**Name: Ali Haider CMS ID: 404065**

**Section: SE-13 A Date: 4th March 2024**

**Numerical Methods Lab # 05**

**Task 01 (Bisection Method):**

**MATLAB Code:**

syms x;

f\_prompt = "Enter function: ";

a = input(f\_prompt, 's');

f = str2func(['@(x) ' a]);

xl = input('Enter lower guess:');

xu = input('Enter upper guess:');

tol = input('Enter tolerance (recommended 0.001):');

while f(xl) \* f(xu) > 0

disp("Wrong Guess! Enter Again")

xl = input('Enter lower guess:');

xu = input('Enter upper guess:');

end

counter = 0;

error = Inf;

while error > tol && counter < 100

xr = (xl + xu) / 2;

if f(xl) \* f(xr) < 0

xu = xr;

else

xl = xr;

end

error = abs((xu - xl) / 2);

counter = counter + 1;

end

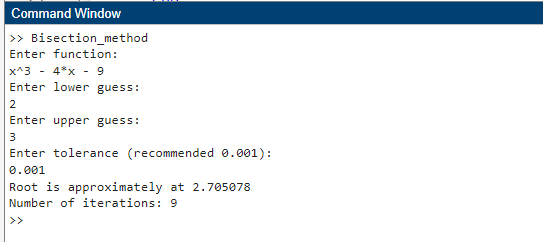
fprintf('Root is approximately at %f\n', xr);

fprintf('Number of iterations: %d\n', counter);

**Explanation:**

* This code demonstrates the implementation of the bisection method in MATLAB for numerical root finding.
* The core functionality lies in the iterative refinement of the search interval based on the signs of the function at its boundaries.
* The optional input validation loop is added to check the condition for existing of root.
* The maximum number of iterations (100) can be adjusted based on the function's complexity and desired convergence speed. While loop is not efficient because it would result in bugs like wrong declaration of errors

**Output Screenshots:**



**(2)**

A screenshot of a computer program

Description automatically generated

**(3) In case of wrong Guesses:**

A screen shot of a computer program

Description automatically generated

**Task 02 (Newton – Raphson Method):**

**MATLAB Code:**

a=input('Enter the function in the form of variable x:','s');

x(1)=input('Enter Initial Guess:');

error=input('Enter allowed Error:');

f=inline(a, 'x');

dif=diff(str2sym(a));

d=inline(dif, 'x');

for i=1:100

x(i+1) = x(i) - f(x(i)) / d(x(i));

err(i) = abs(x(i+1) - x(i));

if err(i) < error

break;

end

end

root=x(i+1);

fprintf("the root is: %f\n", root);

**Explanation:**

This code is a basic implementation of the Newton-Raphson method for finding the root of a given function. It starts by prompting the user to enter a mathematical function in terms of the variable 'x' and initializes the initial guess for the root and the allowed error.

The main loop of the code iterates up to a maximum of 100 times or until the error falls below the specified threshold. Within each iteration, the Newton-Raphson formula is applied to compute the next approximation of the root. The error between successive approximations is calculated and stored in the err array.

If the error falls below the specified threshold, the loop terminates early to save computation time. Finally, the code prints out the approximate root of the function found by the Newton-Raphson method.

**Purpose of diff Function:**

The diff function is used to compute the derivative of the provided function with respect to 'x'. The str2sym function converts the input string into a symbolic expression, which can then be differentiated using the diff function. The derivative function obtained is then converted into an inline function using inline.

The sym and diff functions are crucial for symbolically representing and differentiating the user-provided function, enabling the implementation of the Newton-Raphson method in MATLAB.

**Purpose of sym Function:**

The sym function in MATLAB is used to create symbolic objects, which represent mathematical expressions and variables. These symbolic objects are distinct from numerical values and allow you to perform analytical computations instead of numerical ones.

**Output Screenshots:**

