American International University- Bangladesh

Department of Computer Engineering

COE 3201: Data Communication Laboratory

Title: Study of signal frequency, spectrum, bandwidth, and 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.

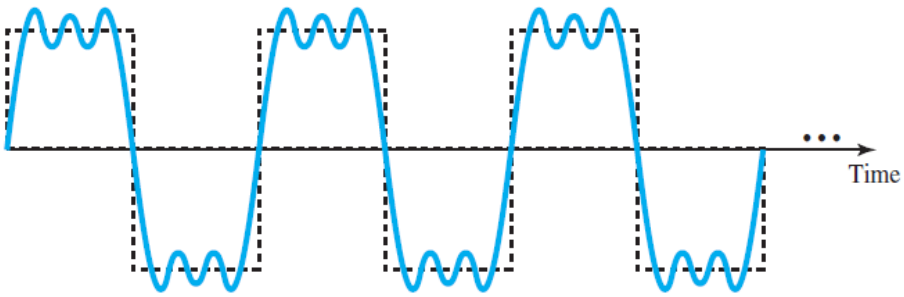


Fig: A composite periodic signal

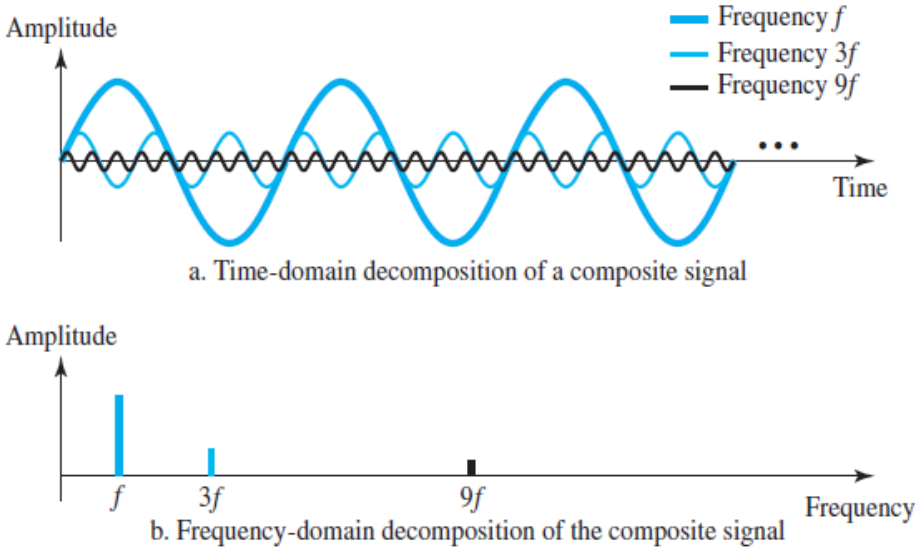
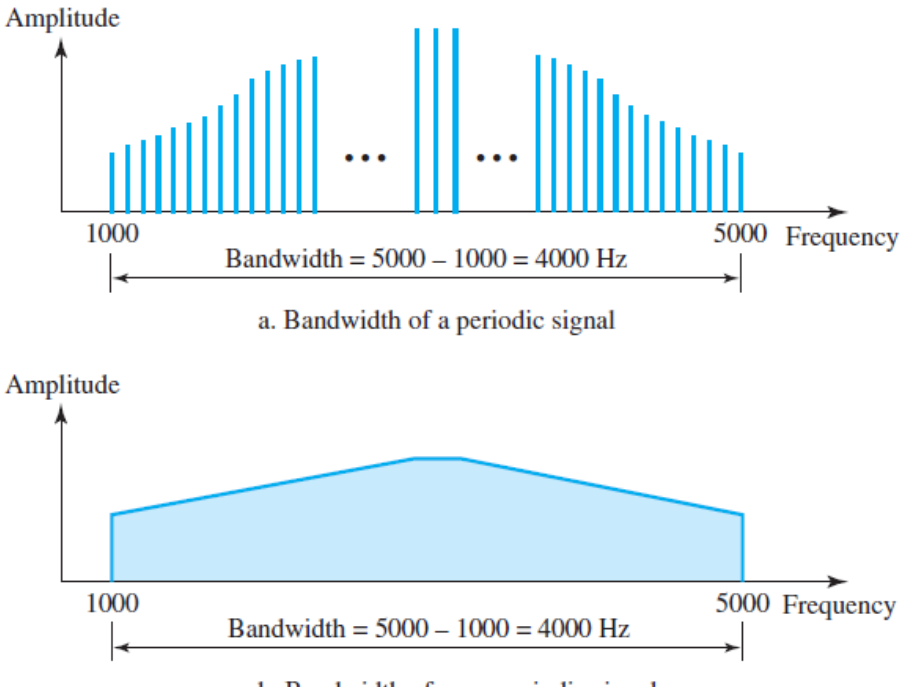


Fig: Decomposition of a composite periodic signal in the time and frequency domains

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. **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.

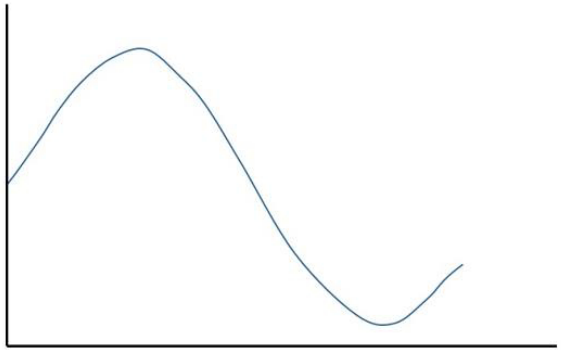


Figure: An example analog signal

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.

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.

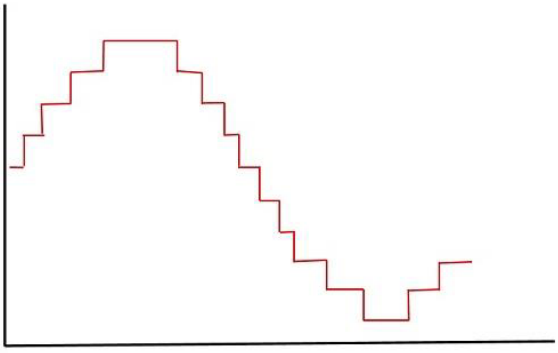


Figure: A quantized signal

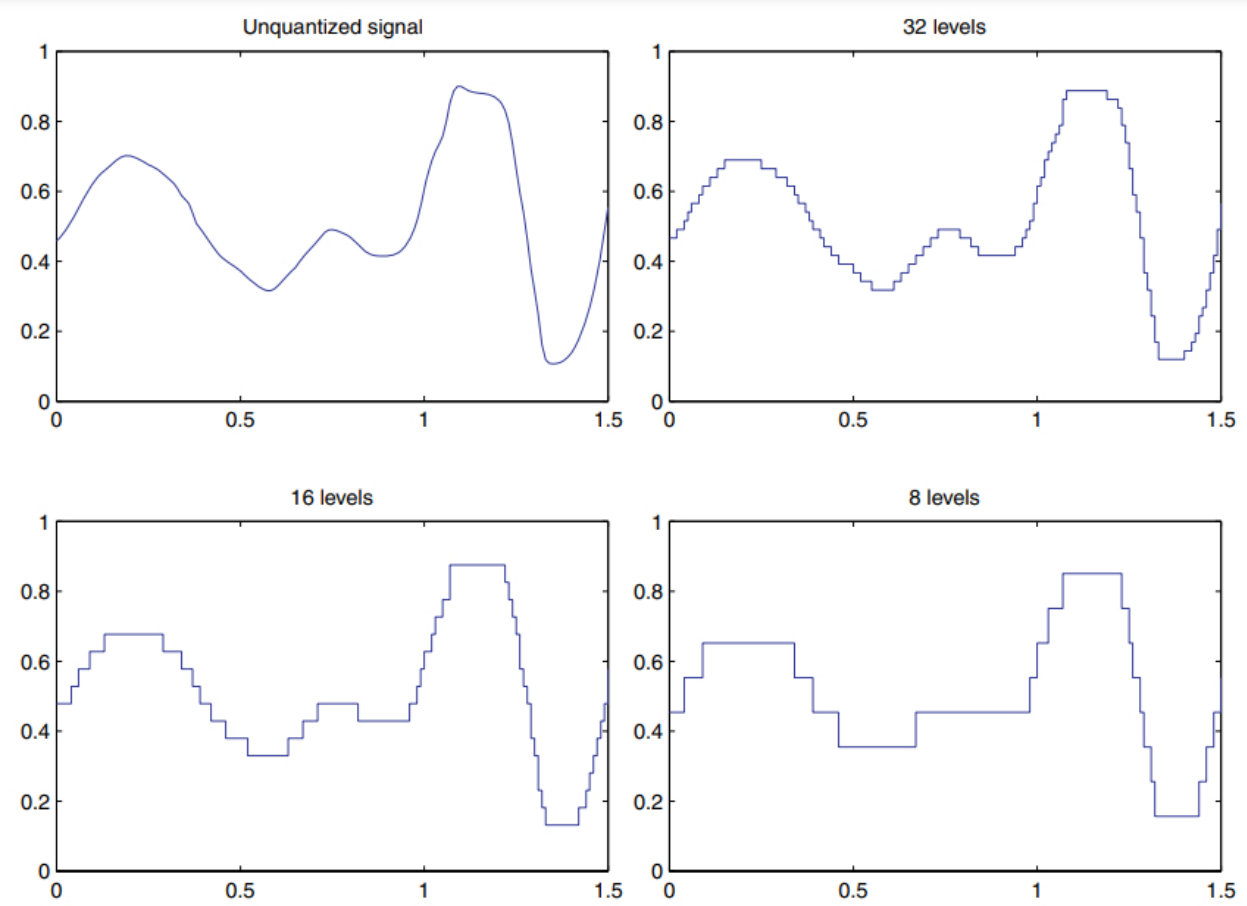


Figure : Quantized versions of an analog signal.

A simple method of quantization is given below:

$\Delta = (x_{max}-x_{min})/(L-1)$; Δ = step size

$L = 2m$; m= number of bits

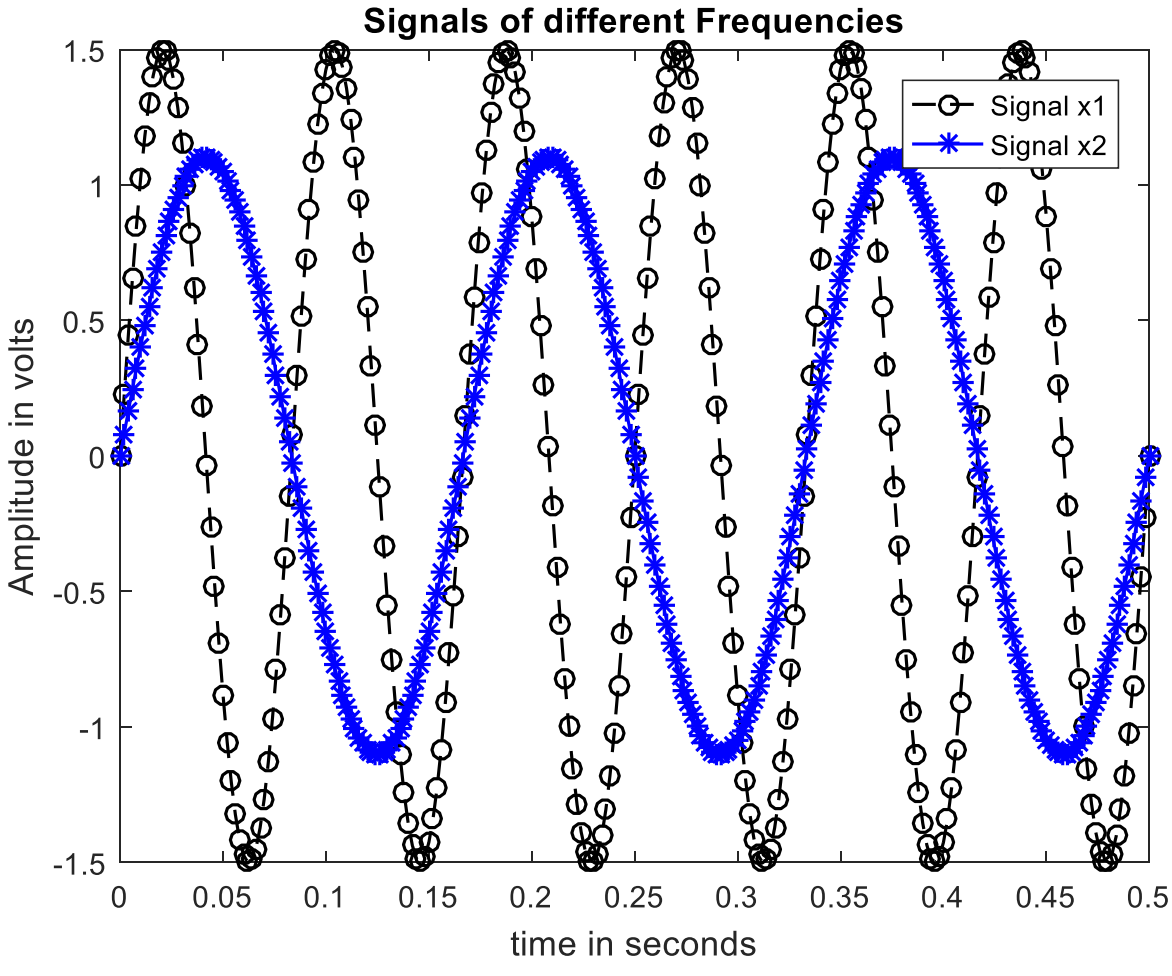
$i = \text{round}\{(x-x_{min})/\Delta\}$

$x_q = x_{min} + i*\Delta$; $i = 0, 1, \ldots, L-1$

where **xmax** and **xmin** are the maximum value and minimum values, respectively, of the analog input signal **x**. The symbol **L** denotes the number of quantization levels, where **m** is the number of bits used in ADC. The symbol **Δ** is the step size of the quantizer or the ADC resolution. Finally, **xq** indicates the quantization level, and **i** is an index corresponding to the binary code.

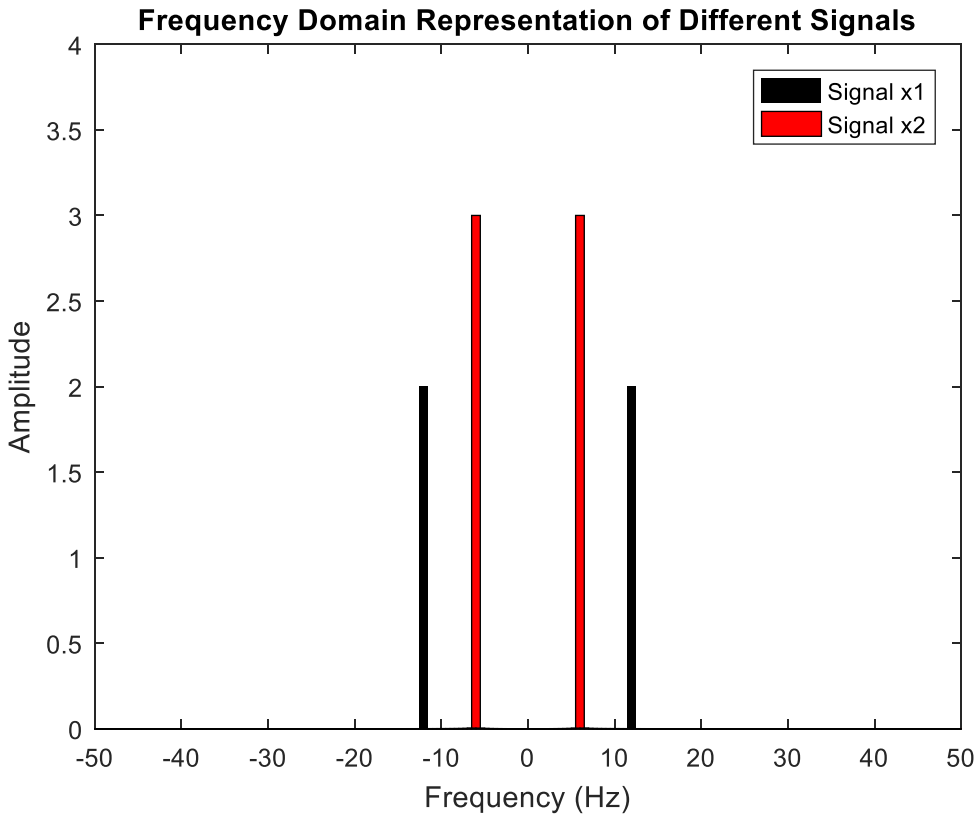
1. Generating sinusoidal signals with different frequencies:

```
clc
clear all
close all
fs = 500; % Sampling frequency
t = 0:1/fs:0.5; % 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)
hold on
plot(t,x2,'b-*','LineWidth',1)
hold off
xlabel('time in seconds')
ylabel('Amplitude in volts')
title('Signals of different Frequencies')
legend('Signal x1','Signal x2')
```



2. Signals can be represented in frequency domain as well:

```
clc
clear all
close all
fs = 5000; % Sampling frequency
t = 0:1/fs:2; % Time duration
f1 = 12; % Frequency of first signal
f2 = 6; % Frequency of second signal
A1 = 2; % Amplitude of first signal
A2 = 3; % Amplitude of second signal
x1 = A1*sin(2*pi*f1*t); % First Signal
x2 = A2*sin(2*pi*f2*t); % Second Signal
nx = length(t); % Total number of samples
%Take fourier transform
fx1 = fft(x1); % Frequency analysis is done here
fx2 = fft(x2);
% Apply fftshift to put it in the form we are used to (see
% documentation)
fx1 = fftshift(fx1)/(nx/2); % Axis correction and scaling are
% done here
fx2 = fftshift(fx2)/(nx/2);
% Next, calculate the frequency axis, which is defined by the
% sampling rate
f = linspace(-fs/2,fs/2,nx);
% fft function in Matlab returns complex numbers that has both
% frequency and phase information
% we will only plot absolute values of the
% fft transformed variables
% to see the frequency domain representations
bar(f, abs(fx1),2,'k')
hold on
bar(f, abs(fx2),2,'r')
hold off
axis([-50 50 0 4])
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Domain Representation of Different Signals');
legend('Signal x1','Signal x2')
```



Class Task:

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

```
signal_x = a1*sin(2*pi*f1*t) + a2*cos(2*pi*f2*t);
```

Here, a1 = (B + G + H), a2 = (C + E + H), f1 = (G + H + 2), and f2 = (E + F+ H). [Assume your ID is AB-CDEFG-H]

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

3. 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 =

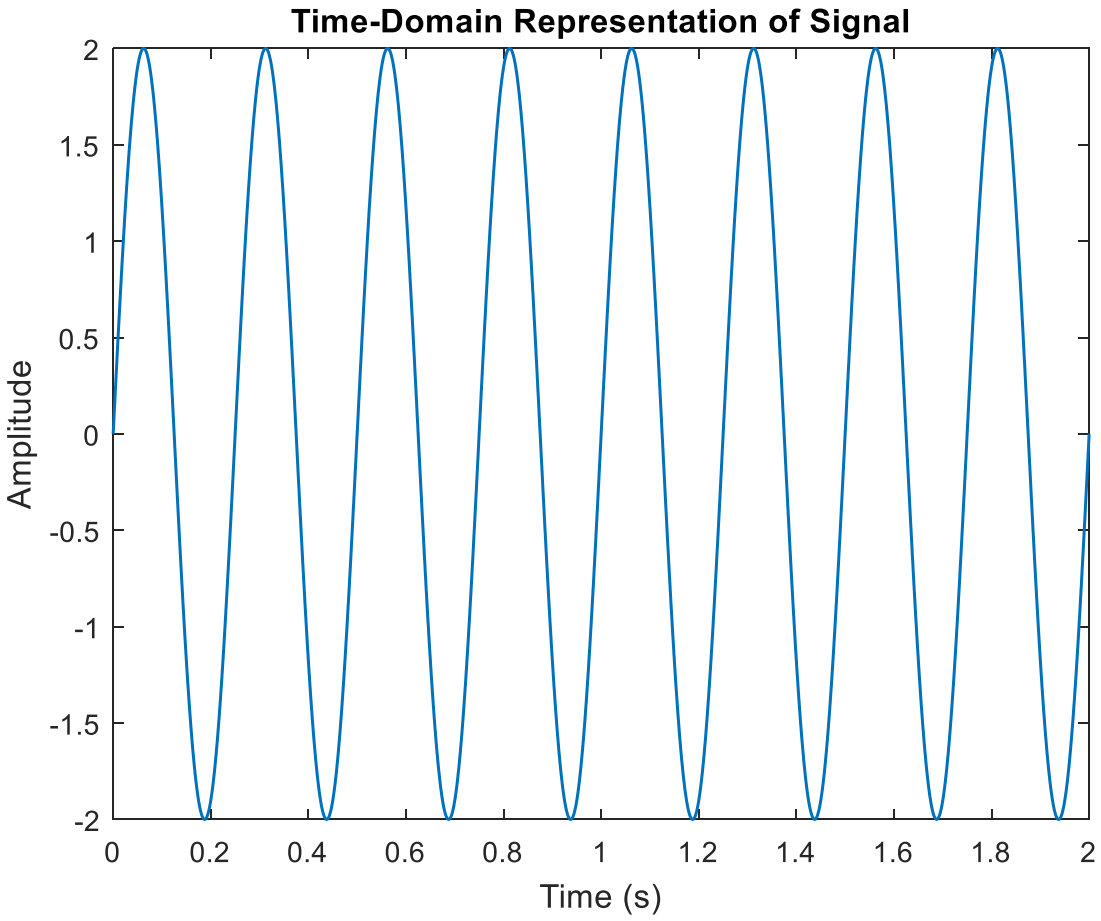
1.9010e+03

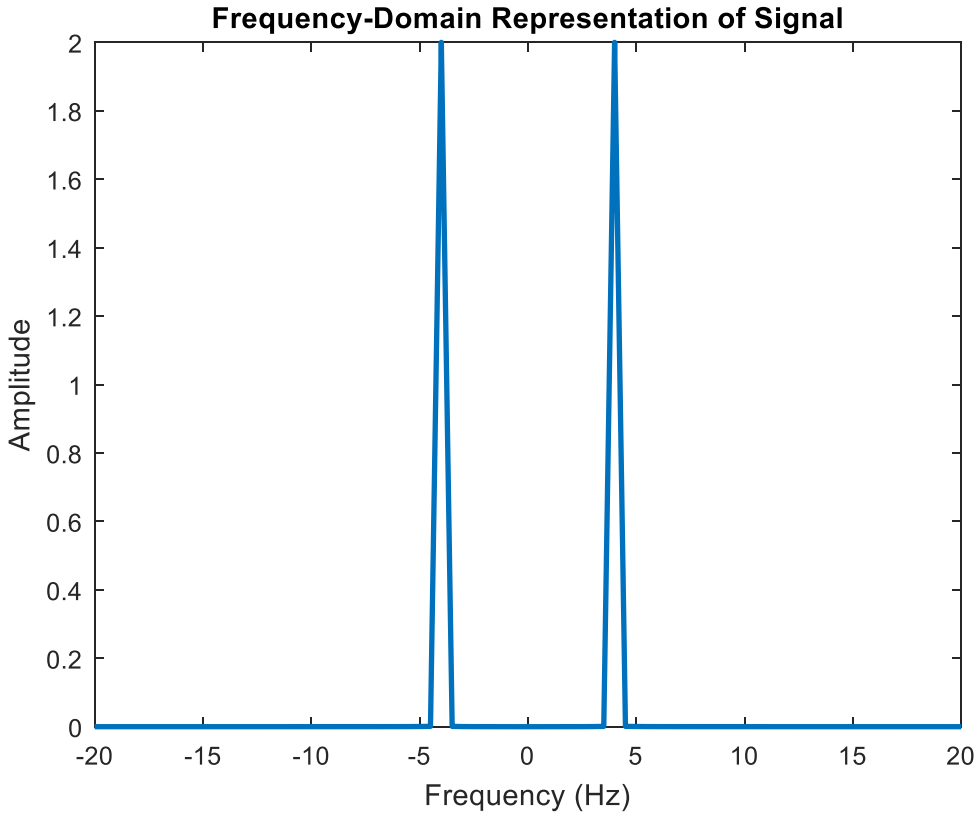
4. Example showing signal processing (noise reduction) is convenient in frequency domain

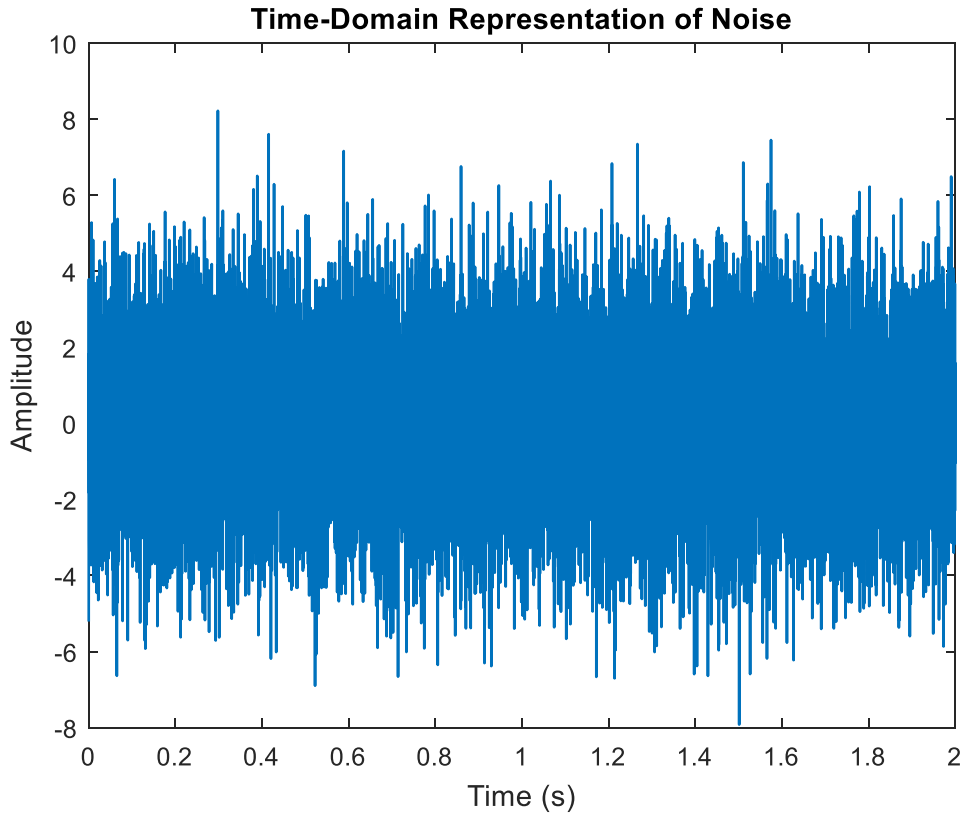
```
close all;
clc;
%Define number of samples to take
fs = 8000;
f = 4; %Hz
%Define signal
t = 0:1/fs:2;
signal = 2*sin(2*pi*f*t);
nx = length(t); % Total number of samples
%Plot to illustrate that it is a sine wave
plot(t, signal,'linewidth',1);
title('Time-Domain Representation of Signal');
xlabel('Time (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)/(nx/2);
% Scaling done by dividing with (fs/2)
% Next, calculate the frequency axis,
% which is defined by the sampling rate
f = linspace(-fs/2,fs/2,nx);
% 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),'linewidth',2);
title('Frequency-Domain Representation of Signal');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
xlim([-20 20])
%noise
sd = 2;
noise = sd*randn(size(signal)); % noise power = sd^2
figure
plot(t,noise, 'linewidth', 1)
xlabel('Time (s)');
ylabel('Amplitude');
title('Time-Domain Representation of Noise');
fftNoise = fft(noise);
fftNoise = fftshift(fftNoise)/(nx/2);
figure
plot(f,abs(fftNoise), 'linewidth', 2)
title('Frequency-Domain Representation of Noise');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
xlim([-20 20])
%noisy signal
```

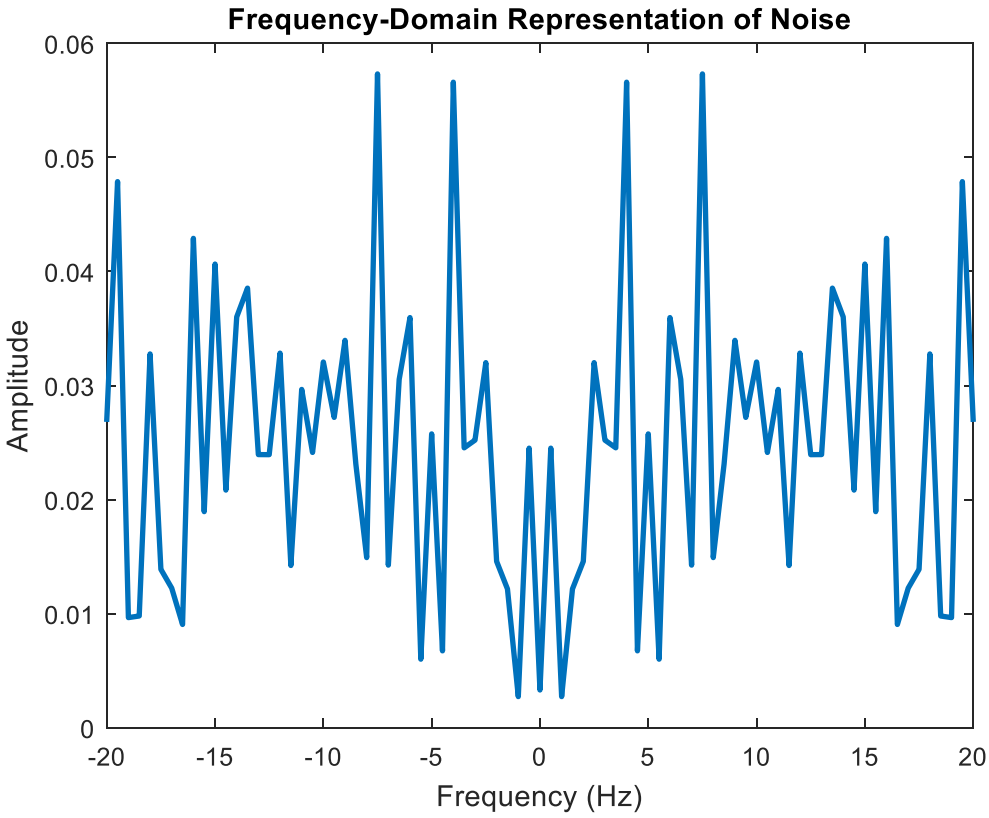
```
noisySignal = signal + noise;
figure
plot(t,noisySignal, 'linewidth', 1)
xlabel('Time (s)');
ylabel('Amplitude');
title('Time-Domain Representation of Noisy Signal');
fftNoisySignal = fft(noisySignal);
fftNoisySignal = fftshift(fftNoisySignal)/(nx/2);
figure
plot(f,abs(fftNoisySignal), 'linewidth', 2)
title('Frequency-Domain Representation of Noisy Signal');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
xlim([-20 20])
```

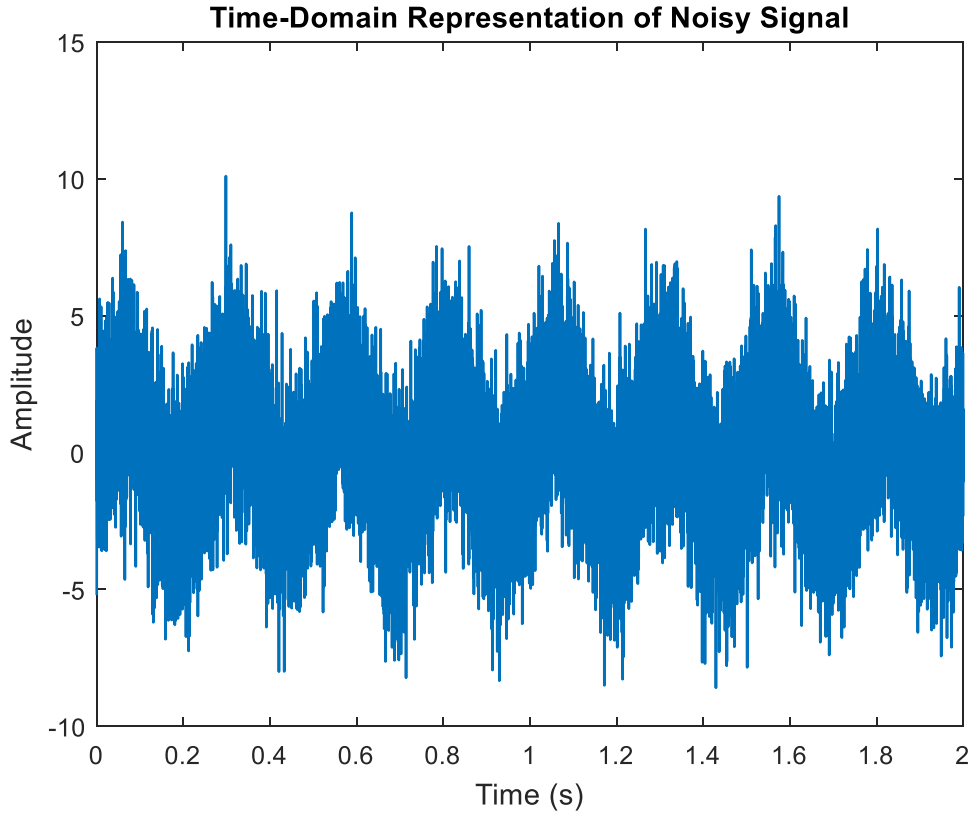
Generated figures from this example:

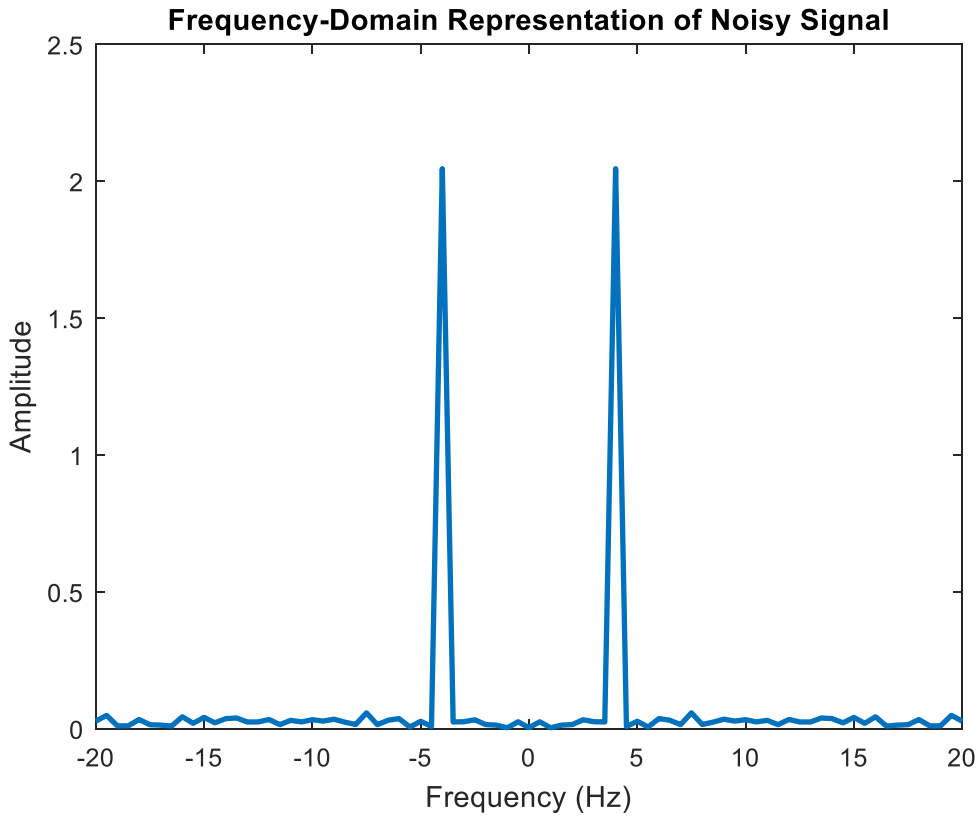






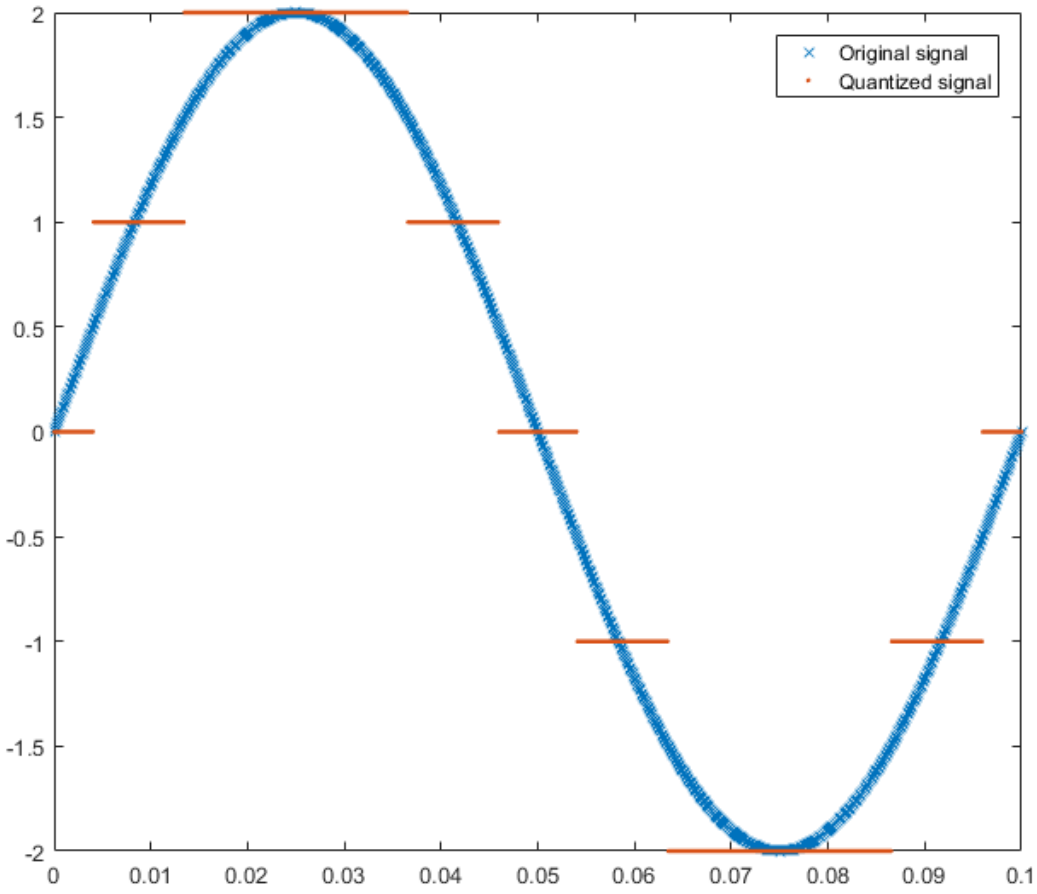






5. Example of Quantization using Matlab bulit-in function '**quantiz**'

```
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 = -1.5:1.5; % 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');
```



6. Example of Quantization **NOT** using Matlab bulit-in function

```
clc
close all
fs = 40e3;% sampling frequency
f = 50;% frequency of the signal
t = 0:1/fs:1/f;%discrete time
A = 2;
x = A*sin(2*pi*f*t);% discrete signal
%--------Quantization-----------%
n = 3;
L =(2^n);
delta=(max(x)-min(x))/(L-1);
xq = min(x)+(round((x-min(x))/delta)).*delta;
%--------END-----------%
plot(t,x,'r-.', 'linewidth',1.5);
hold on;
plot(t,xq,'k-.', 'linewidth',1.5);
%plotting wave forms.
xlabel('time')
ylabel('amplitude')
title('example of manual quantization')
legend('Original signal','quantized signal')
```

Software:

MATLAB2016a

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

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

x1 = a1*cos(2*pi*f1*t), x2 = a2*sin(2*pi*f2*t), x3 = a3*cos(2*pi*f3*t)

signal_x = x1 + x2 + x3

Select the values of the amplitude and frequency as follows: a1 = G + 1, a2 = F + 2, a3 = E + 3, f1 = E + 1, f2 = F + 2, f3 = G + 3.

(a) Show time domain and frequency domain representations of **signal_x** in a single figure window using subplot. Use **axis**, or **xlim**, or **ylim** to appropriately represent the signal.

(b) Quantize **signal_x** in 4 equally distributed levels and provide image for **one cycle** of the original signal and quantized signal. Use **axis**, or **xlim**, or **ylim** to appropriately represent the signal. [Use **quantiz()** function]

(c) Quantize **signal_x** in 8 equally distributed levels and provide image for **one cycle** of the original signal and quantized signal. Use **axis**, or **xlim**, or **ylim** to appropriately represent the signal. [Do not use **quantiz()** function]