000    1.166667    1.253112    -0.285363
2.000000    1.253112    1.293437    -0.129542
2.000000    1.293437    1.311281    -0.056588
2.000000    1.311281    1.318989    -0.024304
2.000000    1.318989    1.322283    -0.010362
2.000000    1.322283    1.323684    -0.004404
2.000000    1.323684    1.324279    -0.001869

Root is: 1.324532
>> 
```

Figure 2: Flase Position method

Finding root of nonlinear equation using Newton-Raphson Method.

Introduction

Newton-Raphson Method

Newton Raphson Method is an open method and starts with one initial guess for finding real root of non-linear equations.

In Newton Raphson method if a is initial guess then next approximated root b is obtained by following formula:

$$b = a - \frac{f(a)}{f'(a)}$$

From the above equation, we get the x intersect of the slope at point $(a, f(a))$. Repeating this process get this point closer and closer to real root of the non-linear equation.[3]

Tools Used

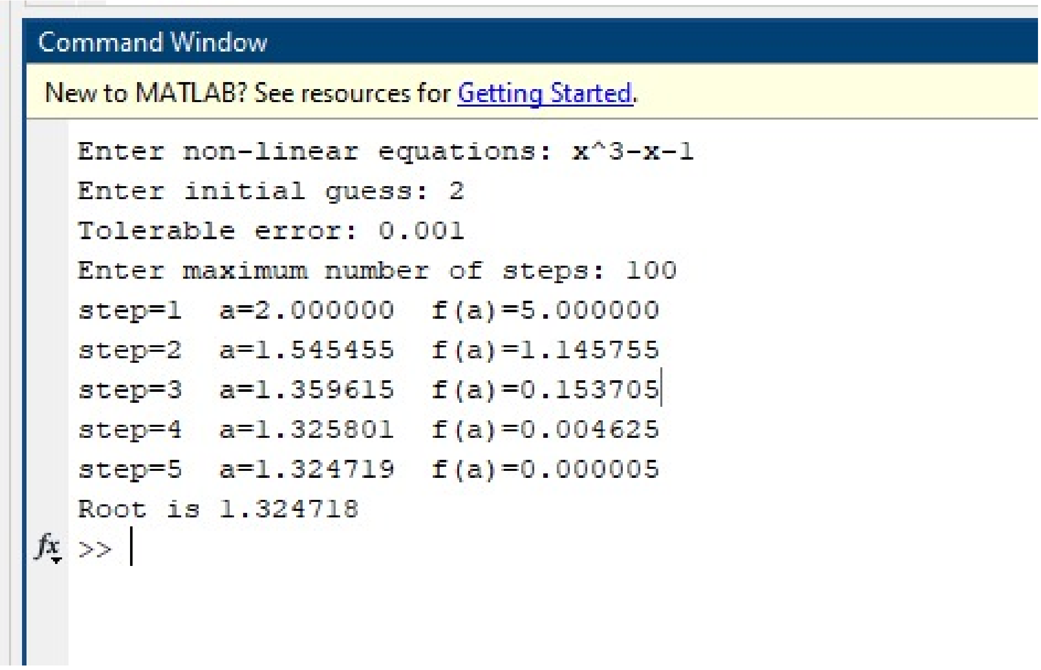
- MATLAB R2021a - for writing and running code.
- MacTeX -LaTeXcompiler.
- VS Code with LaTeXworkshop extension as a text editor.

Process

Code for Newton-Raphson:

```

% Clearing Screen
clc

% Setting x as symbolic variable
syms x;

% Input Section
eqn = input('Enter non-linear equations: ');
a = input('Enter initial guess: ');
e = input('Tolerable error: ');
N = input('Enter maximum number of steps: ');

% Initializing step counter
step = 1;

% Finding derivate of given function
g = diff(eqn,x);

% Finding Functional Value
fa = eval(subs(eqn,x,a));

while abs(fa)> e
    fa = eval(subs(eqn,x,a));
    ga = eval(subs(g,x,a));
    if ga == 0
        disp('Division by zero.');
        break;
    end

    b = a - fa/ga;
    fprintf('step=%d\ta=%f\tf(a)=%f\n',step,a,fa);
    a = b;

    if step>N
       disp('Not convergent');
       break;
    end
    step = step + 1;
end

fprintf('Root is %f\n', a);
```

Output



New to MATLAB? See resources for Getting Started.

```
Enter non-linear equations: x^3-x-1
Enter initial guess: 2
Tolerable error: 0.001
Enter maximum number of steps: 100
step=1  a=2.000000  f(a)=5.000000
step=2  a=1.545455  f(a)=1.145755
step=3  a=1.359615  f(a)=0.153705
step=4  a=1.325801  f(a)=0.004625
step=5  a=1.324719  f(a)=0.000005
Root is 1.324718
>>
```

Figure 3: Newton-Raphson method

Functions

The functions used to do the three methods in MATLAB are as such with brief description of each of them:

syms Create symbolic scalar variables and functions, and matrix variables and functions. By using *syms x*, a variable x is declared for the the code. So anywhere in the code input, if there is x, it can be accessed as a variable.

input() $x = input(prompt)$ displays the text in prompt and waits for the user to input a value and press the Return key. The user can enter expressions, like $pi/4$ or $rand(3)$, and can use variables in the workspace.

eval() Evaluates a MATLAB expressions.

subs() Symbolic substitution. subs(s,new) returns a copy of s, replacing all occurrences of the symbolic scalar variable (declared as *syms x*) in s with *new* (*new* can be a number or another variable), and then evaluates s.

disp() $disp(X)$ displays the value of variable X without printing the variable name.

fprintf() Like the $printf()$ function in C language. Unline $disp()$ using this function data can be written in text.

if_else *if expression*, *statements*, *end* evaluates an expression, and executes a group of statements when the expression is true. An expression is true when its result is nonempty and contains only nonzero elements (logical or real numeric). Otherwise, the expression is false.
The *elseif* and *else* blocks are optional. The statements execute only if previous expressions in the *if... end* block are false. An if block can include multiple elseif blocks.

while *while expression*, *statements*, *end* evaluates an expression, and repeats the execution of a group of statements in a loop while the expression is true. An expression is true when its result is nonempty and contains only nonzero elements (logical or real numeric). Otherwise, the expression is false.

These function are the newly learned ones for these experiments.[4]

References

[1] "Bisection Method Algorithm (Step Wise)," Nov. 2023, [Online; accessed 17. Nov. 2023]. [Online]. Available: https://www.codesansar.com/numerical-methods/bisection-method-algorithm.htm

[2] "Regula Falsi (False Position) Method Algorithm (Step Wise)," Nov. 2023, [Online; accessed 17. Nov. 2023]. [Online]. Available: https://www.codesansar.com/numerical-methods/regula-falsi-or-false-position-method-algorithm.htm

[3] Electrical Technology, "What is the Equivalent Circuit of Induction Motor?" *ELECTRICAL TECHNOLOGY*, Sep. 2022. [Online]. Available: https://www.electricaltechnology.org/2022/04/equivalent-circuit-induction-motor.html

[4] "MATLAB Documentation," Nov. 2023, [Online; accessed 17. Nov. 2023]. [Online]. Available: https://www.mathworks.com/help/matlab/ref