Bilge Kaan Ateş

22102758

16.02.2024

**EEE321 SIGNALS AND SYSTEMS**

**LAB-1: The Musical Scale**

**1) INTRODUCTION:**

In this lab, I looked into sinusoidal impulses, musical notes, and sound creation. By guiding me through the development and evaluation of a range of signals with varying amplitude, frequency, or phase that are connected to the basic sinusoid, the assignment helped me comprehend the relationships between frequency, harmonics, and modulation effects on sound and signals. By utilizing Matlab to create sinusoidal waveforms and compose a simple tune, theoretical concepts are put into practice and audible musical emotions are produced.

**2) LAB:**

* Part 1:

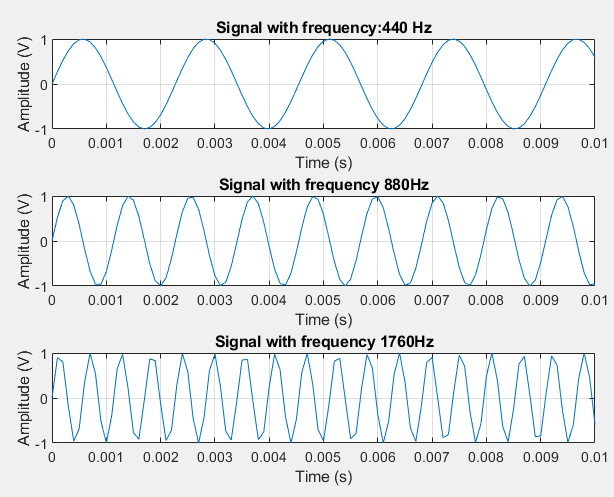


Fig. 1: Note A with different octaves

The sound of note A becomes thinner (high-pitched) as frequency increases. The frequency relationship between notes A (440 Hz), C sharp, and E was examined in this lab work as well. I experienced the harmonic link between E's 554×4 and C sharp's 440×5. Harmonic relation between these signals lead to the feeling of the satisfaction. Then, I create the major triad signal by combining these sinusoids as shown in Fig. 2. I understand the significance of harmonic relationships in music for creating consonant and aesthetically pleasing note combinations. Signals formulas are as follows:

, where

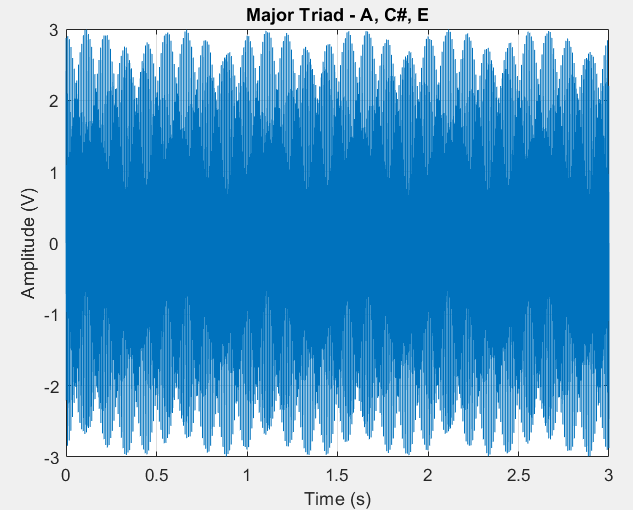


Fig. 2: Major Triad waveform plot

Relevant codes are provided below:

f\_0 = 440; % Given frequency of the signal

end\_time = 3; % time-lenth of the signal

f\_sample = 10000; % Sampling rate (1/T\_sample)

t = linspace(0, end\_time, end\_time\*f\_sample); %sampling array

x1 = sin(2\*pi\*f\_0\*t);

x\_12 = sin(2\*pi\*2\*f\_0\*t);

x\_13 = sin(2\*pi\*4\*f\_0\*t);

figure;

subplot(3,1,1);

plot(t, x1);

xlim([0,0.01]);

xlabel('Time (s)');

ylabel('Amplitude (V)');

title('Signal with frequency:440 Hz');

grid on;

subplot(3,1,2);

plot(t, x\_12);

xlim([0,0.01]);

xlabel('Time (s)');

ylabel('Amplitude (V)');

title('Signal with frequency 880Hz');

grid on;

subplot(3,1,3);

plot(t, x\_13);

xlim([0,0.01]);

xlabel('Time (s)');

ylabel('Amplitude (V)');

title('Signal with frequency 1760Hz');

grid on;

soundsc(x1, f\_sample);

pause(end\_time);

soundsc(x\_12, f\_sample);

pause(end\_time);

soundsc(x\_13, f\_sample);

pause(end\_time);

% Major triad formula

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

figure;

plot(t, s);

xlabel('Time (s)');

ylabel('Amplitude (V)');

title('Major Triad - A, C#, E');

grid on;

soundsc(s, f\_sample);

* Part 2:

φ in the signals corresponds to the phase shift for periodic signals. Its effect is shifting the signal in time domain. However, it neither affects the pitch of the signal nor volume.

(1)

Observing the formula (1) we can see that volume is affected by the amplitude (A) and the pitch is affected by the frequency (). Pitch of the sound depends on the periodic repetition of the signal.

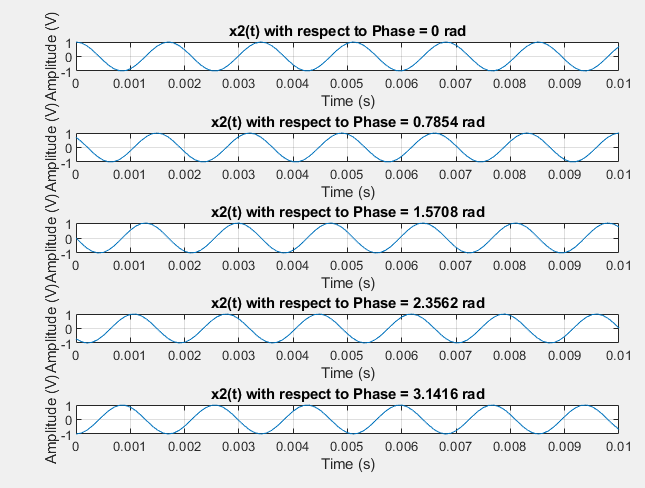


Fig. 3: Phase shifted signals

Observing the Fig. 3 shows us our judgments are correct about the phase.

f\_0 = 587;

duration = 3;

f\_sample = 1/0.0001;

%sampling array

t = linspace(0, duration, duration\*f\_sample);

% Sinusoidal signals with the given phases

phases = [0, pi/4, pi/2, 3\*pi/4, pi];

x2\_phasevar\_signals = zeros(length(phases), length(t));

figure;

for i = 1:length(phases)

x2\_phasevar\_signals(i, :) = cos(2\*pi\*f\_0\*t + phases(i));

subplot(length(phases), 1, i);

plot(t, x2\_phasevar\_signals(i, :));

xlim([0,0.01]);

xlabel('Time (s)');

ylabel('Amplitude (V)');

title(['x2(t) with respect to Phase = ', num2str(phases(i)), ' rad']);

grid on;

end

for i = 1:length(phases)

soundsc(x2\_phasevar\_signals(i, :), f\_sample);

pause(duration);

end

* Part 3:

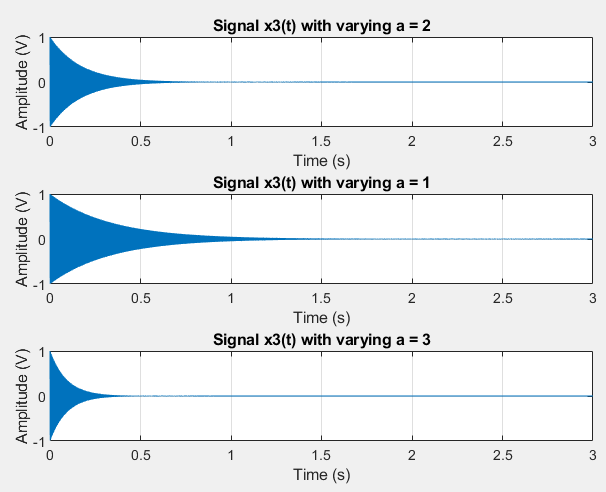


Fig. 4: Exponentially Damped Sinusoids

The exponential term in the signal introduces amplitude decay, which results in a diminishing sinusoidal waveform with respect to time. Therefore, 𝑥3(𝑡) is more likely to a piano sound while 𝑥1(𝑡) sounds more likely to flute note.

a\_values = [2, 1, 3]; % Array of exponent coefficients

f\_0 = 440;

duration = 3;

f\_sample = 10000;

% Generate time vector

t = linspace(0, duration, duration\*f\_sample);

% Compute x3 for each value of a

figure;

for i = 1:length(a\_values)

a = a\_values(i);

%Element-wise multiplication with arrays

x3 = exp(-(a^2 + 2) \* t) .\* cos(2\*pi\*f\_0\*t);

subplot(length(a\_values), 1, i);

plot(t, x3);

xlim([0,3]);

xlabel('Time (s)');

ylabel('Amplitude (V)');

title(['Signal x3(t) with varying a = ', num2str(a)]);

grid on;

soundsc(x3, f\_sample);

pause(duration);

end

* Part 4:

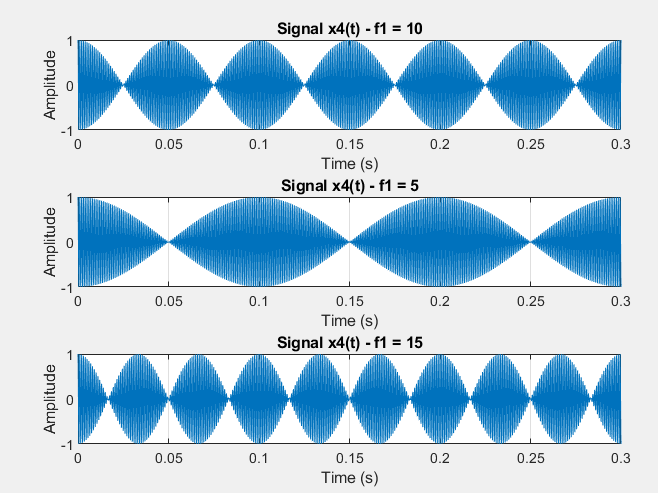


Fig. 5: AM modulated signals

In this part we create AM modulated signals with the following signal where and :

We can expand this signal with the half angle formulas as follows:

So we have:

Since is much smaller compare to , we obtained two cosine signal with very close frequencies. Hence, we observed the beating effect. This beating effect diminishes as becomes closer to and does not have disturbing effect when it is harmonic with the . Note that beating effect is clearer when

f1\_values = [10, 5, 15]; % Array of low-frequencies

f2 = 1000; % High-frequency signal

duration = 3;

f\_sample = 10000;

t = linspace(0, duration, duration\*f\_sample);

figure;

for i = 1:length(f1\_values)

f1 = f1\_values(i);

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

% Plot x4

subplot(length(f1\_values), 1, i);

plot(t, x4);

xlim([0,0.3]);

xlabel('Time (s)');

ylabel('Amplitude');

title(['Signal x4(t) - f1 = ', num2str(f1)]);

grid on;

soundsc(x4, f\_sample);

pause(duration);

end

* Part 5:

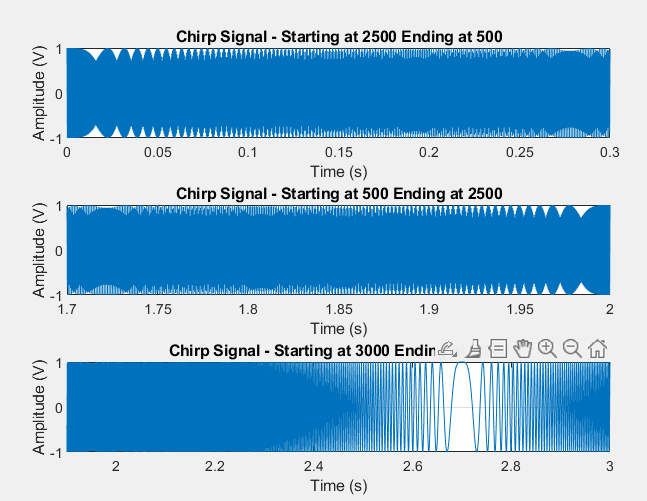


Fig. 6: FM modulated signal

The idea of time-varying frequency is signals, paying special emphasis to chirp signals, which are sinusoids with a linear time-varying frequency. Frequency modulation (FM) is exemplified by chirp signals, the instantaneous frequency of which is found by the time derivative of the angle function. The signal as follows in our case:

We have the above signal, which is frequency modulated with respect to time. For the first case, where frequency goes from 2500 Hz to 500 Hz, the sound is get thicker as time passes. On the other hand, in second case, where frequency goes from 500 Hz to 2500 Hz, the sound is get thinner as time passes. The coefficient determines the chirping speed. When it is high, signal chirps faster while signal chirps slower when μ is smaller.

For the last signal, where frequency goes from 3000Hz to -2000Hz, it first chirps down and when reach the 0Hz frequency it starts to chirps up.

duration = [2, 2, 3];

f\_sample = 10000;

starting\_frequency = [2500, 500, 3000];

ending\_frequency = [500, 2500, -2000];

x\_start = [0, 1.7, 1.9];

x\_end = [0.3, 2, 3];

% Calculate the mu array based on the duration and frequency change

mu = (ending\_frequency - starting\_frequency) ./ (duration.^2);

figure;

for i= 1:3

t = linspace(0, duration(i), duration(i)\*f\_sample);

x5 = cos(2\*pi\*mu(i)\*t.^2 + 2\*pi\*starting\_frequency(i)\*t);

subplot(length(duration), 1, i);

plot(t, x5);

xlim([x\_start(i),x\_end(i)])

ylim([-1,1])

xlabel('Time (s)');

ylabel('Amplitude (V)');

title(['Chirp Signal - Starting at ', num2str(starting\_frequency(i)),' Ending at ',num2str(ending\_frequency(i))]);

grid on;

soundsc(x5, f\_sample);

pause(2)

end

* Part 6:

I composed the “Deniz Üstü Köpürür” song. Since I need the lower and upper octave notes, I also modify the the notename array and notecreate function.

function [note] = notecreate(frq\_no, dur)

note = sin(2\*pi\*[1:dur]/8192\*(440\*2.^((frq\_no-3)/12)));

end

notename = {'G1','G1#','A', 'A#', 'B', 'C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#','A2','A2#','B2','C2'};

song = ["C","C","D","D","D","D","C","C","F","F","G","G","A2","A2","A2","A2","G","G","C2","C2","B2","B2","A2","A2","G","G","F","F","G","G","F","E","D","D","D","E","C", "C","C","C","C", "C","C","C","A2","B2","B2","A2","A2","G","G","F","G","G","F","E","D","D","D","E","C","C","C","D","D","D","D","D","D"];

for k1 = 1:length(song)

idx = strcmp(song{k1}, notename);

songidx(k1) = find(idx);

end

dur = 0.3\*8192;

songnote = [ ];

for k1 = 1:length(songidx)

songnote = [songnote; [notecreate(songidx(k1),dur) zeros(1,75)]'];

end

soundsc(songnote, 8192)