Problem Name:

Finding the root of non-linear equations using Bisection and False Position Method.

Apparatus:

- Computing Device (e.g., Computer)
- Programming Environment (MATLAB R2016b)

Method:

Closed type methods offer numerical solutions for root finding within specified intervals. The Bisection Method starts with an interval, calculates the midpoint, evaluates the function, and iteratively narrows down the interval. Meanwhile, the Regula Falsi Method employs linear interpolation to estimate the root, updates the interval, and refines the approximation through iterative steps. Both methods systematically converge to the root, providing effective solutions for non-linear equations.

Algorithm:

- ### Bisection Method Algorithm

    1. start
    2. Define function f(x)
    3. Choose initial guesses x0 and x1 such that f(x0) f(x1) < 0
    4. Choose pre-specified tolerable error e.
    5. Calculate new approximated root as x2 = (x0 + x1)/2
    6. Calculate f(x0) f(x2)
        a. if f(x0) f(x2) < 0 then x0 = x0 and x1 = x2

b. if f(x0) f(x2) > 0 then x0 = x2 and x1 = x1

c. if f(x0) f(x2) = 0 then goto (8)

7. if |f(x2)| > e then goto (5) otherwise goto (8)
8. Display x2 as root.
9. Stop

- **False Position Method Algorithm**

1. start
2. Define function f(x)
3. Choose initial guesses x0 and x1 such that f(x0) f(x1) < 0
4. Choose pre-specified tolerable error e.
5. Calculate new approximated root as:

   x2 = x0 - ((x0-x1) * f(x0))/(f(x0) - f(x1)

6. Calculate f(x0) f(x2)

   a. if f(x0) f(x2) < 0 then x0 = x0 and x1 = x2

   b. if f(x0) f(x2) > 0 then x0 = x2 and x1 = x1

   c. if f(x0) f(x2) = 0 then goto (8)

7. if |f(x2)|>e then goto (5) otherwise goto (8)
8. Display x2 as root.
9. Stop

Code:

Bisection Method Code-

```
%Clearing Screen
clc

% Setting x as symbolic variable
syms x;

% Input Section
y = 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(y,x,a));
fb = eval(subs(y,x,b));

% Implementing Bisection Method
if fa*fb > 0
   disp('Given initial values do not bracket the root.');

end
while(fa*fb>0)
```

```
    a = a+1;
    b = b+1;
    fa = eval(subs(y,x,a));
    fb = eval(subs(y,x,b));

end
    c = (a+b)/2;
    fc = eval(subs(y,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(y,x,c));

  end
  fprintf('\nRoot is: %f\n', c);
```

False Position Method Code-

```
% Clearing Screen
clc

% Setting x as symbolic variable
syms x;

% Input Section
y = 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(y,x,a));
fb = eval(subs(y,x,b));

% Implementing Bisection 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(y,x,c));
    fprintf('\n\na\t\tb\t\tc\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(y,x,b));
        else
            a =c;
            fa = eval(subs(y,x,a));
        end
        c = a - (a-b) * fa/(fa-fb);
        fc = eval(subs(y,x,c));
    end
    fprintf('\nRoot is: %f\n', c);
end
```

Input:

- **For Bisection Method**

  Enter non-linear equations: sin(x)+cos(x)+exp(x)-8

  Enter first guess: 2

  Enter second guess: 3

  Tolerable error: 0.00001

- **For Regula Falsi Method**

  Enter non-linear equations: sin(x)+cos(x)+exp(x)-8

  Enter first guess: 2

  Enter second guess: 3

  Tolerable error: 0.00001

Output:

- **For Bisection Method**

BisectionMethod.m

/MATLAB Drive/MATLAB Reports/BisectionMethod.m

```
%Clearing Screen
clc

% Setting x as symbolic variable
syms x;

% Input Section

y = 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(y,x,a));
fb = eval(subs(y,x,b));

% Implementing Bisection Method

if fa*fb > 0
    disp('Given initial values do not bracket the root.');

end
while(fa*fb>0)
    a = a+1;
    b = b+1;
    fa = eval(subs(y,x,a));
    fb = eval(subs(y,x,b));

end
    c = (a+b)/2;
    fc = eval(subs(y,x,c));
    fprintf('\n\na\t\tb\t\tc\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(y,x,c));

  end
  fprintf('\nRoot is: %f\n', c);
```

Command Window

```
Enter non-linear equations:
sin(x)+cos(x)+exp(x)-8
Enter first guess:
2
Enter second guess:
3
Tolerable error:
0.00001


a                       b                       c                       f(c)
2.000000        3.000000        2.500000        3.979822
2.000000        2.500000        2.250000        1.637635
2.000000        2.250000        2.125000        0.696951
2.000000        2.125000        2.062500        0.275011
2.000000        2.062500        2.031250        0.075106
2.000000        2.031250        2.015625        -0.022202
2.015625        2.031250        2.023438        0.026235
2.015625        2.023438        2.019531        0.001963
2.015625        2.019531        2.017578        -0.010133
2.017578        2.019531        2.018555        -0.004089
2.018555        2.019531        2.019043        -0.001064
2.019043        2.019531        2.019287        0.000449
2.019043        2.019287        2.019165        -0.000307
2.019165        2.019287        2.019226        0.000071
2.019165        2.019226        2.019196        -0.000118
2.019196        2.019226        2.019211        -0.000024
2.019211        2.019226        2.019218        0.000024

Root is: 2.019215
>>
```

Code Issues

Open Files

- **For False Position Method**

RegulaFalsiMethod.m | BisectionMethod.m

/MATLAB Drive/MATLAB Reports/RegulaFalsiMethod.m

```
% Clearing Screen
clc

% Setting x as symbolic variable
syms x;

% Input Section
y = 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(y,x,a));
fb = eval(subs(y,x,b));

% Implementing Bisection 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(y,x,c));
    fprintf('\n\na\t\tb\t\tc\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(y,x,b));
        else
            a =c;
            fa = eval(subs(y,x,a));
        end
        c = a - (a-b) * fa/(fa-fb);
        fc = eval(subs(y,x,c));
    end
    fprintf('\nRoot is: %f\n', c);
end
```

Command Window

```
Enter non-linear equations:
sin(x)+cos(x)+exp(x)-8
Enter first guess:
2
Enter second guess:
3
Tolerable error:
0.00001


a                    b                    c                    f(c)
2.000000        3.000000        2.010374        -0.054516
2.010374        3.000000        2.015152        -0.025119
2.015152        3.000000        2.017349        -0.011551
2.017349        3.000000        2.018358        -0.005306
2.018358        3.000000        2.018821        -0.002437
2.018821        3.000000        2.019034        -0.001119
2.019034        3.000000        2.019132        -0.000514
2.019132        3.000000        2.019177        -0.000236
2.019177        3.000000        2.019197        -0.000108
2.019197        3.000000        2.019207        -0.000050
2.019207        3.000000        2.019211        -0.000023
2.019211        3.000000        2.019213        -0.000010

Root is: 2.019214
>>
```

Discussion:

In this report, the problem involved the implementation of the Bisection and Regula Falsi (False Position) methods for root finding in non-linear equations.

Throughout the coding process, certain challenges surfaced, particularly in handling input parameters. Additionally, working with MATLAB for the first time posed a learning curve, adding an extra layer of complexity to the implementation. Despite these initial hurdles, the report systematically evaluates the performance of closed type methods, emphasizing aspects like convergence rates, accuracy, and stability. The choice between Bisection and Regula Falsi is further explored, considering factors such as function smoothness and the initial guess for the root. The algorithms and code presented here not only address the core problem but also highlight the learning experiences encountered during the coding phase.