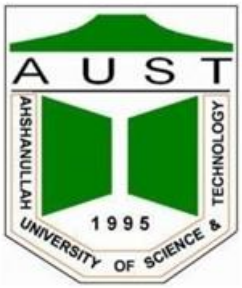

AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT
OF
ELECTRICAL AND ELECTRONIC ENGINEERING

LAB REPORT

COURSE NO	: EEE 2226
COURSE NAME	: Numerical Technique Laboratory
EXPERIMENT NO	: 06
DATE OF SUBMISSION	: 08/09/2025

Submitted by :

Name	: Rafayet Ahammad
ID	: 20230105206
Year	: 2nd
Semester	: 2nd
Section	: D(2)

```
clc;
clear;
close all;

%nonlinear equation input
eqn = input('Enter the nonlinear equation in terms of x
(e.g. x^2-3): ','s');
syms x;
fs = str2sym(eqn);      % symbolic expression
f = matlabFunction(fs); % convert to numerical function
fderiv = matlabFunction(diff(fs)); % derivative for Newton-
Raphson

%4 methods for the user
disp('Choose a method:');
disp('1. Bisection');
disp('2. False Position');
disp('3. Newton-Raphson');
disp('4. Secant');
choice = input('Enter your choice (1-4): ');

roots_found = [];

% ask for guesses from user
switch choice
    case 1 % Bisection
        intervals = input('Enter intervals as [xlow1 xup1;
xlow2 xup2; ...]: ');
        tol  = input('Enter tolerance: ');
        for k = 1:size(intervals,1)
            row = intervals(k,:);
            xlow = row(1);
            xup  = row(2);
```

```
            [root, iter] = bisection_method(f, xlow, xup,
tol);
            disp(['Root in interval [' num2str(xlow) ','
num2str(xup) '] = ' num2str(root)]);
            roots_found(end+1) = root;
        end

    case 2 % False Position
        intervals = input('Enter intervals as [xlow1 xup1;
xlow2 xup2; ...]: ');
        tol  = input('Enter tolerance: ');
        for k = 1:size(intervals,1)
            row = intervals(k,:);
            xlow = row(1);
            xup  = row(2);
            [root, iter] = false_position_method(f, xlow,
xup, tol);
            disp(['Root in interval [' num2str(xlow) ','
num2str(xup) '] = ' num2str(root)]);
            roots_found(end+1) = root;
        end

    case 3 % Newton_Raphson
        guesses = input('Enter initial guesses as [g1; g2;
g3; ...]: ');
        tol = input('Enter tolerance: ');
        for k = 1:size(guesses,1)
            x0 = guesses(k);
            [root, iter] = newton_raphson_method(f, fderiv,
x0, tol);
            disp(['Root from guess ' num2str(x0) ' = '
num2str(root)]);
            roots_found(end+1) = root;
        end
```

```
    case 4 % Secant
        pairs = input('Enter guess pairs as [x01 x02; x11
x12; ...]: ');
        tol = input('Enter tolerance: ');
        for k = 1:size(pairs,1)
            row = pairs(k,:);
            x0 = row(1);
            x1 = row(2);
            [root, iter] = secant_method(f, x0, x1, tol);
            disp(['Root from pair [' num2str(x0) ','
num2str(x1) '] = ' num2str(root)]);
            roots_found(end+1) = root;
        end

    otherwise
        error('Invalid choice');
end

disp('All roots found');


 #User Defined Functions

function [root, iter] = bisection_method(f, xlow, xup, tol)
    ylow = f(xlow);
    yup = f(xup);
    if ylow*yup > 0
        error('Root is not likely in this interval');
    end
    iter = 0;
    while (xup - xlow) >= tol
        iter = iter + 1;
        xmid = (xlow + xup)/2;
```

```
        ymid = f(xmid);
        if ymid == 0
            break;
        elseif ymid * ylow > 0
            xlow = xmid;
        else
            xup = xmid;
        end
    end
    root = (xlow + xup)/2;
end

function [root, iter] = false_position_method(f, xlow, xup,
tol)
    ylow = f(xlow);
    yup = f(xup);
    if ylow*yup > 0
        error('Root is not likely in this interval');
    end
    xm = xup - f(xup)*(xlow-xup)/(f(xlow)-f(xup));
    ym = f(xm);
    iter = 1;
    while abs(ym) >= tol
        if ym * ylow > 0
            xlow = xm;
        else
            xup = xm;
        end
        xm = xup - f(xup)*(xlow-xup)/(f(xlow)-f(xup));
        ym = f(xm);
        iter = iter + 1;
    end
    root = xm;
end
```

```
function [root, iter] = newton_raphson_method(f, fderiv, x0,
tol)
    iter = 0;
    while abs(f(x0)) > tol
        x1 = x0 - f(x0)/fderiv(x0);
        x0 = x1;
        iter = iter + 1;
        if iter > 1000, break; end
    end
    root = x0;
end

function [root, iter] = secant_method(f, x0, x1, tol)
    y0 = f(x0);
    y1 = f(x1);
    iter = 0;
    while abs(y1) > tol
        x2 = x1 - (x0-x1)/(y0-y1)*y1;
        y2 = f(x2);
        x0 = x1; y0 = y1;
        x1 = x2; y1 = y2;
        iter = iter + 1;
        if iter > 1000, break; end
    end
    root = x1;
end
```

The code is uploaded here: https://raw.githubusercontent.com/carbolicacid/MatLab1.0/refs/heads/main/project/R06E01.m