MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF MECHANICAL ENGINEERING

ME 310 – NUMERICAL METHODS

SPRING 2022

HOMEWORK 2

Muhammed Rüstem ŞEVİK – 2446888

I understand that this is an individual assignment. I affirm that I have not given or received any unauthorized help on this assignment, and that this work is my own.

Table of Contents

List of Figures

List of Tables

1. Introduction

In this homework assignment, students are expected to enhance their MATLAB skills and learn various numerical root finding algorithms by constructing a hybrid root finding algorithm in question one, by using secant and false position method in question seven and roots of nonlinear systems in question nine.

2. Body

Question 1

In this question students are expected to write a hybrid root finding algorithm similar to the "fzero" function of MATLAB. The question asks for creation of the root finding interval and after that usage of modified secant method and bisection method in a hybrid way.

Bisection Method formula.

$$x_r = \frac{x_l + x_u}{2} \tag{1}$$

Modified Secant Method formula.

$$x_{i+1} = x_i - \frac{\delta x_i f(x_i)}{f(x_i + \delta x_i) - f(x_i)} \tag{2}$$

Note that in the Phase 1 code works like the output of Run 4 not as described in the question. Also, since the original code is updated some of these test results are not same as the given output.

Code:

```
clc; clear;

f = @(x) sqrt(9.81*x/0.25) * tanh(sqrt(9.81*0.25/x)*4) - 35;
[root, iterations] = findRoot(f, 0, 1e-10, 1000);


function [root, iter] = findRoot(f, x0, tol, maxIter)

% Default value for maximum iteration
if nargin < 4
    maxIter = 1000;
end

% Phase 1 (searching for a bracketing interval):

s_iter = 1;
if isscalar(x0)
    fprintf("<strong> Trial       xL             xU          " + ...
        "     f(xL)           f(xU)        Method  </strong> \n")
```

```
    % initialize search width
    delta = x0/50;
    if delta == 0
        delta = 1/50;
    end
    delta_0 = delta;
    % search for bracketing interval
    sign_change = false;
    while ~sign_change % && delta < abs(x0)
        xL = x0 - delta;
        xU = x0 + delta;
        sign_change = sign(f(xL)) ~= sign(f(xU));
        fprintf('   %d\t %.6f\t %.6f\t %.6f\t %1.5e\t %s\n', ...
            s_iter, xL, xU, f(xL), f(xU), "Search")
        delta = delta_0 + 2 * delta;
        s_iter = s_iter + 1;
        if s_iter == 1001
            break
        end
    end

    % check if bracketing interval was found
    if ~sign_change
        error('Unable to find bracketing interval.');
    end
else
    % use provided interval as bracketing interval
    xL = x0(1);
    xU = x0(2);

    % check if bracketing interval is valid
    if sign(f(xL)) == sign(f(xU))
    error('The function does not change sign in the provided interval.');
    end
end

% Phase 2 (finding the root):
fprintf("<strong> Iter\t     xL\t\t     xU\t\t    root      " + ...
    "     f(root)       Method  </strong> \n")
iter = 1;
pert = (xU - xL) / 100;
xnew = (xL + xU) / 2; % initial guess being the mid-point of the interval
while iter < maxIter
    % fist use modified secant method (MSM)
    if abs(f(xnew)) < tol
        root = xnew;
        return
    end
    x_candidate = xnew - (f(xnew) * pert) / (f(xnew + pert) - f(xnew));
    if xL <= x_candidate && x_candidate <= xU
        xnew = x_candidate;
        fprintf('   %d\t %.6f\t %.6f\t %.6f\t %1.5e\t %s\n',...
            iter, xL, xU, xnew, f(xnew), "Modified secant")
    else % in here we have to swith to BM
        fprintf('   \t %.6f\t %.6f\t %.6f\t %1.5e\t %s\n', xL,...
            xU, x_candidate, f(x_candidate), "Modified secant (failed)")
        xnew = (xU + xL) / 2;
        fprintf('   %d\t %.6f\t %.6f\t %.6f\t %1.5e\t %s\n', iter,...
            xL, xU, xnew, f(xnew), "Bisection")
```

```
        if f(xL) * f(xnew) < 0
            xU = xnew;
            xnew = 0.5*(xL+xnew);
        elseif f(xnew) * f(xU) < 0
            xL = xnew;
            xnew = 0.5*(xnew+xU);
        end

    end

    % increment iterations
    iter = iter + 1;
end

% maxIter reached without finding root
error('Maximum iterations reached without finding root.');
end
```

Question 7

$$f(x) = \ln(x) \tag{3}$$

False Position Method formula.

$$x_{i+1} = x_{i-1} - \frac{(x_i - x_{i-1})f(x_{i-1})}{f(x_i) - f(x_{i-1})} \tag{4}$$

Secant Method formula.

$$x_{i+1} = x_i - \frac{f(x_i)f(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \tag{5}$$

Error definition.

$$\epsilon_a = \left| \frac{\text{Present Approximate Value} - \text{Previous Approximate Value}}{\text{Present Approximate Value}} \right| \cdot 100\% \tag{6}$$

Calculation Details

False Position Method

Table 1: Iteration details for false position method, $x_0 = 0.5$, $x_1 = 5$

Iterations	x	f(x)	ϵ_a
1	1.85463498	0.61768790	-
2	1.21630782	0.19581989	52.48072505
3	1.05852096	0.05687262	14.90635156
4	1.01616935	0.01604002	4.16777111

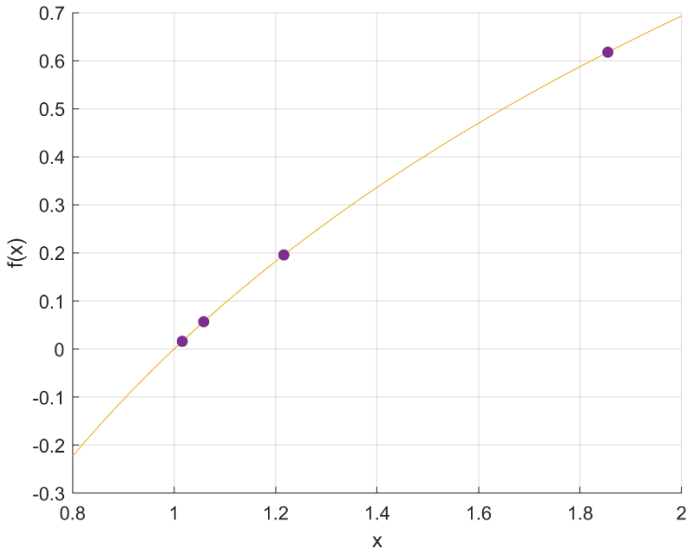

Figure 1: Scatter plot of iterations for false position method

Secant Method

Table 2: Iteration details for secant method, x_0 = 0.5, x_1= 5

Iterations	x	f(x)	ϵ_a
1	1.85463498	0.61768790	-

No plot here because the result is not successful. The method gave a negative number. The function ln(x) is not defined in the negative numbers. So, the process is terminated.

False Position Method

Table 3: Iteration details for false position method, x_0 = 5, x_1= 0.5

Iterations	x	f(x)	ϵ_a
1	1.85463498	0.61768790	-
2	1.21630782	0.19581989	52.48072505
3	0.92001335	-0.08336709	32.20545256
4	1.00848885	0.00845303	8.77307671

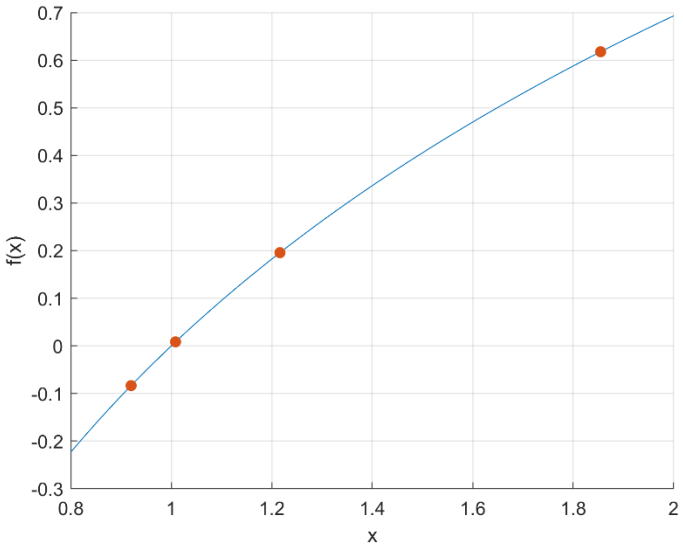

Figure 2: Scatter plot of iterations for secant method

Code:

```
clc; clear;
f = @(x) log(x);
false_position(f, 0.5, 5, 0, 4)
secant(f, 0.5, 5, 0, 4)
secant(f, 5, 0.5, 0, 4)

function false_position(f, x0, x1, e, i)
    fprintf('\n*** FALSE POSITION METHOD ***\n');
    step = 1;
    condition = true;
    previous_xr = 0;
    error = 100 * e;

    while condition
        xm = (x0 * f(x1) - x1 * f(x0)) / (f(x1) - f(x0));
        if xm ~= 0
            error = abs((previous_xr - xm) / xm * 100);
            condition = error > e;
        end
        if xm <= 0 || x0 <= 0 || x1 <= 0
            fprintf("Negative Value in ln")
            return
        end

        f_xm = f(xm);
        fprintf(['Iteration-%d, xm = %0.8f , f(xm) = %0.8f , ' ...
            'error = %0.8f\n'], step, xm, f_xm, error);

        if f(x0) * f(xm) < 0
            x1 = xm;
        elseif f(x0) * f(xm) > 0
            x0 = xm;
        else
            condition = false;
        end

        previous_xr = xm;
```

```
        step = step + 1;

        if step - 1 == i
            break;
        end
    end

    fprintf(['Last Estimate is : %0.8f, Approximate Percent Relative ' ...
        'Error : %.8f, Number of Iterations : %d , Function Value ' ...
        ': %0.8f\n'], xm, error, step-1, f_xm);
end

function secant(f, xs_old, x_s, e, i)
    fprintf('\n*** SECANT METHOD ***\n');
    step = 1;
    condition = true;
    error = 1 + e;

    while condition
        x_i = x_s - (f(x_s) * (xs_old - x_s)) / (f(xs_old) - f(x_s));
        if x_i <= 0 || x_s <= 0 || xs_old <= 0
            fprintf("Negative Value in ln")
            return
        end
        if x_i ~= 0
            error = abs((x_i - x_s) / x_i * 100);
            condition = error > e;
        end
        f_x_i = f(x_i);


        fprintf(['Iteration-%d, xm = %0.8f and f(xm) = %0.8f , ' ...
            'error = %0.8f\n'], step, x_i, f_x_i, error);

        xs_old = x_s;
        x_s = x_i;
        step = step + 1;
        if step -1 == i
            break;
        end
    end
    fprintf(['Last Estimate is : %0.8f, Approximate Percent Relative ' ...
        'Error : %.8f, Number of Iterations : %d , Function Value : ' ...
        '%0.8f\n'], x_s, error, step-1, f_x_i);

end
```

```
x = [1.85463498, 1.21630782, 1.05852096, 1.01616935];
y = [0.61768790, 0.19581989, 0.05687262, 0.01604002];
x = [1.85463498, 1.21630782, 0.92001335, 1.00848885];
y = [0.61768790, 0.19581989, -0.08336709, 0.00845303];

f = @(x) log(x);
hold on
grid on
fplot(f, [0.8 2])
scatter(x, y, "filled")
ylabel("f(x)"); xlabel("x")
exportgraphics(gca, "7c.png", "Resolution",300)
```

Question 9

Rearrange given parametric functions.

$$x_1 = x_2 \tag{7}$$
$$100t = 800 - 100\cos(\alpha)t \tag{8}$$
$$f_1(t,\alpha) = 800 - 100t(\cos(\alpha) + 1) \tag{9}$$
$$y_1 = y_2 \tag{10}$$
$$y_1 = 80t - 16t^2 = 80\sin(\alpha)t - 0.8t^2 \tag{11}$$
$$f_2(t,\alpha) = 15.2t^2 + 80t(\sin(\alpha) - 1) \tag{12}$$

Plot the functions f_1 and f_2 to obtain the intersection.

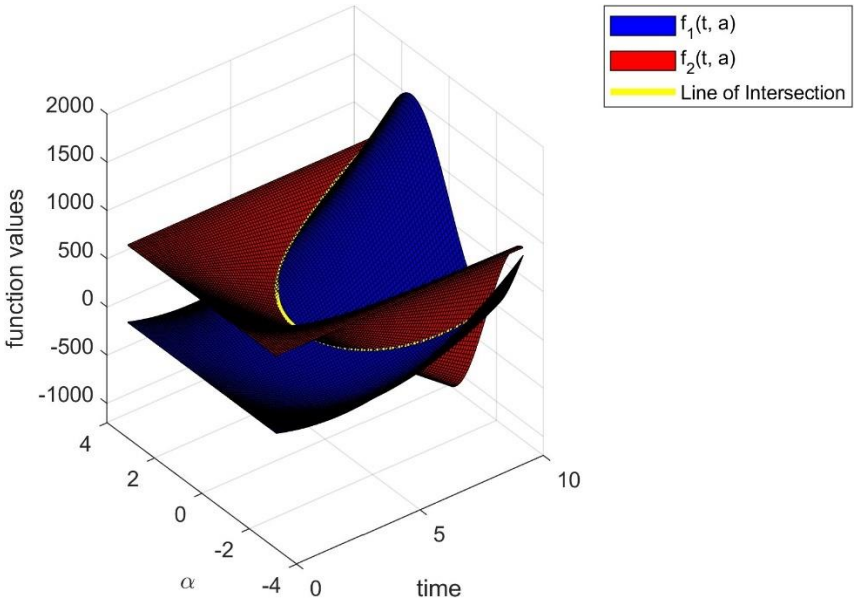

Figure 3: Surfaces f_1 and f_2 and the line of intersection

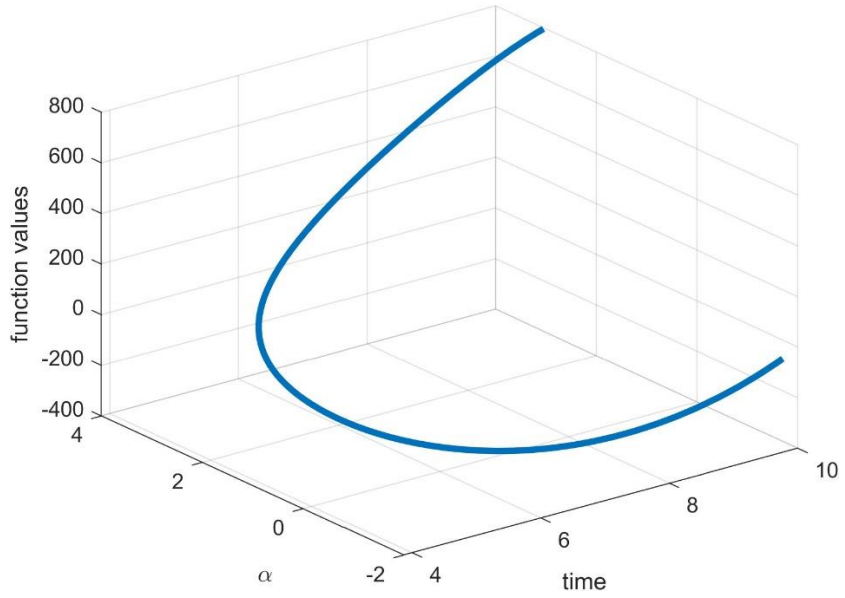

Figure 4: Line of intersection of surfaces f_1 and f_2

Since the intersection of these surfaces is not a single point it can be said that there are infinitely many solutions to the problem.

$$f_1(t,\alpha) = 800 - 100t(\cos(\alpha) + 1) \tag{13}$$

$$\frac{\partial f_1}{\partial t} = -100(\cos(\alpha) + 1), \frac{\partial f_1}{\partial \alpha} = 100t\sin(\alpha) \tag{14}$$

$$f_2(t,\alpha) = 15.2t^2 + 80t(\sin(\alpha) - 1) \tag{15}$$

$$\frac{\partial f_2}{\partial t} = 30.4t + 80(\sin(\alpha) - 1), \frac{\partial f_2}{\partial \alpha} = 80t\cos\ (\alpha) \tag{16}$$

Solution of the system will give the new values for the next iteration.

$$\begin{bmatrix} \frac{\partial f_1}{\partial t} & \frac{\partial f_1}{\partial \alpha} \\ \frac{\partial f_2}{\partial t} & \frac{\partial f_2}{\partial \alpha} \end{bmatrix} \begin{bmatrix} t_{i+1} \\ \alpha_{i+1} \end{bmatrix} = \begin{bmatrix} f_1(t_i,\alpha_i) \\ f_2(t_i,\alpha_i) \end{bmatrix} \tag{17}$$

For Newton Raphson method Jacobian matrix must be created as follows.

$$\begin{bmatrix} \frac{\partial f_1}{\partial t} & \frac{\partial f_1}{\partial \alpha} \\ \frac{\partial f_2}{\partial t} & \frac{\partial f_2}{\partial \alpha} \end{bmatrix} \tag{18}$$

$$\begin{bmatrix} -100(\cos(\alpha) + 1) & 100t\sin(\alpha) \\ 30.4t + 80(\sin(\alpha) - 1) & 80t\cos\ (\alpha) \end{bmatrix} \begin{bmatrix} t_{i+1} \\ \alpha_{i+1} \end{bmatrix} = \begin{bmatrix} f_1(t_i,\alpha_i) \\ f_2(t_i,\alpha_i) \end{bmatrix} \tag{19}$$

The result for the initial conditions $t = 2.5$ and $\alpha = 0.5$

$$t = 4.0542, \alpha = 0.2318$$

Code:

```
clc; clear;

f1 = @(t, a) 800 - 100 * t * (cos(a) + 1);
f2 = @(t, a) 15.2 * t^2 + 80 * t * (sin(a) - 1);
f1t = @(t, a) - 100 * (cos(a) + 1);
f1a = @(t, a) 100 * t * sin(a);
f2t = @(t, a) 30.4 * t^2 + 80 * (sin(a) - 1);
f2a = @(t, a) 80 * t * cos(a);

[t, a] = NR(f1, f2, f1t, f1a, f2t, f2a, 2.5, 0.5, 1e-6, 1000)


function [x, y] = NR(f1, f2, f1x, f1y, f2x, f2y, x0, y0, eps, max_iter)

    iter = 0; error = inf;
    x = x0; y = y0;

    while error > eps && iter < max_iter
        J = [f1x(x, y), f1y(x, y); f2x(x, y), f2y(x, y)]; % Jacobian matrix
        F = [f1(x, y); f2(x, y)];  % function value vector
        del = J \ F;  % Newton-Raphson step
        x = x - del(1);  % update solutions
```

```
        y = y - del(2);
        error = norm(del);  % compute error
        iter = iter + 1;
    end

end
```

```
clc; clear;

t = linspace(0, 10, 100);
a = linspace(-pi, pi, 100);
[t, a] = meshgrid(t, a);
func1 = 800 - 100 .* t .* (cos(a) + 1);
func2 = 15.2 .* t.^2 + 80 .* t .* (sin(a) - 1);

h1 = surf(t, a, func2, "FaceColor",[0 0 1]);
hold on
h2 = surf(t, a, func1, "FaceColor",[1 0 0]);
ylabel("\alpha"); xlabel("time"); zlabel("function values");
fdiff = func1 - func2;
C = contours(t, a, fdiff, [0 0]);
tL = C(1, 2:end);
aL = C(2, 2:end);
fL = interp2(t, a, func1, tL, aL);
h3 = line(tL, aL, fL, 'Color', 'yellow', 'LineWidth', 3);
legend([h1,h2,h3], {'f_{1}(t, a)', 'f_{2}(t, a)', "Line of Intersection"});
%exportgraphics(gca, "Q9a1.jpeg", "Resolution", 300)
hold off

plot3(tL, aL, fL, LineWidth=3)
ylabel("\alpha"); xlabel("time"); zlabel("function values");
grid on
%exportgraphics(gca, "Q9a12.jpeg", "Resolution", 300)
```

4. Discussion

In this homework I grasp the concepts on how to use basic root finding numerical methods and how to construct hybrid numerical method algorithms. In question one, it is asked to write a hybrid root finding algorithm. In question seven it is asked to write false-position and secant methods in MATLAB. In this question we are also asked to compare the methods. This comparison has a meaning because the secant method gave a negative number in one of the iterations where the function is not defined for negative numbers. So, this demonstrated the weakness of the method. In question nine, we have been asked to plot some given function and solve the problem graphically. It came out that the system has infinitely many solutions. Also, in this question we are asked to write a Newton Raphson code for the system of nonlinear equations. In this problem, since the problem has infinitely many solutions different initial guesses resulted in different results.

5. Conclusion

In conclusion, in this homework, we developed our understanding on numerical root finding methods and how to construct simple numerical root finding algorithms. In this homework I have especially developed my MATLAB skills on plotting. I have learned have to create three- and two-dimensional plots. In my case this homework was very helpful in enhancing my MATLAB skills. I also developed my understanding of the various numerical methods but in this homework the emphasis was on the MATLAB side. To conclude, this homework was very helpful in learning MATLAB and numerical methods.