Assignment 3

1. At the MATLAB command prompt, type `load sample123`. If done properly you now have a variable named `sample`, which is a sampled version of a function $x(t)$ of the form

$$x(t) = \sum_{n=1}^{N} A_n cos(2\pi f_n t)$$

The sampling rate is 8000Hz. Find $x(t)$ as accurately as you can by saying what are the values of N, A_n, and f_n. Include a properly labeled plot of the spectrum of $x(t)$, along with your code and a written outline of what you have done. **Please remember to include the name of your file in the submission.**

2. Consider the 2π periodic function $H(\hat{\omega})$, whose definition on the interval $-\pi \leq \hat{\omega} \leq \pi$ is

$$H(\hat{\omega}) = \begin{cases} 1 & a \leq |\hat{\omega}| \leq b \\ 0 & \text{otherwise} \end{cases}$$

where a and b are numbers such that $0 < a < b < \pi$. This function could serve as the frequency response function of an *ideal band-pass filter.*

   (a) Explain why H can be interpreted as an ideal band-pass filter. What is the role of the parameters $a,\ b$?

   (b) Calculate c_n, the (complex) Fourier series coefficients

   (c) Let a = $pi/3$ and b = $2\pi/3$. Plot $H(\hat{\omega})$ and the real part of the truncated Fourier Series

$$H_{32}(\hat{\omega}) = \sum_{-32}^{32} c_n e^{in\omega}$$

   for $0 \leq \hat{\omega} \leq \pi$ on the same axes.

   (d) On a different set of axes plot $20\log_{10}(|H_{32}(\hat{\omega})|)$ over the same range.

   (e) We know that when a function has a jump discontinuity, the truncated Fourier series does a poor job of reconstructing the function near the jump discontinuity. One way to remedy this is with various *windows.* Instead of using the truncated Fourier Series to approximate $H(\hat{\omega})$,

$$H_N(\hat{\omega}) = \sum_{-N}^{N} c_n e^{in\hat{\omega}},$$

we *window* the Fourier coefficients c_n, i.e., we modify them via multiplication to get new coefficients $\omega_n c_n$ and approximate $H(t)$,

$$H_N^{\omega}(\hat{\omega}) = \sum_{-N}^{N} \omega_n c_n e^{in\hat{\omega}},$$

For the Bartlett window (of order N) we let

$$\omega_n = \begin{cases} (1 - |\frac{n}{N}|) & |n| \leq N \\ 0 & |n| > N \end{cases}$$

For the Blackman window (of order N)

$$\omega_n = \begin{cases} 0.42 + 0.5cos(n\pi/N) + 0.08cos(2n\pi/N) & |n| \leq N \\ 0 & |n| > N \end{cases}$$

For the Hanning window (of order N)

$$\omega_n = \begin{cases} 0.5 + 0.5cos(n\pi/N) & |n| \leq N \\ 0 & |n| > N \end{cases}$$

Take $N = 32$ and make the following plots:

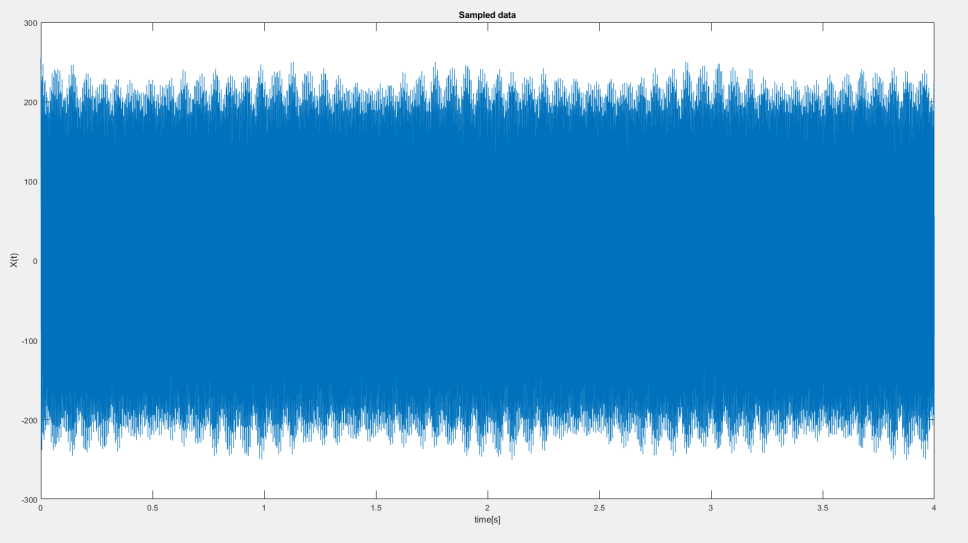
- the original function, $H(\hat{\omega})$ together with each of the three windowed versions H_N^{ω} seperately in three plots. Take $0 \leq \hat{\omega} \leq \pi$.
- a plot of the logarithm of the windowed functions as in (2d) on a single set of axes.

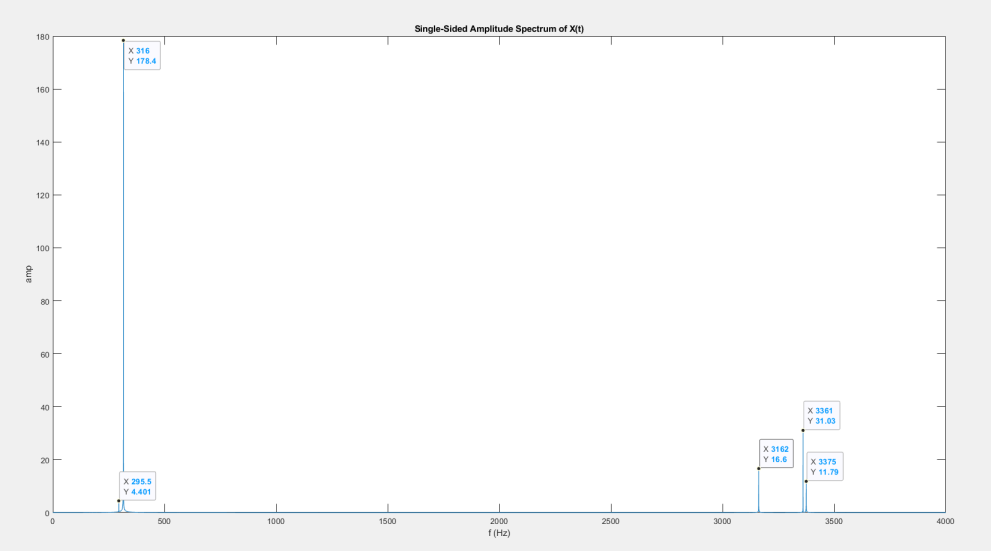
(f) Which of the three would make the best band-pass filter? Why

(g) Using the method above find coefficients $h[0], h[1], \ldots h[16]$ for a FIR band-pass filter with pass band 600—1662Hz for signals sampled at a sampling rate of 8820Hz. Thus you must choose a, b & N in the above along with the window you choose in (2f) to fulfill these criteria. Record a, b & N and your 17 coefficients in h.

Solutions

1. The name of the file received was `sample009.mat`. There was some difficulty with this question, so the reconstruction of the signal may not be completely accurate. Plots of the frequency domain and the windowed sampled data are shown.





From the frequency domain plot, we can visibly see 5 amplitudes. MATLAB was able to order the amplitudes from largest to smallest along with the corresponding position each component is in. The 1^{st} component is the largest amplitude, which is 178.3566. If we were to take component number 10 using MATLAB, the amplitude has size 8.5893, which is not negligible. If any components are discarded, the signal will not be accurately shown. There is no tolerance allowed. Since the sampling frequency is 8000 Hz, by the Nyquist theorem, signals up to 4000 Hz can be reconstructed. In order to better reconstruct the signal, we must avoid truncating and sum all components to

reproduce the signal as accurately as possible.

Using MATLAB, we take N to be the number of terms in our amplitude; $N = 16001$. The frequency resolution is determined by $\frac{F_s}{n} = \frac{8000}{32000} = 0.25\ Hz$. The amplitude A_n of each term is in the n^{th} position of the single sided amplitude spectrum of $x(t)$. The value f_n is found by multiplying the resolution frequency by n; $n * 0.25\ Hz = 8000\ Hz$. If we were to find, for example, the largest amplitude, we can correspond its position. Typing `B(1)` and `I(1)` in the command line yields the largest amplitude and the position of that component, respectively. `B(1)=178.3566`, `I(1)=1265`. Then the frequency of that component is $1265 * 0.25\ Hz = 316.25\ Hz$. Thus, the 1^{st} component (our largest amplitude of 178.3566) has frequency 316.25 Hz, with $n = 1265$.

```
N_number_of_terms_in_sum =

       16001


delta_f =

    0.2500


fn =

        8000


Highest_Amp =

  178.3566


freq_of_highest_amp =

  316.2500
```

Output of the MATLAB code in reproducing the signal is shown below, with added comments.

Listing 1: This is the MATLAB code corresponding to all of question 1

```
clear; close all; clc;
% Load data from file
load sample009.mat
% Compute the number of data
n = length(sample);
% Set the sample frequency
Fs = 8000;
```

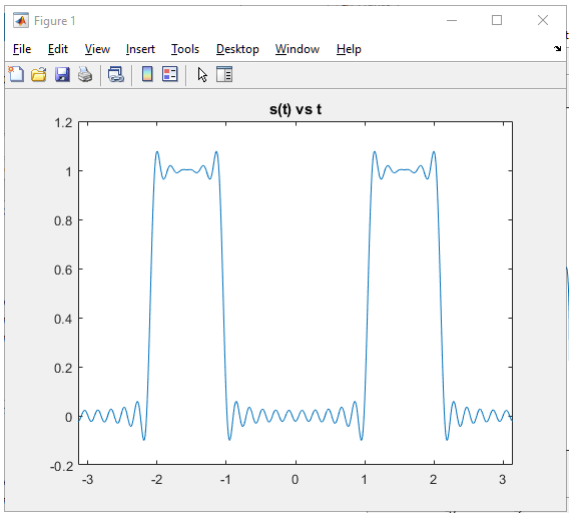
```
% Compute the sample period
T = 1/Fs;
% Define an array with sample times
time = 1/Fs:1/Fs:n*T;
% Plotting data points
figure(1)
plot(time,sample)
title('Sampled data')
xlabel('time[s]')
ylabel('X(t)')
% Compute the fourier transform of the sampled signal
Y = fft(sample);
% Compute the two sided spectrum
P2 = abs(Y/n);
P1 = P2(1:n/2+1);
P1(2:end-1) = 2*P1(2:end-1); %convert two sided spectrum to single sided spectrum
% Define the frequency domain
f = Fs*(0:(n/2))/n;
figure(2)
plot(f,P1)
title('Single-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)')
ylabel('amp')
% Order the spectrum by descending amplitude
% B stores the frequency components in order of Amplitude (first component
% is the most important and so on).
[B,I] = sort(P1,'descend'); %Arrangement I indicates which position each component is in, once ordered
%[B,I] = sort(___)  returns a collection of index vectors
%for any of the previous syntaxes. I is the same size as A and
%describes the arrangement of the elements of A into B
%along the sorted dimension. For example, if A is a vector, then B = A(I).
B = sort(P1, 'descend');

% N, the number of term in the sum, is the number of entries in the array
% P1. Then, N = length(P1) = 16000.
N_number_of_terms_in_sum=length(P1)
% Each point/bin in the FFT output array is spaced by the frequency
% resolution: delta_f = Fs/n = 8000/32000 = 0.25 Hz
 delta_f = Fs/n
% Then fn = n*delta_f
fn=n*delta_f;
fn
Highest_Amp=B(1)
freq_of_highest_amp=I(1)*delta_f

% Amplitude An of each term is in the n-th position of P1.
```

2. (a) $H(\hat{\omega})$ only allows frequencies in the range $a \leq |\omega| \leq b$, and is zero otherwise. Thus, it cancels out all the other frequencies outside this range, making it an ideal band-pass filter.

   (b) Note that

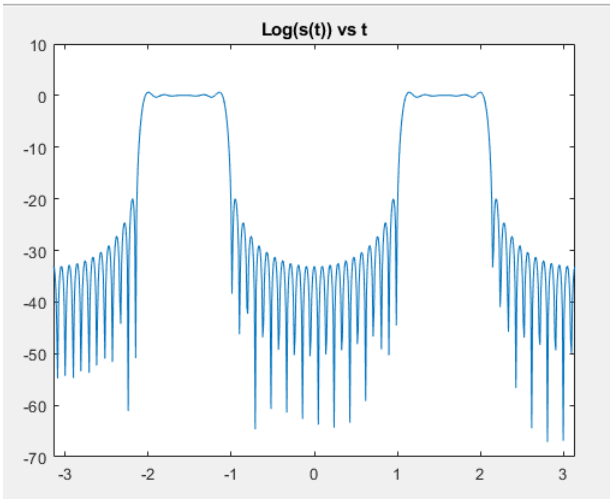
$$\begin{aligned} c_n &= \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\hat{\omega})e^{in\omega}d\omega \\ &= \frac{1}{2\pi}[\int_{-b}^{-a} e^{in\omega}d\omega + \int_{a}^{b} e^{in\omega}d\omega] \\ &= \frac{1}{2\pi in}[e^{-ina} - e^{-inb} + e^{inb} - e^{ina}] \\ &= \frac{1}{2\pi in}[2i\sin(nb) - 2i\sin(na)] \\ &= \frac{\sin(nb) - \sin(na)}{n\pi} \end{aligned}$$

$$\begin{aligned} c_0 &= \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\hat{\omega})d\omega \\ &= \frac{1}{2\pi}[\int_{-b}^{-a} d\omega + \int_{a}^{b} d\omega] \\ &= \frac{1}{2\pi}[-a + b + b - a] \\ &= \frac{2b - 2a}{2\pi} \end{aligned}$$

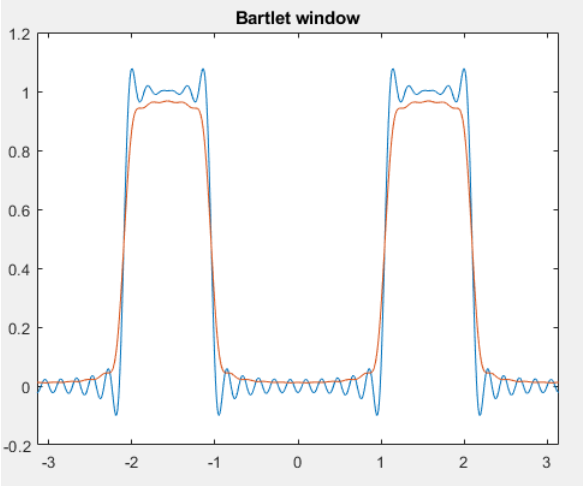
   (c) 

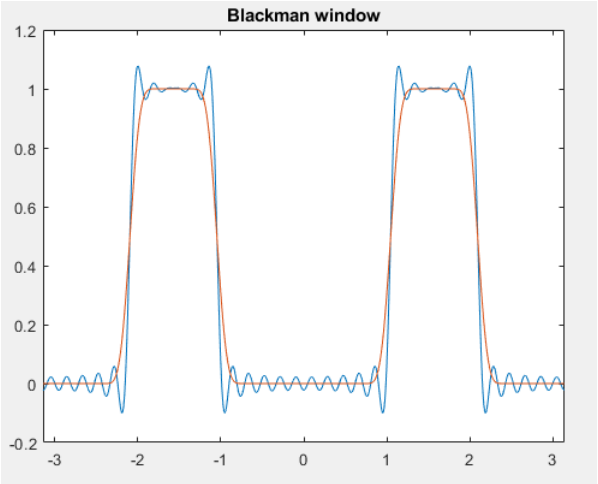
   Plotting the Real Part of the Frequency Response Function

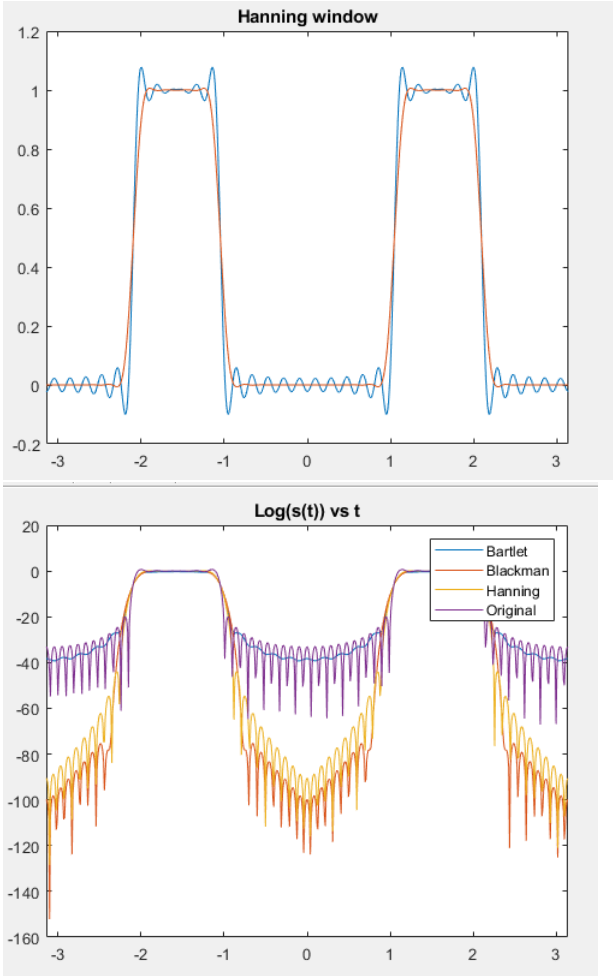
(d)


Log(s(t)) vs t
10
0
-10
-20
-30
-40
-50
-60
-70
-3
-2
-1
0
1
2
3

(e)


Bartlet window
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-3
-2
-1
0
1
2
3


Blackman window
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-3
-2
-1
0
1
2
3



(f) By observation of the last graph, Blackman reproduces the best band pass filter from a to b; it will reject frequencies outside of range $a \leq |\hat{\omega}| \leq b$.

(g) Note that using the Nyquist frequency, we have $|f| \leq \frac{f_s}{2}$ to avoid aliasing. Then the integrand of part (b) over ω goes from $-\pi f_s$ to πf_s. Thus, we shift ω by $\frac{2\pi f}{f_s}$. Then, if we want to pass frequencies 600-1662 Hz, we pick $a^* = \frac{2\pi 600}{8820}$, $b^* = \frac{2\pi 1662}{8820}$. We pick $N = 17$, since MATLAB index starts at 1. From part (f), we picked Blackman as the best filter, and thus, we window coefficients

$$w_0 c_0 = (0.42 + 0.5 + 0.08) * (\frac{2b^* - 2a^*}{2\pi})$$
$$w_n c_n = (0.42 + 0.5cos(\frac{\pi j}{N}) + 0.08cos(\frac{2\pi j}{N})) * (\frac{sin(b^* j) - sin(a^* j)}{j\pi})$$

where c_n, c_0 was calculated from (b)

Thus, using MATLAB, we have coefficients $h[0], h[1], \ldots h[16]$, where the very top is $h[0]$:

```
h[j] =
    0.2408
    0.1606
   -0.0084
   -0.1269
   -0.1262
   -0.0534
    0.0059
    0.0169
    0.0035
   -0.0031
    0.0017
    0.0062
    0.0049
    0.0014
   -0.0003
   -0.0003
   -0.0000
   -0.0000
```

Listing 2: This is the MATLAB code corresponding to all of question 2, with sections

```
clear all, close all;
clc
%% part 2 (c) plot the real coefficients (call this the original function)
a = pi/3;
b = 2*pi/3;
n = [-32:-1 1:32];
w = -pi:0.01:pi;
c0 = (-2*a+2*b)/(2*pi);
cn = (-sin(a*n)+sin(b*n))./(n*pi);
s = c0 + real(sum(diag(cn)*exp(i*transpose(n)*w))); %diag(cn) so that 1st row multiplied by c1, 2nd
      row multiplied by c2, etc...
%and transpose(n) since matrix multiplication operations need to be obeyed
figure(1);
plot(w, s)
xlim([-pi, pi])
title('s(t) vs t')
%% part (d) plot log of original
log_original = 20*log10(abs(c0 + sum(diag(cn)*exp(i*transpose(n)*w))));
figure(2);
plot(w, log_original)
xlim([-pi, pi])
title('Log(s(t)) vs t')
%% part (e) barlet with original
NN = 32;
w10 = 1;
w1n = 1 - abs(n/NN);
sw1 = c0*w10 + real(sum(diag(cn)*diag(w1n)*exp(i*transpose(n)*w)));
figure(3);
plot(w, s), hold on
plot(w, sw1)
xlim([-pi, pi])
title('Bartlet window')
%% part(e) blackman with original
w20 = 0.42 + 0.5 + 0.08;
w2n = 0.42 + 0.5*cos(pi*n/NN) + 0.08*cos(2*pi*n/NN);
sw2 = c0*w20 + real(sum(diag(cn)*diag(w2n)*exp(i*transpose(n)*w)));
figure(4);
plot(w, s), hold on
plot(w, sw2)
xlim([-pi, pi])
title('Blackman window')
%% part (e) Hanning with original
w30 = 0.5 + 0.5;
w3n = 0.5 + 0.5*cos(pi*n/NN);
sw3 = c0*w30 + real(sum(diag(cn)*diag(w3n)*exp(i*transpose(n)*w)));
figure(5);
plot(w, s), hold on
plot(w, sw3)
xlim([-pi, pi])
title('Hanning window')
```

```
%% part (e) plot log of 4 functions
log_Barlet = 20*log10(c0*w10 + sum(diag(cn)*diag(w1n)*exp(i*transpose(n)*w)));
log_Blackman = 20*log10(c0*w20 + sum(diag(cn)*diag(w2n)*exp(i*transpose(n)*w)));
log_Hanning = 20*log10(c0*w30 + sum(diag(cn)*diag(w3n)*exp(i*transpose(n)*w)));
figure(6);
plot(w, log_Barlet), hold on
plot(w, log_Blackman)
plot(w, log_Hanning)
plot(w, log_original)
xlim([-pi, pi])
title('Log(s(t)) vs t')
legend({'Bartlet', 'Blackman', 'Hanning', 'Original'})
%% part (g)
fs = 8820;
a = 2*pi*600/fs;
b = 2*pi*1662/fs;
NN = 17; %since matlab index starts at 1, we want coefficients h[0] to h[16]
h = zeros(NN+1,1);
h(1) = w20*(-2*a+2*b)/(2*pi);
for j=1:NN
  w2 = 0.42 + 0.5*cos(pi*j/NN) + 0.08*cos(2*pi*j/NN);
  h(j+1) = w2*(-sin(a*j)+sin(b*j))/(j*pi);
end

disp('h[j] =')
disp(h)
```