EE 443 Numerical Methods and Introduction to Optimization

Fall 2022 Homework Assignment 5

Arda Turak

2232791

1)

```
% Arda TURAK 2232791

x = load('x.txt');                          % Load x from text files
y = load('y.txt');                          % Load y from text files

x_linearized = 1./x;                               % Linearize the relation for x
y_linearized = sqrt(1./y);                         % Linearize the relation for y

% with using Linear least squares algorithm we have
A_matrix = [ones(size(x_linearized)),x_linearized];     % calculate A_matrix
A_B = pinv(A_matrix) * y_linearized;                    % using pinv(A) function in MATLAB

%% PART A
A = A_B(1)                                              % print the value of A
B = A_B(2)                                              % print the value of B

%% PART B
scatter(x,y)                                       % with using scatter plot
hold on

%% PART C
final_value_of_y = (x.^2)./((A*x+B).^2);       % this is our function
plot(x,final_value_of_y,'LineWidth',2)         % plot our line
xlabel('x');                                   % xlable is x
ylabel('y');                                   % ylable is y
title('y vs x plot');                          % title of the plot
hold on
```

Figure 1

Figure 1 is a solution code for part a, b and c.

Figure 2 is the answer for A and B values.

A = 0.8037

B = 0.5902

```
>> hw5_part1_a

A =

    0.8037


B =

    0.5902

>>
```

Figure 2

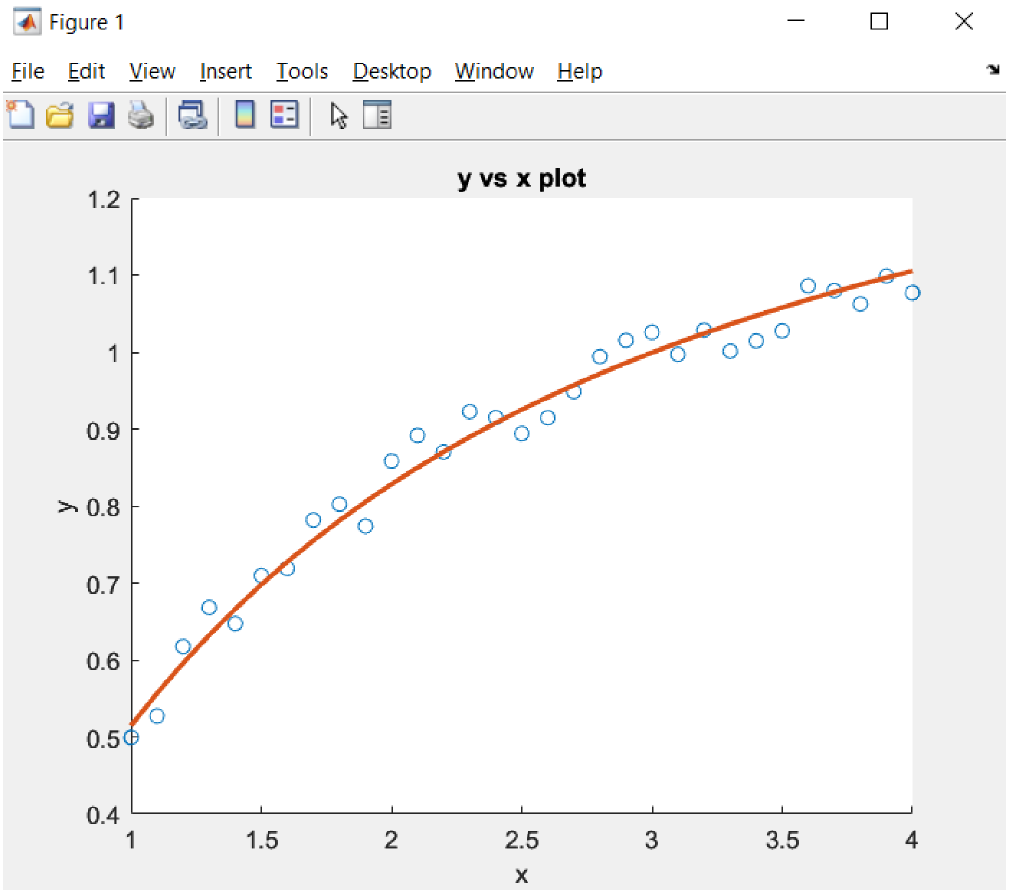

Figure 3

Figure 3 is the y vs x plot for the data given in text files and line in our plot. We can easily see that the line is fitted within the data points.

2)

```
hw5_part3_b.m  hw5_part1_a.m  hw5_part2_a.m  hw5_part2_c.m  +
```

```matlab
% Arda TURAK 2232791

%% PART A & B
func = @(A_B) sum((((x.^2)./(A_B(1)*x+A_B(2)).^2)-y).^2);     % our summation function with A, B, x, and y
A_B_initial = [0.1,0.1];                                          % Starting point
A_B = fminsearch(func,A_B_initial);                               % using fminsearch function we find A_B

A = A_B(1)                                               % print the value of A
B = A_B(2)                                               % print the value of B

final_value_of_y = (x.^2)./((A*x+B).^2);                 % this is our function
plot(x,final_value_of_y,'LineWidth',2)                   % plot our line
legend('calculated values')                              % legend of our plot
hold on
```

Figure 4

Figure 4 is the solution code for question 2 part a and b.

```
New to MATLAB? See resources for Getting Started.

>> hw5_part2_a

A =

    0.8126


B =

    0.5684

fx >> 
```

Figure 5

Figure 5 is the answer for A and B values.

A = 0.8126

B = 0.5684

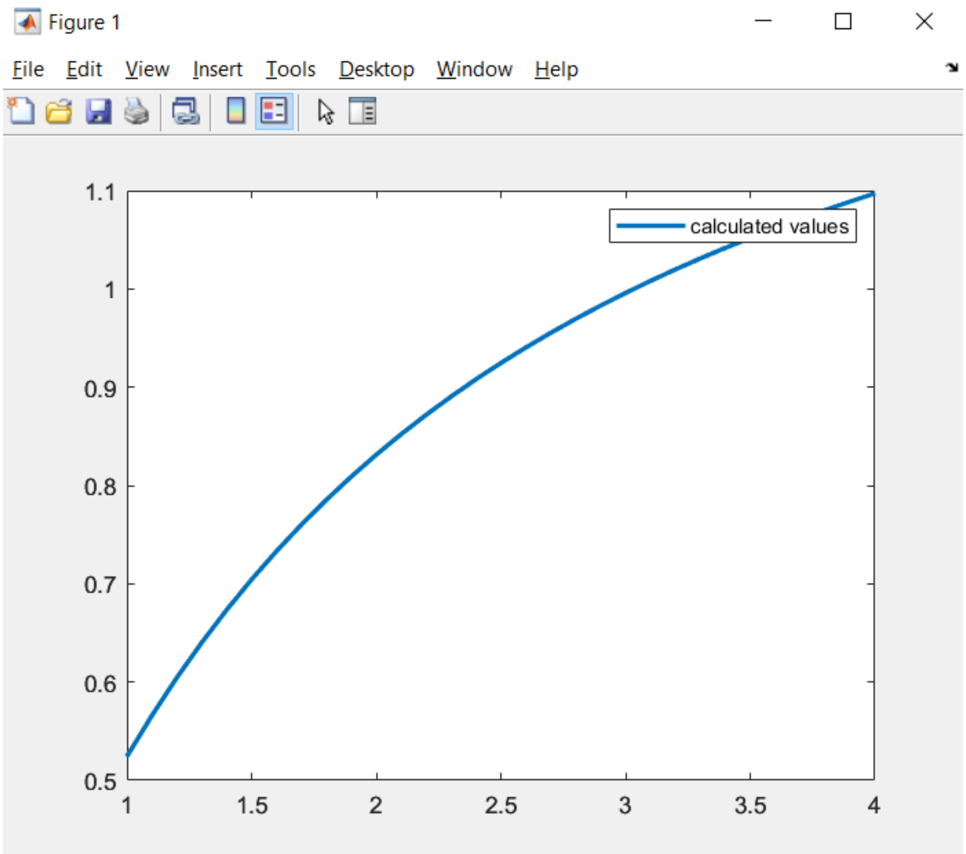

Figure 6

Figure 6 is the calculated values plot.

c)

I copy the code for question 1 to the end of question 2 and plot two graph at the same time with using hold on.

Figure 7 is the code for question 2 part c.

hw5_part3_b.m | hw5_part1_a.m | hw5_part2_a.m | hw5_part2_c.m

```matlab
% Arda TURAK 2232791

%% PART A & B
func = @(A_B) sum((((x.^2)./(A_B(1)*x+A_B(2)).^2)-y).^2);    % our summation function with A, B, x, and y
A_B_initial = [0.1,0.1];                                          % Starting point
A_B = fminsearch(func,A_B_initial);                               % using fminsearch function we find A_B

A = A_B(1)                                         % print the value of A
B = A_B(2)                                         % print the value of B

final_value_of_y = (x.^2)./((A*x+B).^2);           % this is our function
plot(x,final_value_of_y,'LineWidth',2)             % plot our line
legend('calculated values')                        % legend of our plot
hold on

%% FROM QUESTION 1
x = load('x.txt');                     % Load x from text files
y = load('y.txt');                     % Load y from text files

x_linearized = 1./x;                          % Linearize the relation for x
y_linearized = sqrt(1./y);                    % Linearize the relation for y

% with using Linear least squares algorithm we have
A_matrix = [ones(size(x_linearized)),x_linearized];   % calculate A_matrix
A_B = pinv(A_matrix) * y_linearized;                  % using pinv(A) function in MATLAB

%% PART A
A = A_B(1)                                         % print the value of A
B = A_B(2)                                         % print the value of B

%% PART B
scatter(x,y)                                      % with using scatter plot
hold on

%% PART C
final_value_of_y = (x.^2)./((A*x+B).^2);     % this is our function
plot(x,final_value_of_y,'LineWidth',2)       % plot our line
xlabel('x');                                 % xlable is x
ylabel('y');                                 % ylable is y
title('y vs x plot');                        % title of the plot
hold on
```

Figure 7

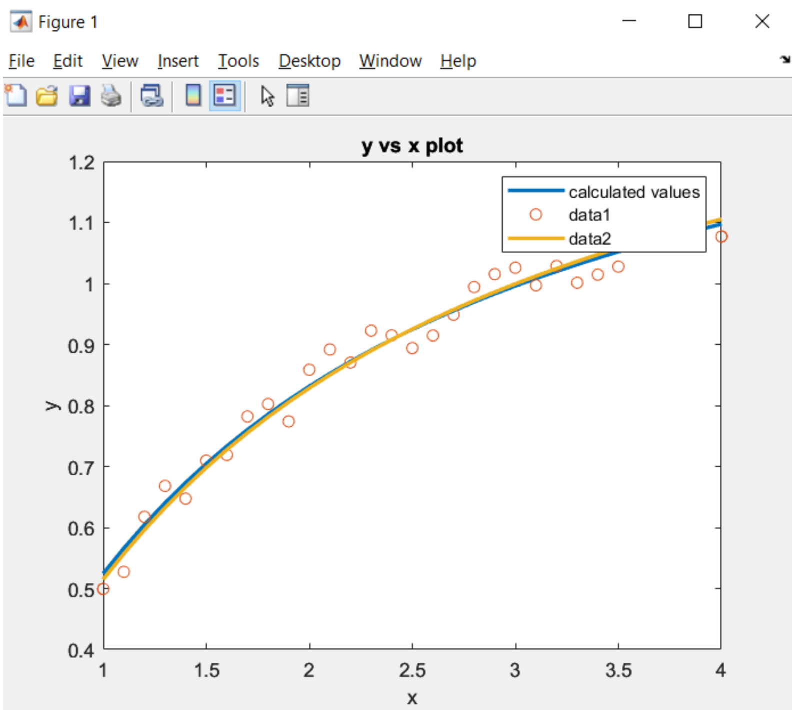

Figure 8

Figure 8 is the plot for 2 different graphs. One is calculated from a summation function and the other is calculated from already given data points. The two graphs are nearly the same with small changes. They have different values of A and B with minor changes and their slopes are nearly the same. We can easily see from the graph.

3)

a)

Q3

ARDA TURAK
2232791

a)

$$\nabla f = \begin{bmatrix} 2x+3y-4 \\ 10y+3x-5 \end{bmatrix}$$

$$\nabla g = \begin{bmatrix} -2.4 \\ -6.8 \end{bmatrix}$$ we have;

equations

① $2x+3y-4-2.4\mu_1 = 0$

② $10y+3x-5-6.8\mu_1 = 0$

③ $-6.8y-2.4x+8.96 = 0$

we have three equations and three unknowns.

$x = 2.6$
$y = 0.4$
$\mu_1 = 1$

At $(2.6, 0.4)$ point, the constraint is active since;

$\mu_1 \neq 0$ and also $(2.6, 0.4)$ gives us;

$-6.8(0.4) - 2.4(2.6) + 8.96 = 0 \leq 0$

$\Rightarrow$ active at the solution point.

Figure 9

Figure 9 is the solution of our constrained optimization problem.

We have found the solution point as (2.6 , 0.4)

b)

hw5_part3_b.m

```
% Arda TURAK 2232791

f = @(x,y) x^2 + 5*y^2 + 3*x*y - 4*x - 5*y;      % Define the objective function
g = @(x,y) -6.8*y - 2.4*x + 8.96;                % Define the inequality constraint


P1 = @(x) f(x(1),x(2)) + g(x(1),x(2))^2;         % Define the penalty function for r = 1
P10 = @(x) f(x(1),x(2)) + 10*g(x(1),x(2))^2;     % Define the penalty function for r = 10
P100 = @(x) f(x(1),x(2)) + 100*g(x(1),x(2))^2;   % Define the penalty function for r = 100

options = optimset('Display','iter');            % Set options for fminsearch

x1 = fminsearch(P1,[0.5,0.5],options);           % Minimize the penalty function for r = 1
x10 = fminsearch(P10,[0.5,0.5],options);         % Minimize the penalty function for r = 10
x100 = fminsearch(P100,[0.5,0.5],options);       % Minimize the penalty function for r = 100

% Print the solution
fprintf('r = 1: x = %f, y = %f\n', x1(1), x1(2));                % for x1
fprintf('r = 10: x = %f, y = %f\n', x10(1), x10(2));             % for x10
fprintf('r = 100: x = %f, y = %f\n', x100(1), x100(2));          % for x100
```

Figure 10

Figure 10 is the solution code for part b.

```
    45           85         -1.72249         contract inside
    46           87          -1.7225         contract inside
    47           89          -1.7225         contract inside
    48           91          -1.7225         contract inside
    49           93          -1.7225         contract inside
    50           95          -1.7225         reflect
    51           97          -1.7225         contract inside
    52           99          -1.7225         contract outside
    53          101          -1.7225         contract inside
    54          103          -1.7225         contract inside
    55          104          -1.7225         reflect
    56          106          -1.7225         contract inside
    57          108          -1.7225         contract inside

Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-04
 and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-04

r = 1: x = 2.568807, y = 0.344490
r = 10: x = 2.596591, y = 0.393925
r = 100: x = 2.599643, y = 0.399390
fx >>
```

Figure 11

Figure 11 is the answer for question 3 part b.

We have found that when r is increasing, we have a smaller error and we get closer to the right x and y values of the point as expected because r is the weighting coefficient.