Section III – Source Code

Problem 1) Computing DFT

%% Part 1: Computing DFT

%% i) Generate 1 Second of an Audible Signal

% Define Variables

fs = 5000; % Sampling Frequency

T = (1/fs); % Sampling Period

f = 432; % Signal Frequency (Holistic!)

n = 0:fs; % Samples

t = n\*T; % Sampling Instances

% Define Functions

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

%% ii) Compute the DFT of the Signal

y = zeros(size(x));

index = [1:(fs+1)];

n = 1;

%%

for q = 1:fs

for k=1:fs

y(n) = y(n) + x(k)\*exp(-2\*pi\*j\*k\*n/(fs));

end

n = n+1;

end;

%% Here's the graph for 2 (Power Spectrum)

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

h = plot(index,abs(y.^2)); % Plot the data

box off; % No love for the box edges

set(h,'Linewidth',3); % I like my lines THICK

set(h,'Color', [0.6 0.6 1]); % Nice Blue Lines

hold on; % Hold that graph

box off; % No, no box!

xlabel('DFT Index');

ylabel('Power');

%% iii) Plot Squares of the Absolute Value of Coefficients v. Physical Frequency

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

h = plot(index,abs(y.^2)); % Plot the data

axis([0, (fs/2), 0, 7E6]);

box off;

set(h,'Linewidth',3); % I like my lines THICK

set(h,'Color', [0.6 0.6 1]); % Nice Blue Lines

hold on; % Hold that graph

box off;

xlabel('Physical Frequency (Hz)');

ylabel('Power');

%% iv) Make 4 sinusoids, sum 'em, plot 4 sinusoids.

fs = 5000;

dur = 1;

x1 = generate\_wave(1,500,fs,dur);

x2 = generate\_wave(0.5,722,fs,dur);

x3 = generate\_wave(0.2,351,fs,dur);

x4 = generate\_wave(0.7,1023,fs,dur);

x5 = x1+x2+x3+x4;

y = fft(x5);

plot\_power\_spectrum(y,fs);

xlim([0 1.2\*1023]);

%% v) Noisy Power Spectrums

x = generate\_wave(1,432,5000,1);

s = 20;

% Notes for various values 's':

% Audible: Up to about 0.8

% Sinusoid is no longer Heard at 0.8-4

% Sinsuoid Becomes hard to discern at 4

% Power Spectrum Becomes a Mess at around 20

xnoisy = x + s\*randn(size(x));

y = direct\_fourier(xnoisy,fs);

plot\_power\_spectrum(y,5000);

xlim([0 1.2\*432]);

%% Signal to noise ratio

Px = sum(x.^2);

Pn = sum(xnoisy.^2);

SN = Px/Pn

sound(xnoisy, 5000);

% Notes about SNR:

% Audible at SNR > 0.4. (Edge case occurs at s=0.8)

% Sound becomes lost at 0.05 < SNR < 0.4 .

% The sinusoid becomes lost when plotted against time at 0.02 < SNR < 0.05.

% The power spectrum is no longer discernable at SNR < 0.02

%% Plot

index = [1:(fs+1)];

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

h = plot(index,xnoisy); % Plot the data

xlim([0 4\*432]);

box off;

set(h,'Linewidth',3);

set(h,'Color', [0.6 0.6 1]); % Nice Blue Lines

hold on; % Hold that graph

box off;

xlabel('Sample Number');

ylabel('Amplitude');

Problem 2) Determining an Unknown Frequency

%% PART 2) UNKNOWN FREQUENCY

%% i) Hidden sinusoids

[noisy1,fs1] = audioread('lab3prob2sound1.wav');

[noisy2,fs2] = audioread('lab3prob2sound2.wav');

[noisy3,fs3] = audioread('lab3prob2sound3.wav');

[noisy4,fs4] = audioread('lab3prob2sound4.wav');

%%

y1 = fft(noisy1);

y2 = fft(noisy2);

y3 = fft(noisy3);

y4 = fft(noisy4);

%%

plot\_power\_spectrum(y1,2\*fs1);

plot\_power\_spectrum(y2,2\*fs2);

plot\_power\_spectrum(y3,2\*fs3);

plot\_power\_spectrum(y4,2\*fs4);

%% 2&4 are vague, let's try filtering the signal and doing it one more time

% Makin' a FIR Filter

length = 500;

h1 = ones(1,length)/length;

% Create 'denoised' signals

ydesnoised1=conv(noisy2,h1,'same');

ydesnoised2=conv(noisy4,h1,'same');

% Get them transformed

y5 = fft(ydesnoised1);

y6 = fft(ydesnoised2);

% Try Plotting the power spectrum again

plot\_power\_spectrum(ydesnoised1,2\*fs2);

plot\_power\_spectrum(ydesnoised2,2\*fs4);

%There appears to be nothing in these signals, as after filtering, they

%still look like nonsense.

%% Guitar Note

[guitar,fsg] = audioread('guitar-note.wav');

yg = fft(guitar);

index = [1:118272];

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

h = plot(index,abs(yg.^2)); % Plot the data

xlim([0 fsg/2]); % Restrict the x-axis

box off;

set(h,'Linewidth',3);

set(h,'Color', [0.6 0.6 1]); % Nice Blue Lines

hold on; % Hold that graph

box off;

xlabel('Physical Frequency (Hz)');

ylabel('Power');

Problem 3: FIR Filter Design

%% Part 3: FIR Design

%% i) Using fir1

fs = 8000;

%% Lowpass

h = fir1(20,pi/8,'low');

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

freqz(h);

h = fir1(200,pi/8,'low');

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

freqz(h);

%% Highpass

h = fir1(20,pi/8,'high');

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

freqz(h);

h = fir1(200,pi/8,'high');

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

freqz(h);

%% Bandpass (500-700)

h = fir1(20,[pi/8 pi/5.714]);

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

freqz(h);

h = fir1(200,[pi/8 pi/5.714]);

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

freqz(h);

%% Generate Signal

dur = 1;

fs = 8000;

x1 = generate\_wave(1,157,fs,dur);

x2 = generate\_wave(1,637,fs,dur);

x3 = generate\_wave(1,853,fs,dur);

x4 = x1+x2+x3;

y = fft(x4);

plot\_power\_spectrum(y,fs);

xlim([0 1.2\*853]);

%% Filter Signal

% Low

h = fir1(200,pi/32,'low');

x5 = filter(h,1,x4);

y2 = fft(x5);

plot\_power\_spectrum(y2,fs);

xlim([0 1.2\*853]);

%%

%High

h1 = fir1(200,(pi/32),'high');

x6 = filter(h1,1,x4);

y3 = fft(x6);

plot\_power\_spectrum(y3,fs);

xlim([0 1.2\*853]);

%%

% Band

h2 = fir1(200,[pi/32 pi/13.5]);

x7 = filter(h2,1,x4);

y4 = fft(x7);

plot\_power\_spectrum(y4,fs);

xlim([0 1.2\*853]);

%% Random Noise

noise=randn(fs,1);

passnoise = filter(h2,1,noise);

y4 = fft(passnoise);

plot\_power\_spectrum(y4,(fs-1));

sound(passnoise, fs);

Problem 4: Denoising Morse Code

[morse,fs] = audioread('noisymorsecode.wav');

morsepower = fft(morse);

index = [1:43221];

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

h = plot(index,abs(morsepower.^2)); % Plot the data

xlim([0 fs/2]); % Restrict the x-axis

box off;

set(h,'Linewidth',3);

set(h,'Color', [0.6 0.6 1]); % Nice Blue Lines

hold on; % Hold that graph

box off;

xlabel('Physical Frequency (Hz)');

ylabel('Power');

%% Band for 2701-ish

h = fir1(5,[(3\*pi/19.9) (3\*pi/18.95)]);

x = filter(h,1,morse);

y = fft(x);

plot\_power\_spectrum(y,43220);

xlim([0 fs/2]);

%% Plot signal against time

n = (0:43220); % Samples

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

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

box off;

set(h,'Linewidth',3);

set(h,'Color', [0.6 0.6 1]); % Nice Blue Lines

hold on; % Hold that graph

box off;

xlabel('Time');

ylabel('Amplitude');

sound(x, fs);

Problem 5: DTMF Tone Generation

%This looks boring, because most of the practical parts are stuck in

%functions.

x = dialSequence('65456665556666545666655654',0.2,0.1,8000);

soundsc(x,8000)

spectrogram(x,blackman(8000\*0.02),0, [] , 8000, 'yaxis')

Problem 6: DTMF Decoding

%% Part 6

%Load in the audio files

[x1,fs1] = audioread('ee430lab3fixedDTMF-1.wav');

[x2,fs2] = audioread('ee430lab3fixedDTMF-2.wav');

[x3,fs3] = audioread('ee430lab3fixedDTMF-3.wav');

[x4,fs4] = audioread('ee430lab3fixedDTMF-4.wav');

[x5,fs5] = audioread('ee430lab3fixedDTMF-5.wav');

[x6,fs6] = audioread('ee430lab3fixedDTMF-6.wav');

% Feed audio files into the DTMF decoder function

[digit,starttime,endtime] = decodeDTMF(x1,fs1);

fprintf('=====REPORTING DTMF DATA FOR FILE 1=====\nDigit : %s\nStart Time : %f Seconds\nEnd Time : %f Seconds\n',digit, starttime,endtime)

[digit,starttime,endtime] = decodeDTMF(x2,fs2);

fprintf('=====REPORTING DTMF DATA FOR FILE 2=====\nDigit : %s\nStart Time : %f Seconds\nEnd Time : %f Seconds\n',digit, starttime,endtime)

[digit,starttime,endtime] = decodeDTMF(x3,fs3);

fprintf('=====REPORTING DTMF DATA FOR FILE 3=====\nDigit : %s\nStart Time : %f Seconds\nEnd Time : %f Seconds\n',digit, starttime,endtime)

[digit,starttime,endtime] = decodeDTMF(x4,fs4);

fprintf('=====REPORTING DTMF DATA FOR FILE 4=====\nDigit : %s\nStart Time : %f Seconds\nEnd Time : %f Seconds\n',digit, starttime,endtime)

[digit,starttime,endtime] = decodeDTMF(x5,fs5);

fprintf('=====REPORTING DTMF DATA FOR FILE 5=====\nDigit : %s\nStart Time : %f Seconds\nEnd Time : %f Seconds\n',digit, starttime,endtime)

[digit,starttime,endtime] = decodeDTMF(x6,fs6);

fprintf('=====REPORTING DTMF DATA FOR FILE 6=====\nDigit : %s\nStart Time : %f Seconds\nEnd Time : %f Seconds\n',digit, starttime,endtime)