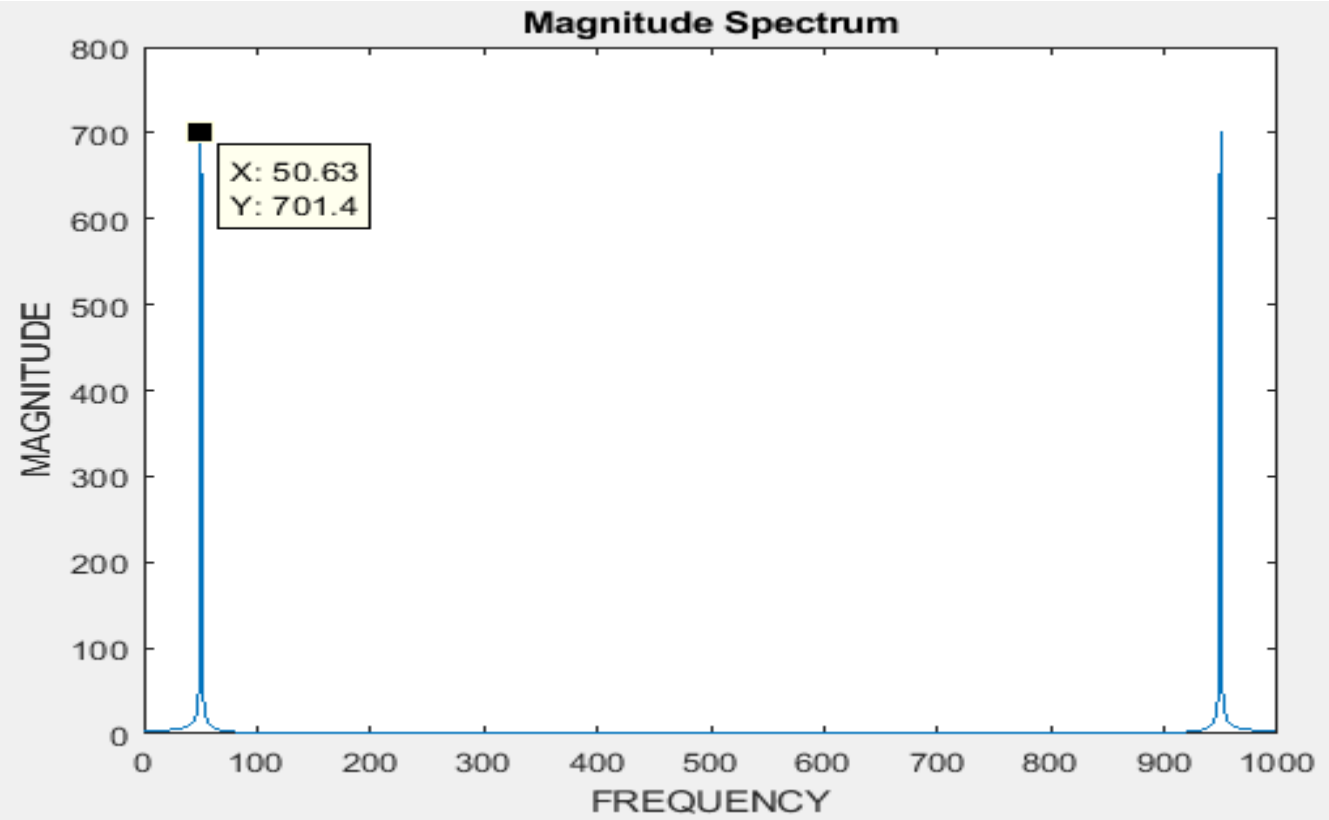
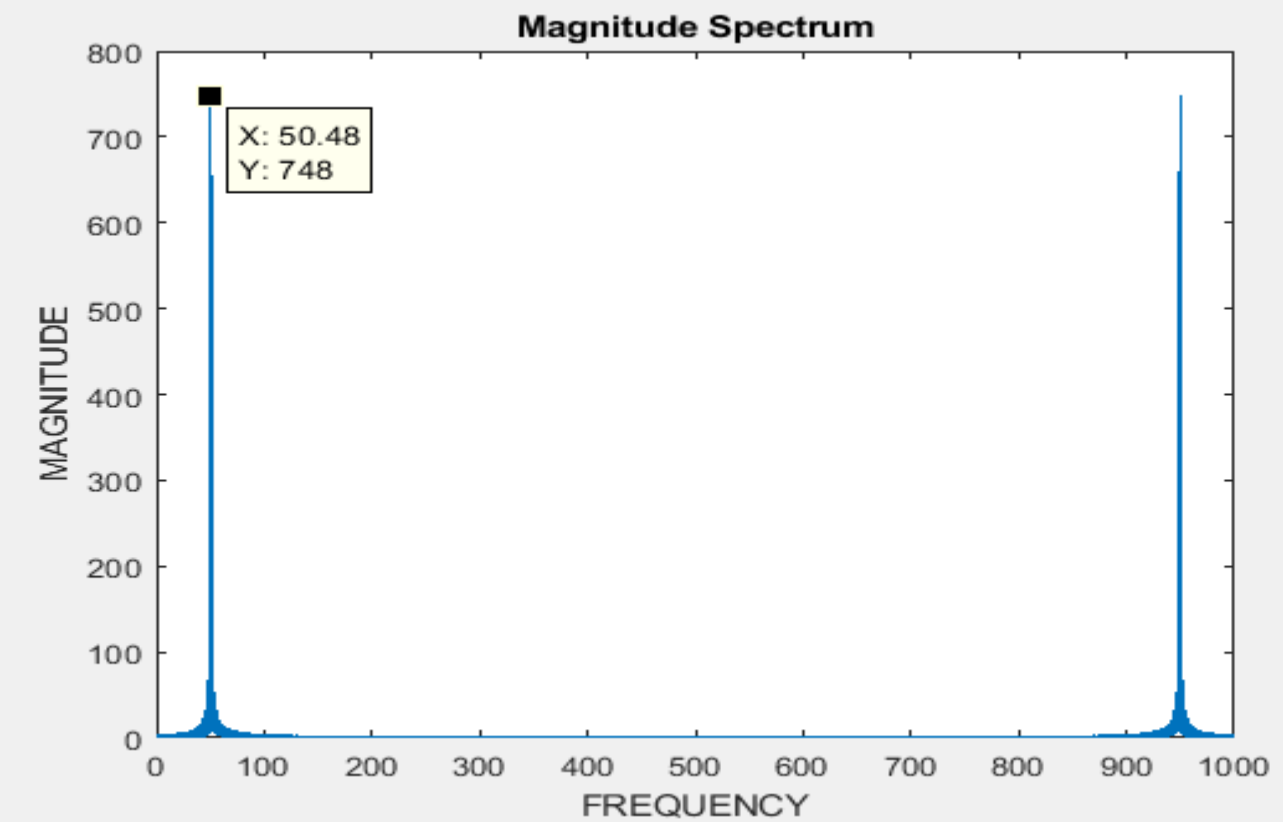
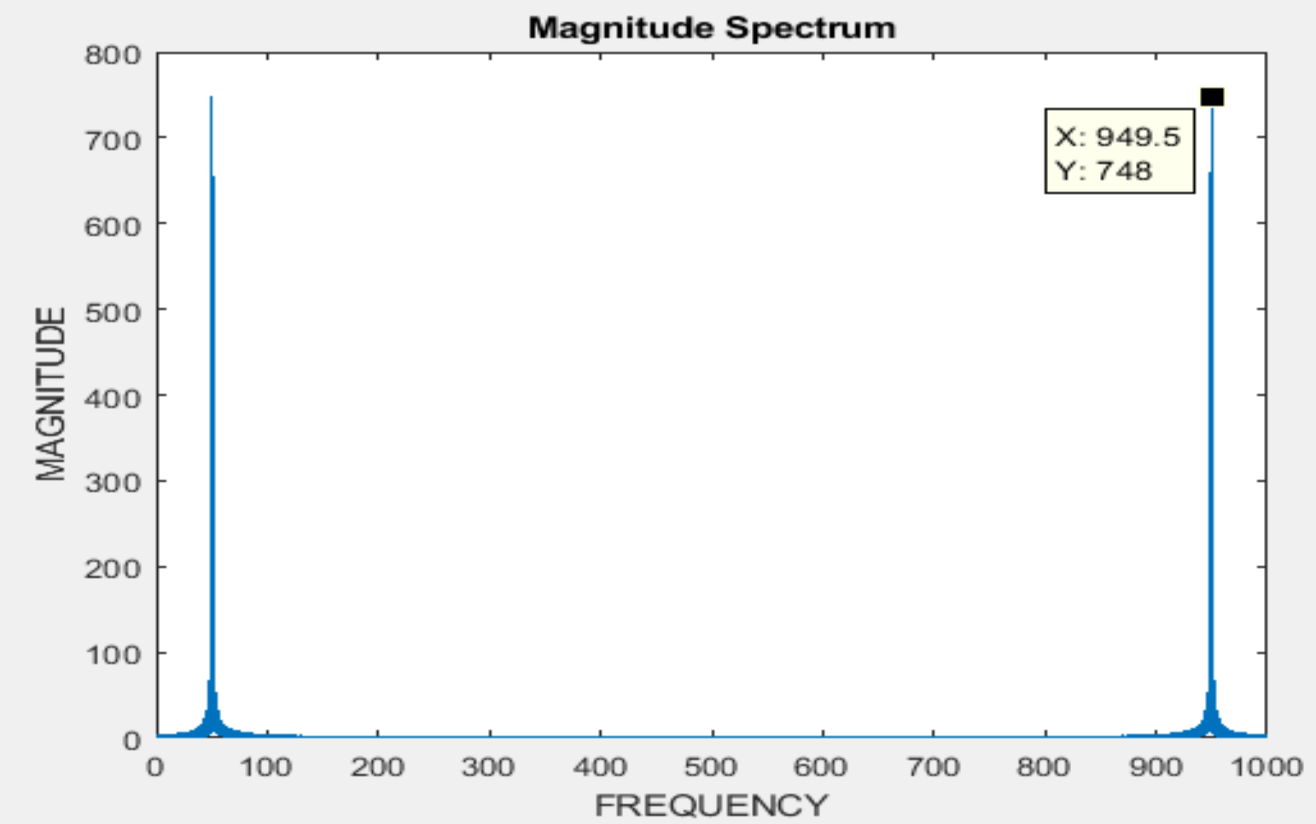
```
diary on
format compact
%Johnny Li
%EEL3135 Fall 2018
%Lab 8 Part 2

%2.1.1

%Construct the vector x to represent the given sinusoid.
%Given sample frequency.
fs=1000;
%Time interval.
tt=0:1/fs:1.5;
%Signal value
xx=cos(2*pi*50.5*tt+0.25*pi);

%2.1.2

%Plot the magnitude spectrum.
%Run the fft(x) function on x.
X=fft(xx);
%FFT of X against the Hertz frequencies of the bins.
ww = 0:(2*pi/length(X)):(2*pi-1/length(X));
freq = ((ww)*fs)/(2*pi);
%Create a new vector that stores the values of abs(X) and plot it.
%Plot Magnitude
plot(freq, abs(X));
ylabel("MAGNITUDE")
xlabel("FREQUENCY")
title("Magnitude Spectrum")

%Find the frequency bins where this maximum occurs.
%The maximum occurs at frequency bin 50.63.
```



```
%Question: Does this exactly match the frequency 50.5 Hz?
%No, the frequency bins are 50.63 Hz, which is really close to matching the
%frequency 50.5 Hz but not exactly it.

%Append 15,000 zeros to the signal.
xx = [xx zeros(1,15000)];
%Run the FFT again.
X=fft(xx);

%Question: What is the effect of the zero padding on the magnitude spectrum
%of the signal?
%The effect of the zero padding on the magnitude spectrum is that it went
%through more samples, causing the lines to thicken as more values are
%calculated,and producing a more accuracy result as the frequency bins are
%closer to 50.5 Hz.
```

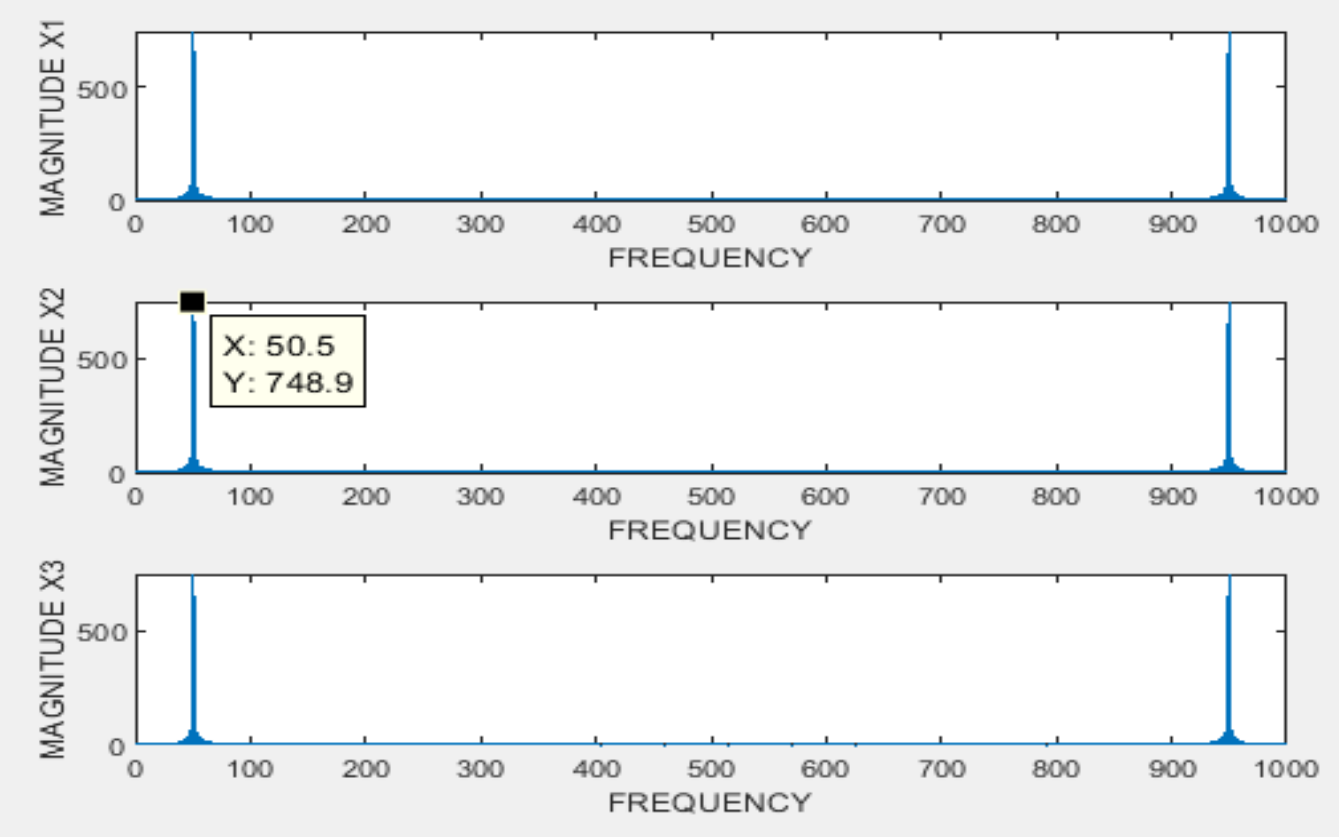



```
%Question: What is the frequency of the bin for which the magnitude is
%maximized?
%The frequency of the bin for which the magnitude is maximized is 50.48 Hz
%and 949.5 Hz.
```

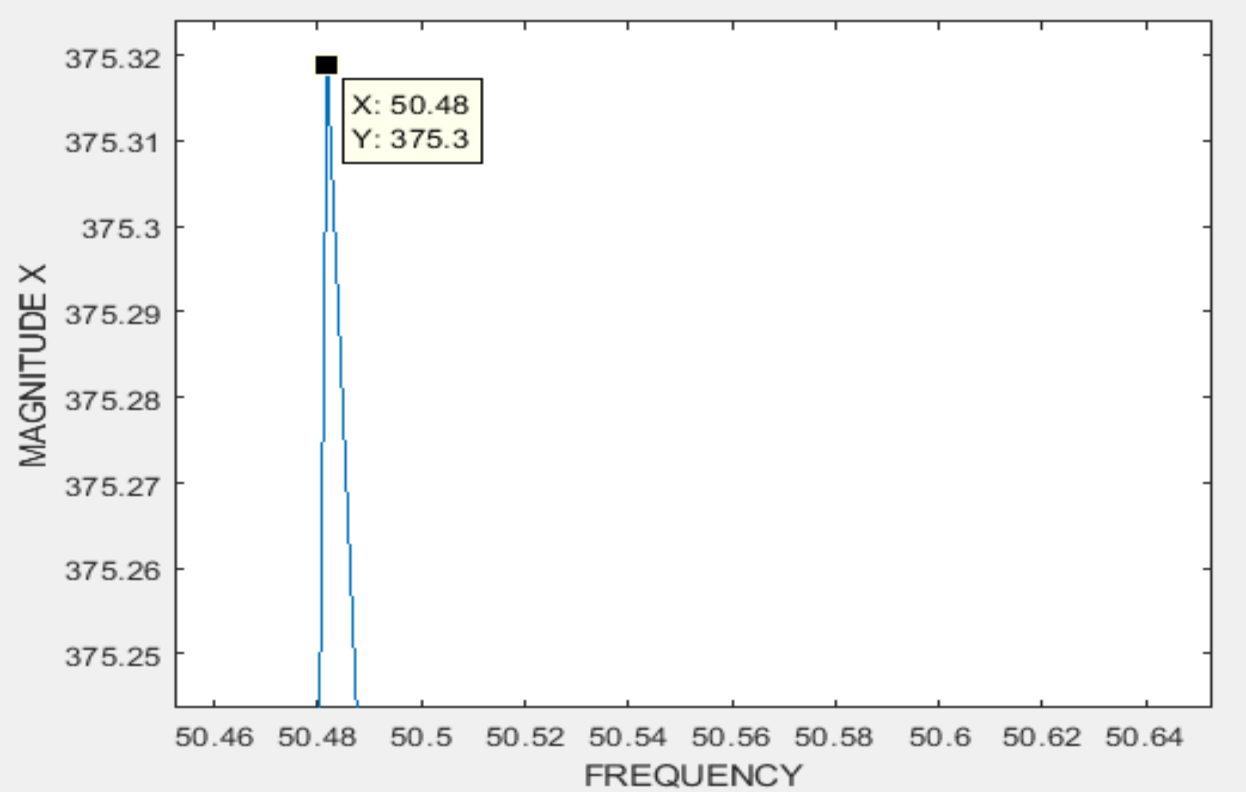



```
%2.2.2

%Question: How does the width of these side lobes compare to the main lobe?
%The width of these side lobes is smaller than the main lobe and gets
%smaller as the lobes appears farther from the main lobe.

%2.2.3

%Padding the signal with 25,000 zeros, 50,000 zeros, and 100,000 zeros.
%Compare the appearances of the magnitude spectrum.
%Given sample frequency.
fs=1000;
%Time interval.
tt=0:1/fs:1.5;
%Signal value
xx=cos(2*pi*50.5*tt+0.25*pi);

%Append 15,000 zeros to the signal.
xx1 = [xx zeros(1,25000)];
%Run the FFT again.
X1=fft(xx1);
xx2 = [xx zeros(1,50000)];
%Run the FFT again.
X2=fft(xx2);
xx3 = [xx zeros(1,100000)];
%Run the FFT again.
X3=fft(xx3);
```

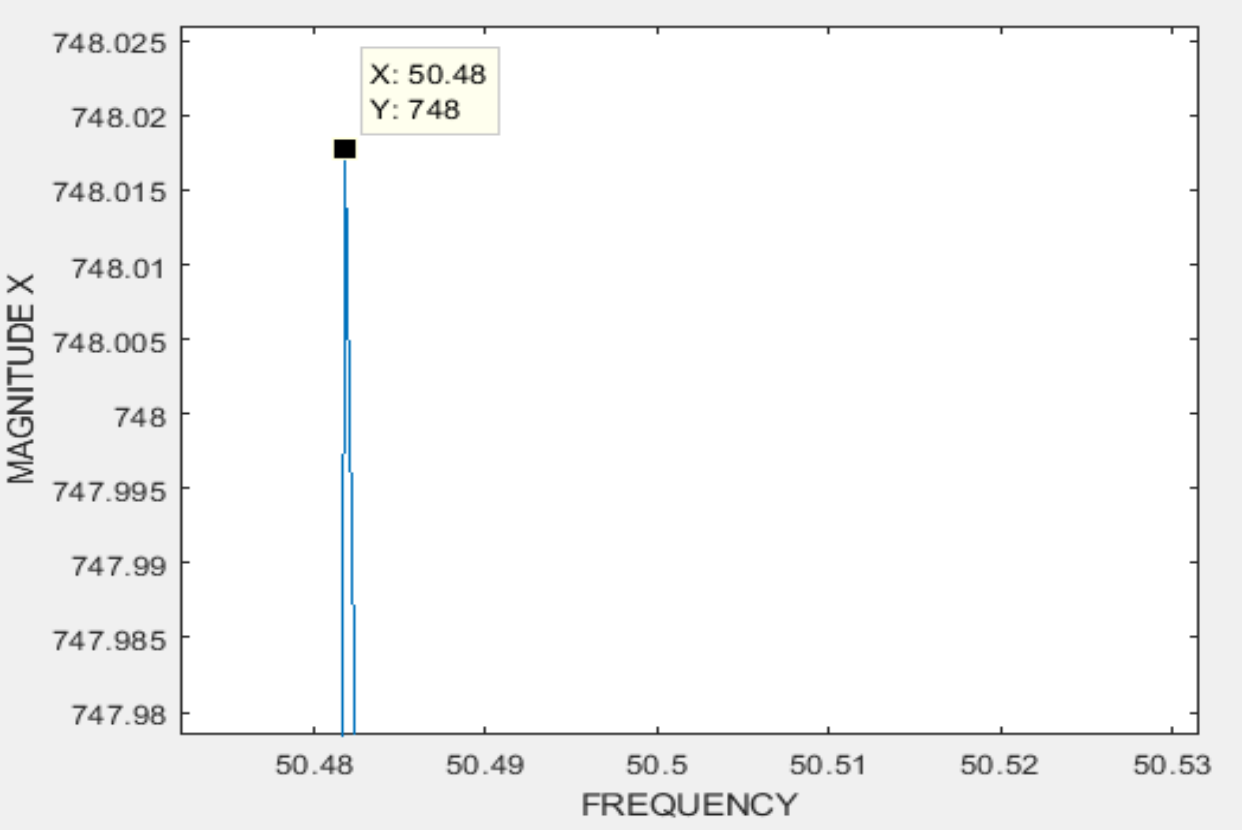
```
%FFT of X against the Hertz frequencies of the bins.
ww1 = 0:(2*pi/length(X1)):(2*pi-1/length(X1));
freq1=(fs*ww1)/(2*pi);

ww2 = 0:(2*pi/length(X2)):(2*pi-1/length(X2));
freq2=(fs*ww2)/(2*pi);

ww3 = 0:(2*pi/length(X3)):(2*pi-1/length(X3));
freq3=(fs*ww3)/(2*pi);

%Plot Magnitude
subplot(3,1,1)
plot( freq1,abs(X1) )
ylabel("MAGNITUDE X1")
xlabel("FREQUENCY")
subplot(3,1,2)
plot( freq2,abs(X2) )
ylabel("MAGNITUDE X2")
xlabel("FREQUENCY")
%Magnitude X2
subplot(3,1,3)
plot( freq3,abs(X3) )
ylabel("MAGNITUDE X3")
xlabel("FREQUENCY")
```



```
%Question: What is the effect of increasing the number of zeros in the
%accuracy of the frequency calculation?
%The effect of increasing the number of zeros is that it reduces the error
%in the frequency bins and thus increase the accuracy of the frequency
%calculation.

%Question: Does the error increase or decrease?
%The error decreases as the number of the number of zeros increases.

%Question: In the case of our signal, how many zeros would need to be added
%to obtain an exact value for the frequency of our input signal?
%50,000 zeros because that was what I found to give me an exact value for the
%frequency of the output signal.

%2.3

%Window x(t) with a Hanning window.
%Given sample frequency.
fs=1000;
%Time interval.
tt=0:1/fs:1.5;
%Signal value
xx=cos(2*pi*50.5*tt+0.25*pi);

%Hanning window.
xx = xx.*hanning(length(xx))';
%Append Zeros
xx = [xx zeros(1,15000)];
%Run the FFT again.
X=fft(xx);

%FFT of X against the Hertz frequencies of the bins.
ww = 0:(2*pi/length(X)):(2*pi-1/length(X));
freq=(fs*ww)/(2*pi);

%Plot Magnitude
plot( freq,abs(X) )
ylabel("MAGNITUDE X")
xlabel("FREQUENCY")

%2.3.2

%Plot the magnitude spectrum of after windowing and zero padding.
```



%Plot the magnitude spectrum of without windowing.



```
%Question: What effect does windowing have on the FFT?
%The effect that windowing has on the FFT is that it removes the lobes.
%Question: How does the magnitude of the maximum peak change with
windowing?
%The magnitude of the maximum peak is almost cut in half with windowing,
%going from 748 to 375.3.

%Question: Is it higher or lower than that of the maximum peak without
%windowing?
%The magnitude of the maximum peak is lower than that of the maximum peak
%without windowing.

%Question: Has the width changed at all?
%The width of the maximum peak with windowing has increased to be larger
%than of the width of the maximum peak without windowing.

%2.4.1

%Load the ECG data.
load('heartratelabdata.mat')

%2.4.2

%Divide the signal into subsamples of equal width, using a window about 2
%minutes long.
function out = Get_AverageHeart_Rate(in)
%Get_AverageHeart_Rate calculate the average heart rate.

%Given Sample
fs = 125;
%Total samples = 60sec *60min *125 Hz = 450,000.
N=450000;

% 2.4.3

%Iterate through the windows and take the FFT for each window.It is
%suggested that there be an overlap between each window as the function
%iterates.
for i = 1 : 30
%Initial run
    if(i == 1)
    %Store the approriate value
        y = input(1 : 15000+1250);
    else
        %Creates overlay
        y = input(15000*(i-1)-1250 : 15000*i);
    end

%2.4.4

%Take the absolute value of the FFT of the sample and identify the peak
%closest to bin zero, but not at zero.
%Perform the windowing using the hanning function.
y = y.*hanning(length(y))';
%Append Zeros
```

```
yP = [y zeros(1,15000)];
%Run the FFT again.
in = fft(yP);
%Create the magnitude vector
magn = abs(in);

%2.4.5

%Record the index/bin of the peak you identified. Convert this bin number
%to frequency in Hz, or beats per second. Multiply this frequency by 60 to
%obtain the heart rate, in beats per minute, for each window.
%Lower edge rate
HRL = 60;
%Upper edge rate
HRU = 100;
%Lower Limit
LowLimit = HRL*(length(magn))/(fs*60);
%Higher Limit
HighLimit = HRU*(length(magn))/(fs*60);
%Setting Limits, TA code
%Storing the results
[~,index] = max(magn(LowLimit : HighLimit));

%2.4.6

%Store these average heart rates into a vector, and then produce a plot
%showing the change in the average heart rate over time.
%Output vector
out(i) = (index + LowLimit)*(fs*60)/(length(magn));
end
end


%2.4.7

%Write the above into a function that takes a data vector as input,
%calculates the heart rate foreach two-minute window, and plots heart rate
%over time.
function HeartRate = PlotAHR(am101,am105,am107,cm110)
%Plot heart rate.

%am101
AM101 = Get_AverageHeart_Rate(am101);
subplot(4,1,1);
plot(AM101);
title('The Average Heart Rate: am101');
ylabel('AHR');

%am105
AM105 =Get_AverageHeart_Rate(am105);
subplot(4,1,2);
plot(AM105);
title('The Average Heart Rate: am105');
ylabel('AHR');
```

```
%am107
AM107 = Get_AverageHeart_Rate(am107);
subplot(4,1,3);
plot(heartrateAM107);
title('The Average Heart Rate: am107');
ylabel('AHR');

%cm110
CM110 = Get_AverageHeart_Rate(cm110);
subplot(4,1,4);
title('The Average Heart Rate: cm110');
ylabel('AHR');
xlabel('Time');

%2.4.8

%Use that function to produce the four plots, one for each signal.
```