CT303 Lab Report **LAB-4/5**
Date-November 7,2022
Name-Divya Kirtikumar Patel
Student ID-202001420
Lab Group- 6

LAB - 4

1). From [1], perform experiment 1 to study the three methods of sampling: Natural, Sample and hold, and Flat top. You need to follow the procedures given in the manual and observe the output as suggested in the observations. You are then required to study the output with the switch faults.

Experiment 1: Analog signal to sampling reconstruction

Objective: To study different type of sampling signal and do reconstruction of that signal

1). Natural sampling

2). Sample and Hold

3). Flat Top sampling

2). From [1], perform experiment 2 to study effect of different sampling frequencies on a signal. You need to follow the procedures given in the manual and observe the output as suggested in the observations. You are then required to study the output with the switch faults.

Experiment 2: To study effect of different sampling frequencies.

Objective: To study the effect of different sampling frequencies on the reconstruction signal.

3). From [1], perform experiment 4 to study the effect of the order of the filter on the reconstructed signal. You need to follow the procedures given in the manual and observe the output as suggested in the observations. You are then required to study the output with the switch faults.

Experiment 3: Study of 2^{nd} order and 4^{th} order low pass butterworth filters.

Objective: To study the 2^{nd} order and 4^{th} order low pass butterworth filters on the reconstruction of the signal.

Results of the above 3 experiments:

Natural sampling:

2nd order:

2KHz

4KHz

8KHz


RIGOL STOP
0.00V
2.00V
Time 200.0us
0.0000s

16KHz


RIGOL STOP
0.00V
2.00V
Time 200.0us
0.0000s

32KHz

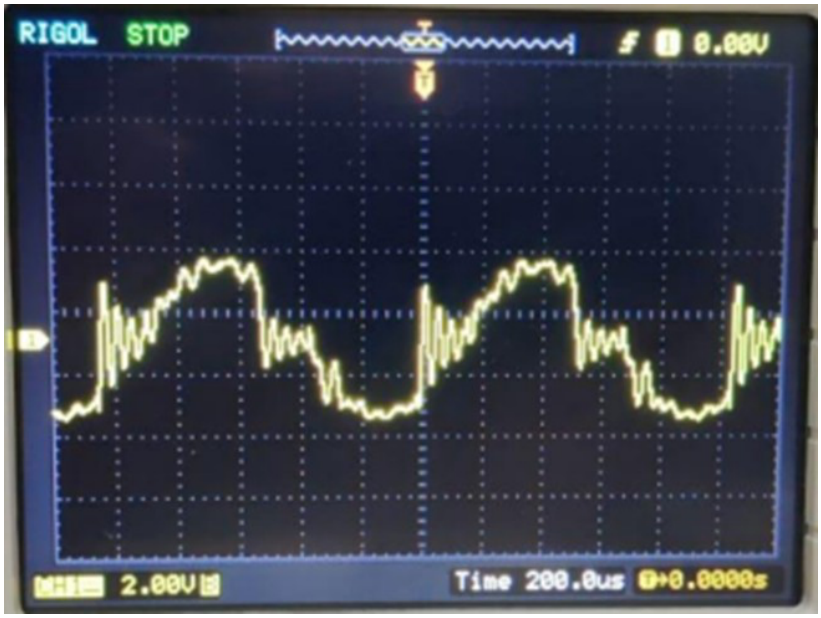

RIGOL
STOP
0.00V
2.00V
Time 200.0us
0.0000s

64KHz

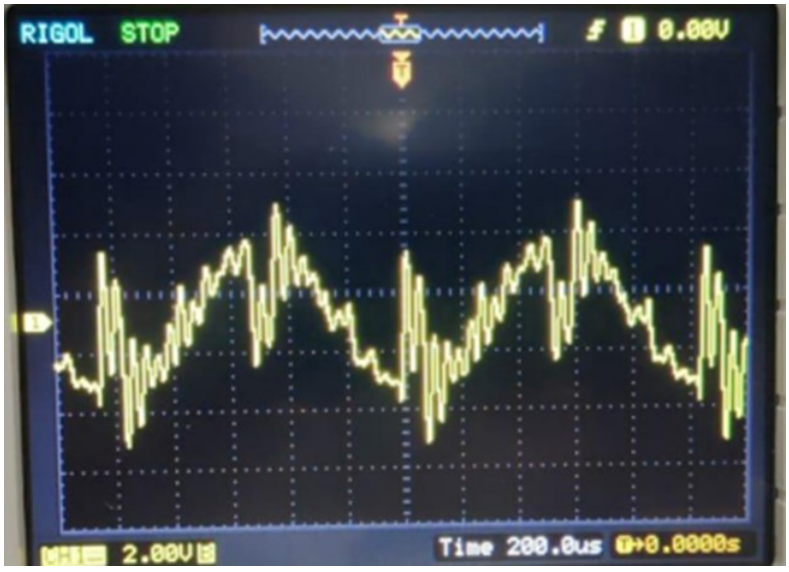

RIGOL
STOP
0.00V
2.00V
Time 200.0us
0.0000s

4th order:

2KHz



4KHz



8KHz


RIGOL
STOP
0.00V
2.00V
Time 200.0us
0.0000s

16KHz


RIGOL
STOP
0.00V
2.00V
Time 200.0us
0.0000s

32KHz


RIGOL
STOP
0.00V
2.00V
Time 100.0us
0.0000s

64KHz


RIGOL
STOP
0.00V
1.00V
Time 100.0us
0.0000s

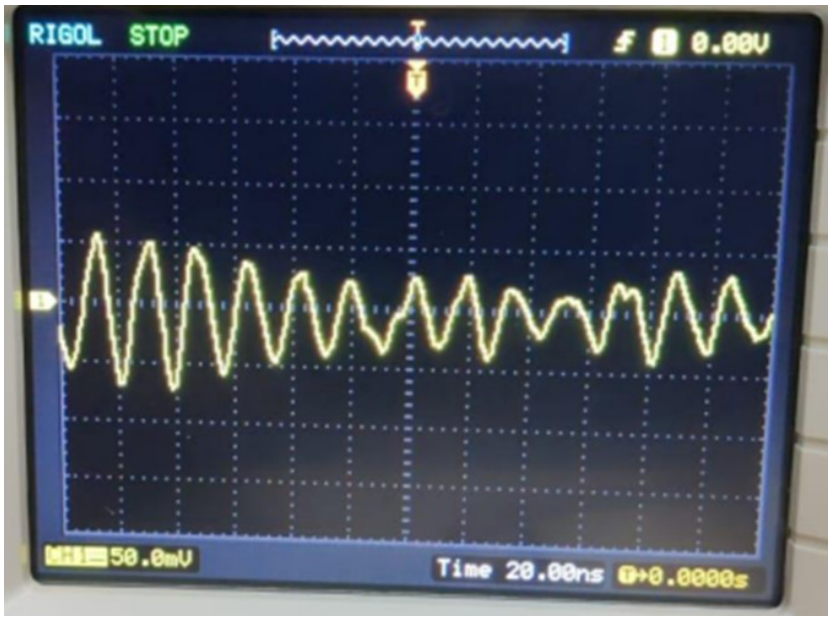
Sample and Hold:

2nd order:

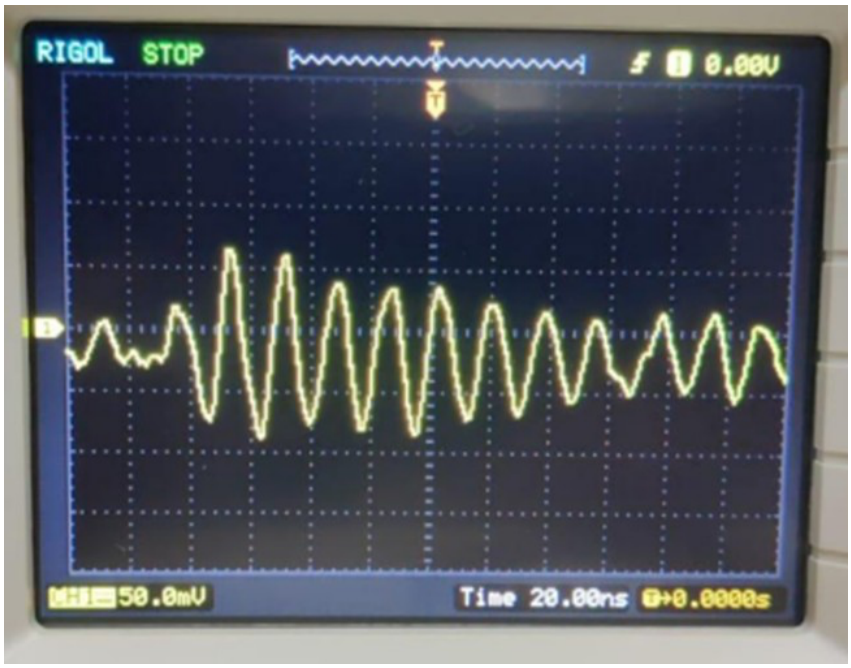
2KHz

4KHz

8KHz


RIGOL STOP
0.00V
50.0mV
Time 20.00ns
0.0000s

16KHz


RIGOL STOP
0.00V
50.0mV
Time 20.00ns
0.0000s

32KHz


RIGOL STOP
-12.0mV
100mV
Time 10.00ns
0.0000s

64KHz


RIGOL STOP
-12.0mV
100mV
Time 10.00ns
0.0000s

4th order:

2KHz

4KHz

8KHz


RIGOL STOP
-48.0mV
200mV
Time 10.00us 0+0.0000s

16KHz


RIGOL STOP
64.0mV
200mV
Time 10.00us 0+0.0000s

32KHz


RIGOL
STOP
-120mV
500mV
Time 10.00us
0.0000s

64KHz


RIGOL
STOP
-112mV
200mV
Time 5.000us
0.0000s

Flat Top Sampling:

2nd order:

2KHz

4KHz

8KHz


RIGOL
STOP
0.00V
1.00V
Time 200.0us 0+0.0000s

16KHz


RIGOL
STOP
-40.0mV
1.00V
Time 100.0us 0+0.0000s

32KHz


RIGOL STOP
1 0.00V
1.00V
Time 100.0us 0+0.0000s

64KHz


RIGOL STOP
1 -80.0mV
1.00V
Time 100.0us 0+0.0000s

4th order:

2KHz

4KHz

8KHz


RIGOL STOP
640mV
2.00V
Time 200.0us
0.0000s

16KHz


RIGOL STOP
400mV
2.00V
Time 200.0us
0.0000s

32KHz


RIGOL STOP
600mV
1.00V
Time 100.0us 0.0000s

64KHz


RIGOL STOP
60.0mV
1.00V
Time 100.0us 0.0000s

4. Study section 2.5.2 from [2]. Subsequently, study example 2.5.7. Then analyze code fragment 2.5.1 and implement it in MATLAB. Once done, solve problem 4 (all parts) in the laboratory assignment under section "software Lab 2.0" on page number 86 of the textbook.

```
%% Experiment 2 - 4(a)

t = -8:1/16:8;

s = 3*sinc(2*t-3);

[X,f,df] = contFT(s,-8,1/16,1e-3);

 figure(1);

title('Magnitude Response of s(t)');

plot(f,abs(X));

ylabel('Magnitude |X(f)|');

xlabel('Frequency (MHz)');


%% Experiment 2 - 4(b)

figure(2);

title('Phase response of s(t)');

plot(f,angle(X));

ylabel('Phase(X(f))');

xlabel('Frequency (MHz)');

%% contFT function

function [X,f,df] = contFT(x,tstart,dt,df_desired)

%INPUTS

%x = vector of time domain samples, assumed uniformly spaced

%tstart= time at which first sample is taken

%dt = spacing between samples
```

```
%df_desired = desired frequency resolution


%OUTPUTS
%X=vector of samples of Fourier transform

%f=corresponding vector of frequencies at which samples are obtained

%df=freq resolution attained (redundant--already available from
difference of consecutive entries of f)

%minimum FFT size determined by desired freq res or length of x

Nmin=max(ceil(1/(df_desired*dt)),length(x));

%choose FFT size to be the next power of 2

Nfft = 2^(nextpow2(Nmin));

%compute Fourier transform, centering around DC

X=dt*fftshift(fft(x,Nfft));

%achieved frequency resolution

df=1/(Nfft*dt);

%range of frequencies covered

f = ((0:Nfft-1)-Nfft/2)*df; %same as f=-1/(2*dt):df:1/(2*dt) - df

%phase shift associated with start

time X=X.*exp(-1i*2*pi*f*tstart);

end
```

Figure 1:

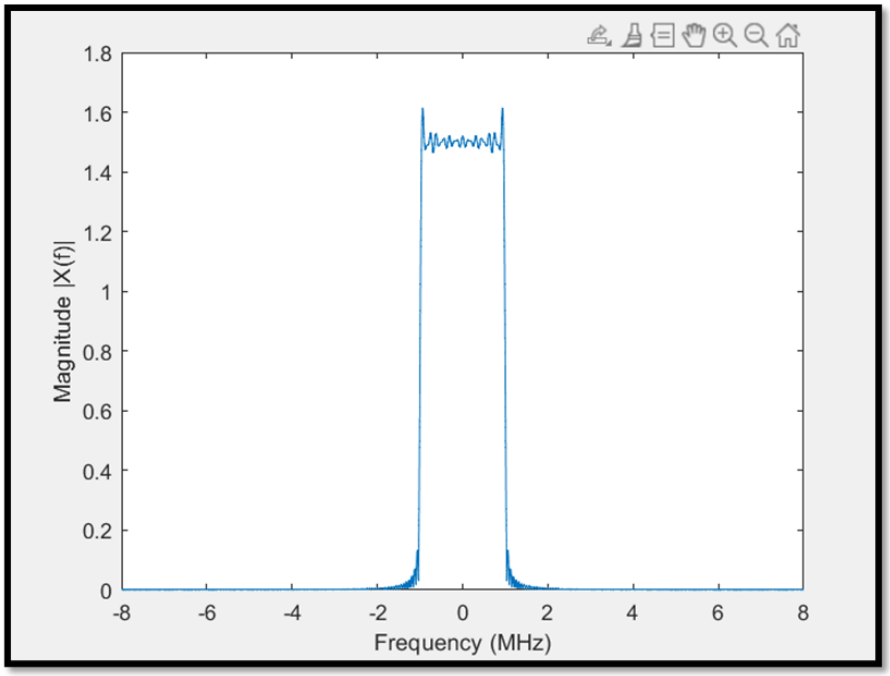
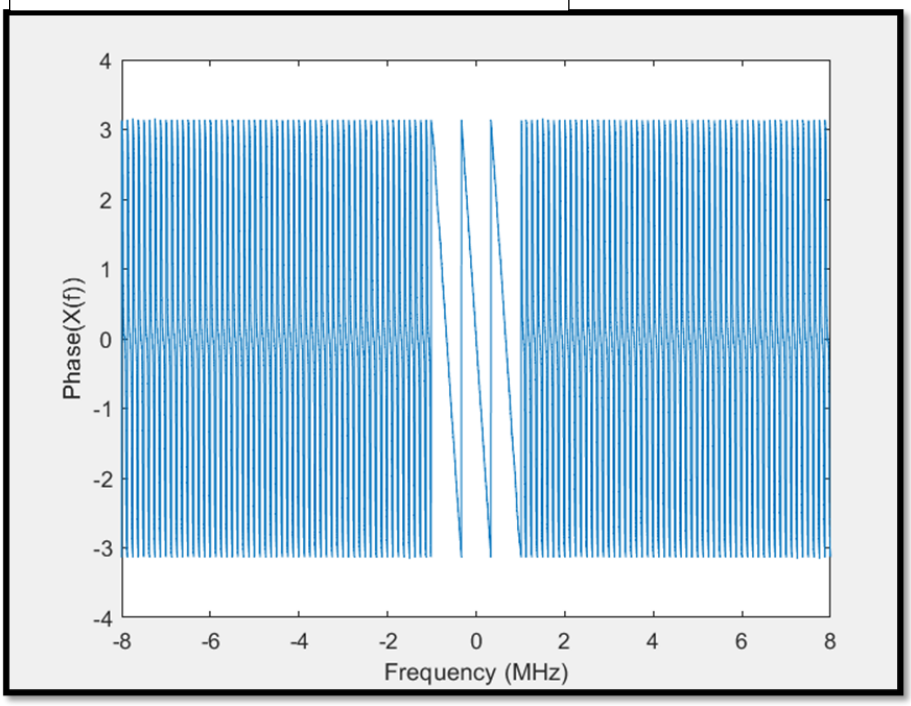


Figure 2:



Observation:

- Figure 1 shows the magnitude response of rectangular pulse 3*sinc(2t-3).
- However, due to the sampling and truncation of the time domain sinc pulse has some ripples at the corner. This effect is known as Gibbs Phenomenon.
- Figure 2 shows the Phase response of 3*sinc(2t-3). The Phase plot has a meaning in the interval f = [-1,1]

LAB - 5

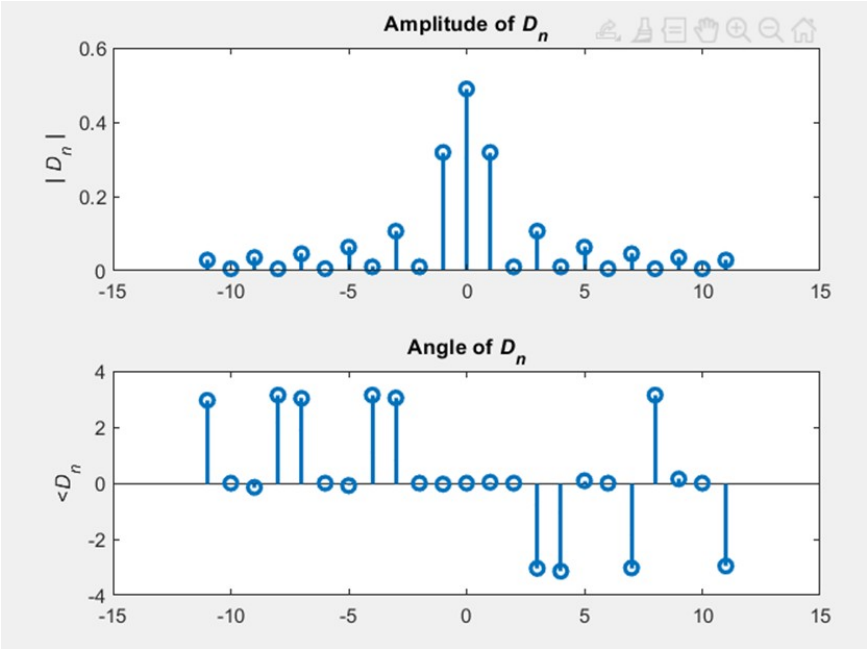
1. From section 2.8 in [2], study "Numerical Computation of Coefficients ...", and numerically obtain the Fourier series coefficients of the square pulse periodic signal, given in example 2.4.

```
j = sqrt(-1); % Define j for complex algebra
b = pi/2 ; a = -pi/2; %determine one signal
period tol = exp(-5); % set integration error
tolerance
T = 2*(b-a); % length of the
period t = -10:0.01:10;
N = 11; %Number of FS coefficients on each side of zero
frequency
Fi = (-N:N)*2*pi/T ; % Set frequency range
% now calculate D_O and store it in D(N+l) ;
Func = @(t) funct_sqr(t) ;
D(N+1) = (1/T)*(quad(Func,a,b,tol)); % Using quad.m
integration
```

```
for i = 1 : N
%Calculate Dn for n=l,...,N(stored in D(N+2)...D(2N+l)
Func = @(t)exp(-j*2*pi*t*i/T).*funct_sqr(t);
D(i+N+1)=(1/T)*quad(Func,a,b,tol);
% Calculate Dn for n=-N,...,-1(stored in
D(l)...D(N) Func=@(t)exp(j*2*pi*t*(N+1-
i)/T).*funct_sqr(t);
D(i)=(1/T)*quad(Func,a,b,tol);
end
subplot(211);
s1 = stem((-N:N),abs(D));
set(s1,'Linewidth',2);
ylabel('| {\itD}_{\itn} |');
title('Amplitude of
{\itD}_{\itn}'); subplot(212);
s2 = stem((-N:N), angle(D));
set(s2 ,'Linewidth',2) ;
ylabel('<{\itD}_{\itn}'); title('Angle of
{\itD}_{\itn}');
% (funct_tri.m)
% A standard tri angle function of base - 1
to 1 function y = funct_tri(t)
% Usage y = func_tri(t)
```

```
% t = input variable i
y = ((t>-1) - (t>1)).*(1-abs(t)) ;
end
%Used inbuilt function of MATLAB for generating a Square Wave.
function s = funct_sqr(t)
s = (square (t +
pi/2)+1)/2; end
```

Figure:



Observation:

- The above figure shows the Fourier Series coefficients of Square Wave. The magnitude Spectrum of Dn is symmetric for a Square Pulse. Phase Spectrum is the additive inverse on both the sides of origin for a Square Pulse.

2. Numerically solve problem 1.10 from chapter 1 in [1]. For the same,

refer to section 1.2.1 and illustrative problem 1.4 (along with its Matlab script) given on page 14 of the textbook.

1.10 A periodic signal x(t) with period To=6 is defined by x(t)= for |t|<=3. This signal passes through an LTI system with impulse response given by

h(t) = {e^-t , 0 ≤ t ≤ 4

{0 otherwise

Numerically determine and plot the discrete spectrum of the output signal.

Ans:

Fourier series coefficient of x(t):

$$a_n = \frac{1}{T_o}\int_{-\frac{T_o}{2}}^{\frac{T_o}{2}} x(t)e^{-\frac{j2\pi nt}{T_o}}dt$$

$$= \frac{1}{6}\int_{-3}^{3} x(t)e^{-\frac{j2\pi nt}{6}}dt$$

$$= \frac{1}{6}\int_{-3/2}^{3/2} 1.e^{-\frac{j2\pi nt}{6}}dt = \frac{1}{6}\left(-\frac{2jsin\left(2*\pi*\frac{n}{6}*\frac{3}{2}\right)}{-\frac{j2\pi n}{6}}\right)$$

$$= \sin\left(\pi * \frac{n}{2}\right)/(\pi * n)$$

$$= \frac{1}{2}sinc\left(\frac{n}{2}\right)$$

$$\therefore a_n = \frac{1}{2}sinc\left(\frac{n}{2}\right)$$

Plot of discrete spectrum of x(t):

```
%% Experiment 2
clc;
clear; close
all; clear vars;

n = [-80:80];

%Fourier Series coefficient of x(t)
x=(1/2)*(sinc(n/2));

%Sampling Interval ts
= 1/40;

%time vector

t = [-0.5:ts:4.5];

hh = exp(-t(21:181)./2);

%Impulse Response
fs = 1/ts;

h = [zeros(1,20) hh zeros(1,20)];

%Transfer Function H
= fft(h)/fs;

%Frequency Resolution df
= fs/80;

f = [0:df:fs] - fs/2;

%Rearrange H

H1 = fftshift(H); y =
x.*H1(21:181);

figure(1);
stem(n,x,'-o');
```

```
title('Discrete spectrum of x(t)');

figure(2);

stem((-600:6:600),abs(H1),'-o');

title('Magnitude of H(n/6)');
figure(3);

stem(n,abs(y),'-o');

title('Discrete spectrum of output');


%% Experiment 2
clc;
clear; close
all; clear vars;

n = [-80:80];

%Fourier Series coefficient of x(t)
x=(1/2)*(sinc(n/2));

%Sampling Interval ts
= 1/40;

%time vector

t = [-0.5:ts:4.5];

hh = exp(-t(21:181)./2);

%Impulse Response
fs = 1/ts;

h = [zeros(1,20) hh zeros(1,20)];

%Transfer Function H
= fft(h)/fs;
```

```
%Frequency Resolution df
= fs/80;

f = [0:df:fs] - fs/2;

%Rearrange H

H1 = fftshift(H); y =
x.*H1(21:181);

figure(1);
stem(n,x,'-o');

title('Discrete spectrum of x(t)');


figure(2);

stem((-600:6:600),abs(H1),'-o');

title('Magnitude of H(n/6)');
figure(3);

stem(n,abs(y),'-o');

title('Discrete spectrum of output');
```

Figure 1:

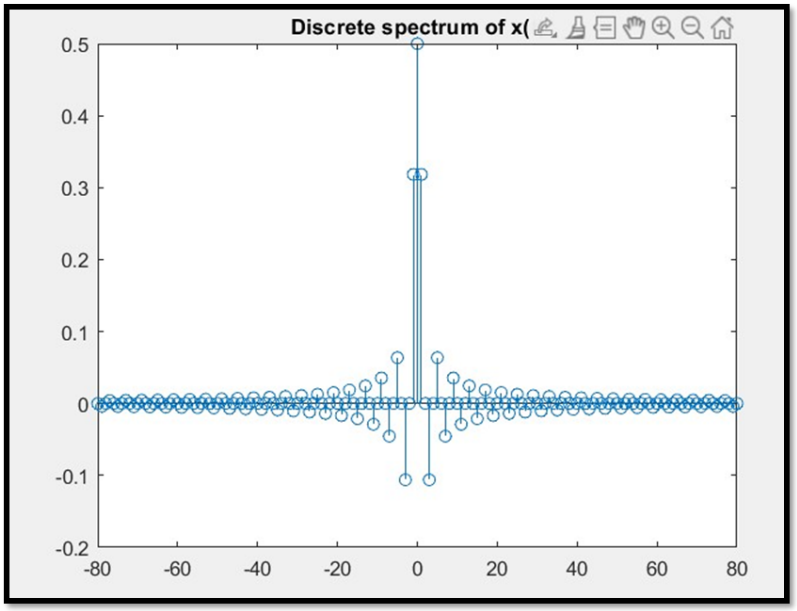


Figure 2:

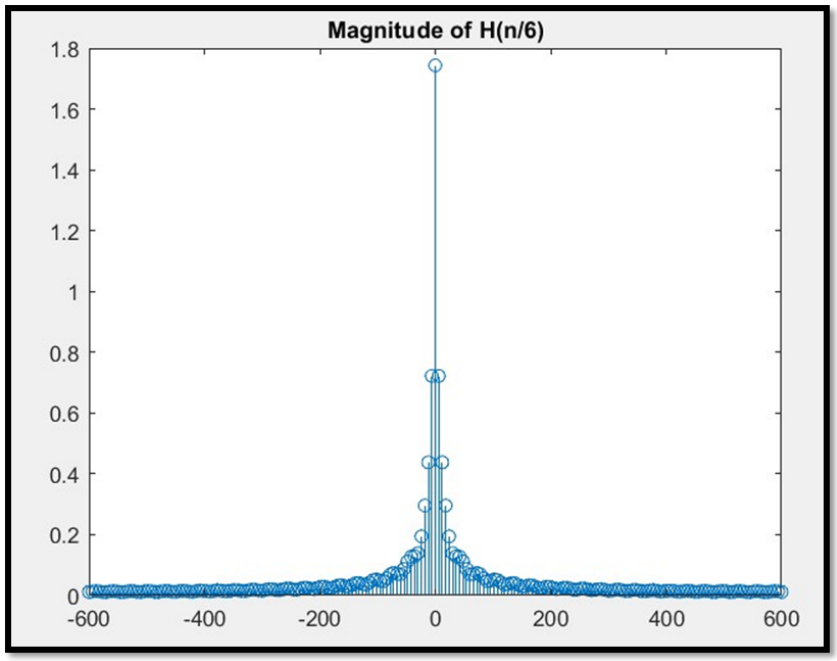
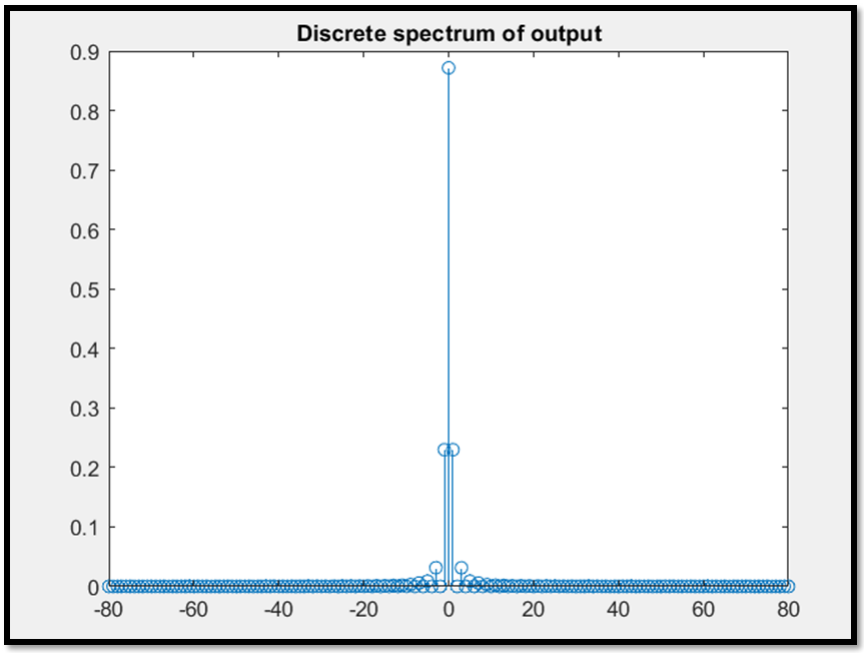


Figure 3:



Observations:

- Xn = (1⁄2) x sinc(n/2) is plotted in figure 1.
- H(n/T), where T = 6; is plotted in figure 2.
- Yn = (1⁄2) x sinc(n/2) x H(n/6) is plotted in figure 3.

3. From section 6.9 in [2], study the M-files “Exsample.m”, “uniquan.m”, “sampandquant.m”, and “ExPCM.m”. Based on the knowledge gained, write your own code for sampling and reconstructing sum of two cosine functions of duration 2 seconds and frequencies 5 Hz and 8 Hz, respectively. You need to choose the sampling rate yourself, considering all the aspects of sampling theory studied in the lectures. In your implementation, it is important for you to understand the implementation of ideal-low-pass-filtering operation.

Ans:

```
td=0.002;

% original sampling rate 500 Hz
t=0:td:2.; % time interval of 1 second
xsig=cos(2*5*pi*t)+cos(2*8*pi*t);

% 1Hz+3Hz sinusoids
Lsig=length(xsig);

ts=0.02; %new sampling rate =
50Hz. Nfactor=ts/td;

% send the signal through a 16-level uniform quantizer
[s_out,sq_out,sqh_out,Delta,SQNR]=sampandquant(xsig,16,td,ts);

% receive 3 signals:

% 1. sampled signal s_out

% 2. sampled and quantized signal sq_out

% 3. sampled, quantized, and zero-order hold signal sqh out
```

```
% calculate the Fourier
transforms
Lfft=2^ceil(log2(Lsig)+1);


Fmax=1/(2*td);

Faxis=linspace(-Fmax,Fmax,Lfft); Xsig=fftshift(fft(xsig,Lfft));
S_out=fftshift(fft(s_out,Lfft));

% Examples of sampling and reconstruction using

% a) ideal impulse train through LPF

% b) flat top pulse reconstruction through LPF

% plot the original signal and the sample signals in time

% and frequency domain
figure(1); subplot(311);

sfig1a=plot(t,xsig,'k'); hold on;

sfiglb=plot(t,s_out(1:Lsig),'b'); hold
off; set(sfig1a,'Linewidth',2);
set(sfiglb,'Linewidth',2.); xlabel ( '
time (sec) ' ) ;

title('Signal {\it g}({\it t}) and its uniform samples');

subplot(312); sfiglc=plot(Faxis,abs(Xsig));
xlabel('frequency (lie)');
axis([-150 150 0 300])

set(sfiglc,'Linewidth',1);
title('Spectrum of {\it g} ({\it t})');

subplot(313);
sfigld=plot(Faxis,abs(S_out));
xlabel('frequency (lie)');
```

```
axis([-150 150 0 300/Nfactor])


set(sfiglc,'Linewidth',1);
title('Spectrum of {\it g}_T({\it t})');

% calculate the reconstructed signal from ideal sampling and

% ideal LPF

% Maximum LPF bandwidth equals to BW=floor((Lfft/Nfactor)/2);


BW=40; %Bandwidth is no larger than
10Hz. H_lpf=zeros(1,Lfft);
H_lpf(Lfft/2-BW:Lfft/2+BW-1)=1; %ideal LPF
S_recv=Nfactor*S_out.*H_lpf; % ideal filtering
s_recv=real(ifft(fftshift(S_recv))); % reconstructed f-domain
s_recv=s_recv(1:Lsig); % reconstructed t-domain


% plot the ideally reconstructed signal in time

% and frequency domain
figure(2) subplot(211);

sfig2a=plot(Faxis,abs(S_recv));
xlabel('frequency (Hz)');
axis([-150 150 0 300]);
title('Spectrum of ideal filtering
(reconstruction)'); subplot(212);
sfig2b=plot(t,xsig,'k-.',t,s_recv(1:Lsig),'b');
legend('original signal','reconstructed signal');
xlabel('time (sec)');

title('original signal vs ideally reconstructed signal');
set(sfig2b,'Linewidth',2);
```

```
% non-ideal reconstruction ZOH=ones(1,Nfactor);
s_ni=kron(downsample(s_out,Nfactor),ZOH);
S_ni=fftshift(fft(s_ni,Lfft));

S_recv2=S_ni.*H_lpf; % ideal filtering
s_recv2=real(ifft(fftshift(S_recv2))); % reconstructed f-
domain

s_recv2=s_recv2(1:Lsig); % reconstructed t-domain

% plot the ideally reconstructed signal in time

% and frequency domain
figure(3) subplot(211);

sfig3a=plot(t,xsig,'b',t,s_ni(1:Lsig),'b'); xlabel('time (sec)');

title('original signal vs flat-top
reconstruction'); subplot(212);
sfig3b=plot(t,xsig,'b',t,s_recv2(1:Lsig),'b--');
legend('original signal','LPF reconstruction');
xlabel('time (sec)');

set(sfig3a,'Linewidth',2);

set(sfig3b,'Linewidth',2); title('original and

flat-top reconstruction after LPF');


function [q_out,Delta,SQNR]=uniquan(sig_in,L)

% Usage
% [q_out,Delta,SQNR]=uniquan(sig_in,L)
% L - number of uniform quantization levels
% sig_in - input signal vector
% Function outputs:
```

```
% q_out - quantized output

% Delta - quantization interval

% SQNR - actual signal to quantization noise
ratio sig_pmax=max(sig_in); % finding the
positive peak sig_nmax=min(sig_in); % finding
the negative peak Delta=(sig_pmax-sig_nmax)/L;
% quantization interval
q_level=sig_nmax+Delta/2:Delta:sig_pmax-Delta/2; % define Q-

levels L_sig=length(sig_in); % find signal length

sigp=(sig_in-sig_nmax)/Delta+1/2; % convert into 1/2 to L+1/2
range

qindex=round(sigp); % round to 1, 2, ... L levels
qindex=min(qindex,L);
% eleminate L+1 as a rare possibility
q_out=q_level(qindex); % use index vector to generate
output SQNR=20*log10(norm(sig_in)/norm(sig_in-q_out));
%actual SQNR value

end



% (sampandquant.m)

function
[s_out,sq_out,sqh_out,Delta,SQNR]=sampandquant(sig_in,L,td,ts)

% Usage

[s_out,sq_out,sqh_out,pelta,SQNR1=sampandquant(sig_in,L,td,fs)

% L - number of uniform quantization levels

% sig_in - input signal vector
```

```
% td - original signal sampling period of sig_in

% ts - new sampling period

% NOTE: td*fs must be a positive integer;

% Function outputs:

% s_out - sampled output

% sq_out - sample-and-quantized output

% sqh_out- sample,quantize,and hold output


% Delta - quantization interval

% SQNR - actual signal to quantization noise ratio
if (rem(ts/td,1)==0)

nfac=round(ts/td); p_zoh=ones(1,nfac);
s_out=downsample(sig_in,nfac);
[sq_out,Delta,SQNR]=uniquan(s_out,L);
s_out=upsample(s_out,nfac);
sqh_out=kron(sq_out,p_zoh);

else

warning ( ' Error ! ts / td is not an integer ! '

) ; s_out= []; sq_out= [];
sqh_out= [];
Delta = [];

SQNR = [];

end
```

Figure 1:

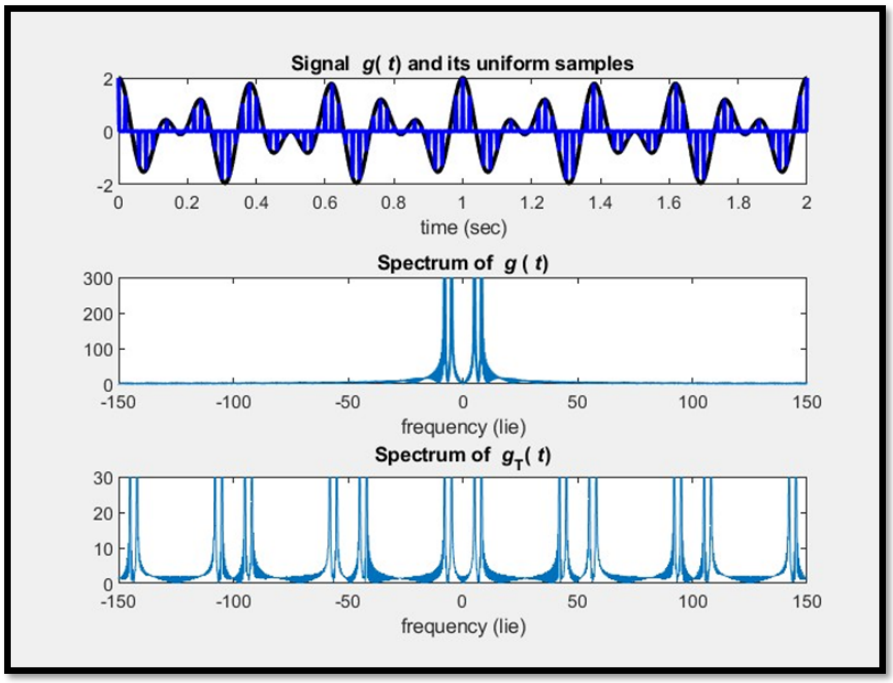


Figure 2:

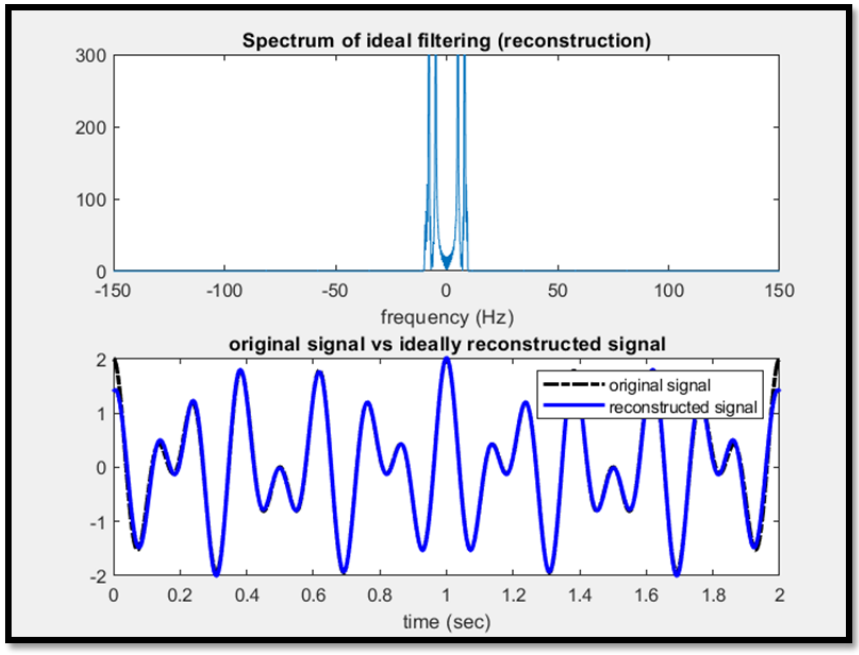
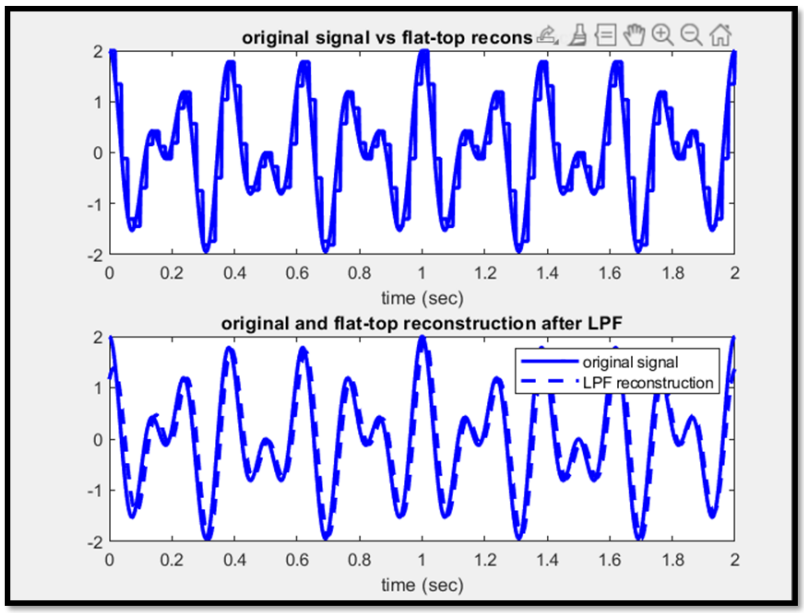


Figure 3:



Observations:

- Signal is sampled at a rate of 50 Hz and 16 levels Uniform PCM System is used. The sampled output of g(t) and spectrum of g_delta(t) are shown in figure 1.
- After Sampling, the signal is passed through a LPF of BW = 40 Hz. Now, the original signal is reconstructed using this filtered output. The filtered output and reconstructed signal are shown in figure 2.
- Figure 3 shows the Flat Top reconstructed signal which is obtained from the filtered Signal. The flat top reconstructed signal is then passed through LPF. The output of LPF is also shown in Figure 3.