American International University- Bangladesh
Department of Computer Engineering
COE 3201: Data Communication Laboratory

Title: **Study of signal frequency, spectrum, bandwidth, bit rate, quantization using MATLAB**

Abstract:

This experiment is designed to-

1.To understand the use of MATLAB for solving communication engineering problems.
2.To develop understanding of MATLAB environment, commands and syntax.

Introduction:

I. **Frequency:** The frequency of a wave describes how many waves go past a certain point in one second. Frequency is measured in Hertz (usually abbreviated Hz), and can be calculated using the formula:

$V = f\lambda$

where V is the velocity of the wave (in ms^{-1}), f is the frequency of the wave (in Hz), and λ (the Greek letter lambda) is the wavelength of the wave (distance from one peak / trough to the next, in m). Frequency is the rate of change with respect to time. Change in a short span of time means high frequency. Change over a long span of time means low frequency.

II. **Spectrum:** Usually we represent signals in time domain. But signals can be represented in frequency domain as well. When signals are represented in frequency domain they are called spectrum.

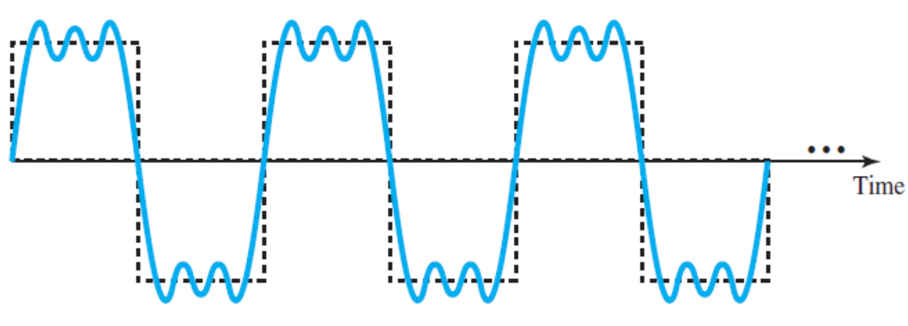


Fig: A composite periodic signal

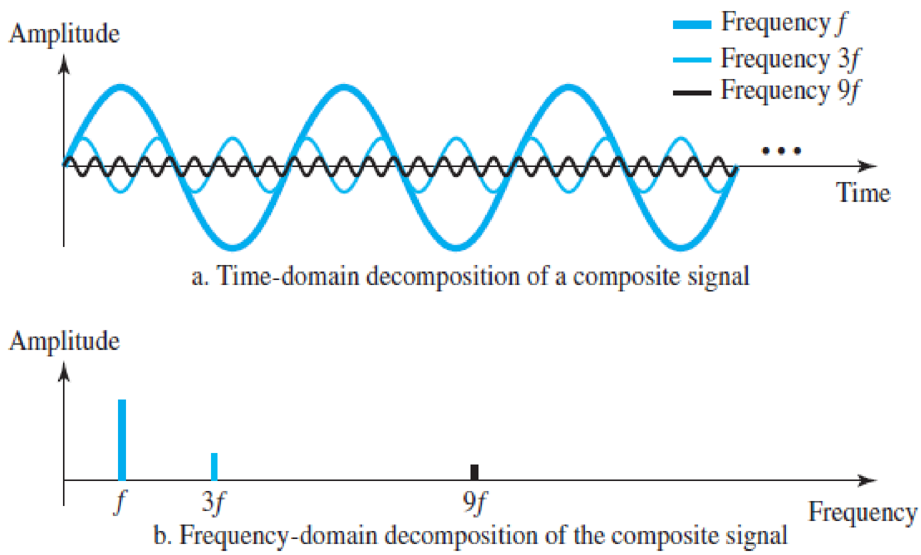
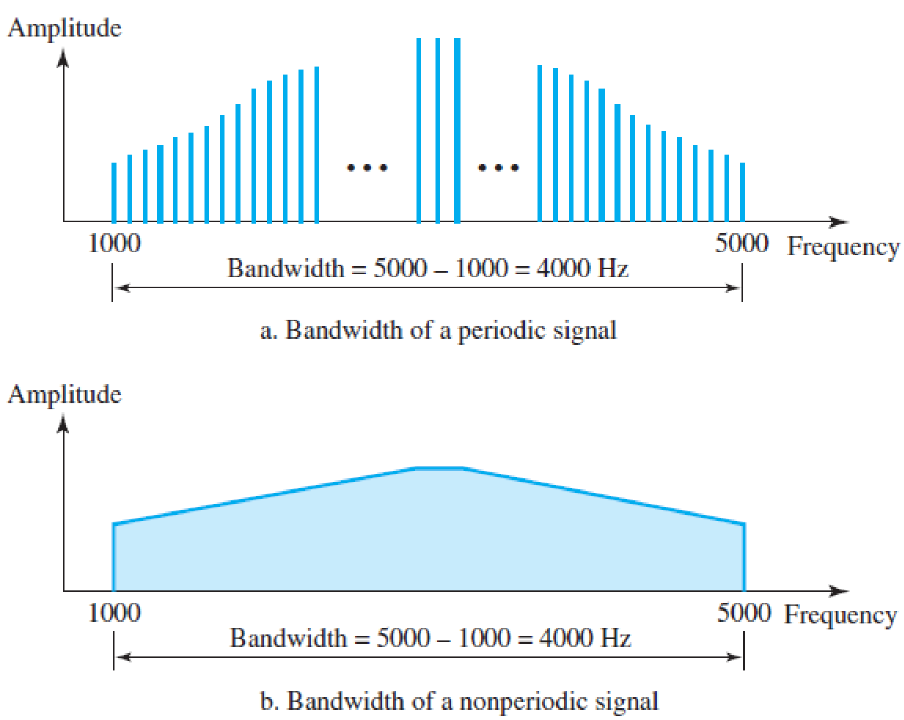


Fig: Decomposition of a composite periodic signal in the time and frequency domains

III. **Bandwidth:** Bandwidth is the range of frequency a signal contains in it. If a composite signal is made up of multiple sinusoids of 100, 250, 300, and 400 Hz. Then its bandwidth is the difference of the highest and lowest frequency components. So here the bandwidth of the signal is (400-100) = 300 Hz.



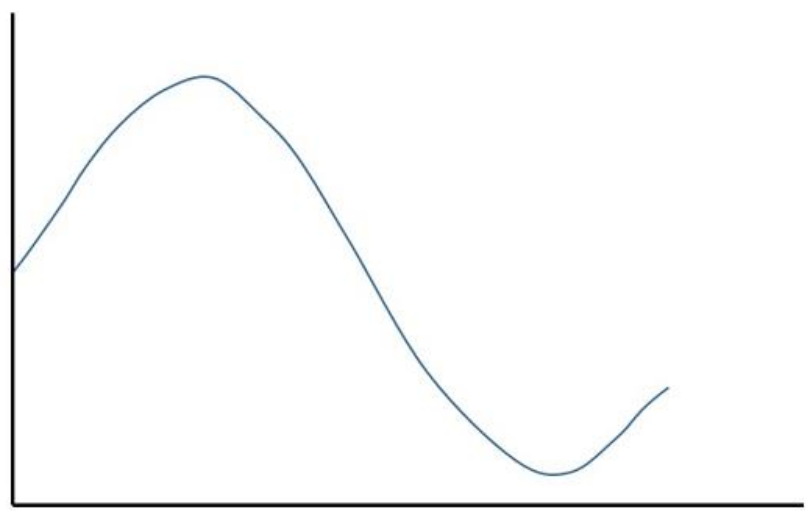
IV. **Bit rate:** Bit Rate, sometimes also written as bitrate, indicates the amount of data that is processed or delivered over a certain time span. The data is measured in bits while the time is measured in seconds, resulting in a bit/s or bps (bits by seconds) unit. When dealing with a high Bit Rate, the unit can also be expressed in kilobit (kbps), megabit (mbps), gigabit (gbps), and so on.

When talking about the Bit Rate of audio or video files, it often affects the quality of the audio data in both cases. As a rule of thumb, the higher the Bit Rate, the better the audio quality. For example, an MP3 file with a Bit Rate of 192 kbps will have a considerably better quality than a file that has a Bit Rate of only 32 kbps. Furthermore, the Bit Rate affects the image quality of a video file.

V. **Quantization:** The digitization of analog signals involves the rounding off of the values which are approximately equal to the analog values. The method of sampling chooses a few points on the analog signal and then these points are joined to round off the value to a near stabilized value. Such a process is called as Quantization.

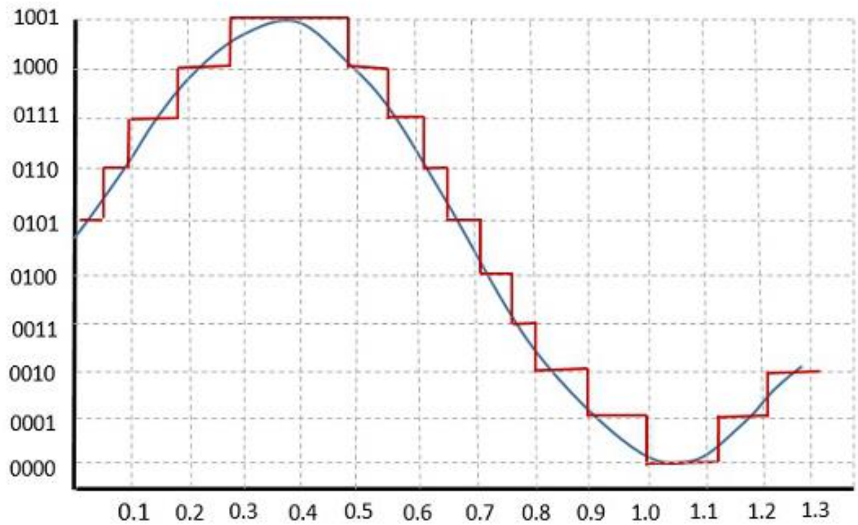
Quantizing an Analog Signal:

The analog-to-digital converters perform this type of function to create a series of digital values out of the given analog signal. The following figure represents an analog signal. This signal to get converted into digital has to undergo sampling and quantizing.



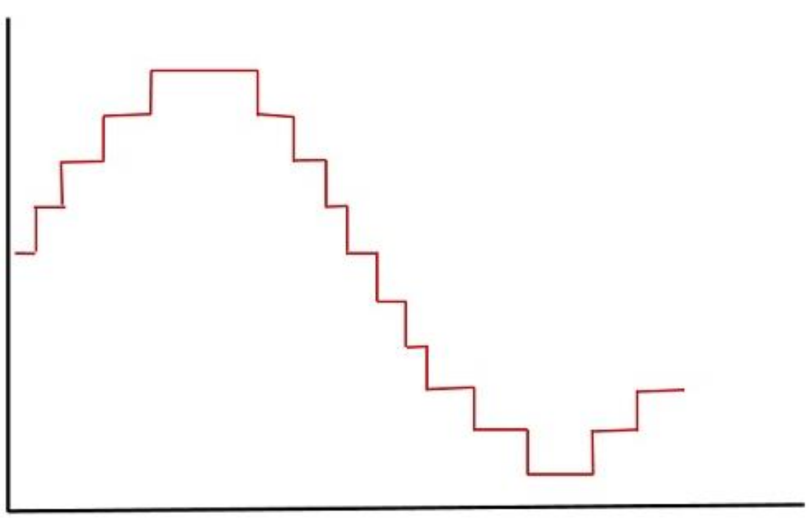
The quantizing of an analog signal is done by discretizing the signal with a number of quantization levels. Quantization is representing the sampled values of the amplitude by a finite set of levels, which means converting a continuous-amplitude sample into a discrete-time signal.

The following figure shows how an analog signal gets quantized. The blue line represents analog signal while the brown one represents the quantized signal.



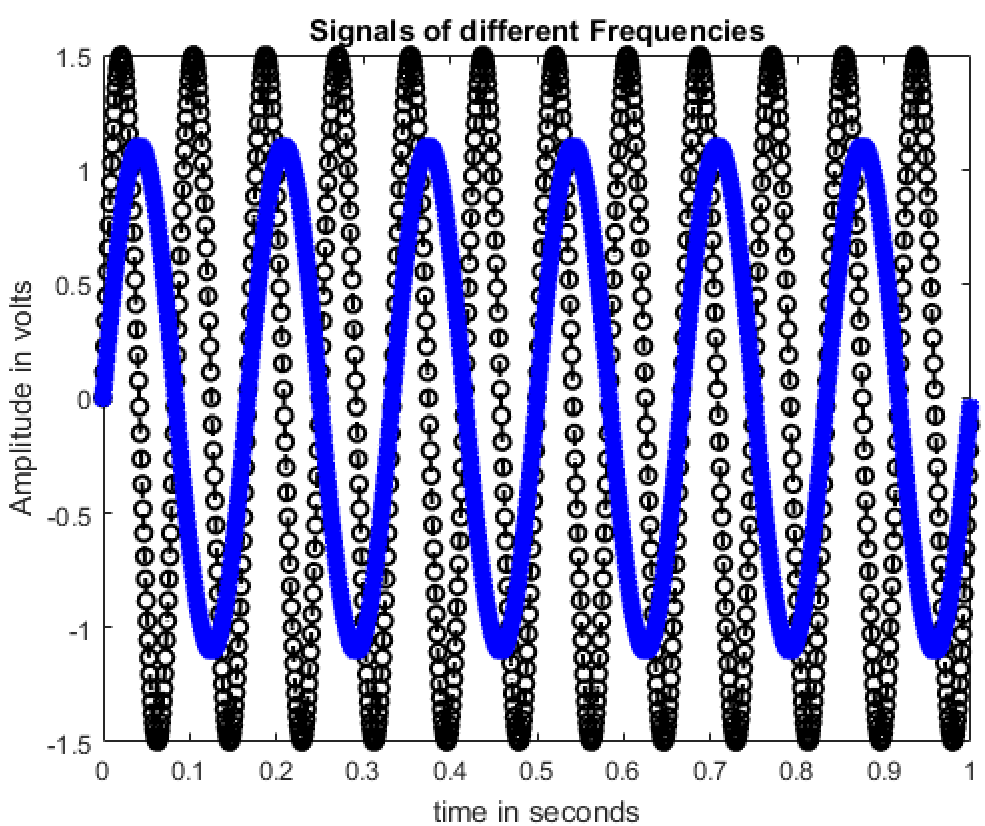
Both sampling and quantization result in the loss of information. The quality of a Quantizer output depends upon the number of quantization levels used. The discrete amplitudes of the quantized output are called as representation levels or reconstruction levels. The spacing between the two adjacent representation levels is called a quantum or step-size.

The following figure shows the resultant quantized signal which is the digital form for the given analog signal.



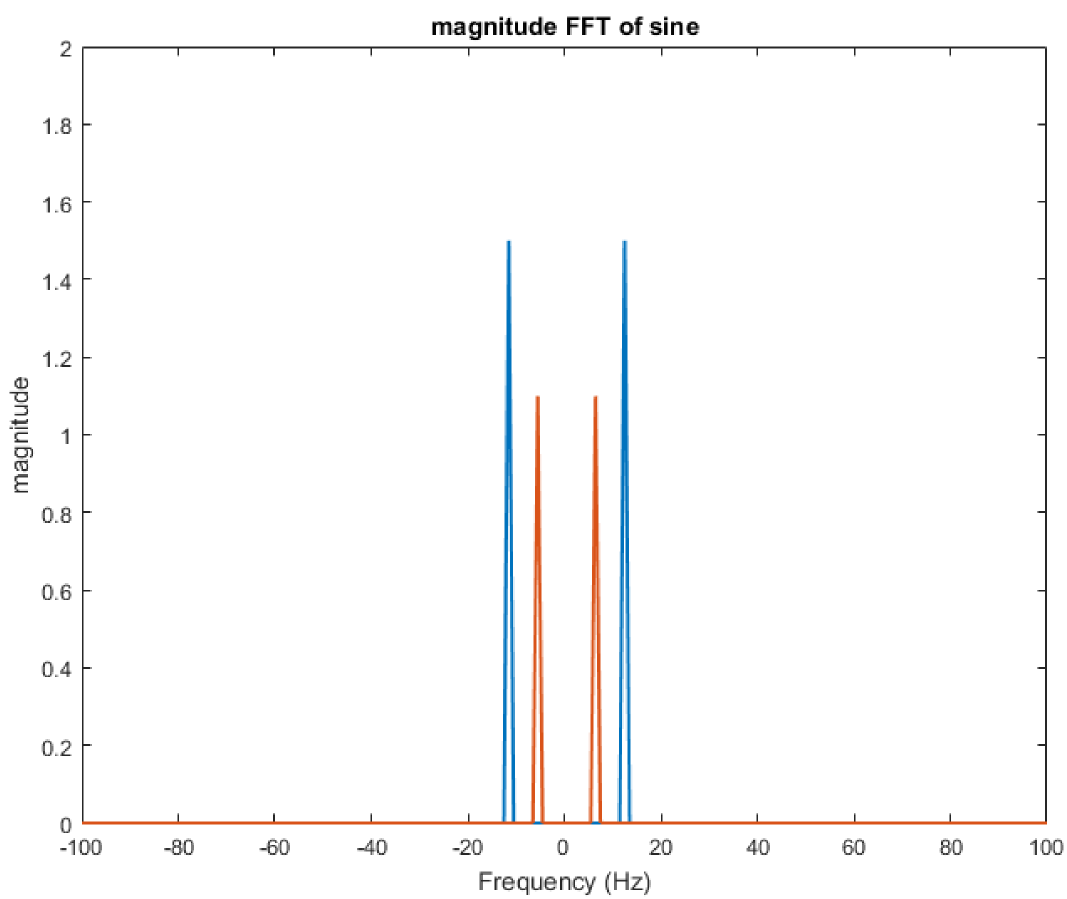
Generating a sinusoidal signal with different frequencies:

```
fs = 1000; % Sampling frequency
t = 0:1/fs:1-1/fs; % Time duration
f1 = 12; % Frequency of first signal
f2 = 6; % Frequency of second signal
A1 = 1.5; % Amplitude of first signal
A2 = 1.1; % Amplitude of second signal
x1 = A1*sin(2*pi*f1*t); % First Signal
x2 = A2*sin(2*pi*f2*t); % Second Signal
%Plotting both signals in time domain
plot(t,x1,'k--o','LineWidth',1.5)
hold on
plot(t,x2,'b-*','LineWidth',2)
hold off
xlabel('time in seconds')
ylabel('Amplitude in volts')
title('Signals of different Frequencies')
```



These signals can be represented in frequency domain as well:

```
%Take fourier transform
fx1 = fft(x1);
fx2 = fft(x2);
%apply fftshift to put it in the form we are used to (see
documentation)
fx1 = fftshift(fx1)/(fs/2);
fx2 = fftshift(fx2)/(fs/2);
%Next, calculate the frequency axis, which is defined by the
sampling rate
f = fs/2*linspace(-1,1,fs);
%Since the signal is complex, we need to plot the magnitude to
get it to
%look right, so we use abs (absolute value)
figure;
plot(f, abs(fx1), f, abs(fx2),'LineWidth',1.5);
title('magnitude FFT of sine');
axis([-100 100 0 2])
xlabel('Frequency (Hz)');
ylabel('magnitude');
```



Similar task can be done where we use a composite signal instead of signals x1 and x2. Suppose our composite signal is

```
cx = 1.1*sin(2*pi*2*t)+1.3*cos(2*pi*10*t)+1.5*sin(2*pi*20*t);
```

*******Show this signal in time domain and frequency domain representation.**

Example of Bandwidth calculation:

```
fs = 8000; % Sampling frequency
t = 0:1/fs:1-1/fs; % Time duration
cx = 1.1*sin(2*pi*100*t) + 1.3*cos(2*pi*300*t) +
1.5*sin(2*pi*2000*t);
bandwidth = obw(cx,fs)
```

bandwidth =

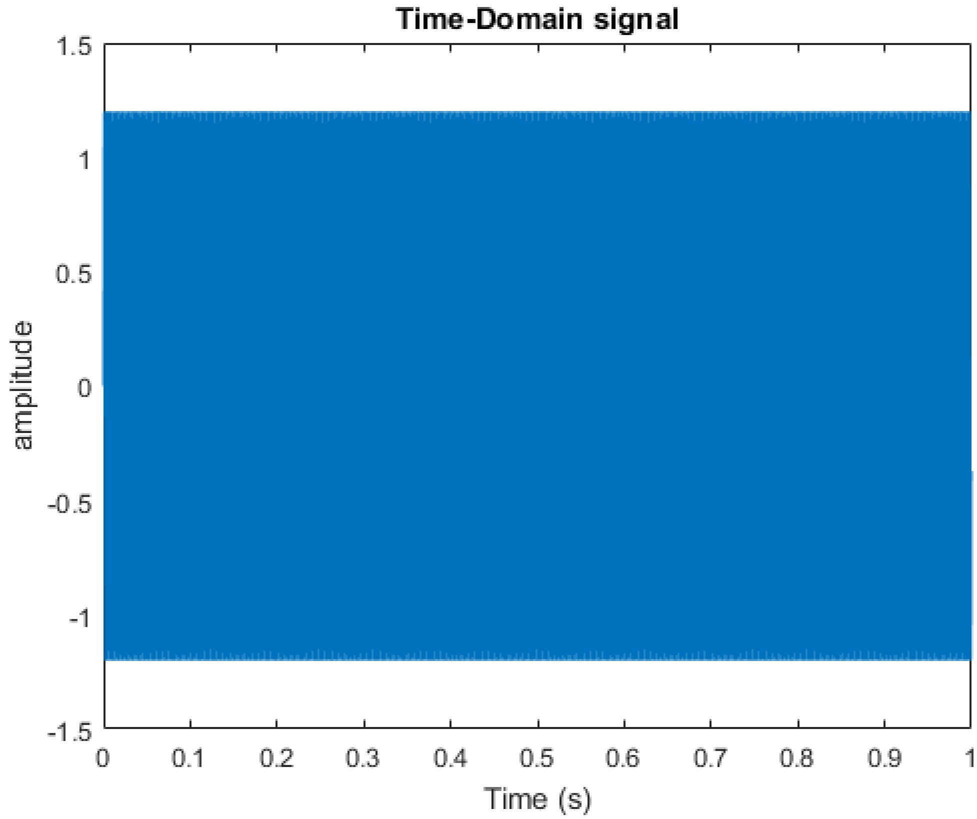
1.9010e+03

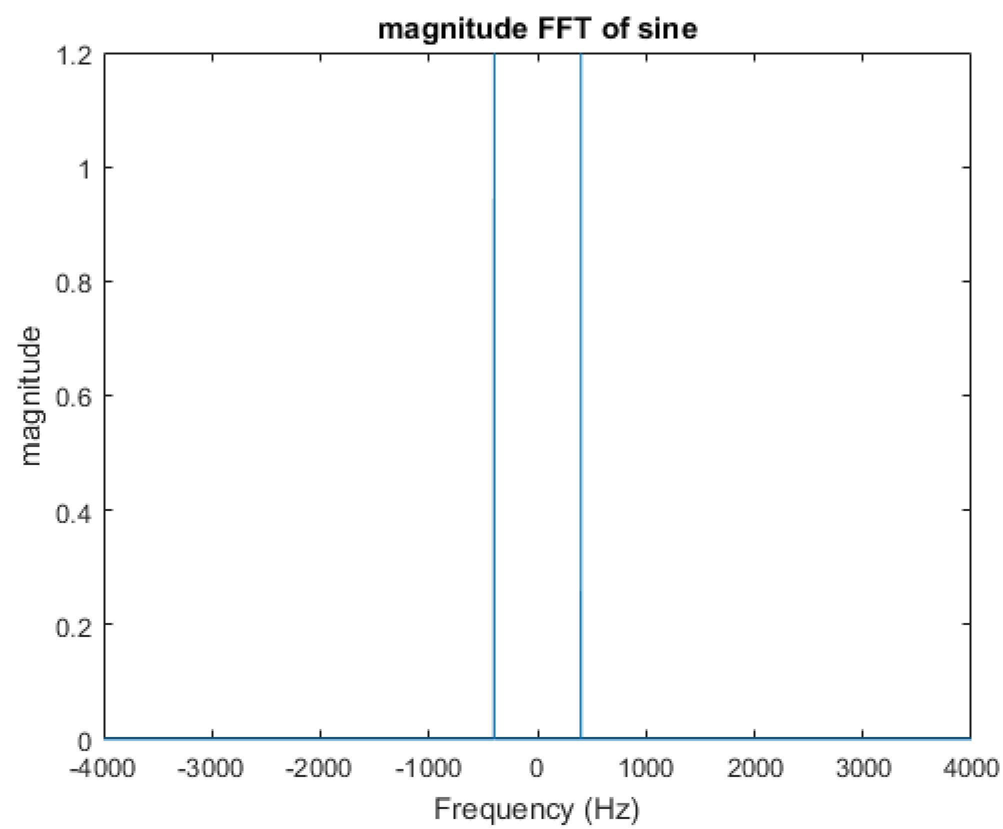
Another example of generating signals in time domain, adding noise and representing these signals in frequency domain

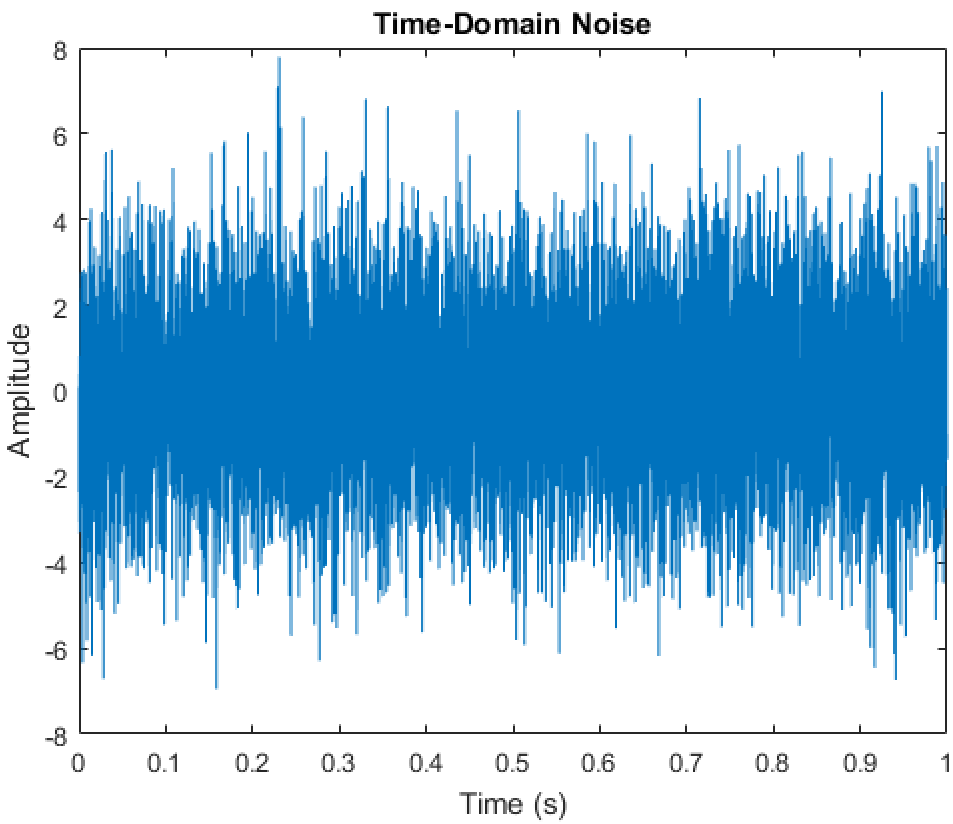
```
close all;
clc;
%Define number of samples to take
fs = 8000;
f = 400; %Hz
%Define signal
t = 0:1/fs:1-1/fs;
signal = 1.2*sin(2*pi*f*t);
%Plot to illustrate that it is a sine wave
plot(t, signal);
title('Time-Domain signal');
xlabel('Time (s)');
ylabel('amplitude');
%Take fourier transform
fftSignal = fft(signal);
%apply fftshift to put it in the form we are used to (see
documentation)
fftSignal = fftshift(fftSignal)/(fs/2);%scaling done by dividing
with (fs/2)
%Next, calculate the frequency axis, which is defined by the
```

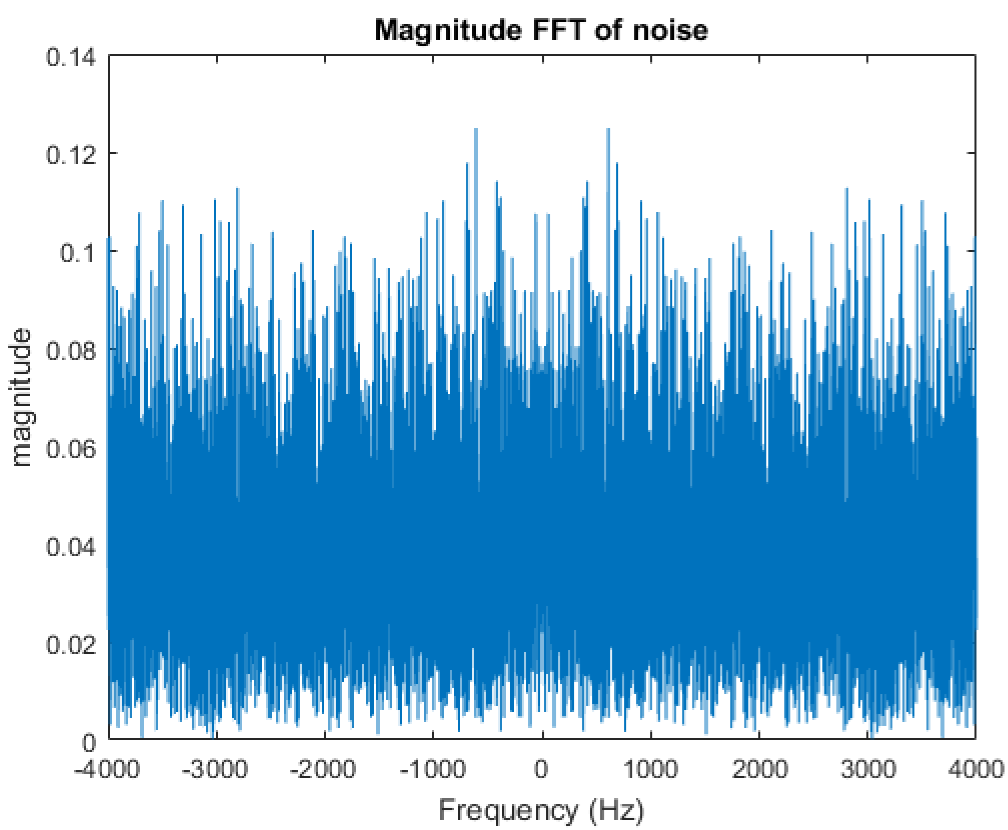
```
sampling rate
f = fs/2*linspace(-1,1,fs);
%Since the signal is complex, we need to plot the magnitude to
get it to
%look right, so we use abs (absolute value)
figure;
plot(f, abs(fftSignal));
title('magnitude FFT of sine');
xlabel('Frequency (Hz)');
ylabel('magnitude');
%noise
noise = 2*randn(size(signal));
figure
plot(t,noise)
xlabel('Time (s)');
ylabel('Amplitude');
title('Time-Domain Noise');
fftNoise = fft(noise);
fftNoise = fftshift(fftNoise)/(fs/2);
figure
plot(f,abs(fftNoise))
title('Magnitude FFT of noise');
xlabel('Frequency (Hz)');
ylabel('magnitude');
%noisy signal
noisySignal = signal + noise;
figure
plot(t,noisySignal)
xlabel('Time (s)');
ylabel('Amplitude');
title('Time-Domain Noisy Signal');
fftNoisySignal = fft(noisySignal);
fftNoisySignal = fftshift(fftNoisySignal)/(fs/2);
figure
plot(f,abs(fftNoisySignal))
title('Magnitude FFT of noisy signal');
xlabel('Frequency (Hz)');
ylabel('magnitude');
```

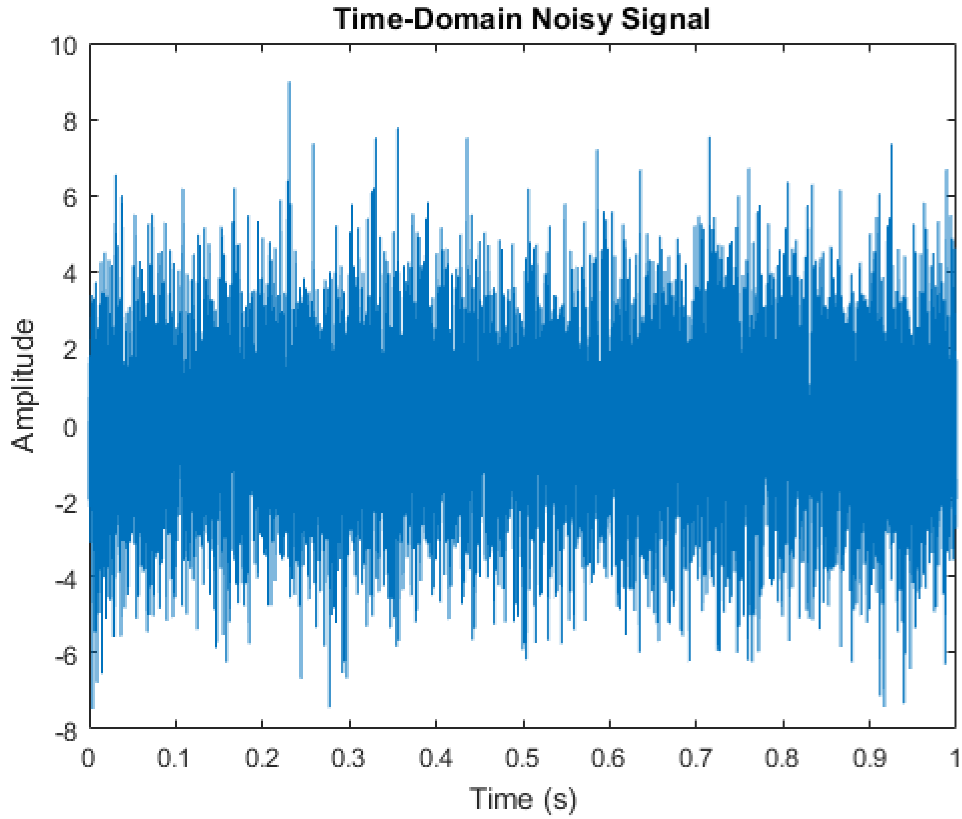
Generated figures from this example:

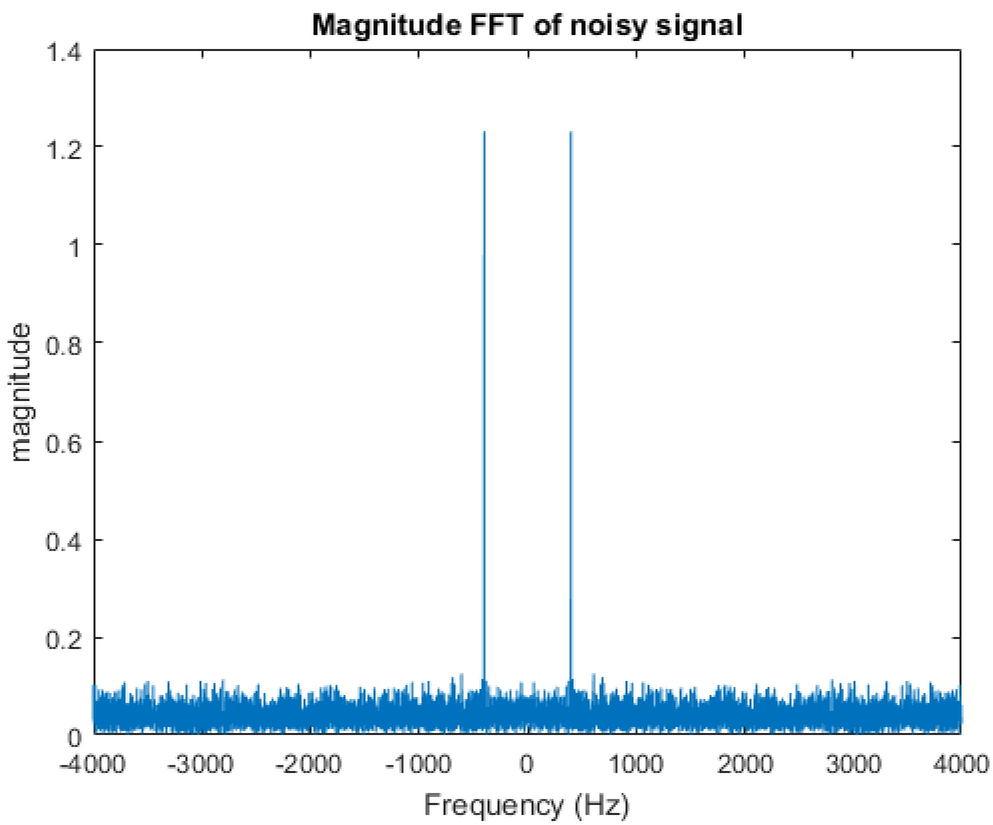

Time-Domain signal
1.5
1
0.5
0
-0.5
-1
-1.5
amplitude
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time (s)


magnitude FFT of sine
1.2
1
0.8
0.6
0.4
0.2
0
magnitude
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000
Frequency (Hz)


Time-Domain Noise
Amplitude
Time (s)
8
6
4
2
0
-2
-4
-6
-8
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

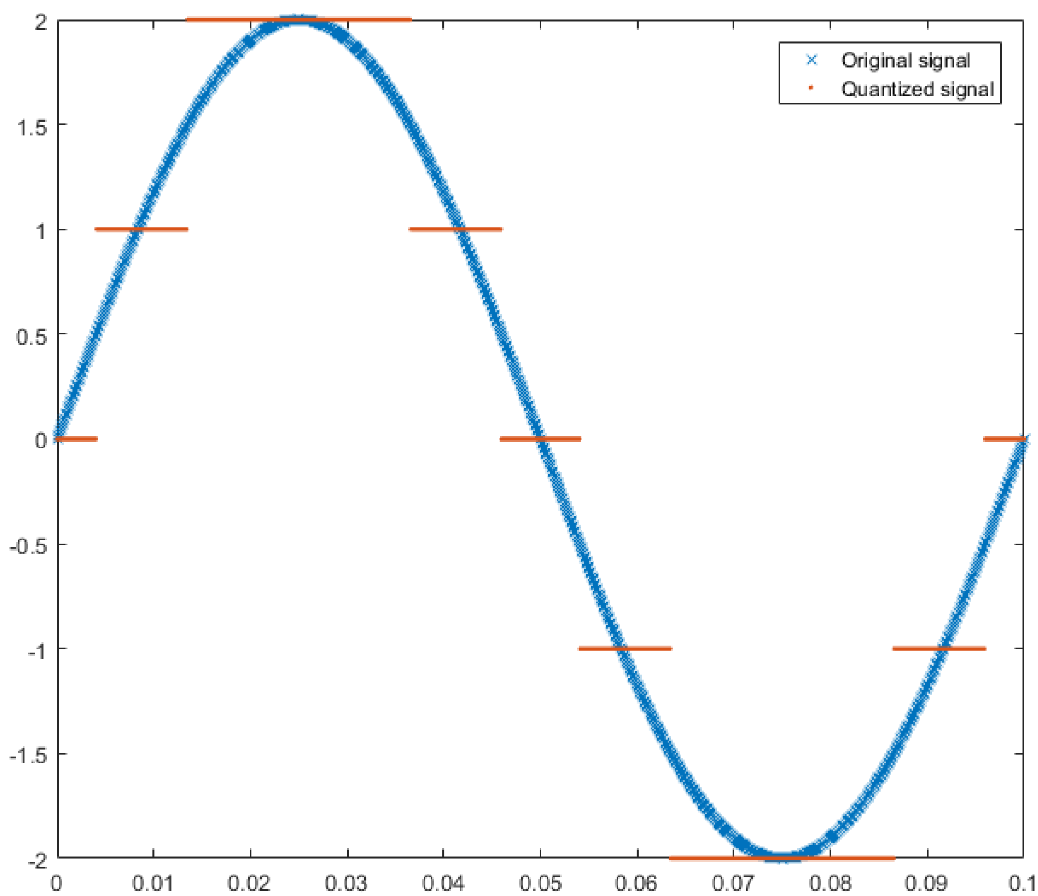

Magnitude FFT of noise
magnitude
Frequency (Hz)
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000


Time-Domain Noisy Signal
Amplitude
Time (s)
10
8
6
4
2
0
-2
-4
-6
-8
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Magnitude FFT of noisy signal
magnitude
Frequency (Hz)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000

Example of scalar quantization in MATLAB:

```
fs = 10000;
t = [0:1/fs:0.1];
f = 10; % Times at which to sample the sine function
sig = 2*sin(2*pi*f*t); % Original signal, a sine wave
partition = [-1.5, -0.5, 0.5, 1.5]; % Length 4, to represent 5
intervals
codebook = [-2:2]; % Length 5, one entry for each interval
[index,quants] = quantiz(sig,partition,codebook); % Quantize.
figure
plot(t,sig,'x',t,quants,'.')
legend('Original signal','Quantized signal');
```



<u>Software:</u>

MATLAB2016a

Performance Task for Lab Report: (your ID = AB-CDEFG-H)

**Generate a composite signal using two simple signals as,

$x_1(t) = A_1 \cos(2\pi(C*100)t)$ $x_2(t) = A_2 \cos(2\pi(F*100)t)$

$x_3(t) = x_1(t) + x_2(t)$

(a) Select the value of the amplitudes as follows: let A_1 = GD and A_2 = AF.

(b) Make a plot of x_3 over a range of t that will exhibit approximately 2 cycles. Make sure the plot starts at a negative time so that it will include t = 0, and make sure that you have at least 20 samples per period of the wave.

(c) Plot x_3 in frequency domain and calculate its bandwidth.

(d) Quantize x_3 in 6 equally distributed levels and provide image for one cycle of the original signal and quantized signal.