Bhargav S

1BM18EC025

VI Sem 'A2' Batch

1. To write MATLAB code to generate a PCM Signal and reconstruct the original signal back from it.

<u>Input Signal:</u> `sin(2*pi*t)-sin(6*pi*t)`

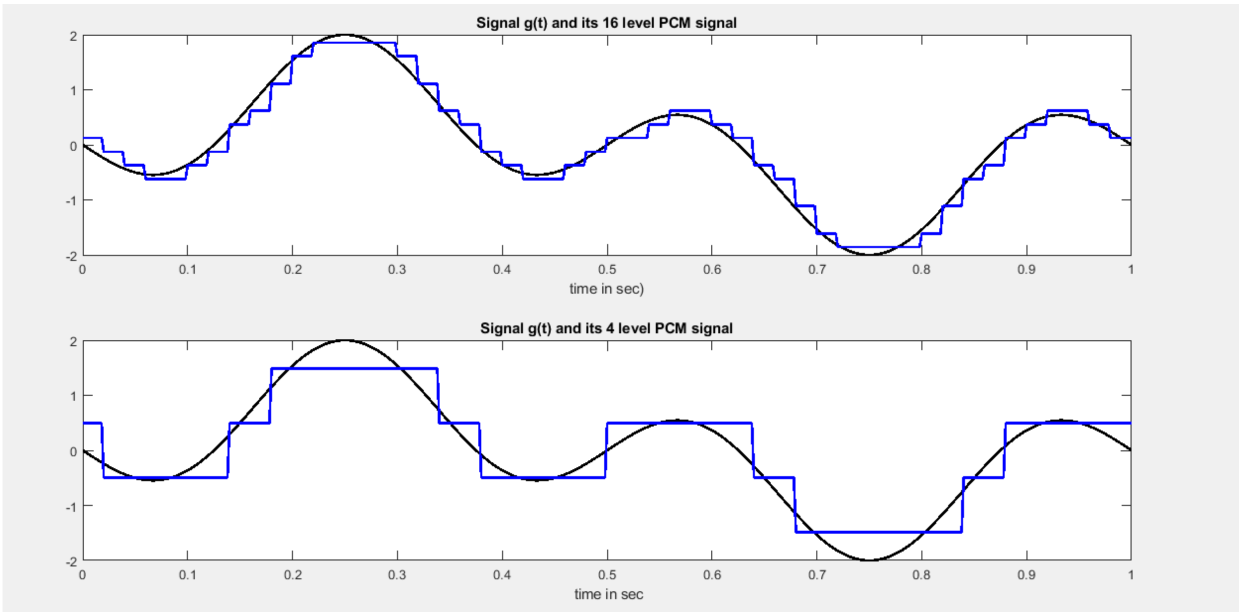
<u>MATLAB Code:</u>

```
clear;
clf;
td = 0.002;
t = 0:td:1.;
xsig = sin(2*pi*t)-sin(6*pi*t);
Lsig = length(xsig);
Lfft = 2 ^ ceil ( log2 ( Lsig) + 1 ) ;
Xsig = fftshift ( fft (xsig , Lfft ) );
Fmax= 1 / ( 2 * td) ;
%Faxis = linspace( -Fmax , Fmax , Lfft ) ;
ts = 0.02;
Nfact=ts / td;
[s_out , sq_out , sqh_out1 , Delta , SQNR] =sampandquant ( xsig , 16 , td ,
ts );
figure (1) ;
subplot (2,1,1) ; sfig1 = plot(t, xsig , 'k' , t , sqh_out1(1:Lsig ) , ' b
');
set(sfig1,'Linewidth',2);
title ( ' Signal g(t) and its 16 level PCM signal ')
xlabel ( ' time in sec) ' );
[ s_out , sq__out , sqh_out2 , Delta , SQNR] = sampandquant ( xsig, 4,td , ts
) ;
subplot (2,1,2) ; sfig2 = plot (t,xsig , 'k' , t , sqh_out2(1:Lsig) , ' b ' )
;
set (sfig2 ,'Linewidth' ,2);
title ( ' Signal g(t) and its 4 level PCM signal ')
xlabel ( ' time in sec' );

Lfft=2 ^ ceil ( log2 ( Lsig) +1) ;
Fmax= 1/ ( 2 * td ) ;
Faxis=linspace ( -Fmax , Fmax , Lfft ) ;
SQH1 = fftshift ( fft ( sqh_out1 , Lfft) );
SQH2 = fftshift ( fft ( sqh_out2 , Lfft) );
BW=10 ;
H_lpf=zeros (1, Lfft ) ; H_lpf ( Lfft/2-BW : Lfft/2+BW- 1 ) = 1;
S1_recv = SQH1.*H_lpf ;
s_recv1=real ( ifft ( fftshift( S1_recv) ));
```

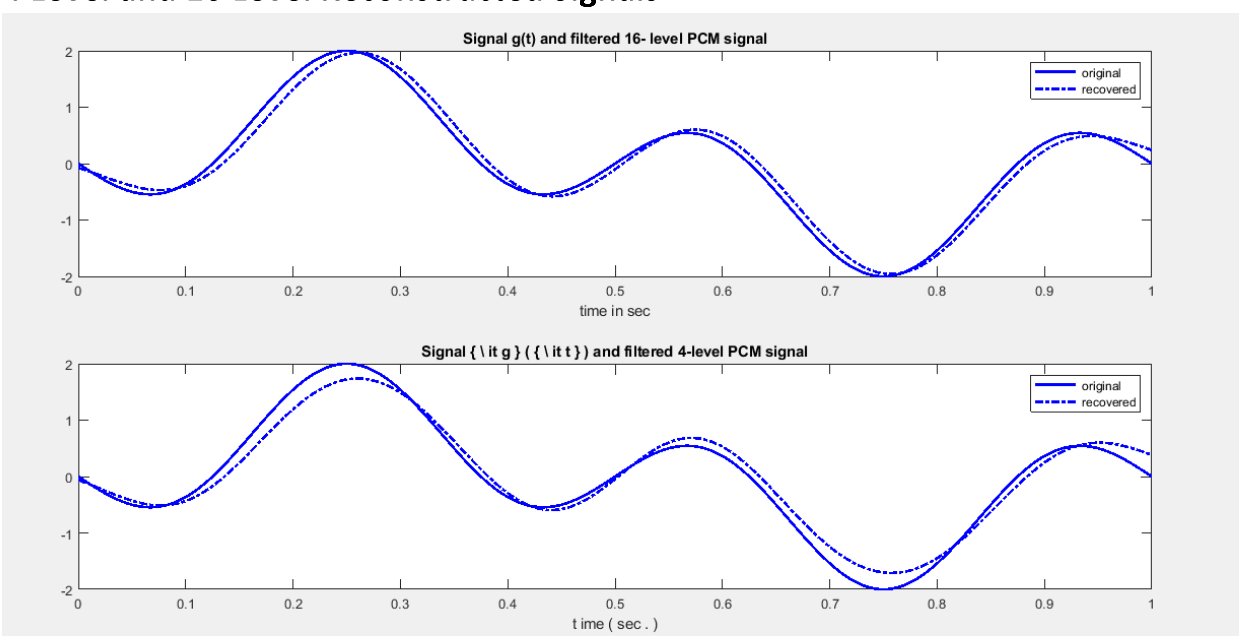
```
s_recv1 = s_recv1 (1:Lsig) ;
S2_recv= SQH2.*H_lpf;
s_recv2 =real ( ifft ( fftshift ( S2_recv) ) );
s_recv2 = s_recv2 (1:Lsig) ;
figure (2)
subplot (2,1,1); sfig3 = plot (t,xsig, ' b-' , t , s_recv1 , ' b-. ');
legend ( ' original ', ' recovered ')
set ( sfig3 , 'Linewidth' ,2) ;
title ( ' Signal g(t) and filtered 16- level PCM signal ')
xlabel ( ' time in sec ' ) ;
subplot (2,1,2) ; sfig4 = plot (t,xsig , ' b-' , t , s_recv2 (1:Lsig) , ' b-.
' ) ;
legend ( ' original ', ' recovered ')
set ( sfig4 , ' Linewidth ' ,2) ;
title ( ' Signal { \ it g } ( { \ it t } ) and filtered 4-level PCM signal ')
xlabel ( ' t ime ( sec . ) ' ) ;




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
end




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


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 ;
%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
```

Output Waveforms

- **16 Level and 4 Level PCM Signals**



- **4 Level and 16 Level Reconstructed Signals**



Observations:

- Sampling frequency: 500Hz
- L = 16 uniform quantization levels
- The resulting PCM signal is shown in fig 1. This PCM signal is low-pass-filtered at the receiver and compared against the original message signal, as shown in fig.1 and 2.
- The recovered signal is similar to the original signal g(t).
- We can conclude that a smaller number of quantization levels (L = 4) leads to a much larger approximation error.
- From the two reconstructed signals, we observe that the reconstructed signal matches the input signal to a greater extent when the number of levels of quantization increases.
- This is because for every level increase, SQNR = (6n + 1.8) dB increases by 6dB.

Result:

4-Level and 16-Level PCM signals are constructed and reconstructed signals are obtained and compared with the input.

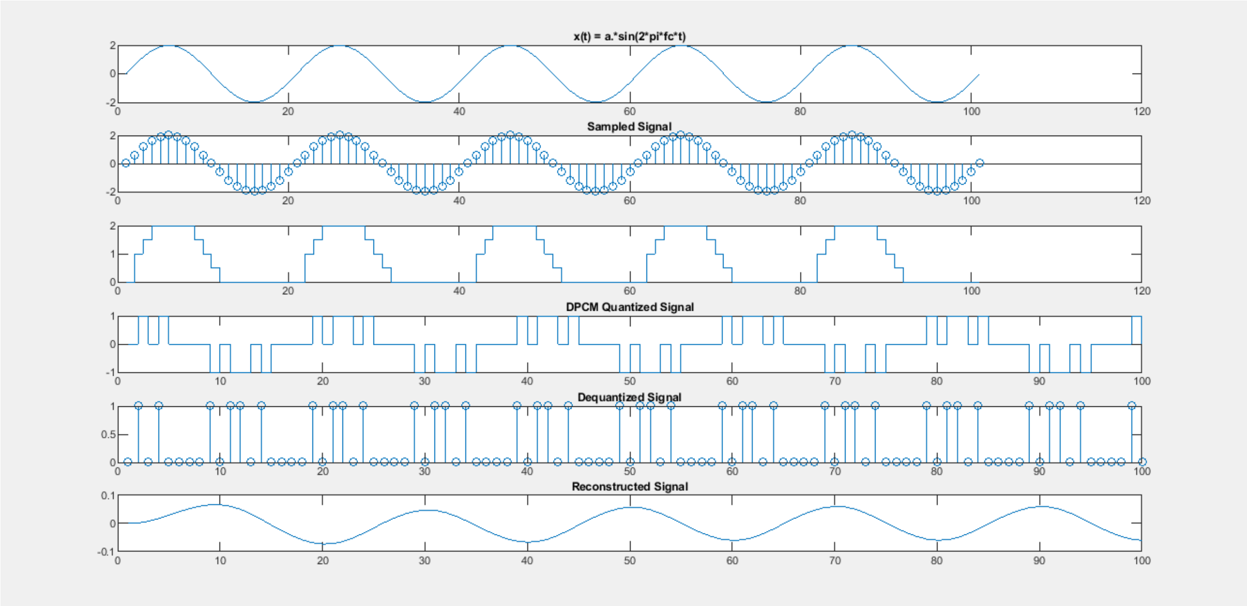
2. To write MATLAB code to generate a DPCM Signal and reconstruct the original signal back from it.

MATLAB Code:

```
clc;
clear all;
close all;
t=0:0.01:1;%time interval of 1 second
a=2;
fc=5;
x=a*sin(2*pi*fc*t);
subplot(6,1,1);
plot(x)
title('x(t) = a.*sin(2*pi*fc*t)');
subplot(6,1,2);
stem(x)
l=zeros(1,100,'int32');
m=zeros(1,100,'int32');
title('Sampled Signal');
```

```
[index,quants] = quantiz(x,[0:0.5:4],[0:0.5:4.5]);
for N=1:length(t)
m(N)=x(N);
end
for N=2:99
if m(N-1)<m(N)
l(N)=1;
elseif m(N-1)==m(N)
l(N)=0;
else
l(N)=-1;
end
end
subplot(6,1,3);
stairs(quants);
subplot(6,1,4);
stairs(l)
title('DPCM Quantized Signal');
z1=dec2bin(abs(l));
z2=bin2dec(z1);
subplot(6,1,5)
stem(z2);
title('Dequantized Signal')
[b,a]=butter(2,0.03,'low');
k=filter(b,a,l);
subplot(6,1,6);
plot(k)
title('Reconstructed Signal');
```

Output Waveforms:



Observations:

- The error in reconstructed signal is less compared to PCM.
- The bandwidth consumed by a DPCM signal is lesser than that of a PCM signal.
- Importance of DPCM in solving errors resulting from commonly used modulation techniques as the Pulse Coded Modulation observed.

Result:

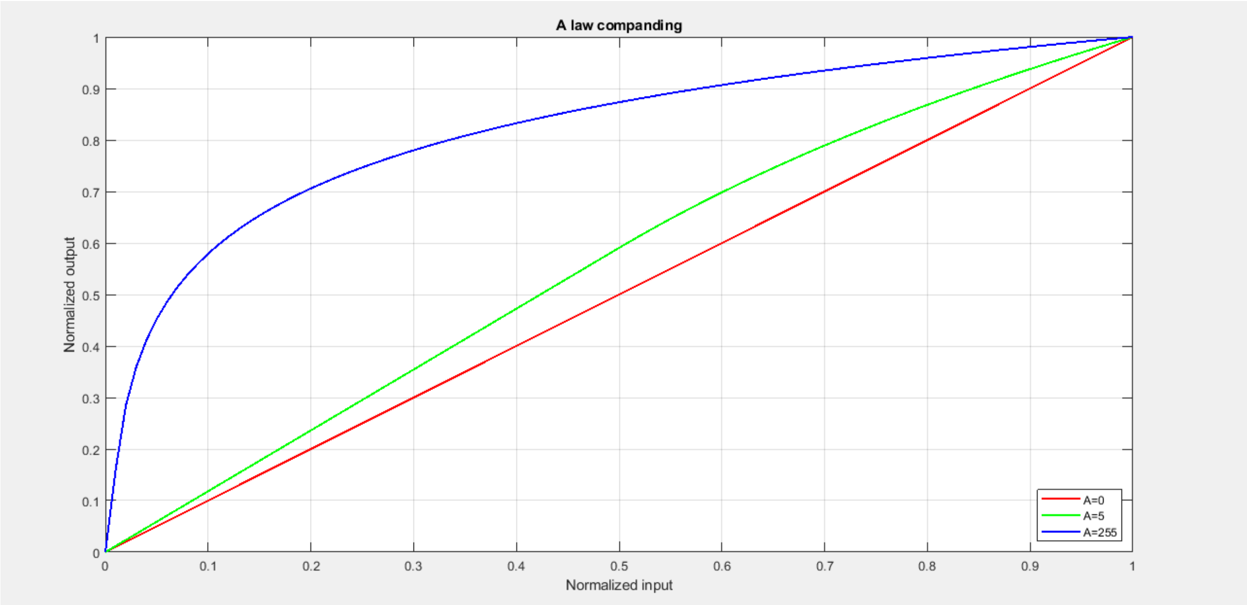
DPCM signal is produced and the output signal is reconstructed and compared with the input.

3. To demonstrate A-law Companding Technique.

MATLAB Code

```
% non-uniform quantization: A law companding
clc;
clear all;
close all;
% when A=1, no compression occurs and input=output
% when A > 1, compression is more
x=0:0.01:1; %range of x is from 0 to 1 with step size=0.20
a=[1 2 87.56]; %A values considered for evaluation
for i=1: length(a)
    for j=1:length(x)
        if x(j)>=0&&x(j)<=(1/a(i))
            y(i,j)=(a(i)*x(j))/(1+log(a(i)));
        elseif x(j)>(1/a(i))&&x(j)<=1
            y(i,j)=(1+log(a(i)*x(j)))/(1+log(a(i)));
        end
    end
end
plot(x,y(1,:),'r','linewidth',1.5);
hold on;
plot(x,y(2,:),'g','linewidth',1.5);
plot(x,y(3,:),'b','linewidth',1.5);
grid on;
legend('A=0','A=5','A=255','location','southeast');
xlabel('Normalized input');
ylabel('Normalized output');
title('A law companding');
```

Output Waveforms



Observation:

- The dynamic range capability of the compander improves with the increase in A.
- The SNR for low-level signals increases at the expense of the SNR for high-level signals.
- To accommodate these two conflicting requirements a compromise is usually made in choosing the value of parameter A.
- The typical value used in practice is A = 87.6.

Result:

A-law normalized input vs normalized output curve is plotted and studied.

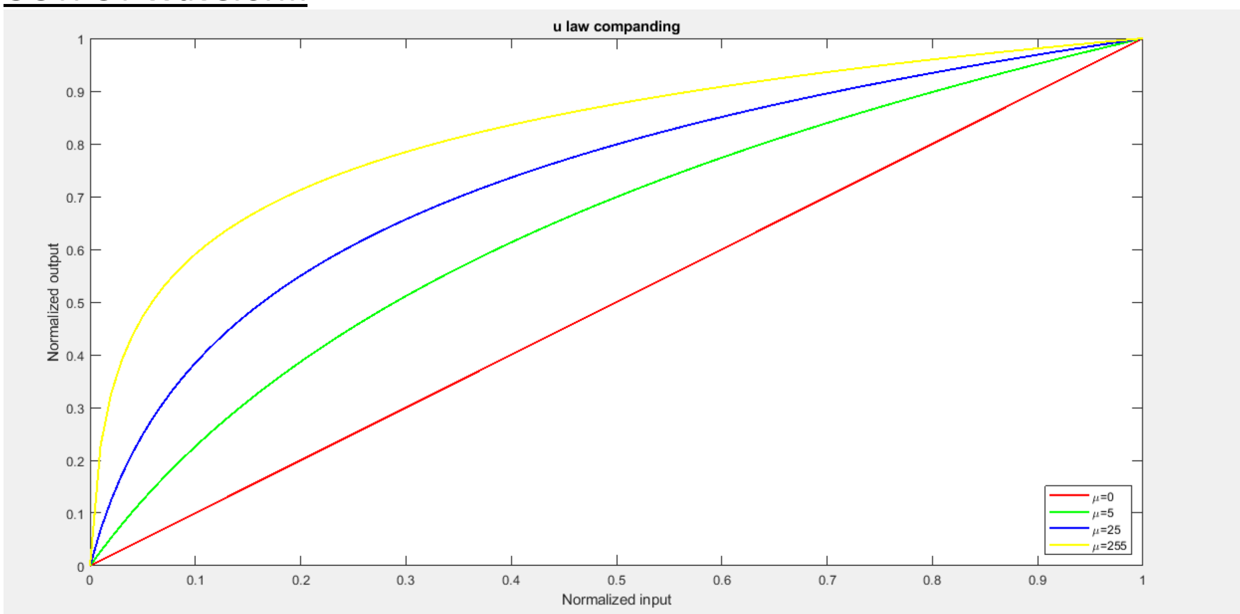
4. To demonstrate μ Law Companding Technique.

MATLAB Code

```
% non-uniform quantization: u law companding
clc;
clear all;
close all;
x=0:0.01:1; %range of x is from 0 to1 with step size=0.1
mu=[0 5 25 255];%u values considered for evaluation
%when u=0, there is no compression (input = output)
% more the value of u, more is the compression
```

```
for i=1:length(mu)
    for j=1:length(x)
        if mu(i)==0
            y(i,j)=x(j);
        else
            y(i,j)=log(1+(mu(i)*x(j)))/log(1+mu(i));
        end
    end
end
plot(x,y(1,:),'r','linewidth',1.5);
hold on;
plot(x,y(2,:),'g','linewidth',1.5);
plot(x,y(3,:),'b','linewidth',1.5);
plot(x,y(4,:),'y','linewidth',1.5);
legend('\mu=0','\mu=5','\mu=25','\mu=255','location','southeast');
xlabel('Normalized input');
ylabel('Normalized output');
title('u law companding');
```

OUTPUT Waveform



Observations

- The dynamic range capability of the compander improves with the increase in μ .
- The SNR for low-level signals increases at the expense of the SNR for high-level signals.
- To accommodate these two conflicting requirements a compromise is usually made in choosing the value of parameter μ .
- The typical value used in practice is μ = 255.

- The μ -law algorithm provides a slightly larger dynamic range than the A-law.

Result

μ -law normalized input vs normalized output curve is plotted and studied