**EEE411 Advanced Signal Processing**

**Assignment II: Practical Report Digital Transmission System with Matched Filte**

XUN XU

1926930

2019/11/23

# Problem1

## 1.1

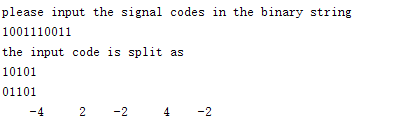
Mapping table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
|  | +4 | +2 | -2 | -4 |

Table1: mapping from bit pairs to modulation symbols

When we enter a binary string, we need to identify the two-digit binary string each time, map it to the corresponding result, and store it in a new array.

Outcome:



It can be seen that the binary string is successfully identified as the ask modulation symbols.

Code of mapping:

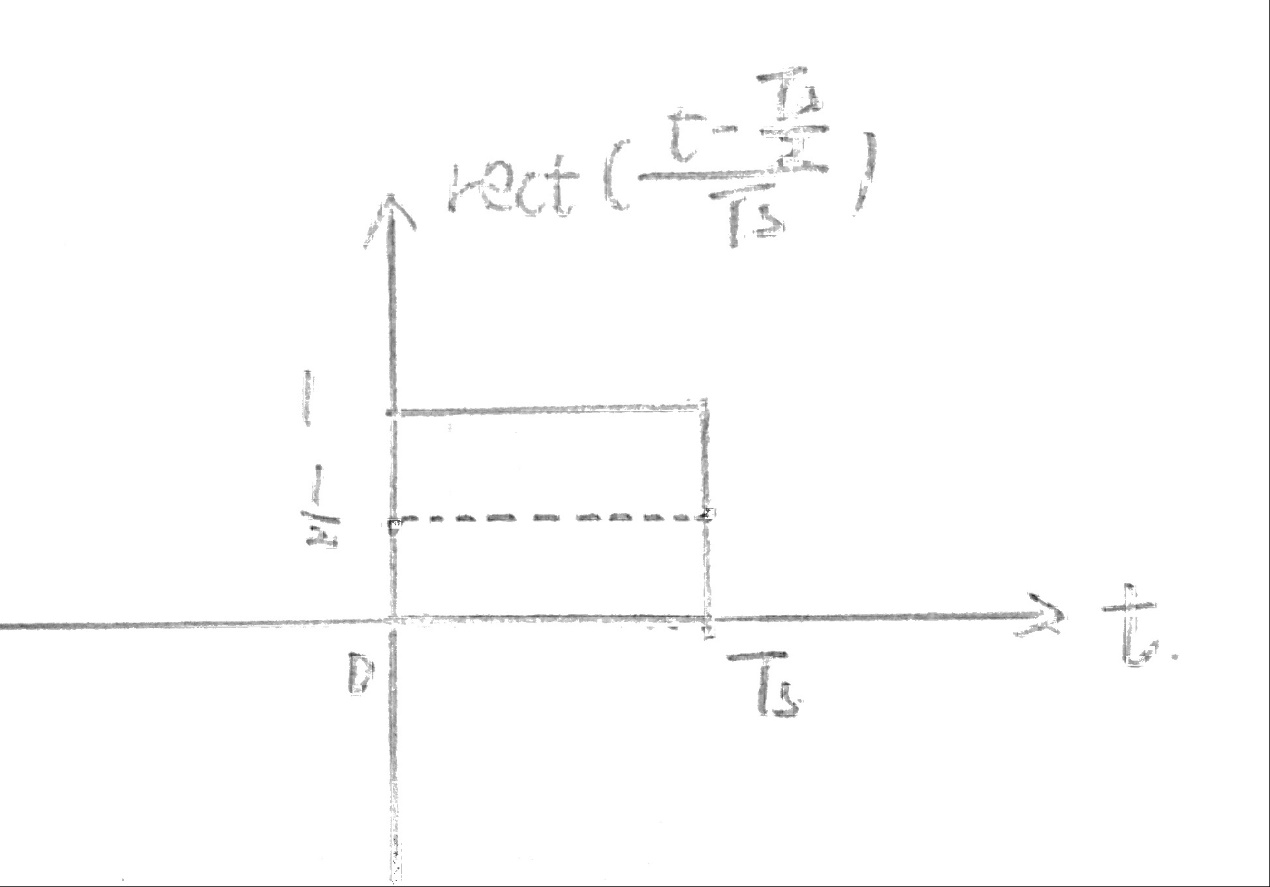
function q= mapping(array)  
 len=2\*length(array);  
 q=zeros(len);  
 for i=1:2:len  
 switch strcat(array(i),array(i+1))  
 case '00'  
 q(i)=4;  
 case '01'  
 q(i)=2;  
 case '11'  
 q(i)=-2;  
 case '10'  
 q(i)=-4;  
 end  
 end  
end

Code of problem1:

clc  
py=input('please input the signal codes in the binary string\n','s');  
while rem(length(py),2)  
 disp('your input is worng');  
 py=input('please reinput the signal codes\n','s');  
end  
array=reshape(py,[],length(py)/2);%reshape the input signal for identify  
disp('the input code is split as');disp(array);  
outcome=mapping(array);  
ask=outcome(:,1)';  
ask=ask(1:2:length(py));  
disp(ask);

# Problem2:

Outcome:



Figure：sketch of the transmit pulse

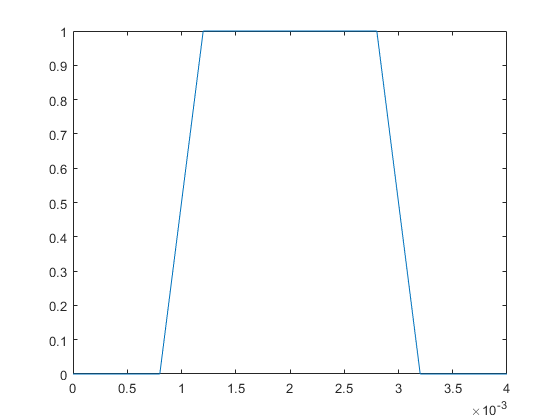
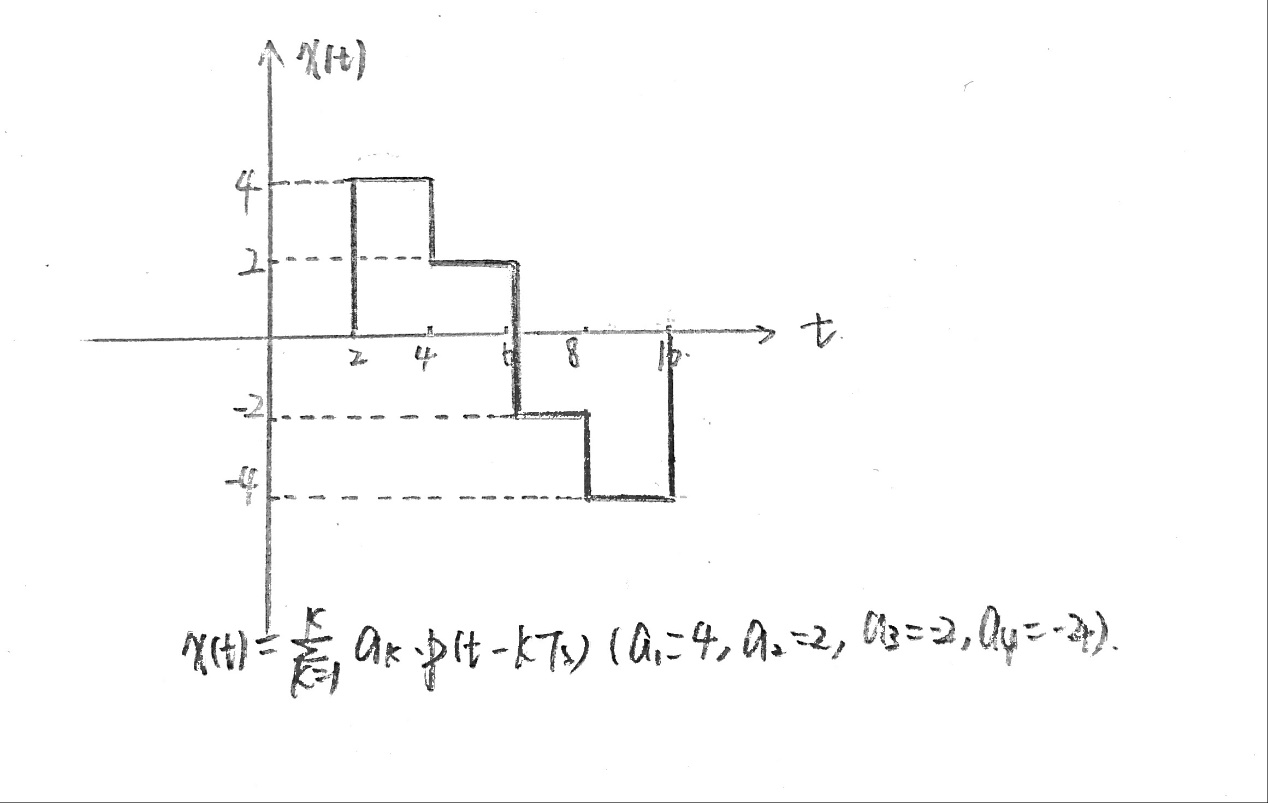


Figure1: transmit pulse by MATLAB

Code:

ts=0.002;  
t=0:0.0002:0.004;  
p=rect((t-ts)/ts);  
figure;  
plot(t,p)

# Problem3



Figure：Sketch by hand the transmit signal for a sequence a1,a2,a3,a4 of four chosen modulation symbols

# Problem4:

Outcome:

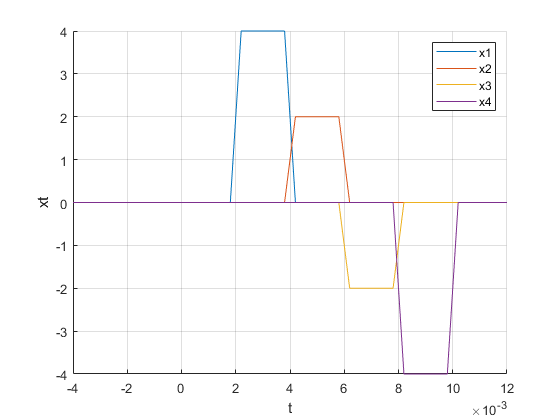


Figure: transmit signal of x1,x2,x3,x4 in the same figure

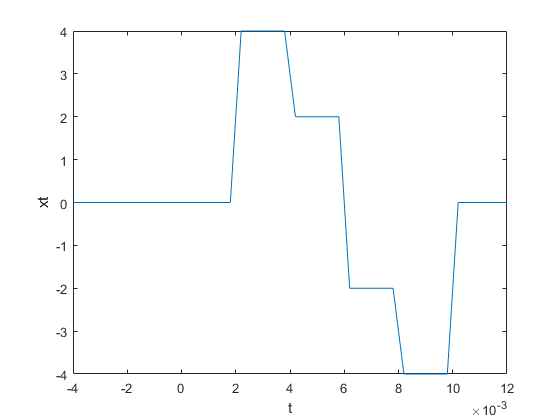


Figure: transmit signal of xt=x1+x2+x3+x4

Code:

clc;  
clear;  
Ts=0.002;  
T0=Ts/10;  
t=-2\*Ts:T0:6\*Ts;  
p=@(t)rect((t-Ts/2)/Ts);  
x1=4\*p(t-1\*Ts);  
x2=2\*p(t-2\*Ts);  
x3=-2\*p(t-3\*Ts);  
x4=-4\*p(t-4\*Ts);  
figure;  
hold on;  
grid on;  
plot(t,x1,t,x2,t,x3,t,x4);  
xlabel('t');  
ylabel('xt');  
legend('x1','x2','x3','x4');  
x=x1+x2+x3+x4;  
figure;  
plot(t,x);  
xlabel('t');  
ylabel('xt');

# Problem5:

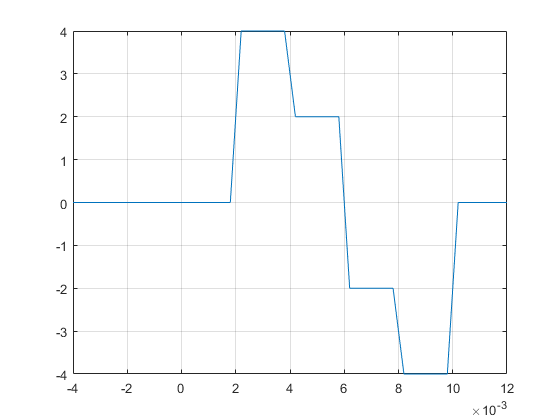


Figure: transmit signal of xt (modulator)

Comparing it to the results of the previous two questions, the modulator signal is the superposition of a single signal in the time domain. The results are basically consistent with the previous results.

Code:

clc  
py=input('please input the signal codes in the binary string\n','s');  
while rem(length(py),2)  
 disp('your input is worng');  
 py=input('please reinput the signal codes\n','s');  
end  
array=reshape(py,[],length(py)/2);%reshape the input signal for identify  
disp('the input code is split as');disp(array);  
outcome=mapping(array);  
ask=outcome(:,1)';  
ask=ask(1:2:length(py));  
disp(ask);  
[t,modulation]=modulator(ask);  
plot(t,modulation);  
grid on;

code of modulator:

function [t,transmit\_signal]=modulator(symbols)  
  
Ts=0.002;  
T0=Ts/10;  
t = -2\*Ts:T0:6\*Ts;  
n = size(symbols ,2);  
p = @(t)rect((t-Ts/2)/Ts);  
transmit\_signal=0;  
for i=1:n  
 transmit\_signal=transmit\_signal+symbols(1,i)\*p(t-i\*Ts);  
end  
end

# Problem6:

Outcome:

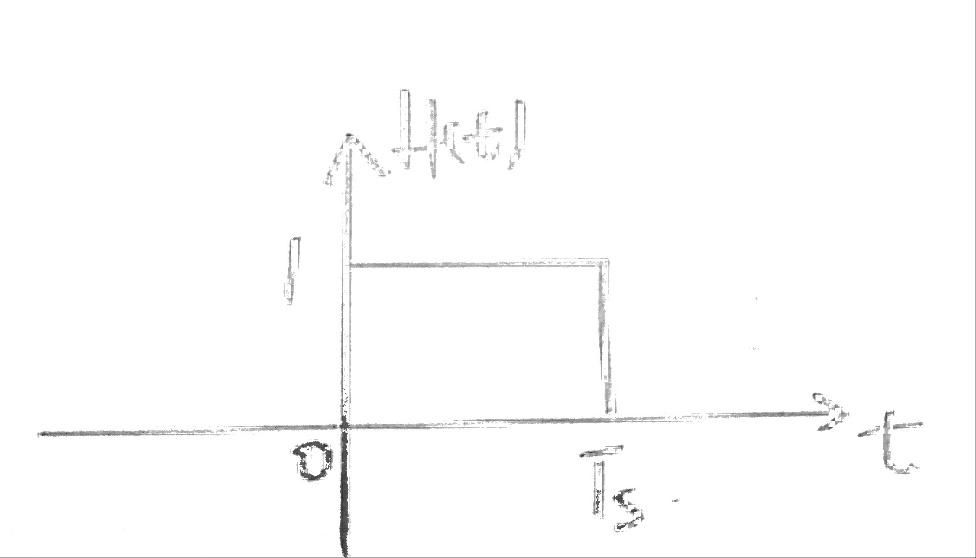
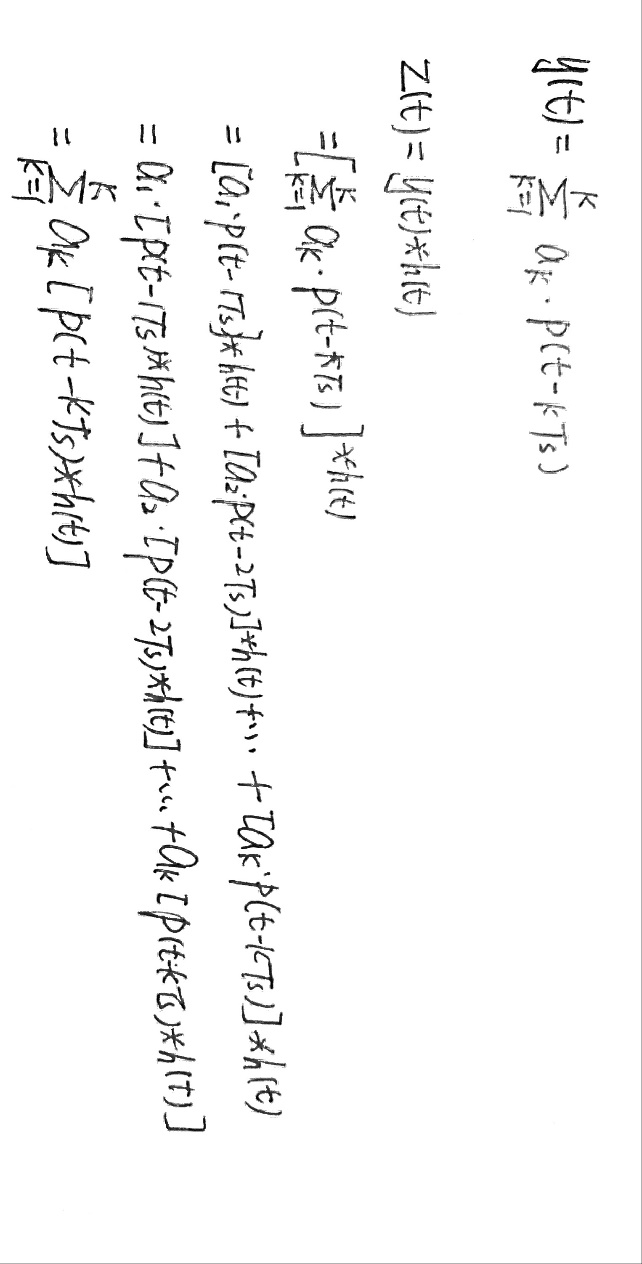


Figure: Sketch of the impulse response of the receive ﬁlter

# Problem7:

The derivation of this formula is as follows:



Expanding the formula and plugging it in we can get the result. We use the distribution rate and the binding law here.

# Problem8:

Outcome:

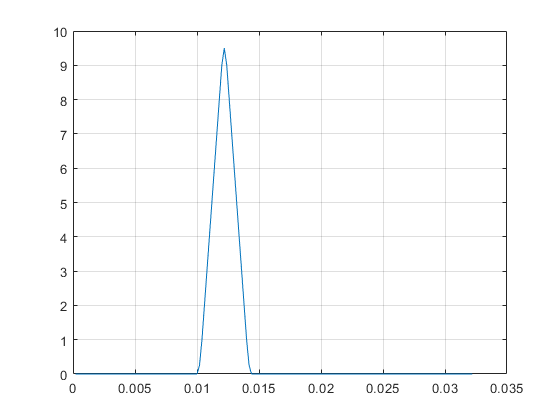


Figure: output of receive filter

The impulse response h (t) is convolved with the signal x, and the result y is the result of x passing through the matched filter. So the result is as expected

Code:

clc,clear;  
Ts=0.002;  
T0=Ts/10;  
t=-2\*Ts:T0:6\*Ts;  
p=@(t)rect((t-Ts/2)/Ts);  
  
k=1;  
y=p(t-k\*Ts);  
h=p(Ts-t);  
  
t1=T0\*(1:length(conv(y,h)));  
g=conv(y,h);  
plot(t1 ,g);  
grid on

# Problem9:

## 9.1 transmit signal x1(t) and x2(t)

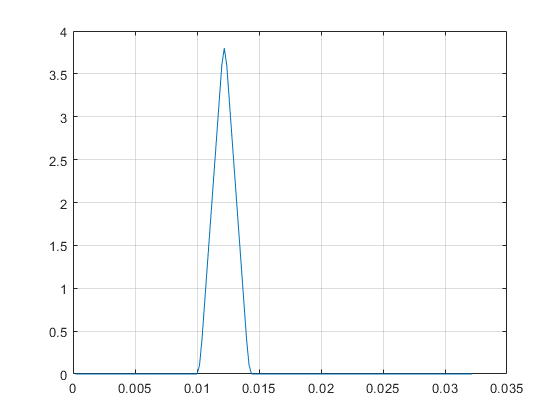


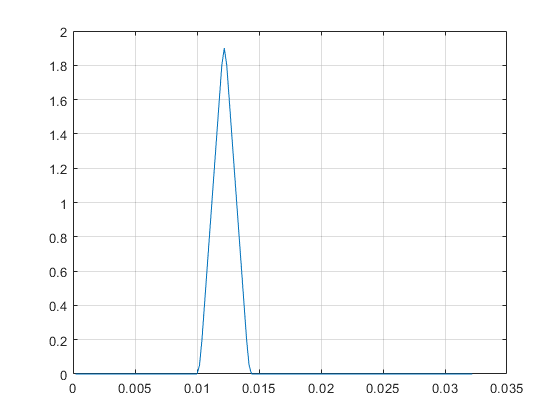
Figure: transmit signal x1(t) 

Figure: transmit signal x2(t)

The result is as expected.

## 9.2

Outcome:

图片包含 文字

描述已自动生成

Figure: transmit signal x(t)

Code:

Ts=0.002;  
T0=Ts/10;  
t=-2\*Ts:T0:6\*Ts;  
p=@(t)rect((t-Ts/2)/Ts);  
h=p(t);  
  
y1=4\*p(t-1\*Ts);  
z1=T0/Ts\* conv(y1,h);  
tz1=T0\*(1:(length(z1)));  
figure;plot(tz1,z1);  
grid on;  
  
y2=2\*p(t-1\*Ts);  
z2=T0/Ts\*conv(y2,h);  
tz2=T0\*(1:(length(z2)));  
figure;plot(tz2,z2);  
grid on;  
  
py=input('please input the signal codes in the binary string\n','s');  
while rem(length(py),2)  
 disp('your input is worng');  
 py=input('please reinput the signal codes\n','s');  
end  
array=reshape(py,[],length(py)/2);%reshape the input signal for identify  
disp('the input code is split as');disp(array);  
outcome=mapping(array);  
ask=outcome(:,1)';  
ask=ask(1:2:length(py));  
disp(ask);  
  
[t,y]=modulator(ask);  
  
z=T0/Ts\*conv(y,h);  
tz=T0\*(1:(length(z)));  
figure;plot(tz,z);  
grid on;

# Problem10:

Outcome:

图片包含 文字

描述已自动生成 Figure: Transmission signal before sampling

For the input string 10011100, the sampled values are as follows:



It can be seen that the result after sampling is basically close to the value of the original ask signal.

Code:

function output= sample(signal)  
% Ts=0.002;  
N=10;  
len=length(signal);  
output=[];  
for t=61:10:(len-1)  
 output=cat(2, output, signal(t));   
end  
end

# Problem11:

Do experiments with various noise variance, and discuss the eﬀect of noise on y(t),z(t) and zk. Is it easier to estimate the transmitted modulation symbol ak from y(t) or from z(t)?

Z(t)

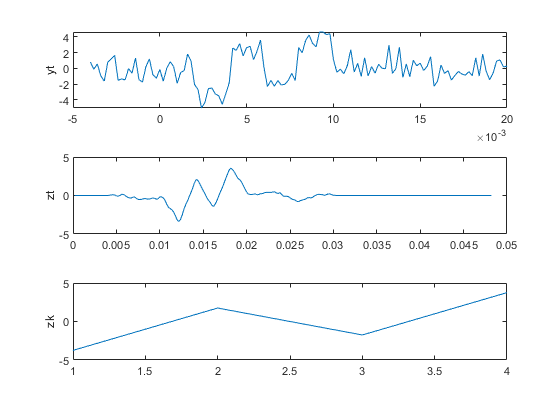


Figure: output signal of y(t),z(t) and z(k) (noise variance=1)

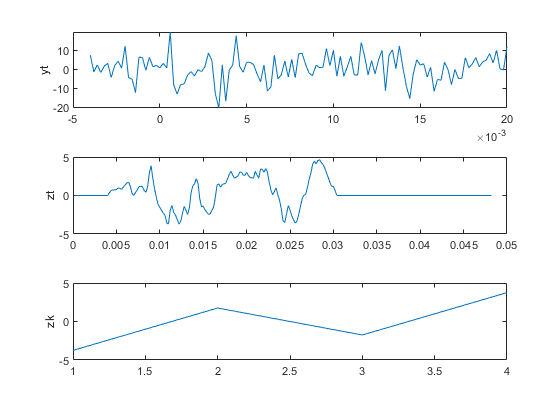


Figure: output signal of y(t),z(t) and z(k) (noise variance=50)

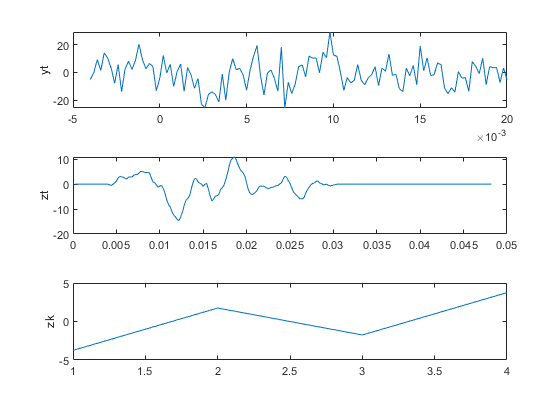


Figure: output signal of y(t),z(t) and z(k) (noise variance=99)

By comparing these pictures, we can find that under the influence of different noise variances, it can be seen that as the noise variance continues to increase( the degree of noise impact increases), y(t) and z(t) are greatly affected. And the signal graph is becoming increasingly unrecognizable but the results of z(k) are basically the same.

Comparing y(t) and z(t), we see that the waveform of y(t) is more unrecognizable and the waveform of z(t) is easier to estimate. Is it easier to estimate the transmitted modulation symbol ak from z(t).

Code:

clc  
Ts=0.002;  
T0=Ts/10;  
t=-2\*Ts:Ts/10:8\*Ts;  
p=@(t)rect((t-Ts/2)/Ts);  
h=p(t);  
%Generate modulation signal xt  
py=input('please input the signal codes in the binary string\n','s');  
while rem(length(py),2)  
 disp('your input is worng');  
 py=input('please reinput the signal codes\n','s');  
end  
array=reshape(py,[],length(py)/2);%reshape the input signal for identify  
disp('the input code is split as');disp(array);  
outcome=mapping(array);  
ask=outcome(:,1)';  
ask=ask(1:2:length(py));  
disp(ask);  
[t,x]=modulator(ask);  
%Increase noise interference output  
for varnoise=1:49:99  
y=x+sqrt(varnoise)\*randn(size(x));  
figure;  
subplot(311);  
plot(t,y);  
ylabel('yt');  
  
p=@(t)rect((t-Ts/2)/Ts);  
h=p(Ts-t);  
z=T0/Ts\*conv(y,h);  
tz=T0\*(1:(length(z)));  
subplot(312);  
plot(tz,z);  
ylabel('zt');  
  
zk=sample(z);  
subplot(313);  
disp(zk);  
plot(zk);  
ylabel('zk');  
end

# Problem12:

Compare the input value with +2 +4 -4 -2 to determine which one is closest to the input value. The results after running are as follows:

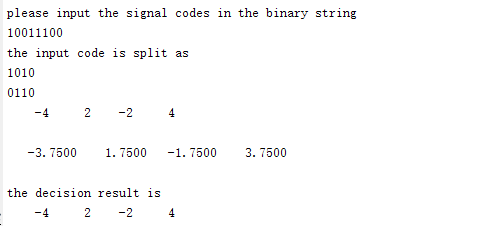


Figure: results of decision

Code:

function ak= decision(zk)  
len=length(zk);  
ak=[];  
for i=1:1:len  
 distance\_1=abs(4-zk(i));  
 distance\_2=abs(2-zk(i));  
 distance\_3=abs(-4-zk(i));  
 distance\_4=abs(-2-zk(i));  
 distance=[distance\_1,distance\_2,distance\_3,distance\_4];  
 value=[4,2,-4,-2];  
 [~,n]=min(distance);  
 ak(i)=value(n);  
end  
end

# Problem13:

Compare the data after decision with +4 +2 -2 -4 and then store the binary strings corresponding to these values in a new array and display. The results after demapping are as follows:

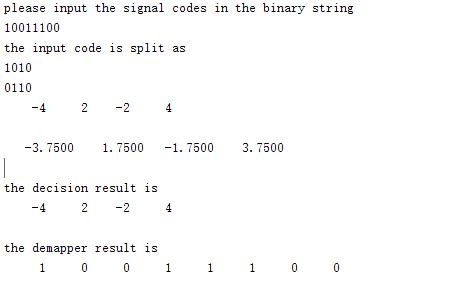


Figure: results of demapper

Code:

function output =demapper(input)  
L=length(input);  
for i=1:L  
 if input(i)==4  
 temp(:,i)=[0,0];  
 elseif input(i)==2  
 temp(:,i)=[0,1];  
 elseif input(i)==-2  
 temp(:,i)=[1,1];  
 elseif input(i)==-4  
 temp(:,i)=[1,0];  
 end  
end  
k=1;  
for i=1:L  
output(k)=temp(1,i);  
output(k+1)=temp(2,i);  
k=k+2;  
end  
end

# Problem14:

This question is a synthesis of the previous question. We input a two-bit combination binary string of any length, which is mapped to the ask symbol under the action of the mapper. The ask symbol then passes through the modulator to generate the signal x(t), then Gaussian white noise is added to the channel of x(t) and x(t) is converted to y (t) after passing through the channel. Then y(t) is converted to the impulse response z(t) after the matched filter, and z(t) is then passed to the sampler, decision, and demapper(three signal processing units). Thus we can restore the original signal binary string.

Outcome:

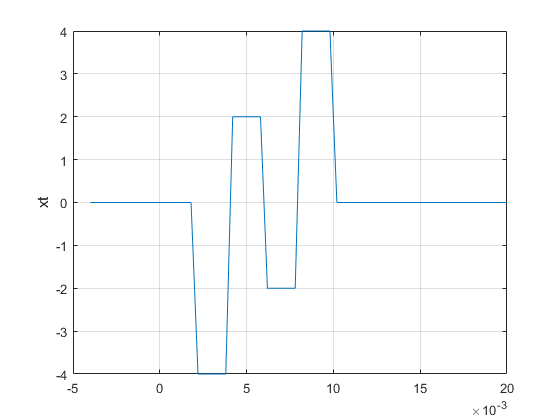


Figure: x(t) signal waveform

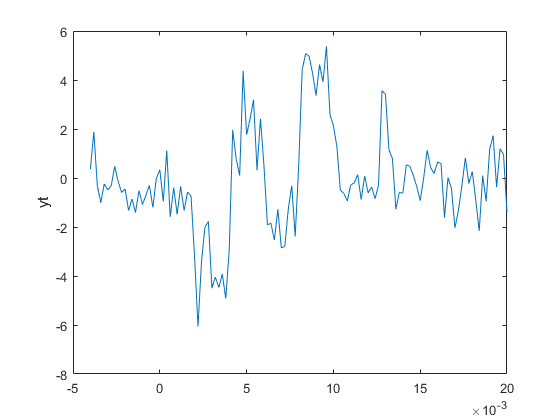


Figure: y(t) signal waveform

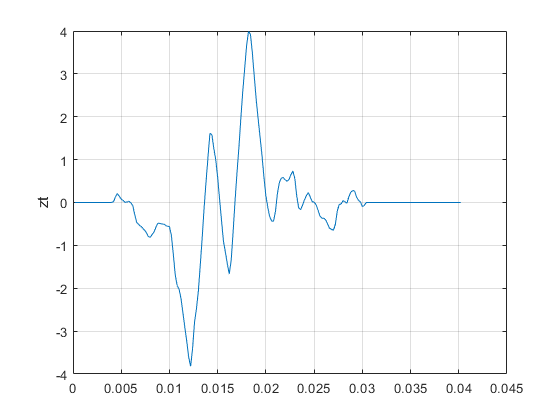


Figure: z(t) signal waveform

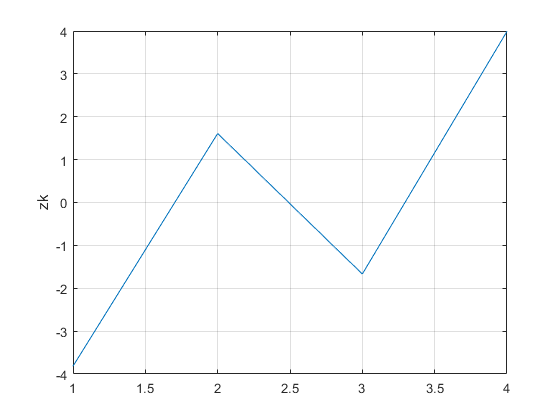


Figure: z(k) signal waveform

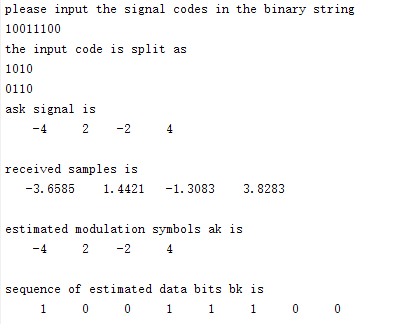


Figure: signal modulation and recovery process

We can see that the recovered signal is consistent with the original signal.

Code:

clc;  
Ts=0.002;  
T0=Ts/10;  
t=-2\*Ts:Ts/10:6\*Ts;  
p=@(t)rect((t-Ts/2)/Ts);  
h=p(t);

## Transmitter: bk to ask

py=input('please input the signal codes in the binary string\n','s');  
while rem(length(py),2)  
 disp('your input is worng');  
 py=input('please reinput the signal codes\n','s');  
end  
array=reshape(py,[],length(py)/2);%reshape the input signal for identify  
disp('the input code is split as');disp(array);  
outcome=mapping(array);  
ask=outcome(:,1)';  
ask=ask(1:2:length(py));  
disp('ask signal is');disp(ask);

## Modulator: ask to x(t)

[t,xt]=modulator(ask);  
figure;plot(t,xt);  
grid on;  
ylabel('xt');

## Channel noise: x(t) to y(t)

varnoise = 1;  
yt=xt+sqrt(varnoise)\*randn(size(xt));  
figure;plot(t,yt);  
ylabel('yt');

## Receiver filter: y(t) to z(t)

zt=T0/Ts\*conv(yt,h);  
tz=T0\*(1:(length(zt)));  
figure;plot(tz,zt);  
grid on;  
ylabel('zt');

## Sampler: z(t) to z(k)

zk=sample(zt);  
disp('received samples is');disp(zk);  
figure;plot(zk);  
grid on;ylabel('zk');

## Decision: z(k) to estimated modulation symbols ak

ak=decision(zk);  
disp('estimated modulation symbols is');disp(ak);  
figure;plot(ak);  
grid on;ylabel('ak');

## Demapper: ak to estimated data bits bk

bk=demapper(ak);  
disp('sequence of estimated data bits is');disp(bk);  
figure;plot(bk);  
grid on;ylabel('bk');

# Problem15:

When we test script for very low noise ,for example , varnoise=1, the bit error rate is very low, and the result is about 1.25%. When we test script for very high noise ,for example , varnoise=100, the bit error rate is 50%.

Code:

clc;  
ask=[-4,2,-2,4];  
py=[1 0 0 1 1 1 0 0];  
error\_rate\_all=0;  
py\_num=[1,0,0,1,1,1,0,0];  
Ts=0.002;  
T0=Ts/10;  
t=-2\*Ts:Ts/10:6\*Ts;  
p=@(t)rect((t-Ts/2)/Ts);  
h=p(t);  
varnoise = 100;  
  
for i=1:1:1000  
[t,xt]=modulator(ask);  
yt=xt+sqrt(varnoise)\*randn(size(xt));  
zt=T0/Ts\*conv(yt,h);  
tz=T0\*(1:(length(zt)));  
zk=sample(zt);  
ak=decision(zk);  
bk=demapper(ak);  
bk=bk(1:length(py));  
error\_rate=1-length(find(bk==py\_num))/length(py\_num);  
error\_rate\_all=error\_rate+error\_rate\_all;  
average=error\_rate\_all/i;  
disp(average);  
end