CT303

INTRODUCTION TO COMMUNICATION SYSTEMS

Lab 4 and 5

STUDENT NAME : Divya Patel

STUDENT ID : 202001420

LAB-GROUP : 6

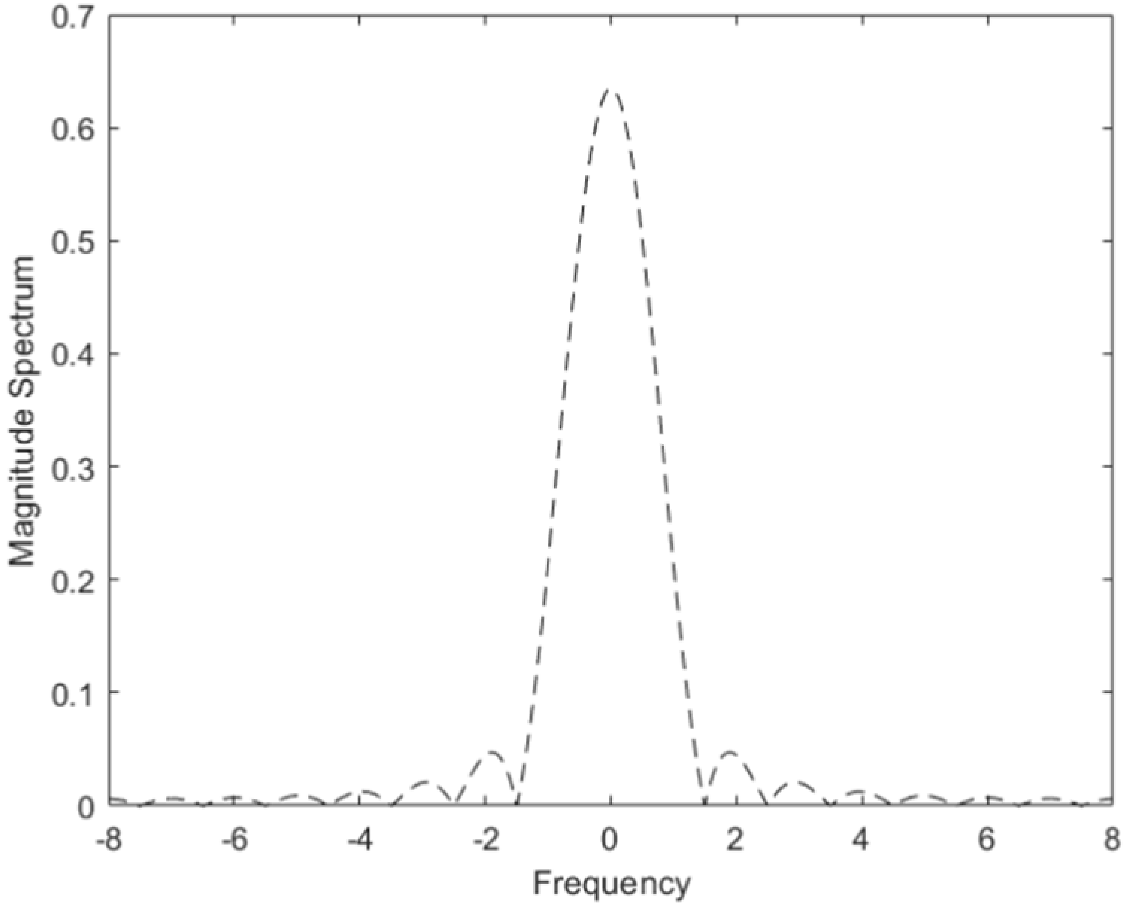
Lab Exercise 4

1. ***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 text book.***

```
ts = 1/16; %sampling interval
time_interval = 0:ts:1; %sampling time instants

%%time domain signal evaluated at sampling instants
signal_timedomain = sin(pi*time_interval); %sinusoidal pulse in our example
fs_desired = 1/160; %desired frequency granularity
Nmin = ceil(1/(fs_desired*ts)); %minimum length DFT for desired frequency granularity

%for efficient computation, choose FFT size to be power of 2
Nfft = 2^(nextpow2(Nmin)) %FFT size = the next power of 2 at least as big as Nmin

%Alternatively, one could also use DFT size equal to the minimum length
%Nfft = Nmin;
%note: fft function in Matlab is just the DFT when Nfft is not a power of 2
%freq domain signal computed using DFT

%fft function of size Nfft automatically zeropads as needed
signal_freqdomain = ts*fft(signal_timedomain,Nfft);
```

```
%fftshift function shifts DC to center of spectrum
signal_freqdomain_centered = fftshift(signal_freqdomain);
fs = 1/(Nfft*ts); %actual frequency resolution attained

%set of frequencies for which Fourier transform has been computed using DFT
freqs = ((1:Nfft)-1-Nfft/2)*fs;

%plot the magnitude spectrum
plot(freqs,abs(signal_freqdomain_centered), '--k');
xlabel('Frequency');
ylabel('Magnitude Spectrum');
```

```
Nfft = 4096
```



4(a) Use the function contFT to compute the Fourier transform of

s(t) = 3sinc(2t − 3), where the unit of time is a microsecond, the signal is sampled at the rate of 16 MHz, and truncated to the range [−8, 8] microseconds. We wish to attain a frequency resolution of 1 KHz or better. Plot the magnitude of the Fourier transform

versus frequency, making sure you specify the units on the frequency axis. Check that the plot conforms to your expectations.

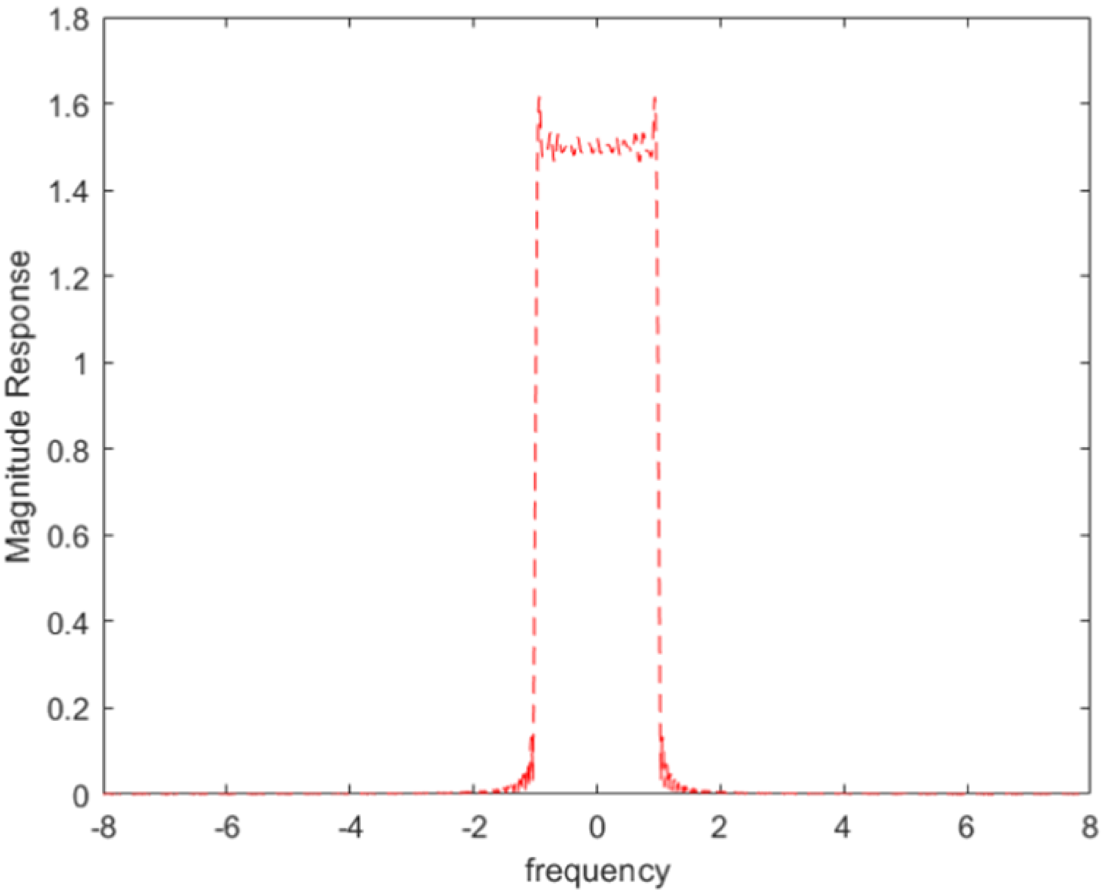
(b) Plot the phase of the Fourier transform obtained in (a) versus frequency (again, make sure the units on the frequency axis are specified). What is the range of frequencies over which the phase plot has meaning?

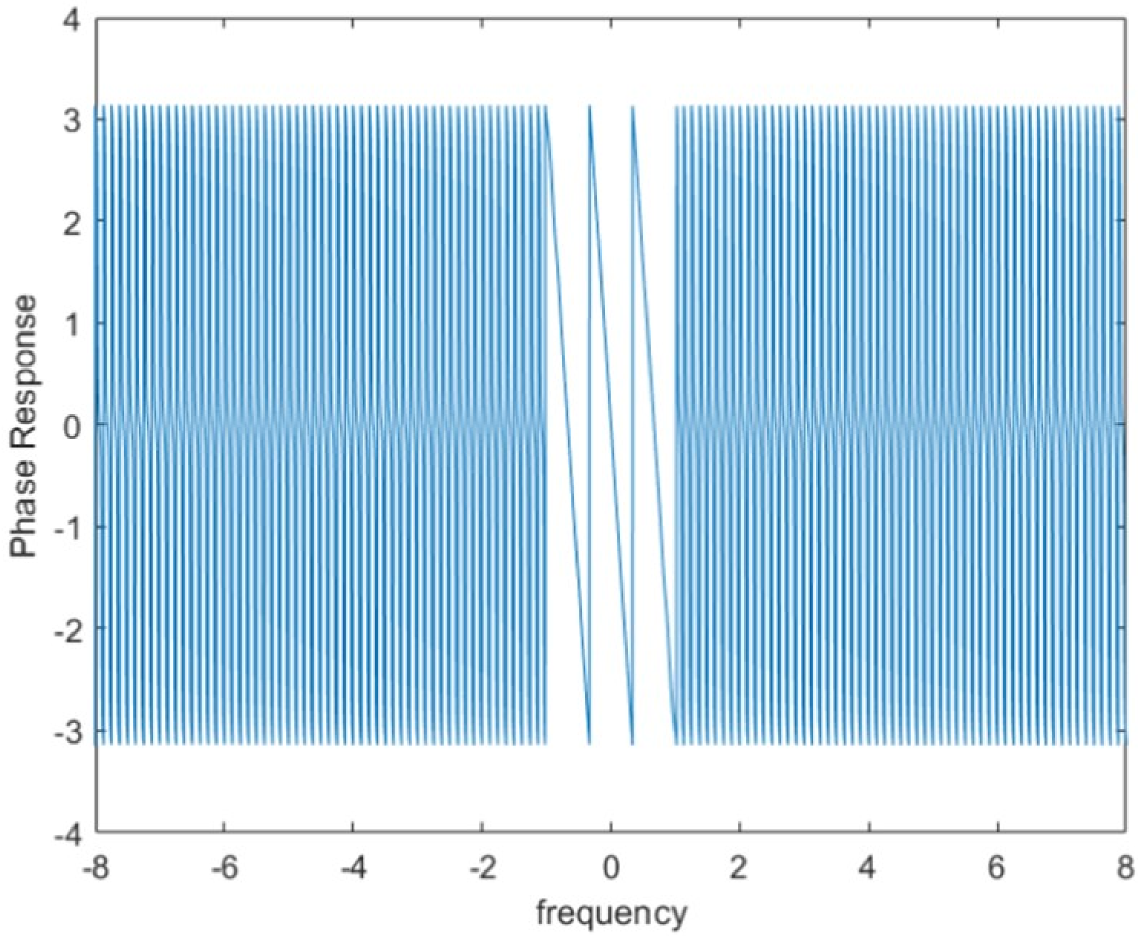
```
clc;
clear;
ts = 1/(16);
t = -8:ts:8;
signal_time_domain = 3*sinc(2*t-3);

%plot(t,signal_time_domain);
[X,f,df] = contFT(signal_time_domain,-8,ts,1e-3);
figure(1);
plot(f,abs(X), '--r');
xlabel('frequency');
ylabel('Magnitude Response');

figure(2);
plot(f,angle(X));
xlabel('frequency');
ylabel('Phase Response');
```

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

%Use Matlab DFT for approximate computation of continuous time Fourier
%transform
%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(-j*2*pi*f*tstart);
end
```

```
Nfft = 16384
df = 9.7656e-04
```



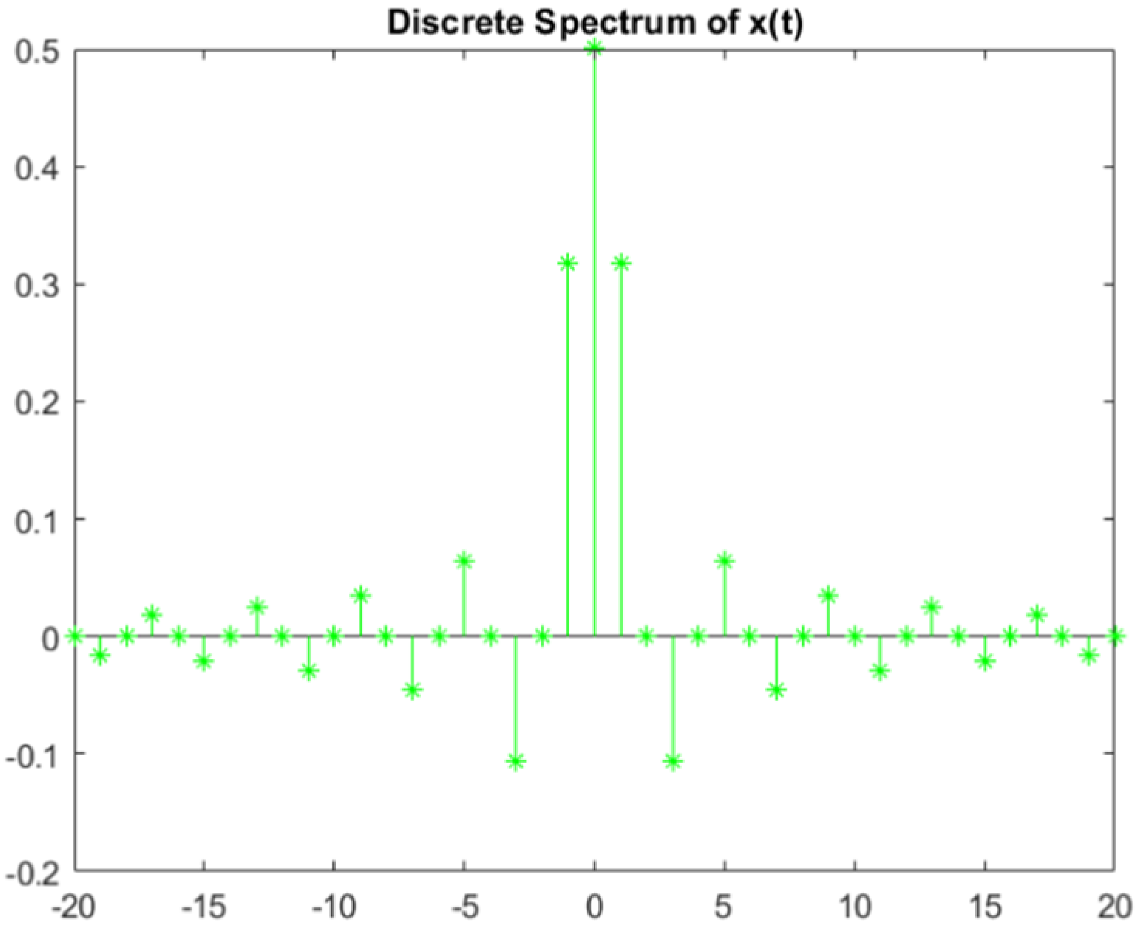


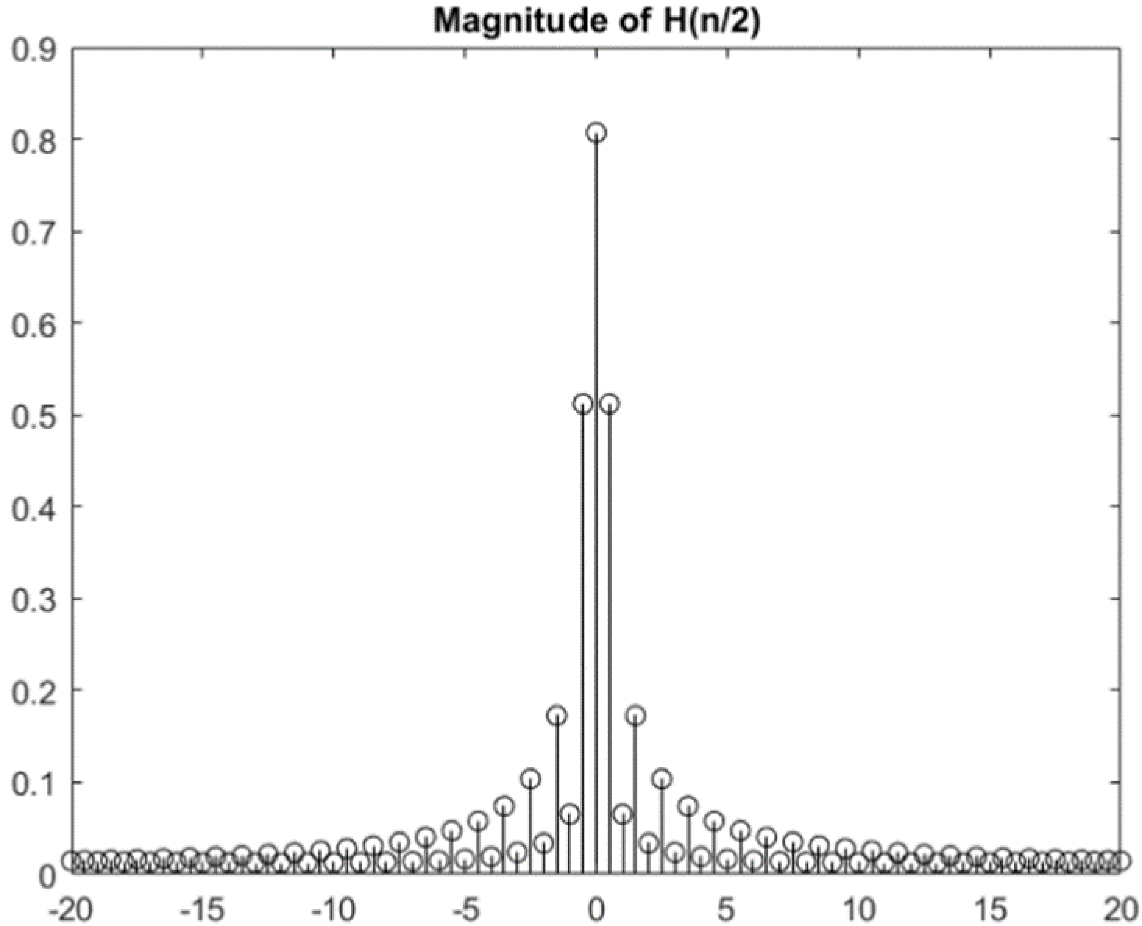
LAB - 5

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.***

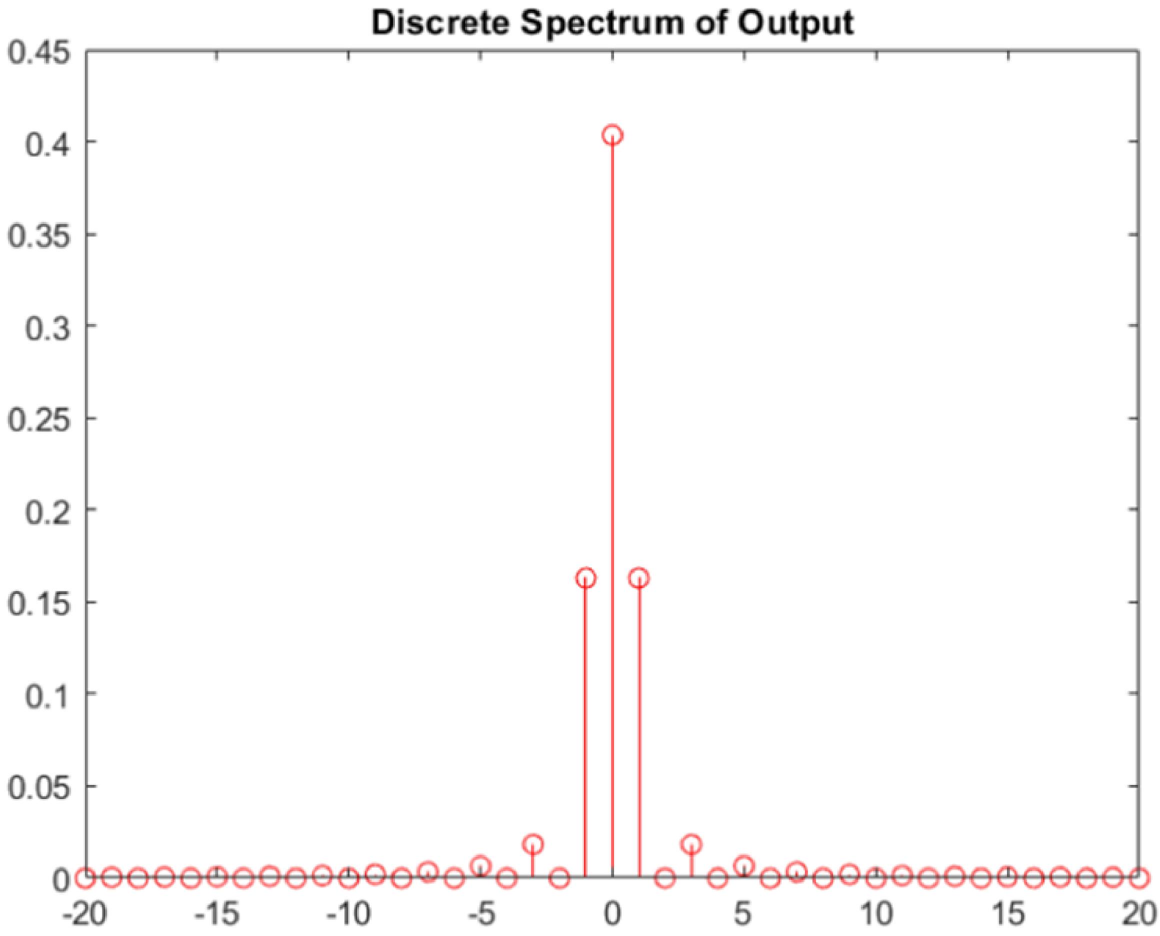
2. ***Numerically solve problem 1.10 from chapter 1 in [1]. For the same, refer to section 1.2.1 and illustrative problem 1.4 (along with its Matlab script) given on page 14 of the text book.***

```
echo on
n = [-20:1:20];
% fourier series co-efficents of x(t)
x = 0.5*sinc(n/2);
ts = 1/40; % time interval
%time vector
t = [-0.5:ts:1.5];
fs = 1/ts;
%impulse response
h = [zeros(1,20),exp(-t(21:61)./2),zeros(1,20)];
H = fft(h)/fs; %fourier transform of impulse response
df = fs/80;
f = [0:df:fs]-fs/2; % frequency interval
H1 = fftshift(H);
y = x.*H1(21:61); %fourier series of output
% plotting
figure(1);
stem(n,x,'-*g');
title('Discrete Spectrum of x(t)');
figure(2);
stem((-20:0.5:20),abs(H1),'-k');
title('Magnitude of H(n/2)');
figure(3);
stem(n,abs(y),'-or');
title('Discrete Spectrum of Output');
```


Discrete Spectrum of x(t)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-20
-15
-10
-5
0
5
10
15
20



Magnitude of H(n/2)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-20
-15
-10
-5
0
5
10
15
20



1. ***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.***

```
clc;
td = 0.002;

%original sampling rate 500 Hz
t= [0:td:2.] ;

%time interval of 1 second
% 1Hz+3Hz sinusoids
xsig = cos(10*pi*t) + cos(16*pi*t) ;
Lsig = length(xsig);
ts = 0.02;
```

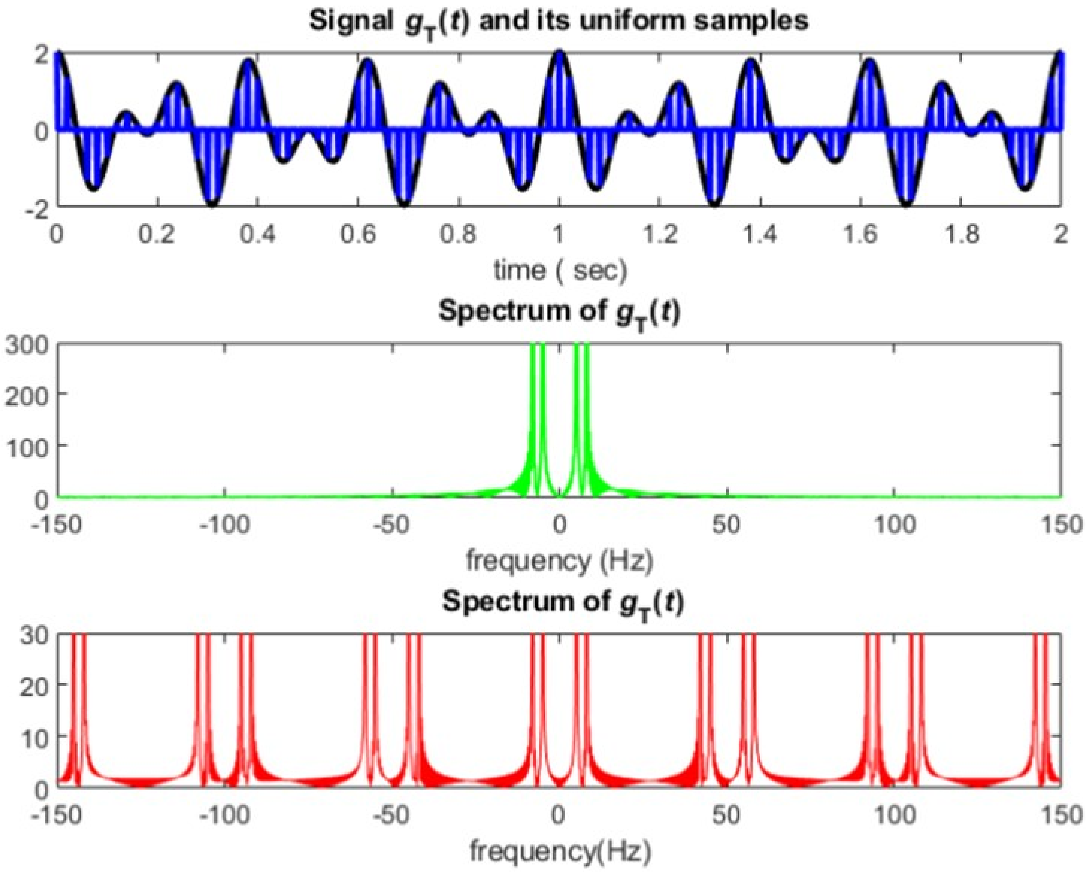
```
%inew sampling rate = 50Hz.
Nfactor = ts/td;

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

% 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);
sfigla = plot (t, xsig, 'k');
hold on;
sfiglb = plot (t, s_out (1:Lsig) , 'b');
hold off;

set (sfigla, 'Linewidth' , 2);
set (sfiglb, 'Linewidth' , 2);
xlabel ('time ( sec)');
title ('Signal {\itg}_T({\itt}) and its uniform samples');

subplot (312) ;
sfiglc = plot (Faxis, abs (Xsig), 'green' );
xlabel ('frequency (Hz)');
axis ([-150 150 0 300])
set (sfiglc, 'linewidth' , 1) ;
title('Spectrum of {\itg}_T({\itt})' ) ;

subplot (313) ;
sfigld = plot (Faxis, abs (S_out), 'red' ) ;
xlabel ('frequency(Hz)') ;
axis ( [-150 150 0 300/Nfactor] )
set(sfiglc , 'linewidth' , 1 ) ;
title ( ' Spectrum of {\itg}_T({\itt})' ) ;

BW= 10; %Bandwidth i s no l arger than l 0H z .
H_lpf = zeros (1 , Lfft ) ;
H_lpf (Lfft / 2 - BW : Lfft / 2+BW- 1 ) = 1; % i deal LPF
S_recv = Nfactor * S_out .* H_lpf ; % ideal f i l ter ing
s_recv = real(ifft (fftshift (S_recv))); % recons t ructed £ - domain
s_recv = s_recv (1 : Lsig ) ;
```

```
figure (2)
subplot(211);
sfig2a = plot ( Faxis , abs ( S_recv ), 'magenta' ) ;
xlabel ('frequency(Hz)');
axis ([ -150 150 0 60 ]) ;
title ('Spectrum of ideal filtering (reconstruction)');

subplot(212) ;
sfig2b = plot (t , xsig , 'k-.', t, s_recv (1 : Lsig ) , 'cyan');
legend ('original signal' , 'reconstructed signal');
xlabel ('time(sec)');
title ('original signal versus 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, 'k', t, s_ni (1:Lsig) , 'y' );
xlabel ('time (sec)') ;
title ('original signal versus flat-top reconstruction');

subplot (212 );
sfig3b = plot (t, xsig, 'b' , t, s_recv2 (1:Lsig) , 'g--' ) ;
legend ('original signal' , 'LPF reconstruction') ;
xlabel ('time (sec)') ;
set (sfig3a, 'Linewidth' , 2) ; set (sfig3b, 'Linewidth' , 2) ;
title ('original and flat-top reconstruction after LPF');
```

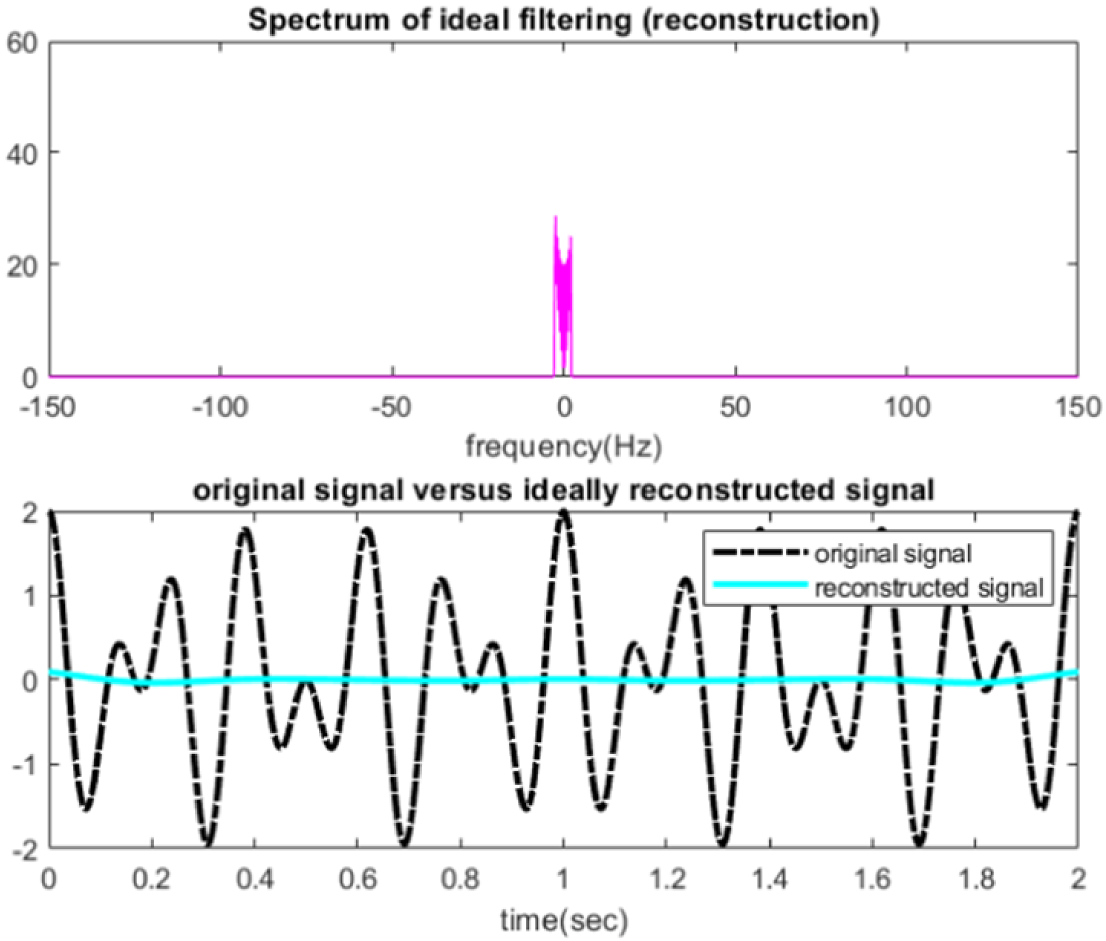
```
function [sqnr,a_quan,code] = u_pcm(a,n)
amax = max(abs(a));
a_quan = a/amax;
b_quan = a_quan;
d = 2/n;
q = d.*(0:n-1);
q = q-((n-1)/2)*d;
for i = 1:n
a_quan(find((q(i)-d/2 <= a_quan) & (a_quan <= q(i)+d/2)))=...
q(i).*ones(1,length(find((q(i)-d/2 <= a_quan) & (a_quan <= q(i)+d/2))));
```

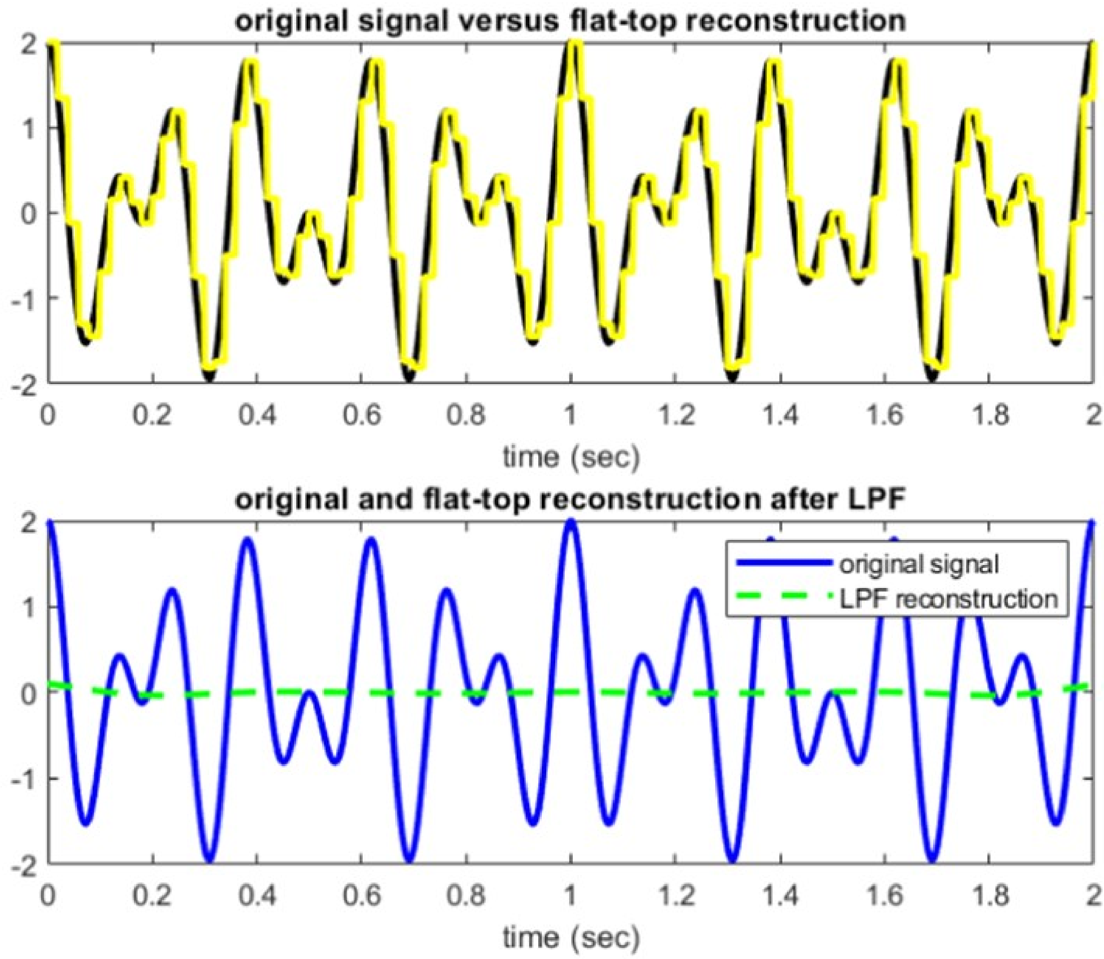
```
b_quan(find( a_quan==q(i) ))=(i-1) *ones(1,length(find( a_quan==q(i))));
End
a_quan = a_quan*amax;
nu = ceil(log2(n));
code = zeros(length(a),nu);

for i = 1:length(a)
for j = nu:-1:0
if (fix(b_quan(i)/(2^j)) == 1)
code(i,(nu-j)) = 1;
b_quan(i) = b_quan(i) - 2^j;
end
end
end
sqnr = 20*log10(norm(a)/norm(a-a_quan));
End
Code for 8 level & 16 level

clc;
echo on
t = 0:0.01:2;
yval = zeros(1,length(t));
f1 = @(x)(x); %Function
f2 = @(x)(-x+2);
for i = 1:length(t)
p = t(i);
if( p >= 0) && ( p < 1)
yval(i) = f1(p);
elseif ( p >= 1) && ( p < 2)
yval(i) = f2(p);
else
yval(i) = 0;
end
end
a = yval;
end
[sqnr8, aquan8, code8] = u_pcm(a,8);
[sqnr16, aquan16, code16] = u_pcm(a,16);
%Press a key to see the SQNR for N = 8
%pause
Sqnr8
%pause
% Press a key to see the SONR for N = 16
Sqnr16
% Press a key to see the plot of the signal and its quantized versions
%pause
figure;
plot(t,a,t,aquan8,'k-','linewidth',0.8);
legend('Original function', '8 level PCM Quantized output','Location','south');
xlabel('Time (t)');
ylabel('f(t) and f^{-}(t)');
title('8 level quantized output');
grid on;
figure;
```

```
plot(t,a,t,aquan16,'k-','linewidth',0.8);
legend('Original function', '16 level PCM Quantized output','Location','south');
xlabel('Time (t)');
ylabel('f(t) and f^{-}(t)');
title('16 level quantized output');
grid on;
figure;
plot(t, (a-aquan8), 'k-','linewidth',0.8);
xlabel('Time (t)');
ylabel('Quantization Error for 8 level');
grid on;
figure;
plot(t, (a-aquan16), 'k-','linewidth',0.8);
xlabel('Time (t)');
ylabel('Quantization Error for 16 level');
grid on;
fprintf('SQNR for 8-level Quantization %f\n\n', (sqnr8));
fprintf('SQNR for 16-level Quantization %f\n\n', (sqnr16));
end
function [ s_out , sq_out , sqh_out , Delta , SQNR ] = sampandquant(sig_in,L,td,ts)
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) ;
sq_out = upsample (sq_out, nfac) ;
else
warning ( 'Error! ts/td is not an integer! ') ;
s_out=[] ;
sq_out= [] ;
sqh_out= [] ;
Delta= [ ] ;
SQNR= [] ;
end
function [ q_out , Delta , SQNR ] = uniquan(sig_in , L)
sig_pmax = max(sig_in) ;
sig_nmax = min(sig_in);
Delta = (sig_pmax-sig_nmax) /L;
q_level = sig_nmax+Delta/2 : Delta: sig_pmax-Delta/2; % define Q-levels
L_sig = length(sig_in);
sigp = (sig_in-sig_nmax) /Delta+1/2;
qindex = round (sigp) ;
qindex = min (qindex, L) ;
q_out = q_level (qindex) ;
SQNR = 20*log10 (norm(sig_in)/norm(sig_in-q_out));
end
```



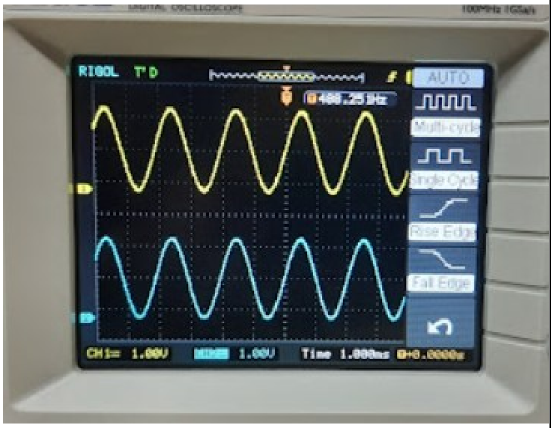
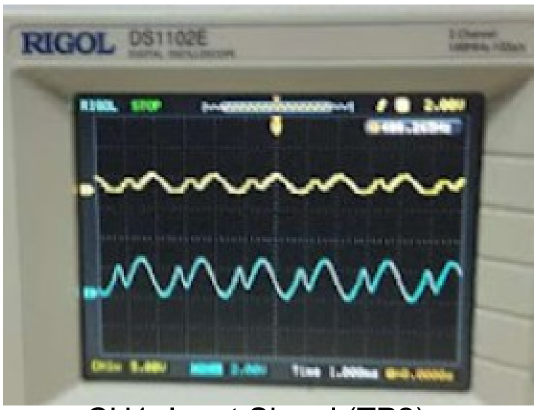
Signal $g_T(t)$ and its uniform samples
time (sec)
Spectrum of $g_T(t)$
frequency (Hz)
Spectrum of $g_T(t)$
frequency(Hz)

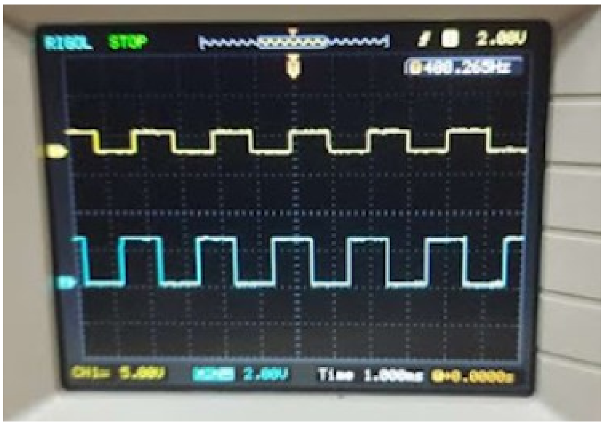
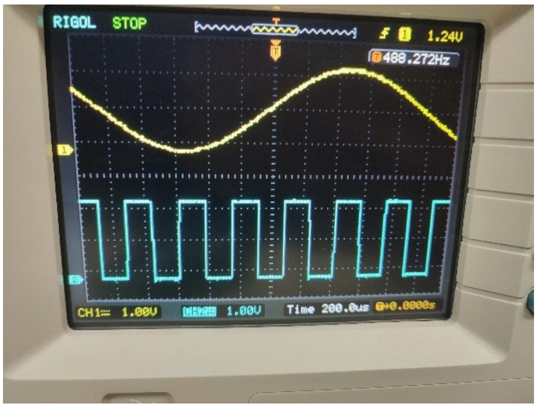

Spectrum of ideal filtering (reconstruction)
60
40
20
0
-150
-100
-50
0
50
100
150
frequency(Hz)
original signal versus ideally reconstructed signal
original signal
reconstructed signal
2
1
0
-1
-2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
time(sec)

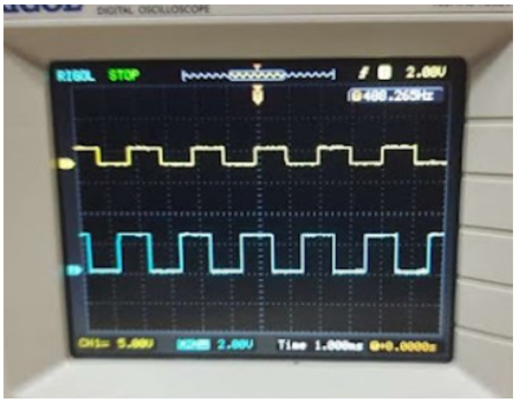


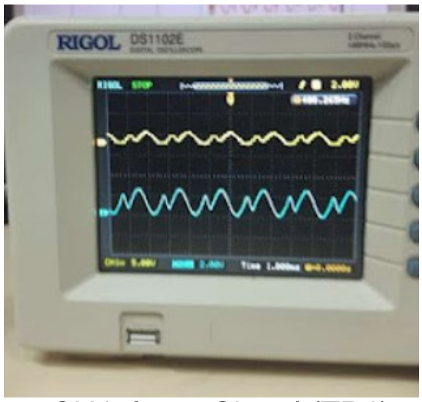
Experiment 8

I/P Signal Type/ Freq.	Channel	Sampling Freq.	Sampled Output

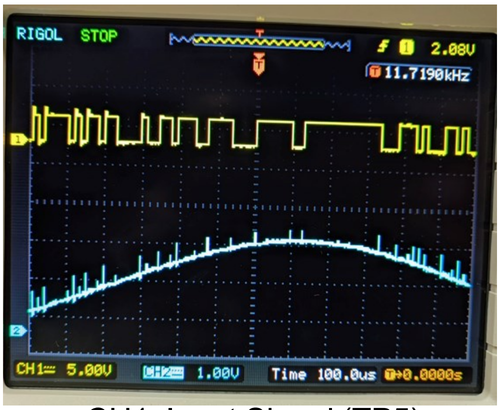
Sine/500 Hz	Channel1	-	 CH1: Input Signal (TP1), CH2: Channel1 input(TP5)
Arbitrary /1.5 kHz	Channel2	-	 CH1: Input Signal (TP2), CH2: Channel1 input(TP10)

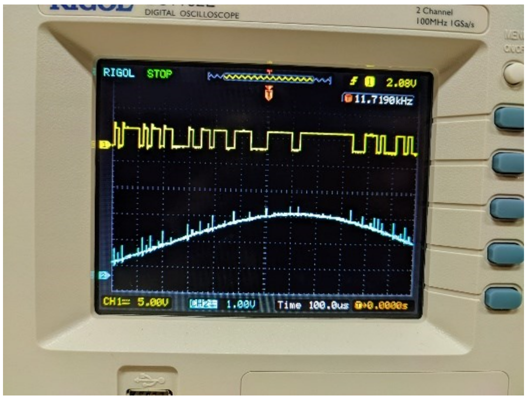
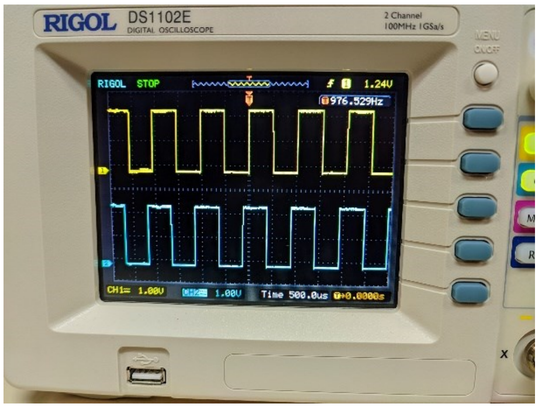
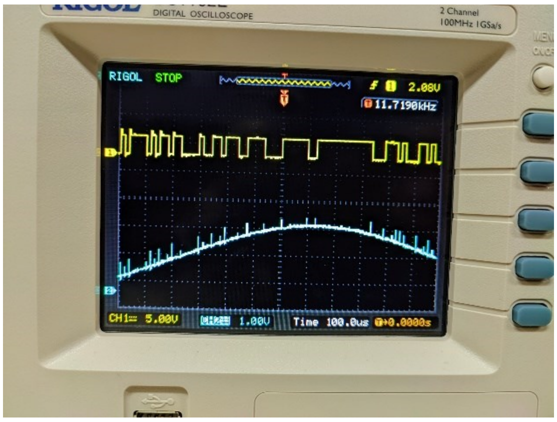
Square /500 Hz	Channel 3	-	 CH1: Input Signal (TP3), CH2: Channel1 input(TP15)
Sine/500 Hz	Channel 1	8 KHz	 CH1: Input Signal (TP5), CH2: Sampled Output(TP7)
Sine/500 Hz	Channel 1	8 KHz	 CH1: Input Signal (TP6), CH2: Sampled Output(TP7)

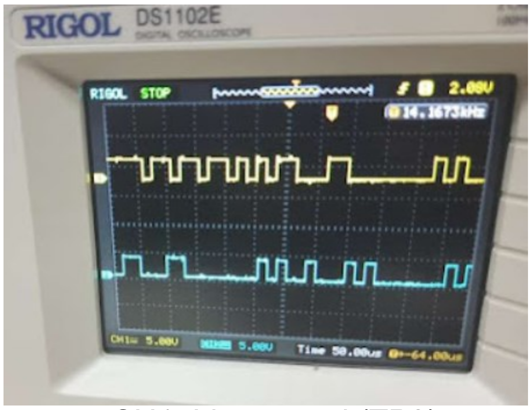
Arbitrary /1.5KHz	Channel2	16 KHz	 CH1: Input Signal (TP2), CH2: Sampled Output(TP12)
Square/500 Hz	Channel3	8 KHz	 CH1: Input Signal (TP3), CH2: Sampled Output(TP17)

Arbitrary /1.5KHz	Channel4	8 KHz	 CH1: Input Signal (TP4), CH2: Sampled Output(TP22)

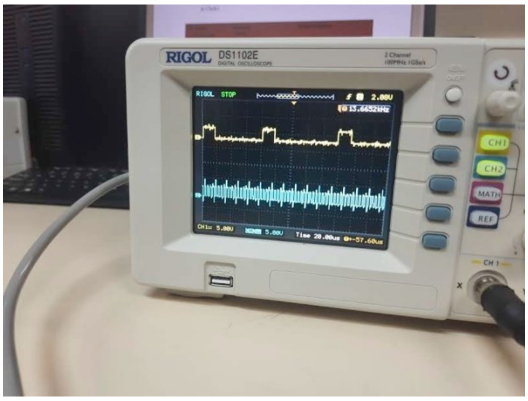
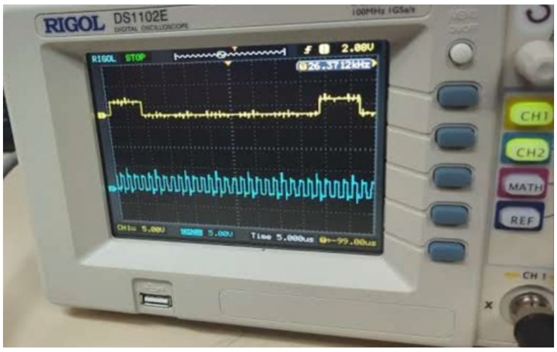
Experiment 9

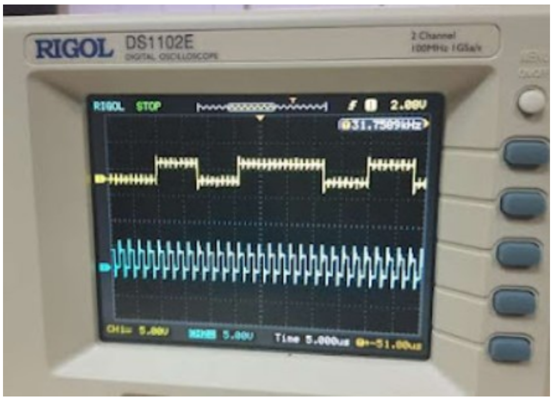
I/P Signal Type/ Freq.	Channel	Sampling Freq.	Sampled Output
Sine/500 Hz	Channel1	8 KHz	 CH1: Input Signal (TP5), CH2: PCM Output (TP9)

Arbitrary/ 500Hz	Channel2	8 KHz	 CH1: Input Signal (TP10), CH2: PCM Output (TP14)
Square/500 Hz	Channel3	8 KHz	 CH1: Input Signal (TP15), CH2: PCM Output (TP19)
Arbitrary /1.0KHz	Channel4	16 KHz	 CH1: PCM Output (TP22), CH2: Sampled Signal (TP24)

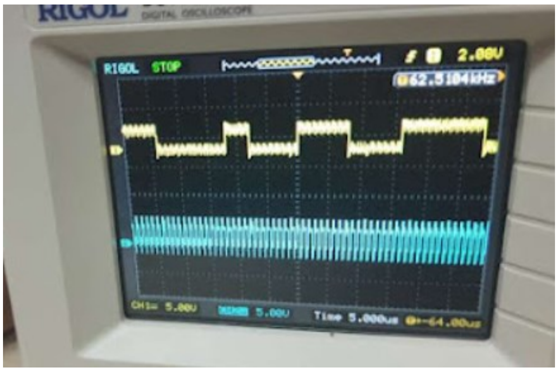
Sine/500 Hz	Channel 1	8 KHz	 CH1: Line speed (TP8), CH2: PCM Output (TP9)
Sine/1. 5 KHz	Channel 2	8 KHz	 CH1: Line speed (TP13), CH2: PCM Output (TP14)
Sine/500 Hz	Channel 3	8 KHz	 CH1: Line speed (TP18), CH2: PCM Output (TP19)

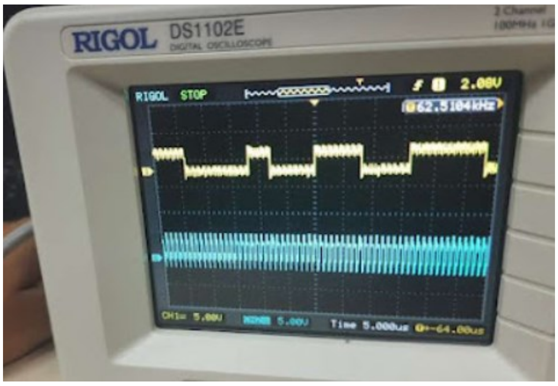
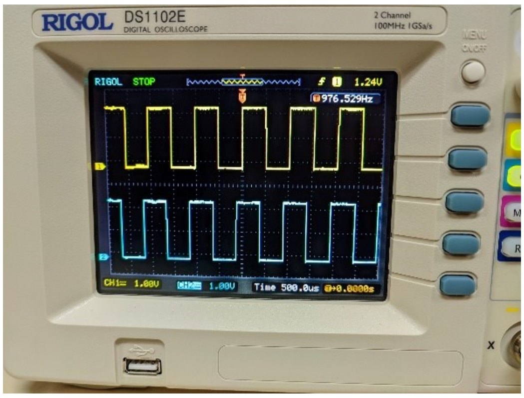
Experiment 10

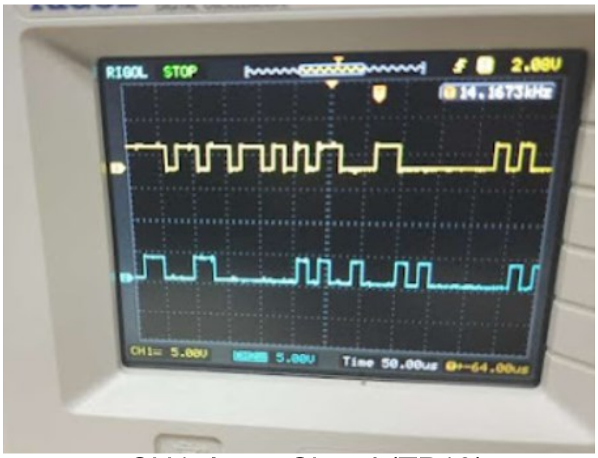
I/P Signal Type/ Freq.	Samplin g Freq.	Sampled Output
Sine/500 Hz	8 KHz	 CH1: Clock for 33 bits frame(TP clock1), CH2: 33 bits frame (TP 26)
Sine/500 Hz	16 KHz	 CH1: Clock for 33 bits frame (TP clock1), CH2: 33 bits frame (TP 26)

Sine/500 Hz	32 KHz	 CH1: Clock for 33 bits frame (TP clock1), CH2: 33 bits frame (TP 26)

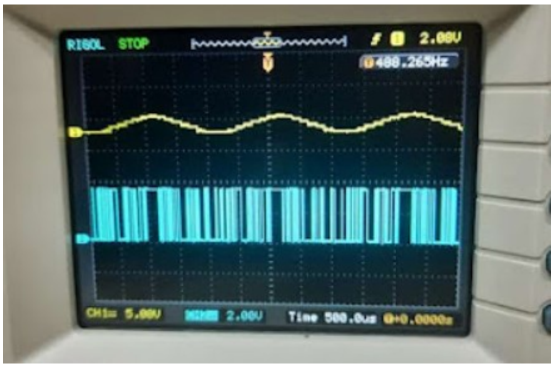
Eriment 11

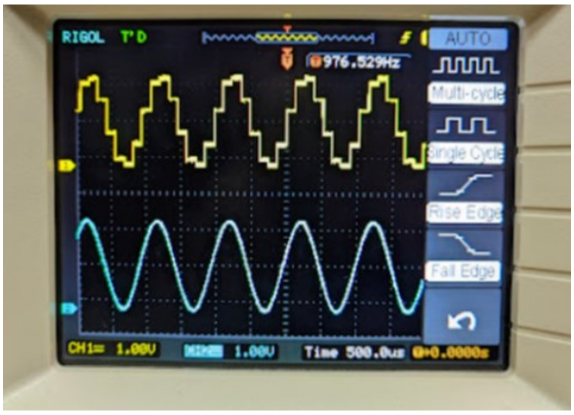
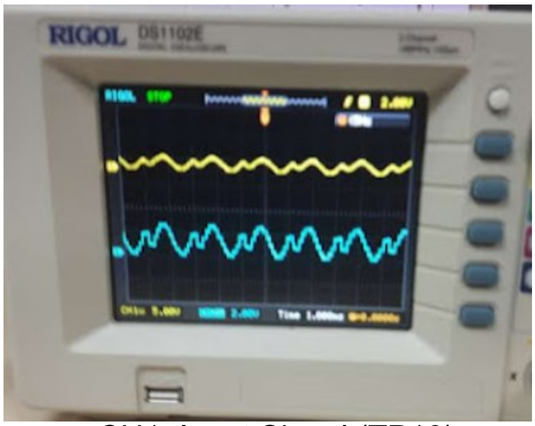
I/P Signal Type/ Freq.	Sampling Freq.	Sampled Output
Sine/500 Hz	8 KHz	 CH1: Clock for 32 bits frame (TP clock2) CH2: 32 bit Frame (TP 27)
Sine/500 Hz	16 KHz	 CH1: Clock for 32 bits frame (TP clock2) CH2: 32 bit Frame (TP 27)

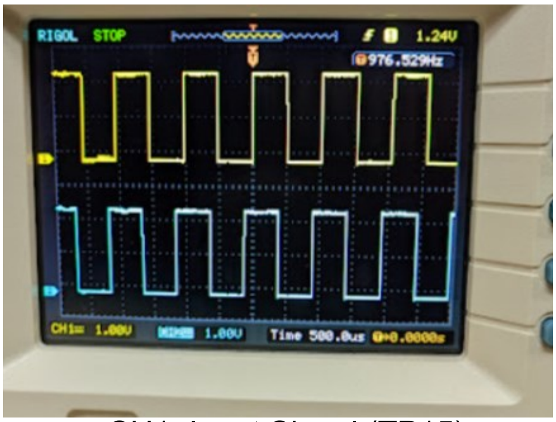
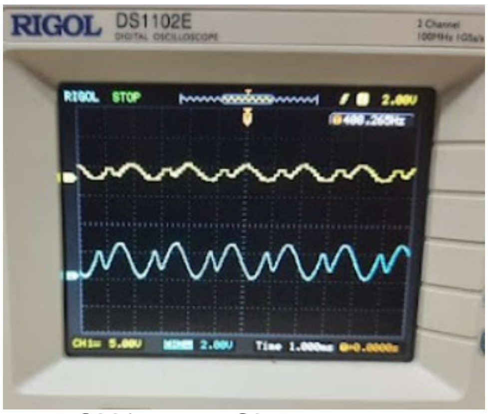
Sine/500 Hz	32 KHz	 CH1: Clock for 32 bits frame (TP clock2) CH2: 32 bit Frame (TP 27)
Sine/500 Hz	8 KHz	 CH1: Clock for 33 bits frame (TP clock1) CH2: Clock for 32 bits frame (TP clock2)

Sine/500 Hz	Channel2	8 KHz	 CH1: Input Signal (TP14), CH2:PCM Output after Demultiplexer (TP32)
Sine/500 Hz	Channel3	8 KHz	 CH1: Input Signal (TP19), CH2:PCM Output after Demultiplexer (TP35)

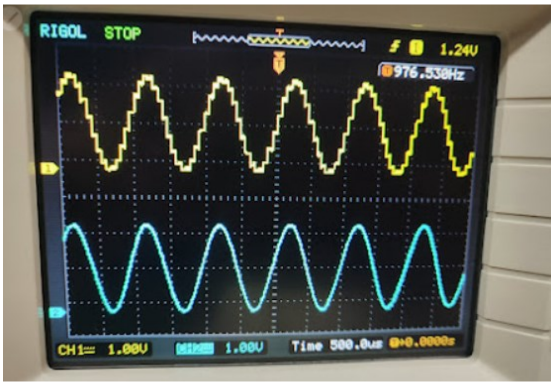
Experiment 13

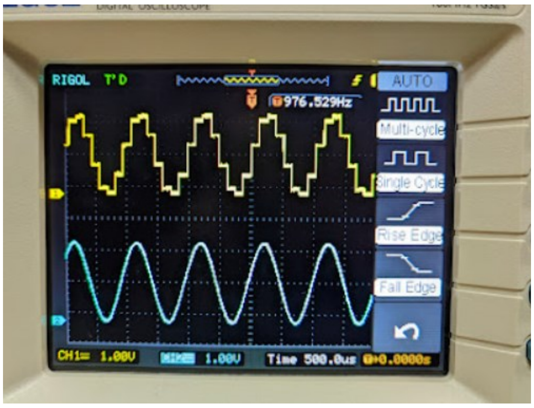
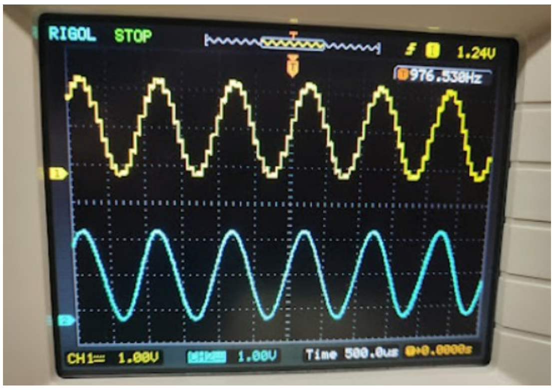
I/P Signal Type/ Freq.	Channel	Samplin g Freq.	Sampled Output
Sine/500 Hz	Channel1	8 KHz	 CH1: Input Signal (TP29), CH2: PCM Output (TP30)

Sine/500 Hz	Channel1	8 KHz	 CH1: Input Signal (TP5), CH2: Demodulated Output (TP30)
Arbitrary/500 Hz	Channel2	8 KHz	 CH1: Input Signal (TP10), CH2: Demodulated Output (TP33)

Square/50 0 Hz	Channel3	8 KHz	 CH1: Input Signal (TP15), CH2: Demodulated Output (TP36)
Arbitrary/500 Hz	Channel4	8 KHz	 CH1: Input Signal (TP20), CH2: PCM Output (TP39)

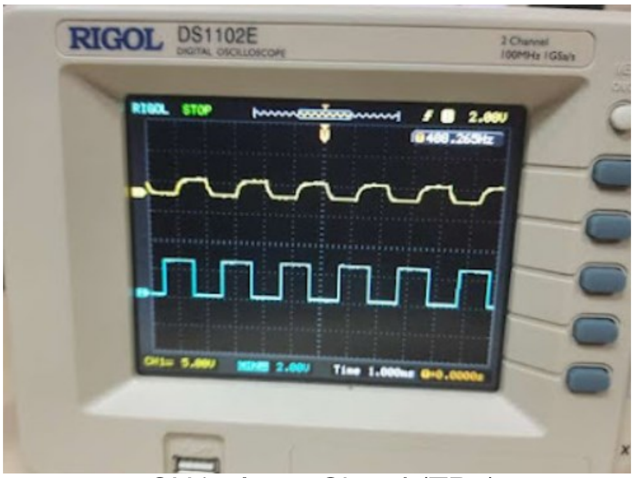
Experiment 14

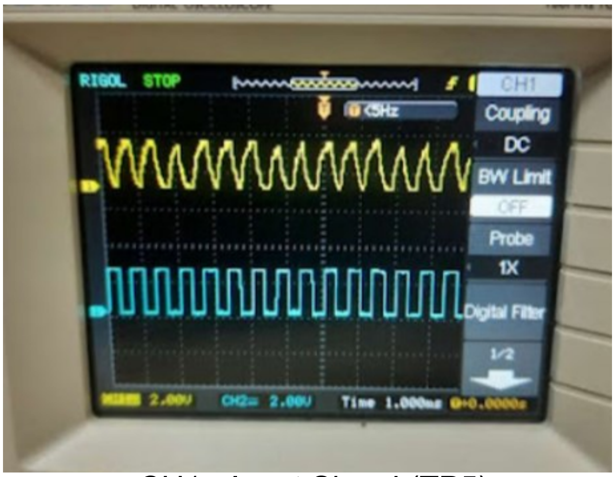
I/P Signal Type/ Freq.	Channel	Sampling Freq.	Sampled Output
Sine/500 Hz	Channel1	8 KHz	 CH1: Demodulated Output (TP30), CH2: Low Pass Filter Output (TP31)

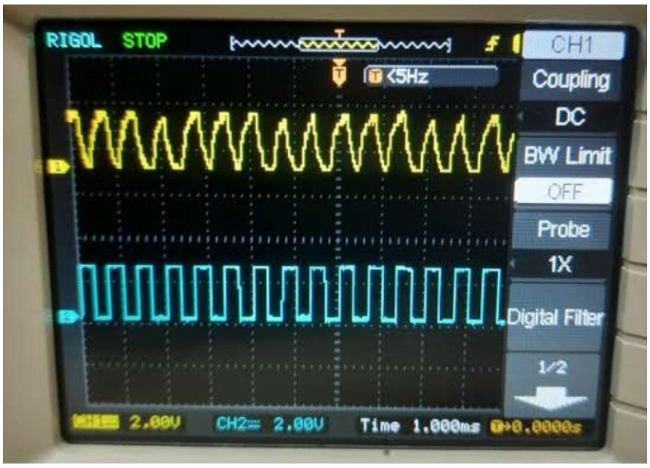
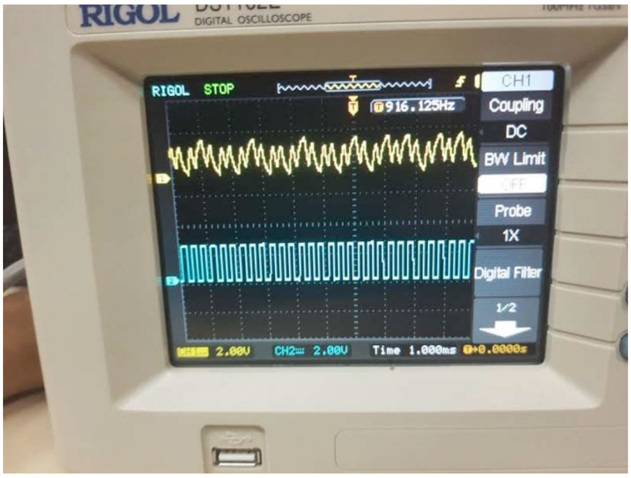
Sine/1.5 KHz	Channel1	8 KHz	 CH1: Demodulated Output (TP30), CH2: Low Pass Filter Output (TP31)
Sine/1.5 KHz	Channel2	8 KHz	 CH1: Demodulated Output (TP33), CH2: Low Pass Filter Output (TP34)
Sine/1.5 Hz	Channel2	16 KHz	 CH1: Demodulated Output (TP33), CH2: Low Pass Filter Output (TP34)

Square/50 0 Hz	Channel3	8 KHz	 CH1: Demodulated Output (TP36), CH2: Low Pass Filter Output (TP37)

Sine/1.5 KHz	Channel3	8 KHz	 CH1: Demodulated Output (TP36), CH2: Low Pass Filter Output (TP37)
Arbitrary/50 0 Hz	Channel4	8 KHz	 CH1: Demodulated Output (TP39), CH2: Low Pass Filter Output (TP40)

Square / 500 Hz	Channel 1	8 KHz	 CH1: Input Signal (TP5), CH2: Low Pass Filter Output (TP31)
Square / 1 KHz	Channel 1	8 KHz	 CH1: Input Signal (TP5), CH2: Low Pass Filter Output (TP31)

Square / 1.5 KHz	Channel 1	8 KHz	 CH1: Input Signal (TP5), CH2: Low Pass Filter Output (TP31)

Square / 2 KHz	Channel 1	8 KHz	 CH1: Input Signal (TP5), CH2: Low Pass Filter Output (TP31)
Square / 3 KHz	Channel 1	8 KHz	 CH1: Input Signal (TP5), CH2: Low Pass Filter Output (TP31)

Note :- Above results are shown filter effects of square wave w.r.t frequency . Type of low-pass filter at the receiver end is 2nd order butterworth active filter with 3-db, cut-off frequency 5 KHz. If you observe the signal at the output of the DAC i.e. input of the filter, you will see the proper square wave. As we have used low-pass filter with cut-off frequency 5 KHz so you are getting curved shape square wave due to the RC effect of the filter at maximum input frequency option i.e. 3