CIEG 675 - Matlab For Engineering Analysis (Lab 3)

Abdul Fayeed Abdul Kadir

PROBLEM 1

```
function [index_min,index_max] = min_max(vector)
% function [index_min,index_max] = min_max(vector)
%
% This function will get an input of data vector and locate the local
% maxima and minima of the curve plotted between equation y = vector.^1.01
% + 4*cos(3*pi.*vector./4) - 2*sin(2*pi.*vector./3) - 0.25 vs. vector. This
% function will output the plot altogether with the local maxima and minima
% marked on the same plot.
%
% Input:
% vector - a vector of data for equation y for the plot's x-axis
%
% Output:
% index_min - vector of indices where the local minima occurs
% index_max - vector of indices where the local maxima occurs

index_min = [];index_max = [];
% Empty array to be filled with indices later on

y = vector.^1.01 + 4*cos(3*pi.*vector./4) - 2*sin(2*pi.*vector./3) - 0.25; % Example of equation y

for i = 1:length(vector)
   % Should place first two ifs for i = 1 and length(vector) to prevent
   % error for the other two elseifs, since MATLAB will get angry for i+1
   % or i-1 which will result in 'index exceeds number of array elements'.

   if i == 1
   elseif i == length(vector)
   % Do nothing to these two points (beg. and end points)
   elseif y(i) > y(i+1) && y(i) > y(i-1)
      % For y-value at position i that is greater than previous and next
      % data, it will be the local maxima
      index_max = cat(1,index_max,i); % concantenate vertically
   elseif y(i) < y(i+1) && y(i) < y(i-1)
      % For y-value at position i that is smaller than previous and next
      % data, it will be the local minima
      index_min = cat(1,index_min,i);
   end % end of first if statement
end % end of for loop

figure('name','Local maxima and minima')
plot(vector,y,'k-','linewidth',1);
hold on
localmin = plot(vector(index_min),y(index_min),'ro','markerfacecolor','r','markersize',7);
localmax = plot(vector(index_max),y(index_max),'bo','markerfacecolor','b','markersize',7);
legend([localmin,localmax],'Local Minimum','Local Maximum','location','north')
% Ways to label legend on certain stuff only
set(gca,'fontname','times','fontsize',12,'xcolor',[0 0 0],'ycolor',[0 0 0])
set(gcf,'color',[1 1 1])
end
x = 0:0.01:10;
[min_idx,max_idx] = min_max(x);
print -dtiff -r600 Problem1_curve
```

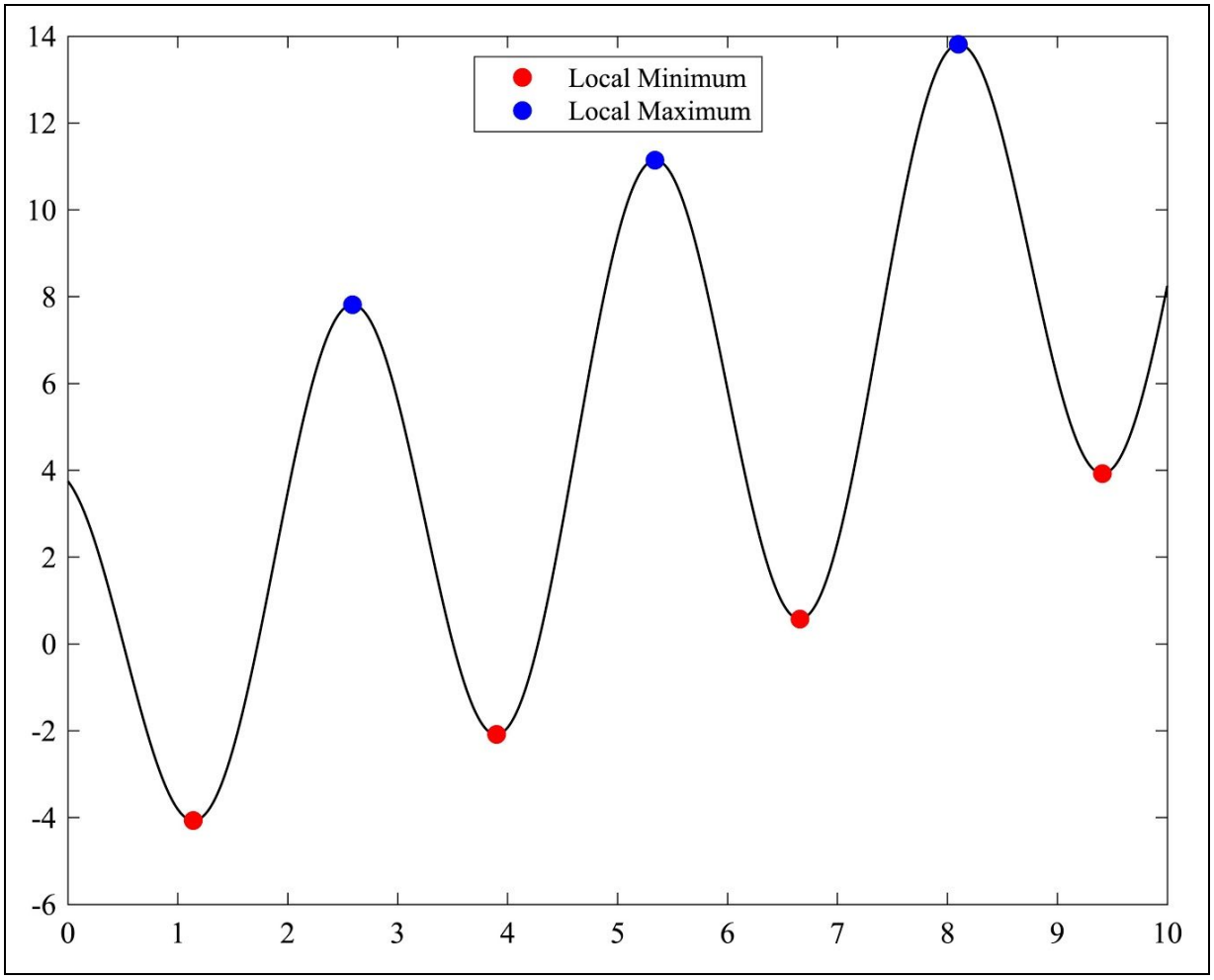

Figure 1: Local Maxima and Minima for Problem 1

PROBLEM 2

```
% * 1 x 2 structure array - multidimensional structure
% * 3 fields needed - 1) string (class: char), 2) cell array (class: cell),
% 3) numeric array (class: double)
% * Each dimension will have different things in each field
% * The structure is college, and the fields are major, favorite spots at
% college, and credit hours per semester.
% * Each field doesn't have to be the same length

% First dimension
college.major = 'Business';
college.fav_spots = {'ISE','Perkins','Roots'}; % {} for cell array
college.credits_per_sem = [12 15 15 12 14 15]; % [] for matrices

% Second dimension
college(2).major = 'Engineering';
college(2).fav_spots = {'Morris','Trabant','Gore','Caesar Rodney'};
college(2).credits_per_sem = [15 18 18 16];
```

PROBLEM 3

```
% * 12 x 1 cell array containing twelve months of the year in order

months = {'January';'February';'March';'April';'May';'June';'July';...
    'August';'September';'October';'November';'December'};
```

PROBLEM 4

```
function [seconds] = day2sec(datenumber)
% function [seconds] = day2sec(dates)
%
% This function will output the number of seconds that have elapsed since
% the first MATLAB datenumber in the input vector. The datenumber vector is
% already in the correct format of MATLAB datenumber and it is increasing
% monotonically.
%
% Input:
% datenumber - Vector of MATLAB datenumber (i.e. 2021/1/23 = 738,179 MATLAB
% datenumber)
%
% Output:
% seconds - Vector of seconds elapsed after the first date in the input
% vector

% days = datenum(dates);
% To change to MATLAB time serial number - number 1 represents Jan 1 0000,
% and everything follows in increasing order from that date (don't have to
% convert to matlab datenum anymore)

datenumber = datenumber - datenumber(1);
% To change the time relative to the first date value in the input vector

seconds = datenumber*24*3600;
% To change the days into seconds

end
```

PROBLEM 5

```
% * Load the data from Atlantic City Temperature Data csv file
% * First column is the dates in MATLAB time, for all day in 2015
% * Second column is the air temperature (degree celcius)
% * Third column is the water temperature (degree celcius)
% * All data are corrected in reference to the first row data
% * Plots of temperature vs. time

temp_data = dlmread('AtlanticCity_TemperatureData.csv');
% Load data from the said file (no specifier needed like %6.5f cuz dlmread
% can understand the existence of columns and rows in the csv)

temp_data = temp_data - temp_data(1,:);
% Corrected in reference to first row data

days = temp_data(:,1) + 1; % First column and make day 0 to day 1
air_temp = temp_data(:,2); % Second column
water_temp = temp_data(:,3); % Third column

figure('name','Water and Air Temperature')
plot(days,air_temp,'k-','linewidth',1.5)
hold on
plot(days,water_temp,'r-','linewidth',1.5)
legend('Air Temperature','Water Temperature','fontweight','bold')
xlabel('Months','fontweight','bold');
ylabel(['Temperature (' char(176) 'C)'],'fontweight','bold');
% use [] to concantenate the naming, and char(176) for the degree symbol
xlim([1,366]);
set(gca,'fontname','times','xcolor',[0 0 0],'ycolor',[0 0 0],'fontsize',12,...
    'xtick',[1,32,60,91,121,152,182,213,244,274,305,335],'xticklabel',months)
set(gcf,'position',[103 70 1000 500],'color',[1 1 1])

print -djpeg Problem5_curve
```

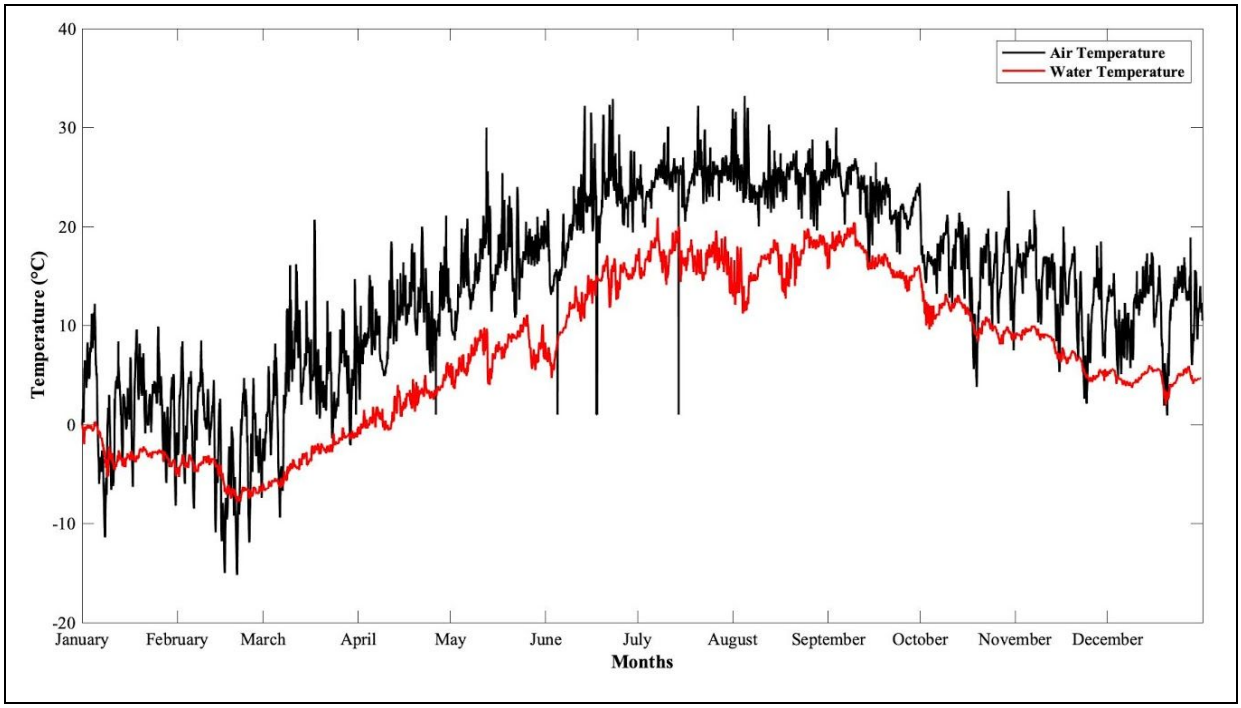

Figure 2: Water and Air Temperature for Problem 5

PROBLEM 6

```
% * Print each line in the surprise.txt file, one character at a time into
% the command window

ct = 0; % a counter
fid = fopen('surprise.txt', 'r'); % open file for reading

% Keep running a loop until EndOfFile (feof)
% (feof(fid) = 1 if end of the file is reached) - 0 otherwise
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

for i = 1:ct % Loop over the entire lines
   for j = 1:length(data{i}) % Accessing each character in each cell lines
      fprintf(data{i}(j)) % Print each character j in line i
      pause(1/500); % To ensure one character at a time printed
   end
   fprintf('\n') % To get to a new line
end
```

PROBLEM 7

```
% * Plot histograms and overlay Gaussian curve on top

y = randn(10000,1); % Generate random # for normal dist.
figure('name','Histograms and Gaussian Distribution')
histogram(y,'normalization','pdf','facecolor','r','edgecolor','k','linewidth',1);
% Plot histograms, with some normalization and probability dist. func.
% (pdf)
hold on

[n,edges] = histcounts(y,'normalization','pdf'); % Info. on the histograms
bins = edges(1:end-1) + diff(edges)/2; % To have the middle points of each bar
new_y = normpdf(bins,0,1); % To create a normalized Gaussian Dist.
plot(bins,new_y,'k-','linewidth',2)
set(gca,'fontname','times','fontsize',12)
ytickformat('%.2f') % 2 sig. figs on ytick label

print -djpeg -r600 Plot_Prob7
```

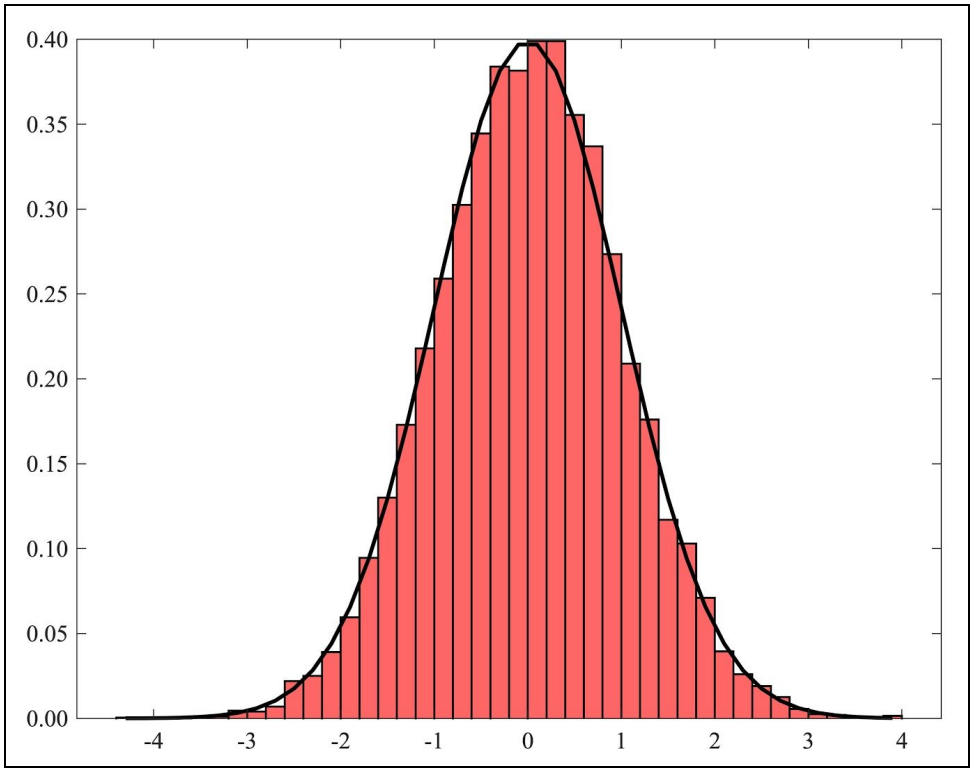

Figure 3: Histogram and Gaussian Distribution for Problem 7

PROBLEM 8

```
% * Perform power spectral density calculation using pwelch
% * Mess with the different parameters (window,noverlap,nfft)
% * Conclude the behavior
% * [Pxx,F] = pwelch(x,window,noverlap,nfft,fs)
```

Varies Window Value

```
t = 0:0.01:10000; % Fs = 1/0.01 = 100; sample frequency
y = 4*cos(2*pi*t) + sin(2*pi*t/0.2) + 0.01*randn(1,length(t)); % fs = 1,5 Hz
% No need to plot this, it's messy
fig_wind = figure('name','PWELCH Randomized WINDOW');
window = [10000,1000,50]; % Default 8
% # of segments I want MATLAB to chop the time series data
% BUT, MATLAB pwelch makes WINDOW value as the # of time series data one
% segment holds, not # of segments to be chopped into

for i = 1:length(window)
    s(i) = subplot(4,1,i);
    [Pxx,F] = pwelch(y,floor(length(y)/window(i)),[],[],100);
    % Rounding down using floor, ceil to round up
    semilogy(F,Pxx,'k-'); legend(['Window = ' num2str(window(i))])
    set(gca,'xticklabel',[],'fontname','times'); xlim([0 10])
end
s(4) = subplot(4,1,4);
[Pxx,F] = pwelch(y,[],[],[],100);
semilogy(F,Pxx,'k-'); legend('Window = 8 (Default)')
set(gca,'fontname','times'); xlim([0 10])

h = axes(fig_wind,'visible','off');
% * essentially what this means is making a new axis on the entire figure,
% but make in invisible (so it won't affect the entire plot)
% * restricts linkaxes and zooming the figure
h.XLabel.Visible = 'on';h.YLabel.Visible = 'on';
% making the label on for the entire figure altogether (structure method)
xlabel(h,'Frequency (Hz)','fontname','times','fontweight','bold');
ylabel(h,'Power Spectral Density (s^2/Hz)','fontname','times','fontweight','bold')
print -djpeg -r600 Pwelch_WindowRandom
```

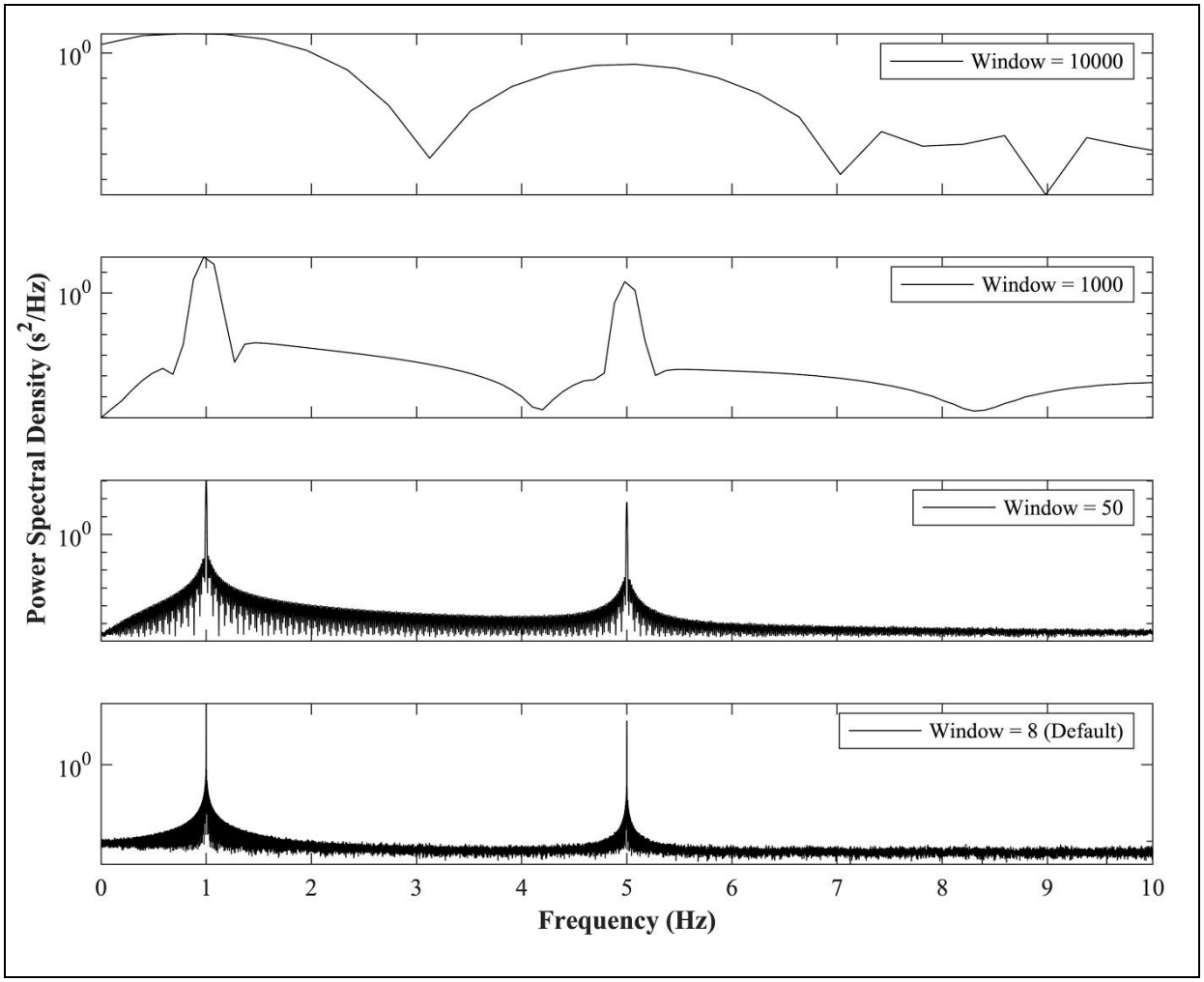

Figure 4: Power Spectral Density by Varying Windows' Values

Figure 4 above shows the effect of varying the window's values of the pwelch function's input. The analysis is only focusing on frequencies of 1 and 5 Hz. Each legend represents the number of segments I wish to store my time series data into for the analysis with pwelch function. MATLAB pwelch's window input parameter on the other hand will assume the input that was assigned represents the number of time series data that I wish to store in each segment. From this figure, the higher the number of segments (windows), the less resolved the peak is. Windows help to smoothen the curve (less noise), and it could be seen that window = 10000 is smoother than window = 8 (Default). However, this smoothness defeats the purpose of showing the actual finding, which are the peaks at the respective frequencies. This is because the smoothing happens by averaging the data values that were segmented based on the number of windows specified in the input. The higher the window's value, the smaller the number of data being segmented into the specified value (less data being averaged), which leads to less accuracy.

Varies Noverlap Value

```
fig_nover = figure('name','PWELCH Randomized NOVERLAP');
noverlap = [0.25,0.75,1]; % Default 50%; 0.5; less than 8 (window's default)

for i = 1:length(noverlap)
    subplot(2,4,i); % Plot focus on 1 Hz
    [Pxx,F] = pwelch(y,[],noverlap(i)*floor(length(y)/8),[],100);
    % WINDOW = floor(length(y)/8) is default
    semilogy(F,Pxx,'r-'); legend(['Noverlap = ' num2str(noverlap(i)*100) '%'])
    set(gca,'fontname','times'); xlim([0.95 1.05]); ylim([0 1e6])
    xlabel('Frequency (Hz)','fontname','times','fontweight','bold');
    ylabel('Power Spectral Density (s^2/Hz)','fontname','times','fontweight','bold')

    subplot(2,4,4+i); % Plot focus on 5 Hz
    semilogy(F,Pxx,'r-'); legend(['Noverlap = ' num2str(noverlap(i)*100) '%'])
    set(gca,'fontname','times'); xlim([4.95 5.05])
    xlabel('Frequency (Hz)','fontname','times','fontweight','bold');
    ylabel('Power Spectral Density (s^2/Hz)','fontname','times','fontweight','bold')
end
s(4) = subplot(2,4,4); % Plot focus on 1 Hz default
[Pxx,F] = pwelch(y,[],[],[],100);
semilogy(F,Pxx,'r-'); legend('Noverlap = 50% (Default)')
set(gca,'fontname','times'); xlim([0.95 1.05])
xlabel('Frequency (Hz)','fontname','times','fontweight','bold');
ylabel('Power Spectral Density (s^2/Hz)','fontname','times','fontweight','bold')

r(4) = subplot(2,4,8); % Plot focus on 5 Hz default
[Pxx,F] = pwelch(y,[],[],[],100);
semilogy(F,Pxx,'r-'); legend('Noverlap = 50% (Default)')
set(gca,'fontname','times'); xlim([4.95 5.05]);
xlabel('Frequency (Hz)','fontname','times','fontweight','bold');
ylabel('Power Spectral Density (s^2/Hz)','fontname','times','fontweight','bold')

set(gcf,'position',[229 95 993 638])

print -djpeg Pwelch_NoverlapRandom
```

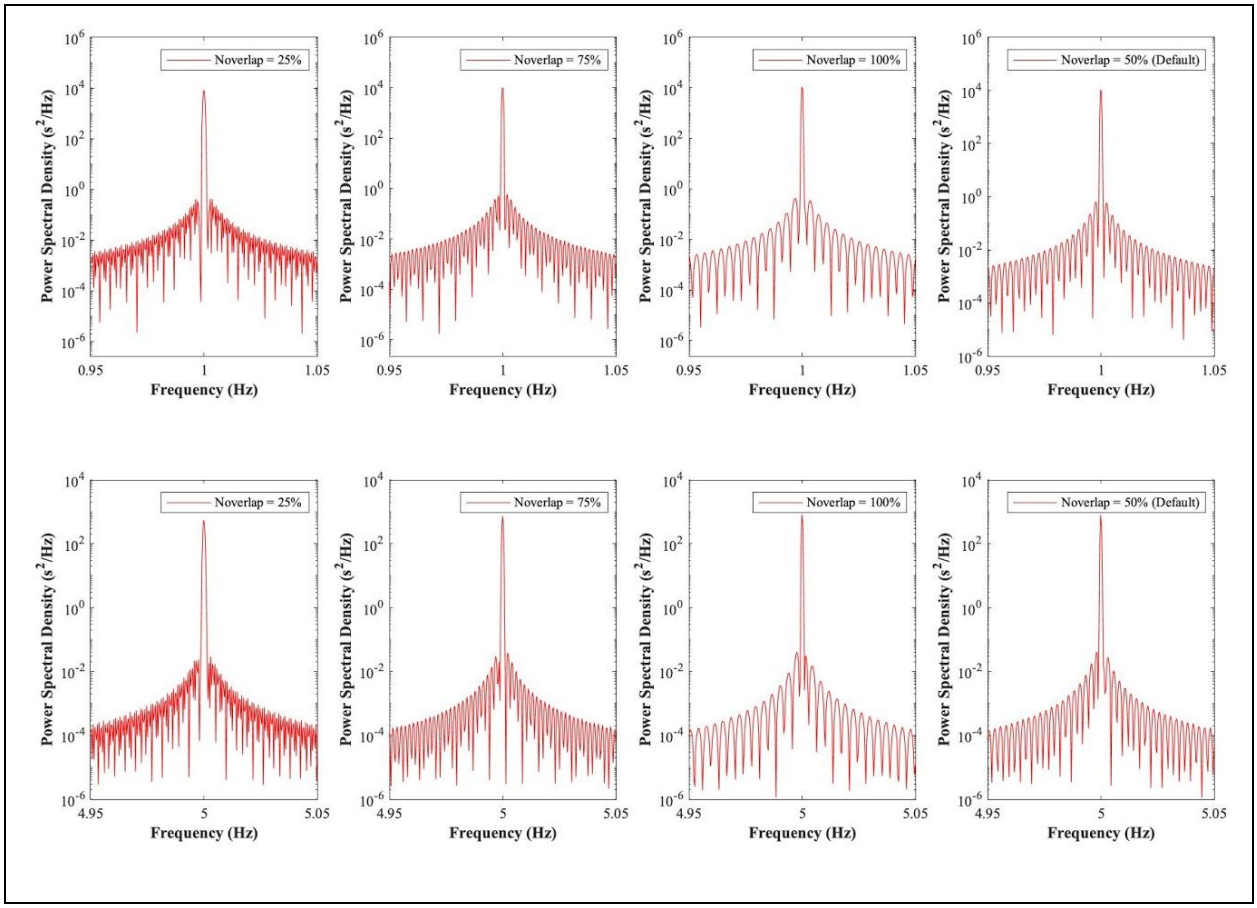

Figure 5: Power Spectral Density by Varying Noverlaps' Values

Figure 5 shows the effect of varying noverlaps' values on PSD. The top four plots show the peaks around 1 Hz, and the bottom four plots represent the peaks around 5 Hz. The default noverlap's value was 50 %, which means there is 50 % of overlapping between adjacent windows data. It could be clearly seen that the smaller the noverlap value, the higher the number of noise around the peaks. This is because the lines are less resolved and less overlapping on one another, which makes it difficult to observe. It is ensured that noverlap value is smaller than the value of window (scalar) or shorter in length than the window's length (vector).

Varies NFFT Value

```
fig_nfft = figure('name','PWELCH Randomized NFFT');
nfft = [128,512,16384]; % 2^7,9,14
for i = 1:length(nfft)
    s(i) = subplot(4,1,i);
    [Pxx,F] = pwelch(y,[],[],nfft(i),100);
    semilogy(F,Pxx,'b-'); legend(['NFFT = ' num2str(nfft(i))])
    set(gca,'xticklabel',[],'fontname','times'); xlim([0 10])
end
s(4)= subplot(4,1,4);
[Pxx,F] = pwelch(y,[],[],[],100);
semilogy(F,Pxx,'b-'); legend('NFFT = Default')
set(gca,'fontname','times'); xlim([0 10]);

h = axes(fig_nfft,'visible','off');
% * essentially what this means is making a new axis on the entire figure,
% but make in invisible (so it won't affect the entire plot)

h.XLabel.Visible = 'on';h.YLabel.Visible = 'on';
% making the label on for the entire figure altogether (structure method)

xlabel(h,'Frequency (Hz)','fontname','times','fontweight','bold');
ylabel(h,'Power Spectral Density (s^2/Hz)','fontname','times','fontweight','bold')

print -djpeg -r600 Pwelch_NFFTRandom
```

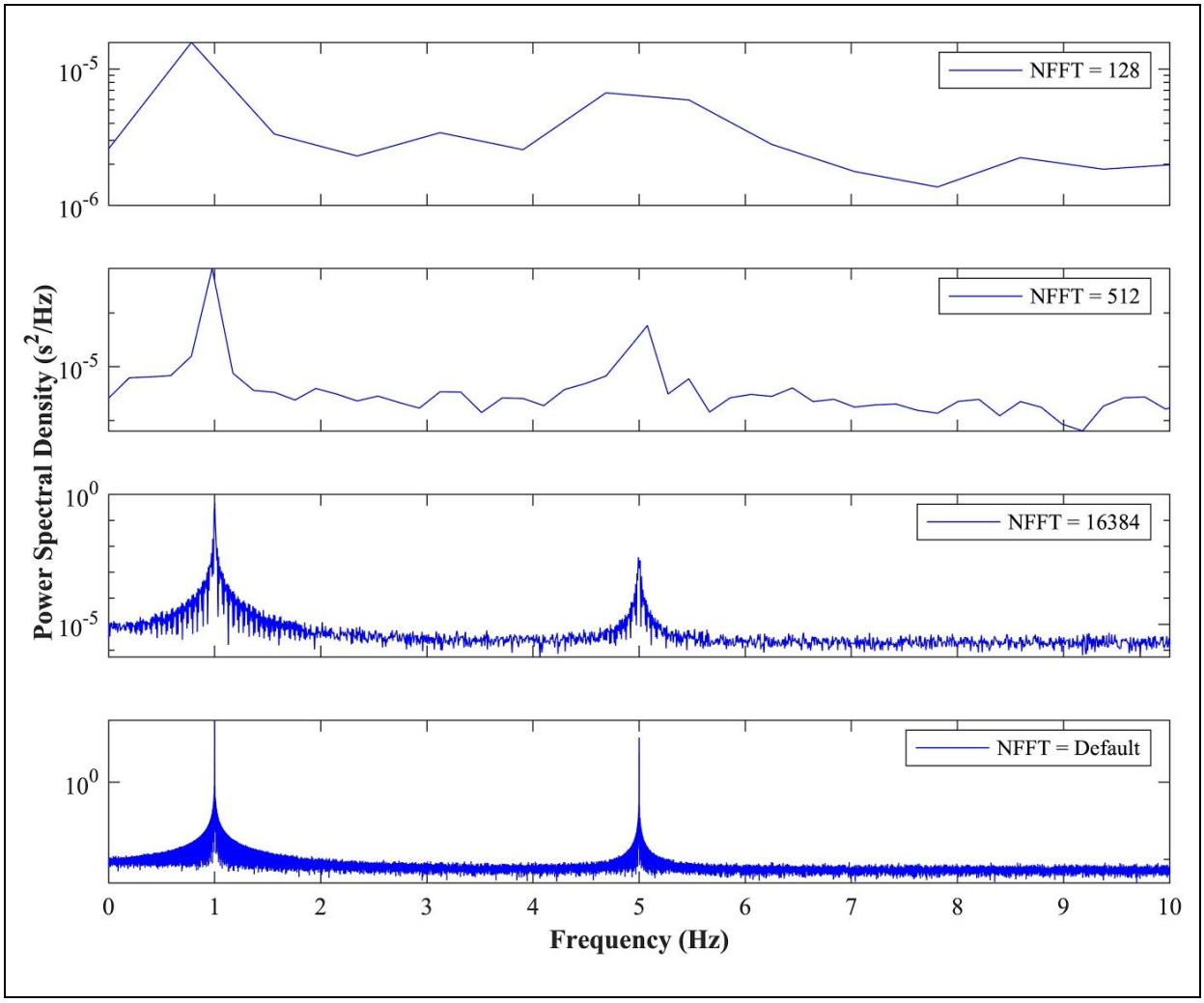

Figure 6: Power Spectral Density by Varying NFFTs' Values

Figure 6 shows the effect of varying NFFT on PSD. Unlike varying windows, it could be seen clearly the distinction in peaks on the respective frequencies. However, the major findings from this variation is that the width of the peaks are changing for different NFFT. As NFFT approaches 0, the broader the peaks become. This is associated with spectral leakage of the peak. Higher NFFT has less spectral leakage. It is ensured that NFFT's value is a factor of 2.