|  |
| --- |
| C:\Users\muyuan.tong10\Desktop\121.png |
| EEE411 Advanced Signal Processing |
| MATLAB Individual Assignment II |
|  |
| Muyuan Tong |
| ID: 10114703  Email: muyuan.tong10@student.xjtlu.edu.cn |

**Problem 1 Mapping**

In order to minimize the difference of adjacent of bits for a symbol, gray code is implemented in this 4ASK modulation.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **4ASK Mapping (gray coding)** | | | **Implementation** | | | |
|  |  |  |  |  |  |  |  |
| **k=1** | 0 | 0 | 4 | 1 | -2 | -2 | 4 |
| **k=2** | 0 | 1 | 2 | 1 | 2 | 0 | 2 |
| **k=3** | 1 | 1 | -2 | -1 | 2 | 0 | -2 |
| **k=4** | 1 | 0 | -4 | -1 | -2 | -2 | -4 |

**Table.1 4ASK Mapping and a simple implement arithmetic**

As Table.1 shows, the modulation symbols satisfied

Corresponding MATLAB function is defined as following list.

function [ Output ] =ASKmapping( Input )

b=reshape(Input,2,[])';

Output= (-(2\*b(:,1)-1)).\*abs((2\*(2\*b(:,2)-1)-(~b(:,2)\*2)));

end

**Listing.1 Defined MATLAB Function for 4ASK mapping with gray code**

The code for testing is given as below, both bit pairs and a sequence of data bits has been tested with correct response.

clc;clear;close all;

x=input('Please input a bit pair/sequence:');

y=ASKmapping(x);

**Listing.2 MATLAB code to test 4ASK mapping**

**1.1 Test different bit pairs:**

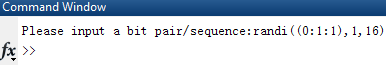
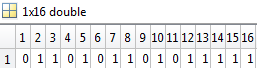


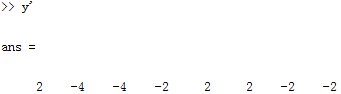
 



**Figure.1 Testing results of four different bit pairs**

**1.2 Test a sequence of data bits: random sequence (like 16 bits)**

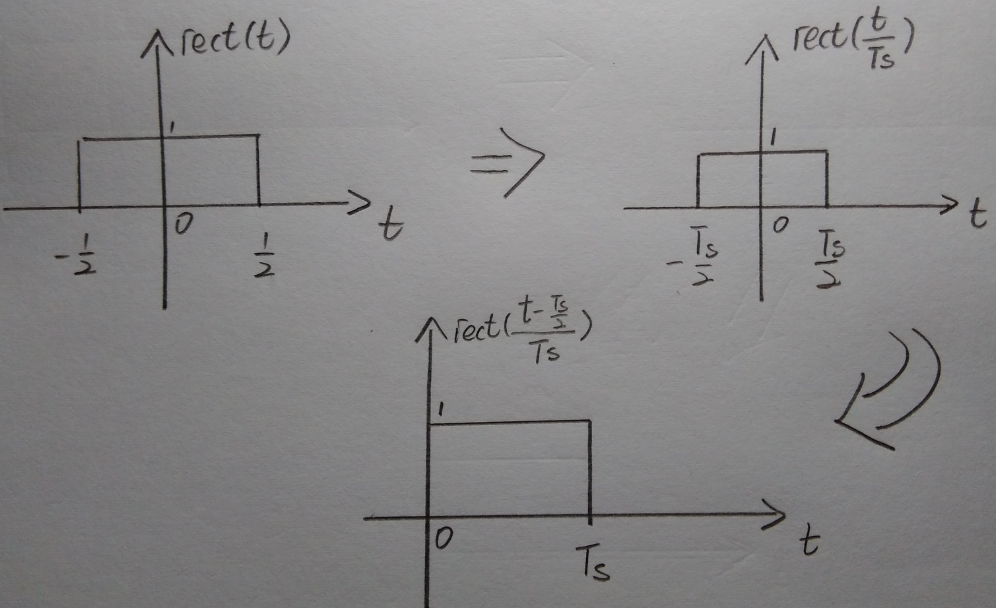


**Figure.2 Testing results of a 16-bit random sequence [0 1 1 0 1 0 1 1 0 1 0 1 1 1 1 1]**

**Problem 2 transmit pulse**

The transmit pulse, which is generated by scaling the function in x-axis by, and shift the result to right side along x-axis by unit.

**2.1 Hand sketching:**



*Scaling*

*Shift*

**Figure.3 hand sketch of transmit pulse (changes from )**

**2.1 Plot transmit pulse**

The symbol period is, in order to get 10 sample intervals per symbol period, the sampling interval should be

Corresponding MATLAB code is showing below. There are 11 sample points in one symbol period. The first sample point is located at with 0.5 amplitude, second sample point is located at, and the eleventh sample point should located at with 0.5 amplitude.

clc;clear;close all;

Ts=0.002;

t=0:Ts/10:Ts;

p=rect((t-Ts/2)/Ts);

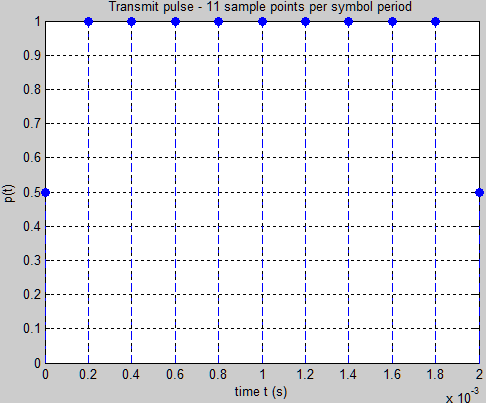
stem(t,p,'fill','b-.');

title('Transmit pulse – 11 sample points per symbol period')

xlabel('time t');

ylabel('p(t)');grid on;

**Listing.4 MATLAB code to plot the transmit pulse**



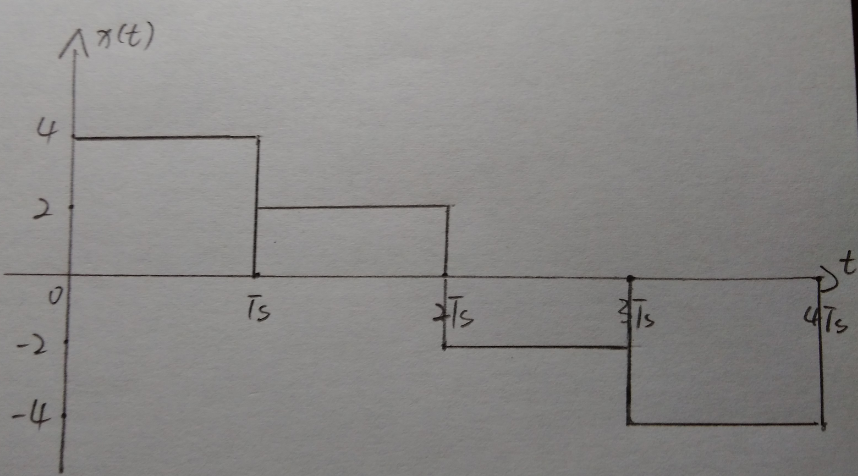
**Figure.5 MATLAB plot of transmit pulse**

**Problem 3 transmit pulse for symbols**

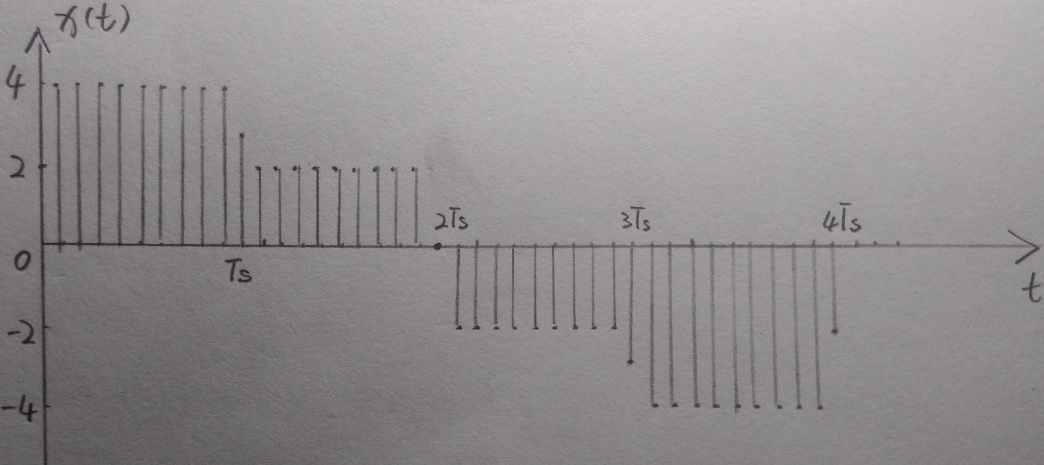
Assume the sequence is exact,, and, which are four chosen modulation symbols. The corresponding transmit signal should be like following expression:

Notice that, the original transmit pulse is already started from. Hence, no need to shift transmit pulse for symbol.

Corresponding hand sketch of this transmit signal is showing as Figure.6.



**Figure.6 transmit signal of a sequence consist of,,** **and**



**Figure.7 transmit signal of this sequence after sampling (using the given MATLAB function)**

Figure.7 is the hand sketch of sampled transmit signal. One thing has to be mentioned is that, the sampled function used here is same as that in MATLAB, which forced the boundary points to 1/2.

**Problem 4 MATLAB plot of transmit pulses for different symbols**

The MATLAB script is showing bellow, which plot the signal, and this MATLAB script includes the given code in lab-script.

clc;clear;close all;

a=4;

Ts=0.002;

l=1:length(a);

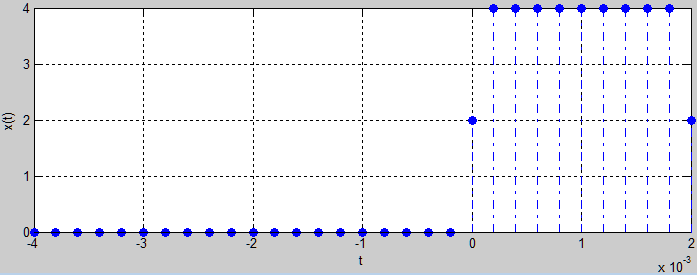
t=-2\*Ts:Ts/10:Ts\*length(a);

p=@(t)rect((t-Ts/2)/Ts);

xx=a\*p(t);

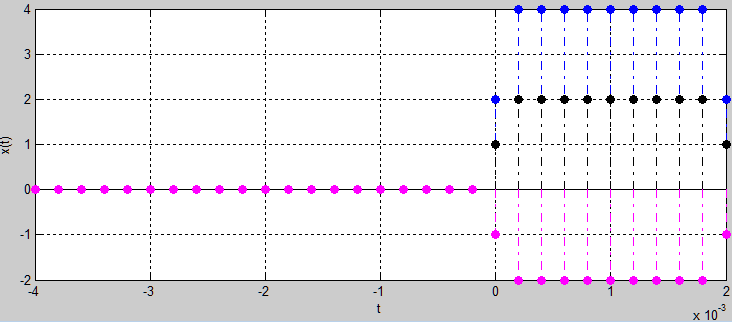
stem(t,xx,'fill','b-.');grid on;xlabel('t');ylabel('x(t)');

**Figure.8 MATLAB script to plot the signal x1(t)**



**Figure.9 the plot for x1(t)**

Next, plot the signals and in the same figure (using ‘hold on’ to indicate respectively), as Figure.10 shows bellow.



**Figure.10 plot at same figure with same start time**

**Problem 5 modulator**

Modified the given sample MATLAB code, add a new variable l, which indicates the sequence number of symbols to be transmitted, and it also related to how far the transmit pulse need to shift for current symbol. Using the function to get corresponding result with two variables t and l in function p.

The MATLAB function of modulator is showing below, as well as the plot of transmit signal is showing in Figure.13. Assume the input sequence is [4; 2; -2; -4].

function [ xt ] = modulator( a, Ts, t)

l=1:length(a);

p=@(t,l)rect((t-(2\*l-1)\*Ts/2)/Ts);

xt=a'\*bsxfun(p,t,l');

end

**Figure.11 MATLAB function for modulator**

clc;clear;close all;

a=[4; 2; -2; -4];

Ts=0.002;

To=Ts/10;

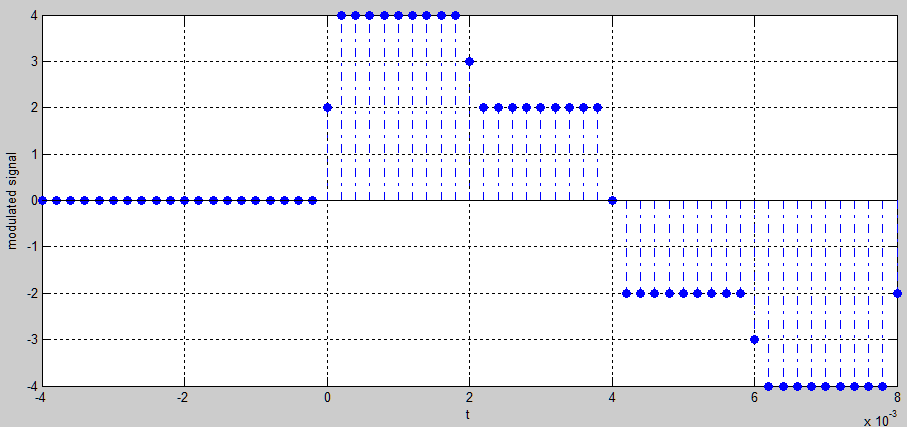
t=-2\*Ts:To:Ts\*length(a);

xt=modulator(a, Ts, t);

stem(t,xt,'fill','b-.');grid on;

xlabel('t');ylabel('modulated signal');

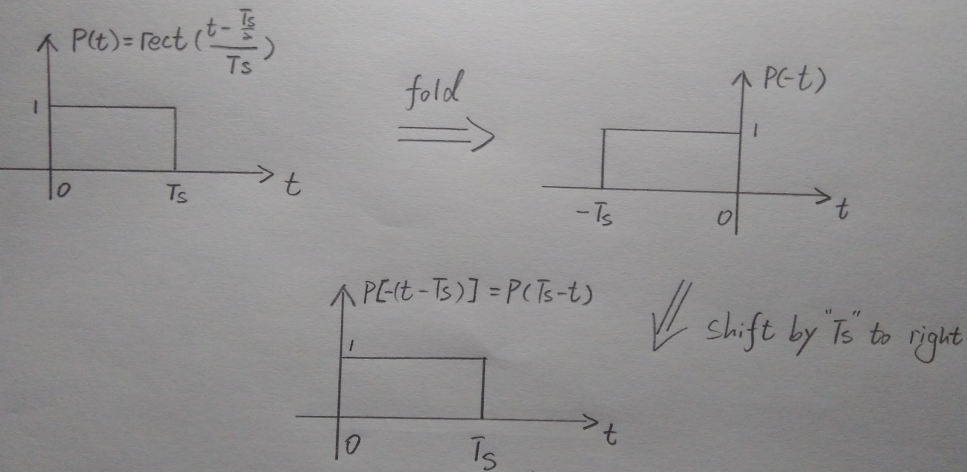
**Figure.12 MATLAB script for modulation**



**Figure.13 the plot of output**

**Problem 6 matched filter**

The matched filter is generated by fold the transmit pulse about y-axis and then shift the result to right by Ts.



**Figure.14 hand sketch of impulse response of the matched filter**

Detail about matched filter is demonstrated in **Problem 11**.

**Problem 7 matched filter**

The channel is noise free. Then we can get:

Then passing through the matched filter the output is:

***(Distribute Law)***

A property used here is distributive law of convolution, which is:

The proof is given as follow:

**Problem 8 matched filter convolution**

**Theoretical result: (k=0, just care about the first symbol)**

Assume the transmit pulse and the matched filter are defined as follow:

where .

Then, calculate the convolution of and:

**MATLAB result: (k=0, just care about the first symbol)**

The MATLAB script to implement this convolution is showing bellow:

clc;clear;close all;

Ts=0.002;

t=0:Ts/10:2\*Ts;

P=@(t)rect((t-Ts/2)/Ts);

p=P(t);

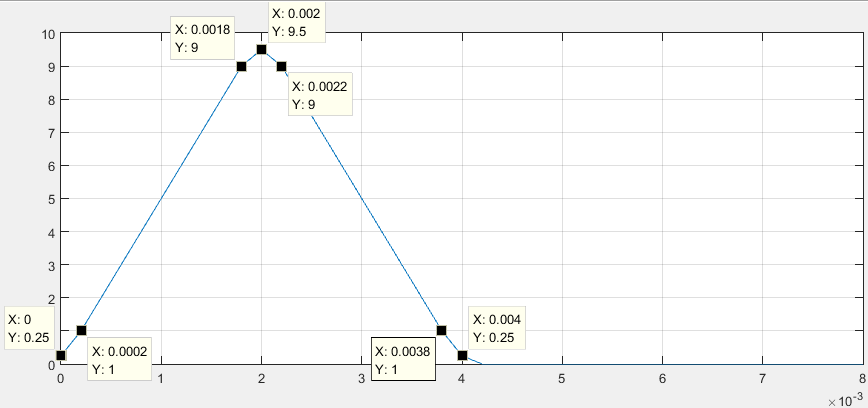
h=P(Ts-t);

y=conv(p,h);

ty=Ts/10\*(1:length(y))-Ts/10; % start from 0

plot(ty,y);grid on;

**Figure.15 MATLAB script to implement convolution of transmit pulse and matched filter (k=0 for first symbol)**



**Figure.15 MATLAB plot of the convolution result**

Now, the MATLAB plot is same as the theoretical result, which is expected.

**Problem 9 outputs of matched filter**

**Transmit signal: x1(t)**

a=4; % symbol a1

Ts=0.002;To=Ts/10;

t=0:To:2\*Ts;

P=@(t)rect((t-Ts/2)/Ts);

p=P(t);

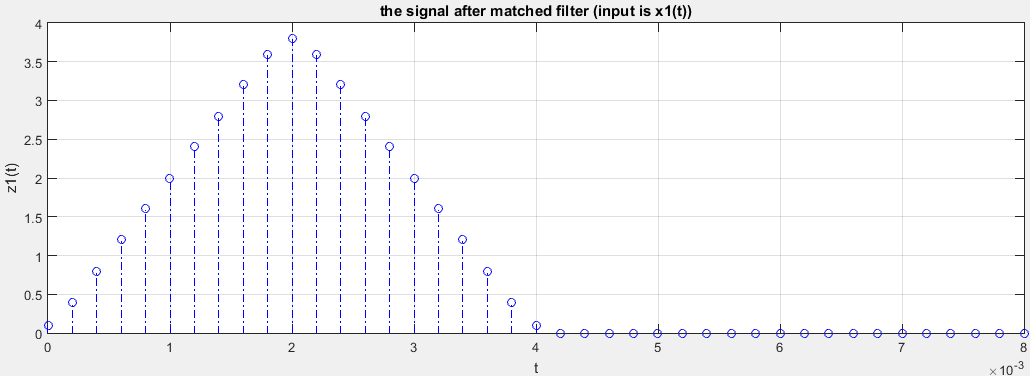
h=P(Ts-t);

z=To/Ts\*a.\*conv(p,h); % To/Ts is used to scaling the amplitude after conv

ty=To\*(1:length(z))-To; % start from 0

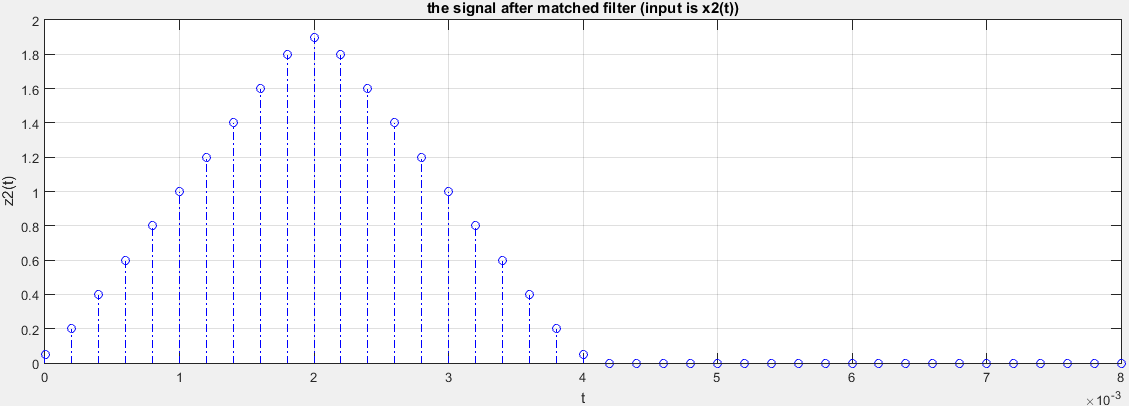
plot(ty,z);grid on;

**Figure.16 MATLAB script to generate signal z(t) for x1(t)**



**Figure.17 the plot of z1(t)**

Using similar MATLAB script to plot z2(t) as follow:



**Figure.18 the plot of z2(t)**

The result shows that, there is a scaling of by multiply. The result is expected as this equation:

Now, transmit the full signal, which contains. The MATLAB script is showing below.

clc;clear;close all;

b=[0 0 0 1 1 1 1 0]; % source code

%%%%%%%%%%%%%%%%% Transmitter %%%%%%%%%%%%%

%%%% mapper %%%%

a=ASKmapping( b );

%%%% sampler %%%%

k=1:length(a);

Ts=0.002;To=Ts/10;

t=-2\*Ts:To:length(a)\*Ts;

p=@(t,k)rect((t-(2\*k-1)\*Ts/2)/Ts);

x=a'\*bsxfun(p,t,k');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%% Channel %%%%%%%%%%%%%%%%%

y=x; % no noise

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%% Receiver %%%%%%%%%%%%%%%%

%%%% matched filter %%%%

tp=0:To:Ts;

h=p(tp,1);

z=To/Ts\*conv(y,h);

tz=To\*(1:length(z))-To-2\*Ts;

%%%% downsampler %%%%

zk=downsample(z,Ts,To);

t\_down\_sampled=-Ts:Ts:Ts\*length(a);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%% Plots %%%%%%%%%%%%%%%%%

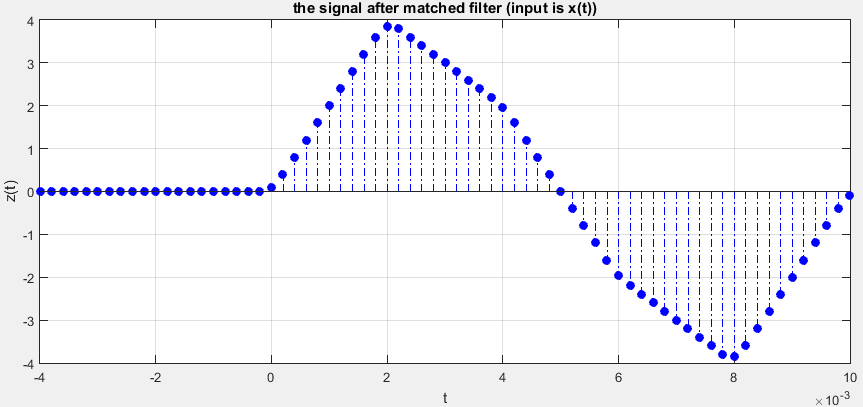
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

stem(tz,z,'fill','b-.'); grid on;

title('transimit signal for x(t)');

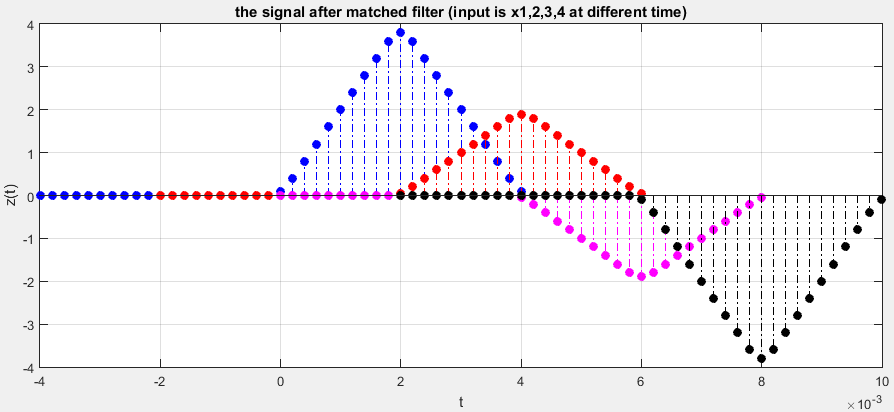
xlabel('t');ylabel('z(t)'); % the signal z after matched filter

**Figure.19 MATLAB script of transmitter and receiver (just include matched filter)**



**Figure.20 the plot of full transmit signal x(t) passing through the matched filter**

This result can be produced by sum four individual signals from time t=0 to the end, as Figure. 21 shows below. These four signals with different color can be seen as four symbols transmit at different time and meanwhile passing through the matched filter.



**Figure.21 the plot of matched transmit signal x1, x2, x3 and x4 (using ‘hold on’ in MATLAB to plot it)**

There are collisions if transmit and match them one by one, so as the whole point of view, the corresponding output of matched filter for the full transmit signal should like the Figure.20 shows, which has summation of collision regions.

**Problem 10 downsample**

From the results of previous questions, it is clear to see that the optimal sampling time for z(t) should be:

(signal started to transmit at t=0)

As Figure.21 shows, this downsample time gets lowest interference between each symbol, and maximum amplitude of output of matched filter.

function [ z\_sampled ] = downsample( z,Ts,To )

z\_sampled=zeros(1,length(z)-Ts/To); % initialize the sampled matrix

z\_sampled=z(Ts/To+1:Ts/To:end-Ts/To);

end

**Figure.22 MATLAB function to do down-sampling (started from 11th point in vector and 10 points interval)**

clc;clear;close all;

b=[0 0 0 1 1 1 1 0]; % source code

%%%%%%%%%%%%%%%%% Transmitter %%%%%%%%%%%%%

%%%% mapper %%%%

a=ASKmapping( b );

%%%% sampler %%%%

k=1:length(a);

Ts=0.002;To=Ts/10;

t=-2\*Ts:To:length(a)\*Ts;

p=@(t,k)rect((t-(2\*k-1)\*Ts/2)/Ts);

x=a'\*bsxfun(p,t,k');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%% Channel %%%%%%%%%%%%%%%%%

y=x;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%% Receiver %%%%%%%%%%%%%%%%

%%%% matched filter %%%%

tp=0:To:Ts;

h=p(tp,1);

z=To/Ts\*conv(y,h);

tz=To\*(1:length(z))-To-2\*Ts;

%%%% downsampler %%%%

zk=downsample(z,Ts,To);

t\_down\_sampled=-Ts:Ts:Ts\*length(a);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%% Plots %%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

stem(tz,z,'fill','b-.'); grid on;

title('transimit signal for x(t)');

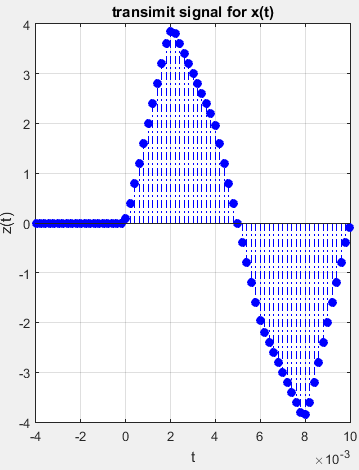
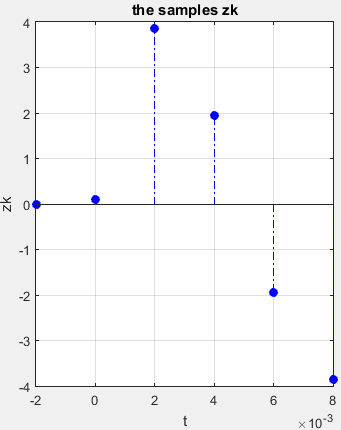
xlabel('t');ylabel('z(t)'); % the signal z after matched filter

figure,stem(t\_down\_sampled,zk,'fill','b-.');

title('the samples zk');grid on;

xlabel('t');ylabel('zk'); % the signal zk after downsample

**Figure.23 Add the downsampler to previous MATLAB script in Figure.19 and plot the result**

**(a) (b)**

**Figure.24 (a) previous signal after matched filter; (b) the down-sample result from (a)**

The first symbol is located at t=Ts=0.002s, and second is located at t=2Ts=0.004s, and so on. Due to we defined in MATLAB has 0.5 boundaries, so we cannot reach exact values for sometimes.

**Problem 11 AWGN**

AWGN standard for additive white Gaussian noise, which is a basic noise model satisfied normal distribution in time domain with an average value of zero.

Now, adding the AWGN channel between the transmitter and the receiver. The previous noiseless channel is changed to AWGN channel, and corresponding MATLAB script in Figure.23 is modified.

**Previous noiseless channel**

%%%%%%%%%%%%%%%%% Channel %%%%%%%%%%%%%%%%%

y=x;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

**AWGN channel**

%%%%%%%%%%%%%%%%% Channel %%%%%%%%%%%%%%%%%

varinoise=1;

y=x+sqrt(varinoise)\*randn(size(x)); % +AWGN

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Now, plot the transmit signal, the noisy receive signal, the signal at the matched filter output, and the print out the samples. Corresponding MATLAB script of ‘Plots’ part is also modified as Figure.25. The results is displayed in subplots form, as Figure.26 with various noise variances.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%% Plots %%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

subplot(4,1,1);

stem(t,x,'fill','b-.'); grid on;

title('Transmit signal: x');

subplot(4,1,2);

stem(t,y,'fill','b-.'); grid on;

title('noisy receive signal: y');

subplot(4,1,3);

stem(tz,z,'fill','b-.'); grid on;

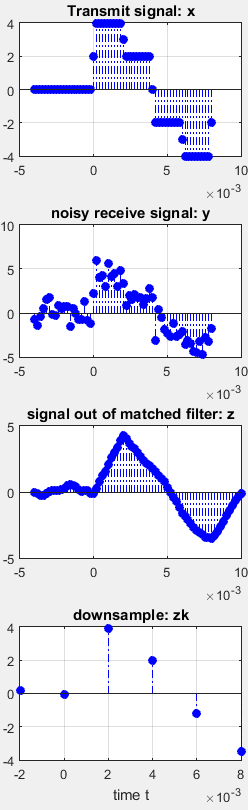
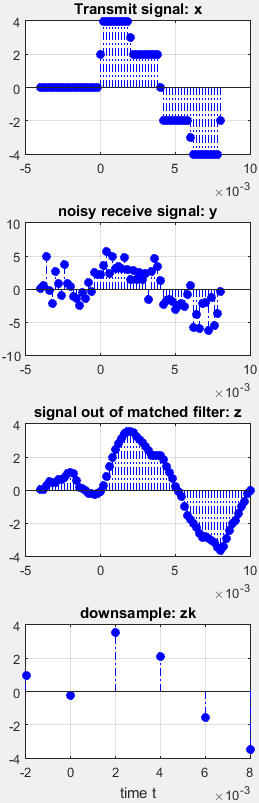
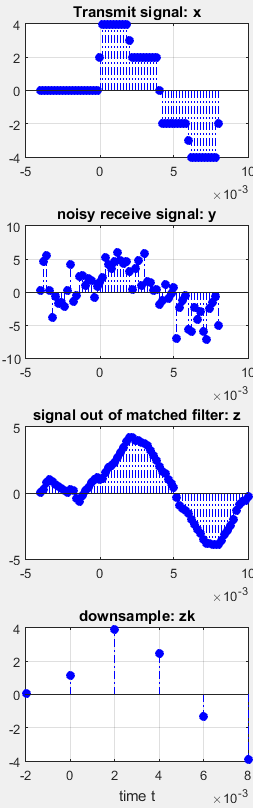
title('signal out of matched filter: z');

subplot(4,1,4);

stem(t\_down\_sampled,zk,'fill','b-.'); grid on;

title('downsample: zk'); xlabel('time t');

**Figure.25 updated ‘Plots’ part of previous MATLAB script in Figure.23**

**(a) Varnoise=1; (b) Varnoise =2; (c) Varnoise =4;**

**Figure.26 the signals at different stages with different noise variances for noisy channel**

From the figure showing above, it is clear to see it is easier to estimate the transmitted modulation symbol from the output signal of matched filter, rather than the noisy received signal.

**Why using Matched filter:**

Assume an optimum receiver minimizes the bit error rate, and the transfer function of it is. Also, assume the received signal after passing through the channel is:

where, is the message signal, and is the AWGN (Addictive White Gaussian Noise), which has power spectrum density (PSD) equal to .

Expressed the received message signal by inverse Fourier transform:

The received average noise power:

Hence, the ratio of received signal power to average noise power, which is SNR can be calculated:

While message sigal and the noise are certain, the value of SNR is determined by the filter.

Appling the ***Schwarz Inequation***, which is

when and only when, two sides are equal.

So, we can derived that:

Up to now, we have:

where E is the energy of message signal. When and only when, two sides are equal (where k is a constant, we can simply take it as 1). In time domain, we have.

To conclude, the matched filter can give best signal to noise ratio, and it is easier to estimate the transmitted modulation symbol from the output of matched filter, rather than the noisy received signal.

**Problem 12 decision block**

The basic ideal of decision block is find which value in [4 2 -2 -4] has minimum distance to the downsampled signal, and then output this value. The MATLAB function of decision block is showing below:

function [ z\_decided ] = decision( z\_sampled )

z\_mid=[1;1;1;1]\*z\_sampled;

z\_standard=[4;2;-2;-4]\*ones(1,length(z\_sampled));

z\_mid2=abs(z\_mid-z\_standard);

[M,N]=min(z\_mid2);

z\_decided=z\_standard(N);

end

**Figure.27 MATLAB function of decision block**

Assume a sequence of downsampled signal is [4.1 3.7 1.8 1.6 -2.1 -1.6 -3.8 -4.4], the algorithm of this function is demonstrated for this example in following table:

|  |  |
| --- | --- |
| **Items** | **Representation in Matrix** |
| **z\_sampled**  () |  |
| **z\_mid**  (repeat 4 times in column) |  |
| **z\_standard**  (four standard symbols in each column) |  |
| **z\_mid2**  (calculate the distance) |  |
| **N**  (position of minimum distance) |  |
| z\_decided  (output) |  |

**Problem 13**

In this stage, the signal in symbol form need to be transformed to corresponding bit form. The algorithm of demapper is showing in following table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Symbol form:** | **Sign of symbol:** | **First bit:** | **Second bit:** | **Bit form** |
| 4 | 1 | 0 | 0 | 0 0 |
| 2 | 1 | 0 | 1 | 0 1 |
| -2 | -1 | 1 | 1 | 1 1 |
| -4 | -1 | 1 | 0 | 1 0 |

(Notice: ‘~’ is the ‘Not gate’ in logic gate)

**Table.3 4ASK Demapping**

MATLAB function of this Demapper is listed as Figure.25.

function [ message ] = Demapper( z\_decided )

message\_in\_bits=zeros(length(z\_decided),2); % initialize the matrix [k,2]

message\_in\_bits(:,1)=~(sign(z\_decided)+1); % get the first bit

message\_in\_bits(:,2)=~(z\_decided-2\*sign(z\_decided)); % get the second bit

message=reshape(message\_in\_bits',1,2\*length(z\_decided),[]); % reshape to [1,2k] matrix

end

**Figure.28 MATLAB function of Demapper**

**Problem 14 entire system**

Up to now, the MATLAB script can be updated with transmitter, channel and receiver. The transmitter include mapper, and sampler; receiver include matched filter, downsampler, decision block and the demapper. The channel is AWGN noisy channel.

clc;clear;close all;

N=10;

source=randi([0 1],1,N ); %source code

%%%%%%%%%%%%%%%%% Transmitter %%%%%%%%

%%%% mapper %%%%

a=ASKmapping( source );

%%%% sampler %%%%

k=1:length(a);

Ts=0.002;To=Ts/10;

t=-2\*Ts:To:length(a)\*Ts;

p=@(t,k)rect((t-(2\*k-1)\*Ts/2)/Ts);

x=a'\*bsxfun(p,t,k');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%% Channel %%%%%%%%%%%%

varinoise=1;

y=x+sqrt(varinoise)\*randn(size(x));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%% Receiver %%%%%%%%%%%

%%%% matched filter %%%%

tp=0:To:Ts;

h=p(tp,1);

z=To/Ts\*conv(y,h);

tz=To\*(1:length(z))-To-2\*Ts;

%%%% downsampler %%%%

zk=downsample(z,Ts,To);

t\_down\_sampled=-Ts:Ts:Ts\*length(a);

%%%% decision block %%%%

z\_decided=decision(zk(3:end));

%%%% demapper %%%%

source\_output=Demapper(z\_decided);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ Output ] =ASKmapping( Input )

b=reshape(Input,2,[])';

Output= (-(2\*b(:,1)-1)).\*abs((2\*(2\*b(:,2)-1)-(~b(:,2)\*2)));

end

function [ z\_sampled ] = downsample( z,Ts,To )

% z\_sampled=zeros(1,length(z)-Ts/To);

z\_sampled=z(Ts/To+1:Ts/To:end-Ts/To);

end

function [ Output ] =ASKmapping( Input )

b=reshape(Input,2,[])';

Output= (-(2\*b(:,1)-1)).\*abs((2\*(2\*b(:,2)-1)-(~b(:,2)\*2)));

end

function [ z\_decided ] = decision( z\_sampled )

z\_mid=[1;1;1;1]\*z\_sampled;

z\_standard=[4;2;-2;-4]\*ones(1,length(z\_sampled));

z\_mid2=abs(z\_mid-z\_standard);

[M,N]=min(z\_mid2);

z\_decided=z\_standard(N);

end

function [ Output ] =ASKmapping( Input )

b=reshape(Input,2,[])';

Output= (-(2\*b(:,1)-1)).\*abs((2\*(2\*b(:,2)-1)-(~b(:,2)\*2)));

end

function [ message ] = Demapper( z\_decided )

message\_in\_bits=zeros(length(z\_decided),2);

message\_in\_bits(:,1)=~(sign(z\_decided)+1);

message\_in\_bits(:,2)=~(z\_decided-2\*sign(z\_decided));

message=reshape(message\_in\_bits',1,2\*length(z\_decided),[]);

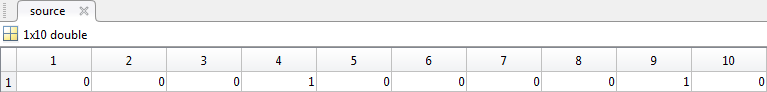
end

**(a) (b)**

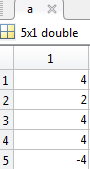
**Figure.28 MATLAB script for entire system: (a) with comment; (b) function scripts**

**Test the system:**

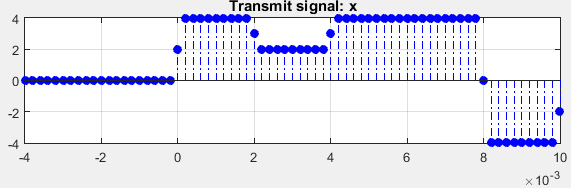
* The input random source code is:



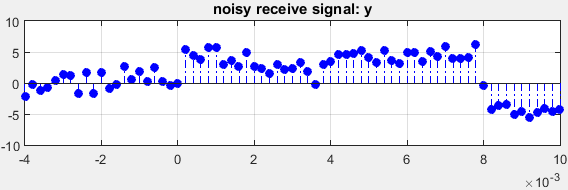
* Mapper output:



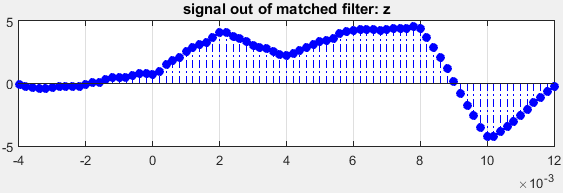
* Sampler output:



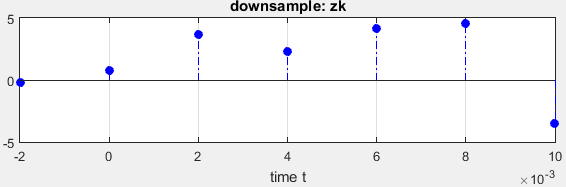
* Channel output (varnoise=1):



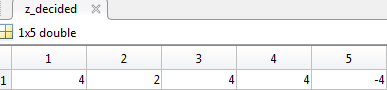
* Matched filter output:



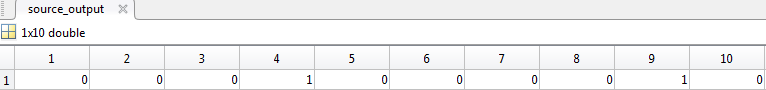
* Downsampler output:



* Decision block output:



* Demapper output:



In this condition of noisy channel (varinoise=1), the estimated data bits is same as the input sequence of data bits.

**Problem 15 response to different noise levels**

Using three more MATLAB scripts to calculate the number of errors, bit error rate (BER) and the signal to noise ratio (SNR). The test stream has 1000 bits, which means there are 500 symbols in it.

No\_error=sum(xor(source,source\_output));

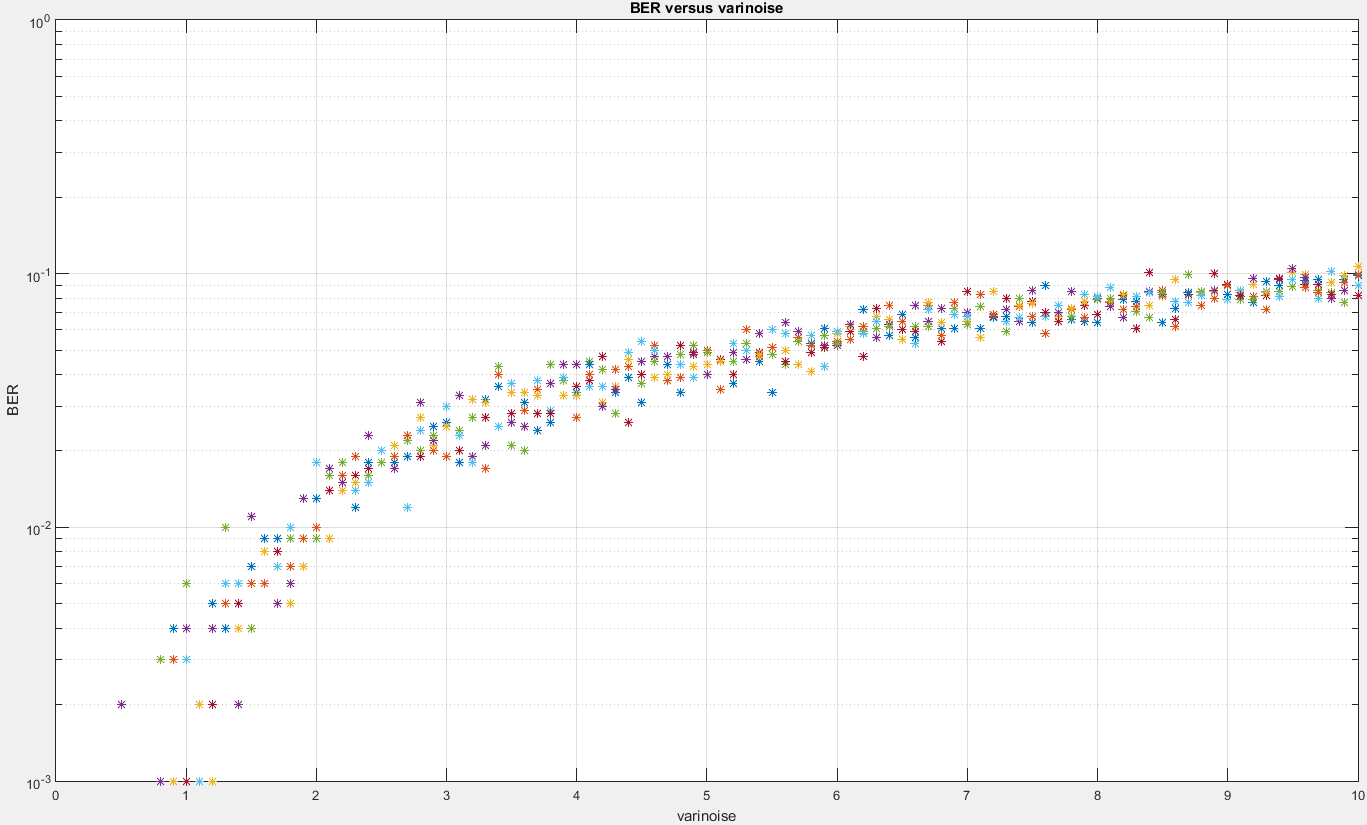
BER=No\_error/length(source);

SNR=snr(source,source-source\_output); %in dB

**Figure.29 MATLAB script to calculate number of errors and the BER**

**Plot the BER versus ‘varinoise’ (variance of noise):**

The noise variance is varying from 0.1 to 10, with 0.1 increment for each time. Taking 5 BERs for each noise level.

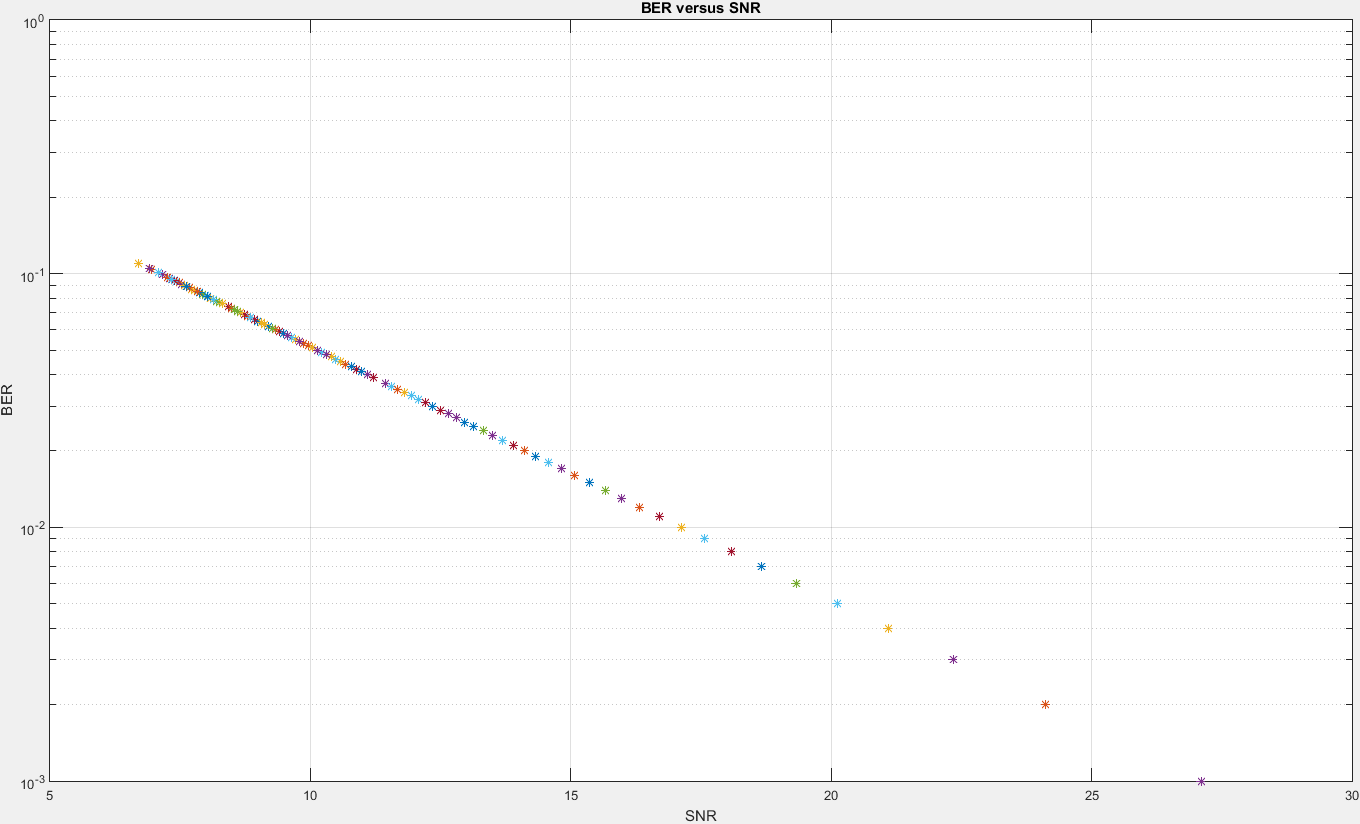


**Figure.30 the BER curve versus various noise variance (0.1~10)**

In Figure.30, at low noise, BER is also very small; in higher noise, BER is relevantly high. BER is approaching to 10%, when variance of noise close to 10.

**Plot the BER versus SNR (dB):**

When signal to noise ratio is low, the bit error rate is relevantly high; when signal noise ratio is high, bit error rate is low. The noise variance is also varying from 0.1 to 10, with 0.1 increment for each time. Taking 5 BERs for each noise level.



**Figure.31 the BER curve versus signal to noise ratio (SNR in dB)**

clc;clear;close all;

N=1000; % number of bit in source code

source=randi([0 1],1,N ); % source code

for vn=0.1:0.1:10;

for i=1:5

%%%%%%%%%%%%%%%%% Transmitter %%%%%%%%%%%%%

%%%% mapper %%%%

a=ASKmapping( source );

%%%% sampler %%%%

k=1:length(a);

Ts=0.002;To=Ts/10;

t=-2\*Ts:To:length(a)\*Ts;

p=@(t,k)rect((t-(2\*k-1)\*Ts/2)/Ts);

x=a'\*bsxfun(p,t,k');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%% Channel %%%%%%%%%%%%%%%%%

varinoise=vn;

y=x+sqrt(varinoise)\*randn(size(x)); % +AWGN

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%% Receiver %%%%%%%%%%%%%%%%

%%%% matched filter %%%%

tp=0:To:Ts;

h=p(tp,1);

z=To/Ts\*conv(y,h);

tz=To\*(1:length(z))-To-2\*Ts;

%%%% downsampler %%%%

zk=downsample(z,Ts,To);

t\_down\_sampled=-Ts:Ts:Ts\*length(a);

%%%% decision block %%%%

z\_decided=decision(zk(3:end));

%%%% demapper %%%%

source\_output=Demapper(z\_decided);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

No\_error=sum(xor(source,source\_output));

BER=No\_error/length(source);

SNR=snr(source,source-source\_output); %in dB

semilogy(vn,BER,'\*');hold on;grid on;

end

end

xlabel('varinoise');ylabel('BER');title('BER versus varinoise');

**Figure.32 MATLAB script for plot Figure.30 (BER VS. ‘varinoise’)**

clc;clear;close all;

N=1000; % number of bit in source code

source=randi([0 1],1,N ); % source code

for vn=0.1:0.1:10;

for i=1:5

%%%%%%%%%%%%%%%%% Transmitter %%%%%%%%%%%%%

%%%% mapper %%%%

a=ASKmapping( source );

%%%% sampler %%%%

k=1:length(a);

Ts=0.002;To=Ts/10;

t=-2\*Ts:To:length(a)\*Ts;

p=@(t,k)rect((t-(2\*k-1)\*Ts/2)/Ts);

x=a'\*bsxfun(p,t,k');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%% Channel %%%%%%%%%%%%%%%%%

varinoise=vn;

y=x+sqrt(varinoise)\*randn(size(x)); % +AWGN

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%% Receiver %%%%%%%%%%%%%%%%

%%%% matched filter %%%%

tp=0:To:Ts;

h=p(tp,1);

z=To/Ts\*conv(y,h);

tz=To\*(1:length(z))-To-2\*Ts;

%%%% downsampler %%%%

zk=downsample(z,Ts,To);

t\_down\_sampled=-Ts:Ts:Ts\*length(a);

%%%% decision block %%%%

z\_decided=decision(zk(3:end));

%%%% demapper %%%%

source\_output=Demapper(z\_decided);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

No\_error=sum(xor(source,source\_output));

BER=No\_error/length(source);

SNR=snr(source,source-source\_output); %in dB

semilogy(SNR,BER,'\*');hold on;grid on;

end

end

xlabel('SNR');ylabel('BER');title('BER versus SNR');

**Figure.33 MATLAB script for plot Figure.31 (BER VS. SNR)**