```
%Assignment
%Digital Communication System
%Group 8 (Rahul(2019191), Varun(2018203),Ankit(2019146))
clc;
clear;
close all;
[audio,Fs,info]= input();
[audio_less,audio_more] = source(audio);
%Spectrum of both audio_less and audio_more
figure,
subplot(2,1,1), plot(abs(fftshift(fft2(audio_less)))),title('Magnitude
 Spectrum with frequency less than Nyquist rate');
subplot(2,1,2), plot(abs(fftshift(fft2(audio_more)))),title('Magnitude
 Spectrum with frequency more than Nyquist rate');
%For Audio_less-------------------------------------------------------
disp('-------------EveryThing For frequency less than Nyquist
 rate--------------------------');
[quant_less,step_size_less,level_val_less]= quantize(audio_less);
[bitstream_less,bitstream_length_less,huff_code_less] =
 encode(quant_less,step_size_less);
[pulse_arr_less] = pulseshaping(bitstream_less,bitstream_length_less);
[row_less,col_less] = size(audio_less);
quantized_decode_level_less =
 receivedecode(bitstream_length_less,pulse_arr_less,huff_code_less,row_less);
decoded_quantized_signal_less =
 tablelookup(quantized_decode_level_less,level_val_less);
subplot(2,1,1),plot(quant_less),title('Encoded Quantized signal');
subplot(2,1,2),plot(decoded_quantized_signal_less),title('Decoded quantized
 signal');
filename = 'received.wav';
audiowrite(filename,decoded_quantized_signal_less,Fs/4);
filename2 = 'send.wav';
audiowrite(filename2,quant_less,Fs/4);
%-------------------------------------------------------------------------
%For Audio More-------------------------------------------------------
disp('-------------EveryThing For frequency more than Nyquist
 rate--------------------------');
[quant_more,step_size_more,level_val_more]= quantize(audio_more);
[bitstream_more,bitstream_length_more,huff_code_more] =
 encode(quant_more,step_size_more);
[pulse_arr_more] = pulseshaping(bitstream_more,bitstream_length_more);
[row_more,col_more] = size(audio_more);
quantized_decode_level_more =
 receivedecode(bitstream_length_more,pulse_arr_more,huff_code_more,row_more);
decoded_quantized_signal_more =
 tablelookup(quantized_decode_level_more,level_val_more);
subplot(2,1,1),plot(quant_more),title('Encoded quantized signal');
subplot(2,1,2),plot(decoded_quantized_signal_more),title('Decoded quantized
 signal');
filename11 = 'received_more.wav';
audiowrite(filename11,decoded_quantized_signal_more,Fs*4);
filename21 = 'send_more.wav';
```

```
audiowrite(filename21,quant_more,Fs*4);
%-----------------------------------------------------------------------
%Here we are taking input mp3 file and then displaying the sampling
%frequency and then plotting the original signal, its magnitude and phase
%in time domain and then spectrum in frequency domain.
function  [output,Fs,info] = input()
  [output,Fs]= audioread('Single_short.mp3');
  info = audioinfo('Single_short.mp3');
  disp("Sampling Frequency in Hz "+ Fs);
  disp("Nyquist Frequency in Hz "+(Fs/2));
   disp('The maximum frequency component of the spectrum is 404.323 at
 76224.');
  subplot(2,2,1),plot(output),title('Audio Signal');
  subplot(2,2,2), plot(abs(output)),title('Magnitude in time domain');
  subplot(2,2,3), plot(angle(output)),title('Phase in time domain');
  subplot(2,2,4), plot(abs(fftshift(fft2(output)))),title('Spectrum in
 frequency domain');
end
%----------------------------------------------------------------------
%here we are downsampling and upsampling the audio. Downsampling done by
%factor of 1/4 and upsampling done by factor of 4.
function [audio_less,audio_more] = source(audio)
   [x,y] = size(audio);
  factor=4;
  audio_downsampled=[ ];
   for i=1:factor:x
      audio_downsampled=[audio_downsampled audio(i,1)];
  end
   audio_less = audio_downsampled.';
   factor=4;
  audio_upsampled=[ ] ;
  for i=1:x
      audio_upsampled=[audio_upsampled audio(i,1)];
      for j=1:factor-1
         audio_upsampled=[audio_upsampled 0];
      end
  end
   audio_more = audio_upsampled.';
end
%----------------------------------------------------------------------
%here we are quantizing the input audio amplitude in 32 levels. 1st level
%having the minimum amplitude and each subsequent level amplitude will
%increase by step_size. Also, here we are calculating MSE for each
%quantization levels and plotting it. Also, we are also assigning 5 bits to
%each level.
function [quan_new,step_size,level_val]= quantize(audio)
   quantized = audio;
  disp('We are using 32 levels for quantization means 5 bits per level');
  levels = 32;
   mx_am = max(quantized);
  mn_am = min(quantized);
   step_size = (mx_am-mn_am)/levels;

  [row,col] = size(quantized);
```

```
  %-------------------------Quantization and MSE Calculation
 ------------------
 mse1 =0;
  q_count = zeros(32,1);
 mse_count =zeros(32,1);
  quan_new = zeros(row,col);
  level_val = mn_am:step_size:mx_am-step_size;
 level_val = level_val.';
 for i=1:row

     near1 = 2;

     val1 = -1;

     ix = -1;
     c =0;
     for j = mn_am:step_size:mx_am-step_size
         dis1 = abs(j-quantized(i,1));
         c=c+1;
         if dis1<near1
             near1=dis1;
             val1 = j;
             ix =c;
         end

     end

     quan_new(i,1) = val1;
     q_count(ix,1)=q_count(ix,1)+1;
     mse1=mse1+(near1*near1);
     mse_count(ix,1)=mse_count(ix,1)+(near1*near1);
 end

 for i=1:32
     mse_count(i,1) = mse_count(i,1)/q_count(i,1);
 end
 mse1 = mse1/row;
  disp("MSE for this sampled audio is  " + mse1);

 figure,stem(mse_count),title('MSE vs Number of quantization levels');
  %------------------Bitstream Generating here -----------------
 bitstream = zeros(row,5);
 for i =1:row
     diff = (quan_new(i,1) + abs(mn_am))/step_size;
     bin = int2bit(diff, 5);
     for j = 1:5
         bitstream(i,j) = bin(j);
     end
 end
  totalbits =5*row;
 disp("Total Bits used in representation of one level is 5");
 disp("Total Bits used in Bitstream "+totalbits);
end
%------------------------------------------------------------------------
```

```
%Here we are first calculating the count of each levels occuring in the
%audio and then calculating probability of occurence of each level.
%Then we are assigning each level with huffman codes. So a level with
%high probability will be assigned with less bits and a level with low
%probability will be assigned with more bits.
function [bitstream,bitstream_length,huff_code] = encode(audio,step_size)
  mn_am = min(audio);
   sampled_and_quantized =audio;
   len = length(sampled_and_quantized);
  count_of_levels = zeros(32, 1);
  for i = 1:len
      quant_val = sampled_and_quantized(i,1);
      diff = 1 + (quant_val + abs(mn_am))/step_size;
      count_of_levels(diff, 1) = count_of_levels(diff, 1) + 1;
  end
   probability_values = zeros(32, 1);
  for i = 1:32
      probability_values(i, 1) = count_of_levels(i, 1)/len;
  end
  figure,stem(probability_values),title('Probability for different levels');
   huff_code = zeros(32,1);
   sum = 0;
  for i = 1:31
      max_val = -1;
      max_idx = -1;
      for j = 1:32
          if(probability_values(j, 1)>max_val)
              max_val = probability_values(j, 1);
              max_idx = j;
          end
      end
      probability_values(max_idx, 1) = -1;
      huff_code(max_idx,1)=sum;
      sum = sum + power(2, i);
  end
  sum = sum+1;
  sum = sum - power(2,31);
  last_idx =-1;
  for i = 1:31
      if(probability_values(i, 1)~=-1)
          last_idx = i;
      end
  end
  huff_code(last_idx,1)=sum;
  [row,col] = size(audio);
   bitstream_length=0;
   for i = 1:row
       quant_val = audio(i,1);
       level = 1 + (quant_val + abs(mn_am))/step_size;

       [ss,cc] = size(dec2bin(huff_code(level)));
       bitstream_length = bitstream_length + cc;
   end
```

```
    disp("The length of the bitstream for using huffman coding " +
 bitstream_length);
   bitstream_huff = zeros(bitstream_length,1);

   pos =1;

   for i = 1:row
       quant_val = audio(i,1);
       level = 1 + (quant_val + abs(mn_am))/step_size;
       binary_code = dec2bin(huff_code(level));
       [rr,cc] = size(binary_code);
        for j = 1:cc
          bitstream_huff(pos,1) = binary_code(1,j)-48;
          pos=pos+1;
       end
   end
   bitstream=bitstream_huff;
end
%-------------------------------------------------------------------
%Here we are obtaining raised cosine pulse in time domain for roll off
%factors of 0.25, 0.5 and 0.75. We observed that the ISI is decreasing as
%we increase the roll of factor.
%Then we are obtaining raised cosine pulse in frequency domain for same
%values of roll off factors.
function [pulse_arr] = pulseshaping(bitstream,bitstream_length)
 %--------------------Roll Of Factor = 0.25-----------------------------------
  roll_off = 0.25;
  time = 0;
  figure
  hold on
   test = 12;
  shif = 14;
  for i = 1:test
      pulse = pulse_shape_time(roll_off, bitstream(i+10000+shif, 1), time);
      plot(-50+time:0.01:50+time,pulse);
      time = time + 20;
  end
  hold off;
 %--------------------Roll Of Factor = 0.5-----------------------------------
  roll_off = 0.5;
  time = 0;
  figure
  hold on
  for i = 1:test
      pulse = pulse_shape_time(roll_off, bitstream(i+10000+shif, 1), time);
      plot(-50+time:0.01:50+time,pulse);
      time = time + 20;
  end
  hold off;
%--------------------Roll Of Factor = 0.75-----------------------------------
  roll_off = 0.75;
  time = 0;
  figure
  hold on
```

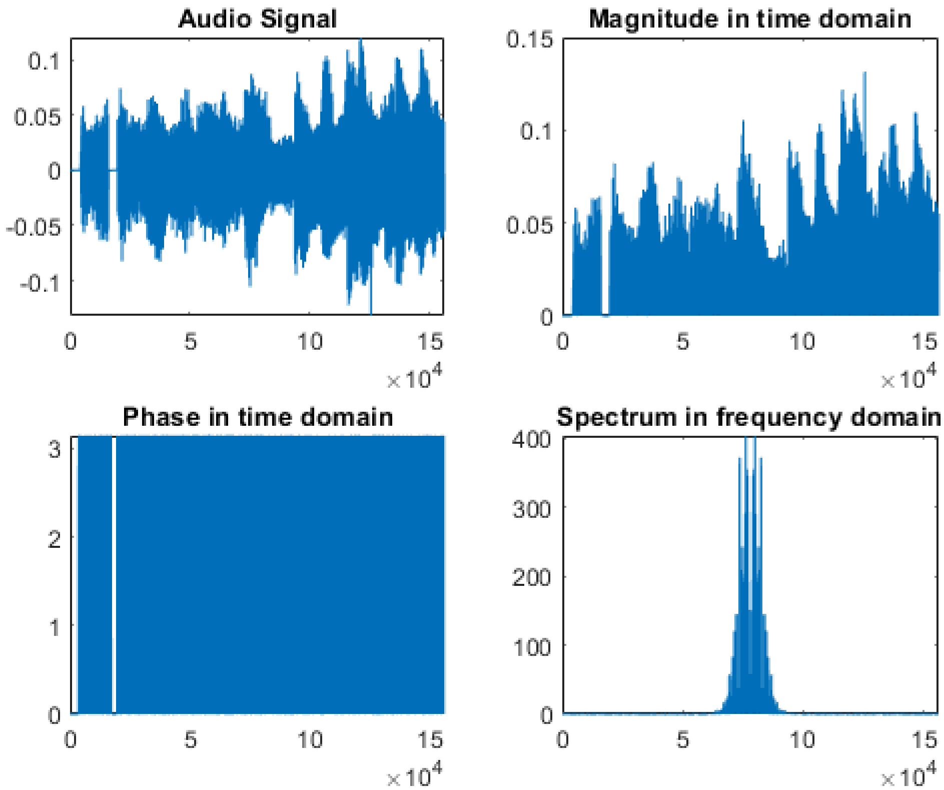
```
  for i = 1:test
      pulse = pulse_shape_time(roll_off, bitstream(i+10000+shif, 1), time);
      plot(-50+time:0.01:50+time,pulse);
      time = time + 20;
  end
  hold off;
%--------------------In Frequency Domain ----------------------------------
%-------------------Roll of Factor = 0.25---------------------------------
  figure,
  roll_off = 0.25;
  time = 0;
  hold on;
  color= 'g';
  shift =0;
   for i = 1:test
      [Pf11, Pf12, Pf13, f11, f12, f13] = pulse_shape_freq(roll_off,
 bitstream(i+10000+shif, 1), time);

      f11=f11+shift;
      f12=f12+shift;
      f13=f13+shift;
      plot(f11, Pf11(f11-shift), color);
      plot(f12, Pf12(f12-shift), color);
      plot(f13, Pf13(f13-shift), color);


      shift = shift+ 0.62;
      if i==1
          color = 'r';
      end
      if i==2
          color = 'y';
      end
  end
  %-------------------Roll of Factor = 0.5---------------------------------
  figure,
  roll_off = 0.5;
  time = 0;
  hold on;
  color= 'g';
  shift =0;
   for i = 1:test
      [Pf11, Pf12, Pf13, f11, f12, f13] = pulse_shape_freq(roll_off,
 bitstream(i+10000+shif, 1), time);

      f11=f11+shift;
      f12=f12+shift;
      f13=f13+shift;
      plot(f11, Pf11(f11-shift), color);
      plot(f12, Pf12(f12-shift), color);
      plot(f13, Pf13(f13-shift), color);


      shift = shift+ 0.62;
```

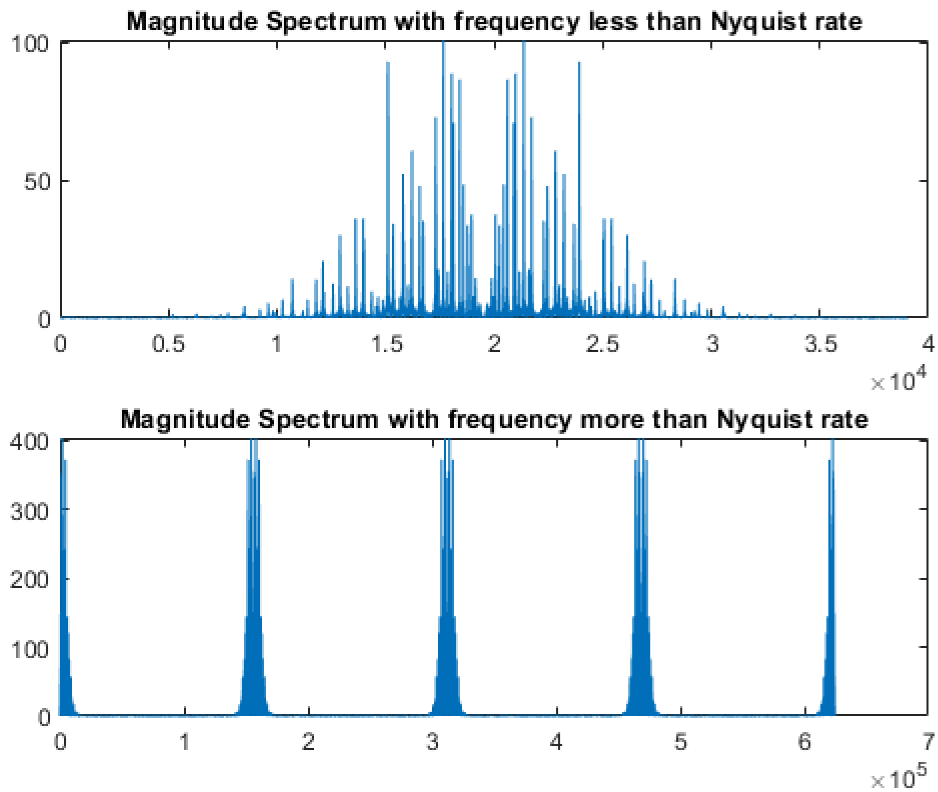
```
      if i==1
          color = 'r';
      end
      if i==2
          color = 'y';
      end
  end
  %--------------------Roll of Factor = 0.75---------------------------------
  figure,
  roll_off = 0.75;
  time = 0;
  hold on;
  color= 'g';
  shift =0;
   for i = 1:test
      [Pf11, Pf12, Pf13, f11, f12, f13] = pulse_shape_freq(roll_off,
 bitstream(i+10000+shif, 1), time);

      f11=f11+shift;
      f12=f12+shift;
      f13=f13+shift;
      plot(f11, Pf11(f11-shift), color);
      plot(f12, Pf12(f12-shift), color);
      plot(f13, Pf13(f13-shift), color);


      shift = shift+ 0.62;
      if i==1
          color = 'r';
      end
      if i==2
          color = 'y';
      end
  end
  disp('The ISI will be lesser when the roll of factor is greater');
   roll_off = 0.5;
   pulse_arr = zeros(bitstream_length,1);
  for i =1:bitstream_length
      pulse = pulse_shape_time(roll_off, bitstream(i, 1), time);
      pulse_arr(i,1) = value(pulse);
  end
end
%------------------------------------------------------------------------
%This function obtains the raised cosine in time domain. For bit value = 0
%the sinc is upside down and for bit value = 1 the raised cosine has
%maximum amplitude = 1.
function [pt1] = pulse_shape_time(roll_off, bit_val, time)
  Tb = 2;
  Rb = 1/Tb;
   %time axis from -10 to 10 with increment of 0.01.
   t = -50:0.01:50;
  %pt11, pt12 in numerator and pt13 in denominator.
   pt11 = sinc(t/Tb);
   pt12 = cos(pi*roll_off*Rb*t);
```

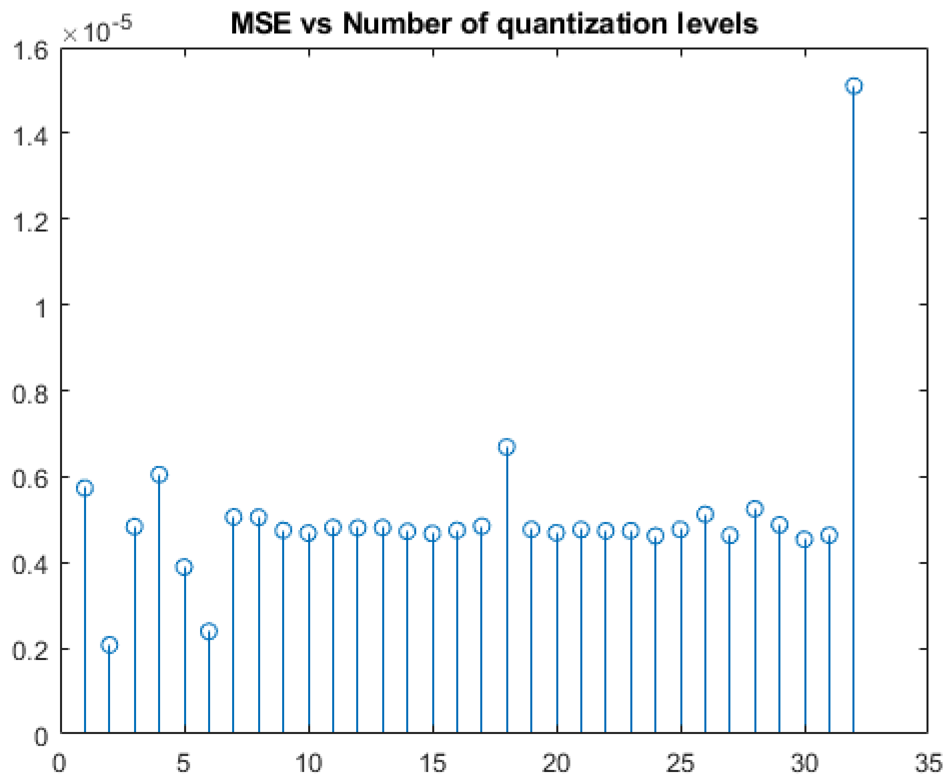
```
  pt13 = 1 - (4*roll_off*roll_off*Rb*Rb*t.^2);

  if(bit_val==0)
      pt1 = -(pt11.*pt12)./(pt13);
  else
      pt1 = (pt11.*pt12)./(pt13);
  end
end
%-----------------------------------------------------------------------
%This function obtains the raised cosine in frequency domain. For bit
%value = 0 the pulse is upside down and for bit value = 1 the pulse
%is in positive axis.
function [Pf11, Pf12, Pf13, f11, f12, f13] = pulse_shape_freq(roll_off,
 bit_val, time)
  Tb = 2;
  Rb = 1/Tb;
   if(bit_val==0)
      Pf11 = @(f) ((-1)*f.^0);
      Pf12 = @(f) ((-0.5)*(1 - (sin(pi*((f - (Rb/2))/(roll_off*Rb))))));
      Pf13 = @(f) 0*f.^0;
      f11 = (0.5*Rb)*(-1+roll_off):0.01:(0.5*Rb)*(1-roll_off);
      f12 = (0.5*Rb)*(1-roll_off):0.01:(0.5*Rb)*(1+roll_off);
      f13 = (0.5*Rb)*(1+roll_off):0.01:0.5;


  else
      Pf11 = @(f) 1*f.^0;
      Pf12 = @(f) 0.5*(1 - (sin(pi*((f - (Rb/2))/(roll_off*Rb)))));
      Pf13 = @(f) 0*f.^0;
      f11 = (0.5*Rb)*(-1+roll_off):0.01:(0.5*Rb)*(1-roll_off);
      f12 = (0.5*Rb)*(1-roll_off):0.01:(0.5*Rb)*(1+roll_off);
      f13 = (0.5*Rb)*(1+roll_off):0.01:0.5;
  end
end
%-----------------------------------------------------------------------
%This function determines if the pulses obtained have maximum positive
%value as 1 or -1. Then depending upon that it obtains the decoded
%bitstream.
function val = value(pulse)
   values = zeros(1,length(pulse));
   for i = 1:1:length(pulse)
     if(abs(-1-pulse(i)) <= abs(1-pulse(i)))
           values(i) = 0;
       else
         values(i) = 1;
     end
   end
    ans = sum(values);
    val = 0;
    if(ans>length(pulse)/2)
        val = 1;
    end
end
%-----------------------------------------------------------------------
```

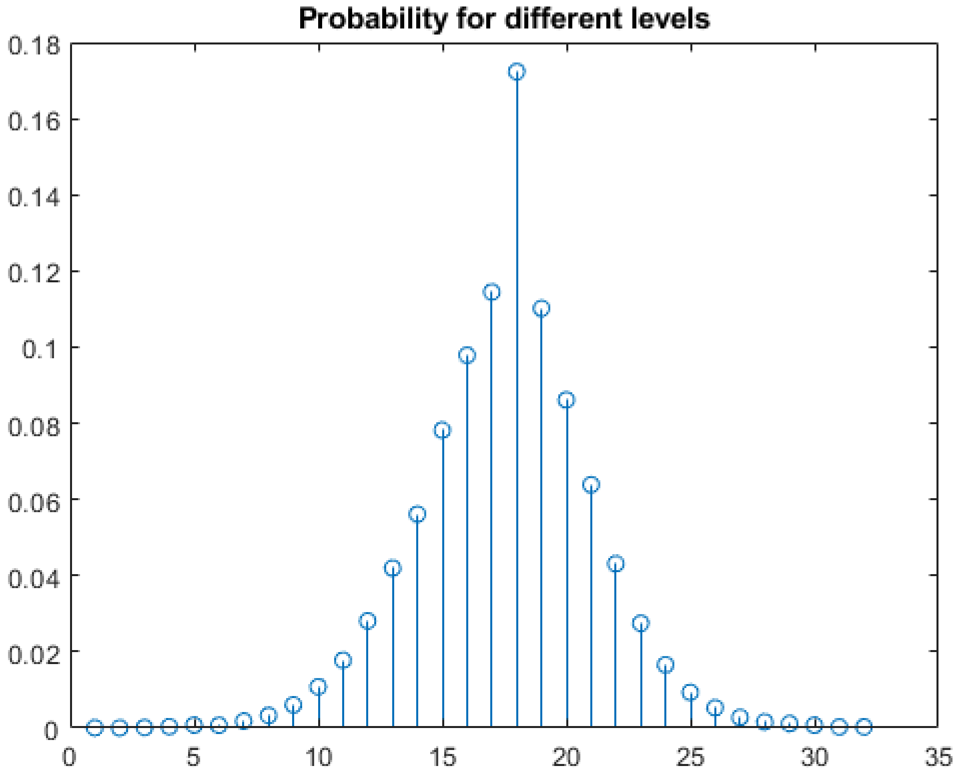
```
%Here we are decoding the obtained bitstream and then finding the
%quantization values associated with the decoded bits. This outputs the
%quantization levels from decoded bitstream.
function [decoded_quantized_values] = receivedecode(bitstream_length,
 received_bitstream, corresponding_huffman, decoded_length)
 i = 1;
 c = 0;
 idx = 1;
 decoded_quantized_values = zeros(decoded_length, 1);
 while(i<=bitstream_length)
     j = i;
     sum = 0;
     while(j<=bitstream_length)
         if(received_bitstream(j, 1)==0 || c==31)
             break;
         end
         j = j+1;
         c = c+1;

     end
     if(c==31)
         c = 30;
     end

     j = i;
     while(c>=0)
         sum = sum + received_bitstream(j, 1)*power(2, c);
         c = c-1;
         j = j+1;
     end
     for k = 1:32
         if(sum==corresponding_huffman(k, 1))
             decoded_quantized_values(idx, 1) = k;
         end
     end
     i = j;
     c = 0;
     idx=idx+1;
 end
end
%-----------------------------------------------------------------------
%Here we are assigning the quantization values corresponding to
%quantization levels. This takes input as quantization levels and then
%outputs the quantized values according to the lookup_values.
function [decoded_output] = tablelookup(decoded_quantized_values,
 lookup_values)
  len = length(decoded_quantized_values);
  decoded_output = zeros(len, 1);
   for i = 1:len
      quant_val = decoded_quantized_values(i, 1);
      decoded_output(i, 1) = lookup_values(quant_val, 1);
  end
 end
```

```
Sampling Frequency in Hz 44100
Nyquist Frequency in Hz 22050
The maximum frequency component of the spectrum is 404.323 at 76224.
-------------EveryThing For frequency less than Nyquist
 rate--------------------------
We are using 32 levels for quantization means 5 bits per level
MSE for this sampled audio is  5.0939e-06
Total Bits used in representation of one level is 5
Total Bits used in Bitstream 195180
The length of the bitstream for using huffman coding 220527
The ISI will be lesser when the roll of factor is greater
-------------EveryThing For frequency more than Nyquist
 rate--------------------------
We are using 32 levels for quantization means 5 bits per level
MSE for this sampled audio is  4.5607e-06
Total Bits used in representation of one level is 5
Total Bits used in Bitstream 3122880
The length of the bitstream for using huffman coding 1315422
The ISI will be lesser when the roll of factor is greater
```

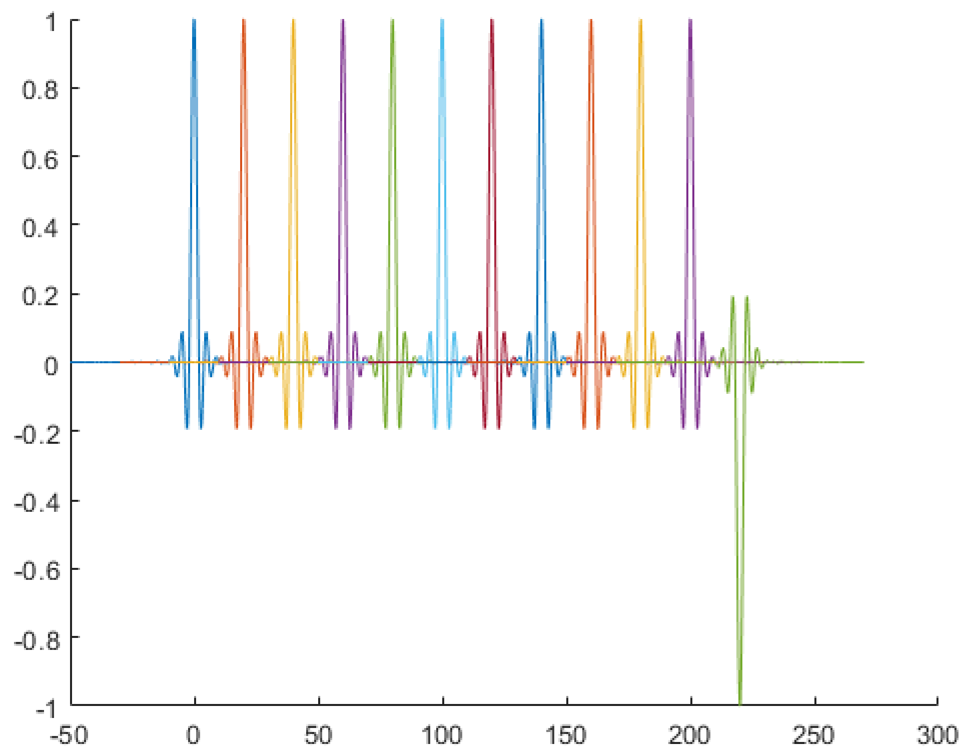



Magnitude Spectrum with frequency less than Nyquist rate
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5
4
×10^4
Magnitude Spectrum with frequency more than Nyquist rate
400
300
200
100
0
0
1
2
3
4
5
6
7
×10^5

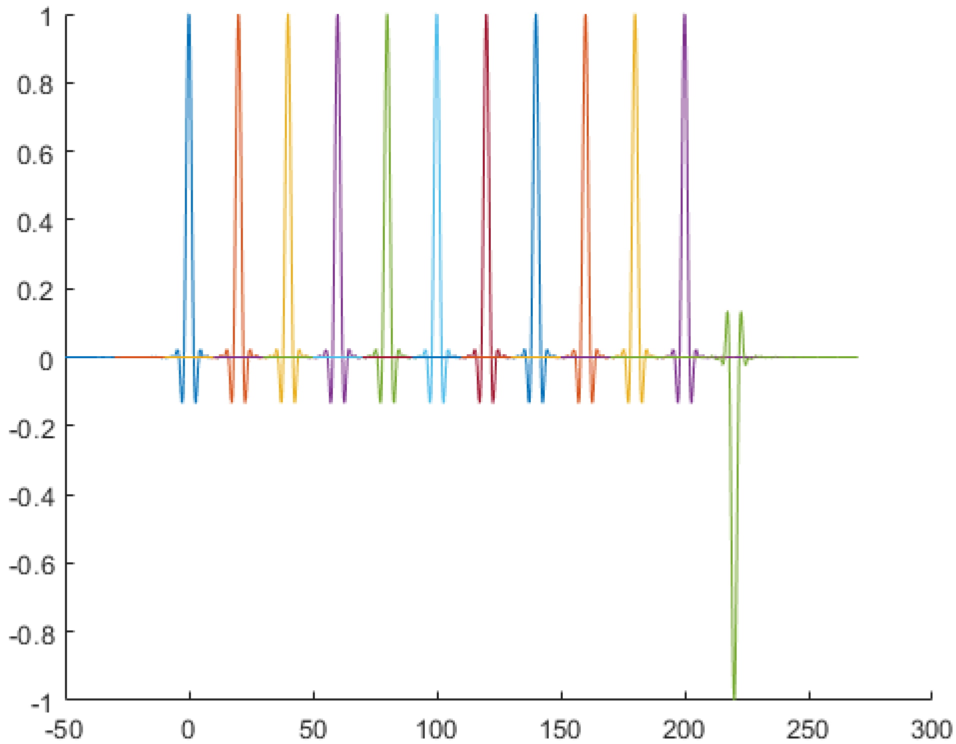

MSE vs Number of quantization levels
×10^-5
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
25
30
35

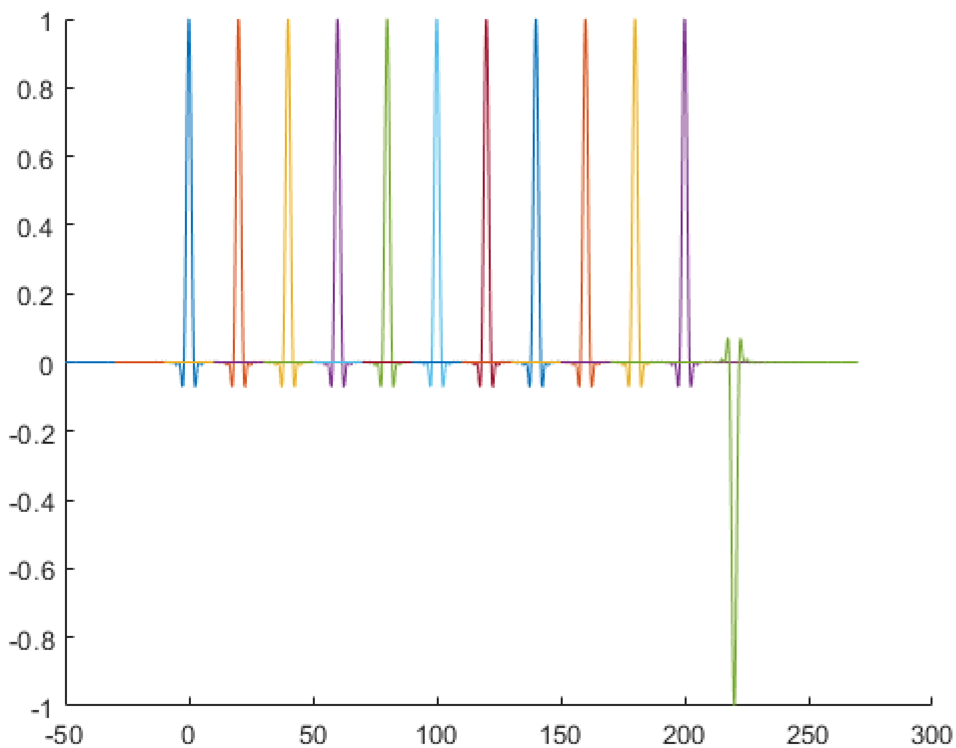


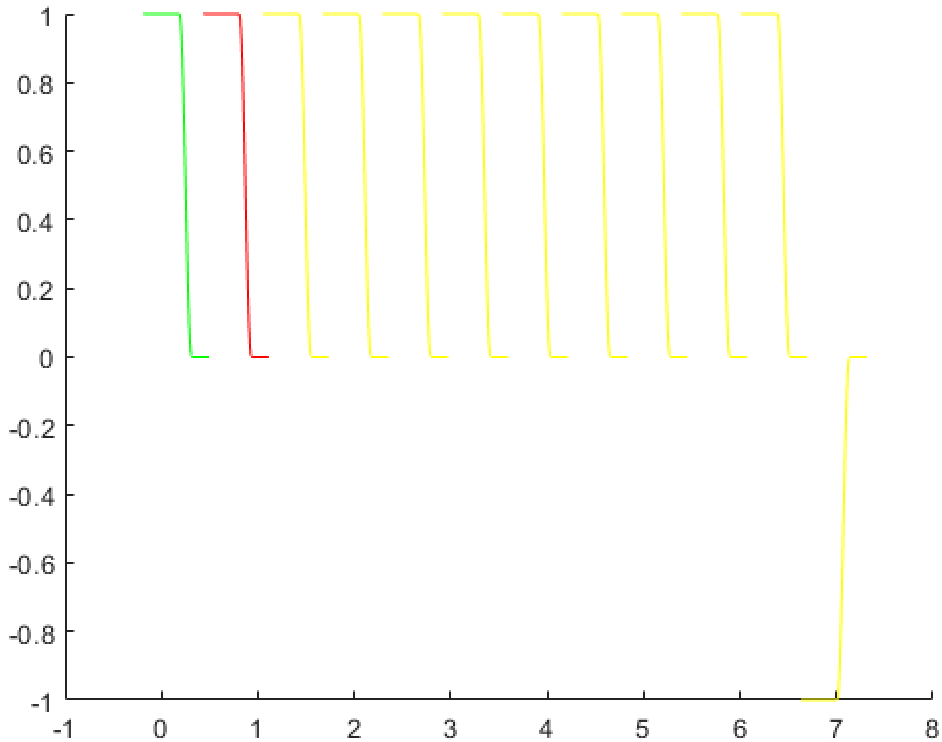
Probability for different levels
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
5
10
15
20
25
30
35

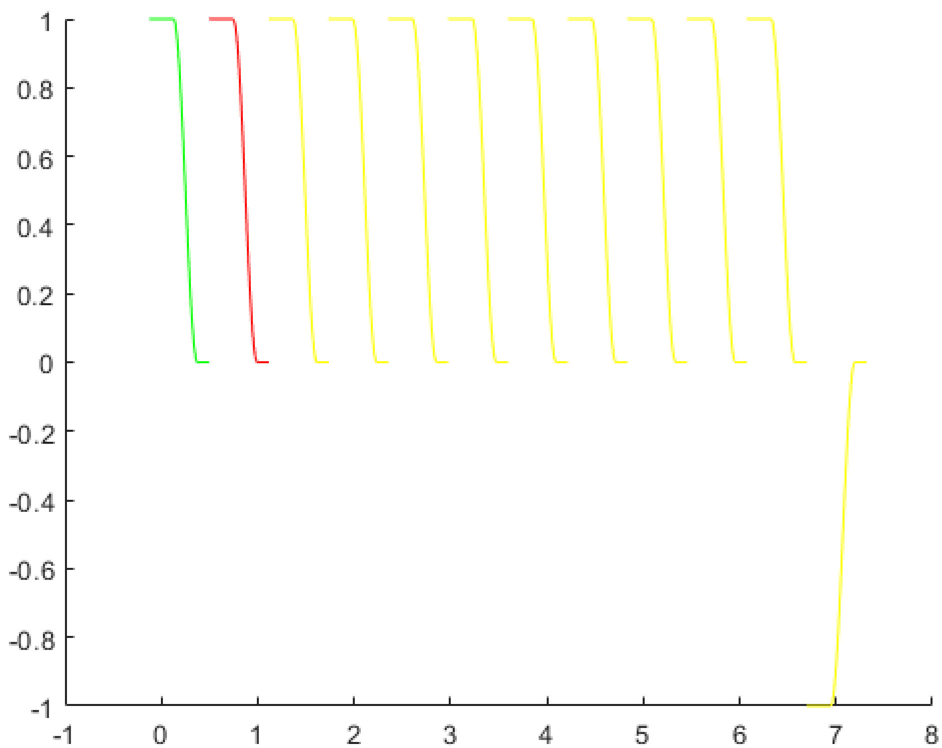


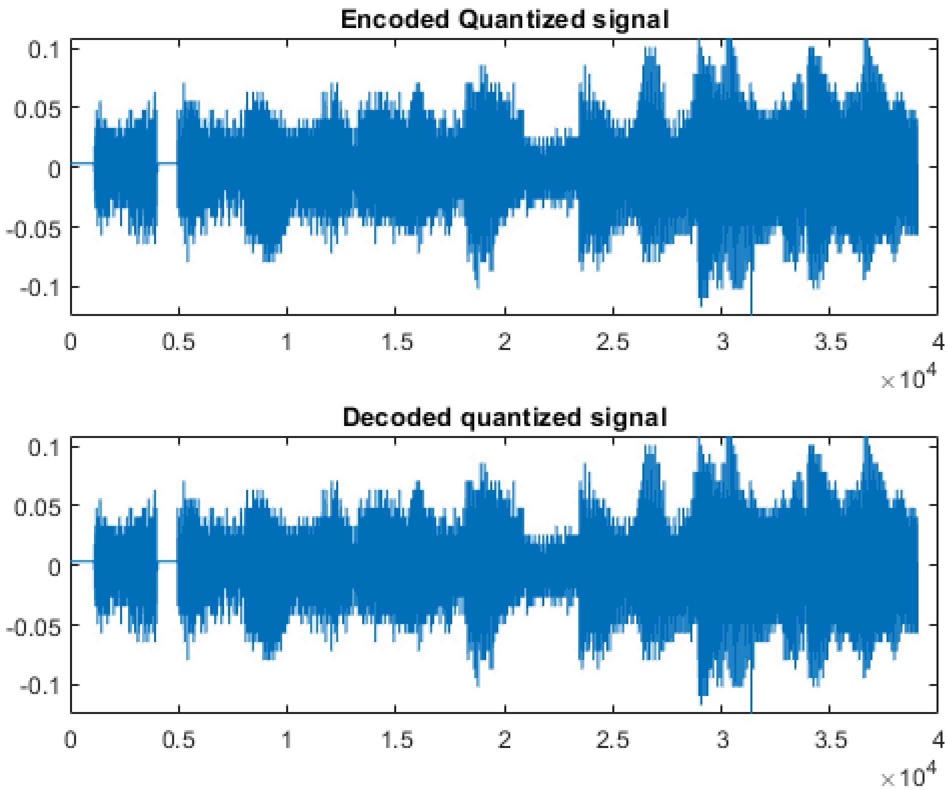
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-50
0
50
100
150
200
250
300

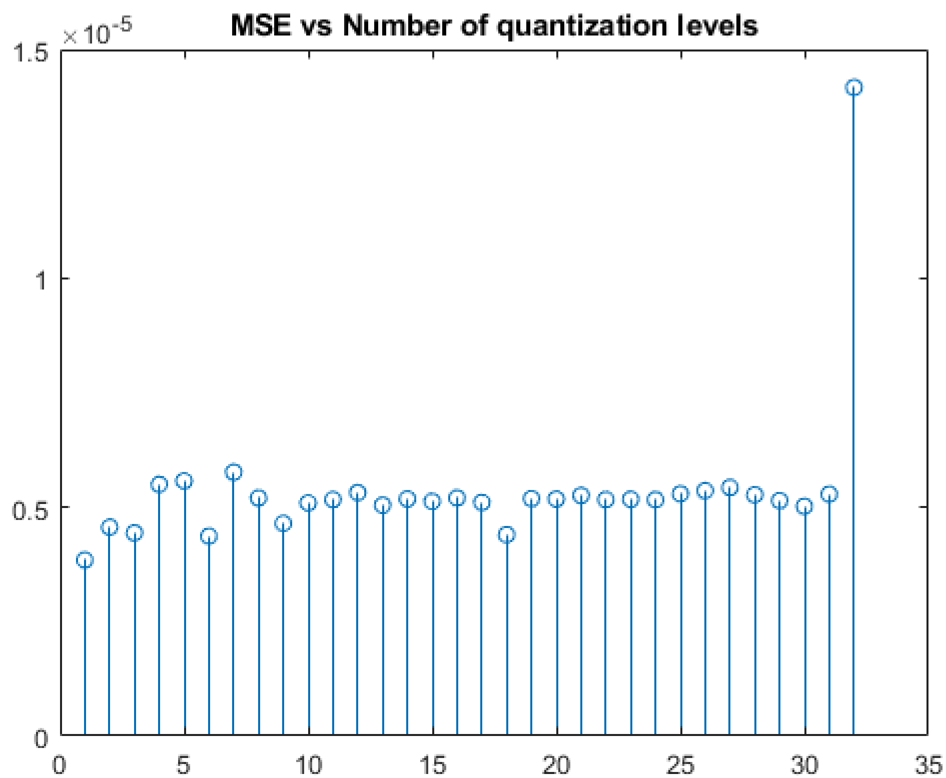

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-50
0
50
100
150
200
250
300

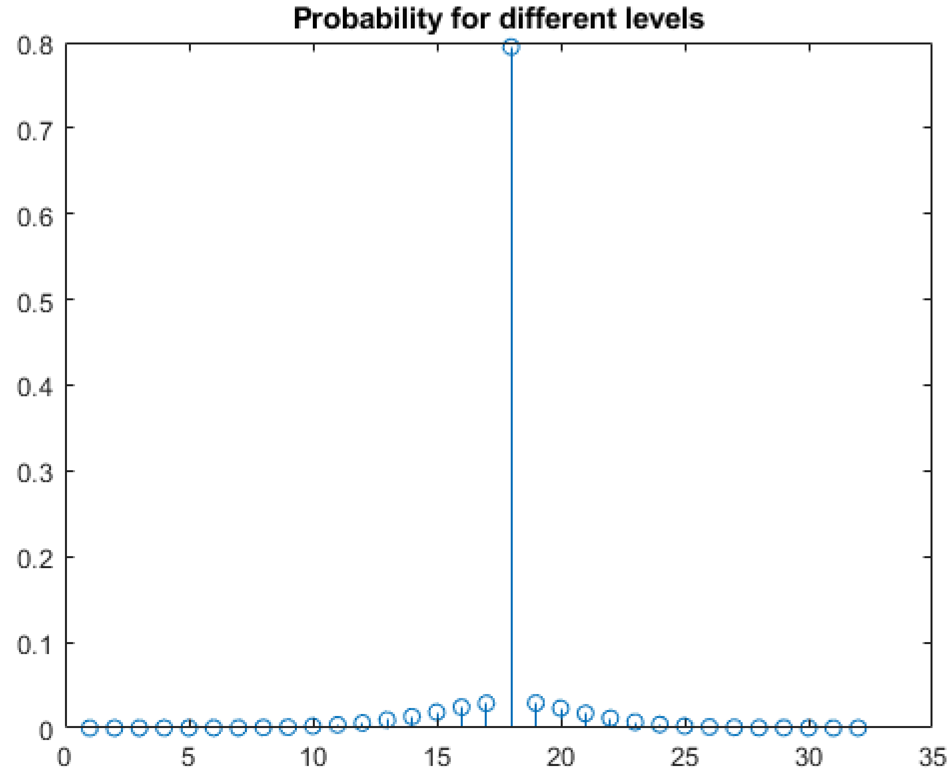

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-50
0
50
100
150
200
250
300

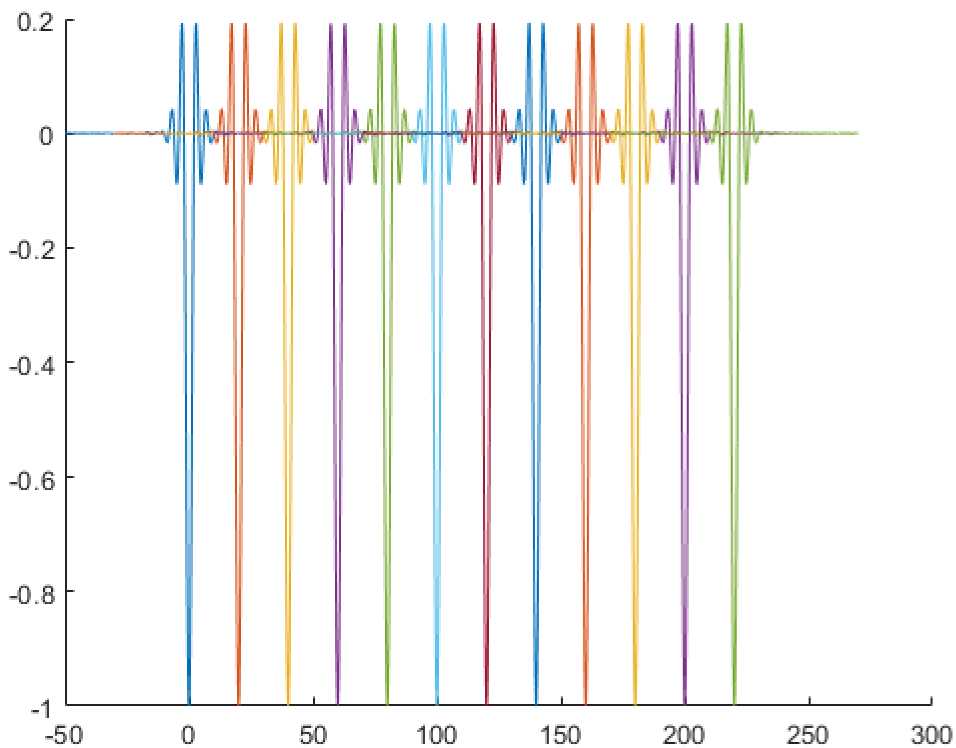

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
0
1
2
3
4
5
6
7
8

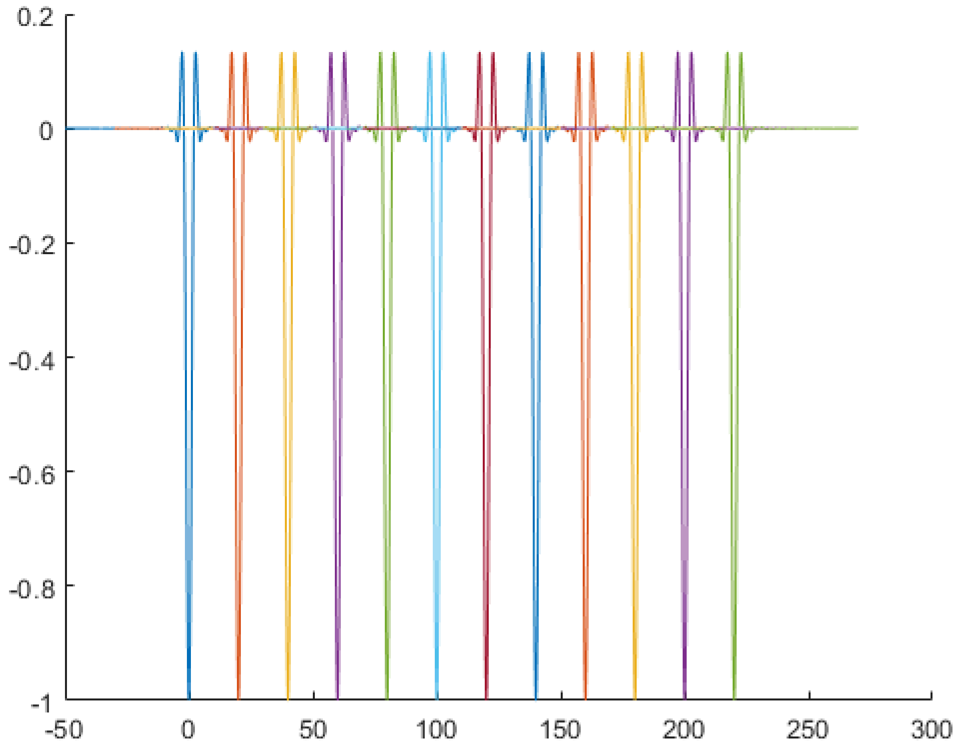

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
0
1
2
3
4
5
6
7
8

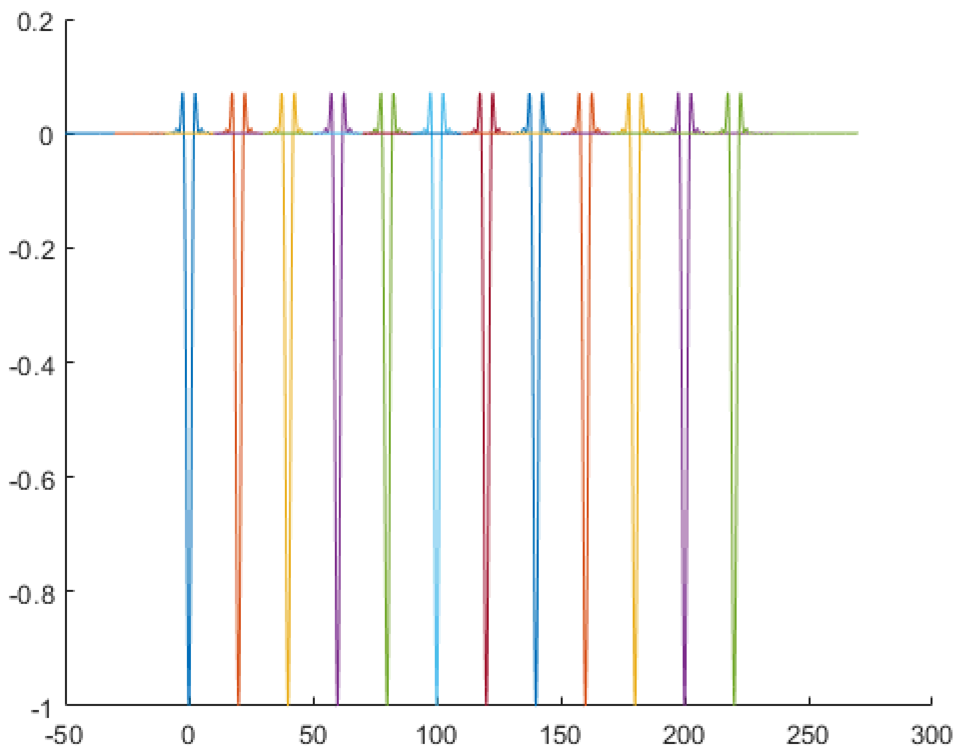

Encoded Quantized signal
0.1
0.05
0
-0.05
-0.1
0
0.5
1
1.5
2
2.5
3
3.5
4
×10^4
Decoded quantized signal
0.1
0.05
0
-0.05
-0.1
0
0.5
1
1.5
2
2.5
3
3.5
4
×10^4

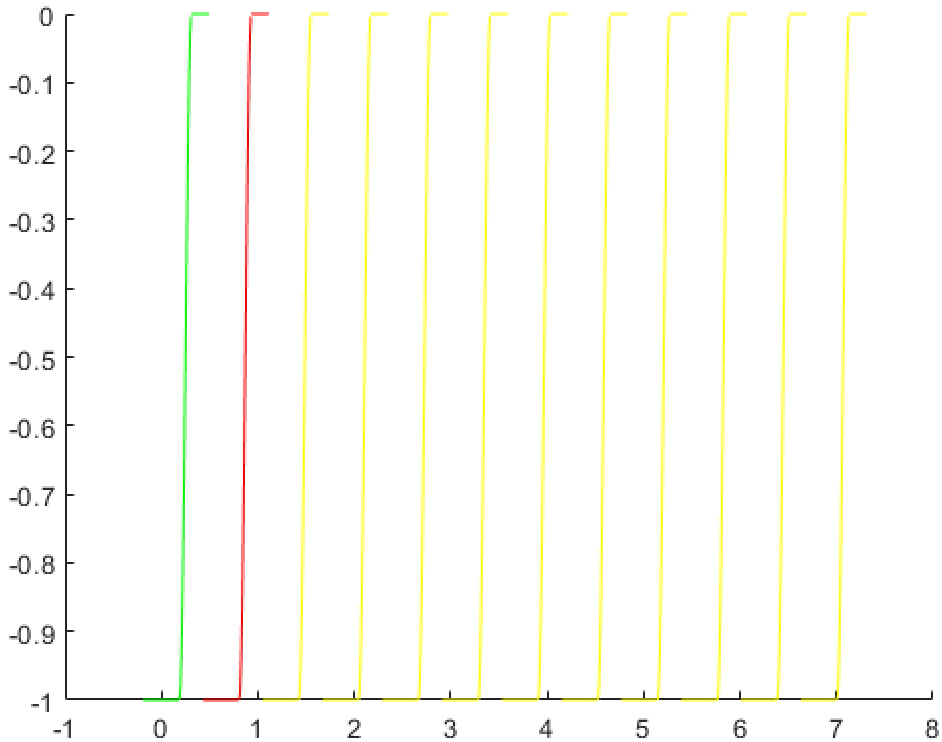

MSE vs Number of quantization levels
×10^-5
1.5
1
0.5
0
0
5
10
15
20
25
30
35

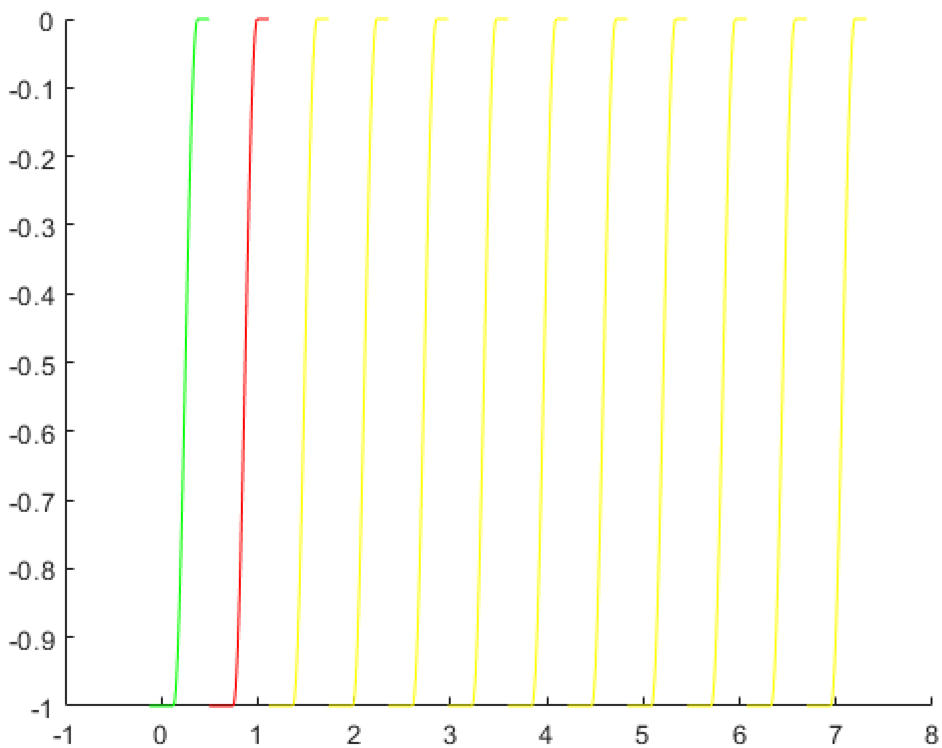

Probability for different levels
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35

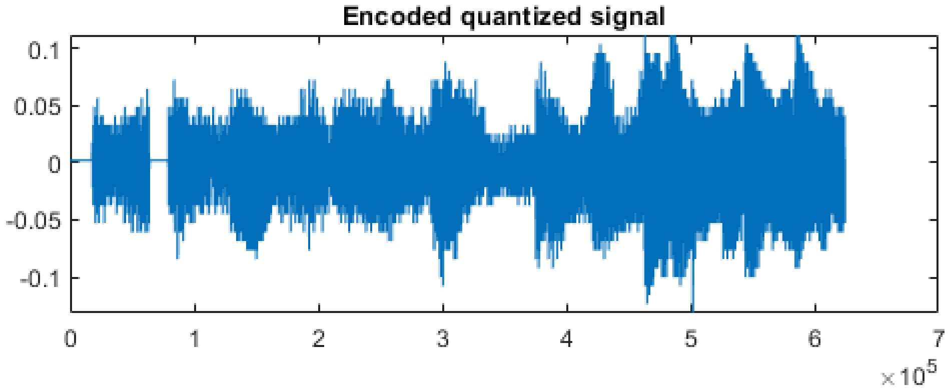

0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-50
0
50
100
150
200
250
300

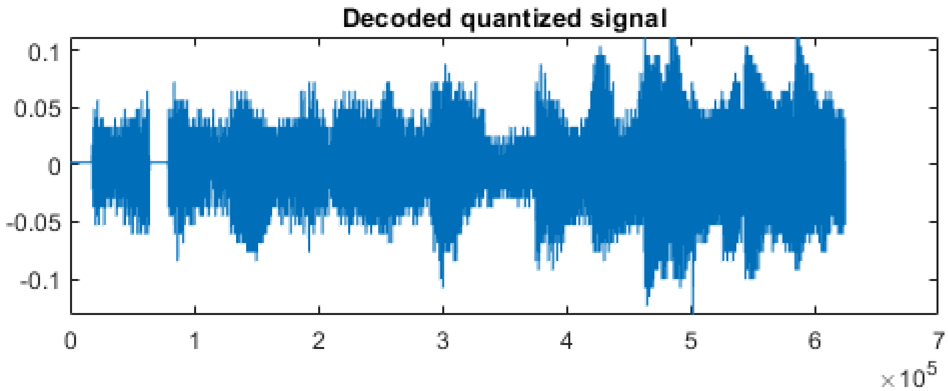

0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-50
0
50
100
150
200
250
300


0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-50
0
50
100
150
200
250
300


0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
-0.9
-1
-1
0
1
2
3
4
5
6
7
8


0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
-0.9
-1
-1
0
1
2
3
4
5
6
7
8





Published with MATLAB® R2021b