Lab #3 - CIEG 675/010

Chrysostomos Karakasis

Due date: January 26, 2021

```
1  %% CIEG 675 LAB#3 Due Tuesday January 26, 2021
2  %% Author: Chrysostomos Karakasis 702529334
3  close all;
4  clear all;
5  % 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.

```
1   %% Part (1)
2   % Write a function that will take an arbitrary, but smooth, vector of data
3   % and locate ALL the local maxima and minima in the vector (call it min_max,
4   % as "minmax" is a built-in function in newer MATLAB versions starting with
5   % R2014b). It should return, as output, the indices of the input vector
6   % where these maxima and minima occur. Test your function on the following
7   % data set that you should make quite smooth by using small increments in
8   % x, where x should go from 0 to 10:
9   % y = x^{1.01}+4*cos(3*pi*x/4)-2*sin(2*pi*x/3)-0.25
10  % In the same figure, plot y vs x, with the maxima and minima (computed by
11  % your function, min_max) on top of the curve as different symbols
12  % (remember you can use >> help plotto see a list of available line/marker
13  % specifiers). DO NOT USE a built in function to find local minima or
14  % maxima. I want you to write the code to do it.
15
16  x = 0:0.01:10; % Create the x-vector with a small enough increment
17  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
18  [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
19  fig1 = figure(1); % Open a new figure
20  hold on; % Use "hold on" to plot all three parabolas on the same plot
21  plot(x,y,'LineWidth',2) % Plot the y vs x data
22  plot(x(min_ind),y(min_ind),'ro','LineWidth',2) % Plot the local minima using a red 'o' symbol
23  plot(x(max_ind),y(max_ind),'mx','LineWidth',2) % Plot the local maxima using a magenta 'x' symbol
24  % Create a new legend specifying everything on the figure
25  legend('Arbitrary-smooth Vector of Data','Local Minima','Local ...
        Maxima','interpreter','latex','location','southeast','FontSize',11)
26  title('LAB \#3 - Part (1) - Min\_max','FontSize',14,'interpreter','latex') % Create a title for the ...
        figure
27  xlabel('X-axis','FontSize',14,'interpreter','latex') % Set a label for the x-axis
28  ylabel('Y-axis','FontSize',14,'interpreter','latex') % Set a label for the y-axis
29  axis([0 10 -10 14]) % Increase axis limits
30  print(fig1,'lab_3_part_1','-depsc','-r600'); % Save figure in a colored .eps format using 600 dpi ...
        resolution
```

```
1   function [min_ind,max_ind] = min_max(y)
2   % function [min_ind,max_ind] = min_max(y)
3
4   % Description: This function receives a smooth vector of data as an input.
5   % Then the first derivative test is utilized to calculate the local extrema
6   % of the vector. Finally, the function returns the indices of the input
7   % vector where these maxima and minima occur.
8
9   % Since the input vector of data is smooth, we can differentiate it
10  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
11
12  min_ind = []; % Initialize a vector that will contain the indices of the local minima
```

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

Command Window Output

```
1  min_ind =
2
3     115    391    667    942
4
5
6  max_ind =
7
8     260    535    811
```
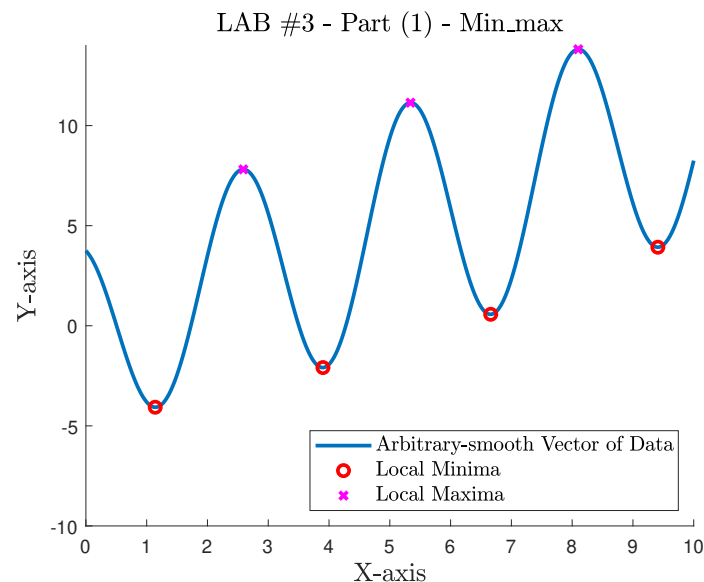

Figure 1: Lab # 3 - Part (1) - Example of the *min_max* function.

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).

```
1  %% Part (2)
2  % Create a 1 x 2 structure array with 3 fields of your choosing. The only
3  % stipulation is that the 1st field must be a string (class: char), the
4  % 2nd field must be a cell array (class: cell), and the 3rd field should be
5  % an array of numeric values (class: double).
6
7  D(1,1).field1 = 'hey'; % Define the first field of the first element of the struct structure ...
        (char:string)
8  D(1,1).field2 = {2}; % Define the second field of the first element of the struct structure (cell)
9  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)
10
11  D(1,2).field1 = 'there'; % Define the first field of the second element of the struct structure ...
        (char:string)
12  D(1,2).field2 = {4}; % Define the second field of the second element of the struct structure (cell)
13  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)
```

```
14
15  D % Show that D is a 1x2 structure array
16  class(D(1).field1) % Show that the 1st field is indeed a string (class: char)
17  class(D(1).field2) % Show that the 2st field is indeed a cell array (class: cell)
18  class(D(1).field3) % Show that the 1st field is indeed an array of doubles (class: double)
```

Command Window Output

```
1   D =
2
3     1x2 struct array with fields:
4
5       field1
6       field2
7       field3
8
9
10  ans =
11
12      'char'
13
14
15  ans =
16
17      'cell'
18
19
20  ans =
21
22      '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'.

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

Command Window Output

```
1   c_a_3 =
2
3     1x12 cell array
4
5     Columns 1 through 11
6
7       {'January'}    {'February'}    {'March'}    {'April'}    {'May'}    {'June'}    {'July'}    ...
            {'August'}    {'September'}    {'October'}    {'November'}
8
9     Column 12
10
11      {'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.

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

```
5   % then from days to seconds. The output vector should be the number of
6   % seconds that have elapsed since the 1st value in the input vector.
7
8   % First, we will create an array with four arbitrary dates in an ascending
9   % order and then we will convert them to MATLAB dates using the 'datenum'
10  % built-in function
11  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)];
12  num_sec = day2secthat(mat_dates_v) % Call the 'day2secthat' function
13
14  % Two versions of the function 'day2secthat' were created
15  num_sec_2 = day2secthat2(mat_dates_v) % Call the 'day2secthat2' function
```

```
1   function [num_sec] = day2secthat(mat_dates_v)
2   % function [num_sec] = day2secthat(mat_dates_v)
3
4   % Description: This function receives as an input a vector of MATLAB dates
5   % and returns as an output a vector containing the number of seconds that
6   % have elapsed since the 1st value (date) in the input vector
7
8   mat2str = datestr(mat_dates_v); % Convert the input vector to string format
9   str2vec = datevec(mat2str); % Convert the string format to a vector of components
10  num_sec = []; % Initialize the output vector that will contain the elapsed number of seconds
11  for i = 1:size(str2vec,1) % Iterate through all dates (elements) of the input vector
12      % Calculate the elapsed number of seconds of each date wrt to the
13      % first one and add it to the output vector.
14      num_sec = [num_sec etime(str2vec(i,:),str2vec(1,:))];
15  end
16  end
```

```
1   function [num_sec] = day2secthat2(mat_dates_v)
2   % function [num_sec] = day2secthat2(mat_dates_v)
3
4   % Description: This function receives as an input a vector of MATLAB dates
5   % and returns as an output a vector containing the number of seconds that
6   % have elapsed since the 1st value (date) in the input vector. This a
7   % second simpler version of the "day2secthat" function
8
9   % Calculate elapsed days of all elements wrt to the first date in the input
10  days_elapsed = mat_dates_v-mat_dates_v(1);
11
12  % Calculate elapsed hours of all elements wrt to the first date in the input
13  hours_elapsed = days_elapsed*24; % Each day has 24 hours
14
15  % Calculate elapsed minutes of all elements wrt to the first date in the input
16  minutes_elapsed = hours_elapsed*60; % Each hour has 60 minutes
17
18  % Calculate elapsed seconds of all elements wrt to the first date in the input
19  num_sec = minutes_elapsed*60; % Each minute has 60 seconds
20  end
```

Command Window Output

```
1   num_sec =
2
3       1.0e+09 *
4
5                    0   0.000000020000000   0.715726693000000   1.513979195000000
6
7
8   num_sec_2 =
9
10      1.0e+09 *
11
12                   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.

```matlab
%% 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
```
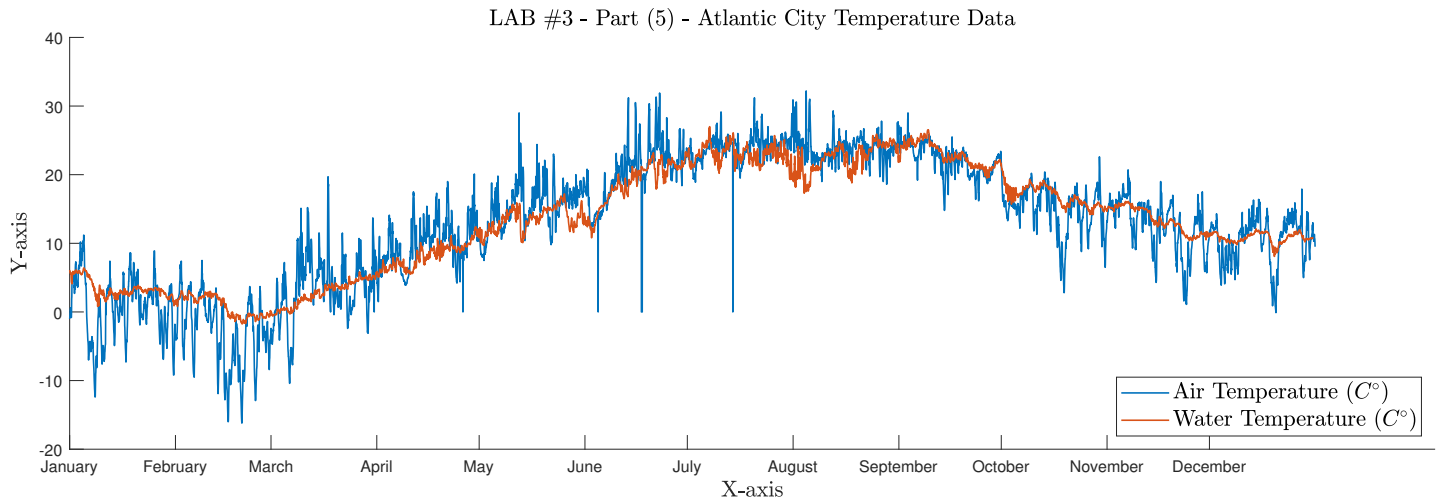
LAB #3 - Part (5) - Atlantic City Temperature Data

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.

```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

```
 1                                  _____
 2                           _,,ad8888888888bba,_
 3                       ,ad88888I888888888888888ba,
 4                     ,88888888I88888888888888888888a,
 5                   ,d888888888I8888888888888888888888b,
 6                  d88888PP""""  ""YY88888888888888888888b,
 7                ,d88"'__,,--------,,,,.;ZZZY8888888888888,
 8               ,8IIl'"                  ;;l"ZZZIII8888888888,
 9              ,I88l;'                   ;lZZZZZ888III8888888,
10            ,II88Zl;.                   ;llZZZZZ888888I888888,
11           ,II888Zl;.               .;;;;;lllZZZ888888I8888b
12          ,II8888Z;;                  `;;;;;''llZZ8888888I8888,
13          II88888Z;'                       .;lZZZ8888888I888b
14          II88888Z; _,aaa,        .,aaaaa,__.l;llZZ88888888I888
15          II88888IZZZZZZZZZ,  .ZZZZZZZZZZZZZZ;llZZ88888888I888,
16          II88888IZZ<'(@@>Z|  |ZZZ<'(@@>ZZZZ;;llZZ888888888I88I
17         ,II88888;  `"""  ;|  |ZZ; `"""    ;;llZ8888888888I888
18          II888888l            `;;            .;llZZ8888888888I888,
19        ,II888888Z;           ;;;        .;;llZZZ88888888I888I
20        IIII888888Zl;     ..,   `;;       ,;;lllZZZ88888888888I888
21        II88888888Z;;...;(_    _)    ,;;;llZZZZ888888888888I888,
22        II88888888Zl;;;;;' `--'Z;.   ..,;;;;llZZZZ88888888888I888b
23        ]I888888888Z;;;;;'   ";llllll;..;;;lllZZZZ88888888888I8888,
24        II888888888Zl.;;"Y88bd888P";;,..;lllZZZZZ88888888888I8888I
25        II8888888888Zl;.;  `"PPP";;;,..;lllZZZZZZZ88888888888I88888
26        II888888888888Zl;;. `;;;l;;;;;lllZZZZZZZZW88888888888I88888
27        `II8888888888888Zl;.    ,;;lllZZZZZZZZZWMZ88888888888I88888
28         II8888888888888888ZbaalllZZZZZZZZZZWWMZZZ8888888888I888888,
29         `II88888888888888888b"WWZZZZZWWWMMZZZZZZI888888888I888888b
30          `II888888888888888888;ZZMMMMMMMZZZZZZZZllI8888888888I88888888
31           `II8888888888888888 `;lZZZZZZZZZZZllllI888888888I8888888,
32            II88888888888888888, `;lllZZZZllllll;;.Y88888888I8888888b,
33            ,II888888888888888888b    .;;lllllll;;;.;..88888888I88888888b,
34            II888888888888888888PZI;.  .`;;;.;;;;..; ...88888888I8888888888,
35            II888888888888PZ;;';;.    ;. .;.  .;. .. Y8888888I88888888888b,
36           ,II888888888PZ;;'                     `8888888I8888888888888b,
37            II888888888'                      888888I8888888888888888b
38           ,II888888888                      ,888888I88888888888888888
39           ,d88888888888                      d888888I8888888888ZZZZZZZ
40         ,ad888888888I                        8888888I8888ZZZZZZZZZZZZZ
41        ,d888888888888888'                     888888IZZZZZZZZZZZZZZZZZZ
42       ,d888888888888P'8P'                      Y888ZZZZZZZZZZZZZZZZZZZZZ
43      ,8888888888888,  "                         ,ZZZZZZZZZZZZZZZZZZZZZZZ
44     d888888888888888,                            ,ZZZZZZZZZZZZZZZZZZZZZZZZ
45     8888888888888888888a,     _                  ,ZZZZZZZZZZZZZZZZ888888888
46     88888888888888888888888ba,_d'               ,ZZZZZZZZZZZZZZZZ88888888888
47     888888888888888888888888888888bbbaaa,,,_____,ZZZZZZZZZZZZZZZ8888888888888888
48     88888888888888888888888888888888888888888888ZZZZZZZZZZZZZZZZ88888888888888888
49     88888888888888888888888888888888888888ZZZZZZZZZZZZZZZZZZZ888888888888888888888
50     8888888888888888888888888888888888ZZZZZZZZZZZZZZZZZ8888888888888888888888888888
51     8888888888888888888888888888ZZZZZZZZZZZZZZZZZ888888888888888888888888888888
52     888888888888888888888888888ZZZZZZZZZZZZZZZ88888888888888888888888888888888
53     88888888888888888888888888ZZZZZZZZZZZZZZZ8888888888888888888888888888888888
54     8888888888888888888888888ZZZZZZZZZZZZZZZZ8888888888888888888888888888888888
55     8888888888888888888888888ZZZZZZZZZZZZZZZZ88888888888888888888888888888888888
```

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.

```
 1  %% Part (7)
 2  % Develop a data set that is 10,000 points long composed of random
 3  % numbers from a normal distribution.  Make a histogram of this data set
 4  % and then overlay a Gaussian curve on top.  You will have to determine how
 5  % to normalize your plot or histogram to get the y-axis to scale appropriately.
 6
 7  rand_data_7 = randn(10000,1); % Create the 10,000 points long vector composed of random numbers from ...
        a normal distribution
 8  fig3 = figure(3); % Open a new figure for plotting
 9  pd_7 = fitdist(rand_data_7,'Normal'); % Fit a normal distribution to the generated data
```

```
10  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
11  y_7 = pdf(pd_7,x_values_7); % Calculate corresponding y-coordinates of fitted curve
12  yyaxis right % Utilize the right vertical side as a new axis for the PDF
13  plot(x_values_7,y_7,'Linewidth',2); % Plot the Power Density Function of the fitted normal distribution
14  set(gca,'YLim',1.5*get(gca,'YLim')) % Increase the limits of the y-axis to avoid overlapping between ...
        lines and the legends
15
16  yyaxis left % Utilize the left vertical side as a new axis for the histogram
17  histogram(rand_data_7) % Create a histogram of the generated data, showing the frequency of each ...
        element inside the vector
18  set(gca,'YLim',1.5*get(gca,'YLim')) % Increase the limits of the y-axis to avoid overlapping between ...
        lines and the legends
19
20  % Create a new legend specifying everything on the figure
21  legend({' Frequency',[' PDF of Fitted Normal Distribution\newline Mean: ',num2str(pd_7.mu),' - SD: ...
        ',num2str(pd_7.sigma)]})
22  xlabel('Numbers in the Random Dataset'); % Add a label for the x-axis
23  yyaxis right % Switch back to the right y-axis
24  ylabel('Normalized Power Density Function'); % Add a label for the right y-axis
25  yyaxis left % Switch back to the left y-axis
26  ylabel('Frequency'); % Add a label for the left y-axis
27  title('LAB \#3 - Part (7) - Histogram - Normal Distribution','FontSize',14,'interpreter','latex') % ...
        Create a title for the figure
28  print(fig3,'lab_3_part_7','-depsc','-r600'); % Save figure in a colored .eps format using 600 dpi ...
        resolution
```
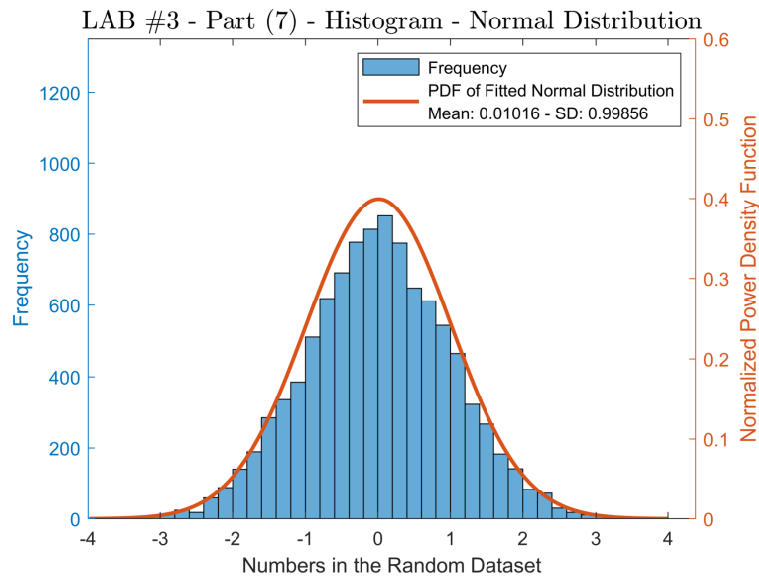

Figure 3: Lab # 3 - Part (7) - Histogram - Normal Distribution.

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:

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

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.

```
1  %% Part (8)
2  % Perform a power spectral density (PSD) calculation using pwelch
3
4  t = 0:0.01:10000; % Make a time vector, t, out to 10000 with spacing of 0.01.
5  y = 4*cos(2*pi*t)+sin(2*pi*t/0.2)+0.01*randn(1,length(t)); % Generate data
6  Fs = 100; % Sampling frequency of input vector
```

```matlab
7
8  % Load Default Values for WINDOW, NOVERLAP, NFFT, based on the information
9  % from "help pwelch"
10
11  % By default, X is divided into the longest possible sections, to get as
12  % close to but not exceeding 8 segments with 50% overlap.
13  WINDOW = floor(length(y)/8); % Define a WINDOW size that will lead to 8 segments
14  NOVERLAP = WINDOW/2; % Set a 50% overlap
15
16  % If NFFT is specified as empty, NFFT is set to either 256 or the next
17  % power of two greater than the length of each section of X, whichever is larger.
18  NFFT = pow2(floor(log2(WINDOW)+1)); % next power of two greater than the length of each section of X
19  if NFFT < 256
20      % If the next of two greater than the length of each section of y (WINDOW) is smaller than 256, ...
            switch to 256
21      NFFT = 256;
22  end
23
24  fig4 = figure(4); % Open a new figure for plotting
25  hold on; % Use "hold on" to plot all three parabolas on the same plot
26  [Pxx,F] = pwelch(y,WINDOW,NOVERLAP,NFFT,Fs); % Plot the data from pwelch for the default values for ...
        the parameters
27  semilogy(F,Pxx,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
28  xlabel('Frequency (Hz)') % Set a label for the x-axis
29  ylabel('Log-PSD [$V^{2}/Hz$]') % Set a label for the y-axis
30  title('LAB \#3 - Part (8) - PSD for default parameters of ...
        pwelch','FontSize',14,'interpreter','latex') % Create a title for the figure
31  print(fig4,'lab_3_part_8_default','-depsc','-r600'); % Save figure in a colored .eps format using 600 ...
        dpi resolution
32  %---------------------------------------------------------------------
33  % Modify only the paremeter WINDOW
34  % The smaller the window, the smaller the resolution of the spectrum
35  [Pxx,F] = pwelch(y,WINDOW,NOVERLAP,NFFT,Fs);
36  WINDOW_1 = WINDOW/1.5; % Lowest possible value, since NOVERLAP has to be lower than WINDOW
37  [Pxx_1,F_1] = pwelch(y,WINDOW_1,NOVERLAP,NFFT,Fs);
38  WINDOW_2 = WINDOW*1.5; % Slightly larger WINDOW than the default value
39  [Pxx_2,F_2] = pwelch(y,WINDOW_2,NOVERLAP,NFFT,Fs);
40  WINDOW_3 = WINDOW*5; % Larger WINDOW than the default value
41  [Pxx_3,F_3] = pwelch(y,WINDOW_3,NOVERLAP,NFFT,Fs);
42
43  fig5 = figure(5); % Open a new figure for plotting
44  hold on; % Use "hold on" to plot all three parabolas on the same plot
45  semilogy(F,Pxx,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
46  hold on; % Use "hold on" to plot all three parabolas on the same plot
47  semilogy(F_1,Pxx_1,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
48  hold on; % Use "hold on" to plot all three parabolas on the same plot
49  semilogy(F_2,Pxx_2,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
50  hold on; % Use "hold on" to plot all three parabolas on the same plot
51  semilogy(F_3,Pxx_3,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
52  legend('Default','WINDOW/1.5','WINDOW*1.5','WINDOW*5')
53  title('LAB \#3 - Part (8) - PSD for various values of the WINDOW ...
        parameter','FontSize',14,'interpreter','latex') % Create a title for the figure
54  % Resize the figure to be wider, so that all the month labels
55  % on the x-axis are not overlapping.
56  set(gcf,'units','normalized') % So that it does not depend on the resolution of the user's monitor
57  set(gcf,'position',[0 0.2843 1 0.3889])
58  xlabel('Frequency (Hz)') % Set a label for the x-axis
59  ylabel('Log-PSD [$V^{2}/Hz$]') % Set a label for the y-axis
60
61  % Define second set of axes to zoom over the first peak
62  ax2 = axes('Position',[0.2 0.6 0.28 0.28],'Box','on'); % Define second set of axes
63  hold on; % Use "hold on" to plot all three parabolas on the same plot
64  semilogy(F,Pxx,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
65  hold on; % Use "hold on" to plot all three parabolas on the same plot
66  semilogy(F_1,Pxx_1,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
67  hold on; % Use "hold on" to plot all three parabolas on the same plot
68  semilogy(F_2,Pxx_2,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
69  hold on; % Use "hold on" to plot all three parabolas on the same plot
70  semilogy(F_3,Pxx_3,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
71  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
72
73  % Define third set of axes to zoom over the second peak
74  ax3 = axes('Position',[0.3 0.25 0.28 0.28],'Box','on'); % Define second set of axes
75  hold on; % Use "hold on" to plot all three parabolas on the same plot
76  semilogy(F,Pxx,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
77  hold on; % Use "hold on" to plot all three parabolas on the same plot
78  semilogy(F_1,Pxx_1,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
79  hold on; % Use "hold on" to plot all three parabolas on the same plot
80  semilogy(F_2,Pxx_2,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
81  hold on; % Use "hold on" to plot all three parabolas on the same plot
```

```matlab
82  semilogy(F_3,Pxx_3,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
83  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
84  print(fig5,'lab_3_part_8_window','-depsc','-r600'); % Save figure in a colored .eps format using 600 ...
        dpi resolution
85  %----------------------------------------------------------------------
86  % Modify only the paremeter NOVERLAP
87  [Pxx,F] = pwelch(y,WINDOW,NOVERLAP,NFFT,Fs);
88  NOVERLAP_1 = NOVERLAP/2;
89  [Pxx_1,F_1] = pwelch(y,WINDOW,NOVERLAP_1,NFFT,Fs);
90  NOVERLAP_2 = NOVERLAP/4;
91  [Pxx_2,F_2] = pwelch(y,WINDOW,NOVERLAP_2,NFFT,Fs);
92  NOVERLAP_3 = NOVERLAP/8;
93  [Pxx_3,F_3] = pwelch(y,WINDOW,NOVERLAP_3,NFFT,Fs);
94  NOVERLAP_4 = NOVERLAP*1.5 % Highest possible value is WINDOW-1, since NOVERLAP has to be lower than ...
        WINDOW
95  [Pxx_4,F_4] = pwelch(y,WINDOW,NOVERLAP_4,NFFT,Fs);
96
97  fig6 = figure(6); % Modify only the 'WINDOW' parameter
98  hold on; % Use "hold on" to plot all three parabolas on the same plot
99  semilogy(F,Pxx,'Linewidth',2)
100 hold on; % Use "hold on" to plot all three parabolas on the same plot
101 semilogy(F_1,Pxx_1,'Linewidth',2)
102 hold on; % Use "hold on" to plot all three parabolas on the same plot
103 semilogy(F_2,Pxx_2,'Linewidth',2)
104 hold on; % Use "hold on" to plot all three parabolas on the same plot
105 semilogy(F_3,Pxx_3,'Linewidth',2)
106 hold on; % Use "hold on" to plot all three parabolas on the same plot
107 semilogy(F_4,Pxx_4,'Linewidth',2)
108 legend('Default','NOVERLAP/2','NOVERLAP/4','NOVERLAP/8','NOVERLAP*1.5')
109 title('LAB \#3 - Part (8) - PSD for various values of the NOVERLAP ...
        parameter','FontSize',14,'interpreter','latex') % Create a title for the figure
110 % Resize the figure to be wider, so that all the month labels
111 % on the x-axis are not overlapping.
112 set(gcf,'units','normalized') % So that it does not depend on the resolution of the user's monitor
113 set(gcf,'position',[0 0.2843 1 0.3889])
114 xlabel('Frequency (Hz)') % Set a label for the x-axis
115 ylabel('Log-PSD [$V^{2}/Hz$]') % Set a label for the y-axis
116
117 % Define second set of axes to zoom over the first peak
118 ax2 = axes('Position',[0.2 0.6 0.28 0.28],'Box','on'); % Define second set of axes
119 hold on; % Use "hold on" to plot all three parabolas on the same plot
120 semilogy(F,Pxx,'Linewidth',2)
121 hold on; % Use "hold on" to plot all three parabolas on the same plot
122 semilogy(F_1,Pxx_1,'Linewidth',2)
123 hold on; % Use "hold on" to plot all three parabolas on the same plot
124 semilogy(F_2,Pxx_2,'Linewidth',2)
125 hold on; % Use "hold on" to plot all three parabolas on the same plot
126 semilogy(F_3,Pxx_3,'Linewidth',2)
127 hold on; % Use "hold on" to plot all three parabolas on the same plot
128 semilogy(F_4,Pxx_4,'Linewidth',2)
129 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
130
131 % Define third set of axes to zoom over the second peak
132 ax3 = axes('Position',[0.3 0.25 0.28 0.28],'Box','on'); % Define second set of axes
133 hold on; % Use "hold on" to plot all three parabolas on the same plot
134 semilogy(F,Pxx,'Linewidth',2)
135 hold on; % Use "hold on" to plot all three parabolas on the same plot
136 semilogy(F_1,Pxx_1,'Linewidth',2)
137 hold on; % Use "hold on" to plot all three parabolas on the same plot
138 semilogy(F_2,Pxx_2,'Linewidth',2)
139 hold on; % Use "hold on" to plot all three parabolas on the same plot
140 semilogy(F_3,Pxx_3,'Linewidth',2)
141 hold on; % Use "hold on" to plot all three parabolas on the same plot
142 semilogy(F_4,Pxx_4,'Linewidth',2)
143 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
144 print(fig6,'lab_3_part_8_noverlap','-depsc','-r600'); % Save figure in a colored .eps format using ...
        600 dpi resolution
145 %----------------------------------------------------------------------
146 % Modify only the paremeter NFFT
147 [Pxx,F] = pwelch(y,WINDOW,NOVERLAP,NFFT,Fs);
148 NFFT_1 = round(NFFT/2.5);
149 [Pxx_1,F_1] = pwelch(y,WINDOW,NOVERLAP,NFFT_1,Fs);
150 % NFFT_2 = round(NFFT/10);
151 % [Pxx_2,F_2] = pwelch(y,WINDOW,NOVERLAP,NFFT_2,Fs);
152 NFFT_3 = NFFT*10 % Highest possible value is WINDOW-1, since NOVERLAP has to be lower than WINDOW
153 [Pxx_3,F_3] = pwelch(y,WINDOW,NOVERLAP,NFFT_3,Fs);
154
155 fig7 = figure(7) % Modify only the 'WINDOW' parameter
```

```matlab
156  hold on; % Use "hold on" to plot all three parabolas on the same plot
157  semilogy(F,Pxx,'Linewidth',2)
158  hold on; % Use "hold on" to plot all three parabolas on the same plot
159  semilogy(F_1,Pxx_1,'Linewidth',2)
160  % hold on; % Use "hold on" to plot all three parabolas on the same plot
161  % semilogy(F_2,Pxx_2,'Linewidth',2)
162  hold on; % Use "hold on" to plot all three parabolas on the same plot
163  semilogy(F_3,Pxx_3,'Linewidth',2)
164  legend('Default','NFF5/2.5','NFT*5','NFT*5')
165  title('LAB \#3 - Part (8) - PSD for various values of the NFFT ...
         parameter','FontSize',14,'interpreter','latex') % Create a title for the figure
166  % Resize the figure to be wider, so that all the month labels
167  % on the x-axis are not overlapping.
168  set(gcf,'units','normalized') % So that it does not depend on the resolution of the user's monitor
169  set(gcf,'position',[0 0.2843 1 0.3889])
170  xlabel('Frequency (Hz)') % Set a label for the x-axis
171  ylabel('Log-PSD [$V^{2}/Hz$]') % Set a label for the y-axis
172
173  % Define second set of axes to zoom over the first peak
174  ax2 = axes('Position',[0.2 0.6 0.28 0.28],'Box','on'); % Define second set of axes
175  hold on; % Use "hold on" to plot all three parabolas on the same plot
176  semilogy(F,Pxx,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
177  hold on; % Use "hold on" to plot all three parabolas on the same plot
178  semilogy(F_1,Pxx_1,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
179  % hold on; % Use "hold on" to plot all three parabolas on the same plot
180  % semilogy(F_2,Pxx_2,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
181  hold on; % Use "hold on" to plot all three parabolas on the same plot
182  semilogy(F_3,Pxx_3,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
183  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
184
185  % Define third set of axes to zoom over the second peak
186  ax3 = axes('Position',[0.3 0.25 0.28 0.28],'Box','on'); % Define second set of axes
187  hold on; % Use "hold on" to plot all three parabolas on the same plot
188  semilogy(F,Pxx,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
189  hold on; % Use "hold on" to plot all three parabolas on the same plot
190  semilogy(F_1,Pxx_1,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
191  % hold on; % Use "hold on" to plot all three parabolas on the same plot
192  % semilogy(F_2,Pxx_2,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
193  hold on; % Use "hold on" to plot all three parabolas on the same plot
194  semilogy(F_3,Pxx_3,'Linewidth',2) % Plot on a semilogy plot with Pxx as a function of F (frequency)
195  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
196  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.
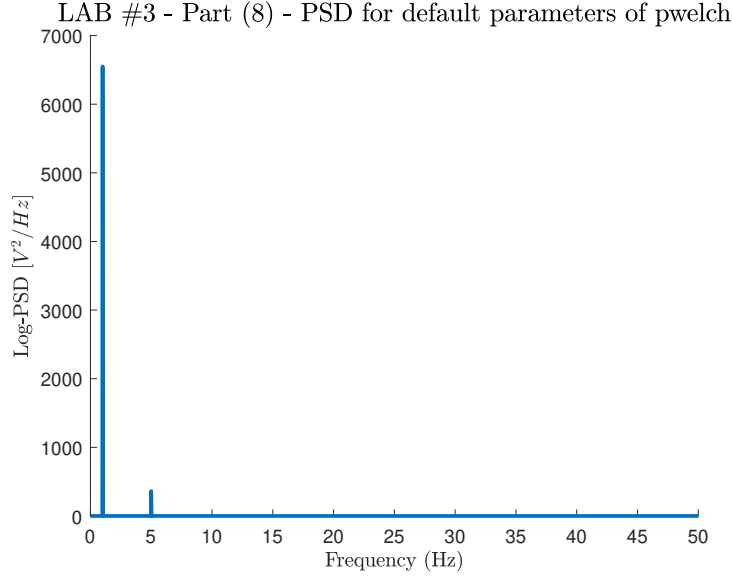
Figure 4: Lab # 3 - Part (8) - Default Parameters.

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.
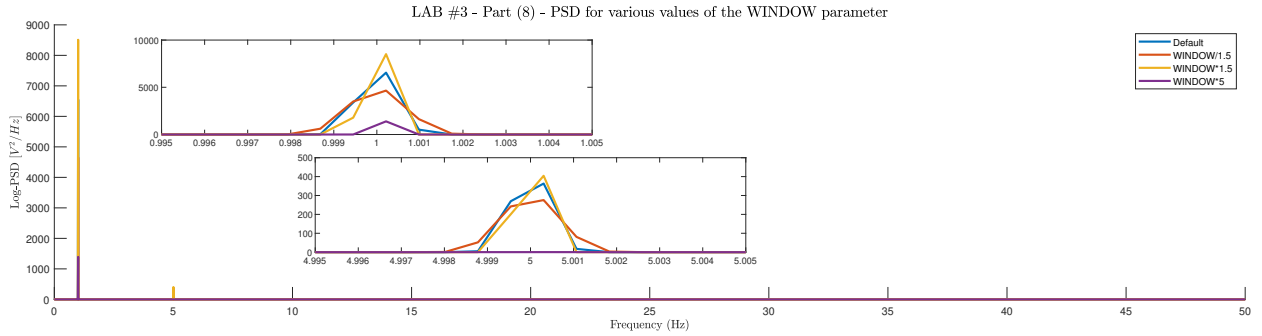

Figure 5: Lab # 3 - Part (8) - Explore $WINDOW$ parameter.

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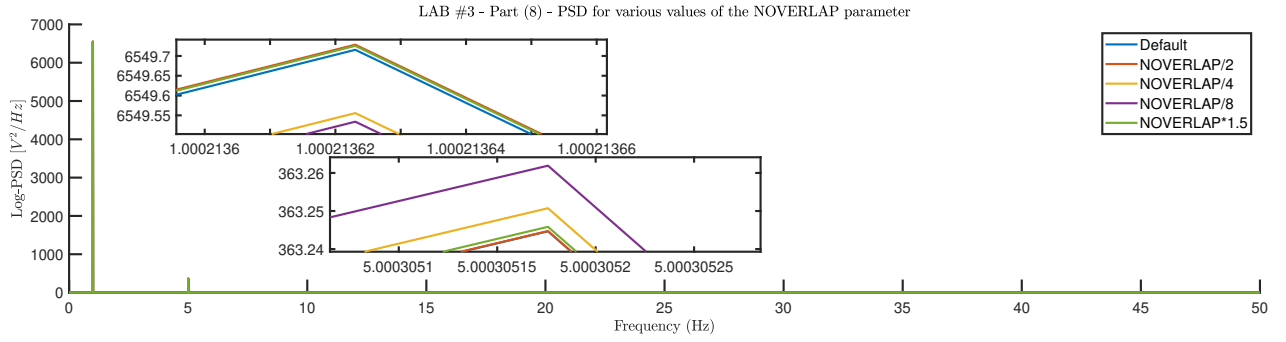
Figure 6: Lab # 3 - Part (8) - Explore $NOVERLAP$ parameter.

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.
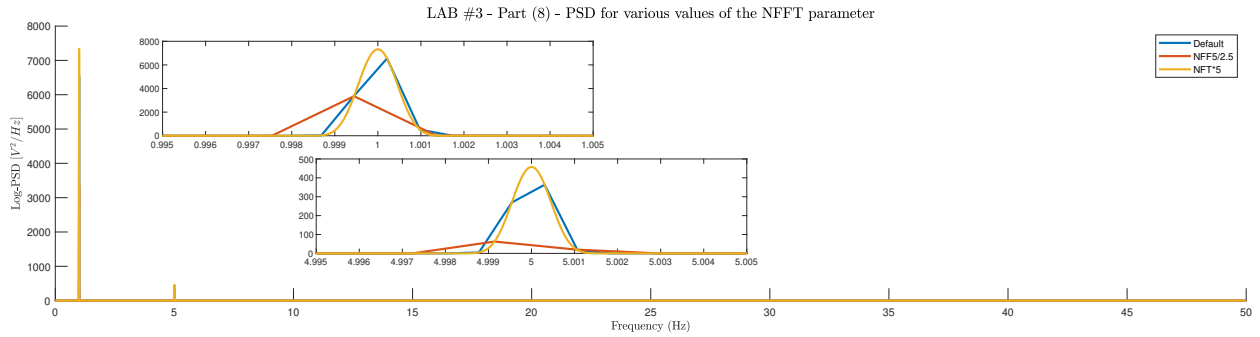

Figure 7: Lab # 3 - Part (8) - Explore $NFFT$ parameter.