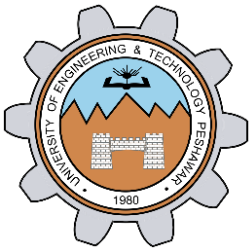
SAMPLING AN ANALOG SIGNAL

LAB # 06



CSE402L Digital Signal Processing Lab

Submitted by: **Shah Raza**

Registration No: **18PWCSE1658**

Class Section: **B**

"On my honor, as a student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature: _____

Submitted to: **Engr. Faiz Ullah**

Tuesday, January 26th, 2021

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Lab Objectives:

Objectives of this lab are as follows:

- Learn about sampling an analog signal

Practical:

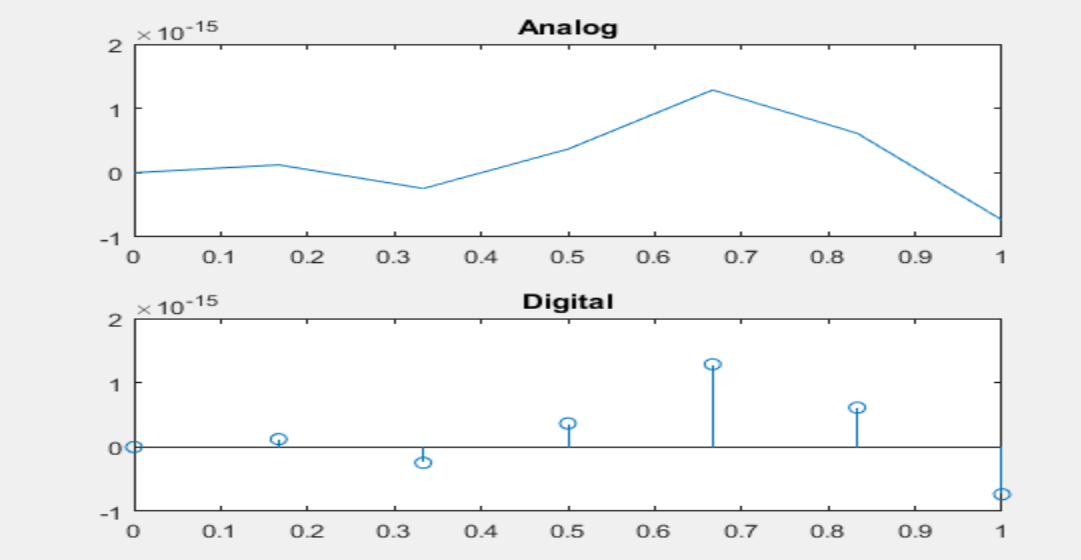
Generate various waveforms like sinusoidal, triangular, damped oscillations and unit step using different sampling frequencies. Plot the waveforms on time axis. Increase and decrease the sampling frequency and note the effect on the representation of waveforms in digital domain.

Code:

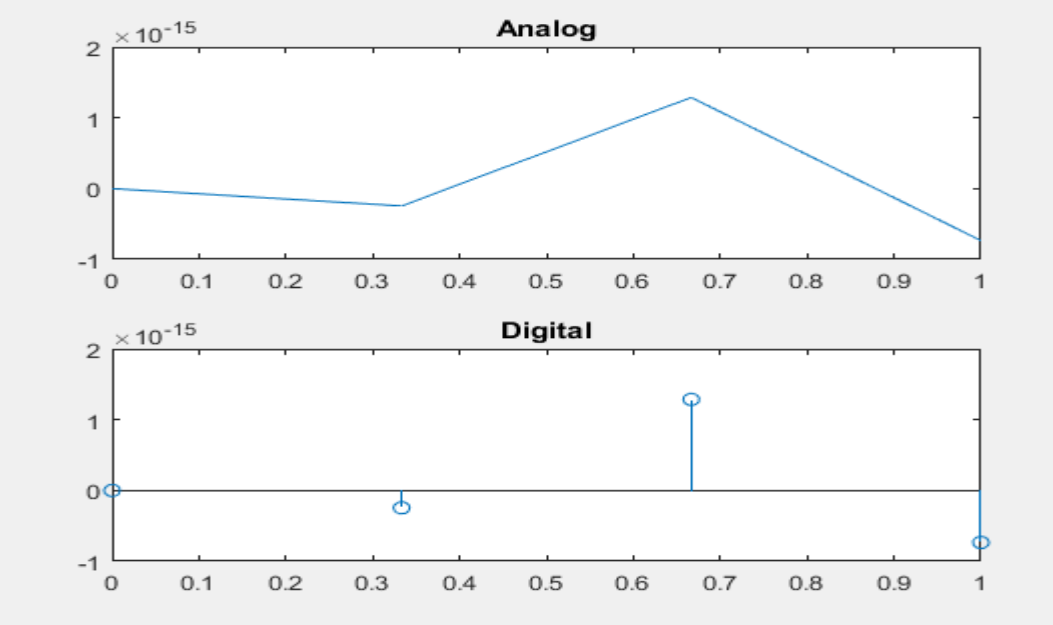
```
clc
clear all
close all
f = 3;
fs = f*2;      %Nyquist Criterion
t = 0:1/fs:1;
x = sin(2*pi*f*t);
subplot(2,1,1)
plot(t,x);
title('Analog');
subplot(2,1,2)
stem(t,x);
title('Digital');
```

Output:

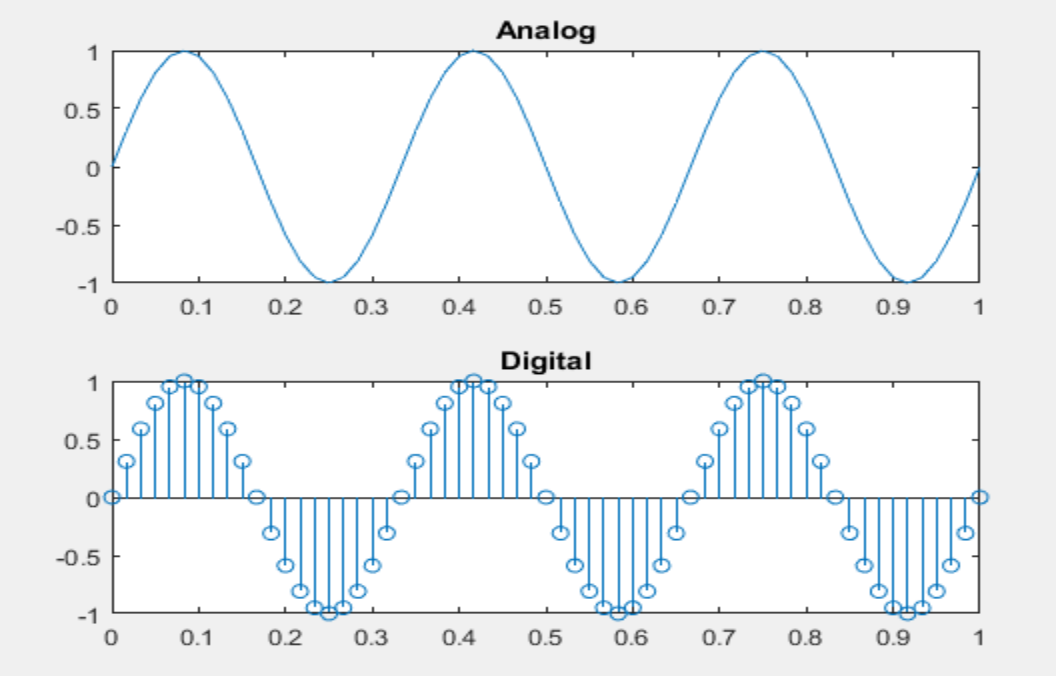
When fs= 2*f:



When fs<2*f:



When fs>2*f:



Discussion:

We can see from the above plots that as we increase the sampling frequency, the sampled form gets nearer to the original signal. And if fs<2*f then we cannot form the original signal.

Questions:

1. **Does the increase in sampling frequency fs makes the digitized waveform a better representation of the analog waveform? If so, why in practice we keep fs just above twice the maximum frequency?**

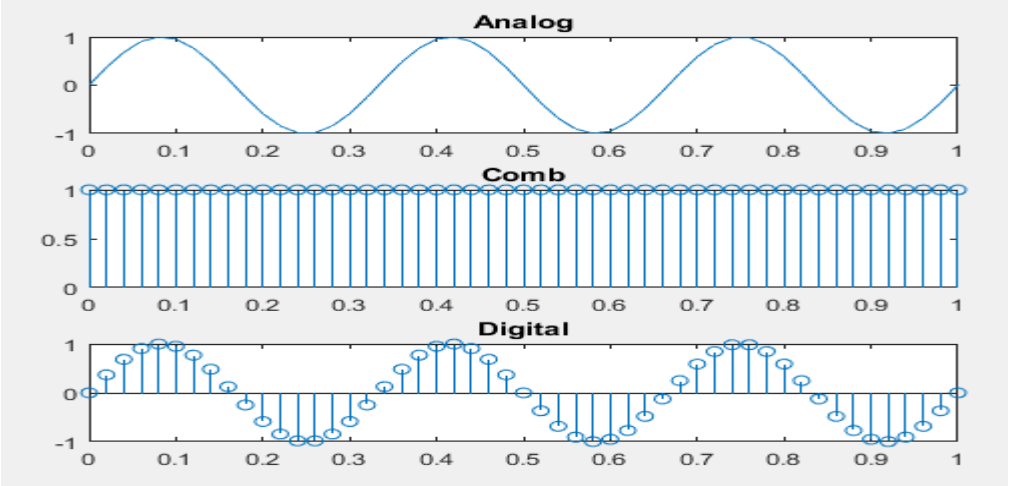
   Answer:
   Yes, the increase in sampling frequency (fs) makes the digitized waveform a better representation of the analog waveform. But in practice we keep fs just above twice the maximum frequency because if we take higher samples then it will require more computation and we certainly don't want that.

2. **Obtain the above results using your own script file. Generate analog and comb functions separately. Multiply the generated analog signal with the comb function, and display the results.**

   Code:

```
clc
clear all
close all
f = 3;
t = 0:1/50:1;
x = sin(2*pi*f*t);
comb = ones(1,51);
Digital = x.*comb;
subplot(3,1,1)
plot(t,x);
title('Analog');
subplot(3,1,2)
stem(t,comb);
title('Comb');
subplot(3,1,3)
stem(t,Digital)
title('Digital');
```

   Output:



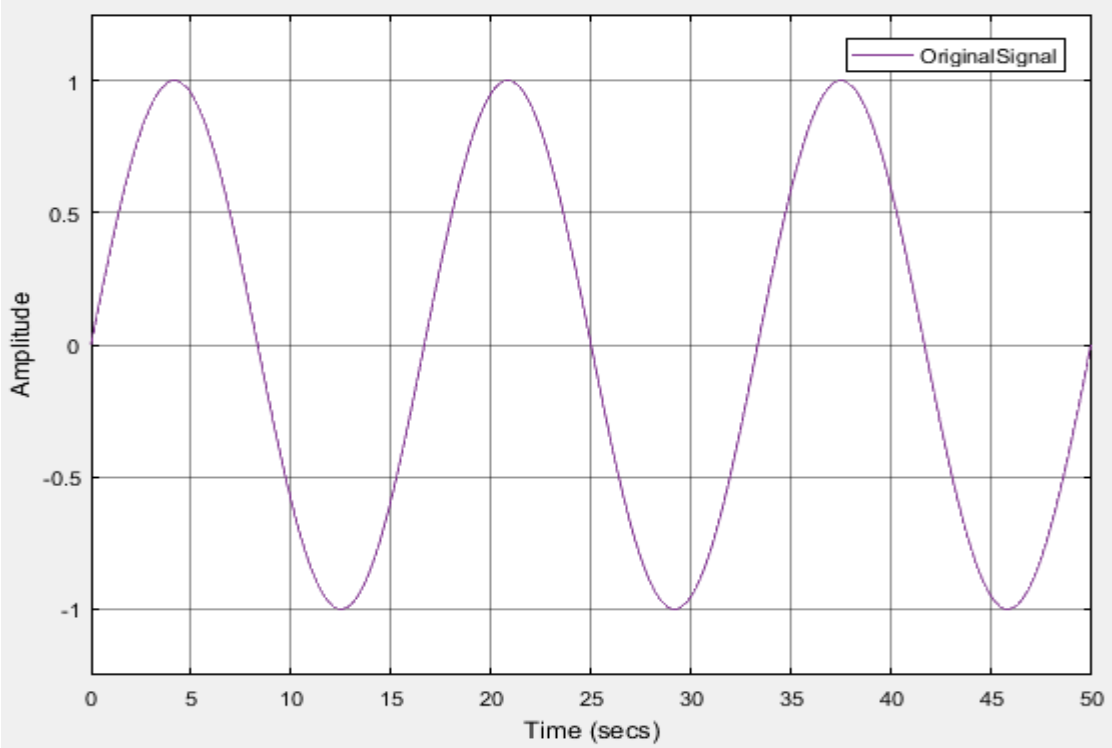
3. **Using MATLAB, generate a sinusoid and then simulate the effect of noise by adding normally distributed random numbers. Save the sinusoid and the noisy output to *.mat files. View the sinusoid and the noisy output via sptool. Hear the two signals and get a grasp of the difference.**

Code:

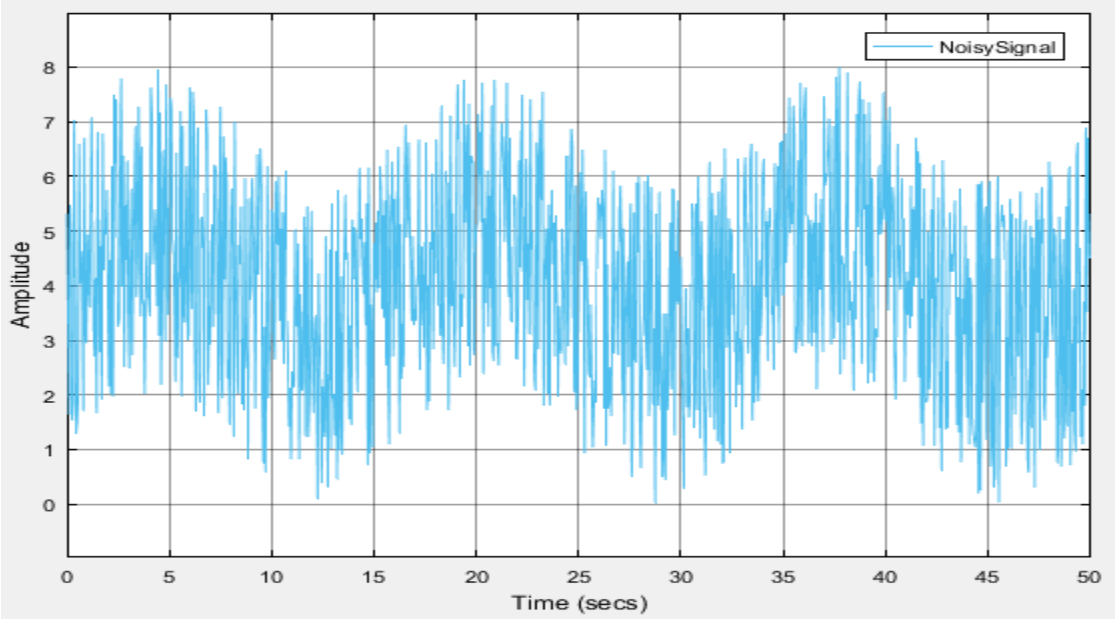
```
clc
clear all
close all
f = 3;
t = 0:1/1000:1;
x = sin(2*pi*f*t);
noise = 1+(7-1)*rand(1,1001);
NoisySignal = x+noise;
save Sinosoid x
save NoisySin NoisySignal
```

Output:

Original Signal:



Noisy Signal:



4. Using a sinusoid of 1KHz, sampled at 48KHz, generate 10KHz sinusoid form this signal in MATLAB. Hint: you need to ignore some of the samples!

Code:

```
clc
clear all
close all
f = 1000;
fs = 10*f;
t = 0:1/fs:1;
x = sin(2*pi*f*t);
stem(t,x);
axis([0 0.02 -2 2]);
title('Sampled Signal');
```

Output: