EE386 Lab 2 - Fourier, Beat, Chirp, Tones, and Bells

Nov. 4, 2018

Lab directed by Chris Hirunthanakorn

Lab conducted by Paco Ellaga

CSULB SID: 009602331

PART 1:

Question A

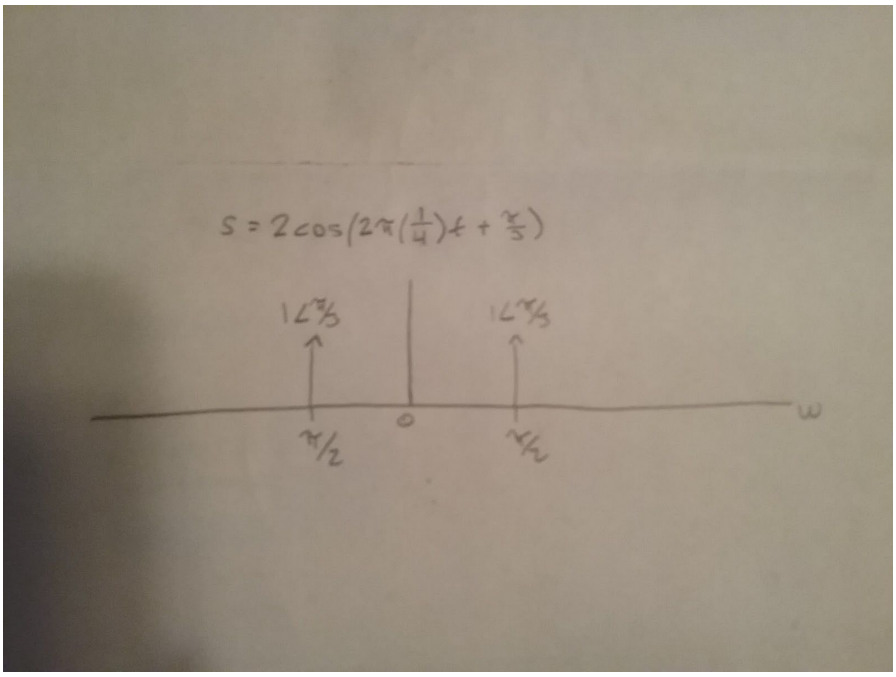

Figure 1: *Hand drawn version of equation 1*

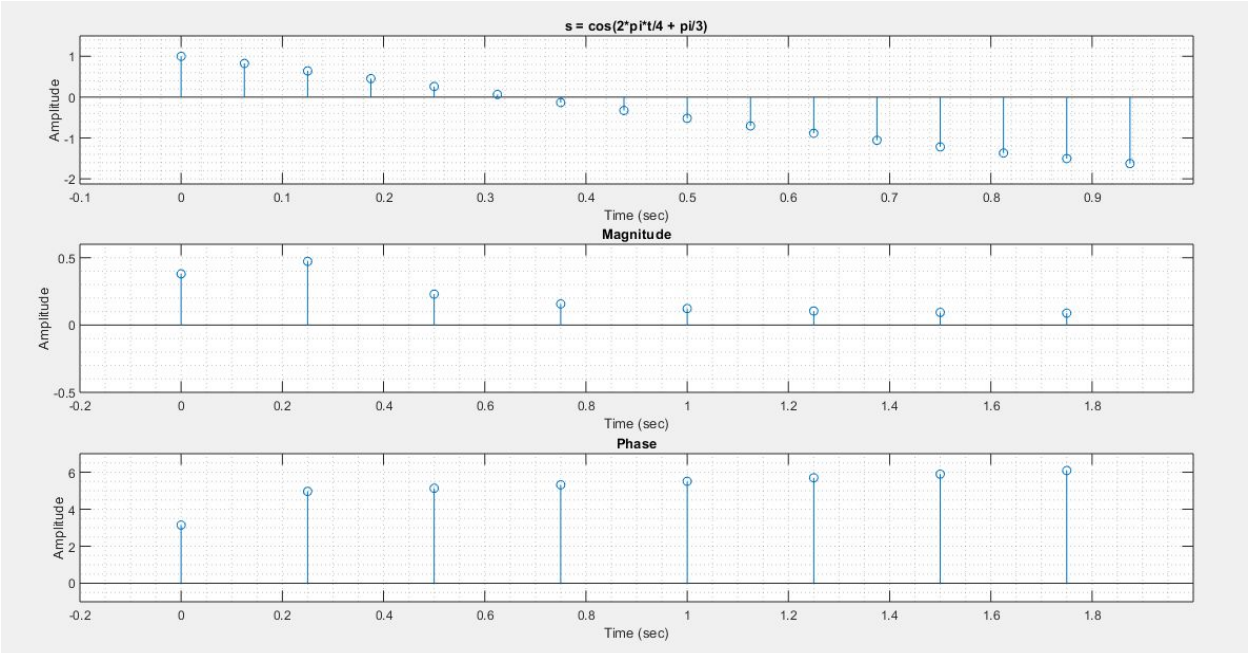

Figure 2: *MATLAB result of equation 1*

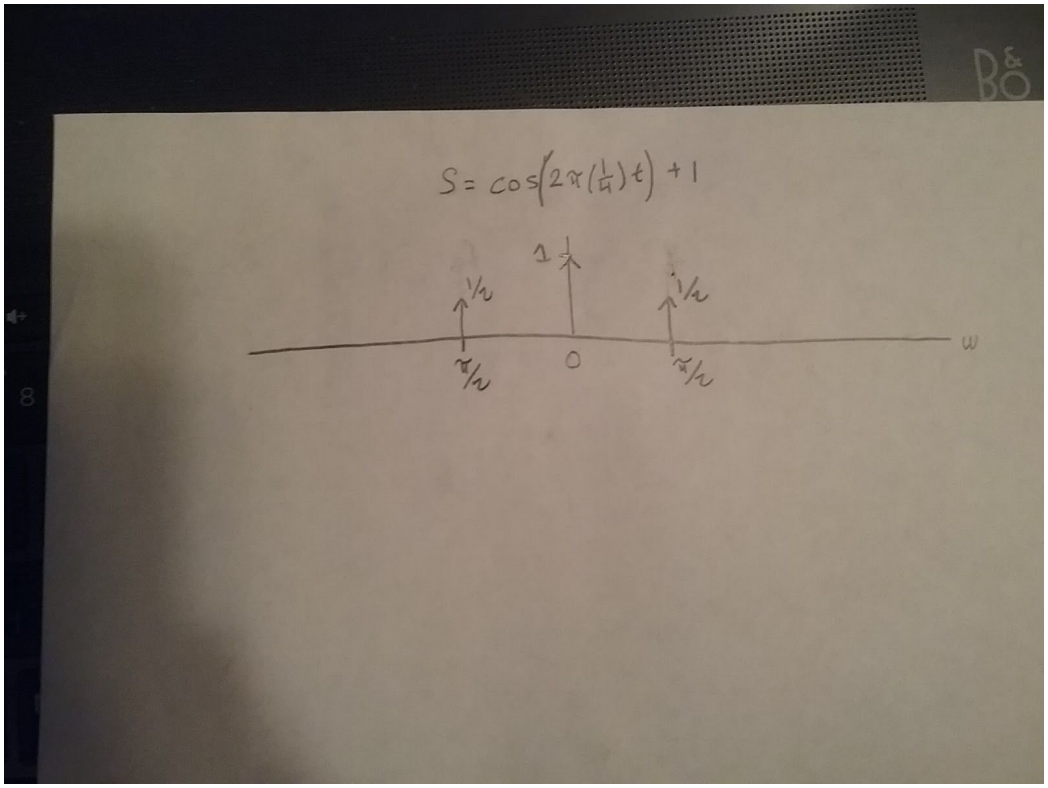

Figure 3: *Hand drawn version of equation 2*

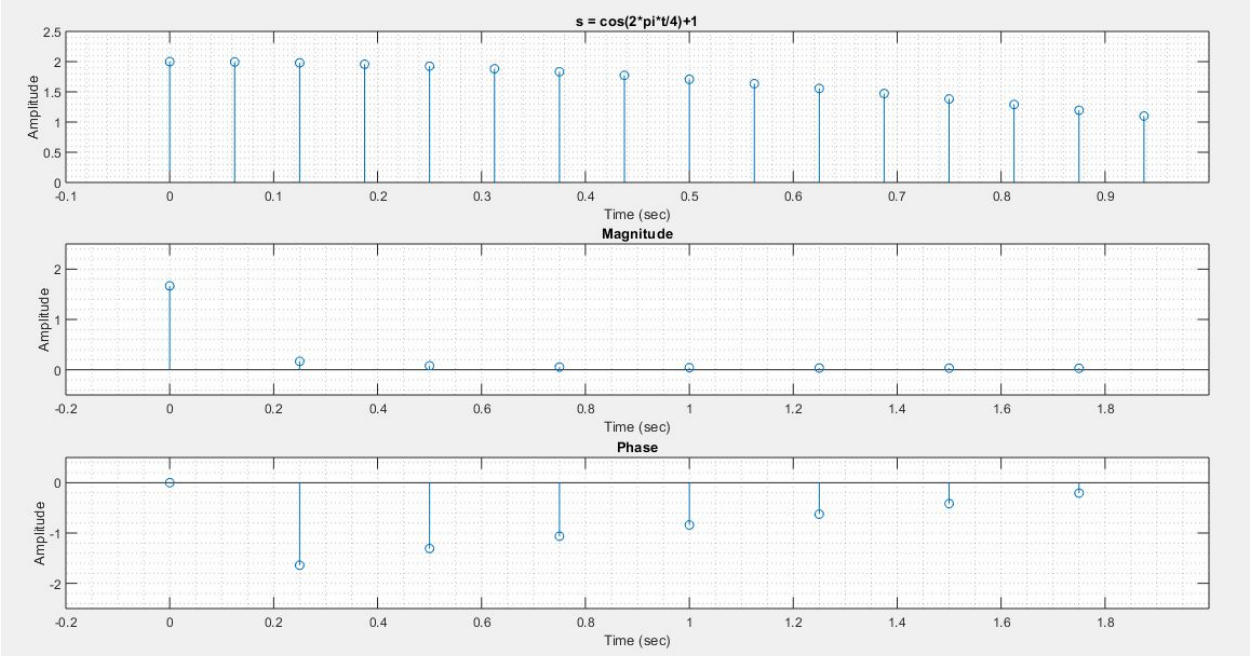

Figure 4: *MATLAB version of equation 2*

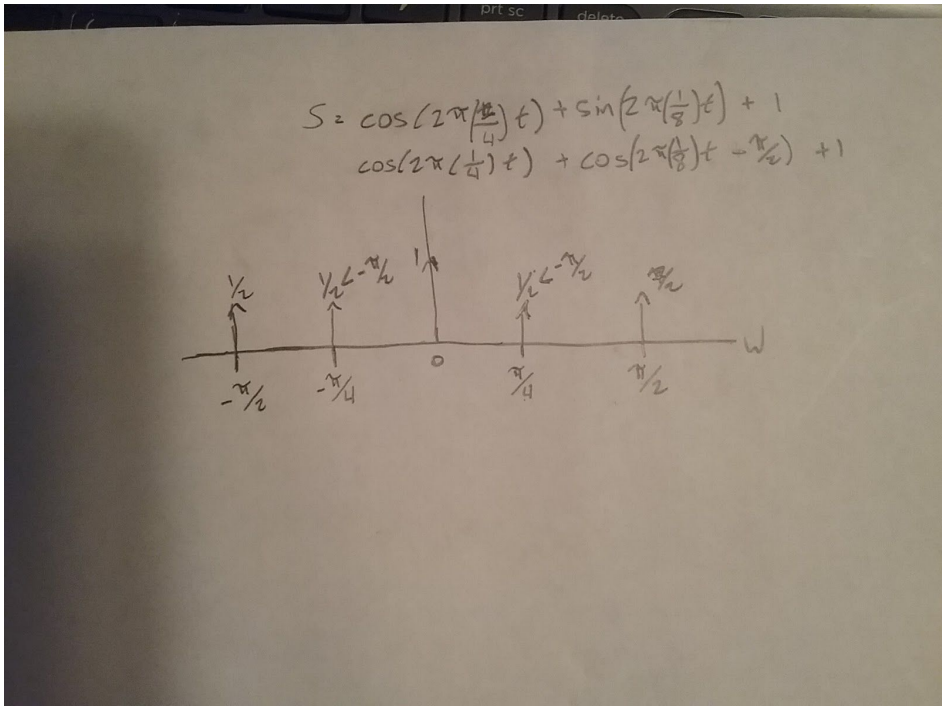

Figure 5: *Hand drawn version of equation 3*

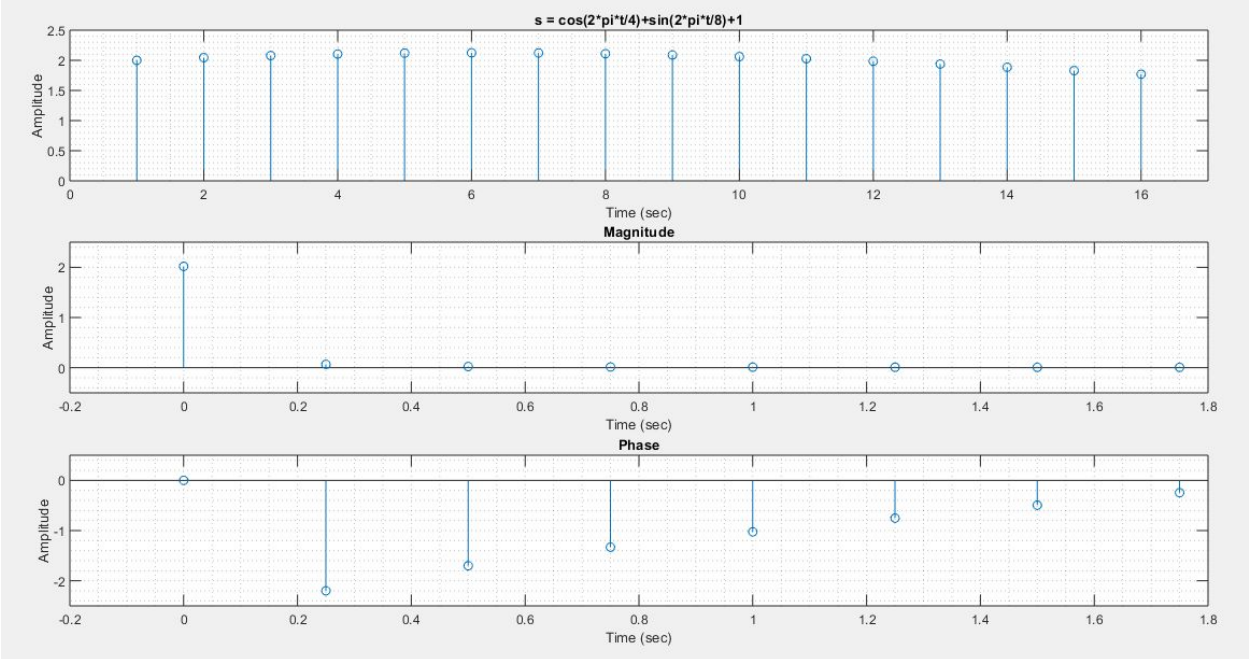

Figure 6: *MATLAB version of equation 4*

Question B-E

Figure 7 is the signal, ss = 2*cos(2*pi*t/2) with a time array of 2 seconds across 2^11 points. ***Figure 8*** is when proper fft shift is applied.

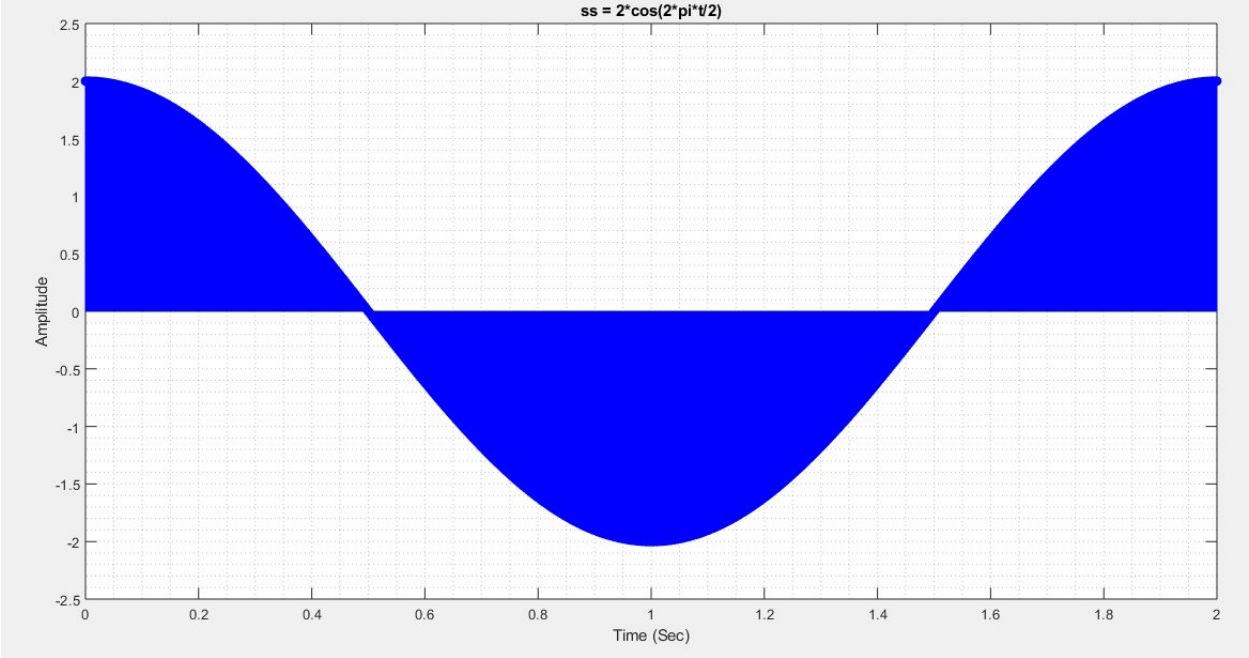

Figure 7: *MATLAB result of* ss = 2*cos(2*pi*t/2)

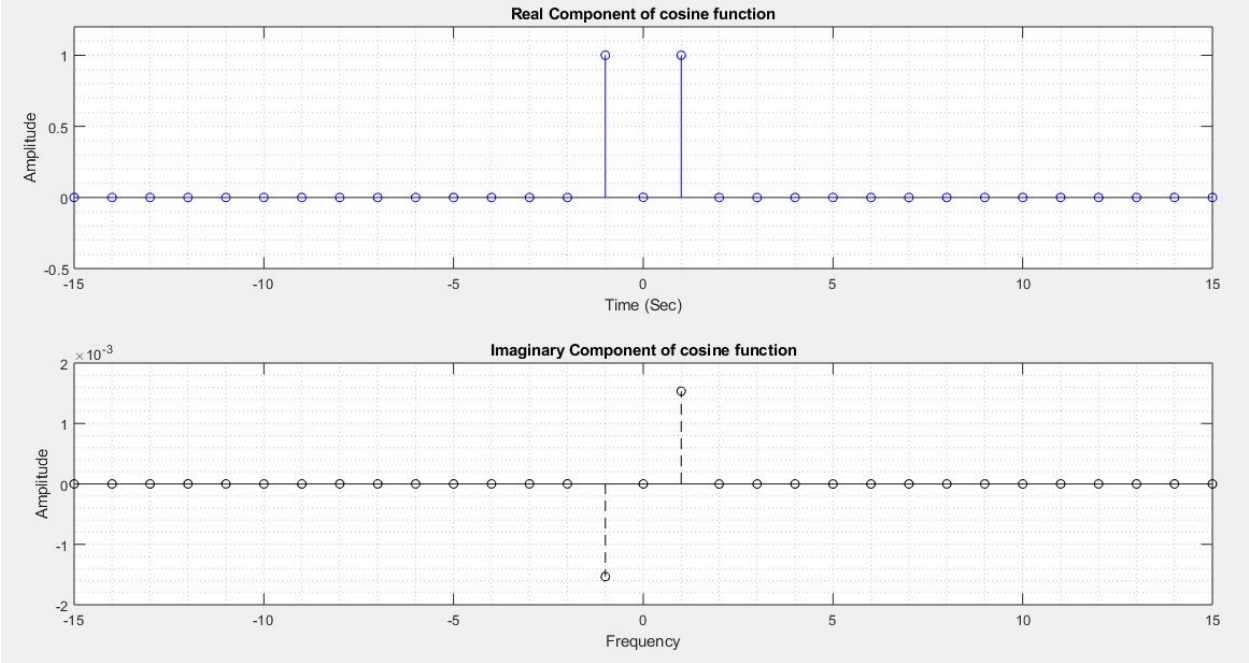

Figure 7: *MATLAB result of* ss = 2*cos(2*pi*t/2) when fft is performed

Question F (Even & Odd Square Waves)

For the odd scenario, ***Figure 8*** is the hand drawn prediction, ***Figure 9 and 10*** are the MATLAB results, and the frequencies where Cn = 0 are whole number integers increasing from -∞ to +∞. For the even scenario, ***Figure 11*** is the hand drawn prediction, ***Figure 12 and 13*** are the MATLAB results, and the frequencies where Cn = 0 are numbers increasing from -∞ to ∞ that are from interval from -∞...-1.5, -0.5, 0.5,... +∞.

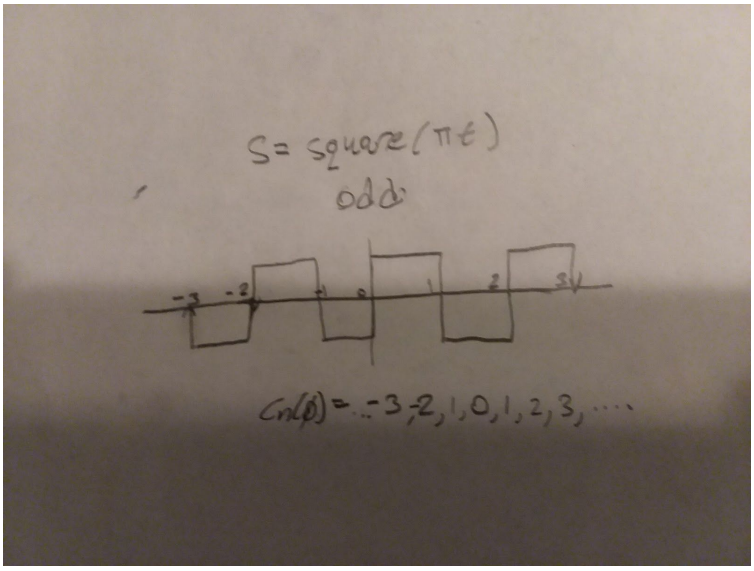

Figure 8: *Hand drawn version of the odd square function*

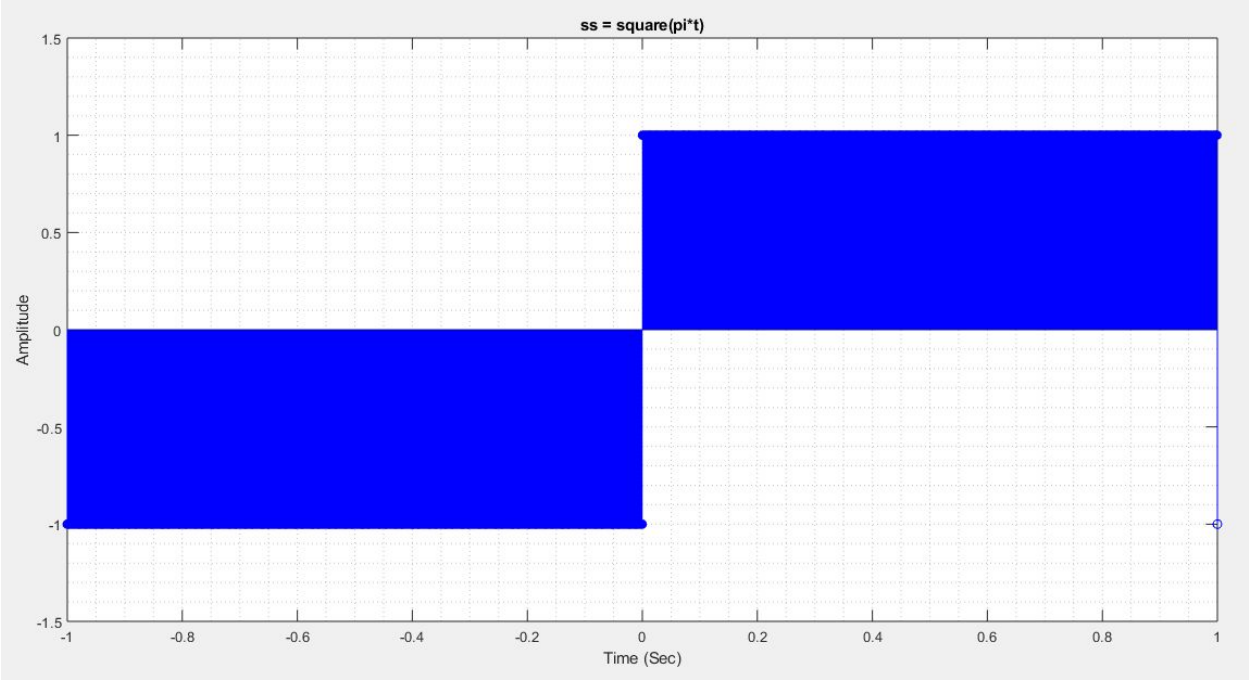

Figure 9: *MATLAB result of the square odd function*

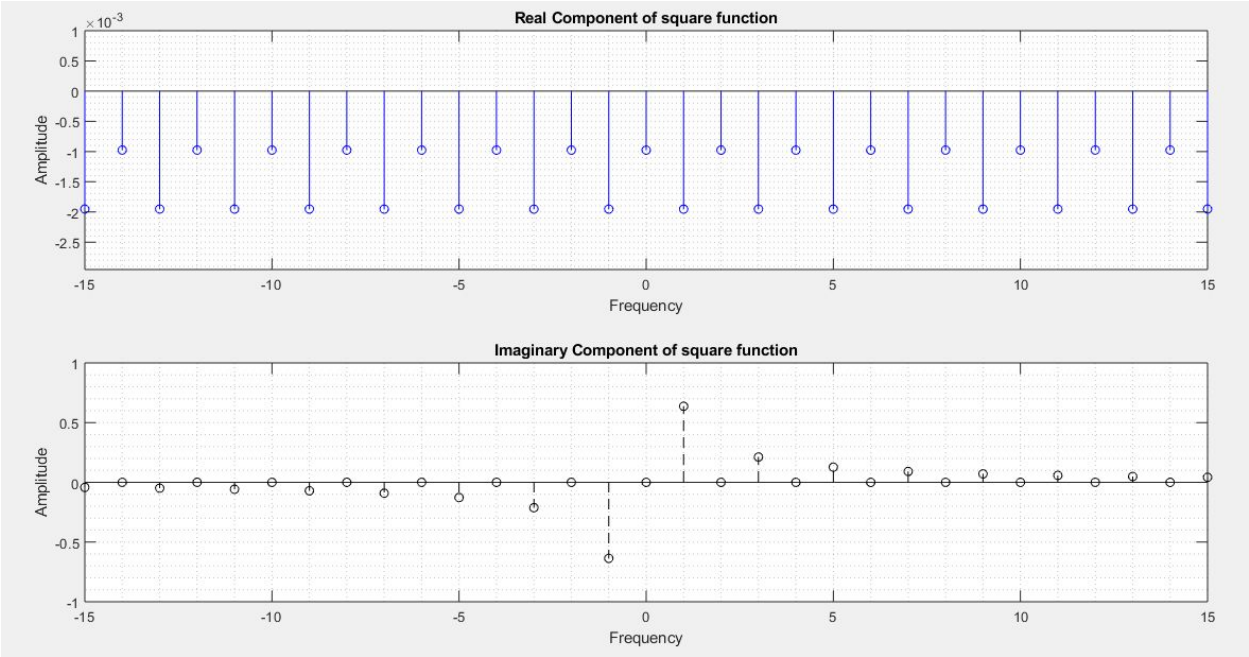

Figure 10: *Real and imaginary components of the square odd function*

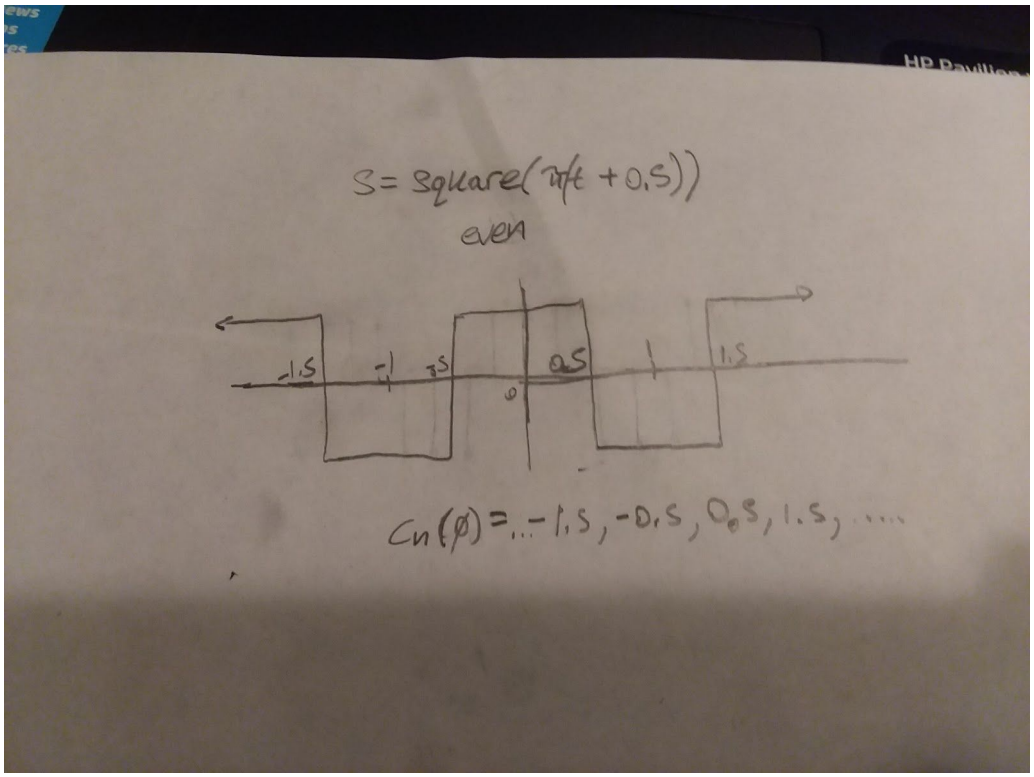

Figure 11: *Hand drawn version of the even square function*

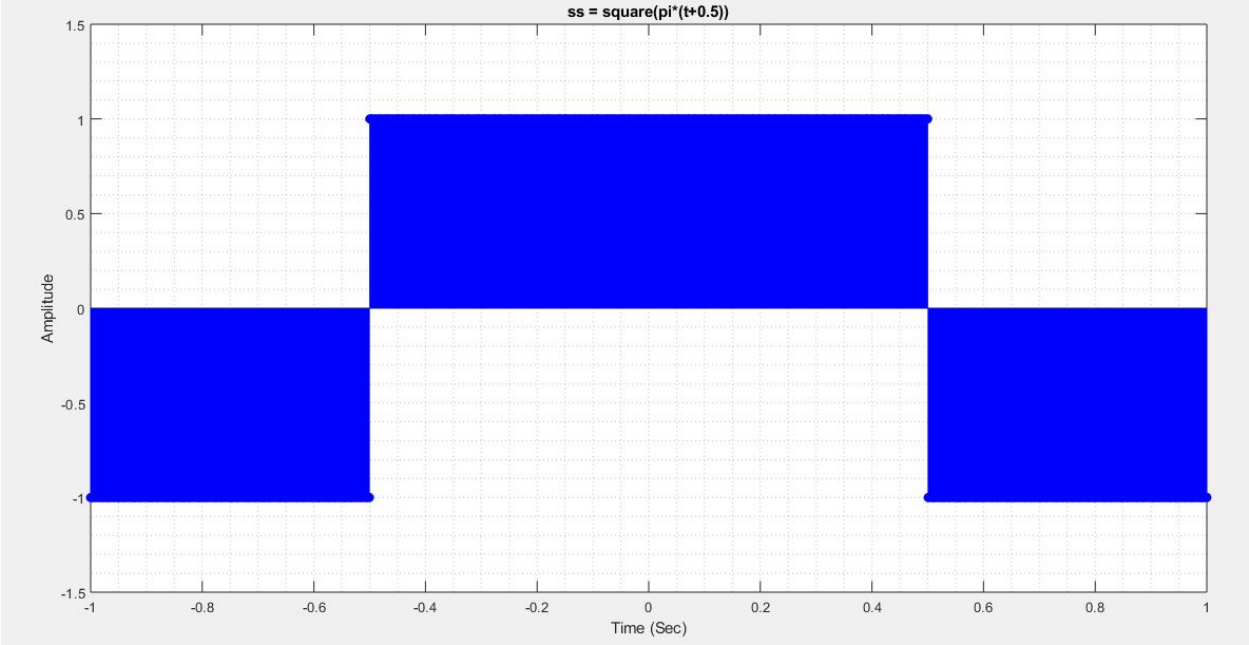

Figure 12: *MATLAB result of the square even function*

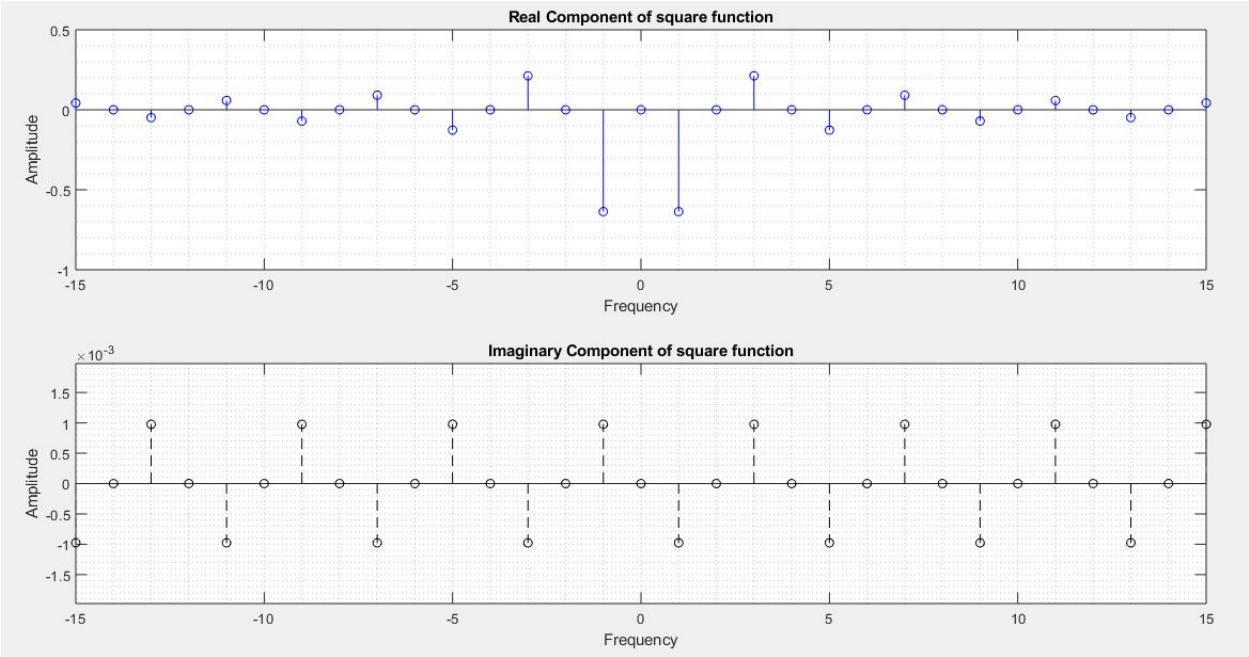

Figure 13: *Real and imaginary components of the square even function*

Part 1 MATLAB Code:

```
%% Lab2 Section 1: a_First Equation
clear all,clc;

% variables and equation
f  = 0.25;                    % fundamental frequency
fs = 4;                       % sample frequency
n  = 0:15;                    % number of sec on time axis
t  = f*n/fs;                  % time axis
s  = 2*cos(2*pi*t/fs+pi/3);          % the periodic signal

figure(1)
subplot(3,1,1)
stem(t,s)                     % stem plot of actual signal
grid minor
xlabel('Time (sec)'), xlim([-0.1 1])
ylabel('Amplitude'), ylim([min(s)-0.5 max(s)+0.5])
title('s = cos(2*pi*t/4)+1')

% setting magnitude and phase equations
S  = (0:1/length(s):0.5-1/length(s))*fs;    % half of the spectrum
SF = (1/length(s))*fft(s);
Sphase=angle(SF);

subplot(3,1,2)
stem(S,abs(SF(1:length(SF)/2)))          % Magnitude measurement
grid minor
xlabel('Time (sec)'), xlim([-0.2 2])
ylabel('Amplitude'), ylim([-0.5 0.6])
title('Magnitude')

subplot(3,1,3)
stem(S,unwrap(Sphase(1:length(SF)/2)))      % Phase measurement
grid minor
xlabel('Time (sec)'), xlim([-0.2 2])
ylabel('Amplitude'), ylim([-1 7])
title('Phase')
```

```
%% Lab2 Section 1: a_Second Equation
clear all,clc;

% variables and equation
f  = 0.25;                    % fundamental frequency
fs = 4;                       % sample frequency
n  = 0:15;                    % number of sec on time axis
t  = f*n/fs;                  % time axis
```

```
s  = cos(2*pi*t/fs)+1;                % the periodic signal

figure(2)
subplot(3,1,1)
stem(t,s)                             % stem plot of actual signal
grid minor
xlabel('Time (sec)'), xlim([-0.1 1])
ylabel('Amplitude'), ylim([0 2.5])
title('s = cos(2*pi*t/4)+1')

% setting magnitude and phase equations
S  = (0:1/length(s):0.5-1/length(s))*fs;    % half of the spectrum
SF = (1/length(s))*fft(s);
Sphase=angle(SF);

subplot(3,1,2)
stem(S,abs(SF(1:length(SF)/2)))             % Magnitude measurement
grid minor
xlabel('Time (sec)'), xlim([-0.2 2])
ylabel('Amplitude'), ylim([-0.5 2.5])
title('Magnitude')

subplot(3,1,3)
stem(S,unwrap(Sphase(1:length(SF)/2)))      % Phase measurement
grid minor
xlabel('Time (sec)'), xlim([-0.2 2])
ylabel('Amplitude'), ylim([-2.5 0.5])
title('Phase')
```

```
%% Lab2 Section 1: a_Third Equation
clear all,clc;

% variables and equation
f  = 0.25;                            % fundamental frequency
fs = 4;                               % sample frequency
n  = 0:15;                            % number of sec on time axis
t  = f*n/fs;                          % time axis
s  = cos(2*pi*t/fs)+sin(2*pi*t/8)+1;    % the periodic signal

figure(3)
subplot(3,1,1)
stem(s)                               % stem plot of actual signal
grid minor
xlabel('Time (sec)'),  xlim([0 17])
ylabel('Amplitude'), ylim([0 2.5])
title('s = cos(2*pi*t/4)+sin(2*pi*t/8)+1')

% setting magnitude and phase equations
```

```
S  = (0:1/length(s):0.5-1/length(s))*fs;    % half of the spectrum
SF = (1/length(s))*fft(s);
Sphase=angle(SF);

subplot(3,1,2)
stem(S,abs(SF(1:length(SF)/2)))            % Magnitude measurement
grid minor
xlabel('Time (sec)'), xlim([-0.2 1.8])
ylabel('Amplitude'), ylim([-0.5 2.5])
title('Magnitude')

subplot(3,1,3)
stem(S,unwrap(Sphase(1:length(SF)/2)))      % Phase measurement
grid minor
xlabel('Time (sec)'), xlim([-0.2 1.8])
ylabel('Amplitude'), ylim([-2.5 0.5])
title('Phase')
```

```
%% Lab2 Section 1: b-e
clear all,clc;

% variables and equation
N  = 2^11;                  % Number of points
t  = linspace(0,2,N);         % time sequence of 2sec and 2^11 points
ss = 2*cos(2*pi*t/2);          % periodic signal

figure(4)
stem(t,real(ss),'-b');         % stem of reg. real signal for 2sec
grid minor
xlabel('Time (Sec)')
ylabel('Amplitude'), ylim([-2.5 2.5])
title('ss = 2*cos(2*pi*t/2)')

cn  = fft(ss)/N;               % find the Fourier series
cn  = fftshift(cn);            % reorder the components
n   = -15:15;                  % define a few Cn near the origin
cnn = cn(1+(N/2)+n);           % cnn = components near the origin

figure(5)
subplot(2,1,1)
stem(n,real(cnn),'-b');        % stem of reg. real fft signal
grid minor
xlabel('Time (Sec)')
ylabel('Amplitude'), ylim([-0.5 1.2])
title('Real Component of cosine function')

subplot(2,1,2)
stem(n,imag(cnn),'--k');       % stem of reg. imag fft signal
```

```
grid minor
xlabel('Frequency')
ylabel('Amplitude')
title('Imaginary Component of cosine function')
```

```
%% Lab2 Section 1: f_odd
clear all,clc;

% variables and equation
N  = 2^11;                  % number of points
t  = linspace(-1,1,N);        % time sequence of 2sec and 2^11 points
ss = square(pi*t);            % periodic signal of 2*cos(2*pi*t/2)
                        % shifted by +90 degrees
figure(6)
stem(t,real(ss),'-b');        % stem of +90 deg phase shift
grid minor
xlabel('Time (Sec)')
ylabel('Amplitude'), ylim([-1.5 1.5])
title('ss = square(pi*(t+0.5))')

cn  = fft(ss)/N;              % find the Fourier series
cn  = fftshift(cn);           % reorder the components
n   = -15:15;                 % define a few Cn near the origin
cnn = cn(1+(N/2)+n);          % cnn = components near the origin

figure(7)
subplot(2,1,1)
stem(n,real(cnn),'-b');       % stem of reg. real fft signal
grid minor
xlabel('Frequency')
ylabel('Amplitude'), ylim([min(real(cnn))-1e-3 1e-3])
title('Real Component of square function')

subplot(2,1,2)
stem(n,imag(cnn),'--k');      % stem of reg. imag fft signal
grid minor
xlabel('Frequency')
ylabel('Amplitude'), ylim([-1 1])
title('Imaginary Component of square function')
```

```
%% Lab2 Section 1: f_even
clear all,clc;

% variables and equation
N  = 2^11;                  % number of points
t  = linspace(-1,1,N);        % time sequence of 2sec and 2^11 points
ss = square(pi*(t+0.5));      % periodic signal of 2*cos(2*pi*t/2)
                        % shifted by +90 degrees
```

```
figure(8)
stem(t,real(ss),'-b');          % stem of +90 deg phase shift
grid minor
xlabel('Time (Sec)')
ylabel('Amplitude'), ylim([-1.5 1.5])
title('ss = square(pi*(t+0.5))')

cn  = fft(ss)/N;                % find the Fourier series
cn  = fftshift(cn);             % reorder the components
n   = -15:15;                   % define a few Cn near the origin
cnn = cn(1+(N/2)+n);            % cnn = components near the origin

figure(9)
subplot(2,1,1)
stem(n,real(cnn),'-b');         % stem of reg. real fft signal
grid minor
xlabel('Frequency')
ylabel('Amplitude'), ylim([-1 0.5])
title('Real Component of square function')

subplot(2,1,2)
stem(n,imag(cnn),'--k');        % stem of reg. imag fft signal
grid minor
xlabel('Frequency')
ylabel('Amplitude')
ylim([min(imag(cnn))-1e-3 max(imag(cnn))+1e-3])
title('Imaginary Component of square function')
```

PART 2:

Question 1 i.A-C

With the first chirp signal, the initial frequency is 200Hz, lasts 3 seconds, the sample frequency is at 11025Hz, and 500Hz/sec. The sound starts at a low frequency, slowly builds up, and ends after 3 seconds. The amplitude slowly becomes bigger longer the signal played.

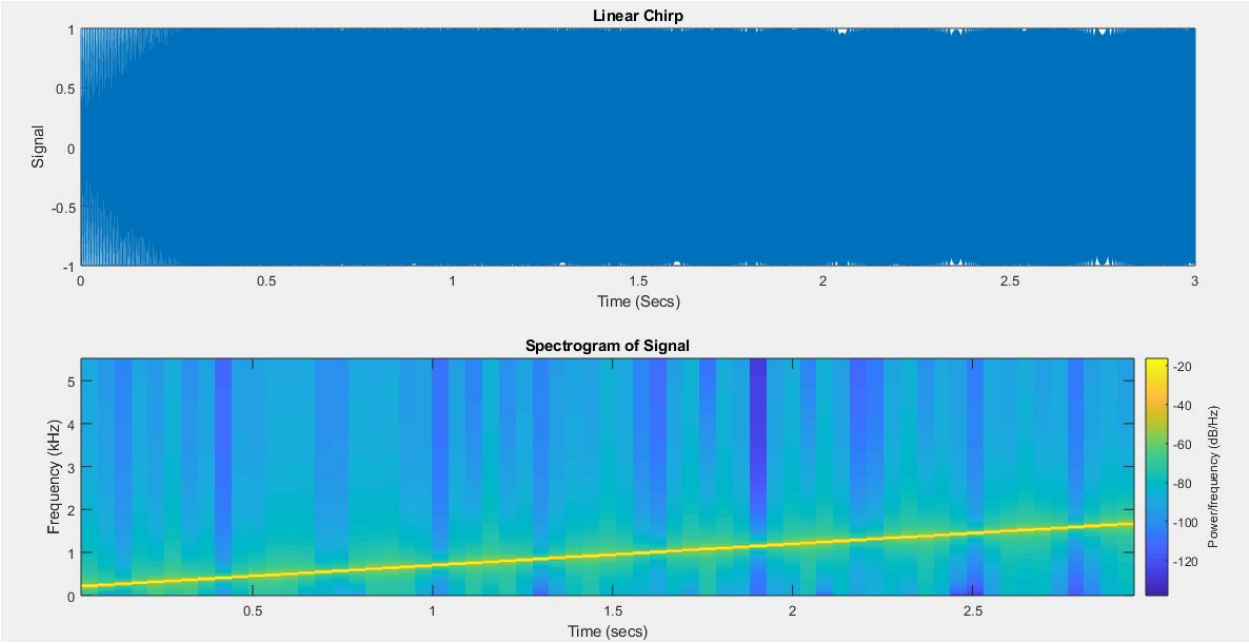

Figure 14: *The 1st chirp signal*

Question 1 ii.A-C

With the second chirp signal, the initial frequency is 4000Hz, the ending frequency is 600Hz, last 2 seconds, and the sample frequency is at 11025Hz. The sound starts at a frequency higher than where the last ended. It also gradually ends to a lower frequency.

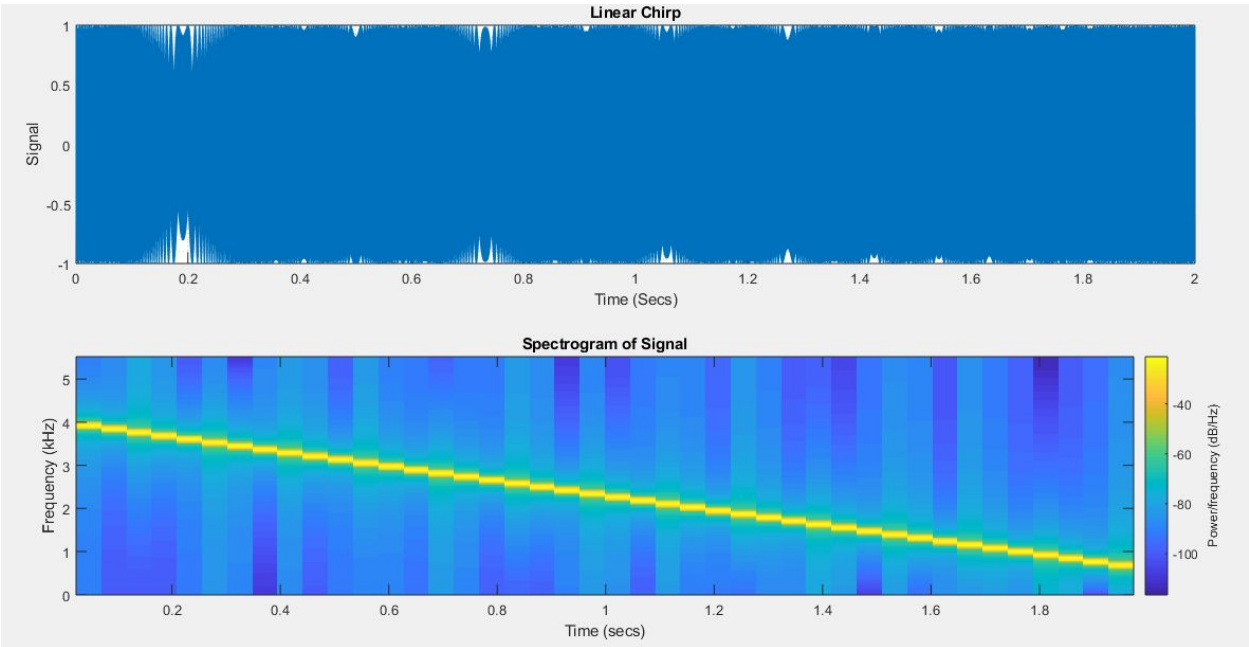

Figure 14: *The 2nd chirp signal*

Question 1 ii.D

The chirp sound mimicked the same frequency as the last and the spectrogram shows relatively same results. Except, for now, the y-axis limit set at 4000 max, which is 2x the initial frequency. And the signal had different size reading in amplitude due the lowered sampling frequency.

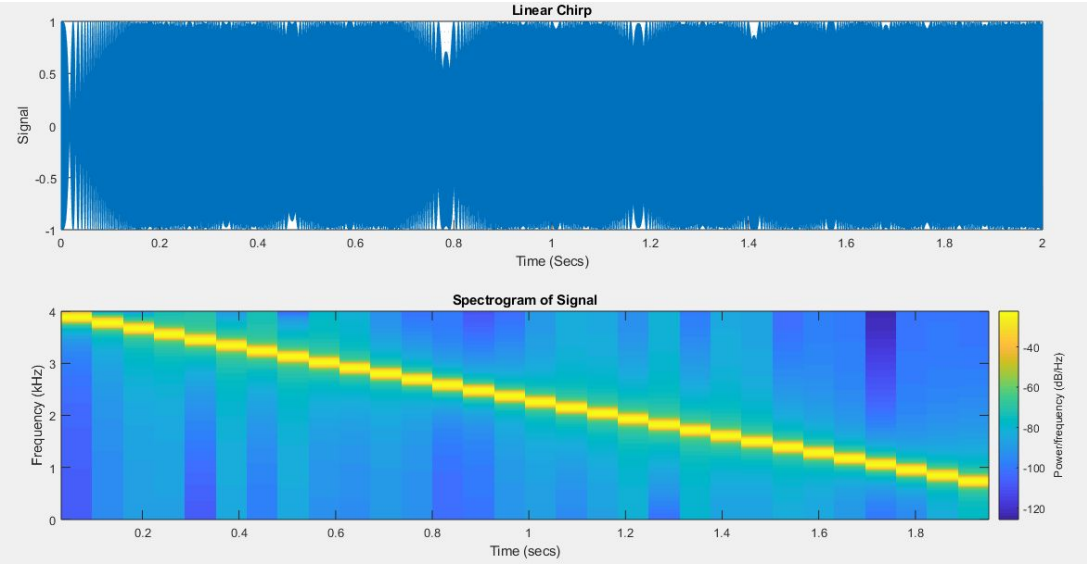

Figure 15: *The 2nd chirp signal where the sampling frequency is now 8kHz*

Question 1 ii.E

The chirp signal needed a sampling frequency 2x higher than the initial frequency and since it did not, it had to build up to where it initially be as if it were to reach 600Hz from 4Khz. The difference is a fractional difference from 2x the initial frequency.

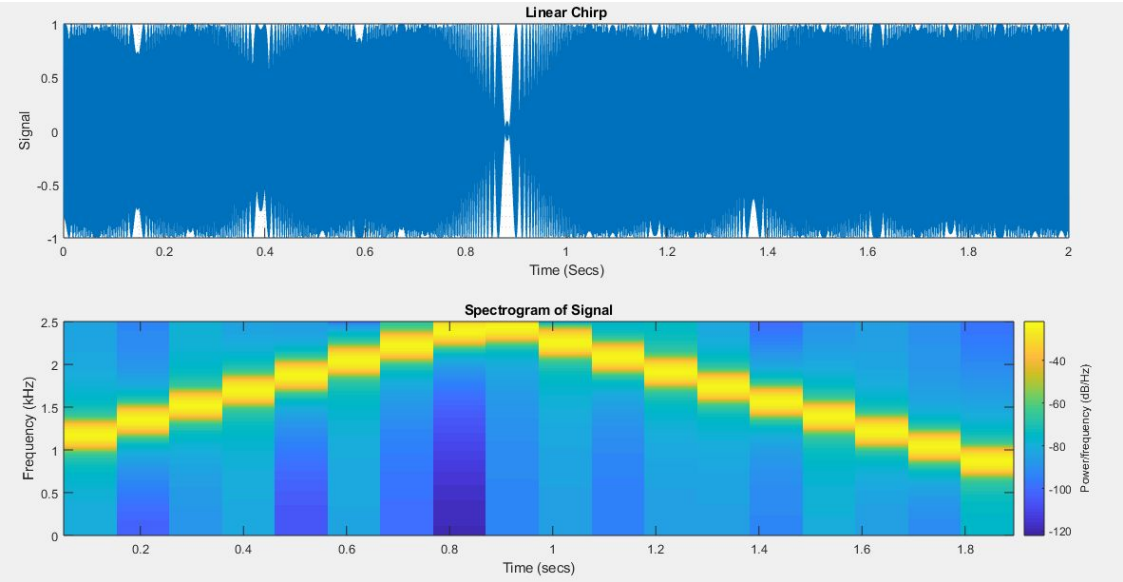

Figure 16: *The 2nd chirp signal where the sampling frequency is now 5kHz*

Question 2 A

On this beat signal, it produces a continuous amber alert like beeping sounds for 2 seconds with the parameters of the carrier frequency = 1500Hz, the carrier frequency = 25Hz, the sampling frequency = 11025Hz, and lasting 2 seconds.

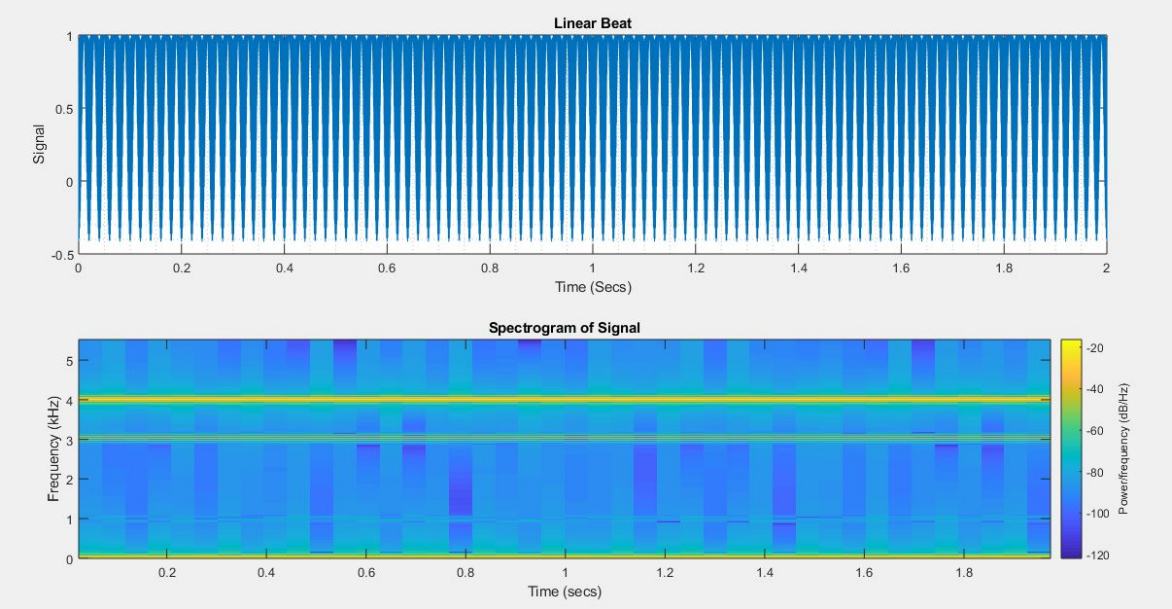

Figure 17: *The beat signal where the duration is 2 seconds long*

Question 2 B

This signal is the same as the last, but the duration is 0.5 seconds long. This signal produces the same sound, but the beeps are more spaced out revealing longer periods for each cycle. The sound runs only for 0.5 seconds.

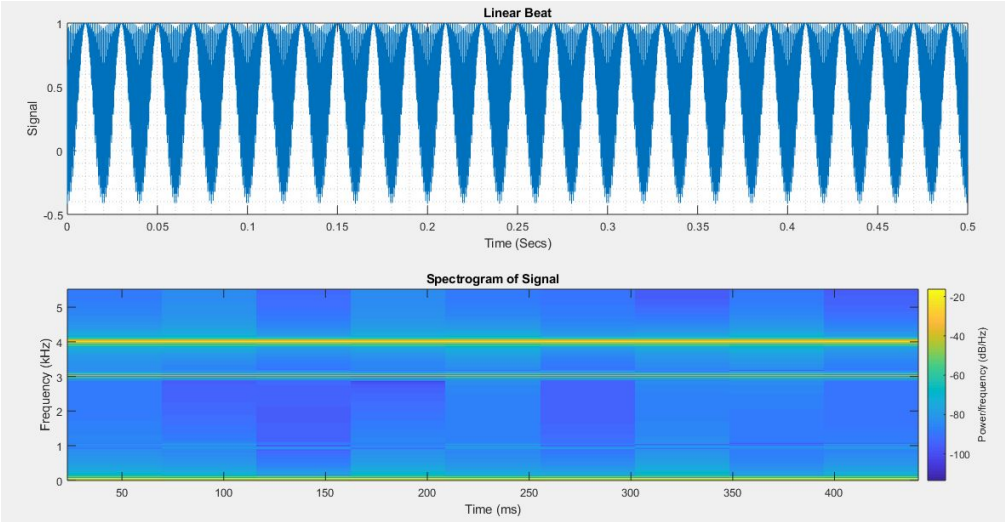

Figure 18: *The beat signal where the duration is 0.5 seconds long*

Question 2 C, D
The signal reads xt = 2*cos(2000*2*pi*t)*cos(25*2*pi*t) where 2000 is the carrier and 25 is the modulated frequency. The signal is periodic and is pinpointed at the period to prove so.

Part 2 MATLAB Code:

```
%% Lab2 Section 2-1: ai,c
%
mychirp(200,1700,3,11025,10)     % mychirp(fi,fend,dur,fsamp,FigNum)
pause(3)

%% Lab2 Section 2-1: aii,c
%
mychirp(4000,600,2,11025,11)     % mychirp(fi,fend,dur,fsamp,FigNum)
pause(2)

%% Lab2 Section 2-1: d
%
mychirp(4000,600,2,8000,12)     % mychirp(fi,fend,dur,fsamp,FigNum)
pause(2)

%% Lab2 Section 2-1: e
%
mychirp(4000,600,2,5000,13)     % mychirp(fi,fend,dur,fsamp,FigNum)
pause(2)

%% Lab2 Section 2-2: a
%
mybeat(1,1,2000,25,2,11025,14)  % mybeat(A,B,fc,fm,dur,fsamp,FigNum)
pause(2)

%% Lab2 Section 2-2: b
%
mybeat(1,1,2000,25,0.5,11025,15)% mybeat(A,B,fc,fm,dur,fsamp,FigNum)
pause(0.5)
```

MATLAB Functions Part 2:

mychirp.m

```
function mychirp(fi,fend,dur,fsamp,FigNum)
%
% MYCHIRP produces a linear chirp signal for chirp functions
%
% Usage:  psi = mychirp(fi,fend,dur,fsamp)
%         psi = 2*pi*u*t^2 + 2*pi*fi*t
```

```
%
% Where    fi = initial frequency
%        fend = ending frequency
%         dur = duration of the signal
%       fsamp = sampling frequency
%
% Returns:
%           x = linear chirp signal and spectrogram analysis

t = 0:1/fsamp:dur;                  % Number of samples for entire duration
u = (fend-fi)/(2*dur);              % change of frequency over time
psi = (2*pi*u.*t.^2)+(2*pi*fi.*t);  % signal breakdown
x = real(exp(1i*psi));              % real portion of signal
soundsc(x,fsamp);

figure(FigNum)
subplot(2,1,1)
plot(t,x)
title('Linear Chirp')
xlabel('Time (Secs)'),ylabel('Signal')
grid minor,box on

subplot(2,1,2)
spectrogram(x,1024,[],1024,fsamp,'yaxis')
title('Spectrogram of Signal')
end
```

---

mybeat.m

```
function mybeat(A,B,fc,fm,dur,fsamp,FigNum)
%
% MYBEAT produces a beat signal for beat functions
%
% Usage: xt = mybeat(A,B,fc,fm,dur,fsamp)
%        xt = 2*A*cos(2*pi*fm*t) * B*cos(2*pi*fc*t)         [multiplication form]
%        xt = A*cos(2*pi*(fc-fm)*t) + B*cos(2*pi*(fc+fm)*t) [summation form]
%
% Where    A = first amplitude
%          B = second amplitude
%         fc = carrier frequency
%         fm = modulated frequency
%        dur = duration of the signal
%      fsamp = sampling frequency
%
% Returns:
%         xt = beat signal and spectrogram analysis
```

```
t = 0:1/fsamp:dur;                          % Number of samples for entire duration

if A==B
   xt = 2*(A.*cos(2*pi*fc.*t)).*(B.*cos(2*pi*fm.*t));      % product form of beat signal

elseif A~=B
   xt = A.*cos(2*pi*(fc-fm).*t)+B.*cos(2*pi*(fc+fm).*t);   % summation form of beat signal

end

rsig = real(exp(1i*xt));                    % real portion of signal
soundsc(rsig,fsamp);

figure(FigNum)
subplot(2,1,1)
plot(t,rsig)
title('Linear Beat')
xlabel('Time (Secs)'),ylabel('Signal')
grid minor,box on

subplot(2,1,2)
spectrogram(rsig,1024,[],1024,fsamp,'yaxis')
title('Spectrogram of Signal')
end
```

PART 3:

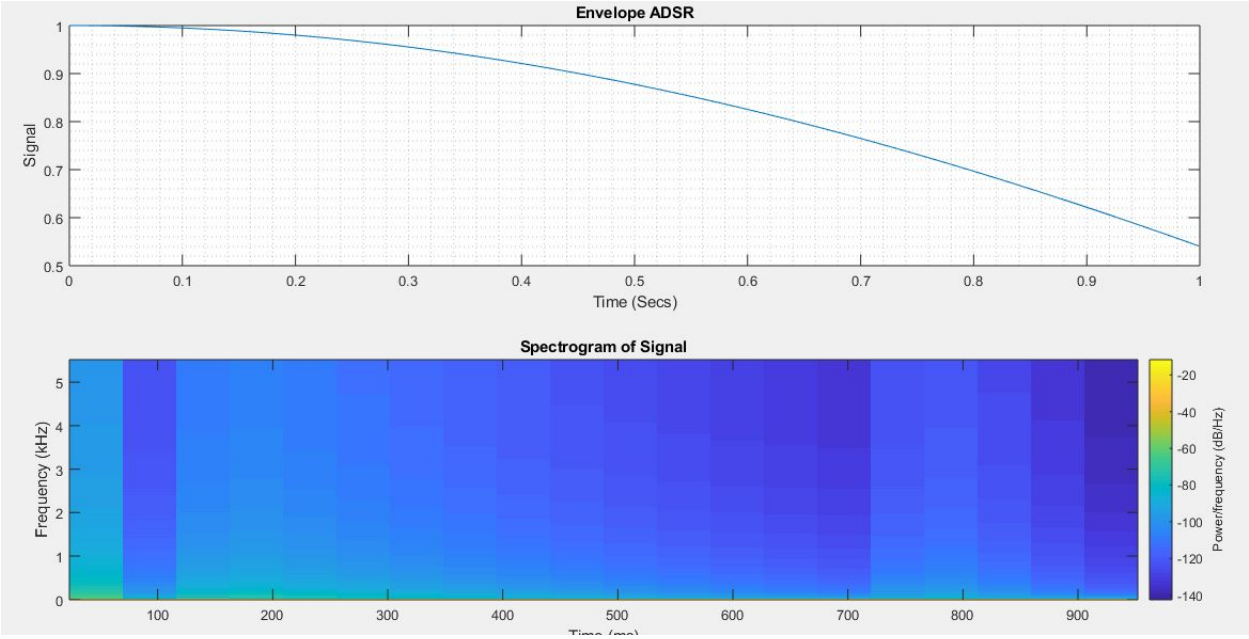

Figure 19: *The enveloped ADSR signal*

Part 3 MATLAB Code:

```
%% Lab2 Section 3
%
target = [1-eps;0.25;0];
gain = [0.005;0.0004;0.00075];
duration = [125;625;250];
a = adsr_gen(target,gain,duration,16);
soundsc(a,11025)
```

Part 3 MATLAB Functions:

```
function a = adsr_gen(target,gain,duration,FigNum)
%
% Input
% target - vector of attack, sustain, release target values
% gain - vector of attack, sustain, release gain values
% duration - vector of attack, sustain, release durations in ms
%
% Output
% a - vector of adsr envelope values

fs = 11025;
a = zeros(fs,1); % assume 1 second duration ADSR envelope
duration = round(duration./1000.*fs); % envelope duration in samp

% Attack phase
start = 2;
stop = duration(1);
for n = [start:stop]
 a(n) = target(1)*gain(1) + (1.0 - gain(1))*a(n-1);
end

% Sustain phase
start = stop + 1;
stop = start + duration(2);
for n = [start:stop]
 a(n) = target(2)*gain(2) + (1.0 - gain(2))*a(n-1);
end

% Release phase
start = stop + 1;
stop = fs;
for n = [start:stop]
 a(n) = target(3)*gain(3) + (1.0 - gain(3))*a(n-1);
```

```
end

rsig = real(exp(1i*a));                    % real portion of signal

figure(FigNum)
subplot(2,1,1)
plot(a,rsig)
title('Envelope ADSR')
xlabel('Time (Secs)'),ylabel('Signal')
grid minor,box on

subplot(2,1,2)
spectrogram(rsig,1024,[],1024,fs,'yaxis')
title('Spectrogram of Signal')
end
```

PART 4:

The signals that were covered were the odd numbered cases in the following figure:

CASE	f_c (Hz)	f_m (Hz)	I_0	τ (sec)	T_{dur} (sec)	f_s (Hz)
1	110	220	10	2	6	11,025
2	220	440	5	2	6	11,025
3	110	220	10	12	3	11,025
4	110	220	10	0.3	3	11,025
5	250	350	5	2	5	11,025
6	250	350	3	1	5	11,025

Figure 20: *Case scenario provided for the lab*

Through this table, the equation that was broken down the bellenv.m was $y(t) = e^{-t/\tau}$. For the bell's amplitude envelope, $A(t) = A_0e^{-t/\tau}$ and $I(t) = I_0e^{-t/\tau}$ for the modulation index envelope.

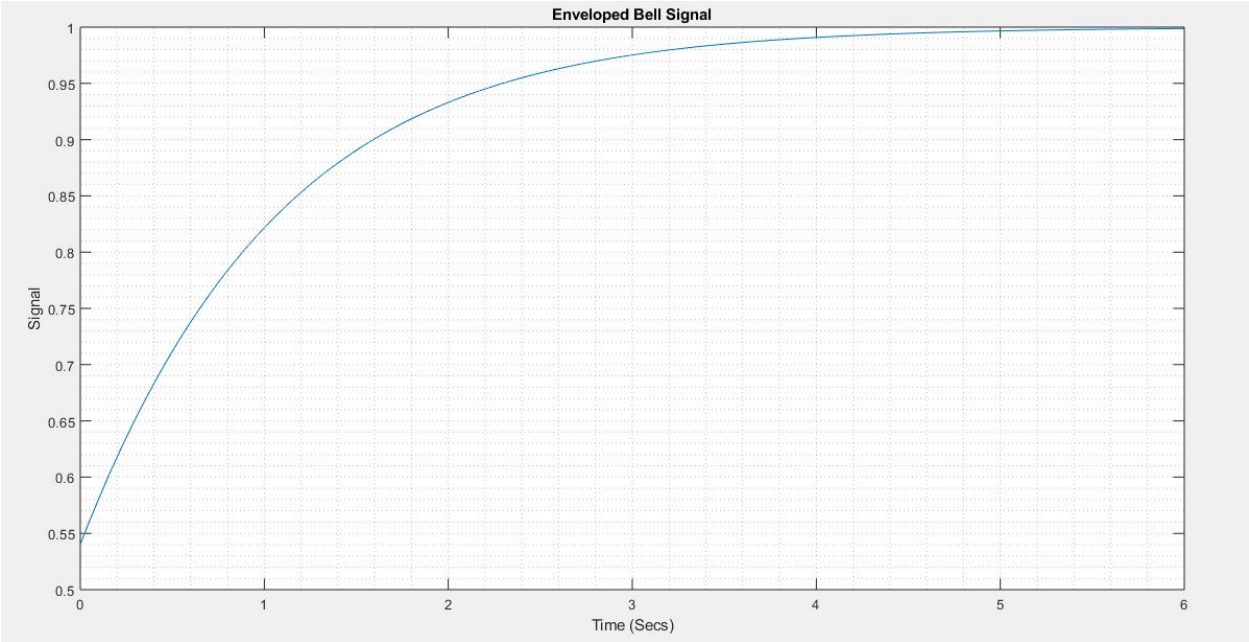

Figure 21: *The envelope of Case 1 as produced by* $y(t) = e^{-t/\tau}$.

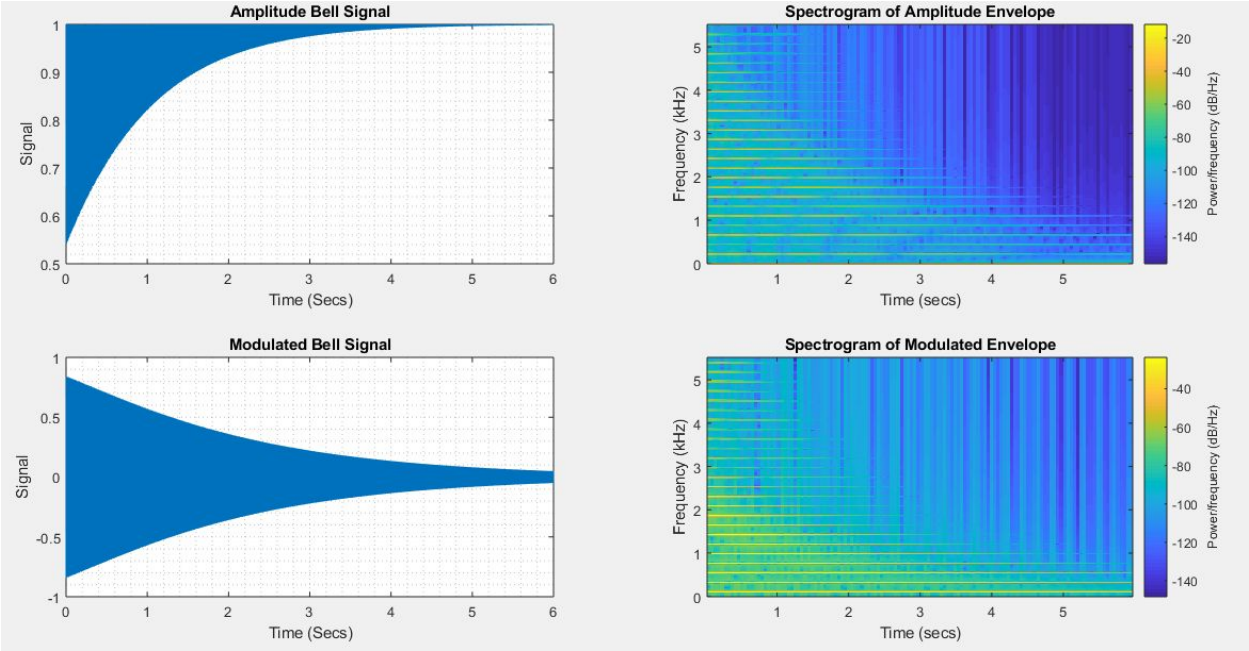

Figure 22: *Both the amplitude and modulated signals of same signal for Case 1*

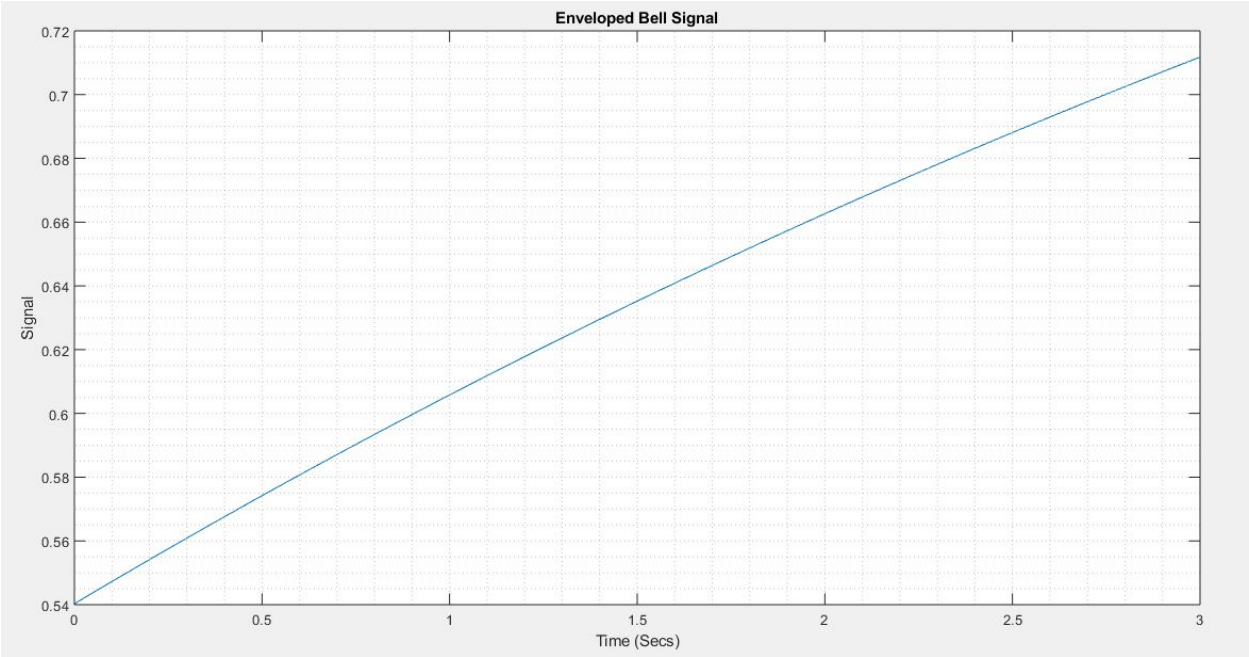

Figure 23: *The envelope of Case 2 as produced by* $y(t) = e^{-t/\tau}$.

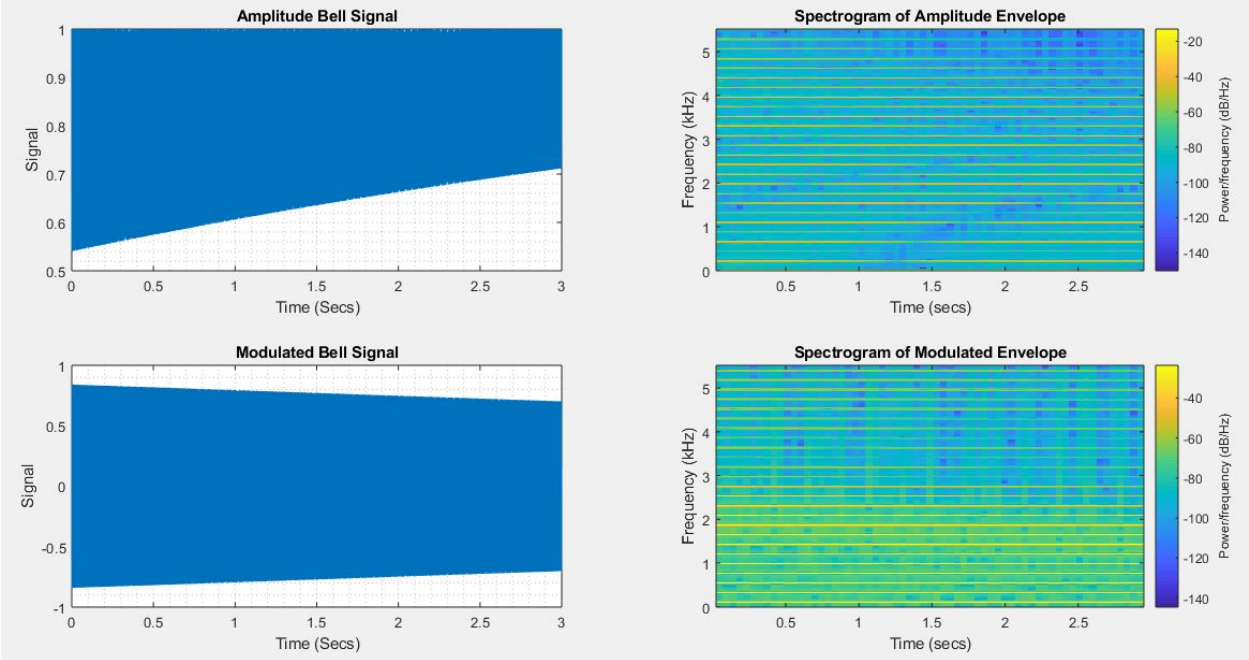

Figure 24: *Both the amplitude and modulated signals of same signal for Case 2*

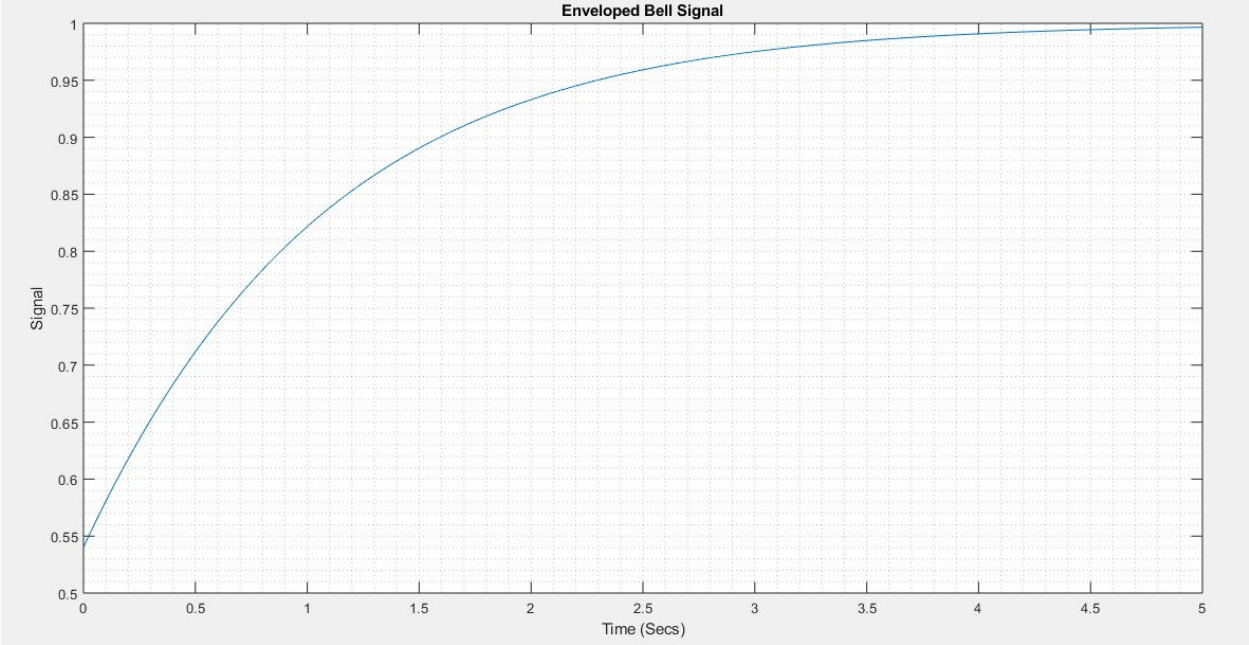

Figure 25: *The envelope of Case 2 as produced by* $y(t) = e^{-t/\tau}$.

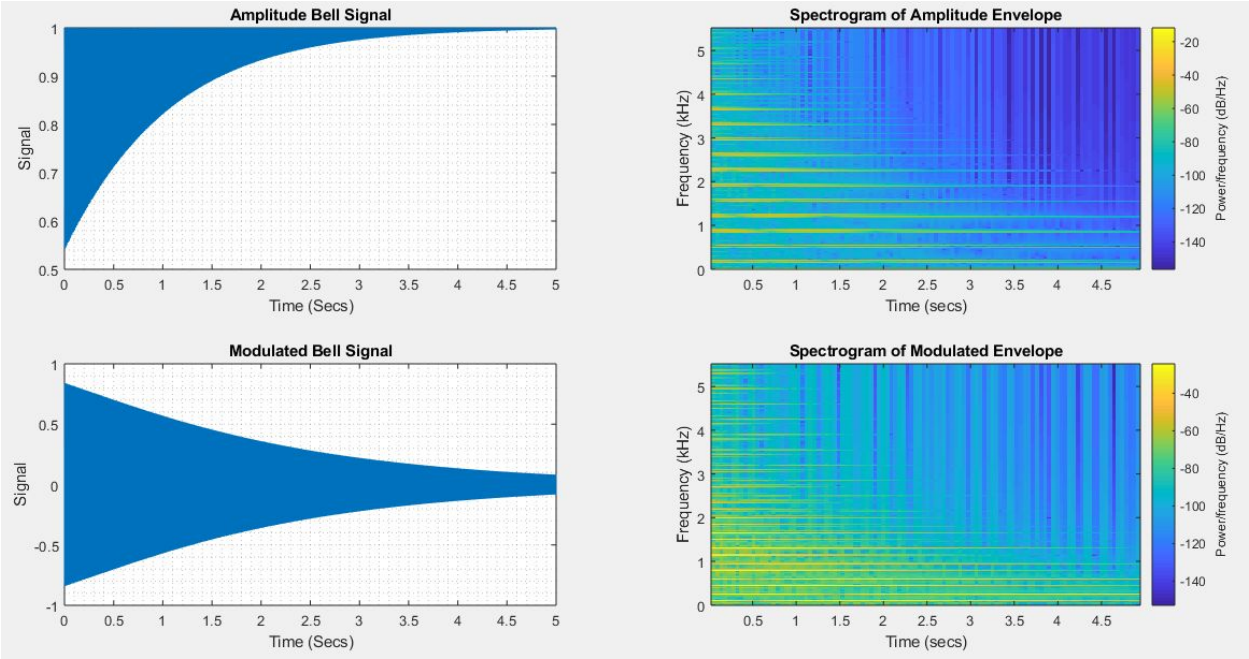

Figure 26: *Both the amplitude and modulated signals of same signal for Case 3*

Part 4 MATLAB Code:

```
%% Lab2 Section 4: Case 1
%
bellenv(2,6,11025,17);          % yy = bellenv(tau,dur,fsamp,FigNum)
pause(6)
bell(110,220,10,2,6,11025,18);  % xx = bell(fc,fm,tau,dur,fsamp,FigNum)
```

```
%% Section 4: Case 3
%
bellenv(12,3,11025,19);         % yy = bellenv(tau,dur,fsamp,FigNum)
pause(3)
bell(110,220,10,12,3,11025,20); % xx = bell(fc,fm,Io,tau,dur,fsamp,FigNum)
```

```
%% Section 4: Case 5
%
bellenv(2,5,11025,21);          % yy = bellenv(tau,dur,fsamp,FigNum)
pause(5)
bell(250,350,5,2,5,11025,22);   % xx = bell(fc,fm,tau,dur,fsamp,FigNum)
```

Part 4 MATLAB Functions:

bell.m

```
function xx = bell(fc,fm,Io,tau,dur,fsamp,FigNum)
%
% BELL produce a bell sound
%
% Usage: xx = bell(fc,fm,Io,tau,dur,fsamp)
%        xx = A(t)*cos(2*pi*fc*t + I(t)*cos(2*pi*fm*t))
%
% Where: fc = carrier freqency
%        fm = modulating frequency
%       tau = decay parameter for A(t) and I(t)
%       dur = duration (in sec.) of the output signal
%     fsamp = sampling rate
%
% Returns:
%        xx = bell sound signal and spectrogram analysis

t    = 0:1/fsamp:dur;                    % Number of samples for entire duration
yy   = exp(-t/tau);                      % A(t)
yy2  = Io*exp(-t/tau);                   % I(t)
xx   = yy.*cos((2*pi*fc.*t)+yy2.*cos(2*pi*fm.*t));  % full bell signal

rsig = real(exp(1i*xx));                 % real portion of signal
soundsc(rsig,fsamp);
```

```
pause(6)
rsig2 = imag(exp(1i*xx));
soundsc(rsig2,fsamp);

figure(FigNum)
subplot(2,2,1)
plot(t,rsig)
title('Amplitude Bell Signal')
xlabel('Time (Secs)'),ylabel('Signal')
grid minor,box on

subplot(2,2,2)
spectrogram(rsig,1024,[],1024,fsamp,'yaxis')
title('Spectrogram of Amplitude Envelope')

subplot(2,2,3)
plot(t,rsig2)
title('Modulated Bell Signal')
xlabel('Time (Secs)'),ylabel('Signal')
grid minor,box on

subplot(2,2,4)
spectrogram(rsig2,1024,[],1024,fsamp,'yaxis')
title('Spectrogram of Modulated Envelope')
end
```

---

bellenv.m

```
function yy = bellenv(tau,dur,fsamp,FigNum)
%
% BELLENV produces an envelope function for bell sounds
%
% Usage: yy = bellenv(tau, dur, fsamp);
%        yy = Ao*e^(-t/tau) = Io*e^(-t/tau)
%
% Where tau = time constant
%       dur = duration of the envelope
%     fsamp = sampling frequency
%
% Returns:
%        yy = decaying exponential envelope
%
```

```
% Note: produces exponential decay for positive tau

t    = 0:1/fsamp:dur;                % Number of samples for entire duration
yy   = exp(-t/tau);                  % full bell envelope expression
rsig = real(exp(1i*yy));             % real portion of signal
soundsc(rsig,fsamp);

figure(FigNum)
plot(t,rsig)
title('Enveloped Bell Signal')
xlabel('Time (Secs)'),ylabel('Signal')
grid minor,box on
end
```