Table of Contents

Lab 2: Exercises 1-6

by: Michelle Pichardo Munoz

```
% Best Practices -- I'll add it to every Excersise for clean up
close all; % closes all open windows
clear all; % deletes the workspace
```

Exercise 1

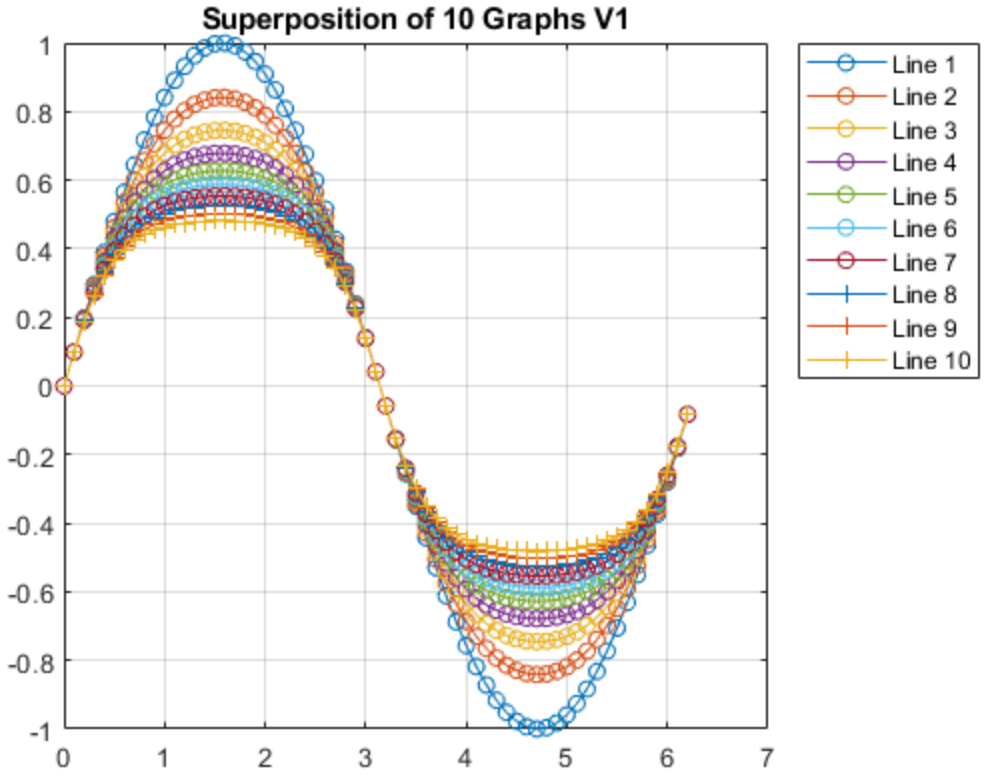
```
% Using a for-loop, calculate the summation of all the integers from 1-10
% i.e.: 1+2+3+4+5+6+7+8+9+10 = 55


% Best Practices
close all; % closes all open windows
clear all; % deletes the workspace


% create a variable to store the result :  set it to zero
sum = 0;

% for loop
%    where i [starts at 1, steps by 1, ends at 10]
%    i = [1,2,3,4,5,6,7,8,9,10]

for i = 1:1:10
    %   take our variable and add it by each i in the set
    sum = i + sum;
%    disp(i)  % uncomment to check CTL+SHIFT+R
end

% Display the final value of our variable
disp (sum)
```

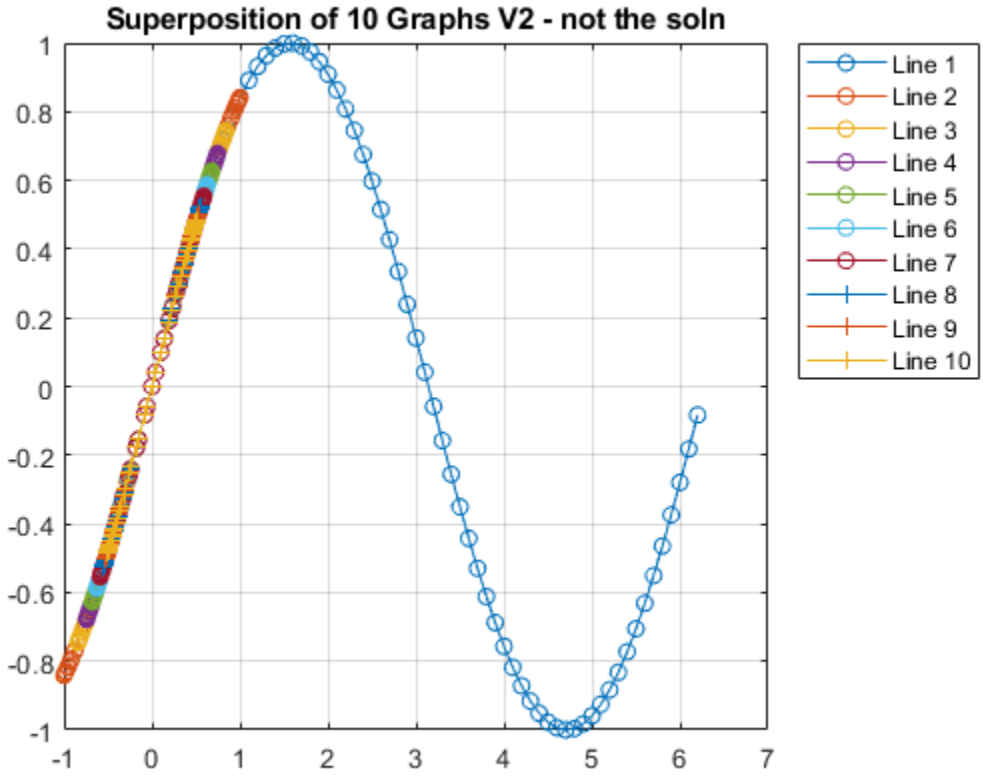
```
    55
```

Exercise 2

```
% Best Practices
close all; % closes all open windows
clear all; % deletes the workspace

% Create a vector x = 0:0.1:2*pi
x = 0:0.1:2*pi;                 % Domain of our functions

% Using a for-loop, generate 10 different plots on the same figure
%   where the curves are y1 = sin(x), y2 = sin(y1), y3 = sin(y2) ...


% Define the data matrix
%   matrix of row vecotors
data_matrix = [x ; sin(x)];

% Define the end condition
%   since we already have the first graph data filed we need 9 more
N = 9;

% Fill the matrix using a for-loop
for i = 1:1:N
    % Create the matrix holding all our data by incrimenting
    %   through the rows
    data_matrix(i+2,:) = sin(data_matrix(i+1,:));
end

% Graph of all fuctions on the x-domain ------------------------------

% Define the figure
fig = figure();
% Define the axes
ax = axes(fig);
% Difine markers
ax.LineStyleOrder = {'-o','-+','-*','-x','-s','-d','-v','->','-h','-^'};
% hold on: retain the current axes and properties
hold on
% add a box about the figure
box on
% add a grid for fun
grid on

% Start for-loop to plot we want 10, so N+1 = 9+1 =10
for i = 1:1:N+1
    %Plot the data by also incrimenting through the rows
    plot(data_matrix(1,:),data_matrix(i+1,:), ...
        'DisplayName', ['Line ', num2str(i)])
end

% add a legend outside of the main graph to reduce clutter
legend('Location','bestoutside')
% add a title
```

```
title('Superposition of 10 Graphs V1')

% hold off: reset any following plot to default properties
hold off

% I still think the wording of the problem needs to be adjusted:--------
%   This is the graph such that I plot:
%   x vs y1, y1 vs y2, and so on

% Define the figure
fig = figure();
% Define the axes
ax = axes(fig);
% Difine markers
ax.LineStyleOrder = {'-o','-+','-*','-x','-s','-d','-v','->','-h','-^'};
% hold on: retain the current axes and properties
hold on
% add a box about the figure
box on
% add a grid for fun
grid on

% Start for-loop to plot we want 10, so N+1 = 9+1 =10
for i = 1:1:N+1
    %Plot the data by also incrimenting through the rows
    plot(data_matrix(i,:),data_matrix(i+1,:), ...
        'DisplayName', ['Line ', num2str(i)])
end

% add a legend outside of the main graph to reduce clutter
legend('Location','bestoutside')
% add a title
title('Superposition of 10 Graphs V2 - not the soln')

% hold off: reset any following plot to default properties
hold off
```


Superposition of 10 Graphs V1
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10


Superposition of 10 Graphs V2 - not the soln
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10

Accident

```
     made a loop for several subplots
     I didn't want to delete it (saved for later)
data_matrix = [x ; sin(x)];
N = 10
for i = 1:1:N
   data_matrix(i+2,:) = sin(data_matrix(i+1,:));
   subplot(N,1,i);
   plot(data_matrix(i,:),data_matrix(i+1,:));
   title(sprintf('Graph of Plot %s', i));
end
```

Exercise 3

```
% Using an if/else statement
%   display one of three different messages depending on the value of the
%   variable grade.

%  If a grade is: >= 0.9  display "You aced the course."
%  If a grade is: 0.8>= grade <= 0.9 display "You almost aced the course."
%  If a grade is: <= 0.8 display "You didn't ace the course... nice try."


% Best Practices
close all; % closes all open windows
clear all; % deletes the workspace

grade = 1;

if grade >= 0.9
    fprintf('You aced the course.\n')
elseif grade > 0.8 && grade < 0.9
    fprintf('You almost aced the course.\n')
elseif grade <= 0.8
    fprintf("You didn't ace the course... nice try.\n")
end
```

You aced the course.

Exercise 4

```
% Useing a while-loop
%   divide 1_000 by 2 until the result is less than 1
%   Count and display on the workspace the total number of iterations

% Best Practices
close all; % closes all open windows
clear all; % deletes the workspace

% Declare our given value
N = 1000;
```

```
% Declare our starting iteration
iterations = 0;

% Define the while-loop
%    we divide by 2 until N is less than 1, then stop
while N > 1
    % Divide
    N = N/2;
    % Keep track of how many times the code is ran
    iterations = iterations + 1;
end
fprintf('Number of iterations= %.f\n', iterations)
fprintf('Final value of N from 1000 = %.4f\n', N)
% check the Workspace for the resulting values
```

```
Number of iterations= 10
Final value of N from 1000 = 0.9766
```

Exercise 5

```
% Program:
%   For any number between [-1,1] calculate and displays
%   the arccosine and arcsine of the number in both deg & radians
%   Output range: [0,180]deg , [0,pi]rad
% e.g.:
%       given 0, arccosine is either 90deg or pi/2 rad
%       and 270deg and 3pi/2 rad

% Best Practices
close all; % closes all open windows
clear all; % deletes the workspace

% define the number to examine
number_angle = 1;

% create if statement with conditionals
%      bounds: [-1,1]

if number_angle >= -1 && number_angle <= 1
    % Store the rad and degree values of the provided number
    x_rad = acos(number_angle);
    y_rad = asin(number_angle);
    x_deg = rad2deg(x_rad);
    y_deg = rad2deg(y_rad);

    % set of print statements returning the rad and deg
    fprintf("Arc-Values of given numeber: %.2f\n", number_angle)
    fprintf("Arcos(%.2f)= %.2f deg, Arcsin(%f) = %.2f deg\n", ...
        number_angle, x_deg, number_angle, y_deg)
    fprintf("Arcos(%.2f)= %.2f rad, Arcsin(%f) = %.2f rad\n", ...
        number_angle, x_rad, number_angle, y_rad)
else
```

```
    % Condition to tell the user they are out of bounds
    fprintf('Out of Bounds, select a number between [-1,1]\n')
end
```

```
Arc-Values of given numeber: 1.00
Arcos(1.00)= 0.00 deg, Arcsin(1.000000) = 90.00 deg
Arcos(1.00)= 0.00 rad, Arcsin(1.000000) = 1.57 rad
```

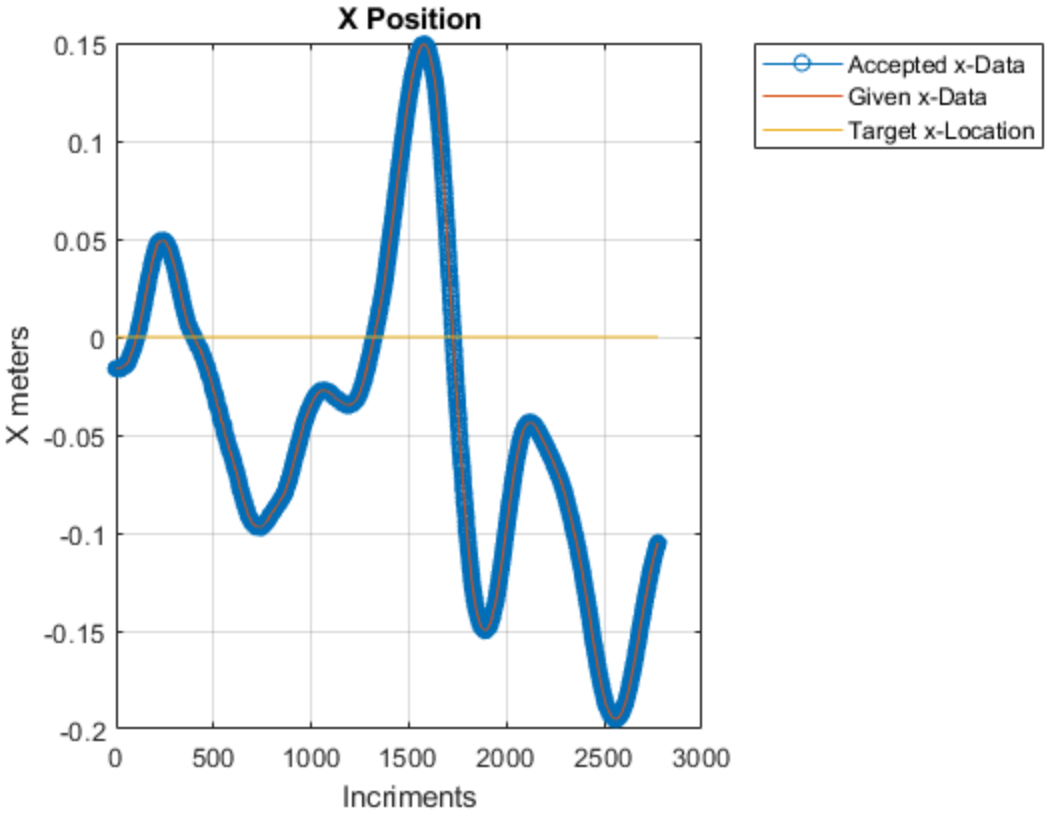
Exercise 6

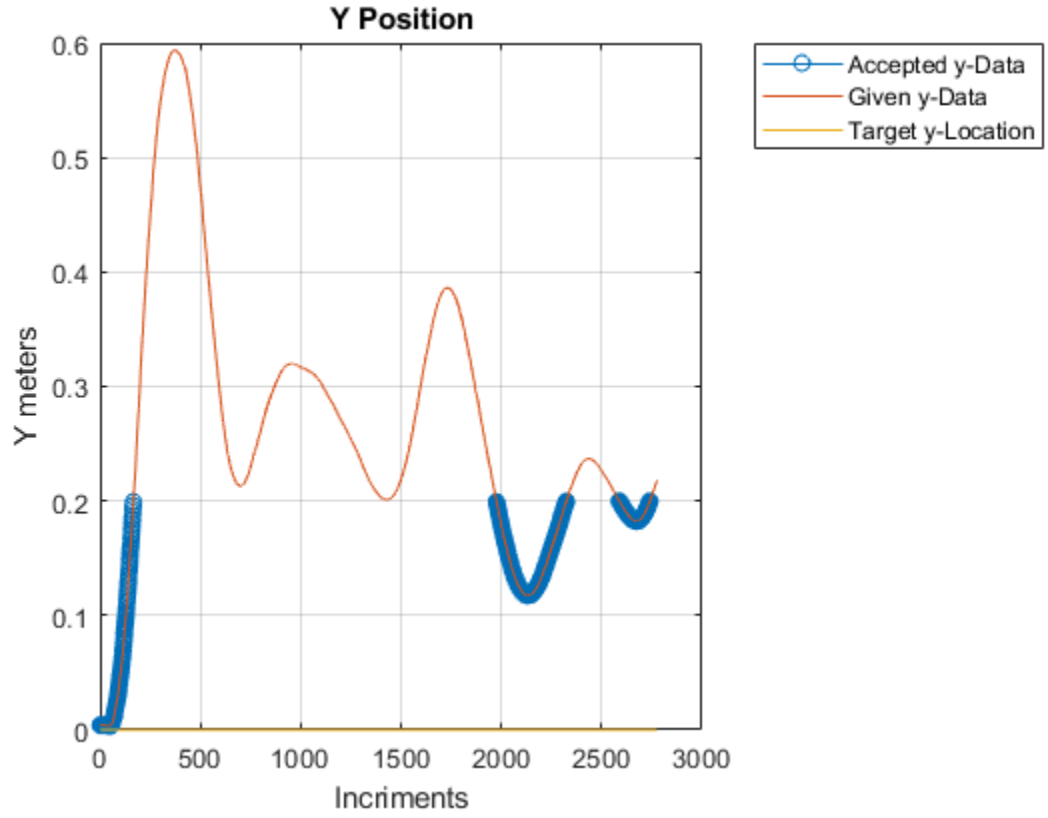
```
% Given the dataset Lab2_Excercise6.mat
%   - includes: x,y,z data of drone
%   - We are only interested in certain instances where the drone flies
%       close to the objective point (x: 0.0m, y:0.0 m, z:0.7m )

% Goal: filter the data & plot
%   - plot when the drone is within +/- 0.2m on all axes
% (a) Filter the xdata to only keep w/in +/- 0.2m
%       - plot the positions as well as the ref.line
% (b) Filter the ydata to only keep w/in +/- 0.2m
%       - plot the positions as well as the ref.line
% (c) Filter the zdata to only keep w/in +/- 0.2m
%       - plot the positions as well as the ref.line
% (d) Include legends, axes, and title for all 3 plots


% Best Practices
close all; % closes all open windows
clear all; % deletes the workspace

% Load in the variables from the .mat file
load Lab2_Exercise6.mat

% Set target parameters
xtarget = 0;
ytarget = 0;
ztarget = 0.7;

% Set the allowed error range
err = 0.2;
% ---------------------------------------- 6 a ----------------------
% Create a vector to store data
xnew = zeros(size(p_x(:,1)));

% Use a for-loop to select the desired points
%   i = [1:1:2775] i.e. p_x(:,1) 1st colum, entire row
for i = 1:length(p_x(:,1))
    % condition:
    %   if  0-002 <= entry <= 0+0.2  the code is:
    if p_x(i,1) >=  xtarget - err && p_x(i,1) <= xtarget + err
        % if true then let that entry replace the same position
        %   of the vector I called xnew
        xnew(i,1) = p_x(i,1);
```

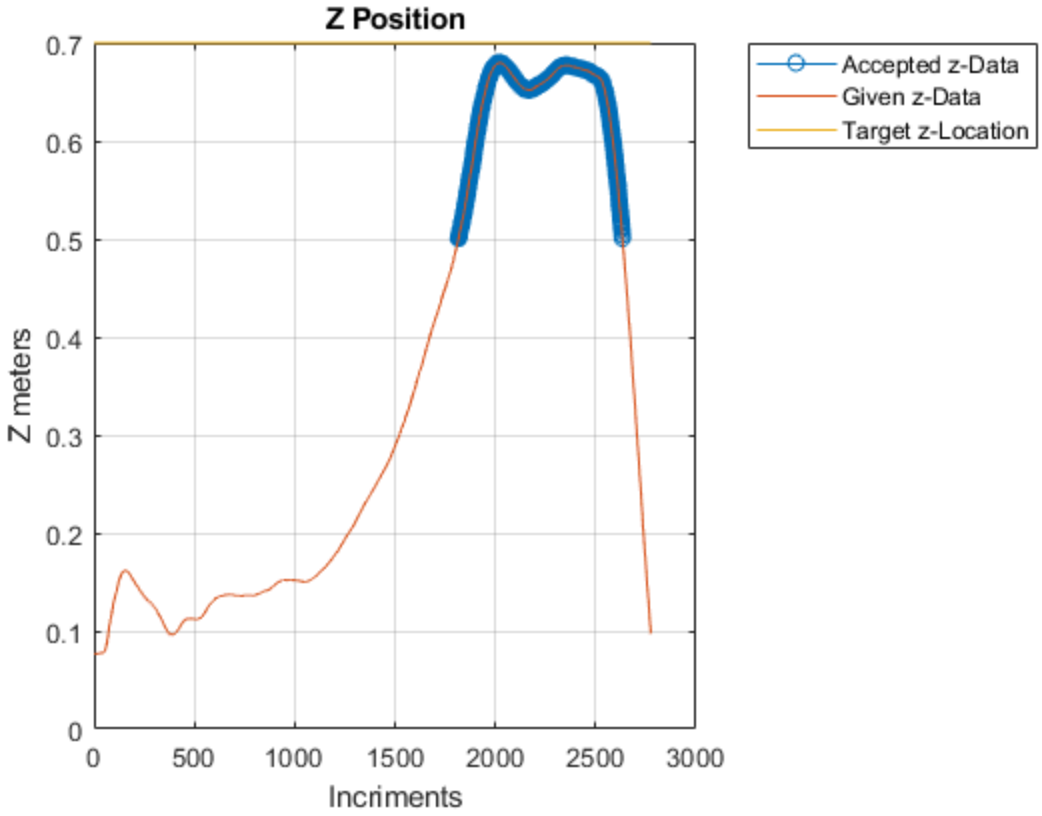
```
    else
        % if false then replace it with NaN (not a number)
        %   this lets me graph it the vecotor
        xnew(i,1) = NaN;
    end
end
figure(1)
hold on
% subplot(3,1,3) %-----------------------------------------
plot(xnew(:,1),'-o','DisplayName', 'Accepted x-Data')
ylabel('X meters')
xlabel('Incriments')
title('X Position')
legend('Location','bestoutside')
grid on
box on
% subplot(3,1,2) %-----------------------------------------
plot(p_x(:,1),'DisplayName', 'Given x-Data')
ylabel('X meters')
xlabel('Incriments')
title('X Position')
legend('Location','bestoutside')
grid on
box on
% subplot(3,1,1) %-----------------------------------------
plot(xtarget*ones(length(p_x),1), 'DisplayName','Target x-Location')
%   Note: created a vector of length p_x composed of 1's
%           multipied it by the constant xtarget to get a vector
%           of all xtarget (I could have used zeros function but meh)
ylabel('X meters')
xlabel('Incriments')
title('X Position')
legend('Location','bestoutside')
grid on
box on
hold off

% ---------------------------------------- 6 b ----------------------
% Note: the process is the same, comments are omitted

ynew = zeros(size(p_y(:,1)));

for i = 1:length(p_x(:,1))
    if p_y(i,1) >  ytarget - err && p_y(i,1) < ytarget + err
        ynew(i,1) = p_y(i,1);
    else
        ynew(i,1) = NaN;
    end
end

figure(2)
hold on
% subplot(3,1,3) %-----------------------------------------
plot(ynew(:,1), '-o','DisplayName', 'Accepted y-Data')
```

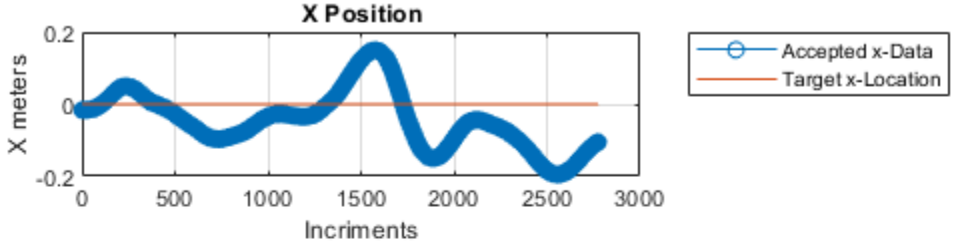
```
ylabel('Y meters')
xlabel('Incriments')
title('Y Position')
legend('Location','bestoutside')
grid on
box on
% subplot(3,1,2) %-----------------------------------------
plot(p_y(:,1),'DisplayName', 'Given y-Data')
ylabel('Y meters')
xlabel('Incriments')
title('Y Position')
legend('Location','bestoutside')
grid on
box on
% subplot(3,1,1) %-----------------------------------------
plot(ytarget*ones(length(p_y),1), 'DisplayName','Target y-Location')
ylabel('Y meters')
xlabel('Incriments')
title('Y Position')
legend('Location','bestoutside')
grid on
box on
hold off

% ---------------------------------------- 6 c ----------------------
% Note: the process is the same, comments are omitted


znew = zeros(size(p_z(:,1)));

for i = 1:length(p_z(:,1))
    if p_z(i,1) >  ztarget - err && p_z(i,1) < ztarget + err
        znew(i,1) = p_z(i,1);
    else
        znew(i,1) = NaN;
    end
end


figure(3)
hold on
% subplot(3,1,3) %-----------------------------------------
plot(znew(:,1),'-o', 'DisplayName', 'Accepted z-Data')
ylabel('Z meters')
xlabel('Incriments')
title('Z Position')
legend('Location','bestoutside')
grid on
box on
% subplot(3,1,2) %-----------------------------------------
plot(p_z(:,1),'DisplayName', 'Given z-Data')
ylabel('Z meters')
xlabel('Incriments')
title('Z Position')
```

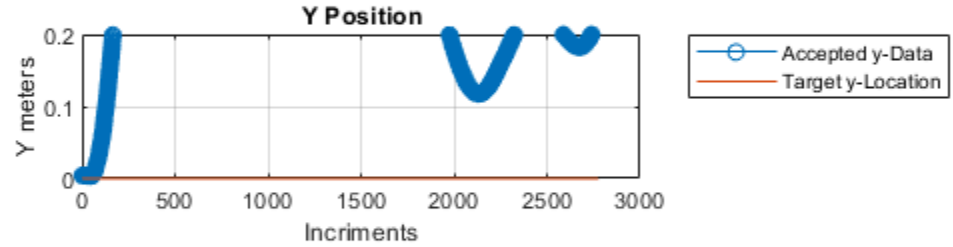
```
legend('Location','bestoutside')
grid on
box on
% subplot(3,1,1) %-----------------------------------------
plot(ztarget*ones(length(p_z),1), 'DisplayName','Target z-Location')
ylabel('Z meters')
xlabel('Incriments')
title('Z Position')
legend('Location','bestoutside')
grid on
box on
hold off

% ---------------------------------------- 6 d ----------------------
% I don't understand what this wants, and I won't assume 3D graphing
figure(4)
subplot(3,1,1)
plot(xnew(:,1),'-o','DisplayName', 'Accepted x-Data')
hold on
plot(xtarget*ones(length(p_x),1), 'DisplayName','Target x-Location')
ylabel('X meters')
xlabel('Incriments')
title('X Position')
legend('Location','bestoutside')
grid on
box on
subplot(3,1,2)
plot(ynew(:,1), '-o','DisplayName', 'Accepted y-Data')
hold on
plot(ytarget*ones(length(p_y),1), 'DisplayName','Target y-Location')
ylabel('Y meters')
xlabel('Incriments')
title('Y Position')
legend('Location','bestoutside')
grid on
box on
subplot(3,1,3)
plot(znew(:,1),'-o', 'DisplayName', 'Accepted z-Data')
hold on
plot(ztarget*ones(length(p_z),1), 'DisplayName','Target z-Location')
ylabel('Z meters')
xlabel('Incriments')
title('Z Position')
legend('Location','bestoutside')
grid on
box on
```

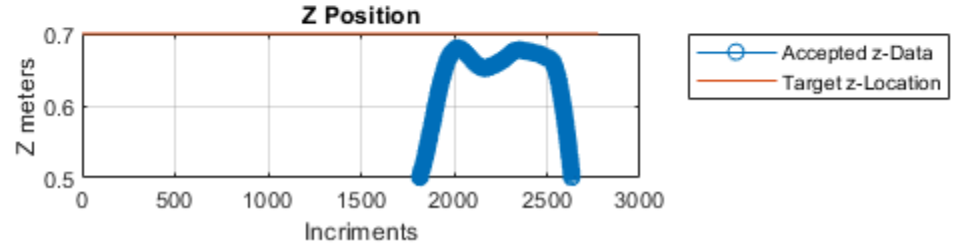

X Position
Accepted x-Data
Given x-Data
Target x-Location
X meters
Incriments
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0
500
1000
1500
2000
2500
3000


Y Position
Accepted y-Data
Given y-Data
Target y-Location
Y meters
Incriments
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
2500
3000


Z Position
Z meters
Incriments
Accepted z-Data
Given z-Data
Target z-Location


X Position
X meters
Incriments
Accepted x-Data
Target x-Location


Y Position
Y meters
Incriments
Accepted y-Data
Target y-Location


Z Position
Z meters
Incriments
Accepted z-Data
Target z-Location

Published with MATLAB® R2022b