1 Experiment No. 2

2 Experiment Title

Solving Non-linear Equation Using Bisection Method in MATLAB.

3 Objective

The objectives of this lab are:

- To determine two approximations between which the root of the equation exists.
- To display the approximations, root and error for each iteration by bisection method in a tabular form.

4 Theory

The Bisection Method is a numerical technique used to find the roots of a non-linear equation of the form:

$$f(x) = 0 \tag{1}$$

This method is based on the Intermediate Value Theorem, which states that if a function $f(x)$ is continuous on the interval $[a, b]$ and $f(a) \cdot f(b) < 0$, then there exists at least one root ξ in (a, b) such that $f(\xi) = 0$.

Algorithm: The Bisection Method follows these steps:

1. Choose two initial points a and b such that $f(a) \cdot f(b) < 0$, ensuring that a root lies within the interval.
2. Compute the midpoint m of the interval:

$$m = \frac{a+b}{2} \tag{2}$$

3. Evaluate $f(m)$:
   (a) If $f(m) = 0$, then m is the root, and the process terminates.
   (b) If $f(a) \cdot f(m) < 0$, set $b = m$ and repeat the process.
   (c) Otherwise, set $a = m$ and repeat the process.
4. Continue iterating until the stopping criterion is met, which can be:
   (a) A predefined error tolerance $|b - a| < \epsilon$.
   (b) A fixed number of iterations.

Convergence Analysis: The Bisection Method has a linear convergence rate. At each iteration, the interval width is halved, leading to the error decreasing as:

$$|x_{n+1} - x_n| = \frac{|b-a|}{2^n} \tag{3}$$

where n is the iteration number. Although slow compared to other methods like Newton-Raphson, the Bisection Method is robust and guarantees convergence if the initial conditions are satisfied.

4.1 Solving Non-linear Equation Using Bisection Method

Here $f(x) = x^3 - 2x + 5$, error is assumed to be 0.001

MATLAB Code

```matlab

clc
clear
f= @(x)  1*x^3 - 2*x + 5;
error = 0.001;
N = 500;

for i = -1000:1000
if f(i) * f(i+1) < 0
b = i;
c = i + 1;
break;
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k = 1:N
m = (b + c) / 2;   % Compute the midpoint of the interval [b, c]
er = abs(b-c);    % Compute the current error (length of the interval)
fprintf('Iteration %d: b = %.6f, c = %.6f, m = %.6f, f(m) = %.6f, error =
    %.6f\n', k, b, c, m, f(m), er);

if f(b)*f(m) < 0
c=m;
else
b=m;
end

if f(m) == 0
fprintf('The root of the equation is= %.6f', m);
break;
elseif er <= error
fprintf('The root of the equation is= %.6f', m);
break;
end
end
```

Listing 1: Solving Non-linear Equation Using Bisection Method in MATLAB.

4.2 Result Shown in Command Window

```
Iteration 1: b = -3.000000, c = -2.000000, m = -2.500000, f(m) = -5.625000,
   error = -5.625000
Iteration 2: b = -2.500000, c = -2.000000, m = -2.250000, f(m) = -1.890625,
   error = -1.890625
Iteration 3: b = -2.250000, c = -2.000000, m = -2.125000, f(m) = -0.345703,
   error = -0.345703
Iteration 4: b = -2.125000, c = -2.000000, m = -2.062500, f(m) = 0.351318,
   error = 0.351318
Iteration 5: b = -2.125000, c = -2.062500, m = -2.093750, f(m) = 0.008942,
   error = 0.008942
Iteration 6: b = -2.125000, c = -2.093750, m = -2.109375, f(m) = -0.166836,
   error = -0.166836
Iteration 7: b = -2.109375, c = -2.093750, m = -2.101562, f(m) = -0.078562,
   error = -0.078562
Iteration 8: b = -2.101562, c = -2.093750, m = -2.097656, f(m) = -0.034714,
   error = -0.034714
Iteration 9: b = -2.097656, c = -2.093750, m = -2.095703, f(m) = -0.012862,
   error = -0.012862
Iteration 10: b = -2.095703, c = -2.093750, m = -2.094727, f(m) = -0.001954,
    error = -0.001954
Iteration 11: b = -2.094727, c = -2.093750, m = -2.094238, f(m) = 0.003495,
   error = 0.003495
The root of the equation is= -2.094238>>
```

Listing 2: Command Window

5 Discussion

In this session, the Bisection Method was introduced to solve a non-linear equation, highlighting its effectiveness in root-finding. Initially, the code was written to assume an interval b and c, within which a root was expected. By iteratively computing the midpoint of the interval and checking the conditions, the root of the equation was determined. It was observed that this method guarantees finding only one root at a time. From the results, it was found that with an error tolerance of 0.001, the method required 11 iterations to converge, yielding a root of -2.094238 with an approximate error of 0.0149%. Overall, the session successfully achieved its objectives for solving equations using MATLAB.