**EE430 Lab 1 Writeup**

Section I – Narrative

Problem 1)

1. Sampling frequency fs was set to 44000, based on CD quality specifications. A signal frequency f was selected and set to 400, as that frequency lies in a comfortable listening region for the human ear. A phase shift, p, was also defined, being set to 0 for this exercise. These values were then used to generate a sinusoid based on the formula *s(t) = cos(2πft+p).*
2. Using the code for the section above, a signal of frequency 1 Hz was then plotted for 3 seconds. This plot can be viewed in Section II, figure 1.
3. The signal was played using the MATLAB command *sound*(x,fs). This was compared against an internet recording of the same signal to verify that the function operated as expected. It was noticed that changing the value for the phase shift ‘p’ caused no audible difference in the sound produced.

Problem 2)

1. This function was written. The source code can be found in Section III – source code.
2. The function synthesizeFromSpectrum was used with the derived values from the first portion, and then stored into a variable ‘s’. This variable was used to produce a tone using the sound function within MATLAB, the tone of which ended up being identical to the first signal, verifying the function.
3. Three chords were generated based on frequency values from Wikipedia, all with amplitude 0.5. These values were then played using the sound function within MATLAB. The sounds produced were compared to piano chord recordings online in order to confirm proper function.
4. The chord C-major was selected for this question, and was offset by pi/3 radians. There was no perceived difference in the way this offset affected the sound of the signal. The signals appeared to differ only in where the peaks were located along the x-axis. Both these graphs are located in Section II – Figures, figure 2 & 3.

Problem 3)

1. This function was written. The source code can be found in Section III – source code.
2. The plotted spectrums can be found in Section II – Figures, Figure 4.

Problem 4)

1. This function was written. The source code can be found in Section III – source code.
2. This function was written. The source code can be found in Section III – source code.
3. This function was written. The source code can be found in Section III – source code.

Problem 5)

1. A spectrum was generated using random values, utilizing the ‘rand’ function within MATLAB, in combination with array combination methods in order to generate both positive values and negative values that were equal in magnitude. The spectrum was then plotted using the plotFromSpectrum function created earlier, and listened to using the ‘sound’ function.
2. A lowpass cutoff frequency fCoLow was specified as 250 Hz. The signal was then filtered using the idealLowpass function, and listened to.
3. A highpass cutoff frequency fCoHigh was specified as 1000 Hz. The signal was then filtered using the idealHighpass function and listened to.
4. The earlier two cutoff frequencies were used with the idealBandpass function in order to accomplish essentially the same thing the last two steps did, albeit at the same time.

Problem 6)

1. The Fourier Coefficients a­k can be expressed by the following:

Section II – Figures

Figure 1: First 3 Seconds of a Plotted Signal of Frequency 1 Hz

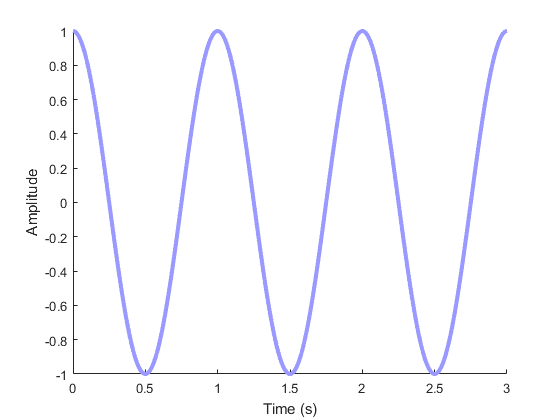


Figure 2: First 0.1 Seconds of a Plotted C-Chord

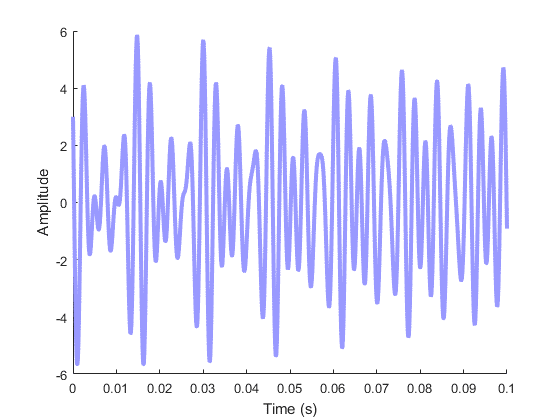
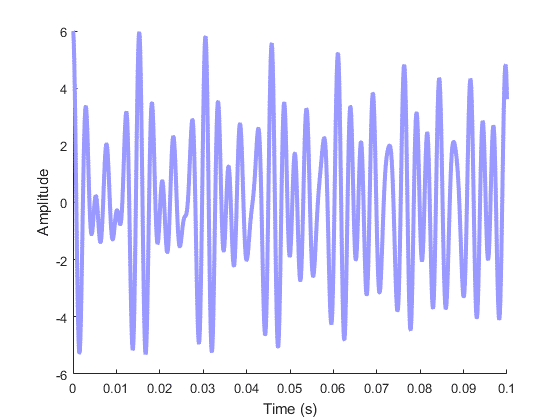


Figure 3: First 0.1 Seconds of a Plotted C-Chord with a pi/3 offset



Section III – Source Code

Filename synthesizeFromSpectrum.m

function x = synthesizeFromSpectrum(specFreq,specAmp,fs,tstart,tend);

% Synthesize From Spectrum

% Attempts to represent a spectra based on the following:

%

% x = synthesizeFromSpectrum(specFreq,specAmp,fs,tstart,tend);

%

% specFreq = An array of Omega Values

% specAmp = An array of Amplitude values of form A\*exp(xj)

% fs = Sampling Frequency

% tstart = Defines start time for sampling frequency

% tend = Defines end time for sampling frequency

%

% Function Parameters:

% s(t) = specAmp\*exp(j\*2\*pi\*SpecFreq\*t)

%

% See also: plotSpectrum,

%

% Ver. 1.0 by BR

T = (1/fs); % Sampling Period

n = tstart\*fs:tend\*fs; % Sampling Instances

t = n\*T; % Total Number of Samples

a = 0;

for k = 1:length(specAmp);

a = a + specAmp(k)\*exp(j\*2\*pi\*specFreq(k)\*t);

end;

x = a;

Filename plotSpectrum.m

function plotSpectrum(specFreq,specAmp);

% Plot Spectrum

% Plots a spectrum based on the following:

%

% x = plotSpectrum(specFreq,specAmp);

%

% specFreq = An array of Omega Values

% specAmp = An array of Amplitude values of form A\*exp(xj)

%

% See also: synthesizeFromSpectrum,

%

% Ver. 1.0 by BR

% Grid of 2 x 1

figure('Color',[1 1 1]);

subplot(2,1,1);

h = stem(specAmp,specFreq);

box off;

grid on;

hold on;

set(h,'Linewidth',3);

set(h,'Color',[0.4 0.4 1]);

xlabel('Frequency (Hz)');

ylabel('Amplitude');

axis tight;

subplot(2,1,2);

h = stem(specAmp,specFreq);

box off;

grid on;

hold on;

set(h,'Linewidth',3);

set(h,'Color',[0.4 0.4 1]);

xlabel('Frequency (Hz)');

ylabel('Amplitude');

axis tight;

Filename idealLowPass.m

function [x,y] = idealLowpass(specFreqIn,specAmpIn,fco);

% Ideal Lowpass

% Cuts any incoming signals below the specified frequency

%

% specFreqIn = An array of Omega Values

% specAmpIn = An array of Amplitude values of form A\*exp(xj)

% fco = Cutoff Frequency

%

% See also: idealHighPass, idealBandPass

%

% Ver. 1.0 by BR

for k = 1:length(specAmpIn);

if (abs(specFreqIn(k)) <= fco)

specAmpIn(k) = 0;

end;

end;

x = specFreqIn;

y = specAmpIn;

Filename idealHighPass.m

function [x,y] = idealHighpass(specFreqIn,specAmpIn,fco);

% Ideal Lowpass

% Cuts any incoming signals above the specified frequency

%

% specFreqIn = An array of Omega Values

% specAmpIn = An array of Amplitude values of form A\*exp(xj)

% fco = Cutoff Frequency

%

% See also: idealHighPass, idealBandPass

%

% Ver. 1.0 by BR

for k = 1:length(specAmpIn);

if (abs(specFreqIn(k)) >= fco)

specAmpIn(k) = 0;

end;

end;

x = specFreqIn;

y = specAmpIn;

Filename idealBandPass.m

function [x,y] = idealBandpass(specFreqIn,specAmpIn,fco1, fco2);

% Ideal Bandpass

% Cuts any incoming signals between the specified frequencies

%

% specFreqIn = An array of Omega Values

% specAmpIn = An array of Amplitude values of form A\*exp(xj)

% fco1 = Cutoff Frequency (low end)

% fco2 = Cutoff Frequency (high end)

%

% See also: idealHighPass, idealBandPass

%

% Ver. 1.0 by BR

for k = 1:length(specAmpIn);

if (abs(specFreqIn(k)) >= fco2)

specAmpIn(k) = 0;

elseif (abs(specFreqIn(k)) <= fco1)

specAmpIn(k) = 0;

end;

end;

x = specFreqIn;

y = specAmpIn;

Filename Lab1.m

%% Lab 1 - Synthesis of Signals from Discrete Spectrum

% By Brandon Rolston

% Hot Boilerplate Action

close all;

clc;

clear all;

%% 1) Synthesize Single Sinusoid

% Define Variables

fs = 44100; % Sampling Frequency

T = (1/fs); % Sampling Period

f = 1; % Signal Frequency

p = 0; % Phase Shift

n = 0:3\*fs; % Samples

t = n\*T; % Sampling Instances

% Define Functions

x = cos(2\*pi\*f\*t+p); % Signal

% Plot The Signal For 3 Seconds

figure('Color',[1 1 1]); % Figure with White Background

h = plot(t,x); % Plot the data

box off; % No love for the box edges

set(h,'Linewidth',3);

set(h,'Color', [0.6 0.6 1]);

hold on;

box off;

xlabel('Time (s)');

ylabel('Amplitude');

% Play The Tone (Probably Want to Comment This Out Later)

fd = 2.5\*fs; % When sampling frequency is forced higher, pitch/play rate increases

% When it is forced lower, the pitch/play rate decreases.

%sound(x,fd); %<---- UNCOMMENT TO HEAR SIGNAL

%% 2) Synthesize from Spectra

specFreq = [261.626, -261.626, 329.628, -329.628, 391.995, -391.995];

specAmp = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5];

startTime = 0;

stopTime = 3;

s = synthesizeFromSpectrum(specFreq,specAmp,fs,startTime,stopTime);

%sound(s,fs);

fs = 44100; % Sampling Frequency

n = 0:0.1\*fs; % Samples

t = n\*T; % Sampling Instances

x1 = cos(2\*pi\*261.626\*t)+cos(2\*pi\*-261.626\*t)+cos(2\*pi\*329.628\*t)+cos(2\*pi\*-329.628\*t)+cos(2\*pi\*391.995\*t)+cos(2\*pi\*-391.995\*t); % Signal

x2 = cos(2\*pi\*261.626\*t+(pi/3))+cos(2\*pi\*-261.626\*t-(pi/3))+cos(2\*pi\*329.628\*t+(pi/3))+cos(2\*pi\*-329.628\*t-(pi/3))+cos(2\*pi\*391.995\*t+(pi/3))+cos(2\*pi\*-391.995\*t-(pi/3)); % Signal (offset)

% Plot The Signal For 0.1 Seconds

figure('Color',[1 1 1]); % Figure with White Background

h = plot(t,x1); % Plot the data

box off; % No love for the box edges

set(h,'Linewidth',3);

set(h,'Color', [0.6 0.6 1]);

hold on;

box off;

xlabel('Time (s)');

ylabel('Amplitude');

figure('Color',[1 1 1]); % Figure with White Background

h = plot(t,x2); % Plot the data

box off; % No love for the box edges

set(h,'Linewidth',3);

set(h,'Color', [0.6 0.6 1]);

hold on;

box off;

xlabel('Time (s)');

ylabel('Amplitude');

%% 3) Plotting Spectrum

Cmajor = [261.626, -261.626, 329.628, -329.628, 391.995, -391.995];

Cminor = [261.626, -261.626, 311.127, -311.127, 391.995, -391.995];

Fmajor = [349.228, -349.228, 440.000, -440.000, 523.251, -523.251];

specAmp = [0.5\*exp(j\*0.3),0.5\*exp(-j\*0.3),0.5\*exp(j\*0.3),0.5\*exp(-j\*0.3),0.5\*exp(j\*0.3),0.5\*exp(-j\*0.3)];

offset = real(log(specAmp))

plotSpectrum(specAmp,Cmajor);

plotSpectrum(specAmp,Cminor);

plotSpectrum(specAmp,Fmajor);

%% 4) Filtering

specFreqIn = [261.626, -261.626, 329.628, -329.628, 391.995, -391.995];

specAmpIn = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5];

fLow = 330;

% Lowpass Test

[x,y] = idealLowpass(specFreqIn,specAmpIn,fLow)

% Highpass Test

specFreqIn = [261.626, -261.626, 329.628, -329.628, 391.995, -391.995];

specAmpIn = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5];

fHigh = 320;

[x,y] = idealHighpass(specFreqIn,specAmpIn,fHigh)

% Bandpass Test

specFreqIn = [261.626, -261.626, 329.628, -329.628, 391.995, -391.995];

specAmpIn = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5];

fLow = 262;

fHigh = 330;

[x,y] = idealBandpass(specFreqIn,specAmpIn,fLow,fHigh)

%% 5) Random Signals

%Generate Random Frequency

bottomFreq = 50;

topFreq = 5000;

randFreq = (topFreq-bottomFreq).\*rand(500,1) + bottomFreq;

randFreq2 = -randFreq;

specFreq = [randFreq;randFreq2];

%Generate Random Amplitude

randAmp = rand(500,1);

randAmp2 = -randAmp;

specAmp = [randAmp;randAmp2];

%Generate Random Phase

randPhase = rand(500,1).\*j;

randPhase2 = - randPhase;

allPhase = [randPhase;randPhase2];

%Combine Amplitude and Phase Appropriately

specAmp = specAmp.\*allPhase;

% Specify Variables as needed for synthesizeFromSpectrum

fs = 44100;

startTime = 0;

stopTime = 3;

% Plot the generated Values

s = synthesizeFromSpectrum(specFreq,specAmp,fs,startTime,stopTime);

%sound(s,fs); %Uncomment to hear nightmaresounds

% Filter Signals

fCoLow = 1000;

fCoHigh = 3000;

specAmpLow = specAmp;

specAmpHigh = specAmp;

specAmpBand = specAmp;

% Highpass

[x,y] = idealLowpass(specFreq,specAmpLow,fCoLow);

plotSpectrum(y,x);

% Lowhpass

[x,y] = idealHighpass(specFreq,specAmpHigh,fCoHigh);

plotSpectrum(y,x);

% Bandpass

[x,y] = idealBandpass(specFreq,specAmpBand,fCoLow,fCoHigh);

plotSpectrum(y,x);

%% 6) Fourier Coefficients and Spectrum