Assignment Report

Course : ECE 340: Digital Communication Systems

Group members :

Himanshu - 2018147
Shubham Manjhi - 2018194
Aman shah - 2018207
Puneet Kumar - 2018176

Answer 1.

We Choose Audio File.

Answer 2.

Contents :-

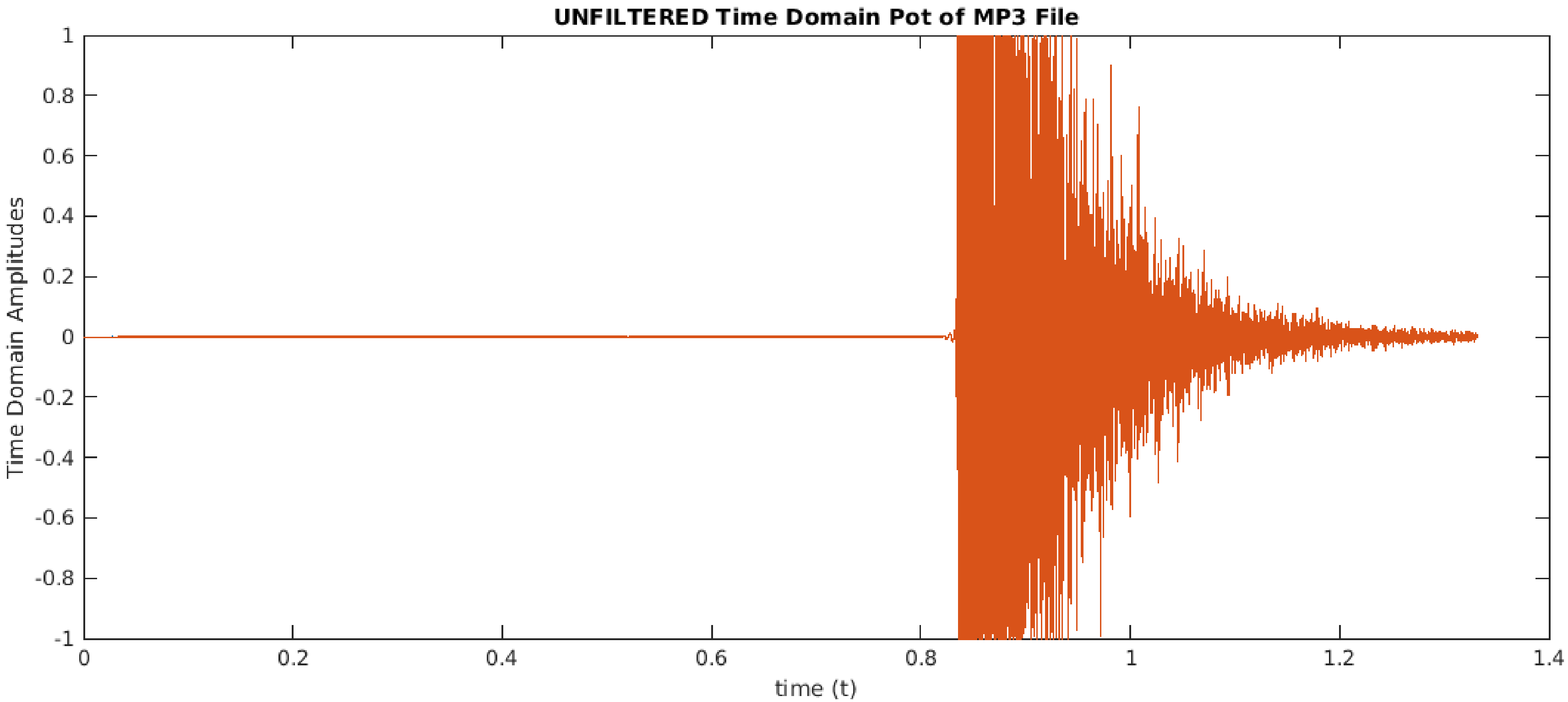
Contents


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Quest 2.)Import the signal into MATLAB and display the characteristics  %
%          of the signal in time domain (amplitude and phase of the sig-  %
%          -nal) and in the frequency domain (the spectrum). Use filters  %
%          wherever necessary to restrict your signal/spectrum to a finite%
%          domains. Develop a function input() to do this.                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Matlab Code %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Answer :-


function [Fs,Data] = input()
```

Unfiltered Audio Signal File in Time Domain

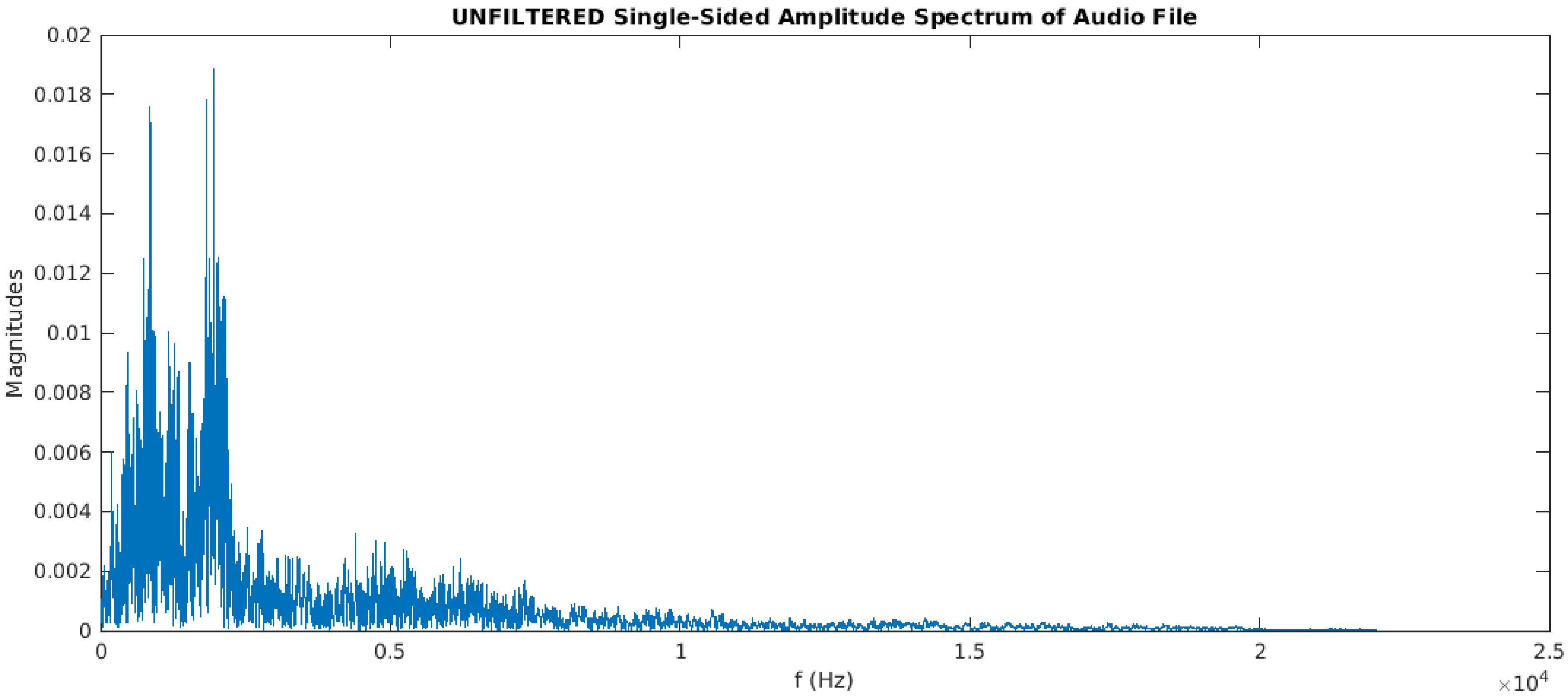
```
close all; clear all;
[data, Fs] = audioread('Audio.mp3');  % Y contains the data file,
                                      % fs contains the sampling


%%%%%%%%%%%%% Plotting Unfiltered Audio Signal File in Time Domain.%%%%%%%%
t = linspace(0,length(data)/Fs,length(data));
figure(1); plot(t,data);
title('UNFILTERED Time Domain Pot of MP3 File');
xlabel('time (t)');
ylabel('Time Domain Amplitudes');
```

```
Warning: Function input has the same name as a MATLAB builtin. We suggest you
rename the function to avoid a potential name conflict.
```



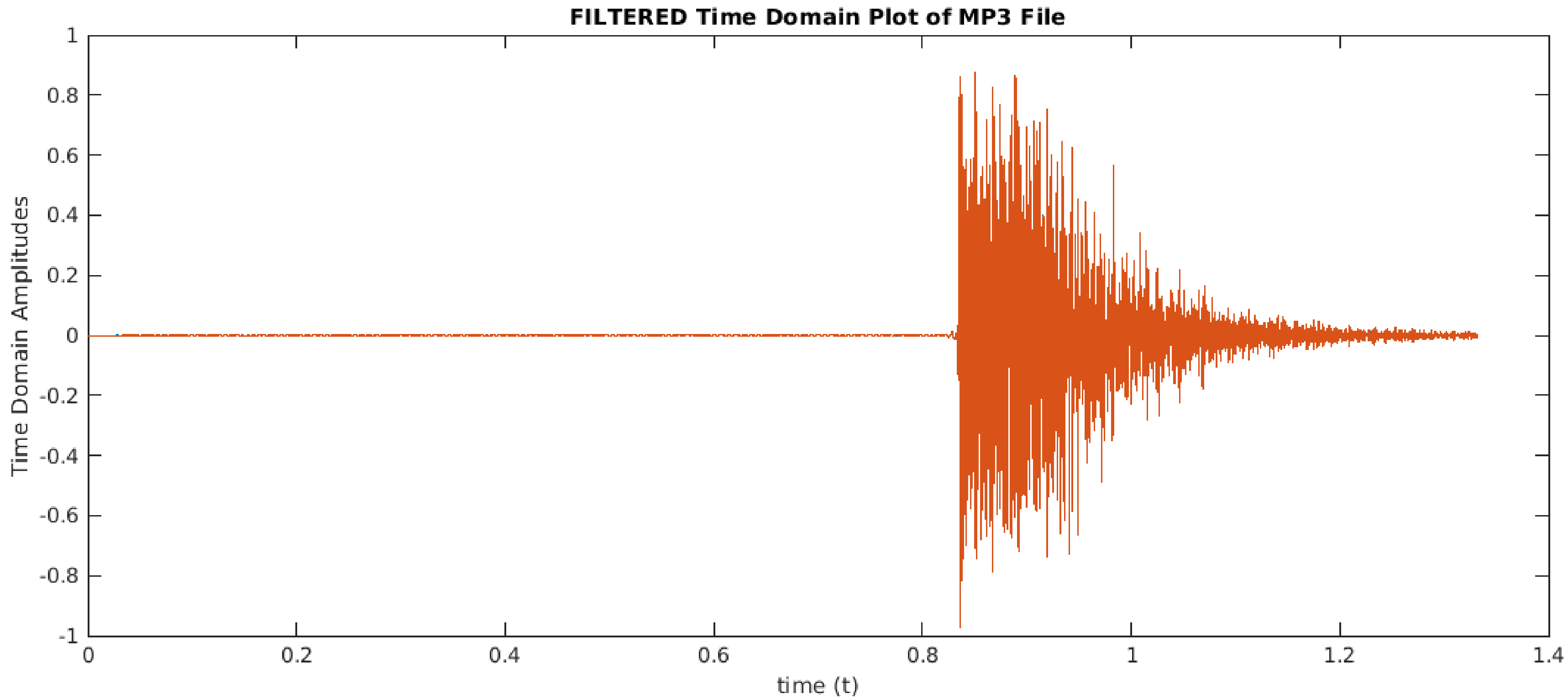
Unfiltered Audio Signal File in Frequency Domain

```
%%%%%%%%%%%% Does the convertion from time domain to frequency domain%%%%%%%
Y = fft(data);
L = length(Y);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);


%%%%%%%%%%% Plotting Unfiltered Audio Signal File in Frequency Domain.%%%%%%
f = Fs*(0:(L/2))/L;
figure(2); plot(f,P1);
title('UNFILTERED Single-Sided Amplitude Spectrum of Audio File');
xlabel('f (Hz)');
ylabel('Magnitudes');
```



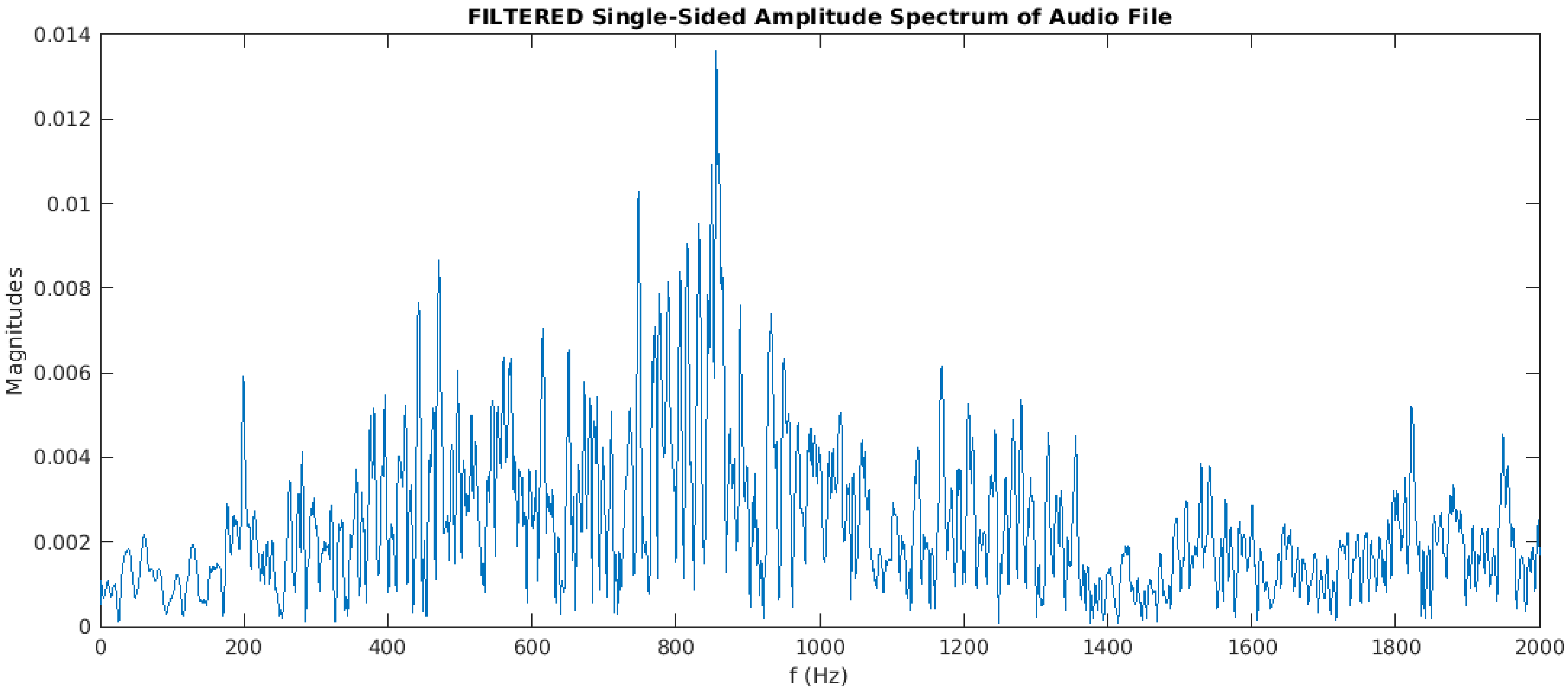
Filtered Audio Signal File in Time Domain

```
%%%%%%%%%%%%%%%%%%%%%%%%%%% FILTERED AUDIO FILE %%%%%%%%%%%%%%%%%%%%%%%%%%%
Wn =  2000/40000;        % 2000/40000; %cutoff frequency desired is 2000Hz
t = linspace(0,length(data)/Fs,length(data));
b1 = fir1(34,Wn,'low');  %low pass filter is used
y_filtered = filter(b1, 1, data);



%%%%%%%%%%%%% Plotting filtered Audio Signal File in Time Domain.%%%%%%%%%%
figure(3);; plot(t,y_filtered);
Data = y_filtered;
title('FILTERED Time Domain Plot of MP3 File');
xlabel('time (t)');
ylabel('Time Domain Amplitudes');
```



Filtered Audio Signal File in Frequency Domain

```
%%%%%%%%%% Does the convertion from time domain to frequency domain %%%%%%%
Y = fft(y_filtered);
L = length(Y);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);



%%%%%%%%%%% Plotting filtered Audio Signal File in Frequency Domain. %%%%%%
f = Fs*(0:(L/2))/L;
figure(4);; plot(f,P1); xlim([0 2000]);
title('FILTERED Single-Sided Amplitude Spectrum of Audio File');
xlabel('f (Hz)');
ylabel('Magnitudes');
```



Let us check Output for Fs :-

```
Fs
```

```
Fs =

       44100
```

Final Filter Data :-

```
data(3000:3020,1:2)
disp("......... etc.")
```

```
ans =

   1.0e-03 *

    0.0305    0.0610
   -0.6104   -0.6104
   -0.6104   -0.6104
   -0.5798   -0.5798
   -0.1221   -0.1221
    0.2441    0.2441
    0.2136    0.2441
    0.1831    0.1831
   -0.3052   -0.3052
   -0.3662   -0.3967
    0.1526    0.1526
    0.4272    0.3967
    0.4272    0.3967
    0.5493    0.5493
    0.3357    0.3052
   -0.1526   -0.1526
   -0.3052   -0.3052
    0.1526    0.1221
    0.5798    0.5798
    0.6104    0.6104
    0.4578    0.4578

......... etc.
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% End Input() Function %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Answer 3.

Contents :-

Maximum Frequency component of the spectrum
Sample these Digital Audio signal using a frequency less than the Nyquist rate.
Original Filter Signal using a frequency less than the Nyquist rate.
New Sample Signal using a frequency less than the Nyquist rate in Time Domain.
New Sample using a frequency less than the Nyq rate in Freqquency Domain.
Sample Digital Audio signal using a frequency Greater than the Nyquist rate.
New Sample using a frequency Greater than the Nyquist rate in Time Domain.
New Sample using a frequency Greater than the Nyq rate in Freq Domain.

Contents

Question 3.)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Quest 3.) Find out the maximum frequency component of the spectrum (after   %
%           possible filtering), and sample the signal using i) a frequency   %
%           less than the Nyquist rate, and ii) a frequency greater than the  %
%           Nyquist rate. Display the spectrum of the sampled signal for both %
%           the cases. Develop a function source() to                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Matlab Code %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Answer :-

```
function [Sample,Fsn,fmax] = source(Data,Fs)   % Fs Sampling frequency
```

Maximum Frequency component of the spectrum

```
Xr = fft(Data);
Xrmag = abs(Xr);
l = length(Xr);
```

```
f = Fs*(0:(l/2))/l;
%figure((f,Xrmag(1:length(f)))
%title('Frequncy Domain :- Original Filter Signal ');
%xlabel('Frequency (Hz)')
%ylabel('Magnitude')stem

[fr,fc] = find(max(Xrmag(1:length(f)))==Xrmag(1:length(f))); % Finds the maximum frequency index
fmax = f(fr,fc);            % Maximum Frequency component is same as the frequency of the signal


Error in source (line 23)
Xr = fft(Data);
```

Sample these Digital Audio signal using a frequency less than the Nyquist rate.

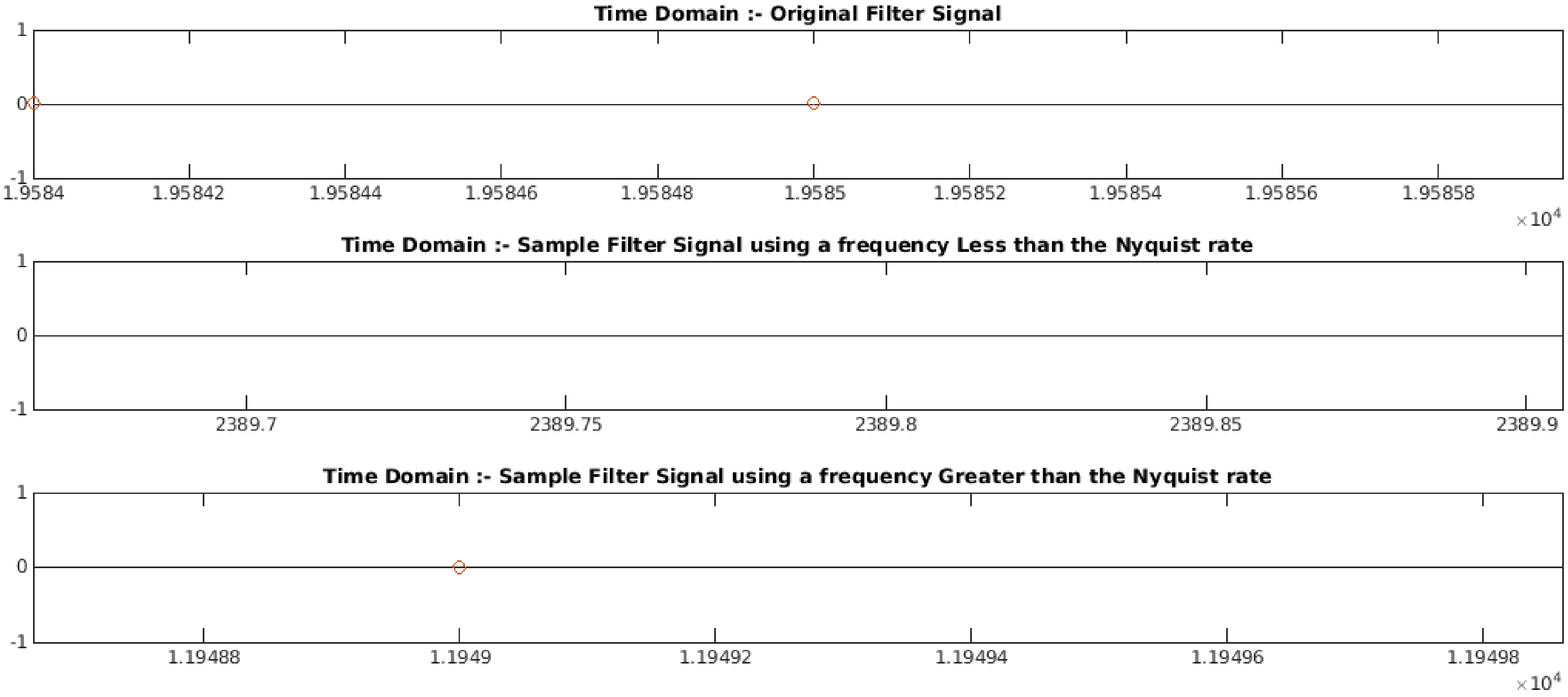
```
%Fsn is the sampling rate which can also be considered as the Nyquist rate.
Fsn = round(2*pi*fmax); %new sampling rate lower than the nyquist rate
Newdata = resample(Data,Fsn,Fs); %sample the using the new sampling rate
x1=length(Data);
x2=length(Newdata);
```

Original Filter Signal using a frequency less than the Nyquist rate.

```
subplot(311);
stem(Data);
title('Time Domain :- Original Filter Signal');
axis([x1/3 (x1/3)+(x1/30000) -1 1])     %zoom in on the plot
```

New Sample Signal using a frequency less than the Nyquist rate in Time Domain.

```
subplot(312);
stem(Newdata);
axis([x2/3 (x2/3)+(x2/30000) -1 1])     %zoom in on the plot
title('Time Domain :- Sample Filter Signal using a frequency Less than the Nyquist rate');
```

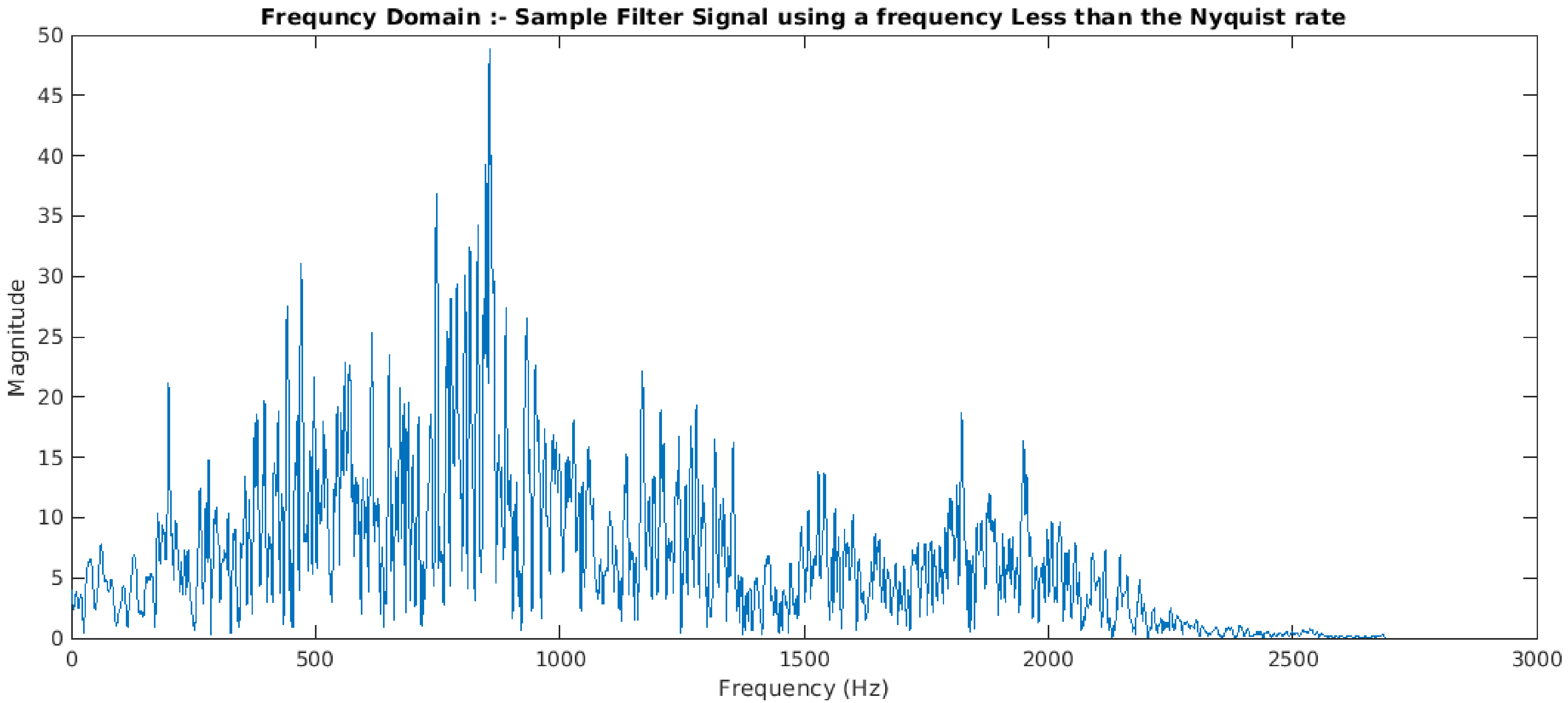


Let us check Output for Fs :-

```
Fs
```

New Sample Signal using a frequency less than the Nyquist rate in Freqquency Domain.

```
Xr = fft(Newdata);

Xrmag = abs(Xr);
l = length(Xr);
f = Fsn*(0:(l/2))/l;
```



Let us check Output for Fs :-

```
Fs
```

Sample these Digital Audio signal using a frequency Greater than the Nyquist rate.

```
%Fs is the sampling rate which can also be considered as the Nyquist rate.
Fsn = round(2*pi*fmax*5); %new sampling rate lower than the nyquist rate
Newdata = resample(Data,Fsn,Fs); %sample the using the new sampling rate
```

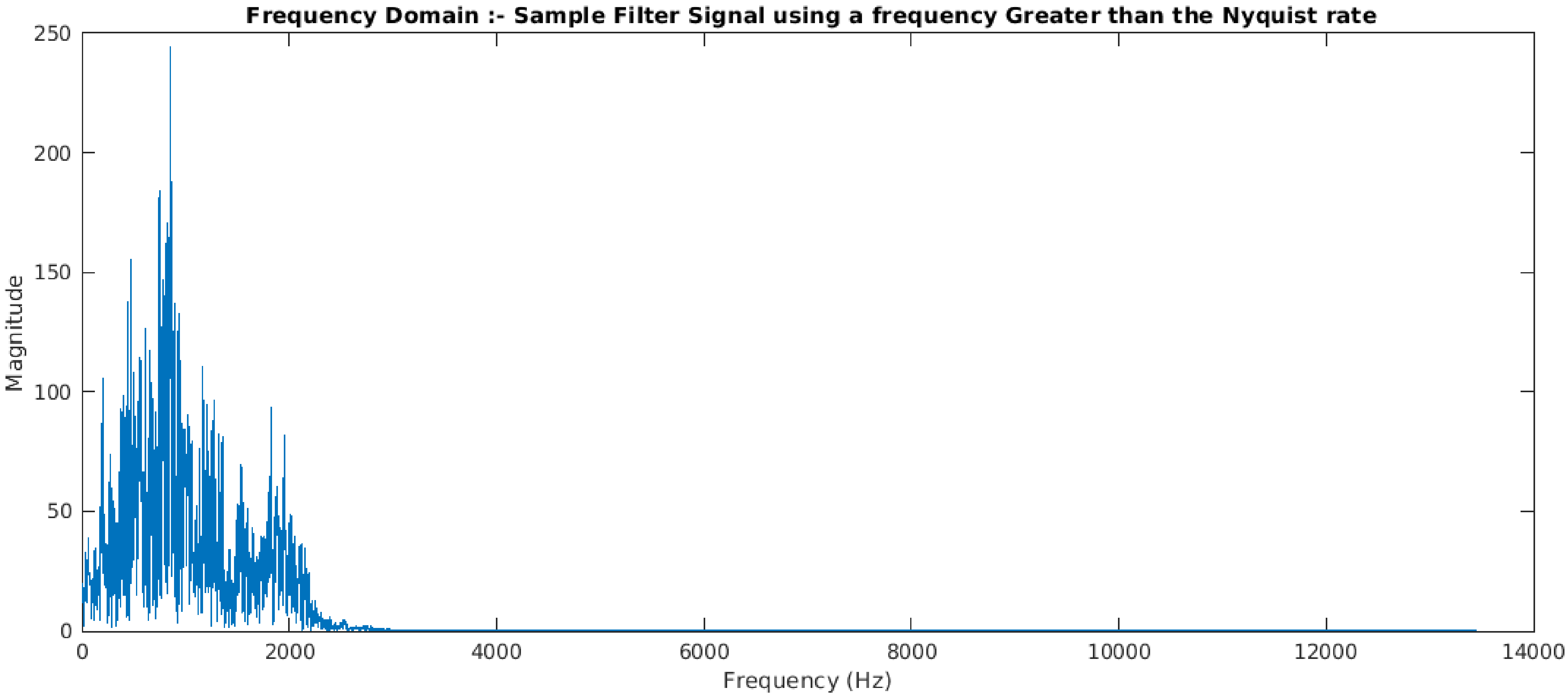
```
x1=length(Data);
x2=length(Newdata);
Sample = Newdata;
```

New Sample Signal using a frequency Greater than the Nyquist rate in Time Domain.

```
subplot(313);
stem(Newdata);
title('Time Domain :- Sample Filter Signal using a frequency Greater than the Nyquist rate');
axis([x2/3 (x2/3)+(x2/30000) -1 1])     %zoom in on the plot


figure();
plot(f,Xrmag(1:length(f)))
title('Frequncy Domain :- Sample Filter Signal using a frequency Less than the Nyquist rate');
xlabel('Frequency (Hz)')
ylabel('Magnitude')
```

New Sample Signal using a frequency Greater than the Nyquist rate in Freqquency Domain.

```
Xr = fft(Newdata);
Xrmag = abs(Xr);
l = length(Xr);
f = Fsn*(0:(l/2))/l;
figure();
plot(f,Xrmag(1:length(f)))
title(' Frequency Domain :- Sample Filter Signal using a frequency Greater than the Nyquist rate');
xlabel('Frequency (Hz)')
ylabel('Magnitude')
```




```
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% End source() Function %%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Answer 4.

Contents

- Number of bits
- Quantizing width
- Quantized signal
- MSE
- Helper funtion for qunatization

Contents

Question 4.)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Quest 4.) Choose a reasonable (up to your discretion) number of levels for  %
%           quantizing the sampled outputs. For your choice of levels calcul- %
%           -ate the MSE of the quantization. Plot an MSE vs number of quant- %
%           -ization levels graph for different values of quantization levels.%
%           Develop a function quantize() to do this.                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Matlab Code %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Answer :-

```
function [y3,TotalBitsUsedInThIsTepresentationBits] = Quantize(Sample)
```

```
    data=Sample(:,1);
    maxsig=max(data);
    minsig=min(data);
    R = max(data)-min(data);         %full-scale range from -1 to 1

Error in Quantize (line 24)
    data=Sample(:,1);
```

Number of bits

```
B1 = 3;        %2^3 = 8 levels
B2 = 4;        %2^4 = 16 levels
B3 = 5;        %2^5 = 32 levels
B4 = 6;        %2^6 = 64 levels
```

```
B5 = 7;        %2^7 = 128 levels
B6 = 8;        %2^8 = 256 levels
```

Quantizing width

```
Q1 = minsig:(R/((2^B1))):maxsig;
Q2 = minsig:(R/((2^B2))):maxsig;
Q3 = minsig:(R/((2^B3))):maxsig;
Q4 = minsig:(R/((2^B4))):maxsig;
Q5 = minsig:(R/((2^B5))):maxsig;
Q6 = minsig:(R/((2^B6))):maxsig;
```

Quantized signal

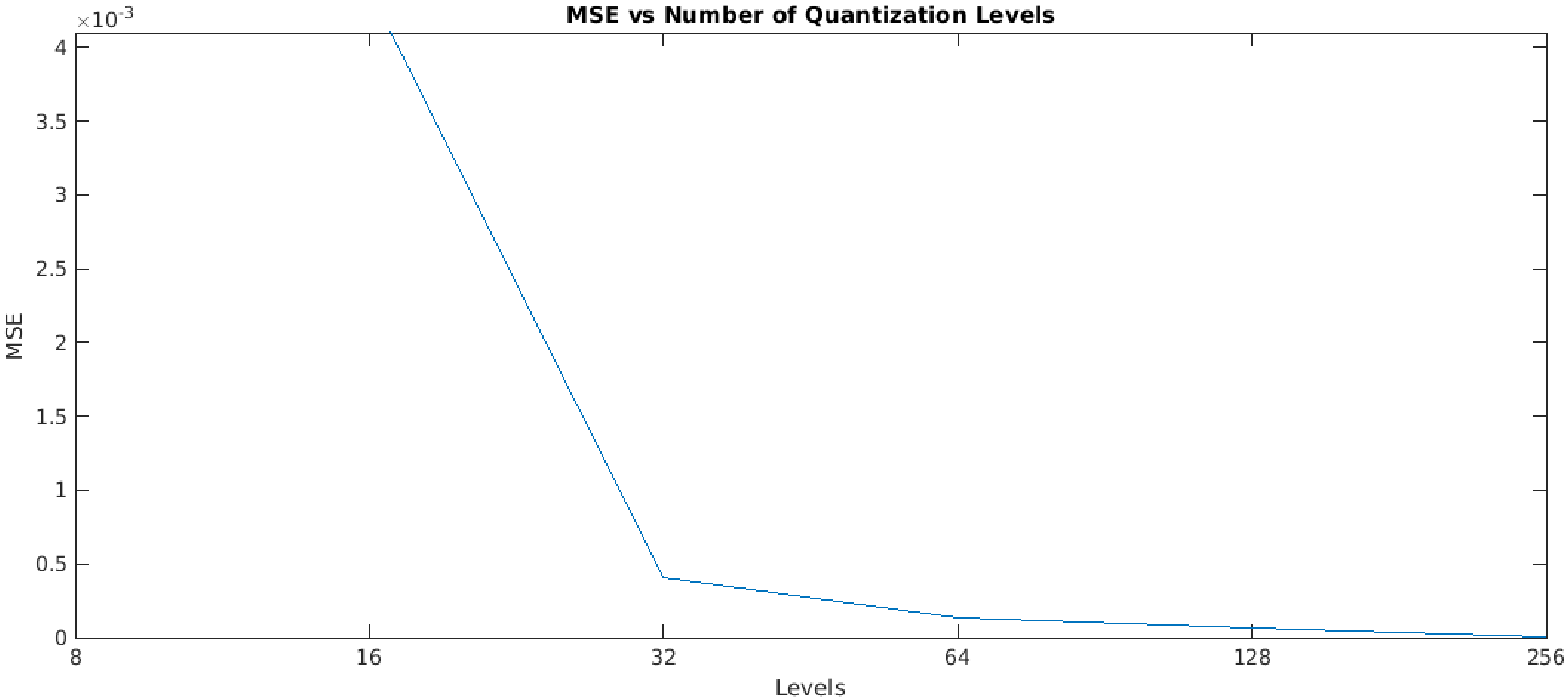
```
y1 = q_help(data,Q1,R/((2^B1)));
y2 = q_help(data,Q2,R/((2^B2)));
y3 = q_help(data,Q3,R/((2^B3)));
y4 = q_help(data,Q4,R/((2^B4)));
y5 = q_help(data,Q5,R/((2^B5)));
y6 = q_help(data,Q6,R/((2^B6)));
```

MSE

```
i=1;
n=length(data);
MSE1=0;
MSE2=0;
MSE3=0;
MSE4=0;
MSE5=0;
MSE6=0;

while i<(n+1)
MSE1 = MSE1 + (1/n)*(data(i)-y1(i)).^2;
MSE2 = MSE2 + (1/n)*(data(i)-y2(i)).^2;
MSE3 = MSE3 + (1/n)*(data(i)-y3(i)).^2;
MSE4 = MSE4 + (1/n)*(data(i)-y4(i)).^2;
MSE5 = MSE5 + (1/n)*(data(i)-y5(i)).^2;
MSE6 = MSE6 + (1/n)*(data(i)-y6(i)).^2;
i=i+1;
end

MSE = [MSE1 MSE2 MSE3 MSE4 MSE5 MSE6];
figure;
plot(MSE)
axis([1 6 0 .0041])
xticks([1 2 3 4 5 6])
xticklabels({'8' '16' '32' '64' '128' '256'})
```

```
ylabel('MSE')
xlabel('Levels')
title('MSE vs Number of Quantization Levels')

levels = 32;
TotalBitsUsedInThIsTepresentationBits = ceil(log2(levels))
%%%disp('Total bits used in this Representation :- ',bits)
```

```
end
```



Let us check Output for Fs :-

```
Fs
```

Helper funtion for qunatization

```
function c=q_help(data,Q,a)
   c=[];
   for i=1:length(data)
    idx=0;
    for j =1:length(Q)
        if(data(i)<=Q(j))
            idx=j;
            break;
        end
    end
    if(idx==1)
        idx=2;
    end
    c(end+1)=Q(idx-1)+a;
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% End source() Function %%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Answer 5

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Quest 5.) For your chosen value of quantization levels (say = 16 or 32 or   %
%           higher etc.) encode the samples into bit-streams. For example, if %
%           you have 32 quantization levels, choose 5 bits to represent each  %
%           bit. Find out the total bits used in this representation.         %              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Matlab Code %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Answer :-

```
function stream = Ques_5(y_32)
    symb=unique(y_32);
    sort_sym=sort(symb);
    disp(sort_sym);
    h=1:length(sort_sym);

    stream=[];
    for i=1:length(y_32)
        idx=y_32(i)==sort_sym;
        stream(end+1)=char(h(idx));
    end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% End Ques_5() Function %%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Answer 6.

Contents

Contents

Question 6.)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Quest 6.) For the chosen number of quantization levels, find out an empir-  %
%           -ical probability distribution of the occurrence of the symbols   %
%           coming out of the quantizer by simply counting the number of times%
%           each symbol is occuring dividing by the total number of symbols.  %
%           Use this probability distribution to do a Huffman coding of the   %
%           symbols. Find out the number of bits in the encoded bit-stream.   %
%           Develop a function encode() to do this.%                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Matlab Code %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Answer :-

```
%d=encode(stream);
%disp(d(1));
%disp(['Length of Stream after Simple Encoding = ',num2str(strlength(string(d(1))))]);
```

Function for simple constant length encoding

```
function [Encoder_Output, bitstream, LengthofStreamAfterHuffmanEncoding]=encode(stream)
    sym=unique(stream);
    Encoder_Output=[];
    code=[];
    for j=1:length(sym)
        code=[code,dec_bin(j)];
    end
    for i=1:length(stream)
        idx=sym==stream(i);
        Encoder_Output=[Encoder_Output,code(idx)];
    end
    Encoder_Output=join(Encoder_Output,'');


    % Huffman Coding Part :-
    bitstream=huff_encode(stream);

    % Length of Stream after Huffman Encoding
    LengthofStreamAfterHuffmanEncoding = num2str(length(bitstream));

end
```

Fuction for dec to binany

```
function f= dec_bin(d)
    h=[];
    while(d>0)
        h(end+1)=mod(d,2);
        d=d-h(end);
        d=d/2;
    end
    i=[h zeros(1,5-length(h))];
    f=join(string(fliplr(i)),'');

end
```

fuction for huffman encoding

```
function en_signal=huff_encode(stream)
    [x,p]=compute_prob(stream);
    [~,code]=huffman(x,p);
    disp(keys(code));
    disp(values(code));
    op_stream=[];
    for i=1:length(stream)
        op_stream=[op_stream,code(stream(i))];
    end
    en_signal=op_stream;
end
```

Fuction for calculating probabilities of the symbols empirically

```
function [sym ,p] = compute_prob(stream)
    sym=unique(stream);
    sum_sym=[];
    for i=1:length(sym)
        sum_sym(i)=(sum(stream==sym(i)));
    end
    p=sum_sym./length(stream);
end
```

Function for huffman encoding

```
function [symbol,sym_code] = huffman(x,p)
    [pr,i]=sort(p);
    disp(pr);
    lenp_orig=length(pr);
    l=1;
    mat(l,:)=i;
    while length(pr)>2
        temp=pr(1)+pr(2);
        pr(2)=temp;
        pr(1)=[];
        [pr,k]=sort(pr);
        mat(l+1,:)=[k,zeros(1,l)];
```

```
        l=l+1;
    end
    symbol=x;
    dict_sym={};
    c_dict={};
    for k=1:length(x)
        dict_sym{end+1}=x(k);
        c_dict{end+1}='';
    end
    global codelist;
    codelist=containers.Map(dict_sym,c_dict);
    tree=huffman_tree(mat,x);
    %celldisp(tree);
    huff_gen(tree,[]);
    sym_code=codelist; %dictionary of binary code along with symbols
end
```

Function for making huffman tree

```
function s=huffman_tree(mat,sym)
    [rows,cols]=size(mat);
    s=cell(cols,1);
    for i=1:cols
        s{i}=sym(i);
    end
    for i=1:rows-1
        s=s(mat(i,1:(32-(i-1))));
        s{2}={s{1},s{2}};
        s(1)=[];
    end

end
```

Helper fuction for generating code

```
function huff_gen(s,code)
    global codelist;
    if (isa(s,'cell'))
        huff_gen(s{1},[code 0]);
```

```
        huff_gen(s{2},[code 1]);
    else
        codelist(s)=char('0' + code); %adding code in codelist
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% End encode() Function %%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Published with MATLAB® R2020b

Answer 7.

Contents

Contents

Question 7.)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Quest 7.) Choose a PAM line-code, i.e., 0 is represented by -1 and 1 is     %
%           represented by +1. Design a raised-cosine filter (don t use built %
%           -in functions) that take the inputs as the output of your Huffman %
%           coded data and converts them into subsequent pulses. Vary the roll%
%           off factor of the filter to visualize the output of three conse-  %
%           -cutive input bits (i.e., plot it in the frequency and time domain%
%           (and comment on the ISI. Develop a function pulseshaping() to do  %
%           this. In the time domain, it will consists of a sequence of
%           modifiedsinc functions.
%                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Matlab Code %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Answer :-

Function for pulse shaping with desired alpha

```
function pulseshaping(Fs,code,alpha)
```

```
pulse=[];
for i=1:length(code)
     if(code(i)=='0')
            pulse(end+1)=-1;
     else
            pulse(end+1)= 1;
     end
```

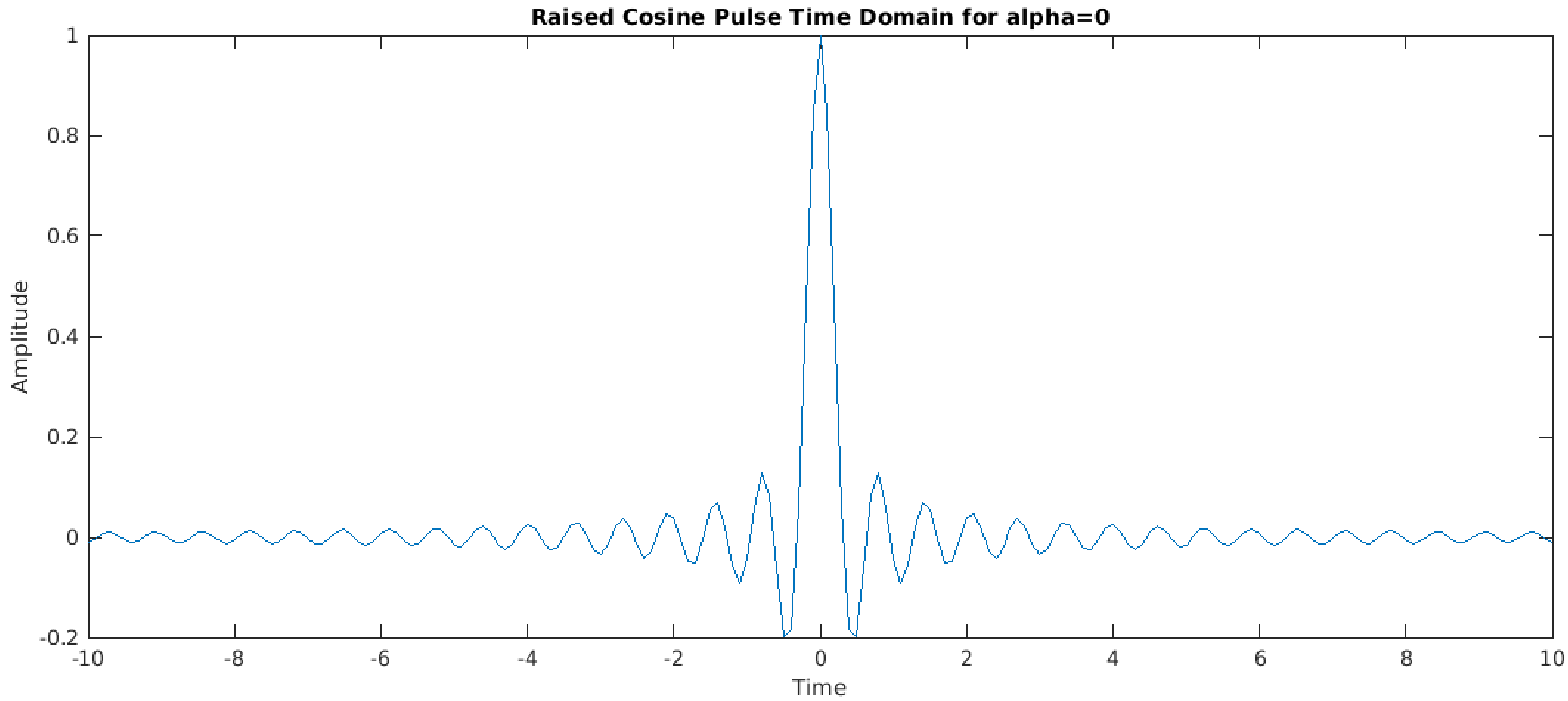
```
end

b1= [-Fs:1/Fs:Fs]*pi ;
b2 = [-Fs:1/Fs:Fs]*alpha;

sin1 = sinc(b1);
cos1 = cos(pi*b2);
cos2 = (1-(2*b2).^2);

zero_p = 10^-10;
cos0 = abs(cos2) < zero_p;
cos3 = cos1./cos2;
cos3(cos0) = pi/4;
```

Raised Cosine filter

```
pt = sin1.*cos3;
f_pt = fft(pt,1024);
```

Raised Cosine Pulse Time Domain for Alpha

```
figure ;plot([-Fs:1/Fs:Fs],[pt]);
xlabel('Time');
ylabel('Amplitude');
title(['Raised Cosine Pulse Time Domain for alpha=',num2str(alpha)]);
```

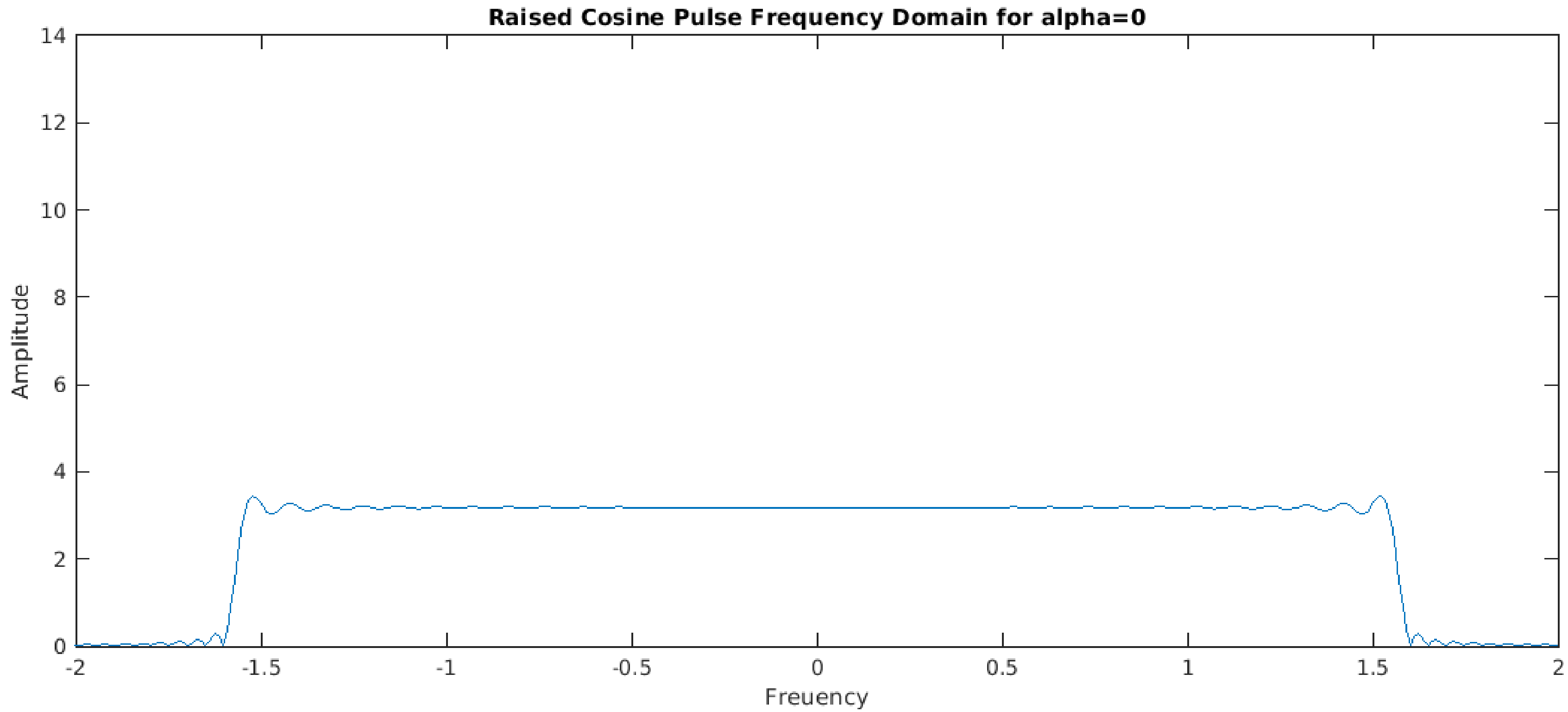


Let us check Output for Fs :-

```
Fs
```

Raised Cosine Pulse Frequency Domain for Alpha

```
figure;plot([-512:511]/1024*Fs, abs(fftshift(f_pt)));
axis([-2 2 0 14])
xlabel('Freuency');
ylabel('Amplitude');
title(['Raised Cosine Pulse Frequency Domain for alpha=',num2str(alpha)]);
```



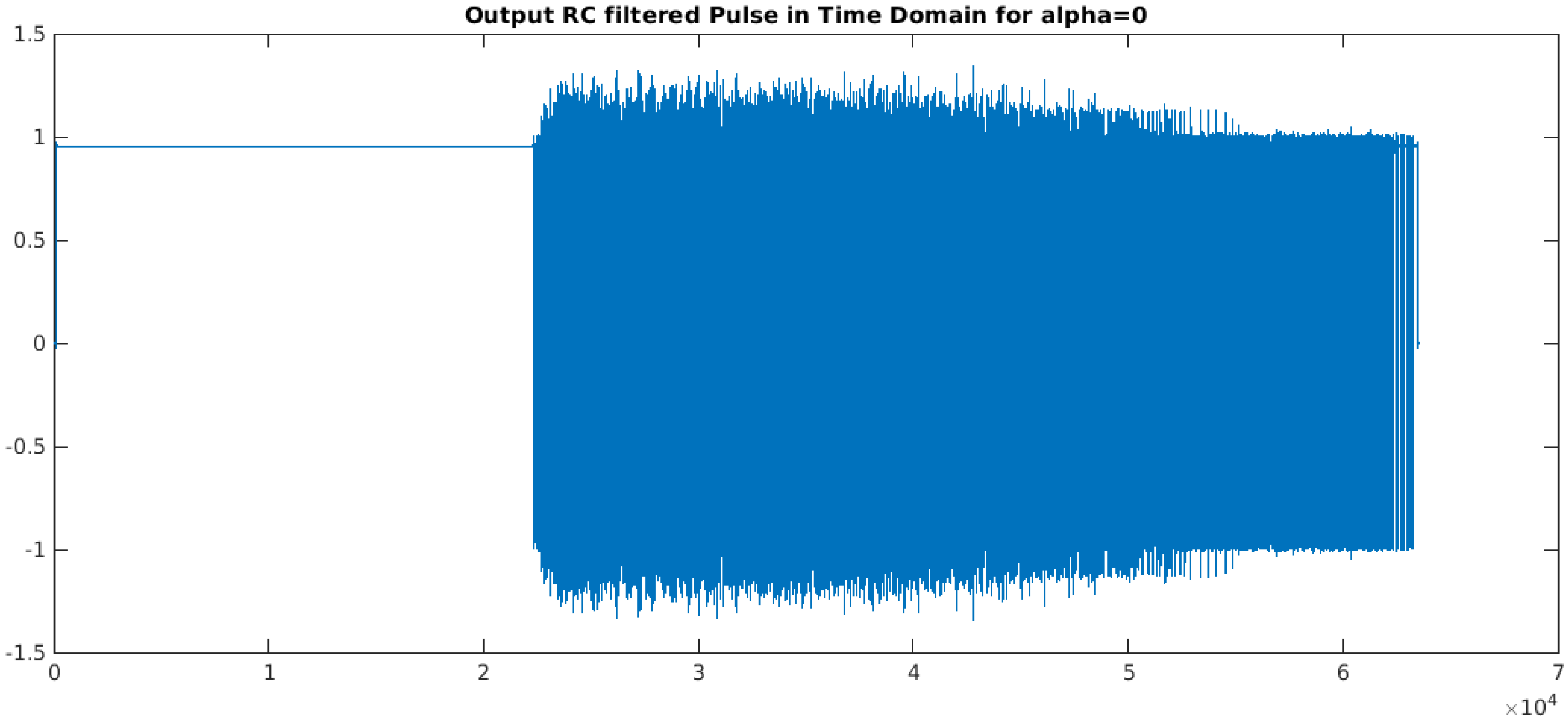
Let us check Output for Fs :-

```
Fs

pr=[];
t2 = -Fs:1/Fs:Fs;
for l = 1:length(t2)
    pr(l) = sinc(t2(l)*pi*10)*((cos(pi*alpha*t2(l)*10))/(1-(2*(alpha)*t2(l)*10)^2));
end
output=conv(pulse,pr);% convulution of pulse and Raised Cosine Filter
t=1:length(output);
f_o = fft(output); % fft of pulse after RC filtering
fc = (0:length(f_o)-1)*Fs/length(f_o);
```

Output RC filtered Pulse in Time Domain for Alpha

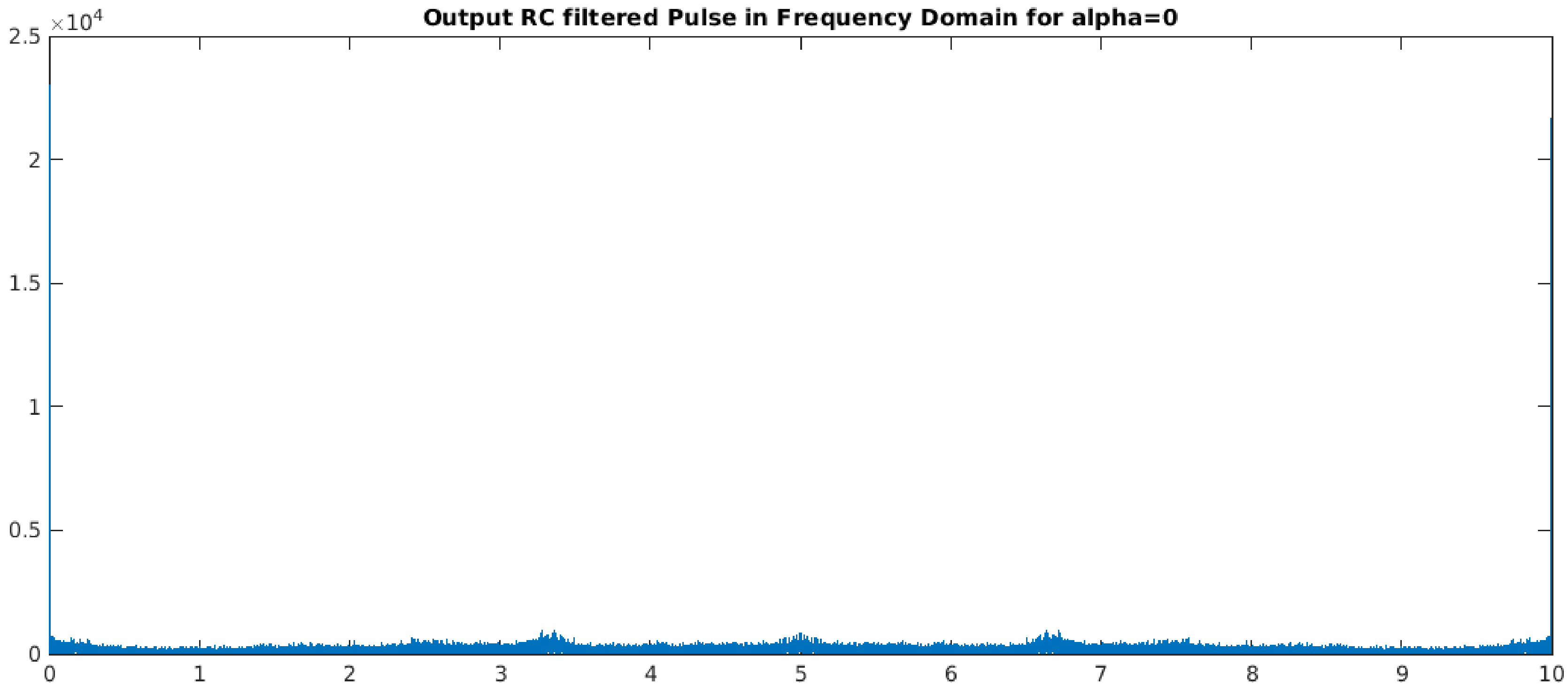
```
figure;
plot(t,output);
```

```
title(['Output RC filtered Pulse in Time Domain for alpha=',num2str(alpha)]);
```

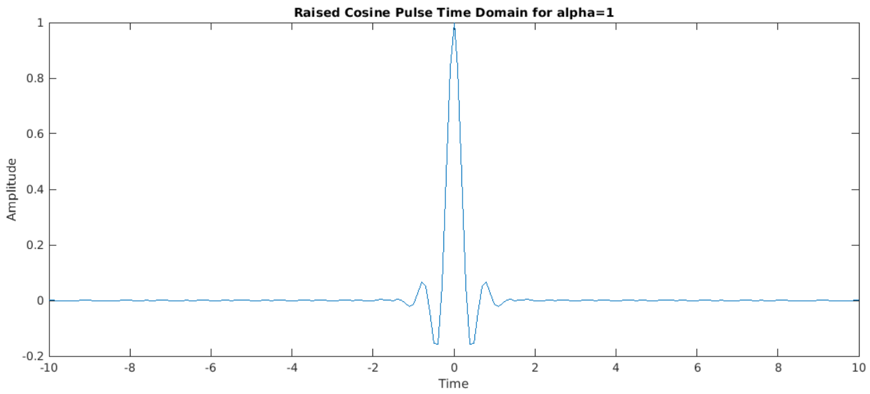


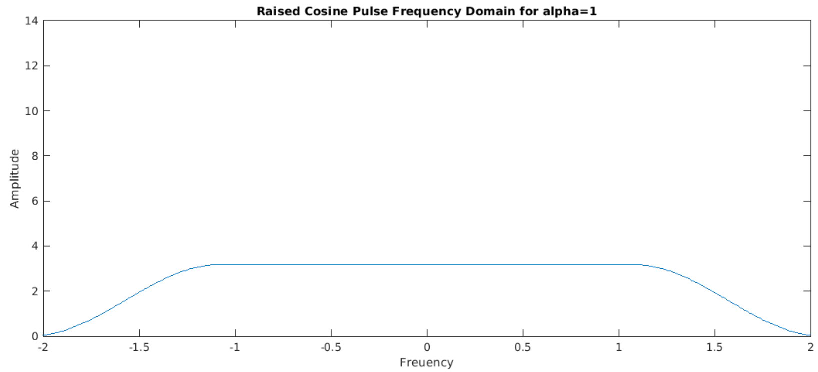
Output RC filtered Pulse in Frequency Domain for Alpha

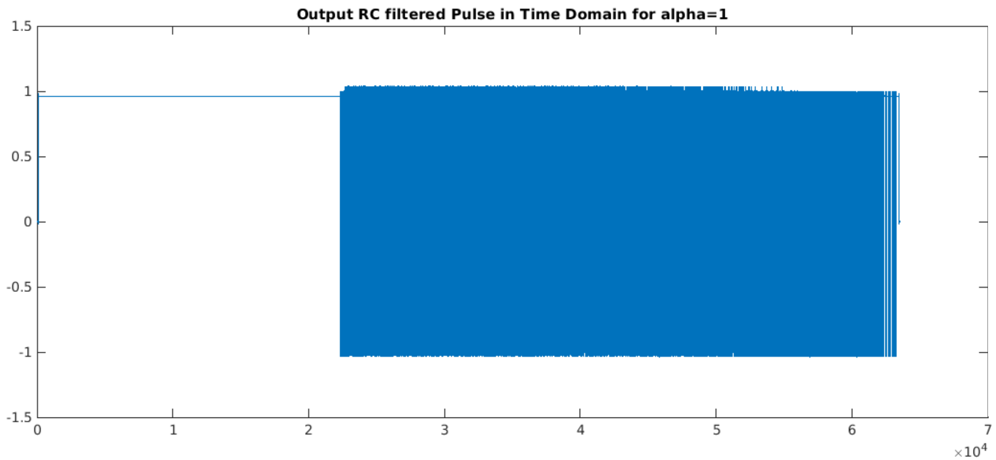
```
figure;
plot(fc,abs(f_o));
title(['Output RC filtered Pulse in Frequency Domain for alpha=',num2str(alpha)]);
```

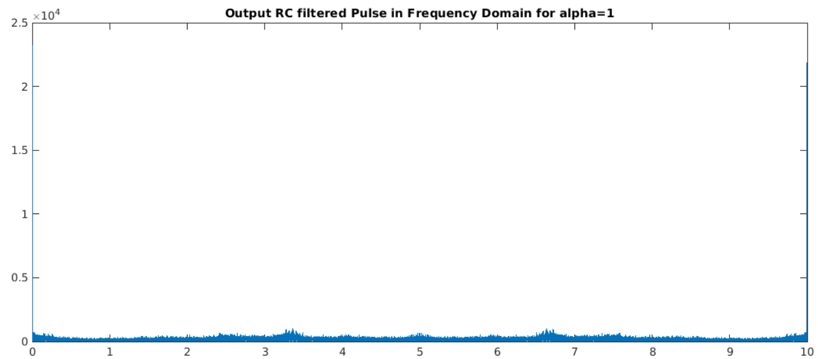

Output RC filtered Pulse in Frequency Domain for alpha=0
×10^4
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6
7
8
9
10

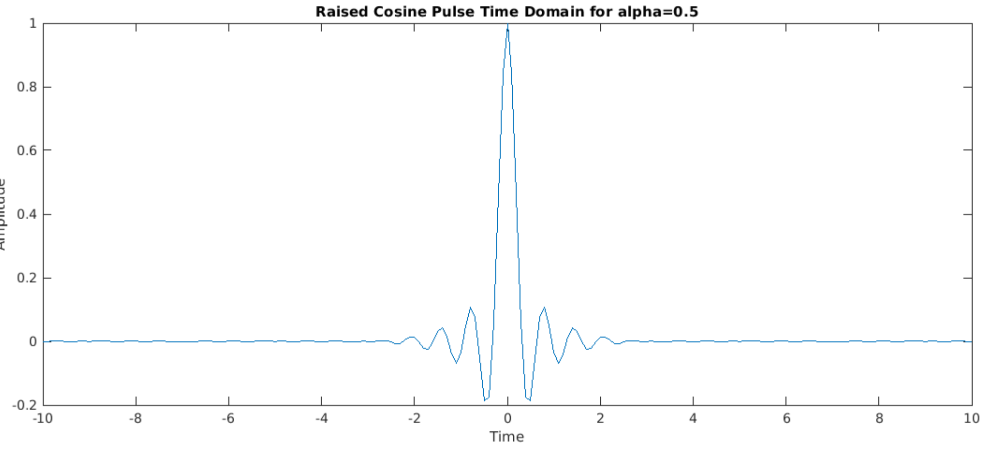
```
end
```

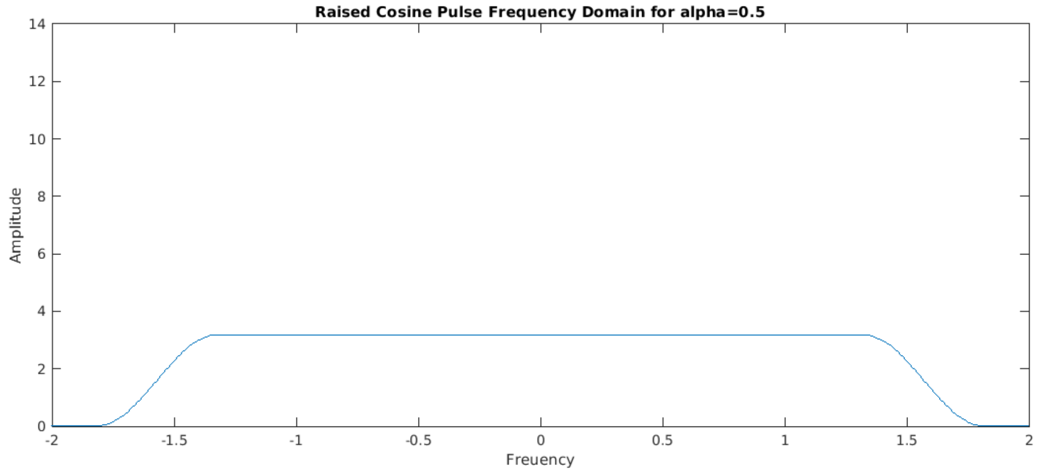

Raised Cosine Pulse Time Domain for alpha=1
Amplitude
Time

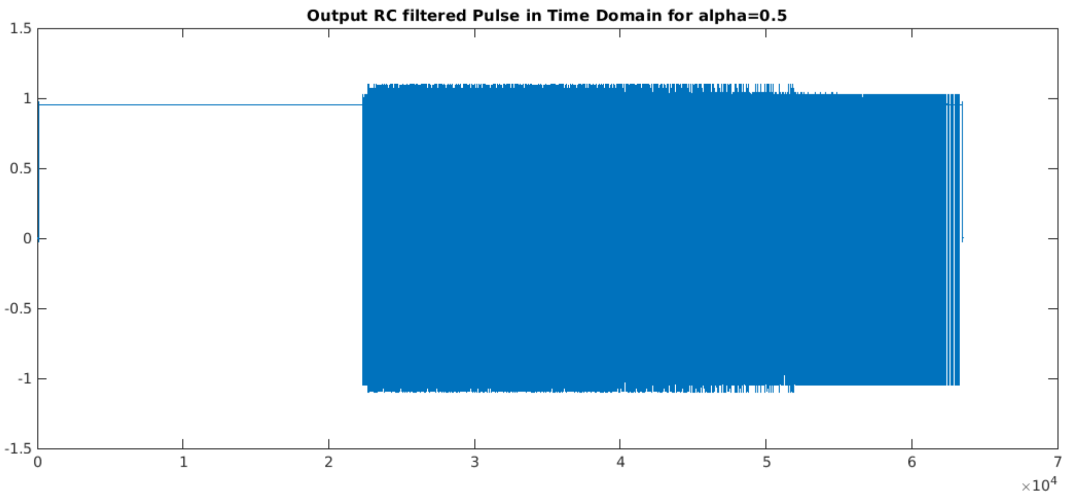

Raised Cosine Pulse Frequency Domain for alpha=1
Amplitude
Freuency

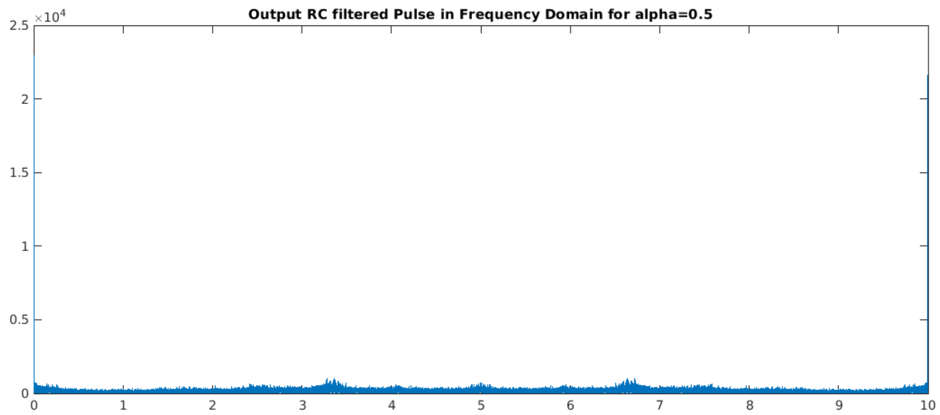

Output RC filtered Pulse in Time Domain for alpha=1


Output RC filtered Pulse in Frequency Domain for alpha=1


Raised Cosine Pulse Time Domain for alpha=0.5
Amplitude
Time


Raised Cosine Pulse Frequency Domain for alpha=0.5
Amplitude
Freuency


Output RC filtered Pulse in Time Domain for alpha=0.5


Output RC filtered Pulse in Frequency Domain for alpha=0.5

Answer 8. Bonus Question

Content :-

Function for sampling and making decision and huffman decoding

Tablelookup()

Contents

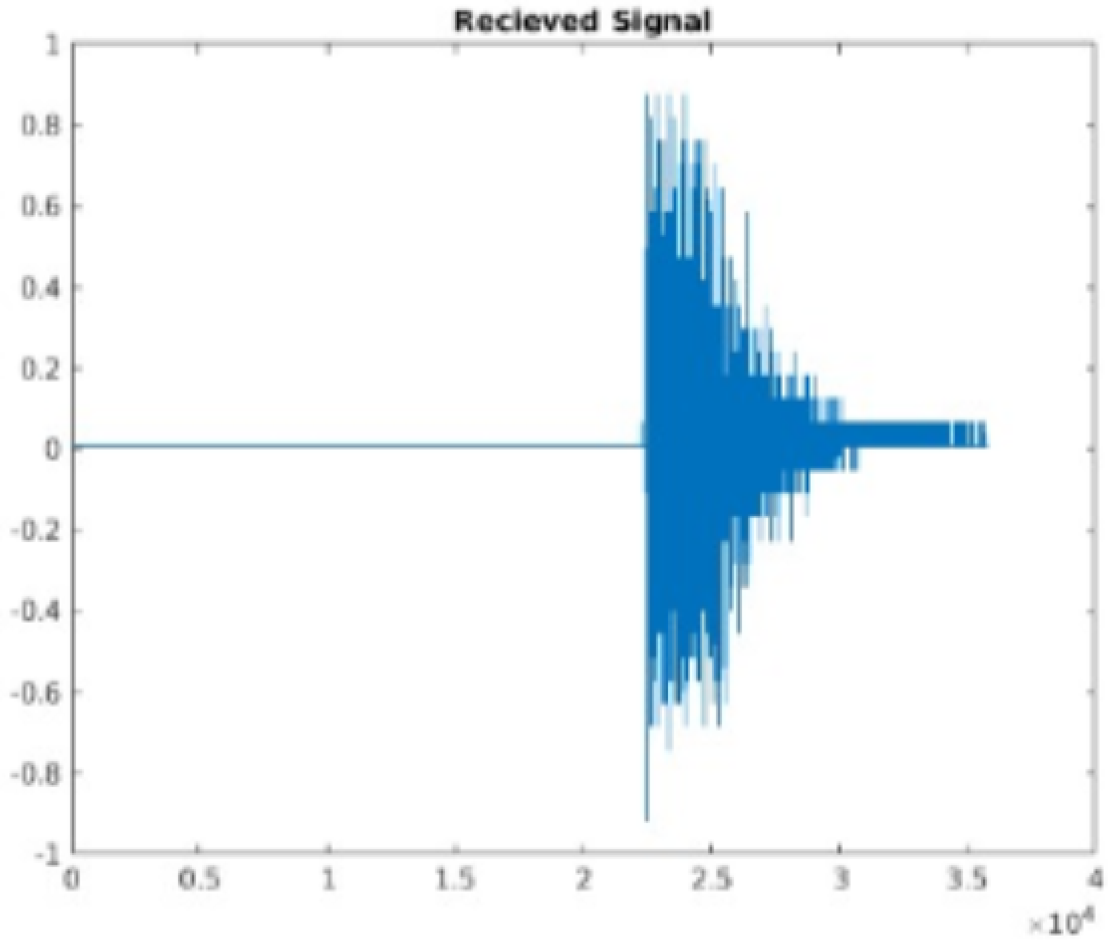
Function for sampling and making decision and huffman decoding

```
function re_sym_stream=reciver(bb_wave,dict,sort_sym)
    upper_th=0.8;
    lower_th=-0.8;
    %disp(bb_wave);
    t=1:length(bb_wave);
    bst=[];
    for i=t
        if(bb_wave(i)>=upper_th)
            bst=[bst, 1];
        else if(bb_wave(i)<=lower_th)
            bst=[bst, 0];
        end
        end
    end
    c_dict={};
    k=keys(dict);
    v=values(dict);
    for j= 1:length(k)
        %disp(k(j));
        A=v{j};
        Output=char(num2cell(A));
        Output=reshape(str2num(Output),1,[]);
        row={k{j},Output};
        c_dict=[c_dict;row];
    end
    sig=huffmandeco(bst,c_dict);
    re_sym_stream=sig;  % re_stream_sym is recieved stream after sampling and huffman decoding
      % program for recieving signal
    re_signal=lookup(re_sym_stream,sort_sym);% re_signal is recived siganl
    figure();
    plot(re_signal);
```

```
    title("Recieved Signal");

end
```

Tablelookup().

```
function re_q=lookup(re_stream,sym_val)
    re_q=[];
    for i=1:length(re_stream)
        re_q(end+1)=sym_val(re_stream(i));
    end
end
```



Published with MATLAB® R2020b

End