Furkan Büyüksarıkulak

22002097

06.02.2025

EEE 321
Signals and Systems
Spring 2024-2025
Lab Assignment 1

In this lab task, we aimed to create various sinusoidal signals using matlab software and listen to these signals. We observed how the generated signals behaved in operations such as addition and multiplication or in different frequencies, amplitudes and phases.

Part 1: Fundamental Frequency and the Harmonics

In this part we created the sinusoidal signal in the form of $x_1(t) = \sin(2\pi\ f_0 t)$ at discrete instants of time. We chose 440 Hz as the starting frequency. We set the uniform sample intervals to 0.0001 s and discretize the signal over 3 s. In order to observe the waves more clearly, we limited the x-axis to 0.01 s with the xlim command. The generated sinusoidal signal can be seen in Figure 1. We also listened to the signal we generated using the soundsc(.) function. What was heard was A musical note.

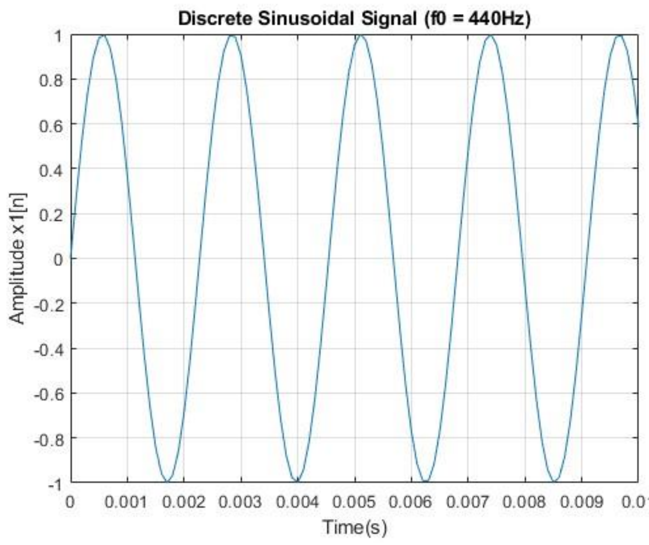

Figure 1 plot of a discrete sinusoidal signal (f0=440 Hz)

Then we tried the same signal by increasing the frequency value by 2 times and 4 times. We heard that the sound heard became thinner. Also, as seen in Figure 2 and Figure 4, the number of waves seen in the first 0.01 seconds increased as the frequency increased. This is due to the decreasing period as the frequency increased.

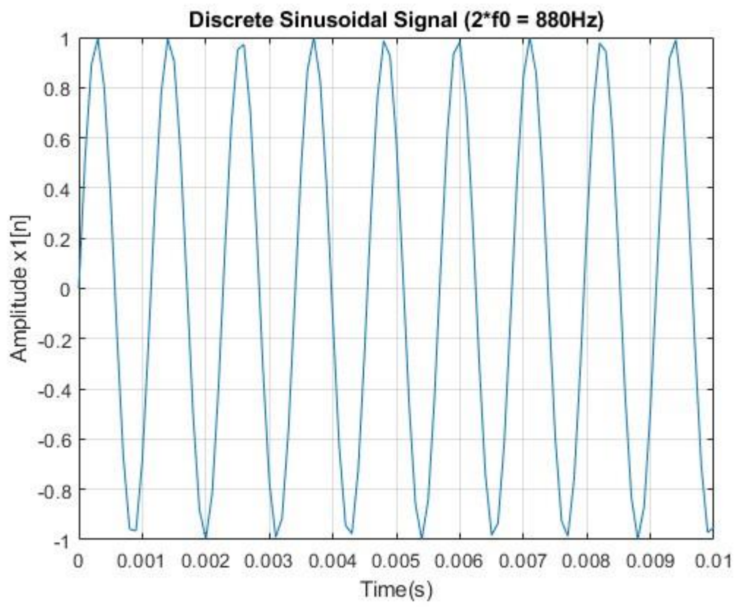

Figure 2 plot of a discrete sinusoidal signal (f0=880 Hz)

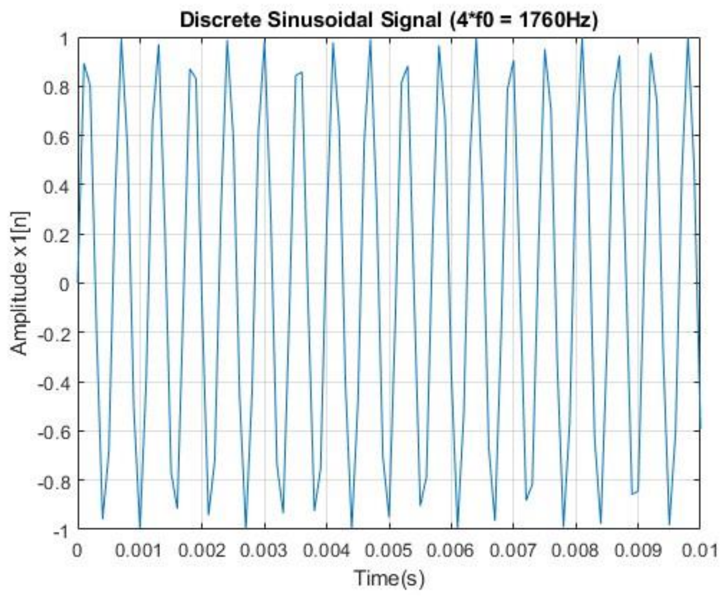

Figure 3 plot of a discrete sinusoidal signal (f0=1760 Hz)

Using the same steps, we created the notes C sharp (554 Hz) and E (659 Hz), whose frequencies we know, and the E note sounds thinner because of its higher frequency.
Using the same steps, we created the notes C sharp (554 Hz) and E (659 Hz), whose

frequencies we know, and the E note was higher frequency, so it sounded thinner. The signal form of the notes indicated in Figures 4 and 5 can be seen.

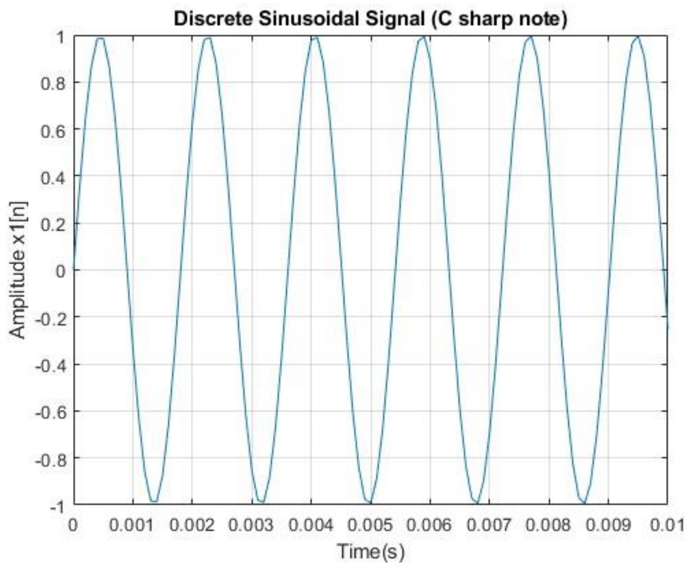

Figure 4 plot of a discrete sinusoidal signal C sharp note

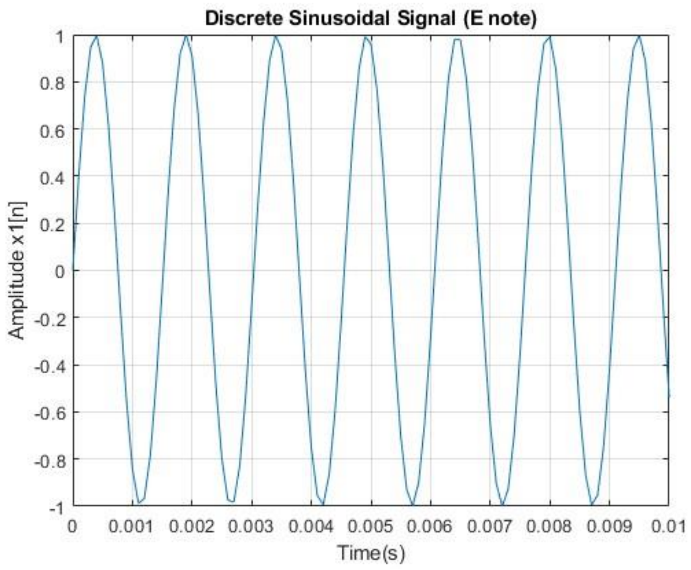

Figure 5 plot of a discrete sinusoidal signal E note

After these, we tried to collect sine signals. We collected the signals of A C sharp and E notes as s(t) = sin(2π440 t) + sin(2π554 t) + sin(2π659 t). Due to the harmonic relationship between these notes, they create a beautiful sound together and this accounts fort he predominance of the majör triad in western music. We recorded this signal in the array named s and listened to it with the soundsc(.) function. We set the duration to 3 seconds and it had a stable and satisfying sound. The waveform can be observed in figure 6.

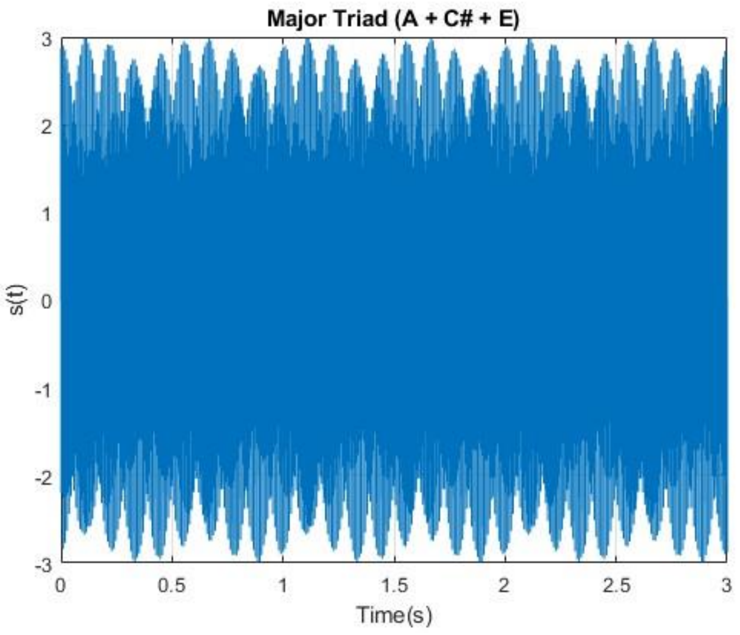

Figure 6 plot of a discrete sinusoidal signal Major Triad (A+ C# + E)

Part 2: The Effect of Phase

In this part, we examined the phase effect on the sinusoid signal. The signal form we used was $x_2 = \cos(2\pi f_0 t + \emptyset)$. f$_0$ frequency is set to 587Hz. The note we get at this frequency is 'D'. The waveform we observe with 0 phase is as shown in the figure.

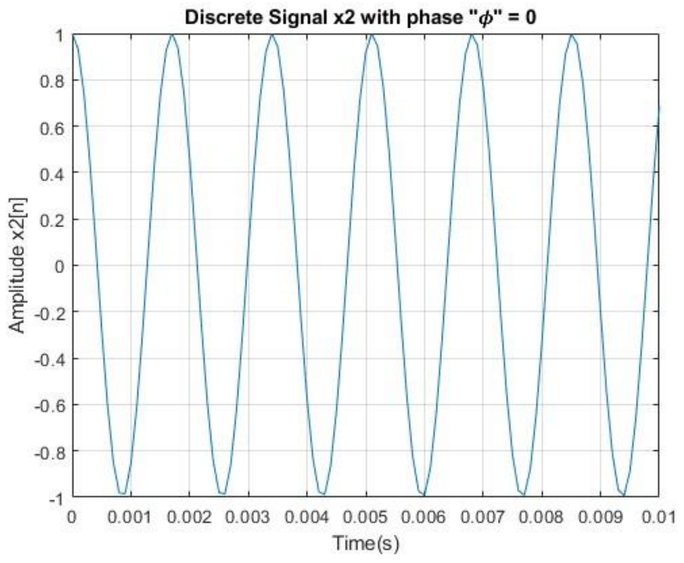

Figure 7 plot of a discrete sinusoidal signal zero phase

We repeated the same process in the following phases: pi/4, pi/2, 3*pi/4 and pi. Since the phase change on a signal does not affect the frequency, no change in the sound heard could be observed, but the waveforms we obtained showed a shift in phase dimension. We can examine this by observing the amplitudes at the clearest point 0 in Figure 8.

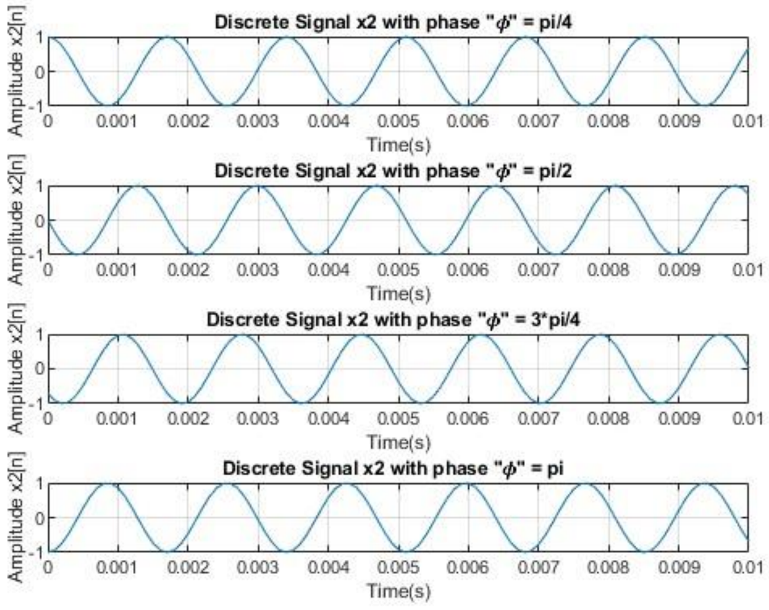

Figure 8 plot of a discrete sinusoidal signal at different phases

Part 3: Sinusoid with Exponentially Decaying Envelope

The x3 equation given in this part:

$$x_3(t) = e^{-(a^2+2)t}\cos(2\pi f_0 t)$$

was computed by taking a as 2 and f_0 as 440Hz. To make element wise multiplication in matlab I use the multiplication operation with dot (.*) and we get the array as

```
        x3 = exp(-(a^2+2)*t) .* cos(2*pi*inifreq*t);
```

Then similar to part 1 we listen the music with soundsc function. Due to the exponential part, we observed a decrease in the amplitude of the signal over time. The decreasing amplitude over time caused the sound to not be heard throughout its duration, but since this was not the case in Part 1, we did not have any problems hearing the sound. Because of the reduction in volume, the x3 signal sounded more like a piano note. The signal formed when a is 2 can be observed in Figure 9.

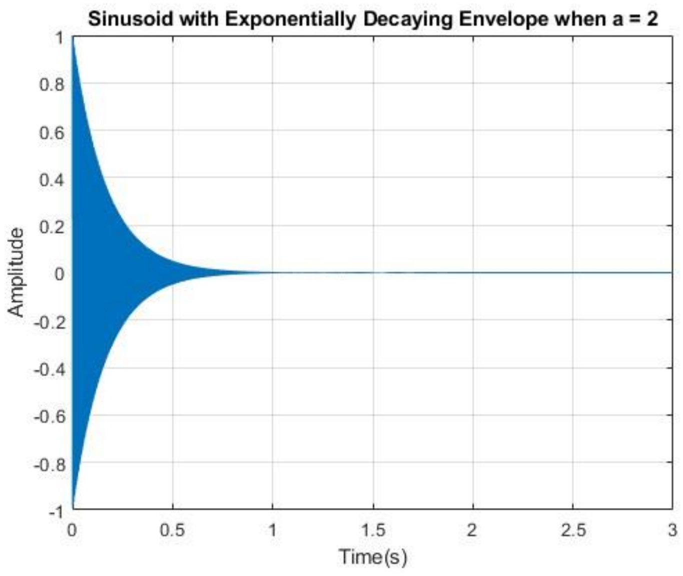

Figure 9 plot of a discrete sinusoidal signal with exponentially decaying envelope at a =2

Then we set the damping factor a to 3 and 1. In this way, we observed that when a was 3, the signal dropped to inaudible levels much faster, while when a was 1, this happened a little longer. We can observe these changes from the amplitude decrease and shape of the waves in Figures 10 and 11.

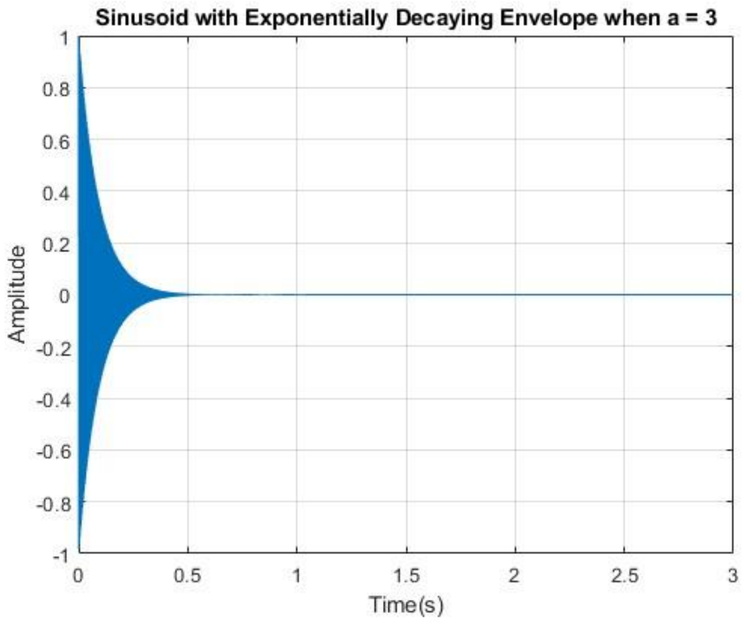

Figure 9 plot of a discrete sinusoidal signal with exponentially decaying envelope at a =3

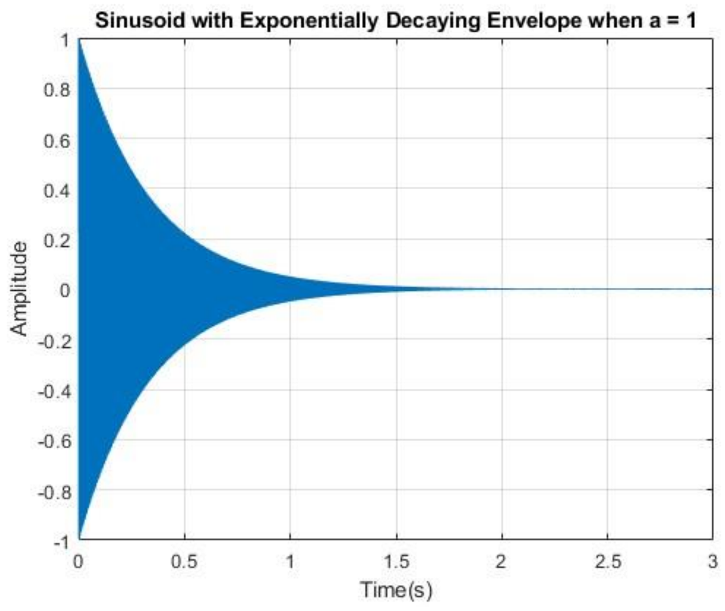

Figure 10 plot of a discrete sinusoidal signal with exponentially decaying envelope at a =1

Part 4: Beat Notes and Amplitude Modulation

Beat note is a audio effect and it is created by multiplication of two sinusoids with different frequencies. While picking one frequency very small and other around 1 we can get best sound which is like warble. To get this type of signal we consider product of two cosine signal as:

```
x4 = cos(2*pi*f1*t) .* cos(2*pi*f2*t);
```

I plotted and listened the signal that created, and I observe beat note with the effect of product of cosine term but I couldn't observe that effect at part 1.

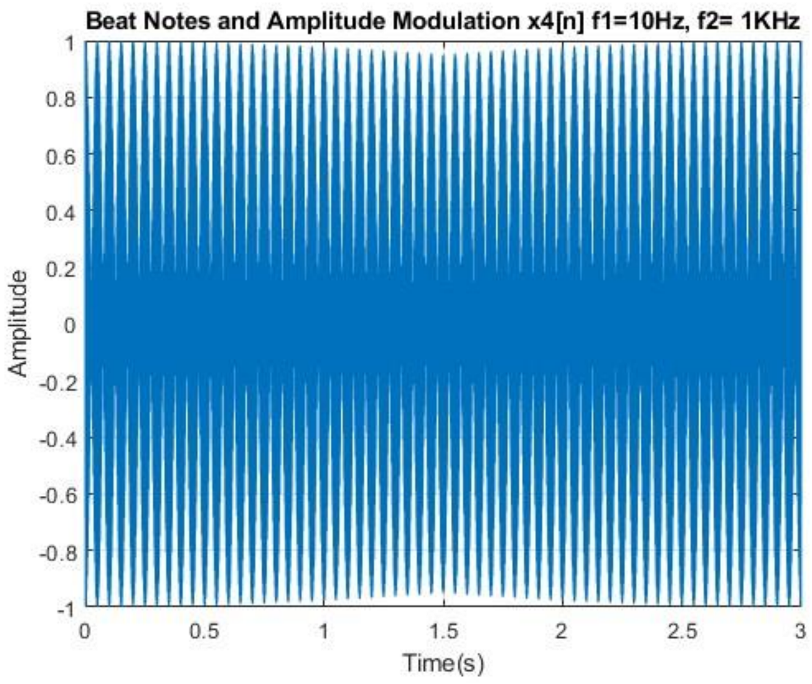

Figure 11 plot of a Beat notes and amplitude modulation when f1 = 10Hz, f2 = 1 KHz

Then we plotted again by changing the f1 frequency to 5 and 15 and listened to the sound. While we observed slower warbler at 5, there was more warbler at 15, which caused a thinner sound.

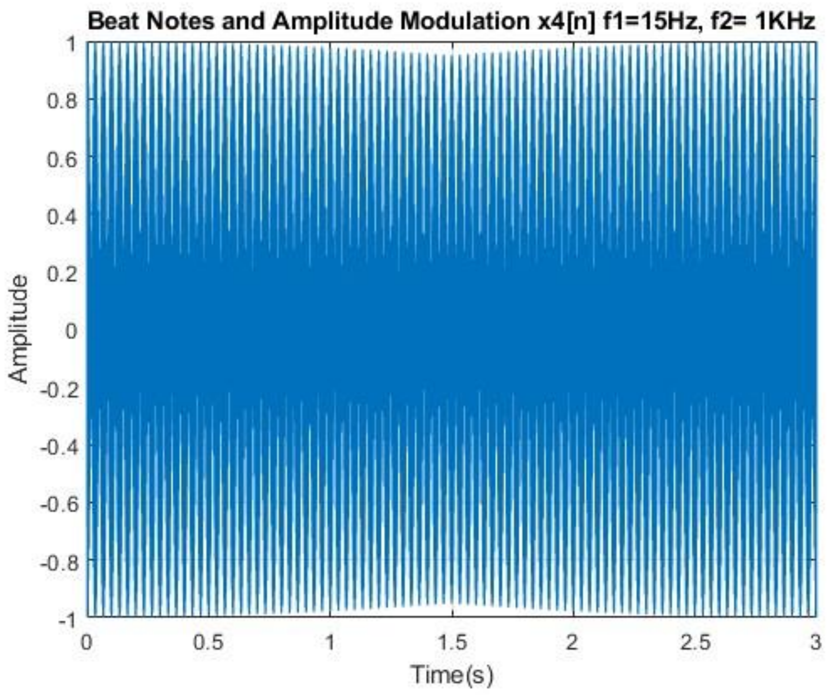

Figure 12 plot of a Beat notes and amplitude modulation when f1 = 15Hz, f2 = 1 KHz

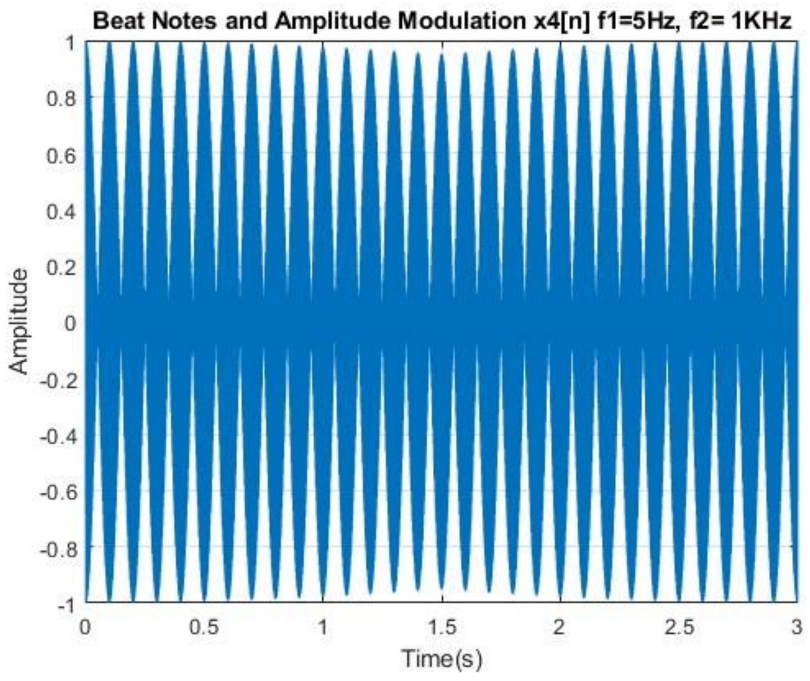

Figure 13 plot of a Beat notes and amplitude modulation when f1 = 5Hz, f2 = 1 KHz

With the well known trigonometric identity

$$cos(\theta 1)cos(\theta 2) = \frac{1}{2}\,[cos(\theta 1\; +\; \theta 2)\; +\; cos(\theta 1\; -\; \theta 2)]$$

Our signal can be expressed as sum of two different cosine signals where f1 is smaller than f2:

```
x4 = 0.5 * (cos(2*pi*(f1+f2)*t) + cos(2*pi*(f2-f1)*t));
```

If two notes in a signal are close to each other and their frequencies are not the same, the beat phenomenon is heard. And if one pitch is brought close to the other, we can no longer observe this effect and this situation is called “in tune”

Part 5: Chrip Signals and Frequency Modulation

In this section, we examined signals that vary as a function of time. Signals whose frequencies are linearly swept are called chrip signals. They vary linearly between the starting and ending points. The signal we created using this logic is

$$x5(t) = \cos(2\pi\mu t\ 2\; +\; 2\pi f0t\; +\; \emptyset)$$

We set the starting frequency to 2500 and the ending frequency to 500 Hz. We set the duration to 2s and used the following equation to find μ.

$$fi(t) = 2\mu t + f0$$

The matlab code version of this calculation is

```
u = (f1-f0)/(2*Dur);
```

I listened to the sound with the settings we used and then changed the frequency from 500 Hz to 2500 Hz and observed again. I heard that the pitch sound was low in the first installation, but when I observed the situation where it went from low frequency to high frequency, I observed an increase in the pitch sound due to time shift. I also observed the how effect changing of the µ value to the signal . As the µ value increases, the frequency change becomes faster, while as µ decreases, the frequency change becomes slower. The fast frequency change creates high pitch sounds.

A Chrip Puzzle

In this section, we created a chirp signal with a starting frequency of 3000Hz and an ending frequency of -2000Hz. Thanks to the negative value here, the chirp signal was observed to trend downwards for a while and then rose. The graph of this effect can be seen below.

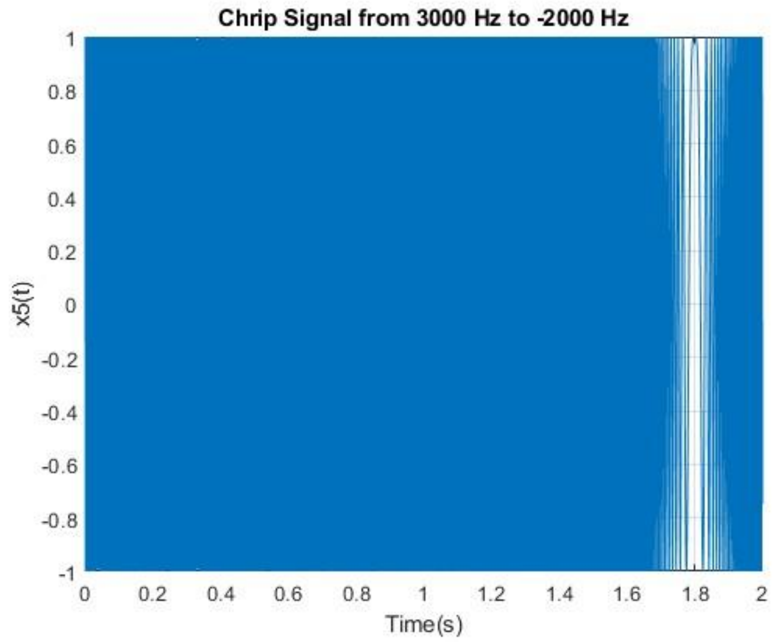

Figure 14 plot of a Chrip signal from 3000 Hz to -2000 Hz

Part 6: Composing Music

In this section, I wrote and saved the notecreate function given at the beginning in a different script. The purpose of this function is to generate sine signals for the desired musical note. The function given below obtains the desired sound wave by matching the notes specified in the song section with the note names and sending the frequency of the corresponding note to the notecreate function. The reason for using the zeros command is to ensure that the notes are distinguished by adding short silences in between. Finally, we listen to the signal with the soundsc command that we used in the previous parts. I created a melody myself using this code.

Conclusion

In this lab task we studied the behavior of various sinusoidal signals. We listened to musical sounds using sinusoidal signals and observed how frequency and phase values affect these sounds. We observed how the multiplication or sum of two signals results in a decrease in amplitude when a sinusoidal signal is multiplied by an exponential function. We learned how the chirping signal behaves. We also learned how to create these signals via MATLAB and how to make songs using notes. In this way, all stages of this lab task were completed successfully.

Codes

Part 1: Fundamental Frequency and the Harmonics

```
%%
% *Part 1: Fundamental Frequency and the Harmonics*

f0 = 440;
Ts = 0.0001;
dur = 3.0;
t = 0:Ts:dur; %time vector

%--f0

x1 = sin(2*pi*f0*t);


figure;
plot(t,x1);
xlabel('Time(s)');
ylabel('Amplitude x1[n]');
title('Discrete Sinusoidal Signal (f0 = 440Hz)');
xlim([0 0.01]);
ylim([-1 1]);
grid on;
```

```matlab
soundsc(x1,10000);
pause(3);

%--2f0
f = 2*f0;
x1 = sin(2*pi*(f)*t);


figure;
plot(t,x1);
xlabel('Time(s)');
ylabel('Amplitude x1[n]');
title('Discrete Sinusoidal Signal (2*f0 = 880Hz)');
xlim([0 0.01]);
ylim([-1 1]);
grid on;


soundsc(x1,10000);
pause(3);

%-- 4f0
f = 4*f0;
x1 = sin(2*pi*(f)*t);


figure;
plot(t,x1);
xlabel('Time(s)');
ylabel('Amplitude x1[n]');
title('Discrete Sinusoidal Signal (4*f0 = 1760Hz)');
xlim([0 0.01]);
ylim([-1 1]);
grid on;


soundsc(x1,10000);
pause(3);

%-- Csharp note
fcsh = 554;

x1 = sin(2*pi*(fcsh)*t);


figure;
plot(t,x1);
xlabel('Time(s)');
ylabel('Amplitude x1[n]');
title('Discrete Sinusoidal Signal (C sharp note)');
xlim([0 0.01]);
ylim([-1 1]);
grid on;


soundsc(x1,10000);
pause(3);
```

```
%-- E nore

fe = 659;

x1 = sin(2*pi*(fe)*t);


figure;
plot(t,x1);
xlabel('Time(s)');
ylabel('Amplitude x1[n]');
title('Discrete Sinusoidal Signal (E note)');
xlim([0 0.01]);
ylim([-1 1]);
grid on;


soundsc(x1,10000);
pause(3);


%-- Major Triad

s = sin(2*pi*440*t)+sin(2*pi*554*t)+sin(2*pi*659*t);

figure;
plot(t,s);
xlabel('Time(s)');
ylabel('s(t)');
title('Major Triad (A + C# + E)');

grid on;


soundsc(s);
%-----------------------------------------------
```

Part 2: The Effect of Phase

```
%%
% *Part 2: The Effect of Phase*

f0 = 587;
Ts = 0.0001;
dur = 3.0;
t = 0:Ts:dur;



%--phase0

phase =0;

x2 = cos(2*pi*f0*t + phase);


figure;
```

```
plot(t,x2);
xlabel('Time(s)');
ylabel('Amplitude x2[n]');
title('Discrete Signal x2 with phase "ϕ" = 0  ');
xlim([0 0.01]);
ylim([-1 1]);
grid on;



soundsc(x2,10000);
pause(3);


%--phase pi/4

phase = pi/4;

x2_1 = cos(2*pi*f0*t + phase);


figure;
subplot(4,1,1);
plot(t,x2);
xlabel('Time(s)');
ylabel('Amplitude x2[n]');
title('Discrete Signal x2 with phase "ϕ" = pi/4  ');
xlim([0 0.01]);
ylim([-1 1]);
grid on;



soundsc(x2_1,10000);
pause(3);


%--phase pi/2

phase = pi/2;

x2_2 = cos(2*pi*f0*t + phase);



subplot(4,1,2);
plot(t,x2_2);
xlabel('Time(s)');
ylabel('Amplitude x2[n]');
title('Discrete Signal x2 with phase "ϕ" = pi/2  ');
xlim([0 0.01]);
ylim([-1 1]);
grid on;


soundsc(x2_2,10000);
pause(3);

%--phase 3pi/4
```

```
phase = 3*pi/4;

x2_3 = cos(2*pi*f0*t + phase);



subplot(4,1,3);
plot(t,x2_3);
xlabel('Time(s)');
ylabel('Amplitude x2[n]');
title('Discrete Signal x2 with phase "ϕ" = 3*pi/4  ');
xlim([0 0.01]);
ylim([-1 1]);
grid on;


soundsc(x2_3,10000);
pause(3);

%--phase pi

phase = pi;

x2_4 = cos(2*pi*f0*t + phase);



subplot(4,1,4);
plot(t,x2_4);
xlabel('Time(s)');
ylabel('Amplitude x2[n]');
title('Discrete Signal x2 with phase "ϕ" = pi  ');
xlim([0 0.01]);
ylim([-1 1]);
grid on;


soundsc(x2_4,10000);
pause(3);

%-----------------------------------------
```

Part 3: Sinusoid with Exponentially Decaying Envelope

```
%%
% *Part 3: Sinusoid with Exponentially Decaying Envelope*


Ts = 0.0001;
inifreq = 440;
dur = 3.0;
t = 0:Ts:dur;


%-- a=2
a=2;
```

```
x3 = exp(-(a^2+2)*t) .* cos(2*pi*inifreq*t);



figure;
plot(t,x3);
xlabel('Time(s)');
ylabel('Amplitude');
title('Sinusoid with Exponentially Decaying Envelope when a = 2');
xlim([0 3]);
ylim([-1 1]);
grid on;


soundsc(x3,10000);
pause(3);

%-- a=1
a=1;

x3 = exp(-(a^2+2)*t) .* cos(2*pi*inifreq*t);



figure;
plot(t,x3);
xlabel('Time(s)');
ylabel('Amplitude');
title('Sinusoid with Exponentially Decaying Envelope when a = 1');
xlim([0 3]);
ylim([-1 1]);
grid on;


soundsc(x3,10000);
pause(3);

%-- a=3
a=3;

x3 = exp(-(a^2+2)*t) .* cos(2*pi*inifreq*t);



figure;
plot(t,x3);
xlabel('Time(s)');
ylabel('Amplitude');
title('Sinusoid with Exponentially Decaying Envelope when a = 3');
xlim([0 3]);
ylim([-1 1]);
grid on;


soundsc(x3,10000);
pause(3);

%--------------------------------
```

Part 4: Beat Notes and Amplitude Modulation

```
%%
% *Part 4: Beat Notes and Amplitude Modulation*

Ts = 0.0001;
dur = 3.0;
t = linspace(0, dur, dur * 1/Ts);

%-- f1=10Hz, f2= 1KHz

f1 = 10;
f2 = 1000;

x4 = cos(2*pi*f1*t) .* cos(2*pi*f2*t);


figure;
plot(t,x4);
xlabel('Time(s)');
ylabel('Amplitude');
title('Beat Notes and Amplitude Modulation x4[n] f1=10Hz, f2= 1KHz');
xlim([0 3]);
ylim([-1 1]);
grid on;


soundsc(x4, 1/Ts);
pause(3);



%-- f1=5Hz, f2= 1KHz

f1_1 = 5;
f2 = 1000;

x4_1 = 0.5 * (cos(2*pi*(f1_1+f2)*t) + cos(2*pi*(f2-f1_1)*t));


figure;
plot(t,x4_1);
xlabel('Time(s)');
ylabel('Amplitude');
title('Beat Notes and Amplitude Modulation x4[n] f1=5Hz, f2= 1KHz');
xlim([0 3]);
ylim([-1 1]);
grid on;


soundsc(x4_1, 1/Ts);
pause(3);

%-- f1=15Hz, f2= 1KHz

f1_2 = 15;
f2 = 1000;
```

```
x4_2 = 0.5 * (cos(2*pi*(f1_2+f2)*t) + cos(2*pi*(f2-f1_2)*t));


figure;
plot(t,x4_2);
xlabel('Time(s)');
ylabel('Amplitude');
title('Beat Notes and Amplitude Modulation x4[n] f1=15Hz, f2= 1KHz');
xlim([0 3]);
ylim([-1 1]);
grid on;


soundsc(x4_2, 1/Ts);
pause(3);

%----------------------
```

Part 5: Chrip Signals and Frequency Modulation

```
%%
% *Part 5: Chrip Signals and Frequency Modulation*


Dur= 2.0;
Ts = 0.0001;
t = linspace(0, Dur, Dur * 10000);
phase = 0;
%--2500Hz to 500Hz
f0 = 2500;
f1 = 500;
u = (f1-f0)/(2*Dur);
x5 = cos(2*pi*u.*(t.^2)+2*pi*f0.*t+phase);



figure;
plot(t,x5);
xlabel('Time(s)');
ylabel('x5(t)');
title('Chrip Signal from 2500 Hz to 500 Hz');
xlim([0 2]);
ylim([-1 1]);
grid on;

soundsc(x5,10000);
pause(3);

%-- 500Hz to 2500Hz
f0 = 500;
f1 = 2500;
Dur= 2.0;
u = (f1-f0)/(2*Dur);

x5 = cos(2*pi*u.*(t.^2)+2*pi*f0.*t+phase);
```

```
figure;
plot(t,x5);
xlabel('Time(s)');
ylabel('x5(t)');
title('Chrip Signal from 500 Hz to 2500 Hz');
xlim([0 2]);
ylim([-1 1]);
grid on;

soundsc(x5,10000);
pause(3);

%-- 3000 to -2000
f0 = 3000;
f1 = -2000;
Dur= 3.0;
Ts = 0.001;
t = linspace(0, Dur, Dur * 10000 );
u = (f1-f0)/(2*Dur);

x5 = cos(2*pi*u.*(t.^2)+2*pi*f0.*t+phase);



figure;
plot(t,x5);
xlabel('Time(s)');
ylabel('x5(t)');
title('Chrip Signal from 3000 Hz to -2000 Hz');
xlim([0 2]);
ylim([-1 1]);
grid on;

soundsc(x5,10000);
pause(3);

%-----------------
```

Part 6: Composing Music

```
function [note] = notecreate(frqno, dur)
    % Generate a sine wave for the given frequency and duration
    note = sin(2*pi*[1:dur]/8192 * (440 * 2.^((frqno-1)/12)));
end



notename = {'A', 'A#', 'B', 'C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#'};

song = {'E', 'G', 'A', 'E', 'G', 'A', 'B', 'D', 'C', 'B', 'A', 'E', 'G', 'A', 'B',
'D', 'C', 'B', 'A', 'G', 'E', 'G', 'A', 'E'};


songidx = zeros(1, length(song));
for k1 = 1:length(song)
```

```
    idx = strcmp(song{k1}, notename);
    songidx(k1) = find(idx);
end


durations = [0.3, 0.3, 0.3, 0.2, 0.2, 0.2, 0.4, 0.4, 0.4, 0.3, 0.3, 0.3, ...
             0.5, 0.2, 0.2, 0.3, 0.3, 0.3, 0.4, 0.4, 0.4, 0.3, 0.3, 0.5];

dur = round(durations * 8192);


songnote = [];
for k1 = 1:length(songidx)
    note_wave = notecreate(songidx(k1), dur(k1));
    songnote = [songnote; [note_wave zeros(1, 50)]'];
end


soundsc(songnote, 8192);
```