**Problem 1 (50 points):**

The roots of this function were found to be 1.003312, 3.664579, and 5.532108. These roots were found using the False Position Method, followed by Newton Raphson’s Method. In order to utilize False Position, an interval is required within which the root in question lies. These intervals were found by graphing the function in MATLAB. A MATLAB function for the given function was created and the outputs of this function, given an array of input, were graphed. After graphing the function, I used the MATLAB data cursor to select points that were close to the zeroes. This step is shown in the inserted graph. Also, as extra assurance, I selected x values where the function was increasing and decreasing and calculated the derivative at those points. In addition, I selected points on either side of the root in question and evaluated the function at the points. Because the derivative changes from positive to negative to positive, we know the function may cross the y-axis three times. Evaluating the function at points between the roots allows for us to say with certainty that there is a root, because the sign of the function changes when there is a root.

The False Position Method was utilized to narrow the root intervals. A 10-iteration function was used for this step. The root intervals are shown below in the MATLAB output section, while the function itself is pasted in the MATLAB code section. After finding the new x-value intervals, the function was evaluated at each of these points (so twice for every root). This step is shown below in the MATLAB output. The x-value that yielded a function value closer to 0 for each root interval is used in the next step, which is the Newton Raphson’s Method. The code is displayed in the MATLAB code section of this problem. This function was used for each x-value to find the three roots of the function that satisfied the question—that is, f(x) evaluates to less than 10^-7. This verification step is used as the while loop condition for the Newton Raphson Method.

As for the reasoning behind using these methods, I used the False Position simply because I wanted to give myself more experience using this method over the Bisection Method. As for finding the roots, I used Newton Raphson’s Method because the given function is simple enough where a derivative MATLAB function can easily be written and used within the code for Newton Raphson’s Method.

Chart

Description automatically generated

**Root intervals:** [0.5, 1.5]; [3, 4]; [5, 6]

**MATLAB output:**

>> deriv\_1 = f\_deriv(1)

deriv\_1 =

12.1000

>> deriv\_3 = f\_deriv(3)

deriv\_3 =

-4.7000

>> deriv\_6 = f\_deriv(6)

deriv\_6 =

15.1000

>> val1 = displayf(0)

val1 =

**'Value of f(x): -20.3400000'**

>> val2 = displayf(2)

val2 =

**'Value of f(x): 5.8600000'**

>> val3 = displayf(5)

val3 =

**'Value of f(x): -2.8400000'**

>> val4 = displayf(6)

val4 =

**'Value of f(x): 5.4600000'**

>> root1 = FalsePostion(.5, 1.5);

iteration 1: 0.500000, 1.148988

iteration 2: 0.921185, 1.148988

iteration 3: 0.921185, 1.069026

iteration 4: 0.921185, 1.017132

iteration 5: 0.983454, 1.017132

iteration 6: 0.983454, 1.005311

iteration 7: 0.997639, 1.005311

iteration 8: 1.002618, 1.005311

iteration 9: 1.002618, 1.004365

iteration 10: 1.002618, 1.003752

**The root of the equation is located between 1.002618 and 1.003752**

>> root2 = FalsePostion(3, 4);

iteration 1: 3.000000, 3.685714

iteration 2: 3.470204, 3.685714

iteration 3: 3.617983, 3.685714

iteration 4: 3.664427, 3.685714

iteration 5: 3.664427, 3.679024

iteration 6: 3.664427, 3.674436

iteration 7: 3.664427, 3.671291

iteration 8: 3.664427, 3.669134

iteration 9: 3.664427, 3.667654

iteration 10: 3.664427, 3.666640

**The root of the equation is located between 3.664427 and 3.666640**

>> root2 = FalsePostion(5, 6);

iteration 1: 5.342169, 6.000000

iteration 2: 5.342169, 5.567258

iteration 3: 5.419187, 5.567258

iteration 4: 5.469852, 5.567258

iteration 5: 5.503181, 5.567258

iteration 6: 5.525106, 5.567258

iteration 7: 5.525106, 5.539529

iteration 8: 5.530042, 5.539529

iteration 9: 5.530042, 5.533288

iteration 10: 5.531152, 5.533288

**The root of the equation is located between 5.531152 and 5.533288**

>> displayf(1.003752)

ans =

**'Value of f(x): 0.0052979'**

>> displayf(1.002618)

ans =

'Value of f(x): -0.0083715'

>> displayf(3.666640)

ans =

'Value of f(x): -0.0102379'

>> displayf(3.664427)

ans =

**'Value of f(x): 0.0007573'**

>> displayf(5.533288)

ans =

'Value of f(x): 0.0099863'

>> displayf(5.531152)

ans =

**'Value of f(x): -0.0080824'**

>> Root1 = NewtonRhapson(1.003752)

iteration 1: 1.003312

iteration 2: 1.003312

**Root = 1.003312**

Root1 =

1.003312310557692

>> Root2 = NewtonRhapson(3.664427)

iteration 1: 3.664579

**Root = 3.664579**

Root2 =

3.664579368860922

>> Root3 = NewtonRhapson(5.531152)

iteration 1: 5.532109

iteration 2: 5.532108

**Root = 5.532108**

Root3 =

5.532108316875385

**MATLAB code:**

*Function:*

function value = f(x)

value = x.^3 - 10.2\*x.^2 + 29.5\*x - 20.34;

*Derivative of function:*

function value = f\_deriv(x);

value = 3\*x.^2 - 20.4\*x + 29.5;

*Display function value at given x:*

function out = displayf(x)

A = f(x);

out = sprintf('Value of f(x): %.7f', A);

*False Position Method Code:*

function Root1 = FalsePostion(xL, xR);

LeftSol = f(xL);

RightSol = f(xR);

if LeftSol\*RightSol < 0

LeftX = xL;

RightX = xR;

count = 0;

while count < 10

XApp = LeftX - ((RightX-LeftX)/(RightSol-LeftSol))\*LeftSol;

XApp\_Sol = f(XApp);

if LeftSol\*XApp\_Sol < 0

RightX = XApp;

else

LeftX = XApp;

end

count = count + 1;

fprintf(['iteration ', num2str(count)])

fprintf(': %.6f', LeftX)

fprintf(', %.6f \n', RightX)

end

else

disp(['There is no root between ', num2str(xL), ' and ', num2str(xR)])

end

Root1 = [];

fprintf('The root of the equation is located between %.6f', LeftX);

fprintf(' and %.6f \n', RightX);

*Newton Raphson’s Method Code:*

function Root = NewtonRhapson(x);

x\_n = x;

Value\_xn = f(x\_n);

count = 0;

while abs(Value\_xn) >= 10^(-7)

Value\_xn = f(x\_n);

Deriv\_xn = f\_deriv(x\_n);

x\_np1 = x\_n - (Value\_xn/Deriv\_xn);

x\_n = x\_np1;

count = count + 1;

fprintf(['iteration ', num2str(count)])

fprintf(': %.6f \n', x\_n)

Value\_xn = f(x\_n);

end

Root = x\_n;

fprintf('Root = %.6f \n', x\_n);

**Problem 2 (50 points):**

The rise time of this system is estimated to be 0.18541.

For this problem, I started by making a function for the system in MATLAB. Then, this system was graphed using MATLAB. For extra visualization, I also added a y = 0.3 and y = 2.7 line in red. To find the rise time of the system, I used the Bisection Method for root finding. Instead of using Newton Raphson or the Secant method, I allowed the root intervals for the Bisection Method to converge by increasing the number of loops within the Bisection Method function. Using the data cursor tool to estimate the t values that would yield y(t) values close to 0.3 and 2.7, I then used these intervals in the Bisection Method function. Because there was no y = 0 axis for this graph, I did some simple arithmetic to establish arbitrary axes for each root interval. By subtracting 0.3 or 2.7 from the y(t) values for each root interval, I was able to use the root finding method because doing so would “push” the y = 0.3 or y = 2.7 axis to a new y = 0 axis. After using the Bisection Method (see MATLAB output), I took these final t values and evaluated y(t) at these values. This was done using the displayy function (see MATLAB code and output) to print the values at the necessary precision. Because I used enough iterations of Bisect function (see MATLAB code), the t values converged enough where I could use either t value of the output interval because they were the same due to the accuracy of the sensor.

Chart

Description automatically generated

**Chart

Description automatically generated**

**Root Intervals:** [0.03, 0.07]; [0.23, 0.24]

**MATLAB output:**

>> root1 = Bisect(0.03, 0.07)

iteration 1: 0.03000, 0.05000

iteration 2: 0.04000, 0.05000

iteration 3: 0.04500, 0.05000

iteration 4: 0.04750, 0.05000

iteration 5: 0.04875, 0.05000

iteration 6: 0.04938, 0.05000

iteration 7: 0.04938, 0.04969

iteration 8: 0.04953, 0.04969

iteration 9: 0.04961, 0.04969

iteration 10: 0.04961, 0.04965

iteration 11: 0.04963, 0.04965

iteration 12: 0.04964, 0.04965

iteration 13: 0.04964, 0.04964

iteration 14: 0.04964, 0.04964

iteration 15: 0.04964, 0.04964

iteration 16: 0.04964, 0.04964

iteration 17: 0.04964, 0.04964

iteration 18: 0.04964, 0.04964

iteration 19: 0.04964, 0.04964

iteration 20: 0.04964, 0.04964

**The root of the equation is located between 0.04964 and 0.04964**

>> root2 = Bisect(0.23, 0.24)

iteration 1: 0.23500, 0.24000

iteration 2: 0.23500, 0.23750

iteration 3: 0.23500, 0.23625

iteration 4: 0.23500, 0.23562

iteration 5: 0.23500, 0.23531

iteration 6: 0.23500, 0.23516

iteration 7: 0.23500, 0.23508

iteration 8: 0.23504, 0.23508

iteration 9: 0.23504, 0.23506

iteration 10: 0.23504, 0.23505

iteration 11: 0.23504, 0.23505

iteration 12: 0.23504, 0.23505

iteration 13: 0.23504, 0.23505

iteration 14: 0.23504, 0.23505

iteration 15: 0.23504, 0.23505

iteration 16: 0.23505, 0.23505

iteration 17: 0.23505, 0.23505

iteration 18: 0.23505, 0.23505

iteration 19: 0.23505, 0.23505

iteration 20: 0.23505, 0.23505

**The root of the equation is located between 0.23505 and 0.23505**

>> out1 = displayy(0.04964)

out1 =

**'Value of y(t): 0.300000'**

>> out2 = displayy(0.23505)

out2 =

**'Value of y(t): 2.700043'**

>> 0.23505 - 0.04964

ans =

**0.18541**

**MATLAB code:**

*Function y(t):*

function value = y(t)

if t >= 0

value = 3.\*(1-exp(-6.\*t).\*(cos(8.\*t)+(3/4).\*sin(8.\*t)));

end

*Graph function:*

x = 0:0.005:5;

val = y(x);

plot(x, val)

grid on

hold on

yline( 2.7, '-r')

yline(0.3, '-r')

hold off

xlabel('t')

ylabel('y(t)')

title('Midterm 1 Problem 2')

*Bisection Method:*

function [Root1, NumIterations] = Bisect(xL, xR);

LeftSol = y(xL) - 0.3;

RightSol = y(xR) - 0.3;

if LeftSol\*RightSol < 0

LeftX = xL;

RightX = xR;

count = 0;

while count < 20

MidX = (LeftX + RightX)/2;

Middle = y(MidX) - 0.3;

if Middle\*(f(LeftX) - 0.3) < 0

RightX = MidX;

else

LeftX = MidX;

end

count = count + 1;

fprintf(['iteration ', num2str(count)])

fprintf(': %.5f', LeftX)

fprintf(', %.5f \n', RightX)

end

else

disp(['There is no root between ', num2str(xL), ' and ', num2str(xR)])

end

Root1 = [];

fprintf('The root of the equation is located between %.5f', LeftX);

fprintf(' and %.5f \n', RightX);

NumIterations = count;

*Value of y(t):*

function out1 = displayy(x)

A = y(x);

out1 = sprintf('Value of y(t): %.6f', A);