LAB5: Baseband Signal Transmission II

I.

Task1:

Ⅰ. Task 1

Consider the following 8-PAM waveforms, where T=1:

$$s_m(t) = A_m g(t), \qquad 0 \le t \le T$$

$$g(t) = \begin{cases} \sqrt{1/T}, & 0 \le t \le T \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad A_m = -3.5 + m, \quad m = 0,1,...,7$$

The mapping rule: 000->0, 001->1,010->2,...,111->7

(1) Give the symbol interval and bit interval.

(2) Give the average energy per symbol and average energy per bit.

(3) Simulate the symbol error rate and bit error rate of 8-PAM when the SNR per bit is 0dB.

Code:

LAB5_1.m

```
T=1;
M=8;
m=log2(8);
Tb=T/m;
SNRindB=0;
   % simulated symbol error rate
   sym_err_prb=symerror(SNRindB)

   % simulated bit error rate
   bit_err_prb=biterror(SNRindB)
```

symerror.m

```
function [p]=symerror(snr_in_dB)
% [p]=symerror(snr_in_dB)
%            symerror    simulates the probability of error for the given
%                  snr_in_dB, signal to noise ratio in dB.
d=1;
SNR=exp(snr_in_dB*log(10)/10);          % signal to noise ratio per bit
sgma=sqrt((21/4)/(2*log2(8)*SNR));          % sigma, standard deviation of noise
N=10000;                          % number of symbols being simulated
% generation of the quarternary data source follows
for i=1:N,
   temp=rand;                                % a uniform random variable over (0,1)
   if (temp<1/8),
      dsource(i)=0;                          % with probability 1/8, source output is "000"
   elseif (temp<2/8),
```

```
        dsource(i)=1;                   % with probability 1/8, source output is "001"
    elseif (temp<3/8),
        dsource(i)=2;                   % with probability 1/8, source output is "010"
    elseif (temp<4/8),
        dsource(i)=3;                   % with probability 1/8, source output is "011"
    elseif (temp<5/8),
        dsource(i)=4;                   % with probability 1/8, source output is "100"
    elseif (temp<6/8),
        dsource(i)=5;                   % with probability 1/8, source output is "101"
    elseif (temp<7/8),
        dsource(i)=6;                   % with probability 1/8, source output is "110"
    else
        dsource(i)=7;                  % with probability 1/8, source output is "111"
    end
end;
% detection, and probability of error calculation
numoferr=0;
for i=1:N,
    % The matched filter outputs
    r=-3.5+dsource(i)+gngauss(sgma);
    % detector follows
    if (r<-3),
        decis=0;                    % decision is "000"
    elseif (r<-2),
        decis=1;                    % decision is "001"
    elseif (r<-1),
        decis=2;                    % decision is "010"
    elseif(r<0)
        decis=3;                    % decision is "011"
    elseif (r<1),
        decis=4;                    % decision is "100"
    elseif (r<2),
        decis=5;                    % decision is "101"
    elseif (r<3),
        decis=6;                    % decision is "110"
    else
        decis=7;                    % decision is "111"
    end;
    if (decis~=dsource(i)),         % if it is an error, increase the error counter
        numoferr=numoferr+1;
    end;
end;
p=numoferr/N;                       % probability of error
```

biterror.m

```
function [p]=biterror(snr_in_dB)
% [p]=biterror(snr_in_dB)
%            biterror    simulates the probability of bit error for the given
%                    snr_in_dB, signal-to-noise ratio in dB.
M=8;                                % quaternary orthogonal signaling
bit=log2(M);                        %the relationship betweent bit and M
E=21/4;
SNR=exp(snr_in_dB*log(10)/10);          % signal-to-noise ratio per bit
sgma=sqrt((21/4)/(2*bit*SNR));              % sigma, standard deviation of noise
N=10000;                            % number of symbols being simulated

numoferr=0;
for i=1:N,
   % generation of the quaternary data source
   temp=rand;                               % a uniform random variable over (0,1)
   if (temp<1/8),
      dsource=[0 0 0];
   elseif (temp<2/8),
      dsource=[0 0 1];
   elseif (temp<3/8),
      dsource=[0 1 0];
   elseif (temp<4/8),
      dsource=[0 1 1];
   elseif (temp<5/8),
      dsource=[1 0 0];
   elseif (temp<6/8),
      dsource=[1 0 1];
   elseif (temp<7/8),
      dsource=[1 1 0];
   else
      dsource=[1 1 1];
   end
% detection, and probability of error calculation
      r=zeros(1,M);
   % matched filter outputs
      for j=1:M
         r(j)=gngauss(sgma);
      end
      Index=dsource(1)*4+dsource(2)*2+dsource(3)+1; %bin2dec(dsource)+1;
      r(Index)=r(Index)+sqrt(E);
   % the detector
   max_r=max(r);
   for l=1:M
```

```
        if(r(l)==max_r)
            h=l-1;
            decision=dec2base(h,2,3)-'0';
             break;
        end
    end
    % Count the number of bit errors made in this decision.
    numoferr=numoferr+sum(dsource~=decision);%find(xor(dsource,decision)==1);
end;
p=numoferr/(bit*N);                    % bit error probability estimate
end
```

Output: Fig.1

Discussion:

1) Give the symbol interval **T=1s** and bit interval $\boldsymbol{T_b = \frac{T}{\log_2 8} = \frac{T}{3} = \frac{1}{3}s}$

2)

average energy per symbol is $\boldsymbol{E_{av} = \frac{1}{8}\sum_{k=1}^{8}\int_0^T s_k(t)^2 dt = \frac{21}{4}}$

average energy per bit is $\boldsymbol{E_b = \frac{E_{av}}{\log_2 8} = \frac{7}{4}}$

In LAB5_1.m, I use two function symerror.m and biterror.m to simulates the probability of error for 0db. In biterror.m, the transformation between binary and decimal number is frequent. And some Function like bin2dec(str) and dec2base(d,b,nin) is useful and concise. bin2dec(str) is not used for it is not difficult to achieve in our code. But dec2base(d,b,nin) is quiet helpful when I find the Operator '%' which means the mod in other Coding language is not exist. Besides, the function sum(dsource~=decision) help to count the error bit. After run the LAB5_1.m, we get the result below.

```
>> LAB5_1

sym_err_prb =

    0.5208


bit_err_prb =

    0.1013
```

II.

Task2:

Simulate the band-limited channel.

(1) Randomly generate a sequence of **ten** 8-PAM signals. The sampling rate fs=1000Hz. Give the signal both in time domain and frequency domain.

(2) Consider a band-limited noiseless channel. The channel response in frequency domain is H(f)=1, |f|≤3Hz, otherwise, H(f)=0. Give the resulted signal waveform after the signal in (1) passing through this channel, both in time domain and frequency domain. Explain the result. What will happen under a larger channel bandwidth?

(Hint: Multiply the spectrum of the input signal with the channel response to get the output signal. Use the functions F2T(.) and T2F(.). You can refer to Problem 1.7 in the Ch.2 ppt.).

Code:

LAB5_2.m

```
% generate 10 8-PAM    signal
N=10;                    % number of symbols being simulated
Sig=zeros(1,3*N);
for i=1:N,
   temp=rand;                          % a uniform random variable over (0,1)
   if (temp<1/8),
      Sig(i:i+2)=[0 0 0];
   elseif (temp<2/8),
      Sig(i:i+2)=[0 0 1];
   elseif (temp<3/8),
     Sig(i:i+2)=[0 1 0];
   elseif (temp<4/8),
      Sig(i:i+2)=[0 1 1];
   elseif (temp<5/8),
      Sig(i:i+2)=[1 0 0];
   elseif (temp<6/8),
      Sig(i:i+2)=[1 0 1];
   elseif (temp<7/8),
     Sig(i:i+2)=[1 1 0];
   else
     Sig(i:i+2)=[1 1 1];
   end
end;
% generate the signal
Fs=1000;Ts=1/Fs;      % fs=1000Hz
T=3;Tb=T/3;              % Sybom /bit time
[t,x]=Sampling(Tb,Fs,Sig);


%the spectrum of this signal
[f,X]=T2F(t,x);
   %generate the lowpass fileter
```

```
H=zeros(1,length(X));
for i=1:length(X)
      if abs(df1(i))<3
            H(i)=1;
      end
end
Y=X.*H;
% output spectrum
% output of the filter as if the signal starts from 0s
Y=Y.*exp(j*2*pi*df1*t(1));
[t,y]=F2T(df1,Y);
t=t/Ts;

figure(1)
%Plot Time-domain
plot(t,x);
xlabel('t');
ylabel('x(t)');
title('The signal x(t)');
figure(2)     %Plot Freq-domain
plot(f, abs(X));
xlabel('f');
ylabel('X(f)');
title('The    spectrum of x(t)');

figure(3)
%Plot Time-domain
plot(t,y);
xlabel('t');
ylabel('y(t)');
title('The Result signal y(t)');
figure(4)     %Plot Freq-domain
plot(f, abs(Y));
xlabel('f');
ylabel('Y(f)');
title('The    Result spectrum of y(t)');
```

Sampling.m

```
function [T,Samp_Sig]=Sampling(Tb,Fs,sig)
% Fucntion Name:Sampling
%Input: Tb,Fs:sig     OutPut:Samp_Sig
%When you call the Function ,u input the tiem for a bit,the
%Sampling rate and the source signal,then output the Samplint Signal
Ts=1/Fs;
Sig=sig;
```

```
len=length(Sig);
T=0:Ts:len*Tb-Ts;          %Widen the Samplint Point
Samp_Sig=T;

for i=0:1:len-1                    %Sampling
    for j=1:1:Tb/Ts
        Samp_Sig(i*Tb/Ts+j)=Sig(i+1);
    end
end

for k=1:1:length(Samp_Sig)     %set the data for different Symbol
    if(Samp_Sig(k)==0)
        Samp_Sig(k)=-1;
    end
end

end
```

Output: Fig.1&Fig.2 &Fig.3&Fig.4&Fig.5

Discussion:

1)

In this task, I used the method In task1 to generate ta sequence of **ten** 8-PAM signals. Then, I use the self-written function Samp_Sig= **Smapling** (Tb,Fs,sig) which is used in Lab4 to Sample the source signal and return it and its time axis. Lastly, Fig.1&Fig.2 plot the wave to show the wave.

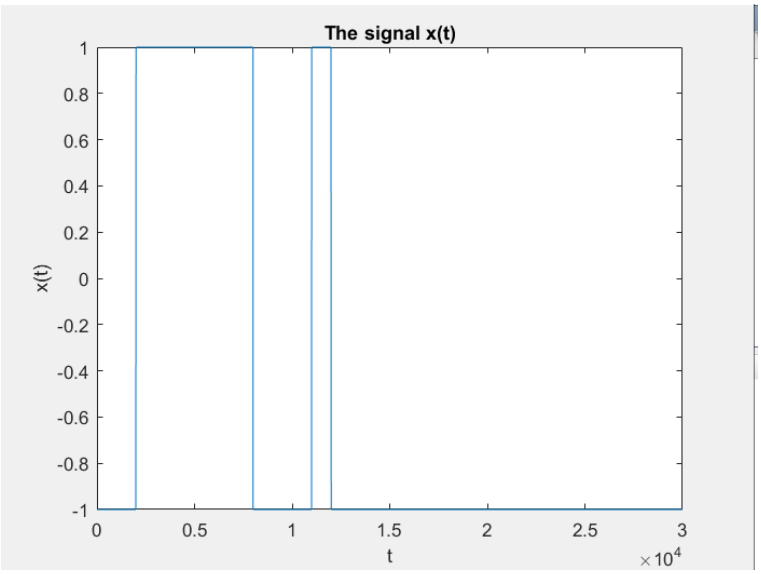

Fig1.the input signal wave

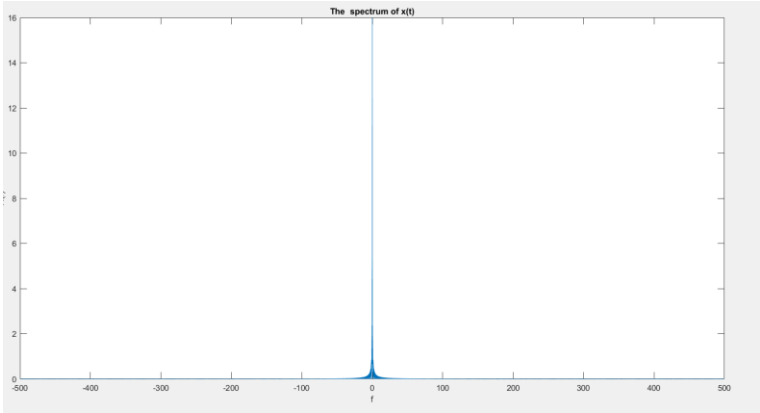

Fig2.the frequency domain of the input signal

2) In Fig.3 and Fig.4 show the wave pass through a band-limited noiseless channel. As we all know, the low pass filter can be expensed into many sin functions which means it has infinite frequency. When I set the cut-off frequency 2Hz, I can find the frequency lose in Fig.4.And in Fig.3,we can find the edge of the changing bit gets ripple which is equal to the effect of smooth .So if we gets the cut-off frequency bigger like 100 in Fig.5, we gets the different scene .In Fig.5, more detail is reserved which is the High frequency component.

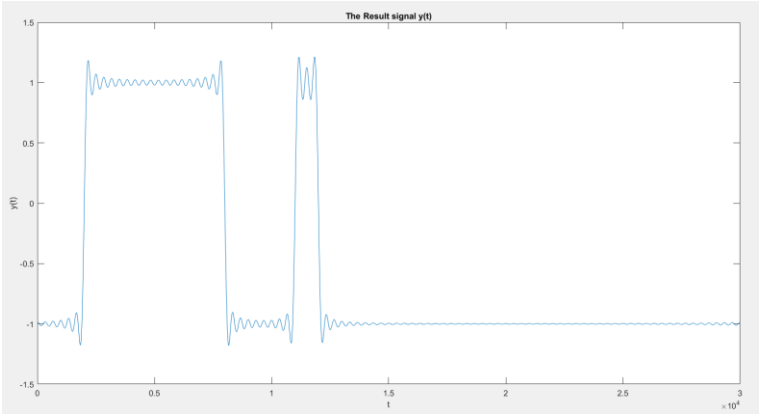

Fig3.the result signal wave

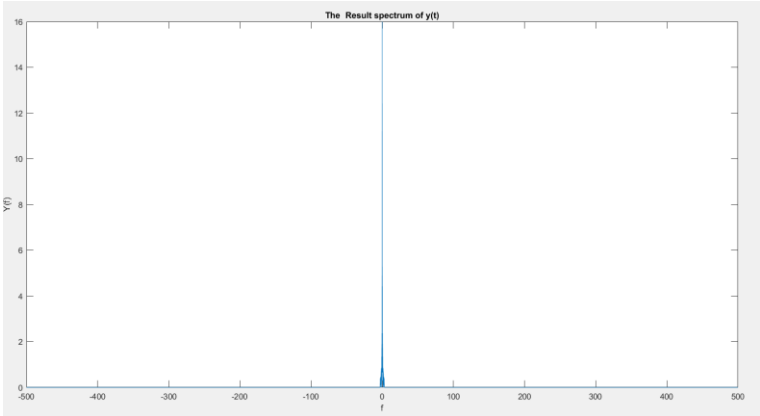

Fig4.the Spectrum of the result signal wave (3HZ)

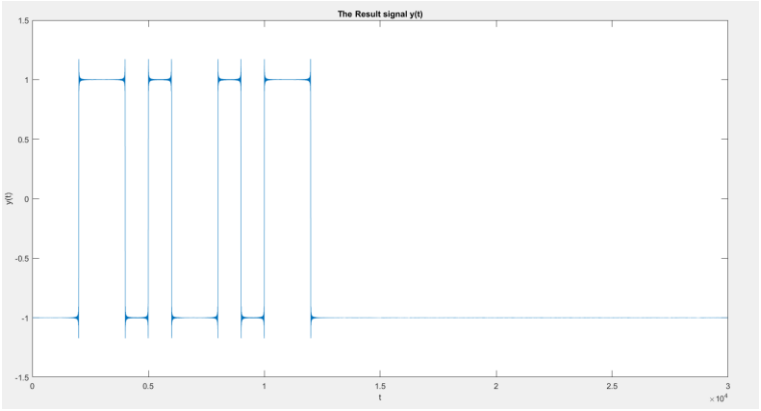

Fig5.the Spectrum of the result signal wave (100HZ)

III.

Task3:

Try to illustrate using Matlab that the raised cosine waveform satisfies the following condition:

$$\sum_{m=-\infty}^{\infty} X\left(f+\frac{m}{T}\right)=T$$

Code:

LAB5_3.m

```
clear all; close all;
Ts= 1 ;
df = 1.0/(20.0*Ts);      %20s duration
f= - 2/Ts:df :2/Ts;
N=length(f);
alpha=[0,0.5, 1] ; %Roll off coefficient

for n = 1 : length(alpha)                    %different alpha
    for k = 1 : length( f)                      %cetain alpha
        if abs(f(k)) > 0.5 *(1 + alpha(n))/ Ts        %rc assignmant
            Xf( n,k)=0;
        elseif abs( f(k) ) < 0.5 *(1 - alpha(n))/Ts
            Xf(n,k)=Ts;
        else
            Xf (n, k )=0.5 *Ts *( 1 + cos( pi *Ts/ (alpha( n) + eps) *(abs(f ( k )) - 0.5* (1 -
alpha( n)) / Ts) ) );
        end
    end
    Xf_L2=[Xf(n,1+(N-1)/2:1+(N-1)*3/4),zeros(1,N-(N-1)/4-1)];   %cetain fre is -2*fs
    Xf_R2=[zeros(1,N-(N-1)/4-1),Xf(n,1+(N-1)/4:1+(N-1)/2)];      %cetain fre is   2*fs
    Xf_L1=[Xf(n,1+(N-1)/4:1+(N-1)*3/4),zeros(1,N-(N-1)/2-1)];   %cetain fre is   -fs
    Xf_R1=[zeros(1,N-(N-1)/2-1),Xf(n,1+(N-1)/4:1+(N-1)*3/4)];   %cetain fre is     fs
    Xf_sum=Xf_L2+Xf_L1+Xf(n,:)+Xf_R1+Xf_R2;                          %get the sum
    %plot them in a figure
```

```
figure;
  plot( f,Xf_L2,f,Xf_L1,f,Xf(n,:),f,Xf_R1,f,Xf_R2,f,Xf_sum);
xlabel ('f/Ts'); ylabel('RC spectrum') ; legend('Cetain Freqence -2*fs','Cetain Freqence -fs','original','Cetain Freqence fs','Cetain Freqence 2*fs','Sum');
end
```

Output: Fig.6&Fig.7 &Fig.8

Discussion:

In this task , I use the raise-cosine frequency response. In code, we can find the three caffeine α.When α equals 0,it turns to the sinc function and it is not the filter without ISI which can be seen in Fig.6.When α gets bigger, it satisfy the the condition. In Fig.7 and Fig.8,we find other sum equals 1 which is the num of T。

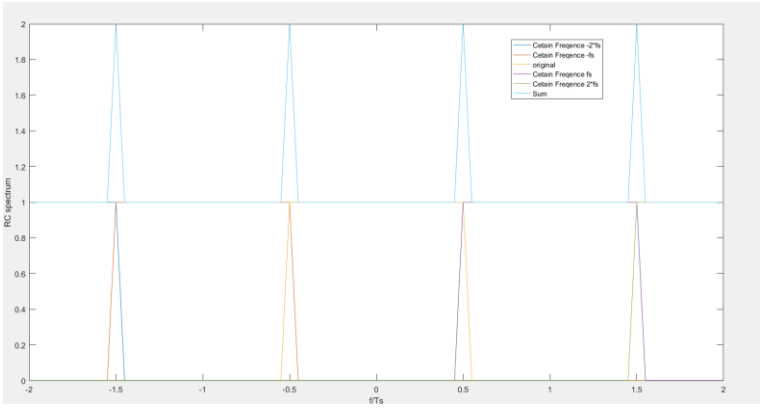

Fig.6 raise-cosine frequency response(α=0)

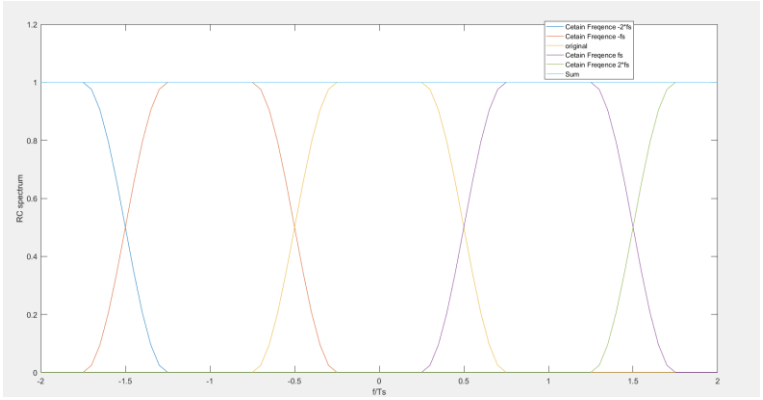

Fig.7 raise-cosine frequency response(α=0.5)

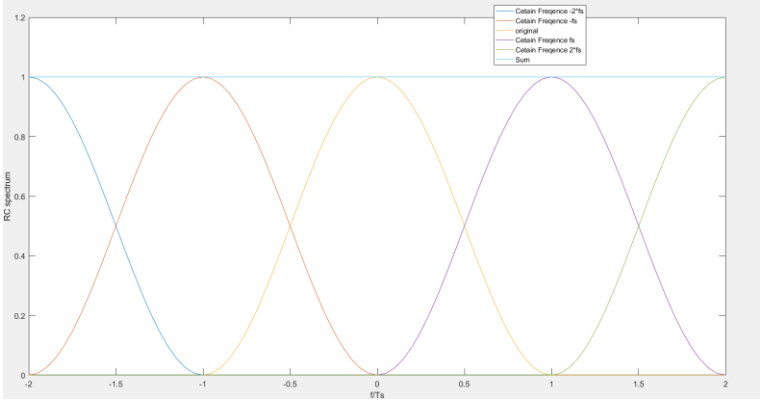

Fig.8 raise-cosine frequency response(α=1)