Taylor Rembos
EEL 3135
Lab 8 Part 2

**Lab 9: An Introduction to Using the FFT Function in MATLAB**

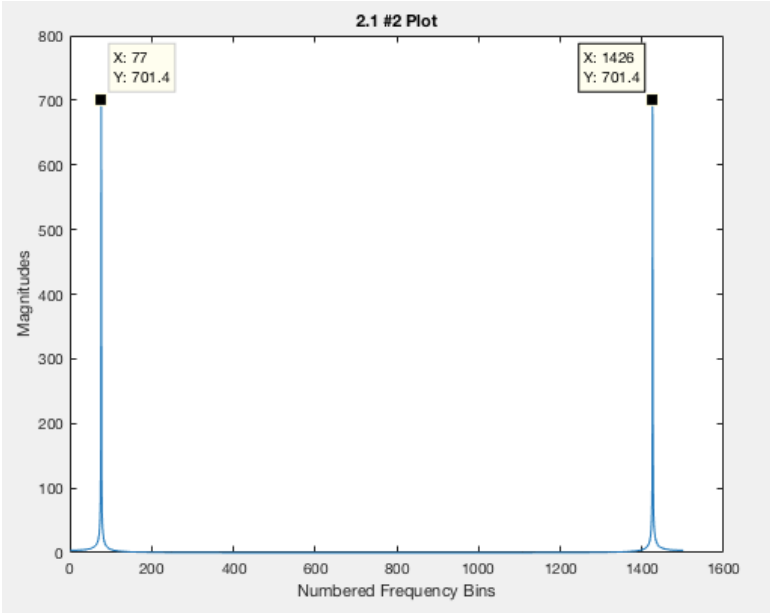
## 2.1 FFT of a Sinusoid with Non-Integer Frequency

$$x(t) = \cos(2\pi(50.5)t + 0.25\pi)$$

**1) Using 1000 samples/second for fs and a time vector of 1.5 seconds, construct the vector xx to represent the given sinusoid.**

**2) Use fft()and assign the output to XX. Plot the magnitude spectrum. Zoom on the left-most peak. You should notice that the spectrum is nonzero at multiple values surrounding this peak. By introducing a non-integer frequency, we see a *spread of energy* across the spectrum. Find the frequency bins where this maximum occurs. Does this exactly match the frequency 50.5 Hz?**

```
>> A = 1;
>> phi = 0.25*pi;
>> fs = 1000;
>> f = 50.5;
>> t = 0:(1/fs):1.5;
>> xx = cos(2*pi*f*t + phi);
>> XX = fft(xx);
>> plot(abs(XX))
>> xlabel('Numbered Frequency Bins')
>> ylabel('Magnitudes')
>> title('2.1 #2 Plot')
```




The left-most maxiumum occurs at a frequency bin value of 77. It does not exactly match the value with a sampling error of about 38%.

## 2.2 Zero Padding

**Zero padding works by concatenating a large sample of zeros (using the zeros(1,N) function, where N is the number of zeros appended) at the end of the signal being analyzed. For example,**

**suppose our signal is $x(t)=3\cos(2\pi(2.5)t+\pi/4)$ . The following script uses vector notation to**
**add 15,000 zeros to the end of xx. Here the sampling frequency is 1000 samples/second and t is a time vector of 1500 samples. The results are plotted below:**

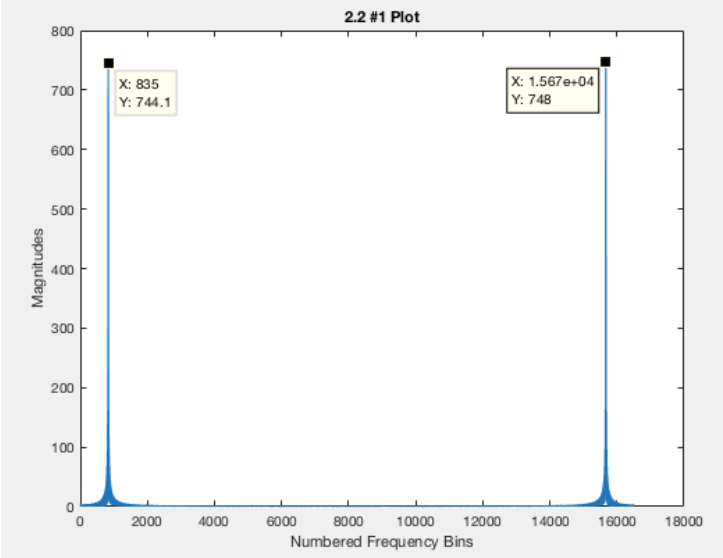
```
xx = 3*cos(2*pi*2.5*t+ pi/4);

xx = [xx zeros(1,15000)];
```

**Now that you have some background information, complete the following steps:**
**1) Append 15,000 zeros to our signal from section 1.7 (the one with frequency 50.5 Hz), and run the FFT again. What is the effect of the zero padding on the magnitude spectrum of the signal? What is the frequency of the bin for which the magnitude is maximized?**

**2) The appearance of the magnitude spectrum may be somewhat odd compared to what we have seen before. There should appear to be a main "lobe" and several smaller "side lobes" on the sides. How does the width of these side lobes compare to the main lobe?**

**3) Try zero padding the signal with 25,000 zeros, 50,000 zeros, and 100,000 zeros. Using the subplot function, compare the appearances of the magnitude spectra. What is the effect of increasing the number of zeros in the accuracy of the frequency calculation? Does the error increase or decrease? In the case of our signal, how many zeros would need to be added in order to obtain an exact value for the frequency of our input signal?**

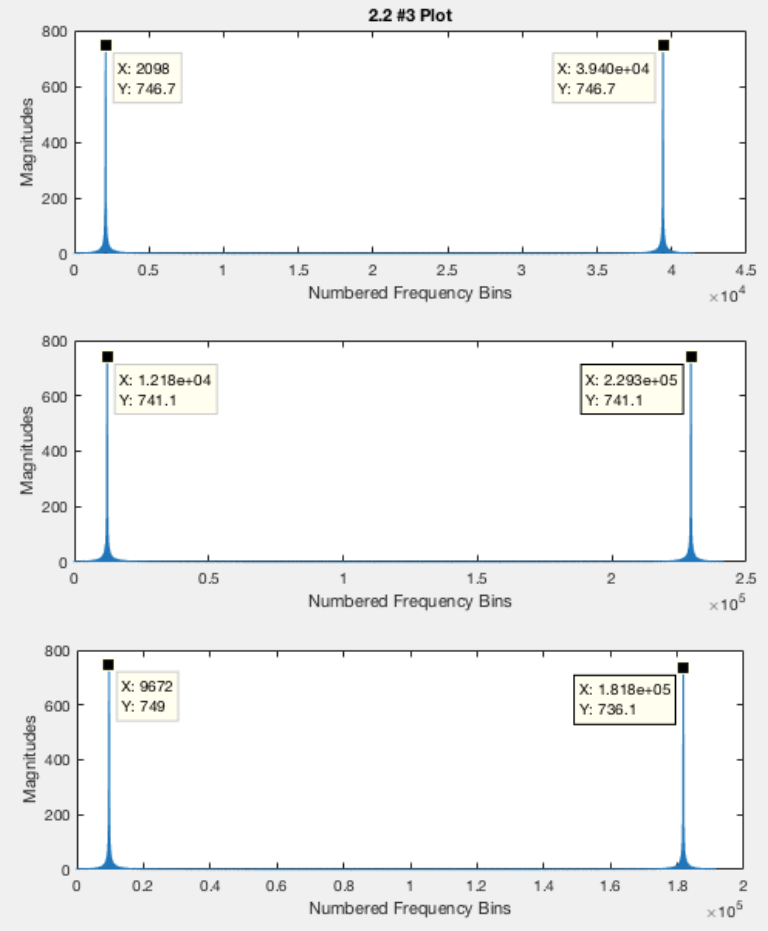
```
>> A = 1;
>> phi = 0.25*pi;
>> fs = 1000;
>> f = 50.5;
>> t = 0:(1/fs):1.5;
>> xx = cos(2*pi*f*t + phi);
>> xx = [xx zeros(1,15000)];
>> XX = fft(xx);
>> plot(abs(XX))
>> xlabel('Numbered Frequency Bins')
>> ylabel('Magnitudes')
>> title('2.2 #1 Plot')
```



The past plot had the left-most maximum value at a frequency bin of 77 with a magnitude of 701.4. This plot slightly increases the magnitude to 744.1 at a frequency bin of 835. Also, as seen in the graph, the line is thicker around the values leading up to and following the maximum values, the “lobes”, meaning more concentrated values as well. For the right most peak, at a bin of 1.567e+04, the magnitude is 748.

The side lobes are much thicker compared to the main lobe, meaning a higher saturation of values at those locations.

```
>> xx = [xx zeros(1,25000)];
>> XX = fft(xx);
>> plot(abs(XX))
>> xlabel('Numbered Frequency Bins')
>> ylabel('Magnitudes')
>> title('2.2 #3 Plot')
>> subplot(2,1,1)
>> plot(abs(XX))
>> xlabel('Numbered Frequency Bins')
>> ylabel('Magnitudes')
>> title('2.2 #3 Plot')
>> xx = [xx zeros(1,50000)];
>> XX = fft(xx);
>> subplot(2,1,2)
>> plot(abs(XX))
>> xlabel('Numbered Frequency Bins')
>> ylabel('Magnitudes')
>> xx = [xx zeros(1,100000)];
>> XX = fft(xx);
>> subplot(2,1,2)
>> plot(abs(XX))
>> xlabel('Numbered Frequency Bins')
>> ylabel('Magnitudes')
```




The appearance remains consistent as well as the magnitude. The numbered frequency bins differ with 25,000 zeros having a left-most peak of 2,098 with magnitude 746.7 and right-most 39,400 with magnitude 746.6, 50,000 zeros having a left-most peak of 12,180 with magnitude 741.1 and right-most 229,300 with magnitude 741.1, and 100000 zeros having a left-most peak of 9,672 with magnitude 749 and right-most 181,800 with magnitude 736.1. In the accuracy of frequency, the increasing of zeros appears to make the frequency closer and closer to being accurate with a fall off point where it then makes it less accurate. The error increases given my plots although I believe the error should decrease if the zeros function is used correctly. The number of zeros that would need to be added for our frequency to be accurate is around 150,000 zeros.

## 2.3 Windowing

**Suppose we have the signal $x(t)=3\cos(2\pi(2.5)t+\pi/4)$ which we padded with zeros in the previous section. If we multiply our signal with a bell-shaped function (the *window*) to reduce the amplitude of the endpoints of the function, we obtain a signal that fades in and out. In this lab we will use the Hann Window, similar to the Hamming window from previous labs, only differing by a constant. The following code implements the Hann Window:**

```
xx = 3*cos(2*pi*2.5*t+ pi/4);
xx = xx.*hanning(length(xx))';
```

**Now, based on that background information, complete the following steps:**

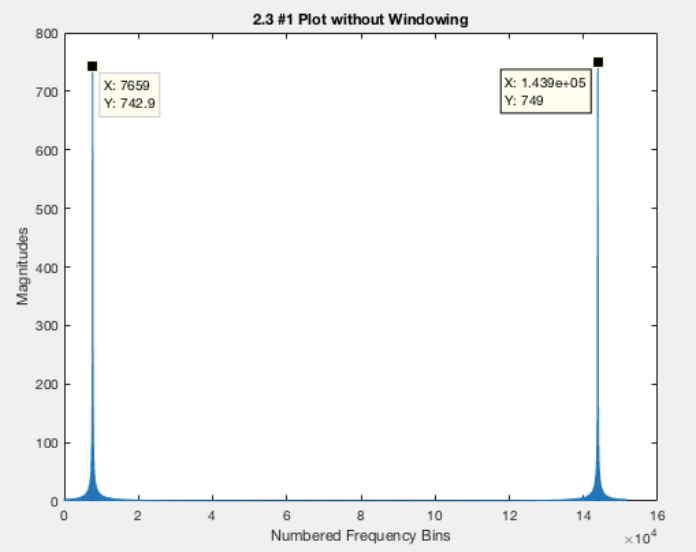
**1) Returning to our signal $x(t)=\cos(2\pi(50.5)t+0.25\pi)$ from the past two sections, we want to reduce the size of the "side lobes" in the plot of the magnitude spectrum. Using the script above, window $x(t)$ with a Hanning window.**
***Note: this needs to be done before zero padding!*** **As before, use a sampling frequency of 1000 samples/second and a time vector of 1.5 seconds.**

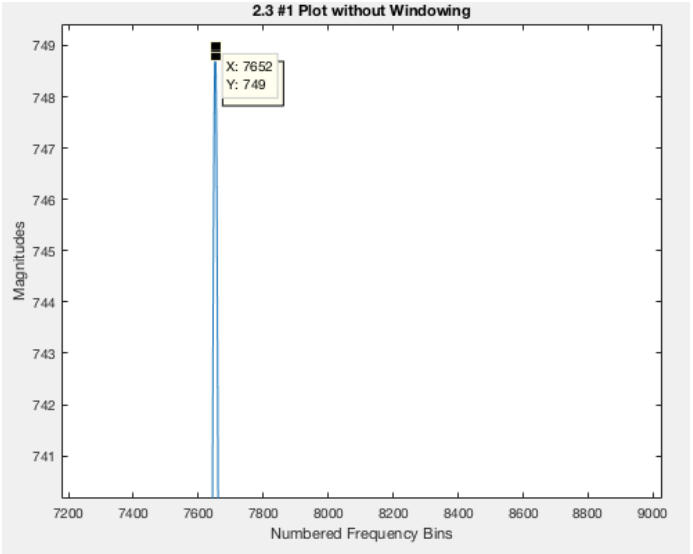
**2) Padding with an appropriately large number of zeros, plot the magnitude spectrum of**
**$x(t)$ magnitude spectra, zoomed in at the peak value, for XX with windowing and XX without windowing. What effect does windowing have on the FFT? How does the magnitude of the maximum peak change with windowing? Is it higher or lower than that of the maximum peak without windowing? Has the width changed at all?**

```
>> %without windowing
>> A = 1;
>> phi = 0.25*pi;
>> fs = 1000;
>> fs = 1000;
>> f = 50.5;
>> t = 0:(1/fs):1.5;
>> xx = cos(2*pi*f*t + phi);
>> xx = [xx zeros(1,150000)];
>> XX = fft(xx);
>> plot(abs(XX))
>> xlabel('Numbered Frequency Bins')
>> ylabel('Magnitudes')
>> title('2.3 #1 Plot without Windowing')

>> %with windowing
>> A = 1;
phi = 0.25*pi;
```

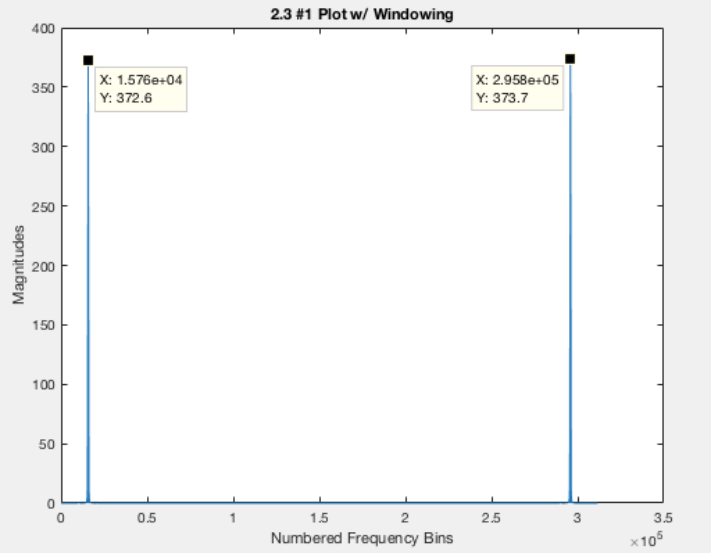
```
fs = 1000;
f = 50.5;
t = 0:(1/fs):1.5;
xx = cos(2*pi*f*t + phi);
xx = xx.*hanning(length(xx))';
xx = [xx zeros(1,150000)];
XX = fft(xx);
plot(abs(XX))
xlabel('Numbered Frequency Bins')
ylabel('Magnitudes')
>> title('2.3 #1 Plot w/ Windowing')
```

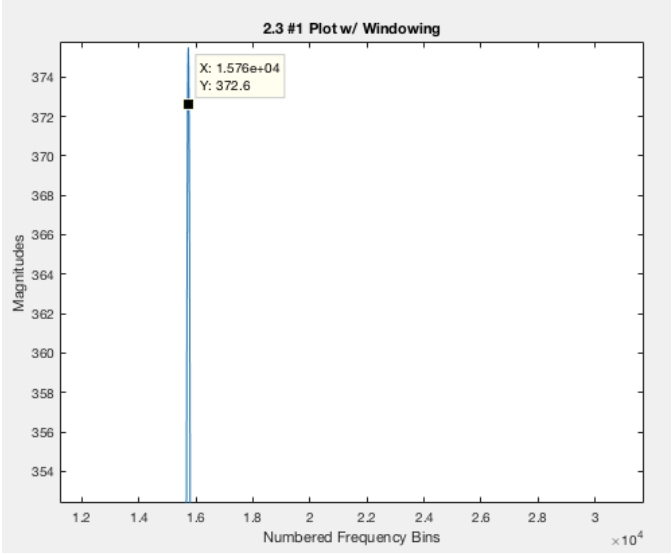
Without windowing and zoomed in




With windowing and zoomed in



Windowing eliminates the saturated values forming oscillations that create the lobes on the side of the peak values, as you can see from the plots. The magnitude of the maximum-peak increases, with windowing from 7658 without windowing to 15,760 with windowing. The width has changed as well.

## 2.4 Determining Average Heart Rate Using FFT

**In this exercise, you will employ the fft() function in MATLAB to automatically extract the heart rate from electrocardiogram (ECG) measurements. The data file on Canvas, heartratelabdata, contains four ECG signals (obtained from http://physionet.org/cgi- bin/atm/ATM). The sampling rate for all data files is 125 Hz. Your goal will be to use the Fourier transform to produce four plots, showing the average heart rate changing over each signal's duration.**

**Key Steps to a Solution**

***Note: You are allowed and encouraged to use anything you have learned to help the FFT identify the heart rate. This may include zero padding, windowing, filters, etc. In addition, you can choose whatever window length or overlap length you feel will give the best result.***

**1. Load each signal in MATLAB: am101, am105, am107, and cm110.**

**2. In this step, you will program a function or write a MATLAB script to read each signal and determine the average heart rate. Since each signal is *one hour* long, you will need to divide the signal into subsamples (or 'windows') of equal width. It is suggested you start by using a window about 2 minutes long. DO THE MATH in order to condense it to 2 minutes with even windows. (*Note:* different window sizes may be needed for a better result)**

**3. In order to plot the average heart rate over time, you will need to 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. This overlap is suggested to start to be about 10 seconds. (*Note:* different overlap sizes may be needed for a better result)**

**4. Take the absolute value of the FFT of the sample (abs(XX)) and identify the peak closest to bin zero, but not at zero. You are looking for the *first harmonic*. Your algorithm should automatically identify the first harmonic each time the function iterates, because it is this peak that stores the frequency information of the heart rate. Note that this may not be the highest peak in your spectrum; often, in the real world, the second or third harmonic, with frequency two or three times the frequency of the first harmonic, may be louder.**

**5. Record the index/bin of the peak you identified. Convert this bin number to frequency in Hz, or beats per second. Note that this may be near one beat per second. Multiply this frequency by 60 to obtain the heart rate, in beats per minute, for each window.**

**6. Store these average heart rates into a vector, and then produce a plot showing the change in the average heart rate over time.**

**7. Write the above into a function that takes a data vector (am101, am105, am107 or cm110) as input, calculates the heart rate for each two-minute window, and plots heart rate over time.**

**8. Use that function to produce the four plots, one for each signal.**

```
>> load('heartratelabdata.mat')
>> load('heartratelabdata.mat', 'am101')
>> load('heartratelabdata.mat', 'am105')
>> load('heartratelabdata.mat', 'am107')
>> load('heartratelabdata.mat', 'cm110')
>> XX101 = fft(am101);
>> XX105 = fft(am105);
>> XX107 = fft(am107);
>> XX110 = fft(cm110);
>> PKS101 = findpeaks(abs(XX101));
>> PKS105 = findpeaks(abs(XX105));
>> PKS107 = findpeaks(abs(XX107));
>> PKS110 = findpeaks(abs(XX110));
>> subplot(2,2,1)
>> plot(PKS101)
>> title('am101')
>> subplot(2,2,2)
>> plot(PKS105)
>> title('am105')
>> subplot(2,2,3)
>> plot(PKS107)
>> title('am107')
>> subplot(2,2,4)
>> plot(PKS110)
>> title('cm110')
```

```
function [ heartrate ] = Find_signal(heartrate)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

index = 1;
num = 2;
bin = [];
```

```
for bin
    XX = fft(heartrate);
    PKS = findpeaks(abs(XX));
    bin + 1;
end
end
```