ECE 158 Digital Signal Processing Lab

Windowing Digital Signals

Student : Corey Gravelle
Perm Number: 8094211
E-mail: cgravelle@umail.ucsb.edu

11/17/2009

Department of Electrical and Computer Engineering, UCSB. ECE158

Introduction:
This lab is an overview of windowing techniques to expose methods of dealing with spectral leakage. I will use techniques in Mathworks' Matlab to review triangular, sinusoidal, rectangular, and hann-shaped windows and compare the results.

1.1. Discrete Fourier Transform (DFT) of zero-padded periodic signal:

Code:

```
function[] = one_one()


f = 220;          %signal freq
Fs = 2048;        %sampling freq
N = 32;           %number samples
n = [0:N-1];

zees = zeros(1, 1024-N);
x = cos(2*pi*f*n/(Fs));
zpad = [x, zees];

xtrans = fft(x);
xabs = abs(xtrans);

ztrans = fft(zpad);
zabs = abs(ztrans);

freq = linspace(0,Fs/2000,
length(zabs)/2);
freq16 = linspace(0,Fs/2000, 17);

plot(freq, zabs(1:floor(length(zabs)/2)));
hold on;
stem(freq16,
xabs(1:floor(length(xabs)/2)+1));
title('FFT of zero-padded x(t)')
xlabel('Frequency(kHz)')
ylabel('Magnitude')
```

Graph:

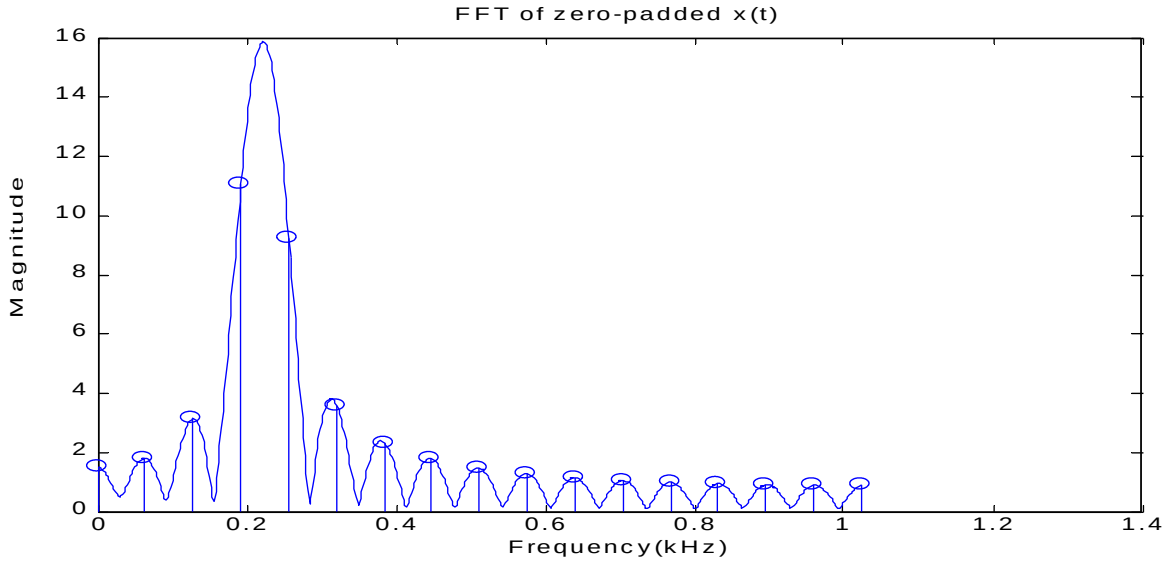


Figure 1: Graph of zero-padded x(t) with frequency 220Hz, taken with 32 samples.

1.2

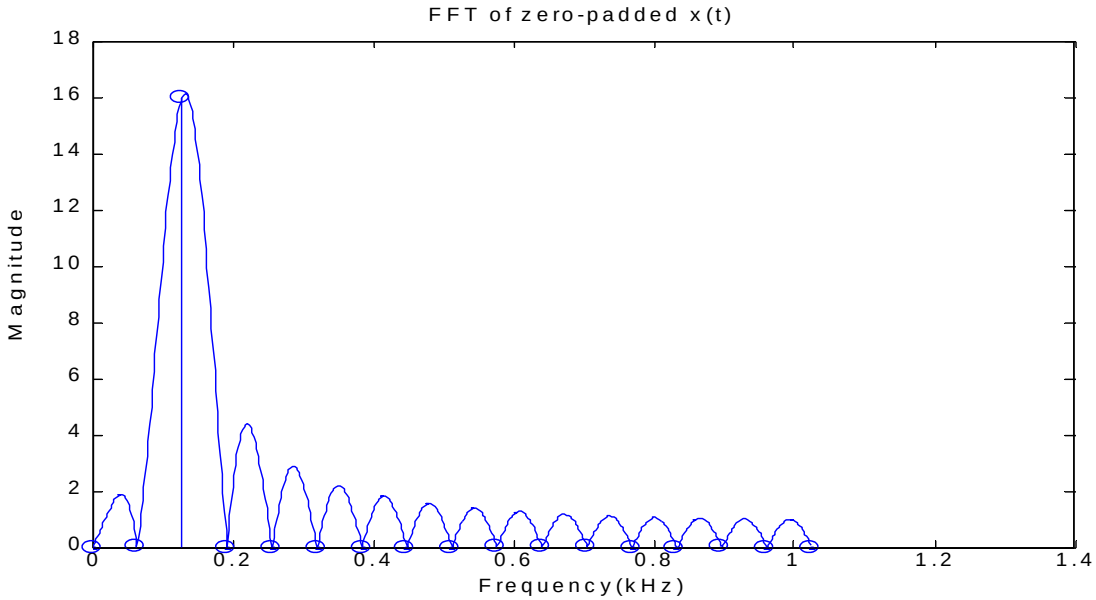


Figure 2: Graph of zero-padded x(t) with frequency 128 Hz

While the graph of the 220 Hz periodic signal provided a stem plot of the peaks of the signal, the graph of the 128Hz signal had only one branch at its frequency, 128Hz. This is because there were an exact number of points taken which resulted in one branch landing on 128 Hz, and this absorbed all possible leaked frequencies' energies in the Discrete Fourier Transform. In the graph of the 220Hz signal, there was no 220Hz branch; if it were the case, it would look like the graph of the 128 Hz signal.

However, the continuous, zero-padded signal continues to display values in between the zeros graphed by the dft. This is from the process of capturing a seemingly infinite signal, cos(Xn), in a finite window (in this case, one with a width of 1024 samples).

1.3

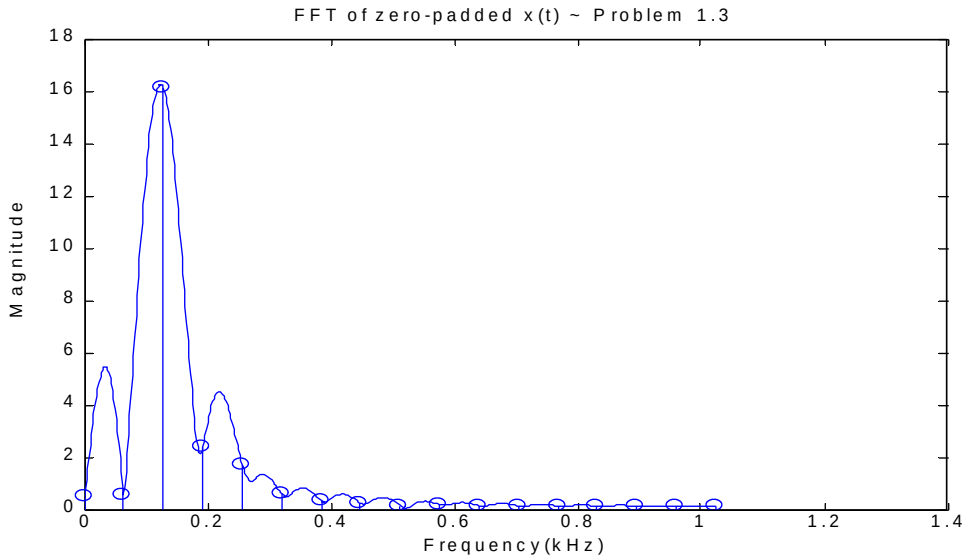


Figure 3A: Graph of new zero-padded x(t) with varying frequencies and with 1024 samples

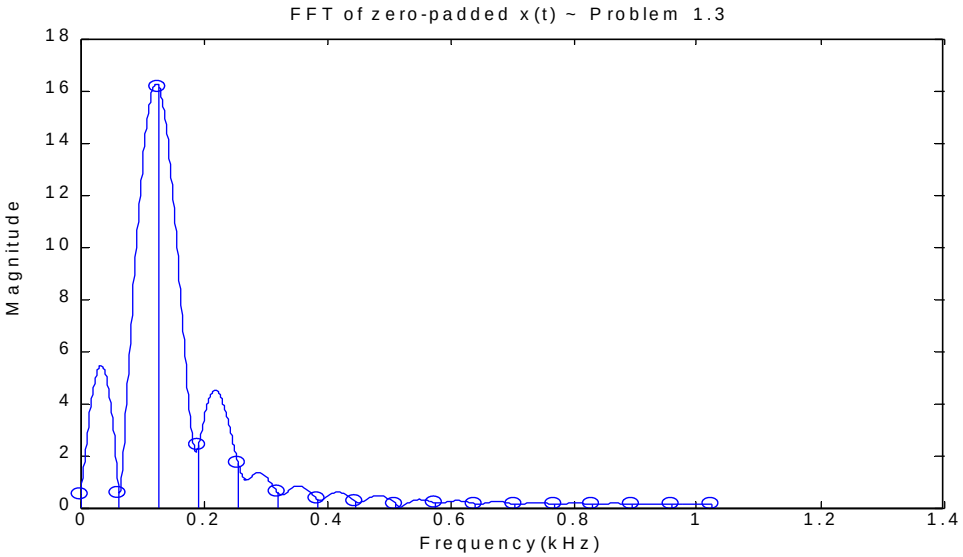
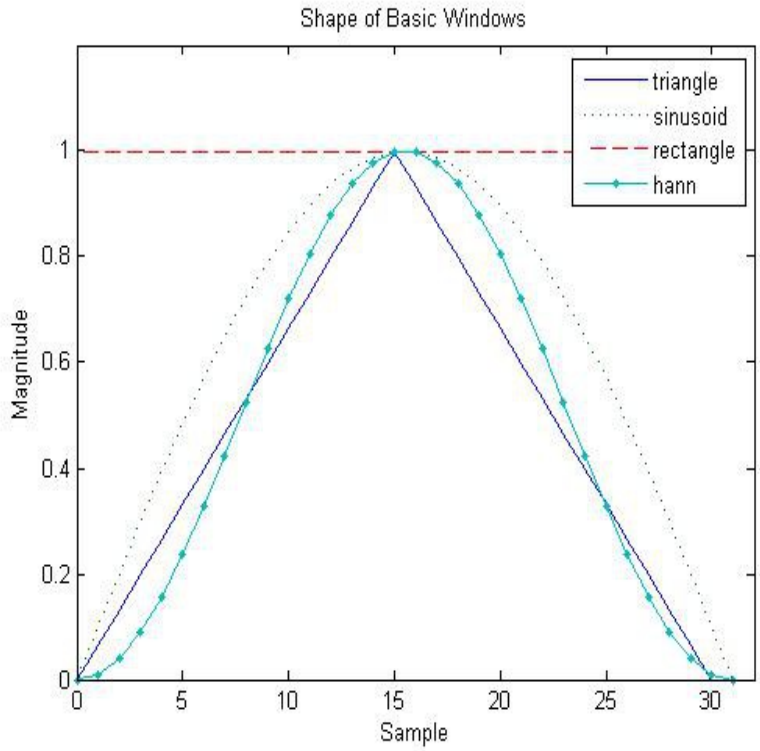


Figure 3B: Graph of new zero-padded x(t) with varying frequencies and with 2048 samples- exactly the same graph as with 1024 samples.

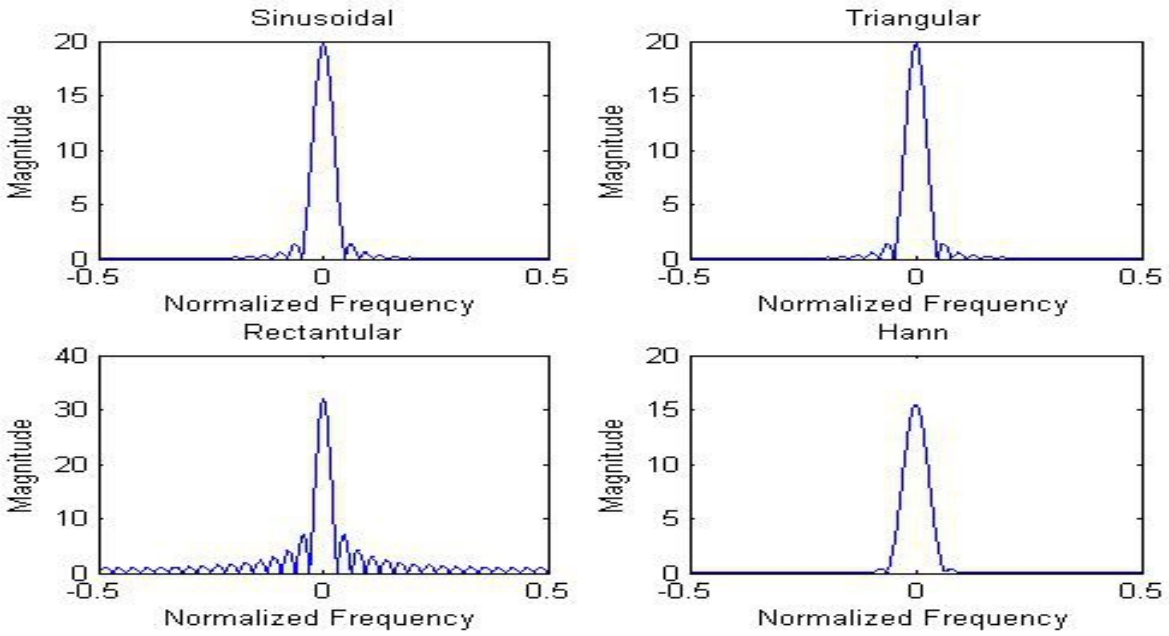
1.4 *Without the knowledge that there are multiple pure tones in this sequence, can you pick out the right frequency components in the magnitude spectrum?*

Without the knowledge of the original signal, it would be very difficult to narrow down the frequencies as 128, 220, and 525 Hz. It is obvious that the most prevalent part of the signal has approximate frequency of 130, but picking out the secondary frequencies would be difficult.

2.1

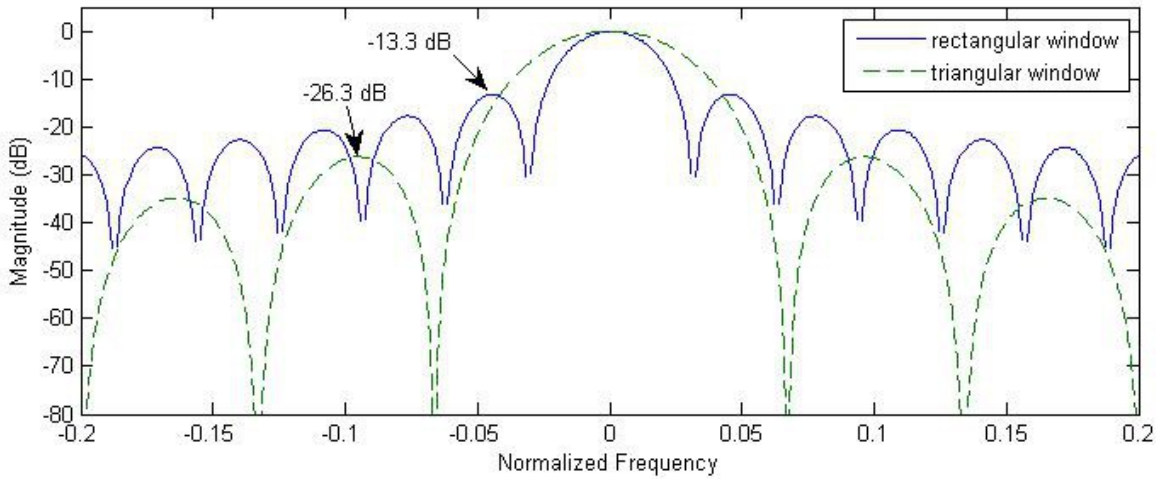


2.2 Plots of Discrete Fourier transforms of normalized frequency responses, and code for part 2:




```
function[] = windows()

Ne = 32;
No = 31;
ne = [0:Ne-1];
no = [0:No-1];
graph1 = 0;
graph2 = 1;

zeeO = zeros(1, 1024-No);
zeeE = zeros(1, 1024-Ne);
rect = ones(1, Ne);
tri  = ((No-1)/2-abs(no-(No-1)./2)).*2/
(No-1);
sine = sin(pi*ne/(Ne-1));
hann = .5*(1-cos(2*pi*ne/(Ne-1)));

rectz = [rect, zeeE];
triz =  [tri,  zeeO];
sinez = [sine, zeeE];
hannz = [hann, zeeE];

rtrans = fftshift(fft(rectz,1024));
rabs = abs(rtrans);
ttrans = fftshift(fft(triz,1024));
tabs = abs(ttrans);
strans = fftshift(fft(sinez,1024));
sabs = abs(strans);
htrans = fftshift(fft(hannz,1024));
habs = abs(htrans);

freqE = linspace(-.5, .5, length(rectz));
freqO = linspace(-.5, .5, length(triz));

if graph1 == 1
    plot(no, tri, ne, sine, ':', ne, rect,
'--', ne, hann, '.-');

legend('triangle','sinusoid','rectangle','
hann');
    axis([0 32 0 1.2]);
    title('Shape of Basic Windows')
    xlabel('Sample')
    ylabel('Magnitude')
end

if graph2 == 1
    subplot(2,2,1);
    plot(freqO, sabs(1:length(sabs)));
    title('Sinusoidal');
    xlabel('Normalized Frequency');
    ylabel('Magnitude')
    subplot(2,2,2);
    plot(freqE, sabs(1:length(sabs)));
    title('Triangular');
    xlabel('Normalized Frequency');
    ylabel('Magnitude')
    subplot(2,2,3);
    plot(freqE, rabs(1:length(rabs)));
    title('Rectantular');
    xlabel('Normalized Frequency');
    ylabel('Magnitude')
    subplot(2,2,4);
    plot(freqE, habs(1:length(habs)));
    title('Hann');
    xlabel('Normalized Frequency');
    ylabel('Magnitude')
end
```

2.3

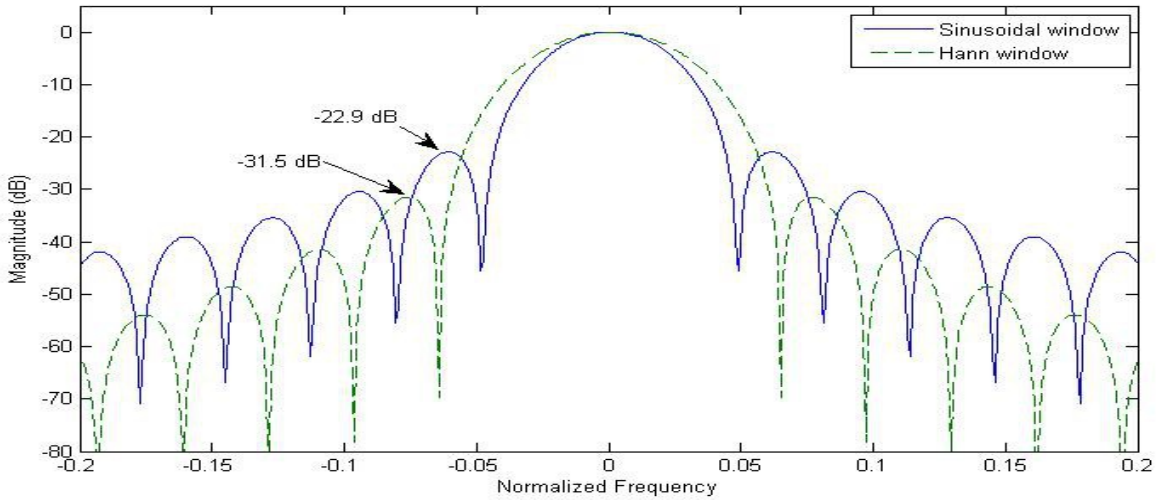


2.4 What is the width of the main-lobe for the triangular window in terms of that for the rectangular window? Do you have any intuition why this is the case?

The width of the rectangular window's main lobe is about 0.06, while the triangular window's central lobe is about twice that. This is because the center of the triangularly-windowed signal has higher priority over the edges, which represent higher frequencies when zero-padded and taken the transform of.

2.5 The height of the rectangular first side-lobe is about -13.3 dB, while the triangular side-lobe is -26.3 dB.

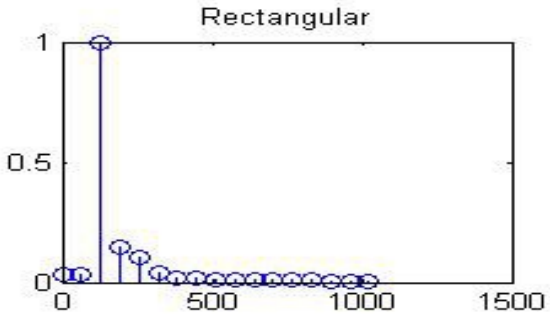
2.6

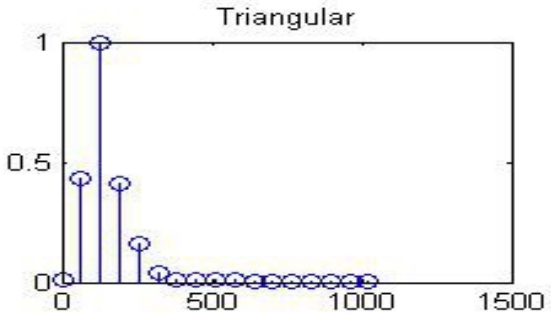


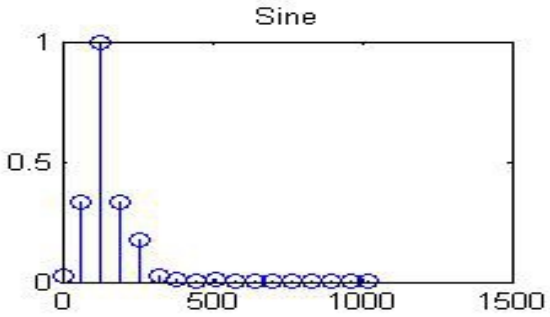
2.7 The normalized width of the sinusoidal window's main lobe is about 0.09, and the Hann window's middle lobe has a width of 0.13.

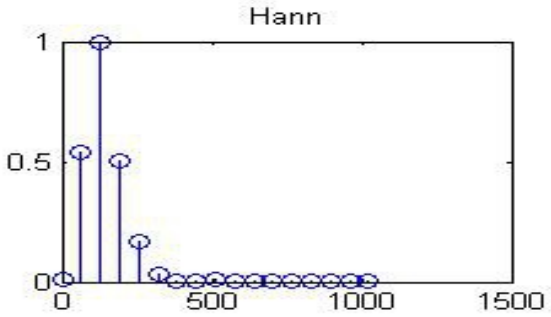
2.8 The Sinusoidal second lobe has height -22.9dB, while the Hann second lobe has height -31.5dB.

3.1 Frequency Response of windowed signal from 1.3, plots and code:



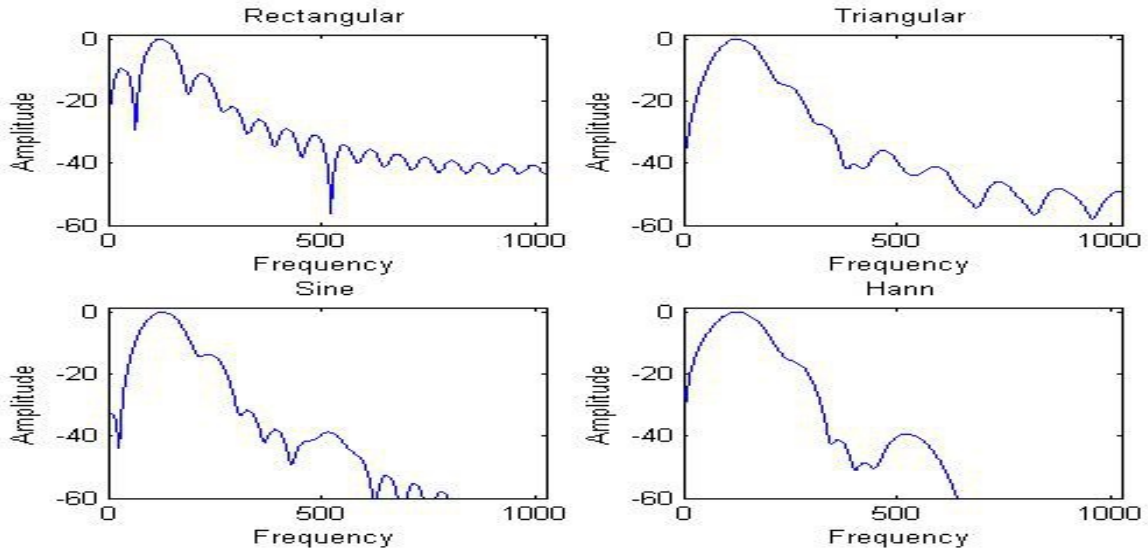







```
function[] = three_one()


Fs = 2048;      %sampling freq
N = 32;         %number samples
n = [0:N-1];
f = n*Fs/N;        %signal freq

%sequence:
zees = zeros(1, 2048-N);
xa = sin(2*pi*128*n/2048);
xb = 0.2.*sin(2*pi*220*n/2048);
xc = 0.01.*cos(2*pi*525*n/2048);
x = xa + xb + xc;
zpad = [x, zees];

xtrans = fft(x);
xabs = abs(xtrans);
ztrans = fft(zpad);
zabs = abs(ztrans);

%filters:
Ne = N;
No = N-1;
ne = [0:Ne-1];
no = [0:No-1];

rect = ones(1, Ne);
tri  = [((No-1)/2-abs(no-(No-1)./2)).*2/
(No-1),0];
sine = sin(pi*ne/(Ne-1));
hann = .5*(1-cos(2*pi*ne/(Ne-1)));
freq = linspace(0,Fs/2000,
length(zabs)/2);
freq16 = linspace(0,Fs/2000, 17);

rx = x.*rect;
rxt = fft(rx);
rectx = abs(rxt)/max(abs(rxt));
tx = x.*tri;
txt = fft(tx);
trix = abs(txt)/max(abs(txt));
sx = x.*sine;
sxt = fft(sx);
sinx = abs(sxt)/max(abs(sxt));
hx = x.*hann;
hxt = fft(hx);
hanx = abs(hxt)/max(abs(hxt));

%plot:
figure
freq16 = linspace(0, Fs/2, 17);
subplot(2,2,1); stem(freq16,
rectx(1:floor(length(xabs)/2)+1));
title('Rectangular');
subplot(2,2,2); stem(freq16,
trix(1:floor(length(xabs)/2)+1));
title('Triangular');
subplot(2,2,3); stem(freq16,
sinx(1:floor(length(xabs)/2)+1));
title('Sine');
subplot(2,2,4); stem(freq16,
hanx(1:floor(length(xabs)/2)+1));
title('Hann');
```

3.2 DFT of windowed and zero-padded signal from 1.3, plot and code:

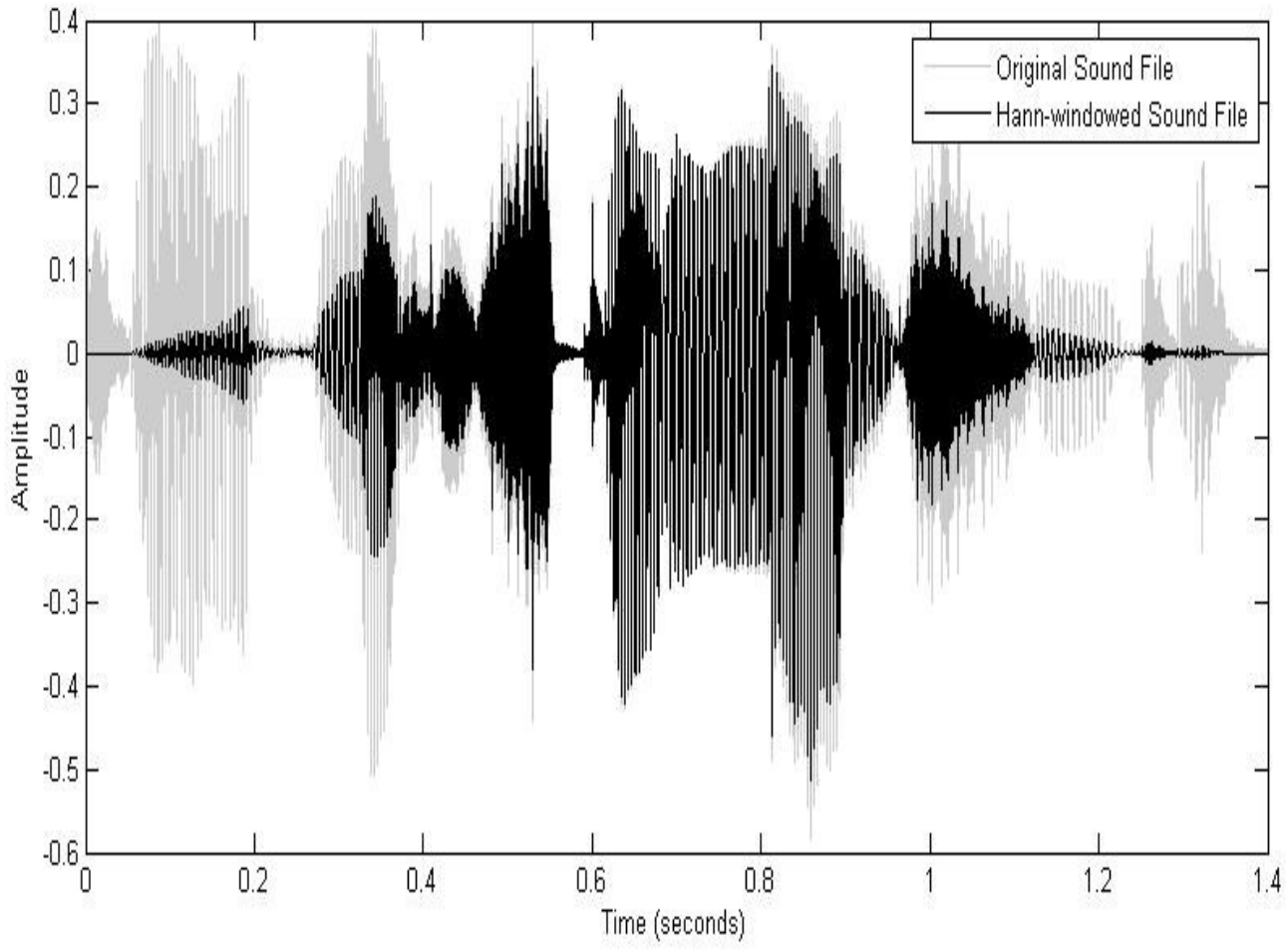



```
function[] = three_two()

Fs = 2048;      %sampling freq
N = 32;         %number samples
n = [0:N-1];
f = n*Fs/N;        %signal freq
graph = 0;

%sequence:
zees = zeros(1, 2048-N);
xa = sin(2*pi*128*n/2048);
xb = 0.2.*sin(2*pi*220*n/2048);
xc = 0.01.*cos(2*pi*525*n/2048);
x = xa + xb + xc;
zpad = [x, zees];

xtrans = fft(x);
xabs = abs(xtrans);
ztrans = fft(zpad);
zabs = abs(ztrans);

%filters:
Ne = N;
No = N-1;
ne = [0:Ne-1];
no = [0:No-1];

rect = ones(1, Ne);
tri  = [((No-1)/2-abs(no-(No-1)./2)).*2/
(No-1),0];
sine = sin(pi*ne/(Ne-1));
hann = .5*(1-cos(2*pi*ne/(Ne-1)));

rx = x.*rect;
rxt = fft(rx,2048);
recx = 20*log10(abs(rxt)/max(abs(rxt)));
tx = x.*tri;
txt = fft(tx,2048);
trix = 20*log10(abs(txt)/max(abs(txt)));
sx = x.*sine;
sxt = fft(sx,2048);
sinx = 20*log10(abs(sxt)/max(abs(sxt)));
hx = x.*hann;
hxt = fft(hx,2048);
hanx = 20*log10(abs(hxt)/max(abs(hxt)));

%plot:
figure
freq16 = linspace(0, Fs/2,
length(recx)/2);
subplot(2,2,1); plot(freq16,
recx(1:floor(length(recx)/2)));
axis([0 1024 -60 1]);
xlabel('Frequency');
ylabel('Amplitude');
title('Rectangular');
subplot(2,2,2); plot(freq16,
trix(1:floor(length(trix)/2)));
axis([0 1024 -60 1]);
xlabel('Frequency');
ylabel('Amplitude');
title('Triangular');
subplot(2,2,3); plot(freq16,
sinx(1:floor(length(sinx)/2)));
axis([0 1024 -60 1]);
xlabel('Frequency');
ylabel('Amplitude');
title('Sine');
subplot(2,2,4); plot(freq16,
hanx(1:floor(length(hanx)/2)));
axis([0 1024 -60 1]);
xlabel('Frequency');
ylabel('Amplitude');
title('Hann');
```

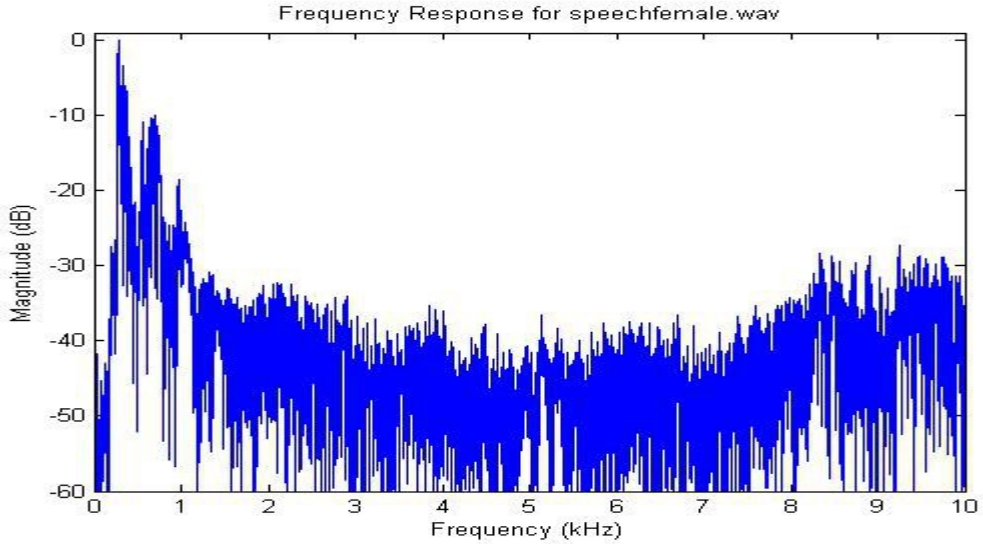
3.3 *Knowing how the main-lobes depend on the type of window, and how the side-lobes decay, for which of these windows can you say with certainty that this signal has three components? For which windows can you say with accuracy the frequency of the middle amplitude sinusoid? Why are the Sine and Hann windows good at revealing the lowest amplitude component, but not the middle amplitude component? Why should the lowest amplitude component have a normalized dB level of –40 dB?*

For the rectangularly windowed signal we know that the window didn't attenuate any signal, so surely there are still three components which make up the DFT. For the rectangular window, one may assume that the middle amplitude sinusoid is consistent with the original signal. The other windows have a wide middle lobe which may group the first and second frequency components. From the original signal, one may see that the lowest amplitude component has an amplitude of 0.01, which in normalized dB is -40 because the highest amplitude component has a magnitude of 1.

3.4

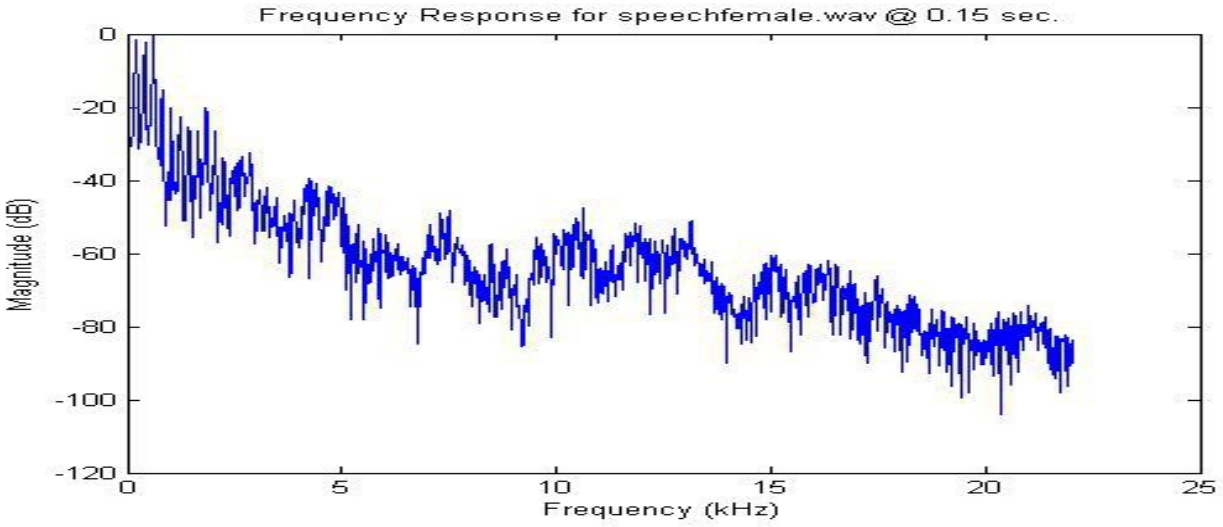


3.5 Frequency response of first 1.4 seconds of speech_female.wav:

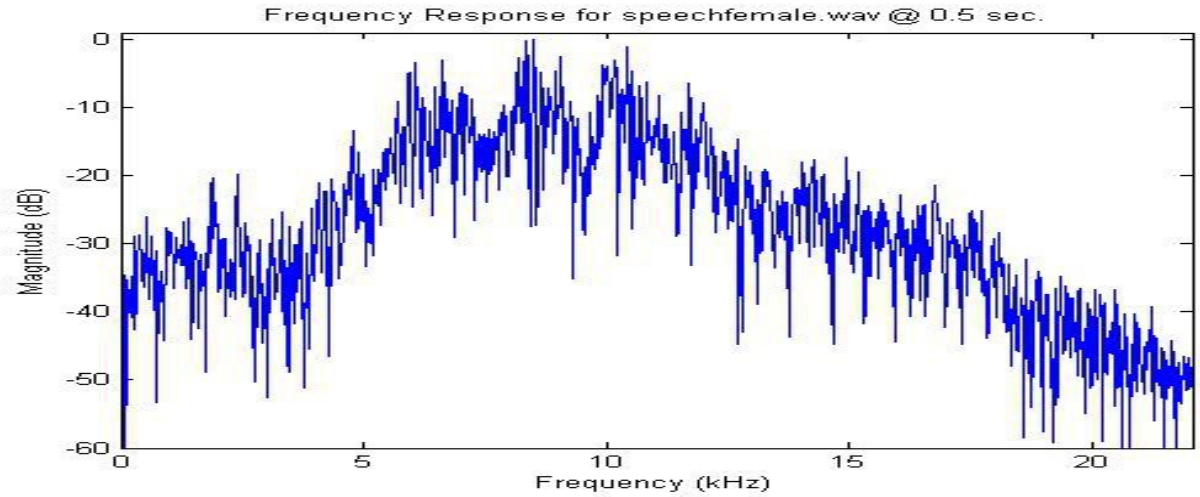



```
function[] = three_five();

[x,Fs] = wavread('speech_female.wav');
x2(1,:) = x(:,1);

%filter:
ne = [0:Fs*1.4];
hann = .5*(1-cos(2*pi*ne/(Fs*1.4)));
han2(:,1) = hann(1,:);
hanned = (x(1:(Fs*1.4)+1).*han2);

hanf = fft(hanned);
hlog = 20*log10(abs(hanf)/max(abs(hanf)));
freq = linspace(0, length(hlog)/2000, length(hlog)/2);
soundsc(hanned)

%plot:
figure
plot(freq, hlog(1:(length(hlog)/2)));
axis([0 10 -60 1]);
title('Frequency Response for speechfemale.wav');
xlabel('Frequency (kHz)');
ylabel('Magnitude (dB)');
```

3.6 Frequency response of a 0.05-second sample starting at 0.15 seconds, plot and code:




```
function[] = three_six()

[x,Fs] = wavread('speech_female.wav');

x2(1,:) = x(:,1);

%filter:
N = 2048;
n = [0:N-1];
hann = .5*(1-cos(2*pi*n/(N*1.4)));
han2(:,1) = hann(1,:);


hanned = (x(.15*Fs:.15*Fs+2047).*han2);

hanf = fft(hanned);
hlog = 20*log10(abs(hanf)/max(abs(hanf)));
freq = linspace(0, Fs/2000, N/2);

%plot:
figure
plot(freq, hlog(1:N/2));
%axis([0 Fs/2000 -60 1]);
title('Frequency Response for
speechfemale.wav @ 0.5 sec.');
xlabel('Frequency (kHz)');
ylabel('Magnitude (dB)');
```



3.7 With a sampling frequency of 44100 Hz, these 2048-sample signals are 2048/44100 seconds long, or about 0.046 seconds.