Lab #3 - CIEG 675/010

Chrysostomos Karakasis

Due date: January 26, 2021

```
%% CIEG 675 LAB#3 Due Tuesday January 26, 2021
%% Author: Chrysostomos Karakasis 702529334
close all;
clear all;
% Functions are located in the same folder as this file
```

1 Part (1)

Write a function that will take an arbitrary, but smooth, vector of data and locate ALL the local maxima and minima in the vector (call it *min_max*, as *minmax* is a built-in function in newer MATLAB versions starting with R2014b). It should return, as output, the indices of the input vector where these maxima and minima occur. Test your function on the following data set that you should make quite smooth by using small increments in x, where x should go from 0 to 10:

$$y = x^{1.01} + 4cos(3\pi x/4) - 2sin(2\pi x/3) - 0.25 \tag{1.1}$$

In the same figure, plot y vs x, with the maxima and minima (computed by your function, *min_max*) on top of the curve as different symbols (remember you can use $\ll$ *help plot* to see a list of available linemarker specifiers). DO NOT USE a built in function to find local minima or maxima. I want you to write the code to do it.

```
%% Part (1)
% Write a function that will take an arbitrary, but smooth, vector of data
% and locate ALL the local maxima and minima in the vector (call it min_max,
% as "minmax" is a built-in function in newer MATLAB versions starting with
% R2014b). It should return, as output, the indices of the input vector
% where these maxima and minima occur. Test your function on the following
% data set that you should make quite smooth by using small increments in
% x, where x should go from 0 to 10:
% y = x^{1.01}+4*cos(3*pi*x/4)-2*sin(2*pi*x/3)-0.25
% In the same figure, plot y vs x, with the maxima and minima (computed by
% your function, min_max) on top of the curve as different symbols
% (remember you can use >> help plotto see a list of available line/marker
% specifiers). DO NOT USE a built in function to find local minima or
% maxima. I want you to write the code to do it.

x = 0:0.01:10; % Create the x-vector with a small enough increment
y = x.^(1.01)+4*cos(3*pi*x/4)-2*sin(2*pi*x/3)-0.25; % Calculate the smooth dataset that will be used ...
    as a testing input for our function
[min_ind,max_ind] = min_max(y) % Call the custom-made 'min_max' to get the locations of the local ...
    extrema using the first-derivative test
fig1 = figure(1); % Open a new figure
hold on; % Use "hold on" to plot all three parabolas on the same plot
plot(x,y,'LineWidth',2) % Plot the y vs x data
plot(x(min_ind),y(min_ind),'ro','LineWidth',2) % Plot the local minima using a red 'o' symbol
plot(x(max_ind),y(max_ind),'mx','LineWidth',2) % Plot the local maxima using a magenta 'x' symbol
% Create a new legend specifying everything on the figure
legend('Arbitrary-smooth Vector of Data','Local Minima','Local ...
    Maxima','interpreter','latex','location','southeast','FontSize',11)
title('LAB \#3 - Part (1) - Min\_max','FontSize',14,'interpreter','latex') % Create a title for the ...
    figure
xlabel('X-axis','FontSize',14,'interpreter','latex') % Set a label for the x-axis
ylabel('Y-axis','FontSize',14,'interpreter','latex') % Set a label for the y-axis
axis([0 10 -10 14]) % Increase axis limits
print(fig1,'lab_3_part_1','-depsc','-r600'); % Save figure in a colored .eps format using 600 dpi ...
    resolution
```

```
function [min_ind,max_ind] = min_max(y)
% function [min_ind,max_ind] = min_max(y)

% Description: This function receives a smooth vector of data as an input.
% Then the first derivative test is utilized to calculate the local extrema
% of the vector. Finally, the function returns the indices of the input
% vector where these maxima and minima occur.

% Since the input vector of data is smooth, we can differentiate it
y_div = diff(y); %Since we not dividing by time (dt), it is not a derivative, but this vector has the ...
    same sign as the derivative

min_ind = []; % Initialize a vector that will contain the indices of the local minima
```

```
max_ind = []; % Initialize a vector that will contain the indices of the local maxima
for i = 2:length(y_div) % Iterate through all elements of the "derivative" vector
    if y_div(i) > 0 && y_div(i-1) < 0       % A sign change from positive to negative in the ...
        "derivative" indicates a local minimum
        min_ind = [min_ind i];              % Append the vector with all local minima with the new one
    elseif y_div(i) < 0 && y_div(i-1) > 0   % A sign change from negative to positive in the ...
        "derivative" indicates a local maximum
        max_ind = [max_ind i];              % Append the vector with all local maxima with the new one
    end
end
end
```

Command Window Output

```
min_ind =

   115   391   667   942


max_ind =

   260   535   811
```

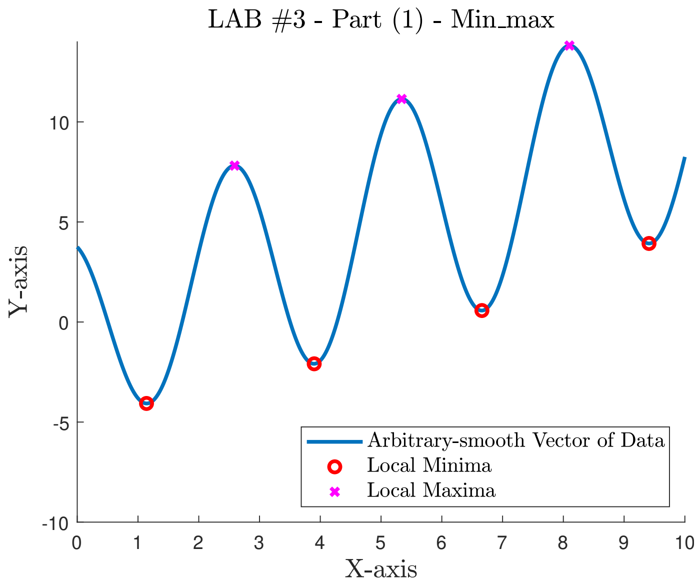

Figure 1: Lab # 3 - Part (1) - Example of the *min_max* function.

2 Part (2)

Create a 1 x 2 structure array with 3 fields of your choosing. The only stipulation is that the 1st field must be a string (class: char), the 2nd field must be a cell array (class: cell), and the 3rd field should be an array of numeric values (class: double).

```
%% Part (2)
% Create a 1 x 2 structure array with 3 fields of your choosing. The only
% stipulation is that the 1st field must be a string (class: char), the
% 2nd field must be a cell array (class: cell), and the 3rd field should be
% an array of numeric values (class: double).

D(1,1).field1 = 'hey'; % Define the first field of the first element of the struct structure ...
    (char:string)
D(1,1).field2 = {2}; % Define the second field of the first element of the struct structure (cell)
D(1,1).field3 = [3.24:0.1:5.46]; % Define the third field of the first element of the struct ...
    structure (array of doubles)

D(1,2).field1 = 'there'; % Define the first field of the second element of the struct structure ...
    (char:string)
D(1,2).field2 = {4}; % Define the second field of the second element of the struct structure (cell)
D(1,2).field3 = [5.46:-0.1:3.24]; % Define the third field of the second element of the struct ...
    structure (array of doubles)
```

```
D % Show that D is a 1x2 structure array
class(D(1).field1) % Show that the 1st field is indeed a string (class: char)
class(D(1).field2) % Show that the 2st field is indeed a cell array (class: cell)
class(D(1).field3) % Show that the 1st field is indeed an array of doubles (class: double)
```

Command Window Output

```
D =

  1x2 struct array with fields:

    field1
    field2
    field3


ans =

    'char'


ans =

    'cell'


ans =

    'double'
```

3 Part (3)

Create a 12 x 1 cell array whose contents are the twelve months of the year, in order, starting with 'January'.

```
% Create a 12 x 1 cell array whose contents are the twelve months of the year, in order, starting ...
    with 'January'.
c_a_3 = {'January','February','March','April','May','June','July','August',...
    'September','October','November','December'}
```

Command Window Output

```
c_a_3 =

  1x12 cell array

  Columns 1 through 11

    {'January'}    {'February'}    {'March'}    {'April'}    {'May'}    {'June'}    {'July'}    ...
        {'August'}    {'September'}    {'October'}    {'November'}

  Column 12

    {'December'}
```

4 Part (4)

Write a function called day2secthat takes an input vector of MATLAB dates and converts it first to time relative to the 1st date value in the input vector (i.e. the first value in the output vector should be zero) and then from days to seconds. The output vector should be the number of seconds that have elapsed since the 1st value in the input vector.

```
%% Part (4)
% Write a function called day2secthat takes an input vector of MATLAB dates
% and converts it first to time relative to the 1st date value in the input
% vector (i.e. the first value in the output vector should be zero) and
```

```matlab
% then from days to seconds. The output vector should be the number of
% seconds that have elapsed since the 1st value in the input vector.

% First, we will create an array with four arbitrary dates in an ascending
% order and then we will convert them to MATLAB dates using the 'datenum'
% built-in function
mat_dates_v = [datenum(1973,1,30,15,32,11) datenum(1973,1,30,15,32,31) datenum(1995,10,6,12,30,24) ...
    datenum(2021,1,21,13,18,46)];
num_sec = day2secthat(mat_dates_v) % Call the 'day2secthat' function

% Two versions of the function 'day2secthat' were created
num_sec_2 = day2secthat2(mat_dates_v) % Call the 'day2secthat2' function
```

```matlab
function [num_sec] = day2secthat(mat_dates_v)
% function [num_sec] = day2secthat(mat_dates_v)

% Description: This function receives as an input a vector of MATLAB dates
% and returns as an output a vector containing the number of seconds that
% have elapsed since the 1st value (date) in the input vector

mat2str = datestr(mat_dates_v); % Convert the input vector to string format
str2vec = datevec(mat2str); % Convert the string format to a vector of components
num_sec = []; % Initialize the output vector that will contain the elapsed number of seconds
for i = 1:size(str2vec,1) % Iterate through all dates (elements) of the input vector
    % Calculate the elapsed number of seconds of each date wrt to the
    % first one and add it to the output vector.
    num_sec = [num_sec etime(str2vec(i,:),str2vec(1,:))];
end
end
```

```matlab
function [num_sec] = day2secthat2(mat_dates_v)
% function [num_sec] = day2secthat2(mat_dates_v)

% Description: This function receives as an input a vector of MATLAB dates
% and returns as an output a vector containing the number of seconds that
% have elapsed since the 1st value (date) in the input vector. This a
% second simpler version of the "day2secthat" function

% Calculate elapsed days of all elements wrt to the first date in the input
days_elapsed = mat_dates_v-mat_dates_v(1);

% Calculate elapsed hours of all elements wrt to the first date in the input
hours_elapsed = days_elapsed*24; % Each day has 24 hours

% Calculate elapsed minutes of all elements wrt to the first date in the input
minutes_elapsed = hours_elapsed*60; % Each hour has 60 minutes

% Calculate elapsed seconds of all elements wrt to the first date in the input
num_sec = minutes_elapsed*60; % Each minute has 60 seconds
end
```

Command Window Output

```
num_sec =

   1.0e+09 *

                    0   0.000000020000000   0.715726693000000   1.513979195000000


num_sec_2 =

   1.0e+09 *

                    0   0.000000020000002   0.715726693000007   1.513979195000008
```

5 Part (5)

Download the file from Canvas called "*AtlanticCity_TemperatureData.csv*". Load the data into MATLAB using *dlmread* (or some other function). The 1st column contains dates in MATLAB time. The 2nd column is air temperature and the 3rd column is water temperature, both in degrees Celsius. The data span all of 2015. Subtract the first date value from all other dates, then plot air temperature and water temperature versus time, on the same axes. Add a legend, labels, etc. Finally, change the ‘xticklabel’ values of the current axes to be your cell array from Problem 3, where the ‘xtick’ locations should be the number of days after January 1st, for each first day of the month (i.e. [1, 32, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335]). Resize the figure to be wider, so that all the month labels on the x-axis are not overlapping.

```
%% Part (5)
% Download the file from Canvas called "AtlanticCity_TemperatureData.csv".
% Load the data into MATLAB using dlmread(or some other function). The 1st
% column contains dates in MATLAB time. The 2nd column is air temperature
% and the 3rd column is water temperature, both in degrees Celsius. The data
% span all of 2015. Subtract the first date value from all other dates,
% then plot air temperature and water temperature versus time, on the same
% axes. Add a legend, labels, etc. Finally, change the 'xticklabel' values
% of the current axes to be your cell array from Problem 3, where the
% 'xtick' locations should be the number of days after January 1st, for each
% first day of the month (i.e. [1, 32, 60, 91, 121, 152, 182, 213, 244, 274,
% 305, 335]). Resize the figure to be wider, so that all the month labels
% on the x-axis are not overlapping.

% Load the data into MATLAB using dlmread
atl_temp_data = dlmread('AtlanticCity_TemperatureData.csv');

% Create a copy of the loaded data
atl_temp_data_norm = atl_temp_data;
% Subtract the first date value from all other dates
atl_temp_data_norm(:,1) = atl_temp_data_norm(:,1) - atl_temp_data_norm(1,1);

% Plot air temperature and water temperature versus time, on the same axes
fig2 = figure(2);
hold on; % Use "hold on" to plot all three parabolas on the same plot
plot(atl_temp_data_norm(:,1),atl_temp_data_norm(:,2),'LineWidth',1); % Plot the air temparature data ...
    wrt time
plot(atl_temp_data_norm(:,1),atl_temp_data_norm(:,3),'LineWidth',1); % Plot the water temparature ...
    data wrt time
% Create a new legend specifying everything on the figure
legend('Air Temperature ($C^{\circ}$)','Water Temperature ...
    ($C^{\circ}$)','interpreter','latex','FontSize',14,'location','southeast')
title('LAB \#3 - Part (5) - Atlantic City Temperature Data','FontSize',14,'interpreter','latex') % ...
    Create a title for the figure
xlabel('X-axis','FontSize',14,'interpreter','latex') % Set a label for the x-axis
ylabel('Y-axis','FontSize',14,'interpreter','latex') % Set a label for the y-axis

% In order to isolate each first day of the month. We assume that for every
% first day of the month, the first entry data will be utilized.
temp = datestr(atl_temp_data(:,1)); % Convert the MATLAB dates back to string format
first_days_per_month = []; % Initialize the array that will store the indices depicting the first ...
    entry for the first day of every month

for i = 1:size(atl_temp_data,1) % Iterate through all dates
    if temp(i,1:2)=='01' % Check whether an element corresponds to the first day of the month
        if (temp(i,end-7:end)=='00:00:00') % Check whether an element also corresponds to the first ...
            entry of the day
            first_days_per_month = [first_days_per_month i]; % When a first-entry/first-day-month is ...
                found, append it to the existing vector
        end
    end
end

% Change the 'xticks' locations to the number of days after January 1st, for each
% first day of the month.
xticks(atl_temp_data_norm(first_days_per_month,1))

% Change the 'xticklabel' values of the current axes to be your cell array from Problem 3
xticklabels(c_a_3)

% Resize the figure to be wider, so that all the month labels
% on the x-axis are not overlapping.
set(gcf,'units','normalized') % So that it does not depend on the resolution of the user's monitor
set(gcf,'position',[0.1411 0.2843 0.7641 0.3889])
print(fig2,'lab_3_part_5','-depsc','-r600'); % Save figure in a colored .eps format using 600 dpi ...
    resolution
```

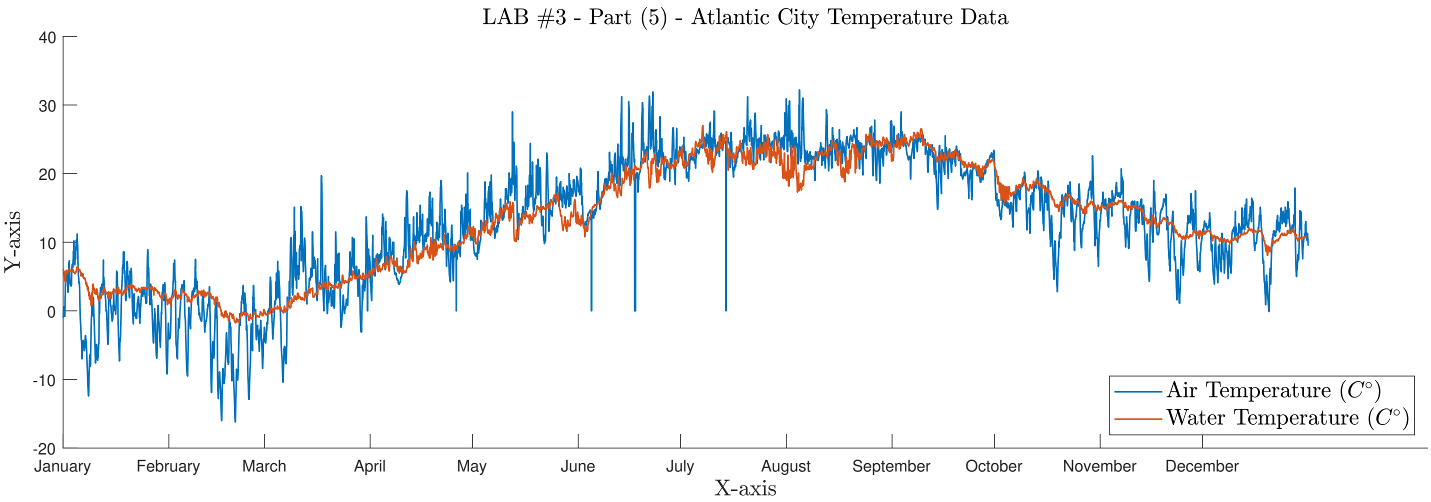

Figure 2: Lab # 3 - Part (5) - Plotting of Atlantic City Temperature Data.

6 Part (6)

Download the file on Canvas called "surprise.txt", which contains multiple lines of text, each with a different number of characters. First, using the method outlined above, load each line of the file into a cell array in MATLAB.

```
%% Part (6)
% Download the file on Canvas called "surprise.txt", which contains
% multiple lines of text, each with a different number of characters.
% First, using the method outlined above, load each line of the file into a cell array in MATLAB.

ct = 0;  % a counter
fid = fopen('surprise.txt', 'r'); % open file for reading
% Keep running a loop until EndOfFile
%   (feof(fid) = 1 if end of the file is reached)
while feof(fid) ≠ 1
    ct = ct + 1; % increase counter by 1
    data{ct} = fgetl(fid); % grab entire line of text from file,store as string in the ct^th cell
end
fclose(fid); % close file
%
% ct should be equal to the number of lines of text in the file

% Then, write a code (it shouldn't be long...mine is only 7 lines) that
% will print each line of text into the command window (fprintf), one
% character at a time, and after each full line has been printed, it should
% then print the newline character combination in MATLAB, fprintf('\n'), in
% order to jump to the next line.

for i = 1:ct % Iterate through all lines of text in the file
    for j = 1:length(data{i}) % Iterate through all characters of each line
        fprintf(data{i}(j)) % Print one character at a time
    end
    fprintf('\n') % At this point a full line has been printed, hence the newline character is printed
end
```

Command Window Output

```
                                  _______
                           _,,ad8888888888bba,_
                        ,ad88888I888888888888888ba,
                      ,88888888I88888888888888888888a,
                    ,d888888888I8888888888888888888888b,
                   d88888PP"""" ""YY88888888888888888888b,
                 ,d88"'__,,--------,,,,.;ZZZY8888888888888,
                ,8IIl'"               ;;l"ZZZIII8888888888,
               ,I88l;'                 ;lZZZZZ888III8888888,
             ,II88Zl;.                 ;llZZZZZ888888I888888,
            ,II888Zl;.               .;;;;;lllZZZ888888I8888b
           ,II8888Z;;                `;;;;;''llZZ8888888I8888,
           II88888Z;'                       .;lZZZ8888888I888b
           II88888Z; _,aaa,      .,aaaaa,__.l;llZZZ88888888I888
           II88888IZZZZZZZZZ,  .ZZZZZZZZZZZZZZ;llZZ88888888I888,
           II88888IZZ<'(@@>Z|  |ZZZ<'(@@>ZZZZ;;llZZ888888888I88I
          ,II88888;   `""" ;|  |ZZ; `"""     ;;llZ888888888I888
          II888888l            `;;          .;llZZ8888888888I888,
         ,II888888Z;           ;;;        .;;llZZZ8888888888I888I
         III888888Zl;    ..,   `;;       ,;;lllZZZ8888888888I888
         II88888888Z;;...;(_    _)      ,;;;llZZZZ8888888888I888,
         II88888888Zl;;;;;' `--'Z;.   .,;;;;llZZZZ88888888888I888b
         ]I888888888Z;;;;'   ";llllll;..;;;lllZZZ88888888888I8888,
         II888888888Zl.;;"Y88bd888P";;,..;lllZZZZ8888888888I8888I
         II8888888888Zl;.; `"PPP";;;,..;lllZZZZZZZ88888888888I88888
         II88888888888Zl;;. `;;;l;;;;lllZZZZZZZZW8888888888I88888
         `II88888888888Zl;.    ,;;lllZZZZZZZZWMZ8888888888I88888
          II8888888888888888ZbaalllZZZZZZZZZWWMZZZ8888888888I888888,
          `II8888888888888888b"WWZZZZZWWWMMZZZZZZI888888888I888888b
           `II8888888888888888;ZZMMMMMMZZZZZZZZllI888888888I8888888
            `II88888888888888 `;lZZZZZZZZZZllll888888888I8888888,
             II88888888888888, `;lllZZZZllllll;;.Y8888888I8888888b,
            ,II888888888888888b   .;;lllllll;;;.;..8888888I8888888b,
            II88888888888888PZI;.  .`;;;.;;;..; ...8888888I888888888,
            II8888888888PZ;;';;.   ;. .;.  .;. .. Y8888888I8888888888b,
           ,II88888888PZ;;'                        `8888888I88888888888b,
           II88888888'                              888888I8888888888888888b
          ,II88888888                              ,888888I8888888888888888
         ,d8888888888                              d888888I888888888ZZZZZZZ
      ,ad88888888888I                              8888888I8888ZZZZZZZZZZZZ
    ,d88888888888888'                              888888IZZZZZZZZZZZZZZZZZ
  ,d888888888888P'8P'                              Y888ZZZZZZZZZZZZZZZZZZZZ
 ,8888888888888,  "                                ,ZZZZZZZZZZZZZZZZZZZZZZZZ
d8888888888888,                                 ,ZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
8888888888888888a,      _                     ,ZZZZZZZZZZZZZZZZZZZZZZ88888888
88888888888888888888ba,_d'                  ,ZZZZZZZZZZZZZZZZZZZ88888888888888
8888888888888888888888888888bbbaaa,,,_______,ZZZZZZZZZZZZZZZ88888888888888888
88888888888888888888888888888888888888888ZZZZZZZZZZZZZZZ888888888888888888888
8888888888888888888888888888888888888888ZZZZZZZZZZZZZZZ8888888888888888888888
888888888888888888888888888888888888888ZZZZZZZZZZZZZZ888888888888888888888888
8888888888888888888888888888888888888ZZZZZZZZZZZZZZ88888888888888888888888888
88888888888888888888888888888888888ZZZZZZZZZZZZZZ8888888888888888888888888888
8888888888888888888888888888888888ZZZZZZZZZZZZZZ88888888888888888888888888888
88888888888888888888888888888888ZZZZZZZZZZZZZZ8888888888888888888888888888888
8888888888888888888888888888888ZZZZZZZZZZZZZZ88888888888888888888888888888888
```

7 Part (7)

Develop a data set that is 10,000 points long composed of random numbers from a normal distribution. Make a histogram of this data set and then overlay a Gaussian curve on top. You will have to determine how to normalize your plot or histogram to get the y-axis to scale appropriately.

```
%% Part (7)
% Develop a data set that is 10,000 points long composed of random
% numbers from a normal distribution.  Make a histogram of this data set
% and then overlay a Gaussian curve on top.  You will have to determine how
% to normalize your plot or histogram to get the y-axis to scale appropriately.

rand_data_7 = randn(10000,1); % Create the 10,000 points long vector composed of random numbers from ... a normal distribution
fig3 = figure(3); % Open a new figure for plotting
pd_7 = fitdist(rand_data_7,'Normal'); % Fit a normal distribution to the generated data
```

```
x_values_7 = pd_7.mu-4*pd_7.sigma:0.01:pd_7.mu+4*pd_7.sigma; % Calculate x-coordinates of gaussian ...
    curve from -4/+4 standard deviations
y_7 = pdf(pd_7,x_values_7); % Calculate corresponding y-coordinates of fitted curve
yyaxis right % Utilize the right vertical side as a new axis for the PDF
plot(x_values_7,y_7,'Linewidth',2); % Plot the Power Density Function of the fitted normal distribution
set(gca,'YLim',1.5*get(gca,'YLim')) % Increase the limits of the y-axis to avoid overlapping between ...
    lines and the legends

yyaxis left % Utilize the left vertical side as a new axis for the histogram
histogram(rand_data_7) % Create a histogram of the generated data, showing the frequency of each ...
    element inside the vector
set(gca,'YLim',1.5*get(gca,'YLim')) % Increase the limits of the y-axis to avoid overlapping between ...
    lines and the legends

% Create a new legend specifying everything on the figure
legend({' Frequency',[' PDF of Fitted Normal Distribution\newline Mean: ',num2str(pd_7.mu),' - SD: ...
    ',num2str(pd_7.sigma)]})
xlabel('Numbers in the Random Dataset'); % Add a label for the x-axis
yyaxis right % Switch back to the right y-axis
ylabel('Normalized Power Density Function'); % Add a label for the right y-axis
yyaxis left % Switch back to the left y-axis
ylabel('Frequency'); % Add a label for the left y-axis
title('LAB \#3 - Part (7) - Histogram - Normal Distribution','FontSize',14,'interpreter','latex') % ...
    Create a title for the figure
print(fig3,'lab_3_part_7','-depsc','-r600'); % Save figure in a colored .eps format using 600 dpi ...
    resolution
```

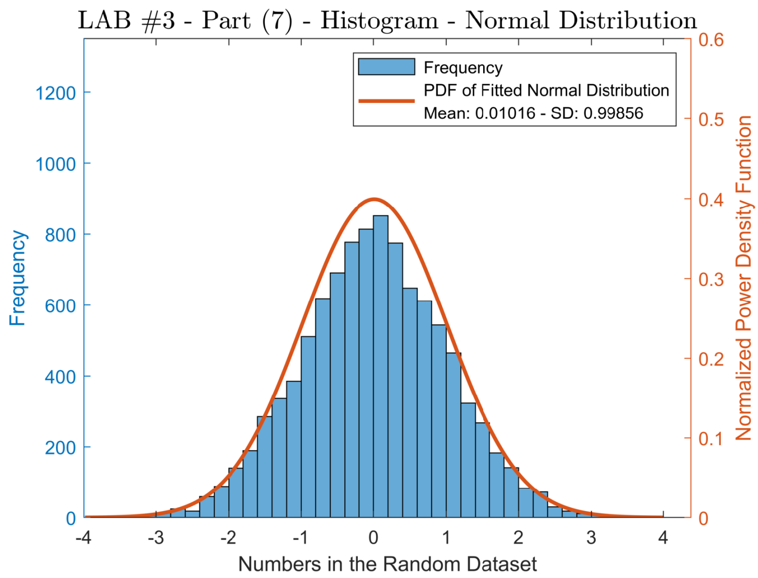

Figure 3: Lab # 3 - Part (7) - Histogram - Normal Distribution.

8 Part (8)

Perform a power spectral density (PSD) calculation using *pwelch* [Pxx,F] =PWELCH(X,WINDOW,NOVERLAP,NFFT,Fs) from the data you will generate as follows: Make a time vector, t, out to 10000 with spacing of 0.01:

$$y = 4cos(2\pi t) + sin(2\pi t/0.2) + 0.01randn(1, length(t)) \tag{8.1}$$

Mess with the different parameters in *pwelch* to see what they do to change the resulting PSD. We talked briefly about them in class. WARNING: DO NOT change more than one parameter at a time. WARNING: you cannot change Fs. That is the frequency of your input time series and is 100 Hz for the time vector I asked you to create.

To see data from *pwelch*, you normally plot on a semilogy plot with Pxx as a function of F (frequency). Tell me what the plot tells you with respect to the given function y. This will involve several plots with accompanying text to describe what you see. I can envision a plot with several curves where you have modified just one of the parameters. Then another plot with several curves where you have modified a different parameter.

```
%% Part (8)
% Perform a power spectral density (PSD) calculation using pwelch

t = 0:0.01:10000; % Make a time vector, t, out to 10000 with spacing of 0.01.
y = 4*cos(2*pi*t)+sin(2*pi*t/0.2)+0.01*randn(1,length(t)); % Generate data
Fs = 100; % Sampling frequency of input vector
```

```
% Load Default Values for WINDOW, NOVERLAP, NFFT, based on the information
% from "help pwelch"

% By default, X is divided into the longest possible sections, to get as
% close to but not exceeding 8 segments with 50% overlap.
WINDOW = floor(length(y)/8); % Define a WINDOW size that will lead to 8 segments
NOVERLAP = WINDOW/2; % Set a 50% overlap

% If NFFT is specified as empty, NFFT is set to either 256 or the next
% power of two greater than the length of each section of X, whichever is larger.
NFFT = pow2(floor(log2(WINDOW)+1)); % next power of two greater than the length of each section of X
if NFFT < 256
    % If the next of two greater than the length of each section of y (WINDOW) is smaller than 256, ...
        switch to 256
    NFFT = 256;
end

fig4 = figure(4); % Open a new figure for plotting
hold on; % Use "hold on" to plot all three parabolas on the same plot
[Pxx,F] = pwelch(y,WINDOW,NOVERLAP,NFFT,Fs); % Plot the data from pwelch for the default values for ...
    the parameters
semilogy(F,Pxx,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
xlabel('Frequency (Hz)') % Set a label for the x-axis
ylabel('Log-PSD [$V^{2}/Hz$]') % Set a label for the y-axis
title('LAB \#3 - Part (8) - PSD for default parameters of ...
    pwelch','FontSize',14,'interpreter','latex') % Create a title for the figure
print(fig4,'lab_3_part_8_default','-depsc','-r600'); % Save figure in a colored .eps format using 600 ...
    dpi resolution
%-----------------------------------------------------------------------------
% Modify only the paremeter WINDOW
% The smaller the window, the smaller the resolution of the spectrum
[Pxx,F] = pwelch(y,WINDOW,NOVERLAP,NFFT,Fs);
WINDOW_1 = WINDOW/1.5; % Lowest possible value, since NOVERLAP has to be lower than WINDOW
[Pxx_1,F_1] = pwelch(y,WINDOW_1,NOVERLAP,NFFT,Fs);
WINDOW_2 = WINDOW*1.5; % Slightly larger WINDOW than the default value
[Pxx_2,F_2] = pwelch(y,WINDOW_2,NOVERLAP,NFFT,Fs);
WINDOW_3 = WINDOW*5; % Larger WINDOW than the default value
[Pxx_3,F_3] = pwelch(y,WINDOW_3,NOVERLAP,NFFT,Fs);

fig5 = figure(5); % Open a new figure for plotting
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F,Pxx,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_1,Pxx_1,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_2,Pxx_2,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_3,Pxx_3,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
legend('Default','WINDOW/1.5','WINDOW*1.5','WINDOW*5')
title('LAB \#3 - Part (8) - PSD for various values of the WINDOW ...
    parameter','FontSize',14,'interpreter','latex') % Create a title for the figure
% Resize the figure to be wider, so that all the month labels
% on the x-axis are not overlapping.
set(gcf,'units','normalized') % So that it does not depend on the resolution of the user's monitor
set(gcf,'position',[0 0.2843 1 0.3889])
xlabel('Frequency (Hz)') % Set a label for the x-axis
ylabel('Log-PSD [$V^{2}/Hz$]') % Set a label for the y-axis

% Define second set of axes to zoom over the first peak
ax2 = axes('Position',[0.2 0.6 0.28 0.28],'Box','on'); % Define second set of axes
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F,Pxx,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_1,Pxx_1,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_2,Pxx_2,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_3,Pxx_3,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
axis(ax2,[0.995 1.005 ax2.YLim]); % Define the axis limits of the second set of axes to observe a ...
    close neighborhood of the target

% Define third set of axes to zoom over the second peak
ax3 = axes('Position',[0.3 0.25 0.28 0.28],'Box','on'); % Define second set of axes
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F,Pxx,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_1,Pxx_1,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_2,Pxx_2,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
```

```
semilogy(F_3,Pxx_3,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
axis(ax3,[4.995 5.005 ax3.YLim(1) 500]); % Define the axis limits of the second set of axes to ...
    observe a close neighborhood of the target
print(fig5,'lab_3_part_8_window','-depsc','-r600'); % Save figure in a colored .eps format using 600 ...
    dpi resolution
%-----------------------------------------------------------------------
% Modify only the paremeter NOVERLAP
[Pxx,F] = pwelch(y,WINDOW,NOVERLAP,NFFT,Fs);
NOVERLAP_1 = NOVERLAP/2;
[Pxx_1,F_1] = pwelch(y,WINDOW,NOVERLAP_1,NFFT,Fs);
NOVERLAP_2 = NOVERLAP/4;
[Pxx_2,F_2] = pwelch(y,WINDOW,NOVERLAP_2,NFFT,Fs);
NOVERLAP_3 = NOVERLAP/8;
[Pxx_3,F_3] = pwelch(y,WINDOW,NOVERLAP_3,NFFT,Fs);
NOVERLAP_4 = NOVERLAP*1.5 % Highest possible value is WINDOW-1, since NOVERLAP has to be lower than ...
    WINDOW
[Pxx_4,F_4] = pwelch(y,WINDOW,NOVERLAP_4,NFFT,Fs);

fig6 = figure(6); % Modify only the 'WINDOW' parameter
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F,Pxx,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_1,Pxx_1,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_2,Pxx_2,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_3,Pxx_3,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_4,Pxx_4,'Linewidth',2)
legend('Default','NOVERLAP/2','NOVERLAP/4','NOVERLAP/8','NOVERLAP*1.5')
title('LAB \#3 - Part (8) - PSD for various values of the NOVERLAP ...
    parameter','FontSize',14,'interpreter','latex') % Create a title for the figure
% Resize the figure to be wider, so that all the month labels
% on the x-axis are not overlapping.
set(gcf,'units','normalized') % So that it does not depend on the resolution of the user's monitor
set(gcf,'position',[0 0.2843 1 0.3889])
xlabel('Frequency (Hz)') % Set a label for the x-axis
ylabel('Log-PSD [$V^{2}/Hz$]') % Set a label for the y-axis

% Define second set of axes to zoom over the first peak
ax2 = axes('Position',[0.2 0.6 0.28 0.28],'Box','on'); % Define second set of axes
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F,Pxx,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_1,Pxx_1,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_2,Pxx_2,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_3,Pxx_3,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_4,Pxx_4,'Linewidth',2)
axis(ax2,[0.995 1.005 ax2.YLim]); % Define the axis limits of the second set of axes to observe a ...
    close neighborhood of the target

% Define third set of axes to zoom over the second peak
ax3 = axes('Position',[0.3 0.25 0.28 0.28],'Box','on'); % Define second set of axes
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F,Pxx,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_1,Pxx_1,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_2,Pxx_2,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_3,Pxx_3,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_4,Pxx_4,'Linewidth',2)
axis(ax3,[4.995 5.005 ax3.YLim(1) 500]); % Define the axis limits of the second set of axes to ...
    observe a close neighborhood of the target
print(fig6,'lab_3_part_8_noverlap','-depsc','-r600'); % Save figure in a colored .eps format using ...
    600 dpi resolution
%-----------------------------------------------------------------------
% Modify only the paremeter NFFT
[Pxx,F] = pwelch(y,WINDOW,NOVERLAP,NFFT,Fs);
NFFT_1 = round(NFFT/2.5);
[Pxx_1,F_1] = pwelch(y,WINDOW,NOVERLAP,NFFT_1,Fs);
% NFFT_2 = round(NFFT/10);
% [Pxx_2,F_2] = pwelch(y,WINDOW,NOVERLAP,NFFT_2,Fs);
NFFT_3 = NFFT*10 % Highest possible value is WINDOW-1, since NOVERLAP has to be lower than WINDOW
[Pxx_3,F_3] = pwelch(y,WINDOW,NOVERLAP,NFFT_3,Fs);

fig7 = figure(7) % Modify only the 'WINDOW' parameter
```

```
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F,Pxx,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_1,Pxx_1,'Linewidth',2)
% hold on; % Use "hold on" to plot all three parabolas on the same plot
% semilogy(F_2,Pxx_2,'Linewidth',2)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_3,Pxx_3,'Linewidth',2)
legend('Default','NFF5/2.5','NFT*5','NFT*5')
title('LAB \#3 - Part (8) - PSD for various values of the NFFT ...
    parameter','FontSize',14,'interpreter','latex') % Create a title for the figure
% Resize the figure to be wider, so that all the month labels
% on the x-axis are not overlapping.
set(gcf,'units','normalized') % So that it does not depend on the resolution of the user's monitor
set(gcf,'position',[0 0.2843 1 0.3889])
xlabel('Frequency (Hz)') % Set a label for the x-axis
ylabel('Log-PSD [$V^{2}/Hz$]') % Set a label for the y-axis

% Define second set of axes to zoom over the first peak
ax2 = axes('Position',[0.2 0.6 0.28 0.28],'Box','on'); % Define second set of axes
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F,Pxx,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_1,Pxx_1,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
% hold on; % Use "hold on" to plot all three parabolas on the same plot
% semilogy(F_2,Pxx_2,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_3,Pxx_3,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
axis(ax2,[0.995 1.005 ax2.YLim]); % Define the axis limits of the second set of axes to observe a ...
    close neighborhood of the target

% Define third set of axes to zoom over the second peak
ax3 = axes('Position',[0.3 0.25 0.28 0.28],'Box','on'); % Define second set of axes
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F,Pxx,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_1,Pxx_1,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
% hold on; % Use "hold on" to plot all three parabolas on the same plot
% semilogy(F_2,Pxx_2,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
hold on; % Use "hold on" to plot all three parabolas on the same plot
semilogy(F_3,Pxx_3,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
axis(ax3,[4.995 5.005 ax3.YLim(1) 500]); % Define the axis limits of the second set of axes to ...
    observe a close neighborhood of the target
print(fig7,'lab_3_part_8_nfft','-depsc','-r600'); % Save figure in a colored .eps format using 600 ...
    dpi resolution
```

Analysis of *pwelch* with default parameters

In Fig.4, the PSD of the generated signal y is illustrated using *pwelch* with its default parameters. Specifically, according to MATLAB: *"By default, X is divided into the longest possible sections, to get as close to but not exceeding 8 segments with 50% overlap."*. As a result, $WINDOW$ was set equal to the floored (rounded toward negative infinity) length of the signal y divided by eight (8). Next, $NOVERLAP$ was set equal to $\frac{WINDOW}{2}$, in order to have 50% overlap. Lastly, according to MATLAB: *"If NFFT is specified as empty, NFFT is set to either 256 or the next power of two greater than the length of each section of X, whichever is larger."*. Therefore, after a quick comparison, $NFFT$ was set equal to the next power of two greater than the length of each section of y. The actual default parameters are listed below:

$$WINDOW = 125000 \quad NOVERLAP = 62500 \quad NFFT = 131072$$

Based on the equation for y (Eq.8.1, it can be easily observed that the signal consists of two sinusoidal signals centered at $f_1 = 1Hz$ and $f_2 = 5Hz$. Hence, we expect its spectrum to have two spikes at those two frequencies, indicating that most of the signal's information-power is located there. By inspecting Fig.4, the above conclusion is verified, meaning that the PSD was calculated properly.

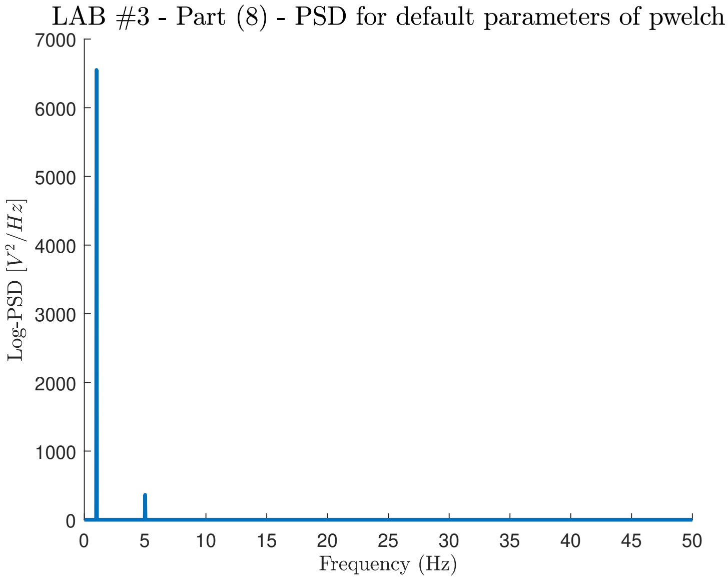

Figure 4: Lab # 3 - Part (8) - Default Parameters.

Explore $WINDOW$ parameter for *pwelch*

In Fig.5, the PSD of the generated signal y is illustrated using *pwelch* with different values only for the parameter $WINDOW$, while the default parameters were utilized for $NOVERLAP$ and $NFFT$. In total, three different values were explored: $WINDOW/1.5$, $WINDOW * 1.5$ and $WINDOW * 5$, where $WINDOW$ is the default value utilized in the previous paragraph. In general, the larger the size of the window, the smaller the number of sections that the signal is divided into. Therefore, as the size of the window increases, the resolution of the PSD decreases. Likewise, by decreasing the size of the window, the resolution should theoretically increase, however since the $WINDOW$ parameter has to remain greater than the $NOVERLAP$ parameter, there is not enough room for exploration without adjusting both parameters. As expected, in Fig.5 it is observed that increasing significantly the length of the window ($WINDOW * 5$), deteriorates quite a lot the resolution of the PSD around the two peaks. Regarding the other two values, it seems that the peaks are somewhat affected, where reducing and increasing slightly the window length leads to lower and higher peak values in the PSD, respectively.

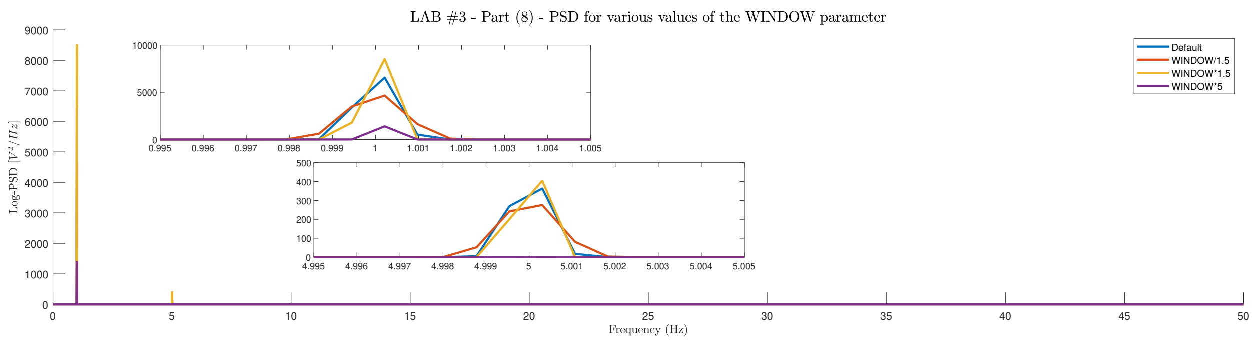

Figure 5: Lab # 3 - Part (8) - Explore $WINDOW$ parameter.

Explore $NOVERLAP$ parameter for *pwelch*

In Fig.6, the PSD of the generated signal y is illustrated using *pwelch* with different values only for the parameter $NOVERLAP$, while the default parameters were utilized for $WINDOW$ and $NFFT$. In total, four different values were explored: $NOVERLAP/2$, $NOVERLAP/4$ and $NOVERLAP/8$, where $NOVERLAP * 1.5$ is the default value utilized in the previous paragraph. In general, overlapping is primarily used to reduce the effect of windowing. The Hamming window utilized by *pwelch* function is taper-shaped, meaning that it drops to 0 (or close to 0) near the frame edges. Naturally, this affects the resulting PSD and some important information might be lost. Therefore, overlapping is employed to reduce these negative effects. The basic idea here is that we can average PSD results form overlapping frames and thus obtain a better frequency representation of our time-domain signal. Again, the highest possible value for $NOVERLAP$ is restricted by the length of the window, hence there is not enough room for exploration without adjusting both parameters. In Fig.6, it is observed that the PSD is not affected significantly by the magnitude of overlapping. Perhaps, this illustrates that this parameter alone is not capable of modifying the PSD given the values of the other two parameters or that its effect is constrained for the given signal. Further analysis is required, where two or even all three parameters would be modified at the same time in order to extract more information.

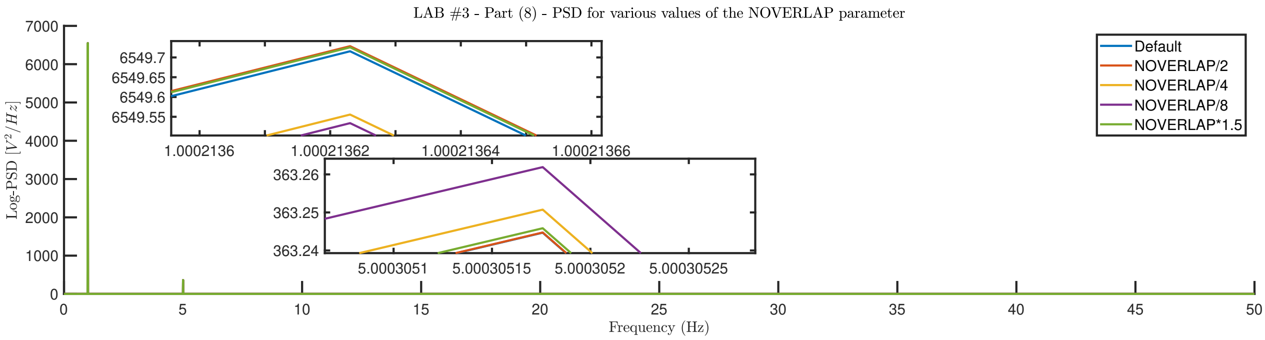

Figure 6: Lab # 3 - Part (8) - Explore $NOVERLAP$ parameter.

Explore $NFFT$ parameter for *pwelch*

In Fig.7, the PSD of the generated signal y is illustrated using *pwelch* with different values only for the parameter $NFFT$, while the default parameters were utilized for $NOVERLAP$ and $WINDOW$. In total, two different values were explored: $NFFT/2.5$ and $NFFT * 5$, where $NFFT$ is the default value utilized in the previous paragraph. In general, the larger the size of the $NFFT$, the higher the resolution of the PSD. As expected, in Fig.7 it is observed that increasing the number of DFT points, significantly improves the resolution of the PSD around the two peaks.

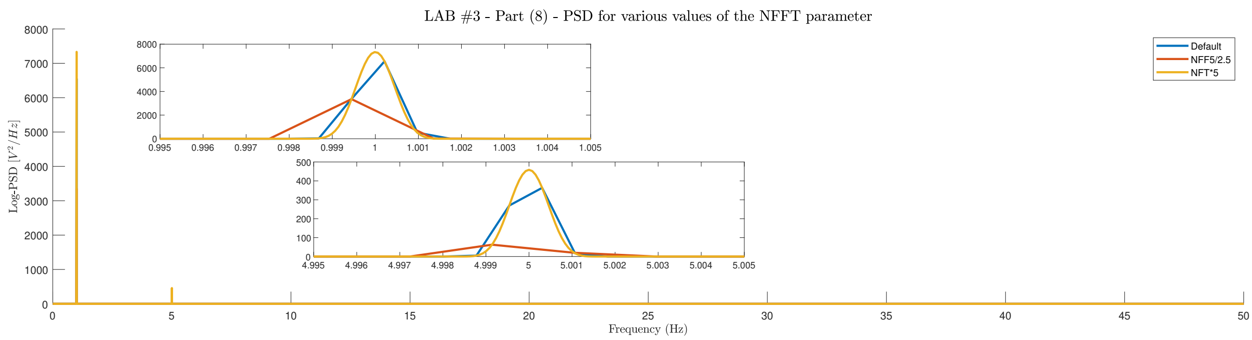

Figure 7: Lab # 3 - Part (8) - Explore $NFFT$ parameter.