EE 427 Lab 1

Table of Contents

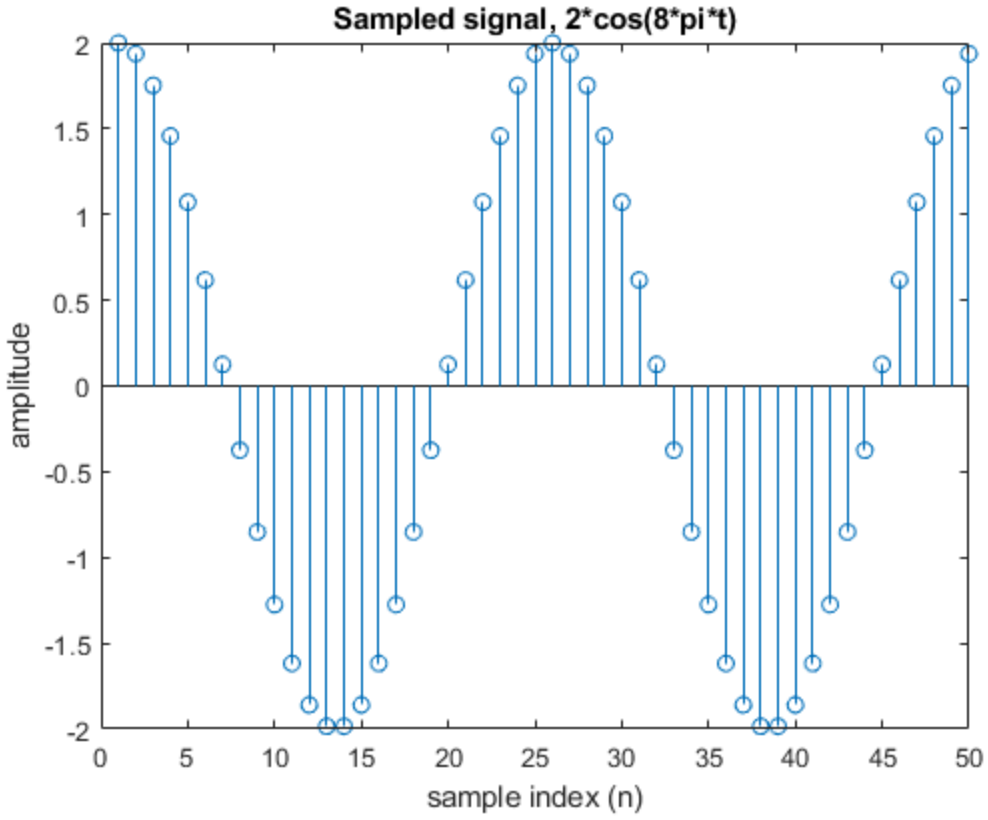
Part A: 1
Part B: 5
Part C 9

Name: Alexander Bolton Chris Fingers Date: 09/22/2019

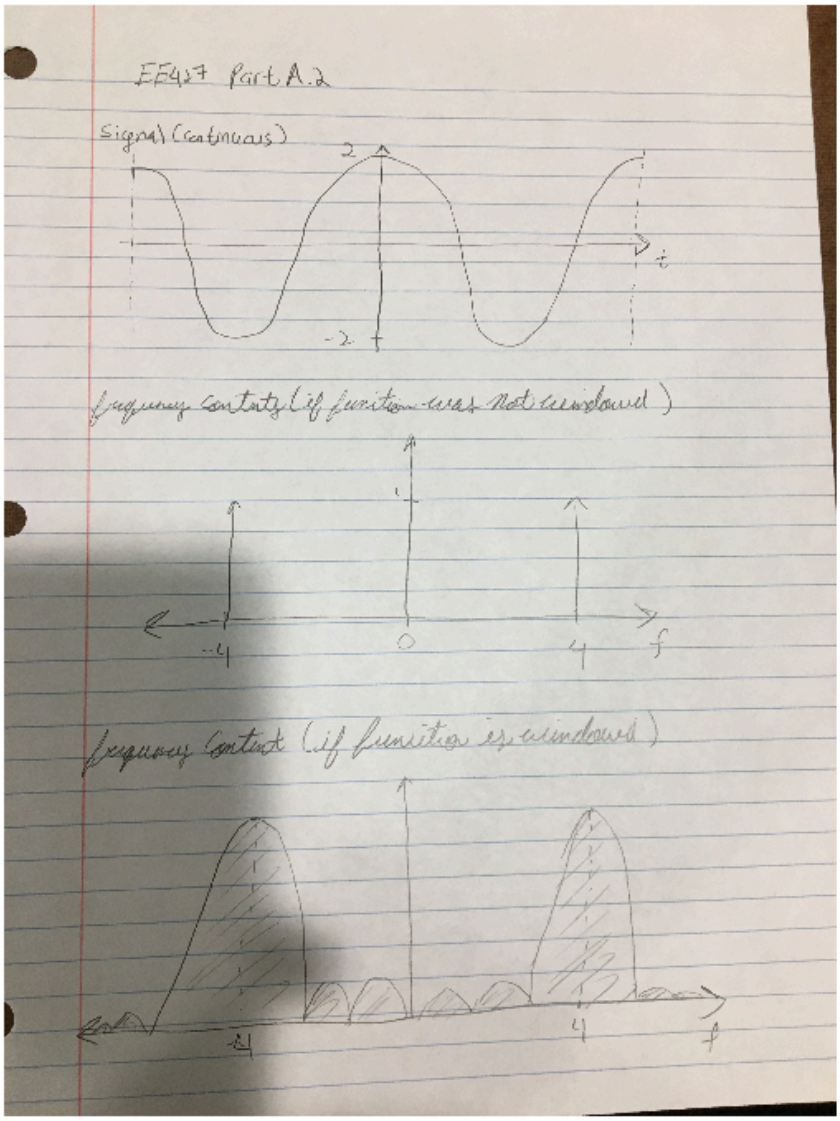
Part A:

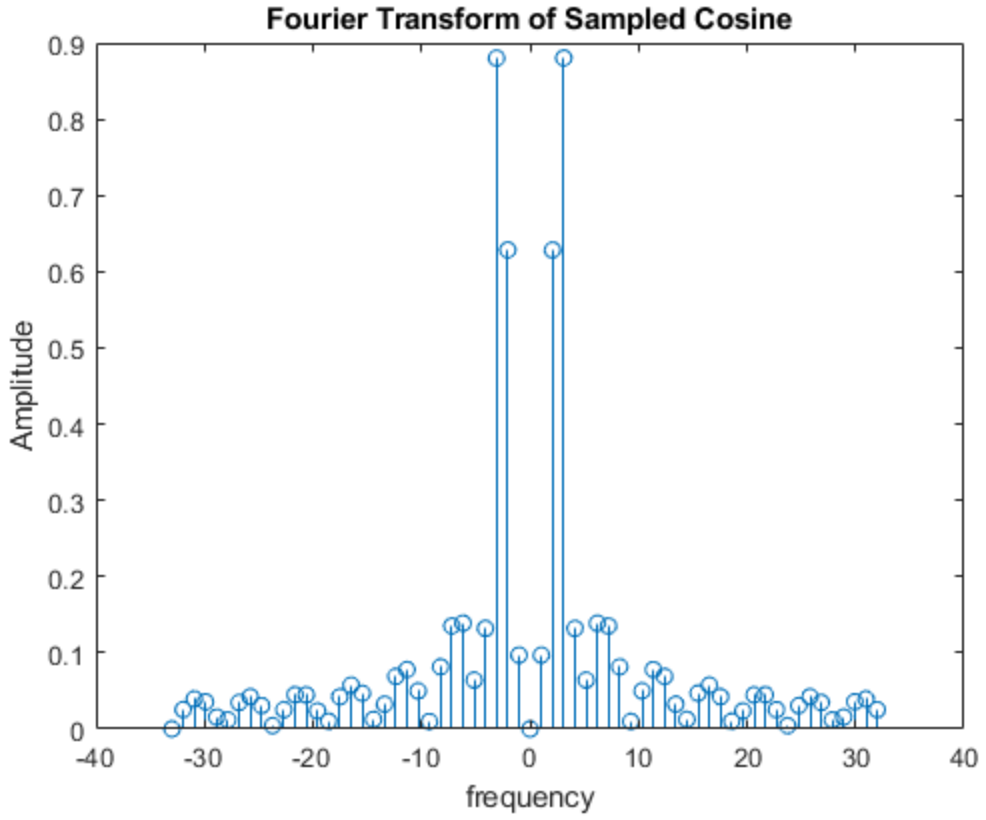
```
% Part 1:

A = 2;
f = 4;
n = -25:24;
fs = 100;

x = A*cos(2*pi*(f/fs)*n);
figure();
stem(x);
title('Sampled signal, 2*cos(8*pi*t)');
ylabel('amplitude');
xlabel('sample index (n)');

% Part 2:
pic = imread('EE427Project1A2.jpeg');
pic = imrotate(pic,-90);
figure();
imshow(pic);
title('Work done by hand');

% Part 3:
L = length(x);
FFTLength = 2^nextpow2(L);
xPrime = fft(x,FFTLength)/L;
xPrime = fftshift(xPrime);

figure();
stem(linspace(-33, 32, length(xPrime)),abs(xPrime));
title('Fourier Transform of Sampled Cosine');
xlabel('frequency');
ylabel('Amplitude');

% Part 4:
figure();
XPSD = pwelch(x);
plot(XPSD);
title('Power Spectral Density of Sampled Cosine');
```

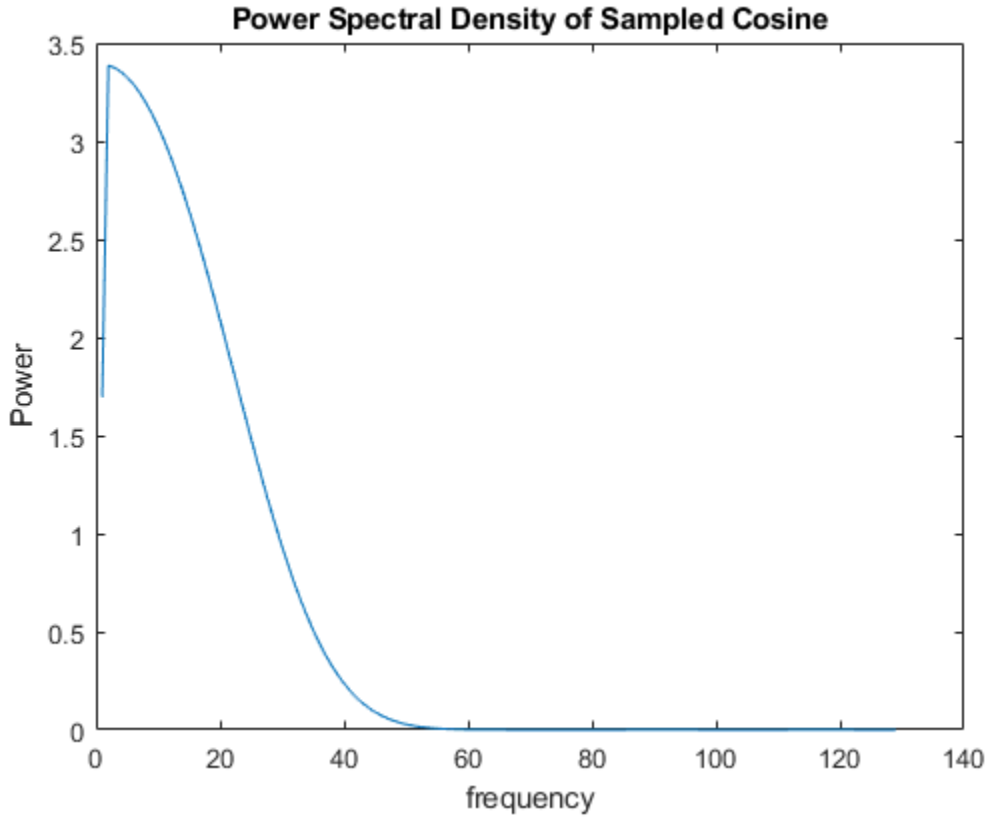
```
xlabel('frequency');
ylabel('Power');

% Part 5:
% The FFT shows the magnitude at which each frequency is present in
 the
% signal being analyzed. The FFT function creates a double-sided
 spectrum
% of the magnitude of each frequency. The PSD, however, shows the
 power
% each frequency contributes to the signal. The pwelch function
 creates a
% single-sided distribution of the power.
```



Work done by hand




Fourier Transform of Sampled Cosine
Amplitude
frequency
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-40
-30
-20
-10
0
10
20
30
40


Power Spectral Density of Sampled Cosine
Power
frequency
3.5
3
2.5
2
1.5
1
0.5
0
0
20
40
60
80
100
120
140

Part B:

POneWOne= audiorecorder(8000,16,1); POneWTwo= audiorecorder(8000,16,1); PTwoWOne= audiorecorder(8000,16,1); PTwoWTwo= audiorecorder(8000,16,1);

speaking part disp('start speaking'); recordblocking(POneWOne,2); % first person speaking first word disp('end'); pause(1);

disp('start speaking'); recordblocking(POneWTwo,2); % first person speaking second word disp('end'); pause(1);

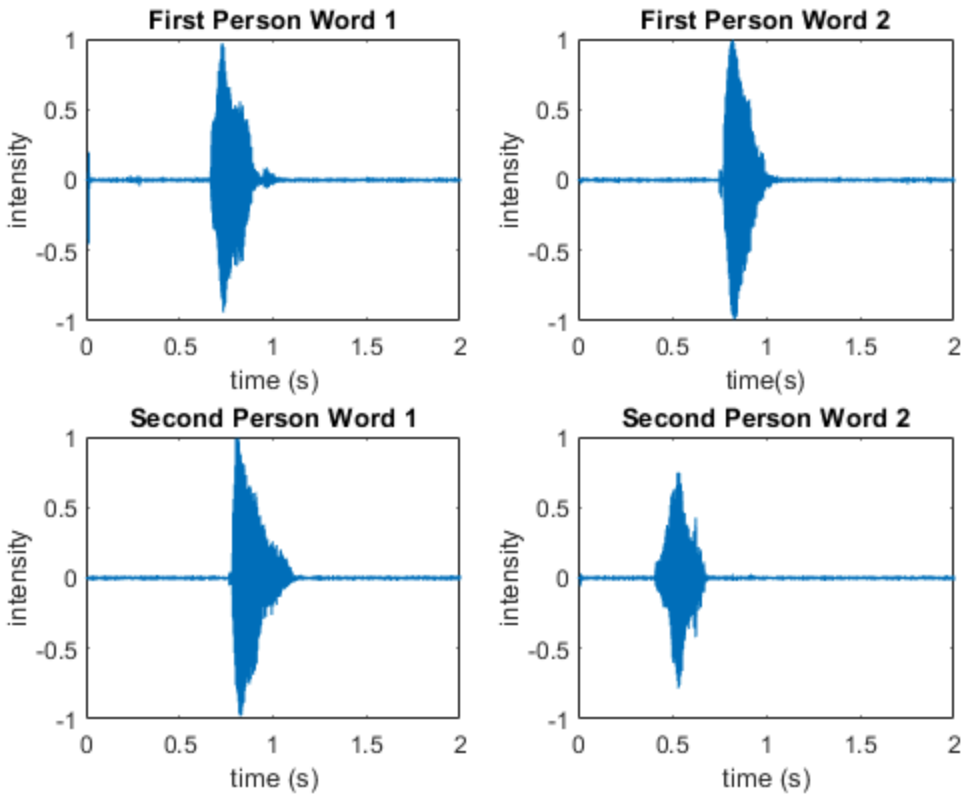
disp('start speaking'); recordblocking(PTwoWOne,2); % second person speaking first word disp('end'); pause(1);

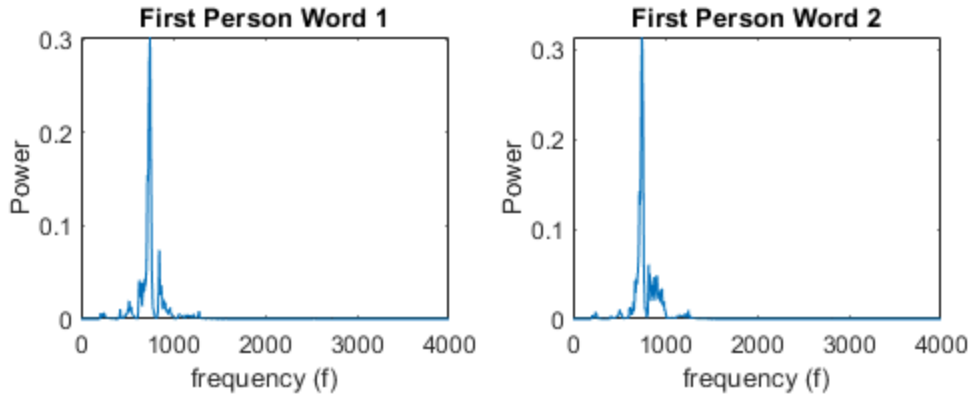
disp('start speaking'); recordblocking(PTwoWTwo,2); % second person speaking second word disp('end');

```
% This part was commented out for the ease of publishing, since we
 didn't
% know when to start and stop speaking for the words, we pre-recorded
 them
% before publishing. The code was, however, used to capture each
 recording

wordOne = getaudiodata(POneWOne);
wordTwo = getaudiodata(POneWTwo);
wordThree = getaudiodata(PTwoWOne);
wordFour = getaudiodata(PTwoWTwo);

figure();
subplot(2,2,1);
plot(linspace(0,2,length(wordOne)),wordOne);
hold on
ylabel('intensity');
xlabel('time (s)');
title('First Person Word 1');
hold off
subplot(2,2,2);
plot(linspace(0,2,length(wordTwo)),wordTwo);
hold on
ylabel('intensity');
xlabel('time(s)');
title('First Person Word 2');
hold off
subplot(2,2,3);
plot(linspace(0,2,length(wordThree)),wordThree);
hold on
ylabel('intensity');
xlabel('time (s)');
title('Second Person Word 1');
hold off
subplot(2,2,4);
plot(linspace(0,2,length(wordFour)), wordFour);
hold on
```

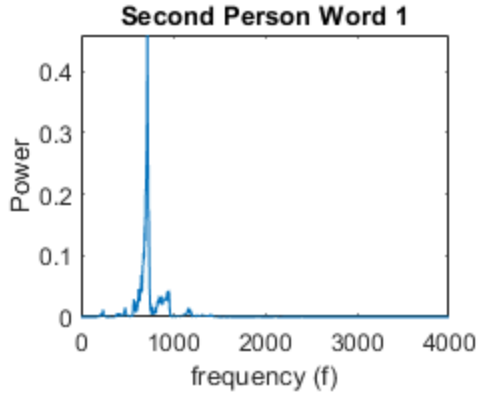
```
ylabel('intensity');
xlabel('time (s)');
title('Second Person Word 2');
hold off

PSD1 = pwelch(wordOne);
PSD2 = pwelch(wordTwo);
PSD3 = pwelch(wordThree);
PSD4 = pwelch(wordFour);

figure();
subplot(2,2,1);
plot(linspace(0,4000,length(PSD1)),PSD1);
hold on
ylabel('Power');
xlabel('frequency (f)');
title('First Person Word 1');
hold off
subplot(2,2,2);
plot(linspace(0,4000,length(PSD2)),PSD2);
hold on
ylabel('Power');
xlabel('frequency (f)');
title('First Person Word 2');
hold off
subplot(2,2,3);
plot(linspace(0,4000,length(PSD3)),PSD3);
hold on
ylabel('Power');
xlabel('frequency (f)');
title('Second Person Word 1');
hold off
subplot(2,2,4);
plot(linspace(0,4000,length(PSD4)), PSD4);
hold on
ylabel('Power');
xlabel('frequency (f)');
title('Second Person Word 2');
hold off

filtWordOne = 20*filter(filt,wordOne);

% The filter used was a high-pass filter with a transition band from
 500 Hz
% to 600 Hz. Applying the high pass filter allowed us to take out the
 lower
% frequencies, which gave the voice a brighter tone.

figure();
subplot(1,2,1)
plot(linspace(0,4000,length(PSD1)),PSD1);
ylabel('Power');
xlabel('frequency (f)');
title('PSD of first person first word')
```

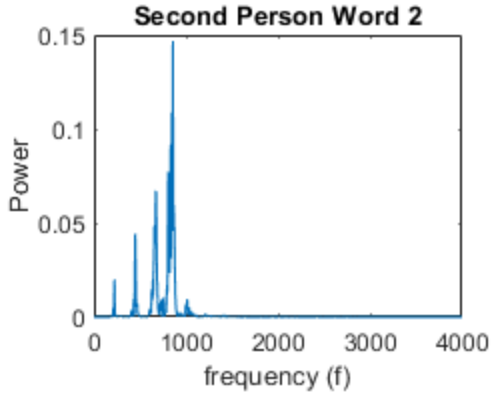
```
subplot(1,2,2)
plot(linspace(0,4000,length(pwelch(filtWordOne))),pwelch(filtWordOne));
ylabel('Power');
xlabel('frequency (f)');
title('PSD of filtered first word');
```

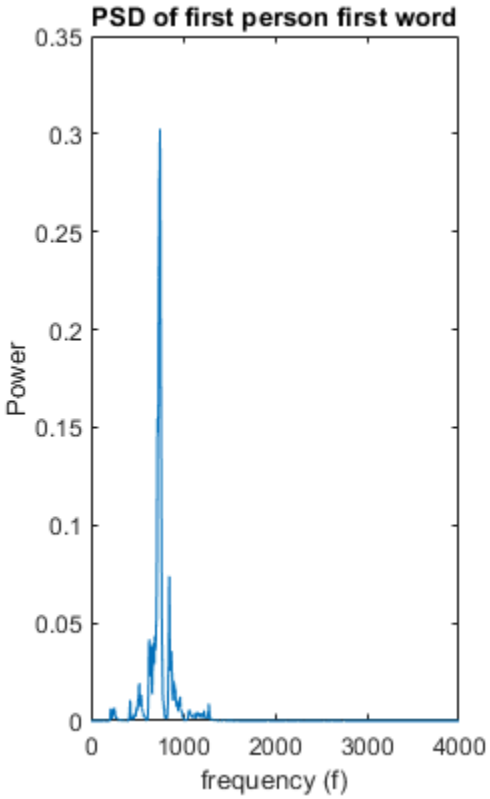


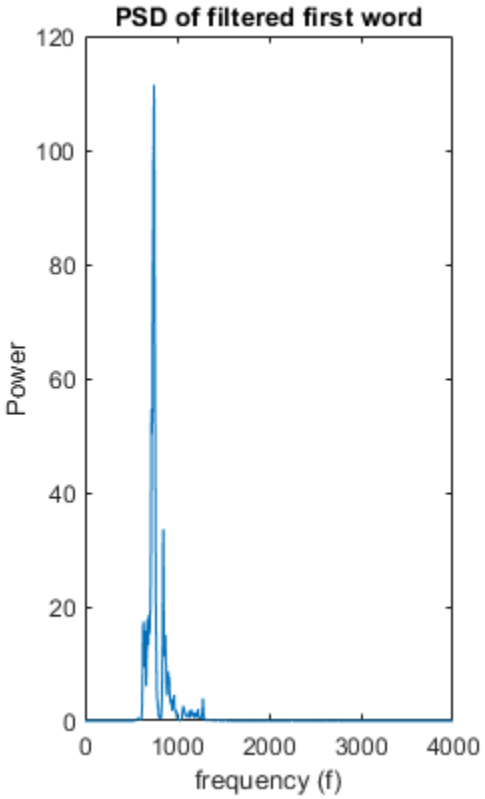

First Person Word 1
Power
frequency (f)
0
0.1
0.2
0.3
0
1000
2000
3000
4000

First Person Word 2
Power
frequency (f)
0
0.1
0.2
0.3
0
1000
2000
3000
4000


Second Person Word 1
Power
frequency (f)
0
0.1
0.2
0.3
0.4
0
1000
2000
3000
4000

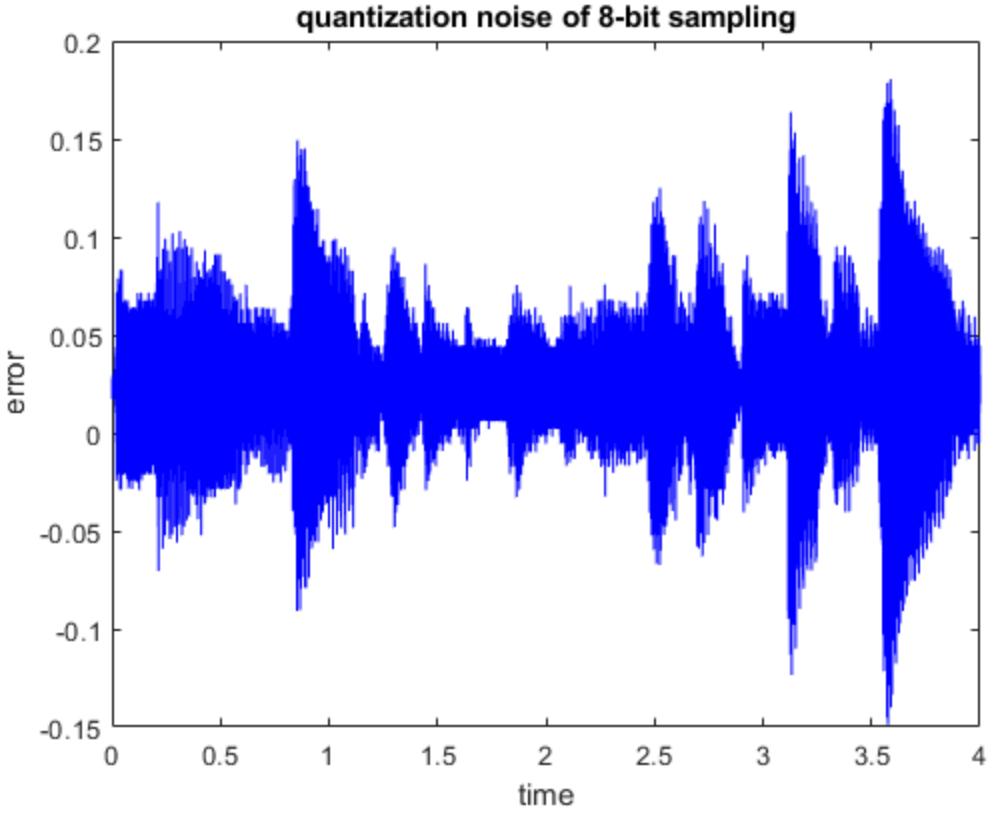

Second Person Word 2
Power
frequency (f)
0
0.05
0.1
0.15
0
1000
2000
3000
4000

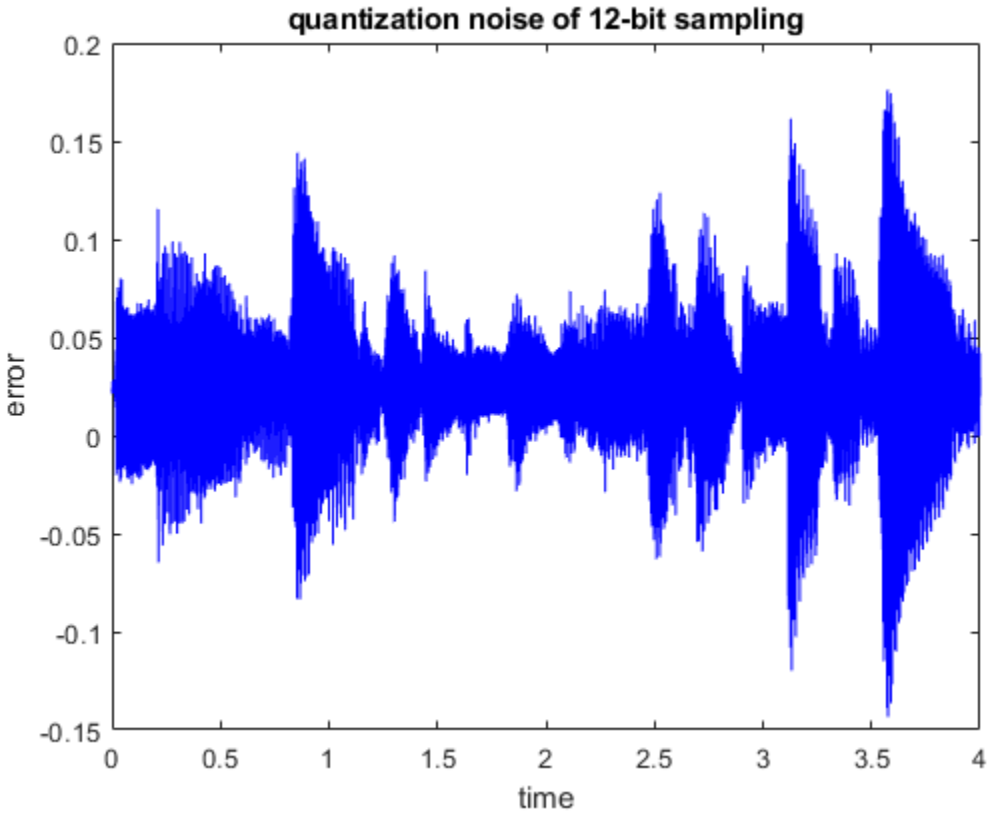

PSD of first person first word
Power
frequency (f)
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0
1000
2000
3000
4000


PSD of filtered first word
Power
frequency (f)
0
20
40
60
80
100
120
0
1000
2000
3000
4000

Part C

```
Original = audioread('EE427ProjectSoundClip.wav');

eightBit = double(uencode(Original,8));
eightBit = eightBit/max(eightBit);
eightBit = 2*(eightBit - ((1/2)*(max(eightBit)-
min(eightBit))+min(eightBit)));
d1 = Original(:,1)-eightBit;

figure();
plot(linspace(0,4,length(d1)),d1,'b');
title('quantization noise of 8-bit sampling');
xlabel('time');
ylabel('error');

twelveBit = double(uencode(Original,12));
twelveBit = twelveBit/max(twelveBit);
twelveBit = 2*(twelveBit - ((1/2)*(max(twelveBit)-
min(twelveBit))+min(twelveBit)));
d2 = Original(:,1)-twelveBit;

figure();
plot(linspace(0,4,length(d2)),d2,'b');
title('quantization noise of 12-bit sampling');
xlabel('time');
ylabel('error');

% when playing the audio back, it sounds similar to the original
 audio. However,
% the quantization adds a "grainy" noise to the background of the
% audio file.
```


quantization noise of 8-bit sampling
error
time
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
0
0.5
1
1.5
2
2.5
3
3.5
4


quantization noise of 12-bit sampling
error
time
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
0
0.5
1
1.5
2
2.5
3
3.5
4

Published with MATLAB® R2019b