Lab 1 – Introduction to Digital Signals using MATLAB

Ranir, 400311161

COMPENG 4TL4

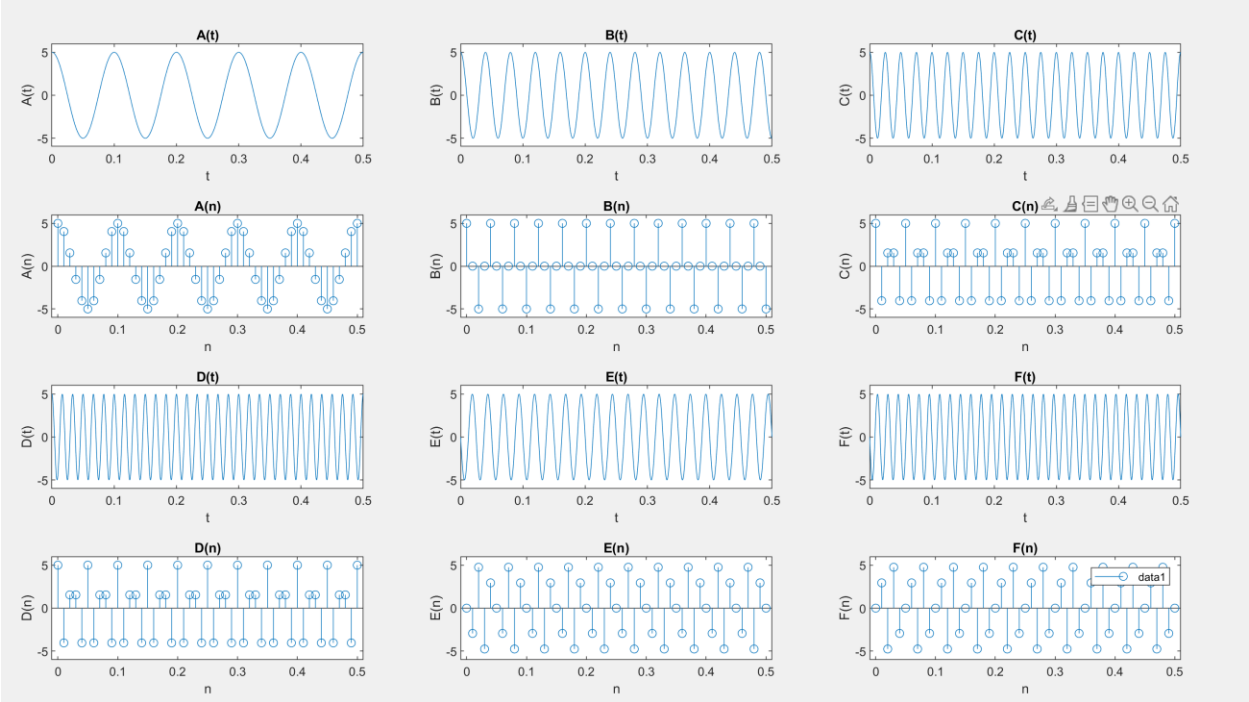
L04

September 24, 2024

1. Point sampling of a sinusoid

a. **Create a MATLAB script that point-samples x(t) at a sampling frequency Fs of 100 Hz, for t = 0 s to 0.5 s. Using the MATLAB function stem(), plot the discrete-time sequence x[n] versus sample number n obtained for each of the following sets of parameters:**

	Amplitude: A	Frequency: f (Hz)	Phase: ϕ (rad)
A	5	10	0
B	5	25	0
C	5	40	0
D	5	60	0
E	5	40	$\frac{\pi}{2}$
F	5	60	$\frac{\pi}{2}$



(b) Visually determine the period of each of the discrete-time signals. Compare the discrete time signals' period and waveform with those of the continuous time signals. Explain the differences in the periods using your knowledge of the Nyquist sampling theorem

Visually inspect period

To visually inspect the periods, I count the peaks or troughs of the signals and divide the total time (0.5 s) by the number of peaks/cycles this will give me the period of one signal. For accuracy, I will do total number of peaks or total numbers of troughs and take the one with more

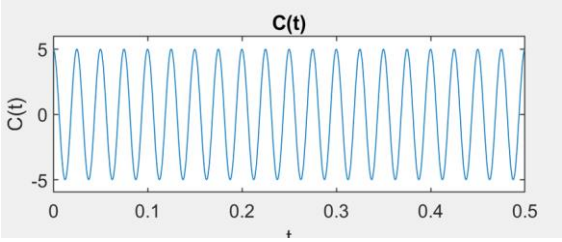
A(n)	0.5s/5.5peaks = 0.091s
B(n)	0.5s/13peaks = 0.038s
C(n)	0.5s/20troughs = 0.025s
D(n)	0.5s/20troughs = 0.025s

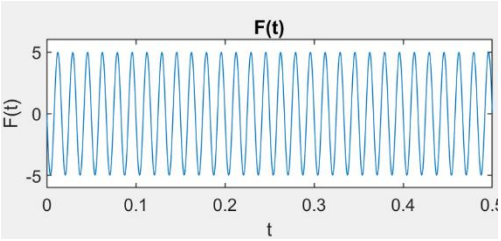
E(n)	0.5s/10peaks = 0.05s
F(n)	0.5s/10peaks = 0.05s

The difference between the continuous and discrete time signals is a matter of how accurate they are. As the frequency increases, the accuracy of the discrete time signal decreases. For example, for D(n), when the discrete time signals period is visually determined, it is 0.5s / 20 troughs = 0.025s. However, when the period is visually determined with the continuous signal, it would be 0.5s/30peaks = 0.0167s. The reason it gets less accurate as the frequency increases is because of the Nyquist Sampling Theorem. This means that when choosing the frequency for the discrete time signal, it should be 2*fs of the continuous signal to be accurate

(c) Explain the effect of the phase change in cases C to F

The phase change between C to F shifts the signal over by the amount of the phase change. This is shows by the difference between the starting point of C and F.

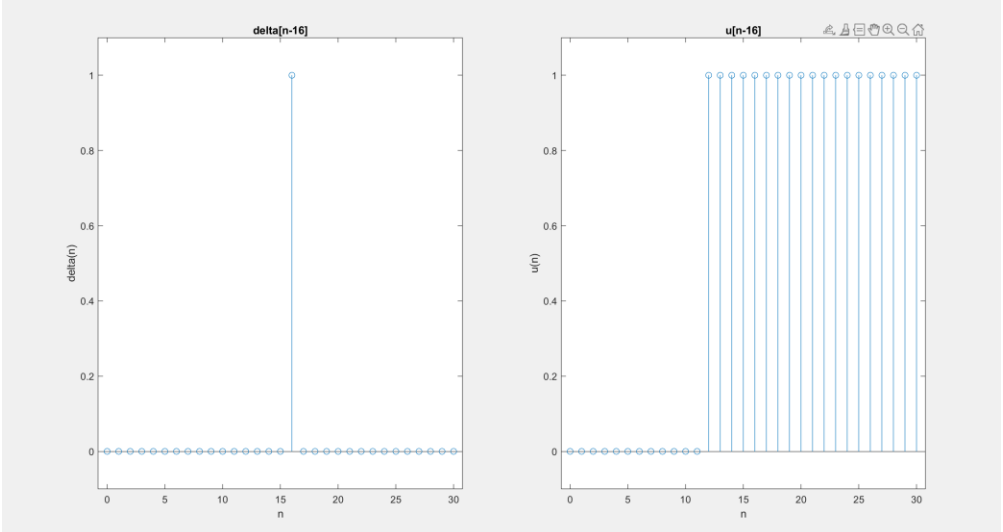




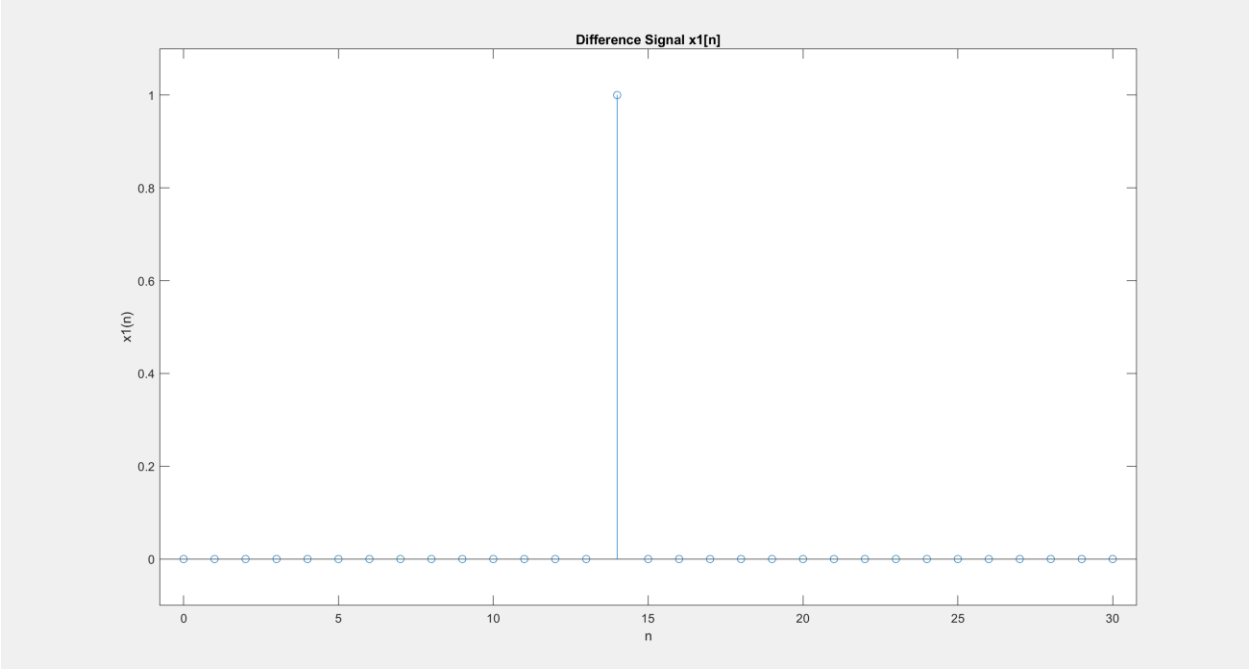
As you can see, C(t) starts at 5 and F(t) starts at 0 due to the pi/4 phase shift

2. Working with unit impulses and unit steps

(a) Create the unit impulse signal δ[n − 16] and the unit step signal u[n − 12] for samples n = 1, 2, . . . , 30 and plot both using the function stem()

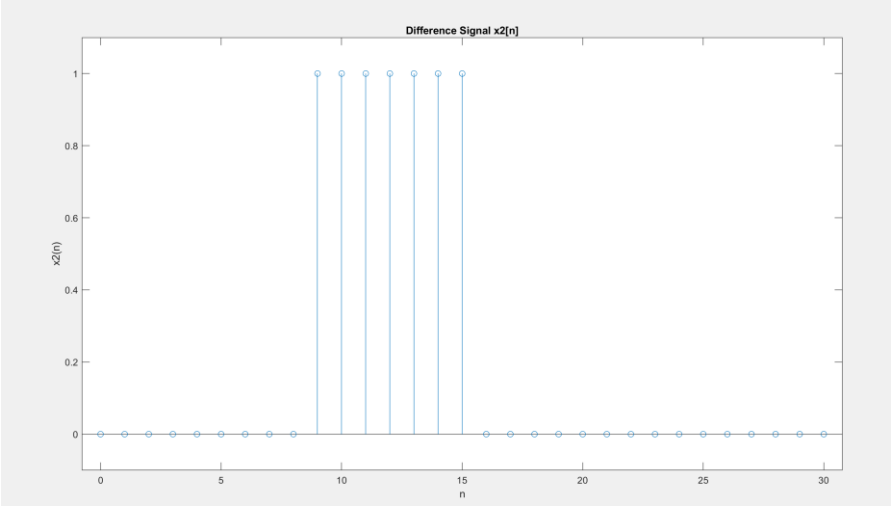


(b) Plot the difference signal x1[n] = u[n − 14] − u[n − 15]. Interpret this signal as δ[n − k] and find the value of k from the plot.



To interpret this as δ[n − k] it would be interpreted as δ[n − 14] because it is shifted to the right by 14

(c) Plot another difference signal x2[n] = u[n − 9] − u[n − 16]. Determine the width of the resulting rectangular "pulse". Express this signal as a sum of unit impulses.

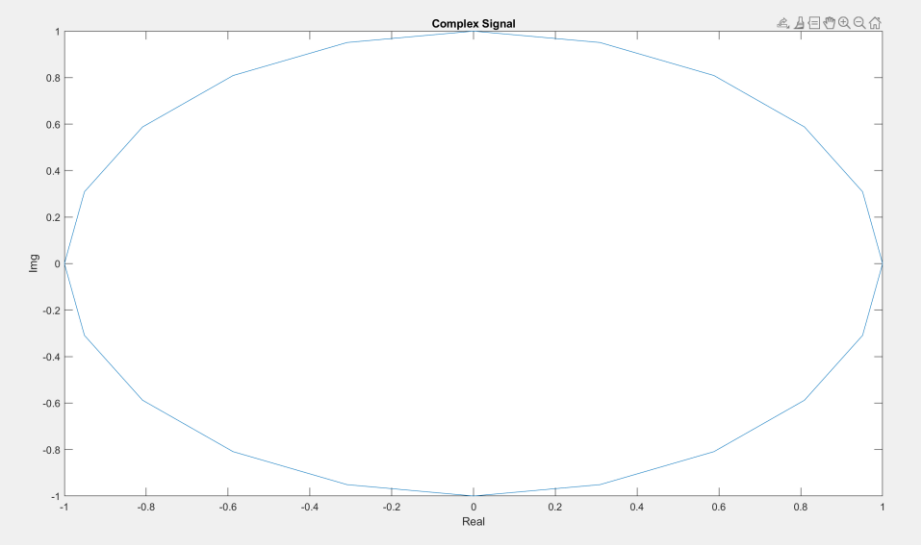


You can express this as a sum of unit impulses

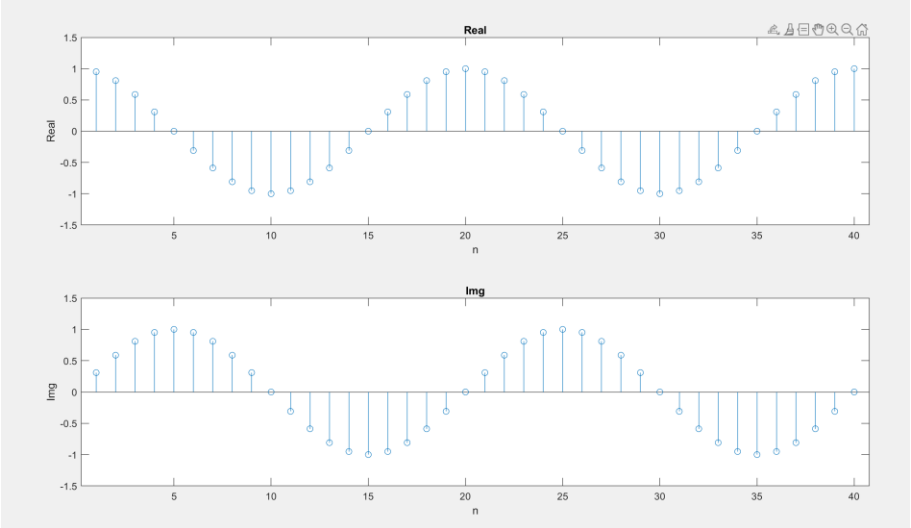
$$x2[n] = \delta[n - 9] + \delta[n - 10] + \delta[n - 11] + \delta[n - 12] + \delta[n - 13] + \delta[n - 14] + \delta[n - 15] + \delta[n - 16]$$

3. Complex signals

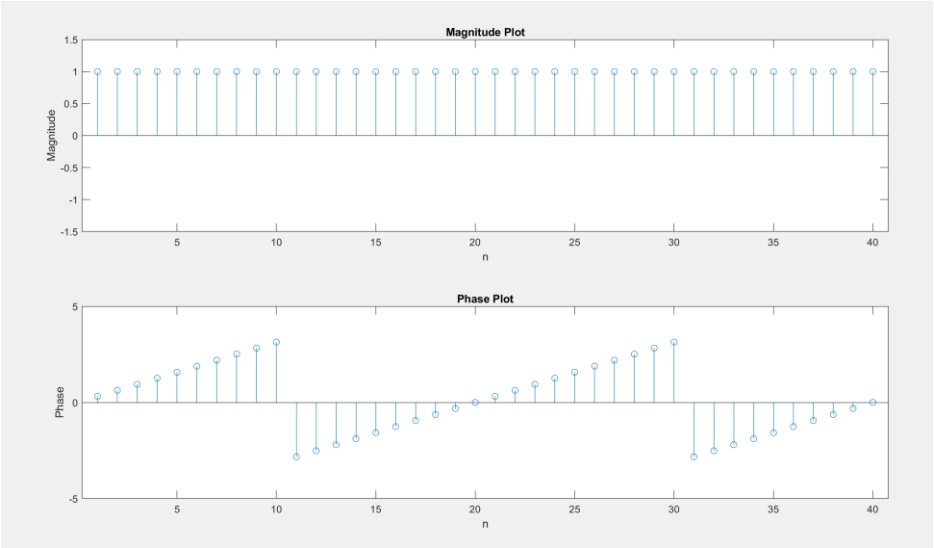
(a) Plot x[n] in the complex plane (i.e., $\Im$(x[n]) imaginary part versus $\Re$(x[n]) real part), using the MATLAB command plot().



(b) **Extract the real and imaginary parts of signal using the functions real() and imag() and plot them versus the sample number n using stem(). Use subplot(2,1,1) and subplot(2,1,2) commands prior to each stem() command to create plots of real and imaginary parts placed on the same figure but different axes.**



(c) Generate another figure with separate stem plots of the magnitude and phase of x[n] versus sample number n. Does the plot of the magnitude make sense? What is the slope of the phase versus sample number curve, and how does this relate to the parameters used to generate x[n]?

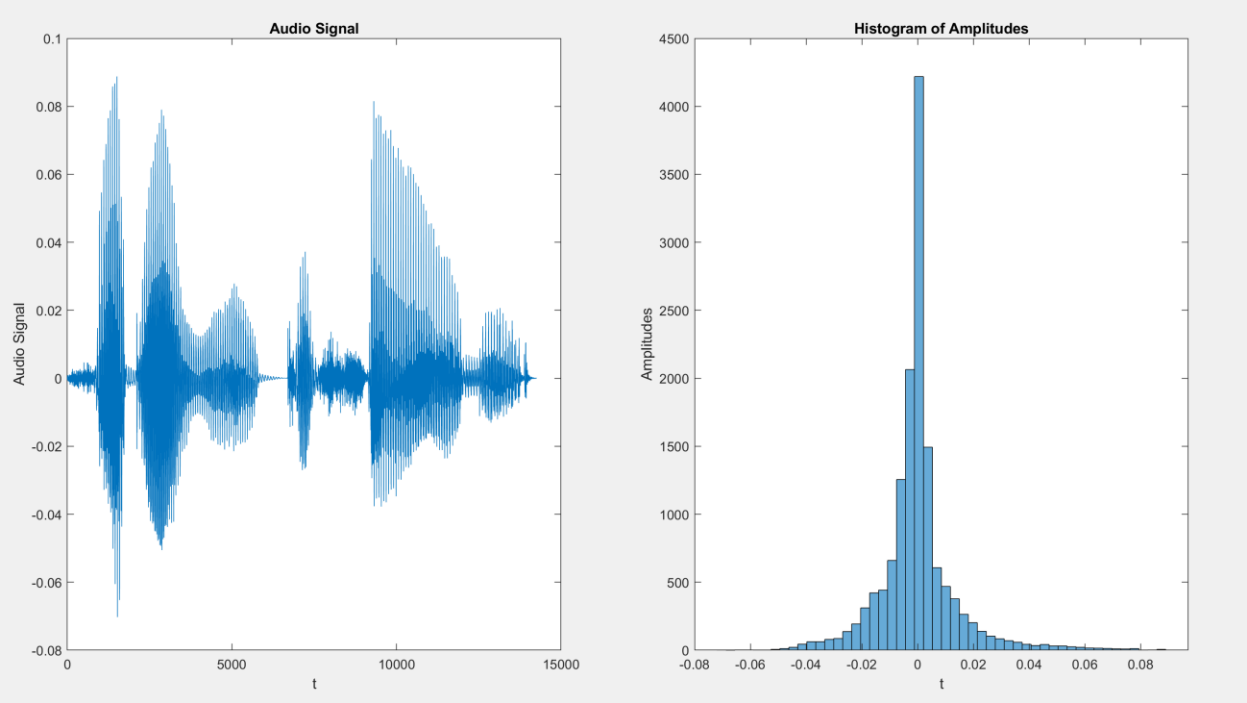


Magnitude: The magnitude of the plot makes sense because the radius of the complex signal is 1

Slope of the phase: The slop of the phase relates to the angular frequency w which is pi/10. Slope is positive so the frequency is positive.

4. Quantization of a speech signal

a) Load the supplied speech signal into MATLAB using the command:
[y,fs] = audioread('defineit.wav');
(b) Plot the speech waveform y using the plot() command, and plot a histogram of amplitude values using the command histogram(y,50). Describe the waveform and the shape of the histogram in terms of:
• The number of bits at which the .wav file was quantized.
The following command can be used to get the number of bits used.
info = audioinfo('defineit.wav');
• The typical probability density function of speech




```
CompressionMethod: 'Uncompressed'
      NumChannels: 1
       SampleRate: 16000
     TotalSamples: 14259
         Duration: 0.8912
            Title: []
          Comment: []
           Artist: []
    BitsPerSample: 16
```

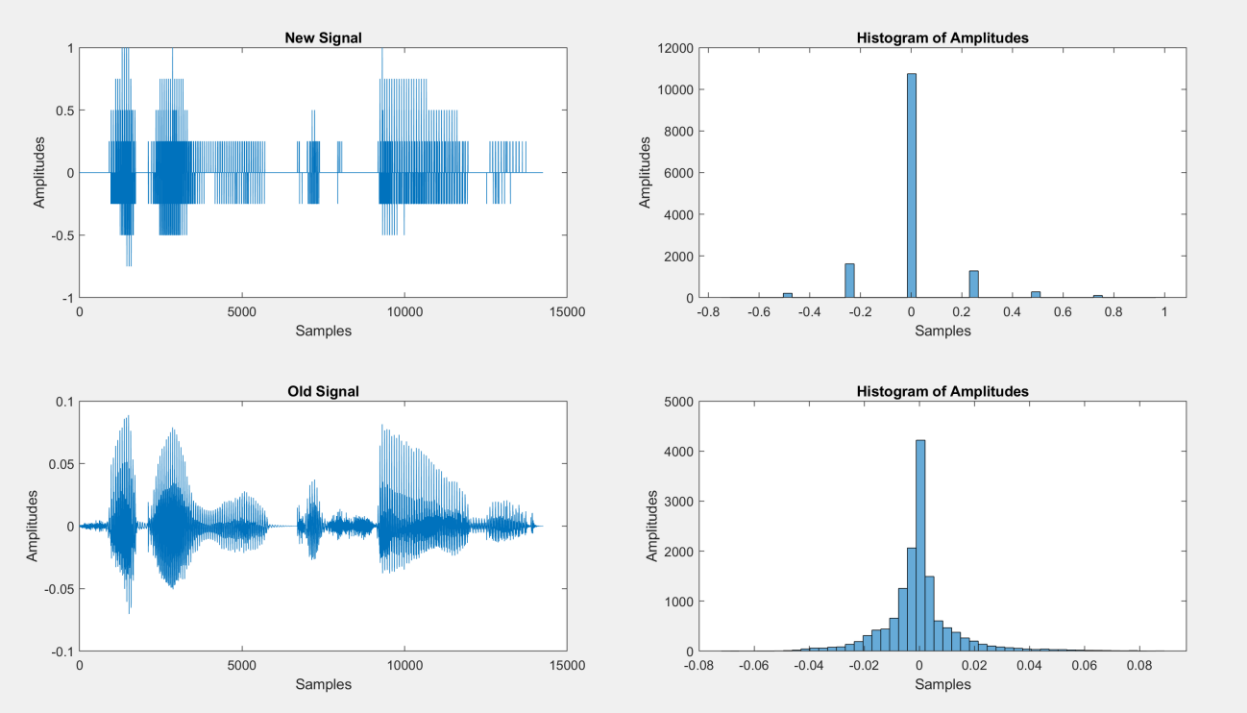
The number of bits that the waveform is quantized at is 16 bits per sample. This means that there can be 2^{16} or 65,536 different amplitude levels.

The pdf of the function is gaussian which makes sense because a lot of the signal is times of quiet or low volume speech so a large chunk of the signal is based around 0 . This is represented by the peak in the histogram being centered around 0

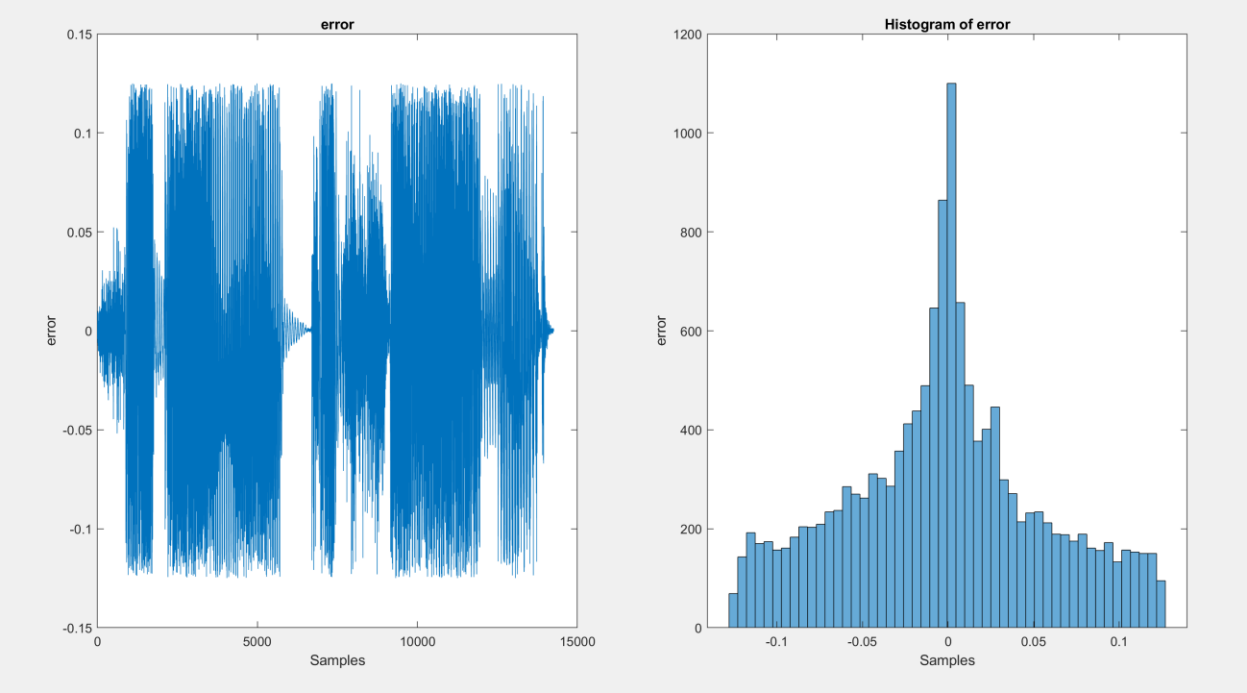
(c) Listen to y using the soundsc(y, fs) command. Describe the sound quality

The sound quality was understandable. This tells me that the original signal probably had a frequency of 800 because the sample rate of the plot is 16000

d) Write a MATLAB script for a 3-bit rounding uniform quantizer for the range [-1 1].
(e) Scale the speech signal y so that no saturation ("peak clipping") will occur when it is passed through the quantizer and full quantization range is used – call this new variable y_scaled. Hint: divide by the maximum amplitude.
(f) Quantize the scaled signal y_scaled (call the quantized signal y3bit) and repeat parts b and c above, but for the newly quantized signal, comparing the new results to the results from parts b and c. In addition, plot the waveform and histogram for the error e = y_scaled - y3bits, and describe this in terms of the standard statistical model of quantization noise



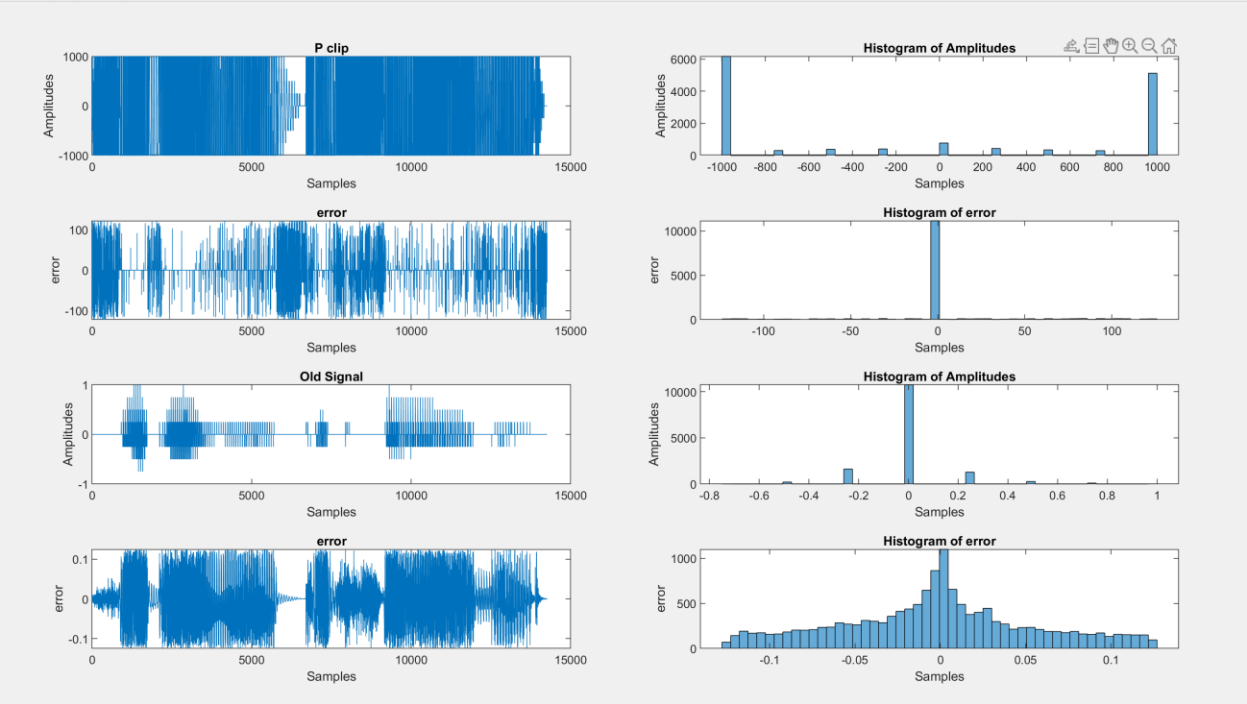
In the old signal the range was between 0.1 and -0.1. After quantization, the range increased to 1 to -1, the signal was clipped at 1 , the amplitudes were rounded into bins which is why the histogram has a much higher number of samples at 0 and fewer bins.



The error shows that a majority of the error is centered around 0. This makes sense because the bins are 0.25

In terms of the standard statistical model of quantization noise, we used a uniform quantizer. This means that it does not take the shape of the model in account. This is why there is much more error around 0 rather than anywhere else. If it was a non uniform quantizer, I would make more bins around 0. The impact on the audio is that there is much signal being lost

(g) Rescale y to greatly exceed the maximum scale range of your 3-bit quantizer, such that substantial peak clipping will occur – call this new variable y_pclip. Quantize this rescaled signal (call it y3bit_pclip) and repeat part f above, but for the peak clipped quantized signal. Again, compare the new results to the previous results



As shown here, when the signal is overly amplified and then scaled back down, it clips a lot more than necessary which causes a substantial amount of the signal to be lost. Since so much of the signal has been clipped, you can see in the histogram of amplitudes that a majority of the signal is at 1 and -1