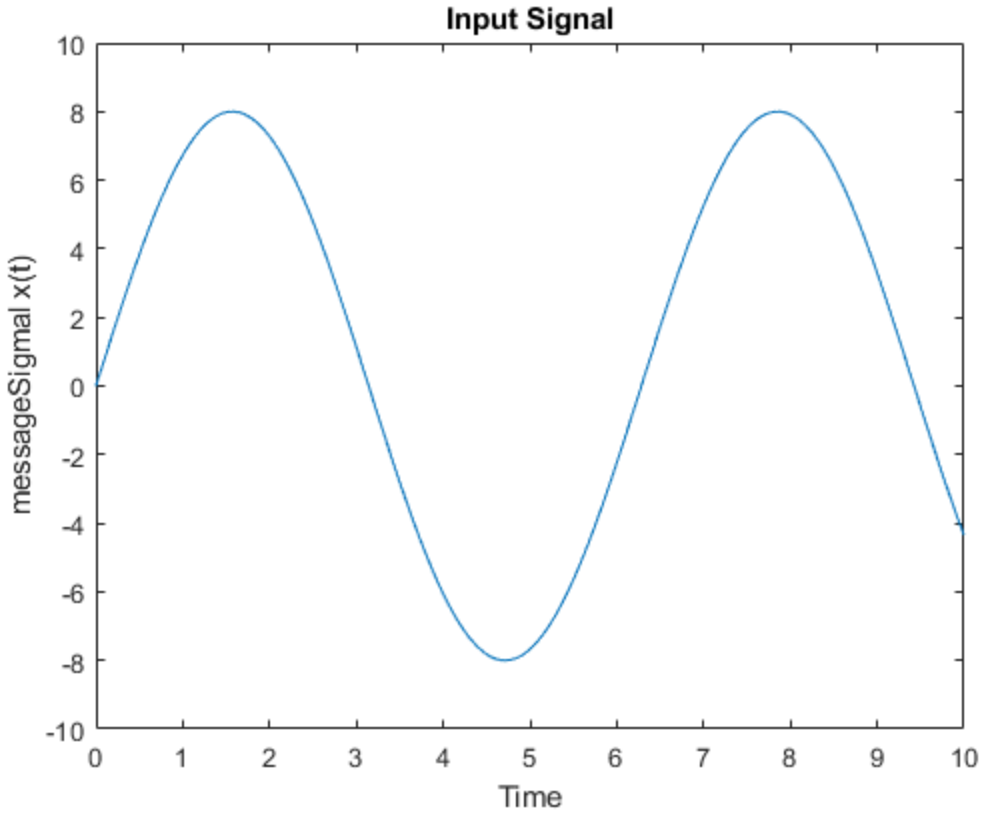
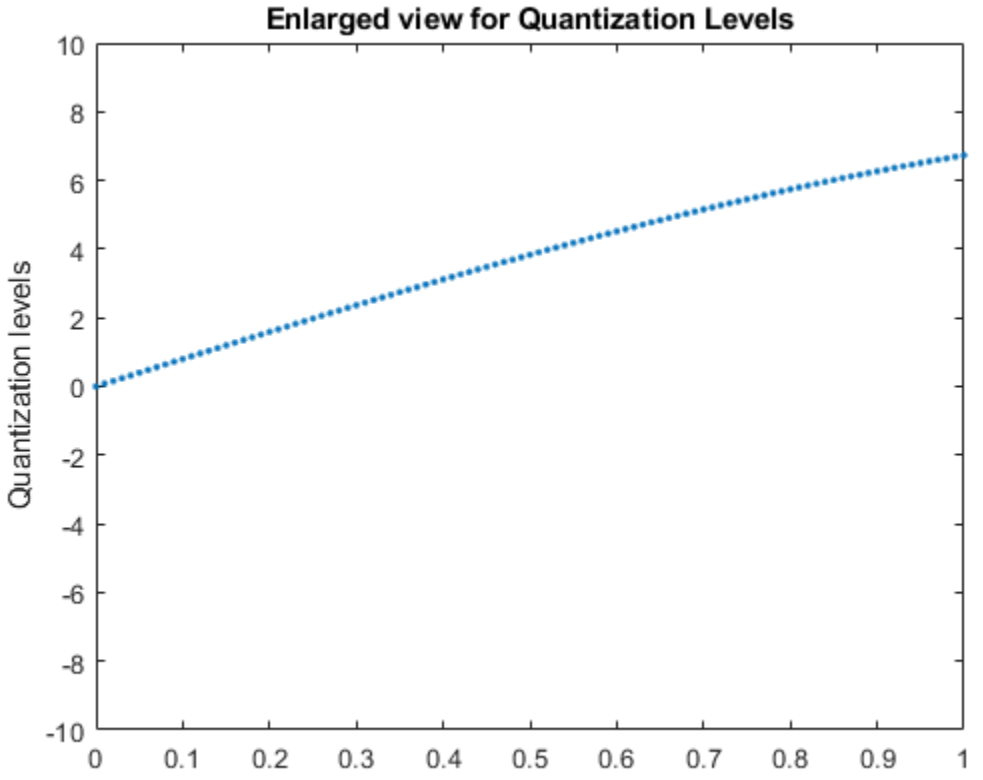
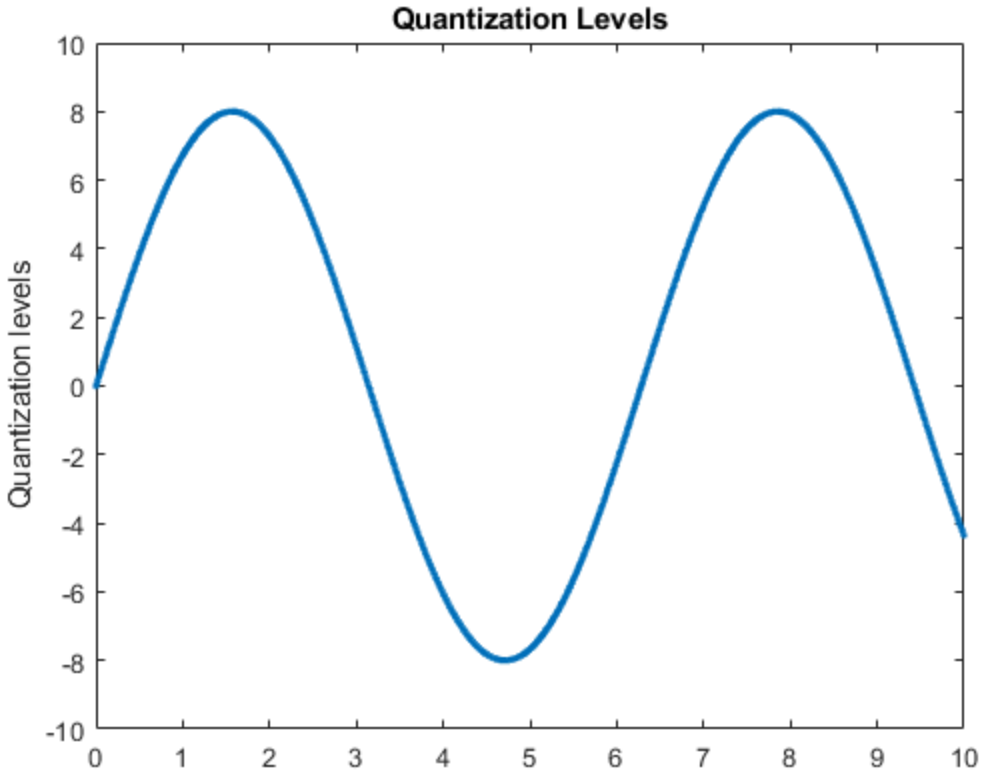
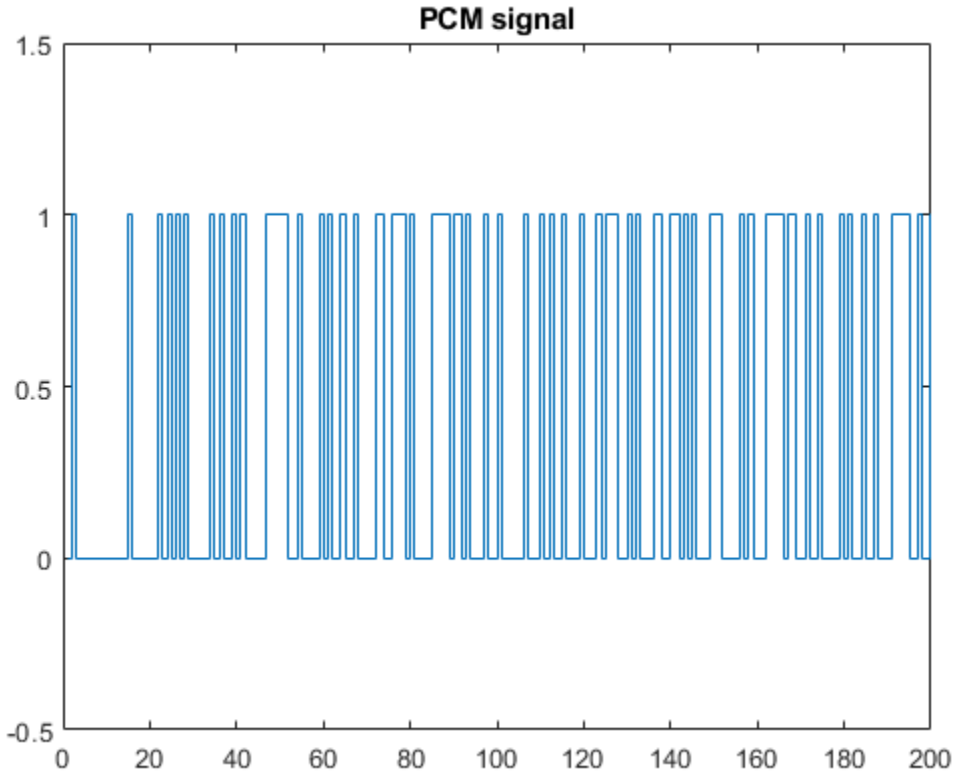
```
%PCS Assignment4
%Name-Jaskirath Singh
%Roll No - 2018150

%Question3

samplingFreq = 100;                          %100Hz sampling frequency
timeStamp = 0:1/samplingFreq:10;
amplitude = 8;
messageSignal = amplitude*sin(timeStamp);  % it is 8sin(t)

%plotting of message signal
figure(1);
plot(timeStamp,messageSignal);
title("Input Signal");
ylim([-10,10]);
ylabel("messageSigmal x(t)");
xlabel("Time")


%QUANTIZING
noOfBits = 12;
quantizedLevels = 2^noOfBits;
%as our amplitude range from -m to +m and it divided into L sublevels
%size of each level is given as 2*m/l
delta = (2*amplitude)/quantizedLevels;      %size of each level
l_array = (-1*amplitude):delta:amplitude;
average_l_array = ((-1*amplitude)-(delta/2)):delta:amplitude
+(delta/2);

%we will be using inbuilt quantixation function for our signal

%description of function
%index = quantiz(sig,partition) returns the quantization levels in
 the real vector signal sig using the parameter partition. partition
 is a real vector whose entries are in strictly ascending order. If
 partition has length n, index is a vector whose kth entry is
%0 if sig(k) ? partition(1)
%m if partition(m) < sig(k) ? partition(m+1)
%n if partition(n) < sig(k)
%[index,quants] = quantiz(sig,partition,codebook) is the same as
 the syntax above, except that codebook prescribes a value for each
 partition in the quantization and quants contains the quantization of
 sig based on the quantization levels and prescribed values. codebook
 is a vector whose length exceeds the length of partition by one.
 quants is a row vector whose length is the same as the length of sig.
 quants is related to codebook and index by
%quants(ii) = codebook(index(ii)+1);
%where ii is an integer between 1 and length(sig)

[index,quants] = quantiz(messageSignal,l_array,average_l_array);
figure(2);
```

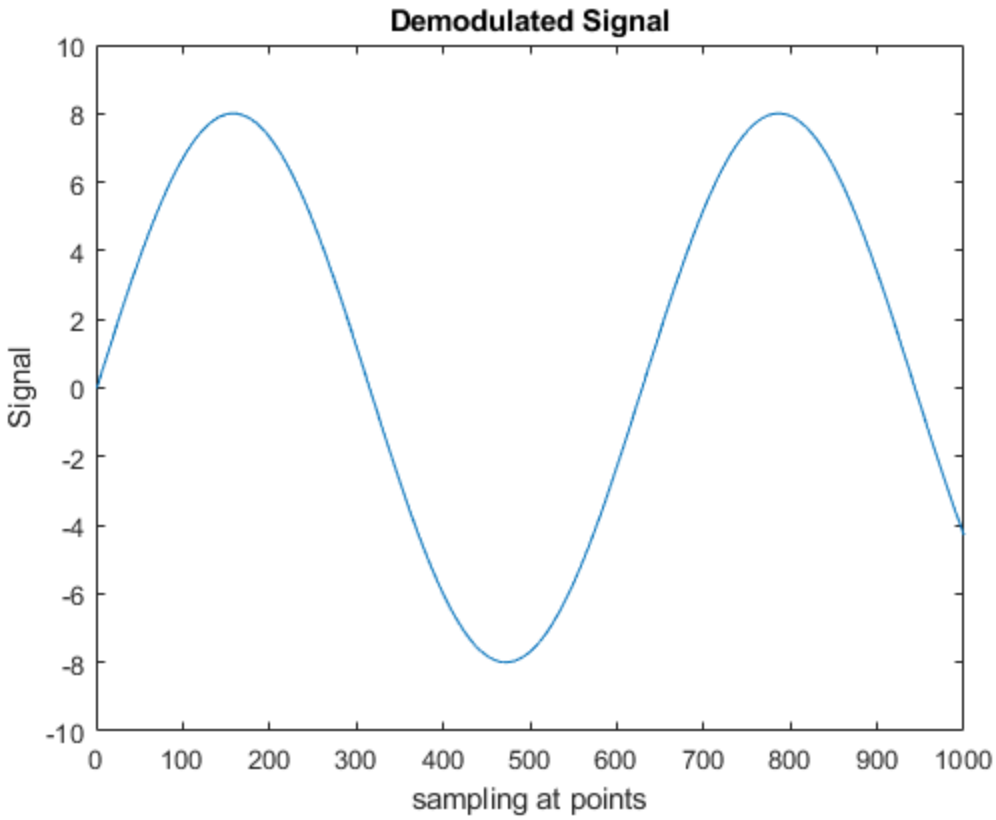
```
plot(timeStamp,quants,".");
title("Enlarged view for Quantization Levels");
xlim([0,1]);
ylim([-10,10]);
ylabel("Quantization levels");

figure(3);
plot(timeStamp,quants,".");
title("Quantization Levels");
ylim([-10,10]);
ylabel("Quantization levels");

%"de2bi" built in function to converts decimal number to binary number
%b = de2bi(d,...,flg) uses flg to determine whether the first column
 of
%b contains the lowest-order or highest-order digits.

binaryCode = de2bi(index,'left-msb');           %using our index value
bcdVector = binaryCode';
pcm = bcdVector(:)';                            %converting matrix
 into vector
figure(4);
stairs(pcm);
ylim([-0.5,1.5]);
xlim([0,200]);
title("PCM signal");

%DEMODULATION
%as we will only receive a binary message we will have to decode it

%processing of received signal
columnsInResult = length(pcm)/(noOfBits+1);
rowsInResult = noOfBits+1;

%The "reshape function" returns a new array with n rows and m columns
%(n*m must equal the number of elements in the original array).

receivedSignal = reshape(pcm,rowsInResult,columnsInResult);
receivedIndex = bi2de(receivedSignal','left-msb');      % Binary to
 Decimal

%main demodulation part
receivedQuants = (delta*receivedIndex)-amplitude+(delta/2);

figure(5);
plot(receivedQuants);
title('Demodulated Signal');
ylabel('Signal');
xlabel('sampling at points');
ylim([-10,10]);
xlim([0;1000]);


%Thus our dequantised signal is same as our messagle signal
```


Input Signal
messageSigmal x(t)
Time
10
8
6
4
2
0
-2
-4
-6
-8
-10
0
1
2
3
4
5
6
7
8
9
10


Enlarged view for Quantization Levels
Quantization levels
10
8
6
4
2
0
-2
-4
-6
-8
-10
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Quantization Levels
Quantization levels
10
8
6
4
2
0
-2
-4
-6
-8
-10
0
1
2
3
4
5
6
7
8
9
10


PCM signal
1.5
1
0.5
0
-0.5
0
20
40
60
80
100
120
140
160
180
200



Published with MATLAB® R2019b