Digital Signal Processing Lab

Expt. No. 1

Sampling



By Pranit Dalal

Roll no. 16EC10016

Group 22 (Tuesday)

**AIM:**

(a)Sampling of a sinusoidal waveform

(b)Sampling at below Nyquist rate and effect of aliasing

(c)Spectrum of a square wave

(d)Interpolation or upsampling

**(a)Sampling of a sinusoidal waveform:**

(i)Theory:

Sampling at Nyquist rate: It refers to the sampling rate being greater than or equal to twice the bandwidth of a bandlimited function or a bandlimited channel. Suppose the highest frequency component, in hertz, for a given analog signal is fmax.

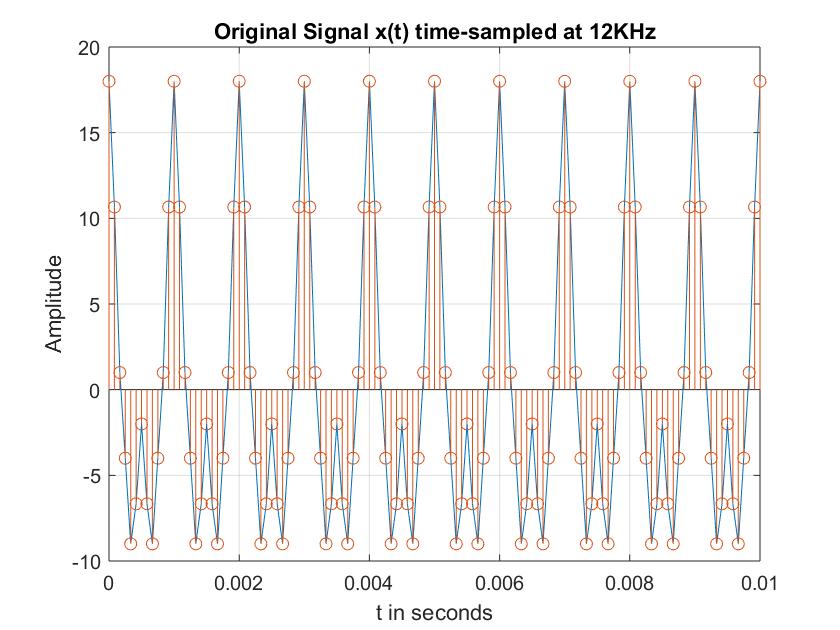
According to the Nyquist Theorem, the sampling rate must be at least 2\*fmax, or twice the highest analog frequency component. The sampling in an analog-to-digital converter is actuated by a pulse generator (clock). If the sampling rate is less than 2\*fmax, some of the highest frequency components in the analog input signal will not be correctly represented in the digitized output. When such a digital signal is converted back to analog form by a digital-to analog converter, false frequency components appear that were not in the original analog signal. This undesirable condition is a form of distortion called Aliasing. Thus, the condition to avoid aliasing is:

fs ≥ 2fmax

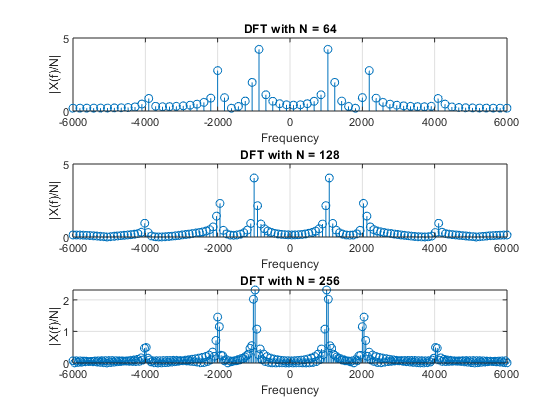
Where, fs=Sampling frequency; fmax=Highest frequency present in a signal, or bandwidth.

(ii)Original signal sampled at 12 KHz

Fs = 12000;  
t = 0:1/Fs:0.01;  
x = 10\*cos(2\*pi\*1e3\*t) + 6\*cos(2\*pi\*2e3\*t) + 2\*cos(2\*pi\*4e3\*t);  
stem(t,x)  
grid on  
title('Original Signal x(t) time-sampled at 12KHz')  
xlabel('t in seconds')  
ylabel('Amplitude')



(iii)Discrete Fourier Transform at different N

Fs = 12000;  
t = 0:1/Fs:0.01;  
N=64;  
x = 10\*cos(2\*pi\*1e3\*t) + 6\*cos(2\*pi\*2e3\*t) + 2\*cos(2\*pi\*4e3\*t);  
y=fft(x,N);  
y=fftshift(y);  
m = abs(y)/N;  
f = -6000:12000/(N-1):6000;  
subplot(311)  
stem(f,m)  
grid on  
title('DFT with N = 64')  
xlabel('Frequency')  
ylabel('|X(f)/N|')  
  
N1 = 128;  
x = 10\*cos(2\*pi\*1e3\*t) + 6\*cos(2\*pi\*2e3\*t) + 2\*cos(2\*pi\*4e3\*t);  
y1=fft(x,N1);  
y1=fftshift(y1);  
m1 = abs(y1)/N1;  
f = -6000:12000/(N1-1):6000;  
subplot(312)  
stem(f,m1)  
grid on  
title('DFT with N = 128')  
xlabel('Frequency')  
ylabel('|X(f)/N|')  
  
N2 = 256;  
x = 10\*cos(2\*pi\*1e3\*t) + 6\*cos(2\*pi\*2e3\*t) + 2\*cos(2\*pi\*4e3\*t);  
y2=fft(x,N2);  
y2=fftshift(y2);  
m2 = abs(y2)/N2;  
f = -6000:12000/(N2-1):6000;  
subplot(313)  
stem(f,m2)  
grid on  
title('DFT with N = 256')  
xlabel('Frequency')  
ylabel('|X(f)/N|')

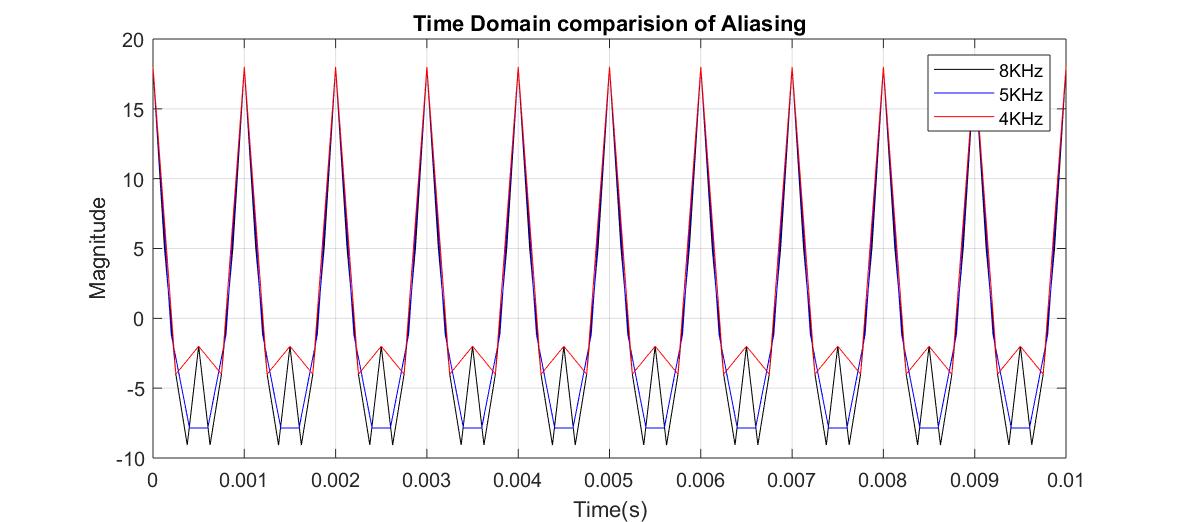
**(b)Sampling at below Nyquist rate and effect of aliasing**

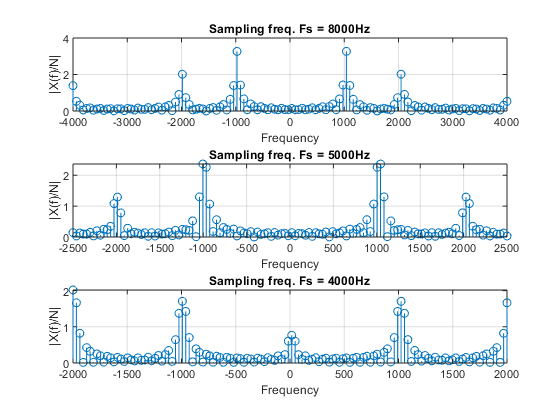
(i)Theory:

In signal processing and related disciplines, aliasing is an effect that causes different signals to become indistinguishable (or aliases of one another) when sampled. In audio, aliasing is the result of a lower resolution sampling, which translates to poor sound quality and static. This occurs when audio is sampled at a lower resolution than the original recording.

(ii)Sampling at Fs = 8 KHz, 5 KHz, 4 KHz

Fs1 = 8000;  
N=128;  
t = 0:1/Fs1:0.01;  
x = 10\*cos(2\*pi\*1e3\*t) + 6\*cos(2\*pi\*2e3\*t) + 2\*cos(2\*pi\*4e3\*t);  
%plot(t,x,'k')  
%hold on  
  
y=fft(x,N);  
y=fftshift(y);  
m = abs(y)/N;  
f = -Fs1/2:Fs1/(N-1):Fs1/2;  
subplot(311);  
stem(f,m);  
grid on  
title('Sampling freq. Fs = 8000Hz')  
xlabel('Frequency')  
ylabel('|X(f)/N|')  
  
  
Fs2 = 5000;  
t = 0:1/Fs2:0.01;  
x = 10\*cos(2\*pi\*1e3\*t) + 6\*cos(2\*pi\*2e3\*t) + 2\*cos(2\*pi\*4e3\*t);  
%plot(t,x,'b')  
%hold on  
  
y1=fft(x,N);  
y1=fftshift(y1);  
m = abs(y1)/N;  
f = -Fs2/2:Fs2/(N-1):Fs2/2;  
subplot(312);  
stem(f,m);  
grid on  
title('Sampling freq. Fs = 5000Hz')  
xlabel('Frequency')  
ylabel('|X(f)/N|')  
  
  
Fs3 = 4000;  
t = 0:1/Fs3:0.01;  
x = 10\*cos(2\*pi\*1e3\*t) + 6\*cos(2\*pi\*2e3\*t) + 2\*cos(2\*pi\*4e3\*t);  
%plot(t,x,'r')  
%hold on  
  
y2=fft(x,N);  
y2=fftshift(y2);  
m = abs(y2)/N;  
f = -Fs3/2:Fs3/(N-1):Fs3/2;  
subplot(313);  
stem(f,m);  
grid on  
title('Sampling freq. Fs = 4000Hz')  
xlabel('Frequency')  
ylabel('|X(f)/N|')





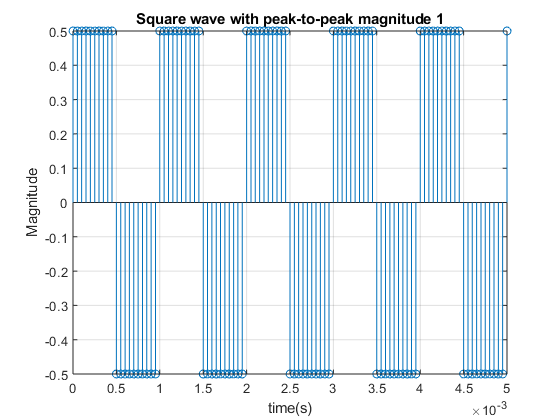
**(c)Spectrum of a square wave**

(i)Theory

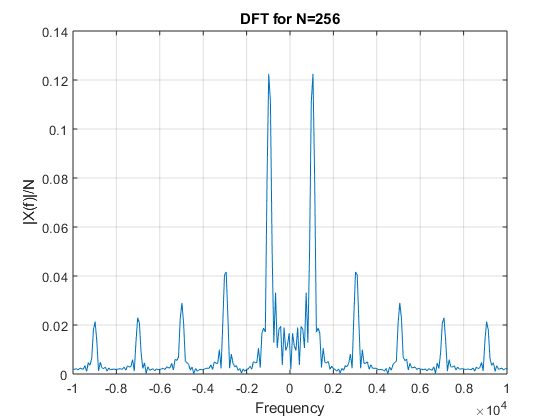
The ideal square wave contains only components of odd-integer harmonic frequencies, and it has an infinite bandwidth. A square wave has ideally infinite bandwidth. For practical purposes, the spectrum beyond 10th harmonic can be neglected.

(ii)Generation of a square wave

Sampling frequency = 20 KHz, Frequency of square wave = 1 KHz

F = 1000;  
Fs = 20000;  
T = 0:1/Fs:5/1000;  
x=0.5\*(square(2\*pi\*F\*T));  
stem(T,x);  
title('Square wave with peak-to-peak magnitude 1')  
xlabel('time(s)')  
ylabel('Magnitude')  
grid on

(iii)Taking DFT of sampled square wave with N=256

F = 1000;  
Fs = 20000;  
T = 0:1/Fs:5/1000;  
x=0.5\*(square(2\*pi\*F\*T));  
n=256;  
y=fft(x,n);  
y=fftshift(y);  
m = abs(y)/n;  
f1 = -Fs/2:Fs/(n-1):Fs/2;  
plot(f1,m)  
title('DFT for N=256')  
xlabel('Frequency')  
ylabel('|X(f)|/N')  
grid on

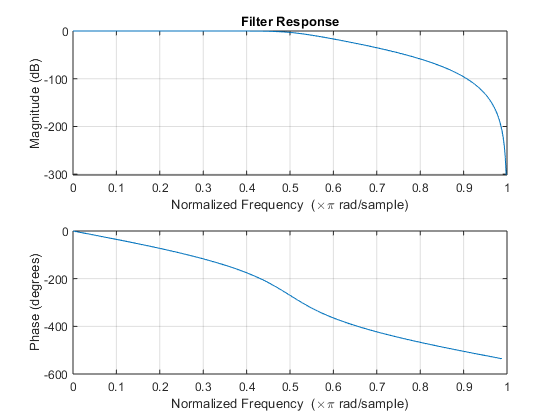
**(d)Interpolation and upsampling**

(i)Theory:

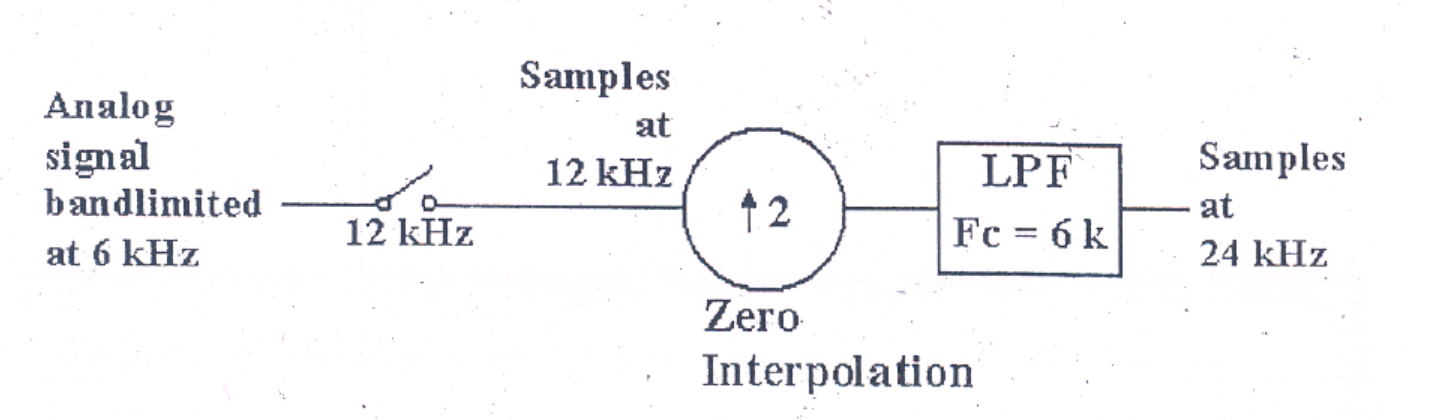
If an analog signal is sampled at a frequency higher than the Nyquist rate (Fs>2Fmax) it is possible to interpolate the intermediate L-1 samples or in other words to obtain the samples at Fs2=LFs1 frequency. This can be simply done by passing the sampled signals through an ideal low pass filter of cut-off frequency F max and sampling it again at a higher rate. But in discrete domain this is achieved. “Interpolation”, is the process of upsampling followed by filtering. The filtering removes the undesired spectral images. “Upsampling” is the process of inserting zero-valued samples between original samples to increase the sampling rate. This is called “zero-stuffing”.

(ii)Butterworth Filter Response

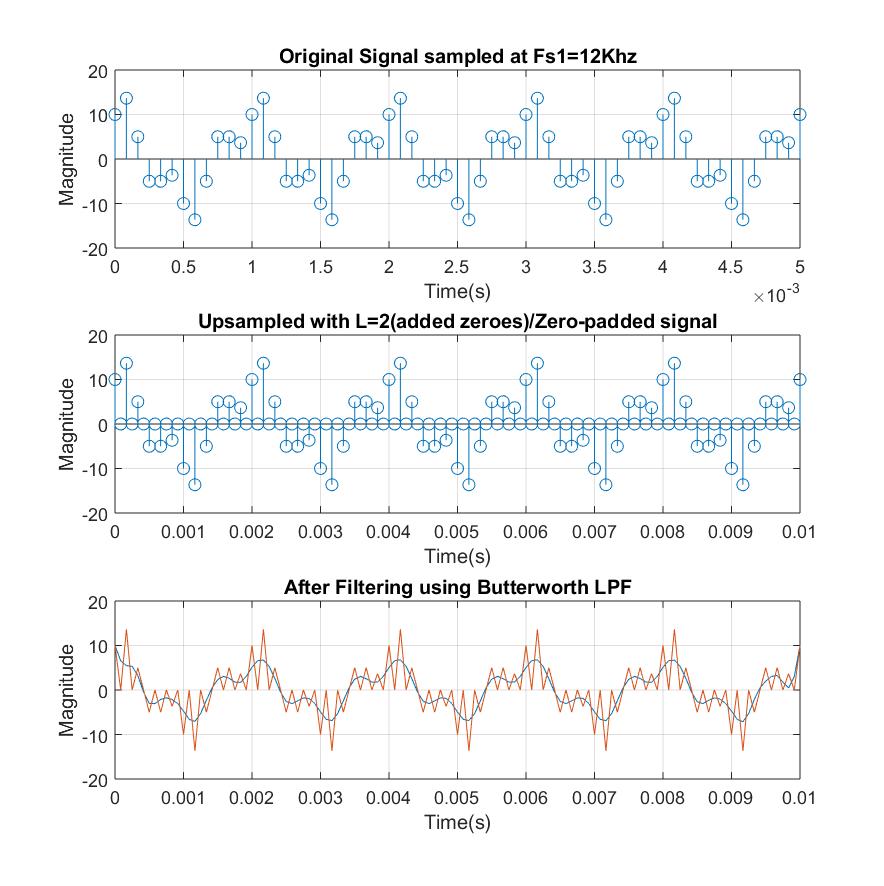
figure  
%Implementing a butterworth Filter on matlab.  
[b,a] = butter(6,(6000/(24000/2)), 'low' ); %sixth order butterworth filter  
freqz(b,a);  
title( 'Filter Response' );



(iii)Upsampling steps



Fs1 = 12000;  
t1 = 0:1/Fs1:0.005;  
x1 = 10\*cos(2\*pi\*1e3\*t1) + 5\*sin(2\*pi\*3e3\*t1) ;  
subplot(311);  
stem(t1,x1);  
grid on  
title('Original Signal sampled at Fs1=12Khz');  
xlabel('Time(s)')  
ylabel('Magnitude')  
  
t2 = 0:1/Fs1:0.01;  
x2 = upsample(x1,2); %L = 2  
x2 = x2(1:end-1);  
subplot(312);  
stem(t2,x2);  
grid on  
title('Upsampled with L=2(added zeroes)/Zero-padded signal')  
xlabel('Time(s)')  
ylabel('Magnitude')  
  
Fs1 = 12000;  
t1 = 0:1/Fs1:0.005;  
x1 = 10\*cos(2\*pi\*1e3\*t1) + 5\*sin(2\*pi\*3e3\*t1) ;  
  
t2 = 0:1/Fs1:0.01;  
x2 = upsample(x1,2); %L = 2  
x2 = x2(1:end-1);  
  
Fs2 = 24000;  
t = 0:1/Fs2:0.01;  
n=33;  
fc=6000;  
Wn = fc/Fs1; %Normalised cutoff freq  
  
[b,a] = butter(n,Wn);  
y2 = filter(b,a,x2);  
y3 = filtfilt(b,a,x2);  
subplot(313);  
grid on  
plot(t2,y3,t2,x2);  
title('After Filtering using Butterworth LPF')  
grid on  
xlabel('Time(s)')  
ylabel('Magnitude')



**Discussions:**

(a)  
i. The sampling frequency is above Nyquist Rate and hence proper DFT is obtainable.  
ii. It can be clearly observed that the Discrete Fourier Transform is more prominent at the constituent frequencies.  
iii. As N increases the sharpness at constituent frequencies increase and hence larger value of N is preferable to get better understanding of the composing signal frequencies.

(b)  
i. When sampled at frequencies lower than the Nyquist Rate we observe aliasing.  
ii. It can be clearly seen when Fs = 4000Hz. The 4 kHz present in the original signal can be seen as a DC component (Frequency = 0).  
iii. The signal can’t be reconstructed from this sampled set as we unwanted frequency components are present and the reconstructed signal would differ a lot from the original signal.

(c)  
i. Square wave is composed of odd-integer harmonic frequencies and has infinite bandwidth.  
ii. The DFT shows peaks at the odd multiples of 1 KHz only. The amplitude decreases with increase in harmonics.  
iii. Higher harmonics are neglected for practical purposes as their amplitudes is negligible.

(d)  
i. To regenerate the sampled signal we upsample it, add zeroes and then interpolate it. Finally it is passed through LPF.  
ii. We have used a Butterworth filter here to generate the upsampled signal. We see that the Filtered signal do not exactly matches with the original upsampled signal due to scaling variations and phase shift (phase shift was resolved by filtfilt).  
iii. We can obtain the interpolated signal by removing the high frequency content by passing it through a digital LPF of cut-off π/L.

**References:**

-Alan V. Oppenheim et.al Discrete Time Signal Processing.

-Wikipedia.

[*Published with MATLAB® R2018a*](https://www.mathworks.com/products/matlab)