EEE411 LAB2 ADVANCED SIGNAL PROCESSING

Author: Zhipeng Ye

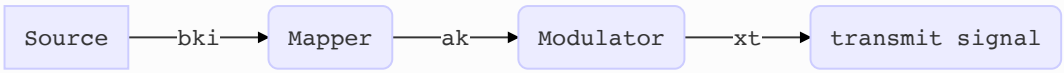
Student ID: 1926908

Faculty: MSc Multimedia Telecommunications

Data: 21th November 2019

Digital modulator

The figure shows a block diagram of a digital transmitter.



Problem 1 (6 points)

Consider $4ASK$ with the modulation alphabet $A = -4, -2, +2, +4$, and a mapping from bit pairs $b_{k1}\,b_{k2}$ to modulation symbols ak, as given in the following table (the index k indicates the time):

$b_{k1}\,b_{k2}$	a_k
00	$+4$
01	$+2$
11	-2
10	-4

Fill in the gaps. (There is no unique solution.)

Write a Matlab function that implements this mapping with a bit pair as the input and the corresponding input and the corresponding modulation symbol as the output.

Code

```
function symbol = modulate(bit_pair)

bit_pair_str = num2str(bit_pair)

length_value = length(bit_pair_str)

if length_value ~= 2
    error('decode error')
end

% hashmap key-value structure
alphabet = containers.Map({'00', '01', '11','10'},...
    {'4', '2', '-2','-4'});

symbol = alphabet(bit_pair)

end
```

Result

```
decode_value = modulate('01')

decode_value =

    '2'
```

Analysis

As we can see from result, I can use this function to implement mapping. However I use hashmap which is called dictionary in python to code, because this kind of data structure is very powerful and can be seen in most program languages. It is a key-value structure and use hash function for mapping fastly. By the way, we can use $if-else$ to implement this function too but it is a not beautiful method.

Next, write a Matlab function that maps a sequence of data bits to the LAB function that maps a sequence of modulation symbols.

Code

```
function origianl_code = decode_binary(encode_squence)

length_value = length(encode_squence);
origianl_code = "";

for i = 1:2:length_value
    str_sequency = [encode_squence(i),encode_squence(i+1)];
    origianl_code((i+1)/2) = modulate(str_sequency);
end

origianl_code = join(origianl_code,',')

end
```

```
str = '00011110';

decode_value = decode_binary(str)
```

Result

```
decode_value =

    "4,2,-2,-4"
```

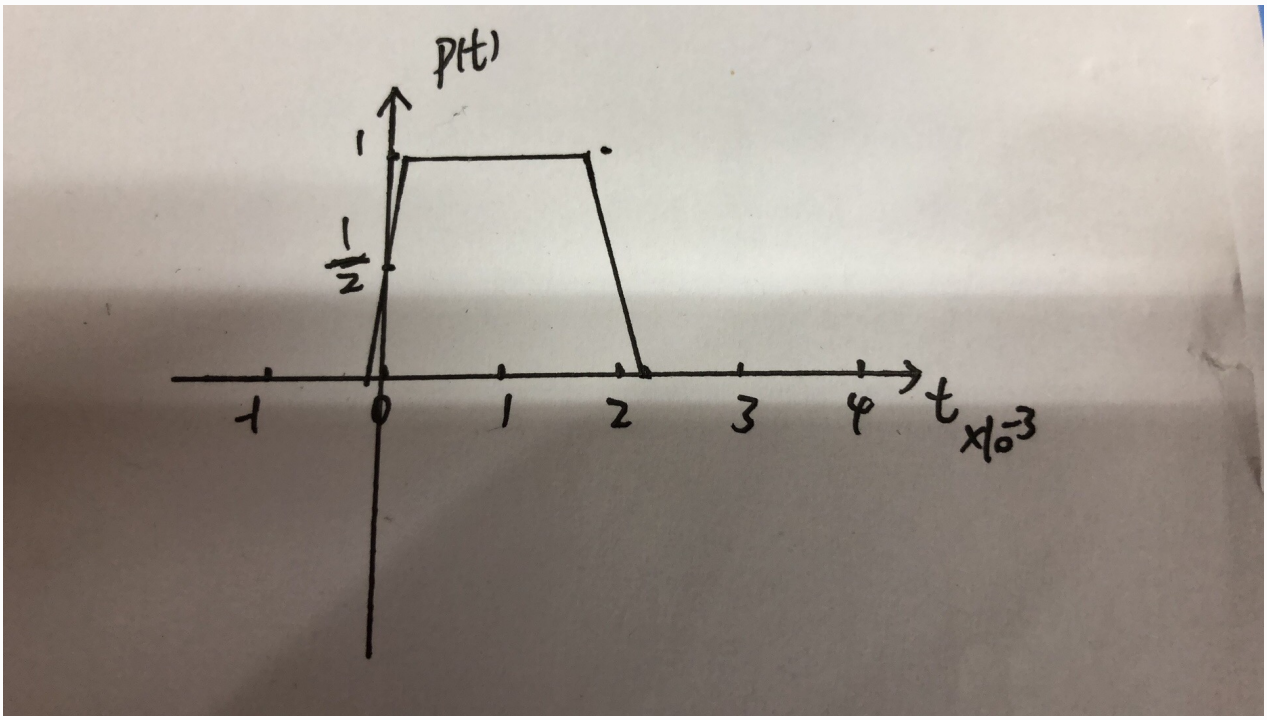
Analysis

As we can see from result, my function can work well. Furthermore, I use 2 steps length to get a bit pair from sequence and use the previous function to map.

Problem 2 (4 points)

Assume the transmit pulse $p(t) = rect(\frac{t-T_s/2}{T_s})$ and the symbol period $T_s = 2msec$. Sketch the transmit pulse by hand (So that you know what to expect in MATLAB). Plot the transmit pulse with MATLAB using a time resolution of 10 sample points per symbol period; pick a reasonable time interval. (You may use $rect.m$ from the ICE.)

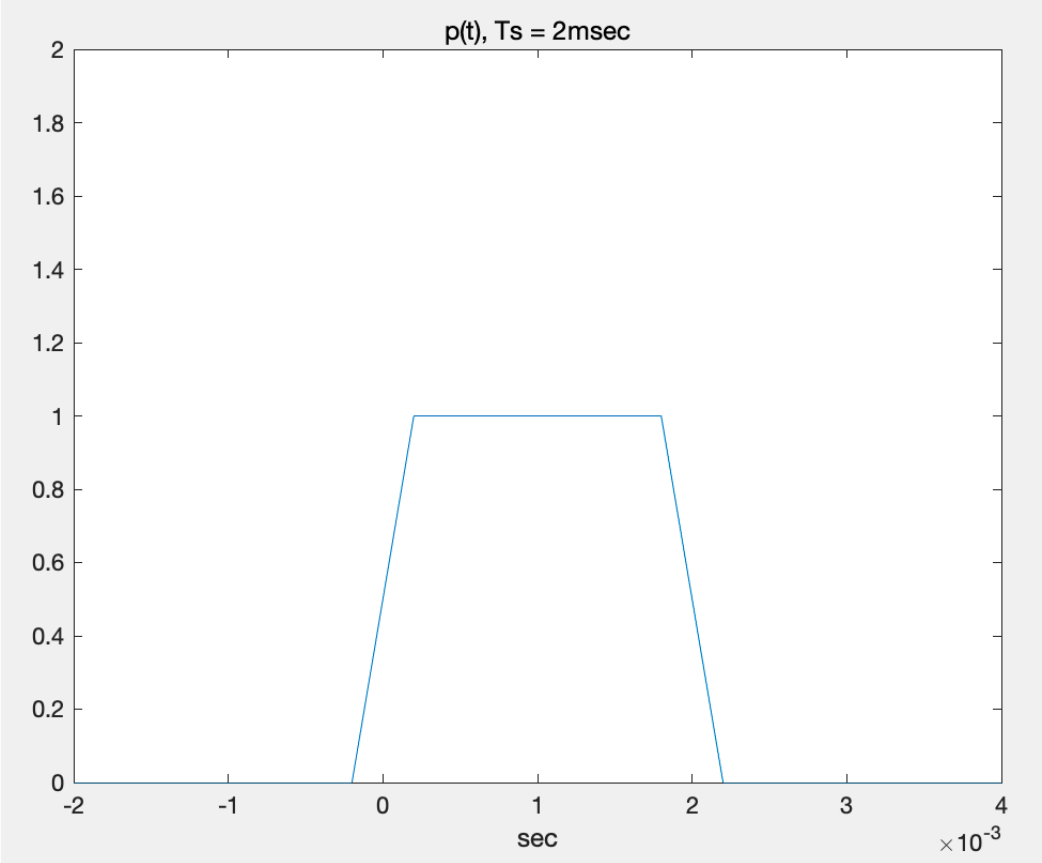
Sketch



Code

```
clear
clc

T_s = 2/1000;
T_0 = T_s/10
t=-1*T_s:T_0:2*T_s;
p=rect((t-T_s/2)/T_s);

plot(t,p);

xlabel('sec')
title('p(t), Ts = 2msec')
axis([-1*T_s,2*T_s,0,2])
```

Result



Analysis

The result is corresponding to sketch.

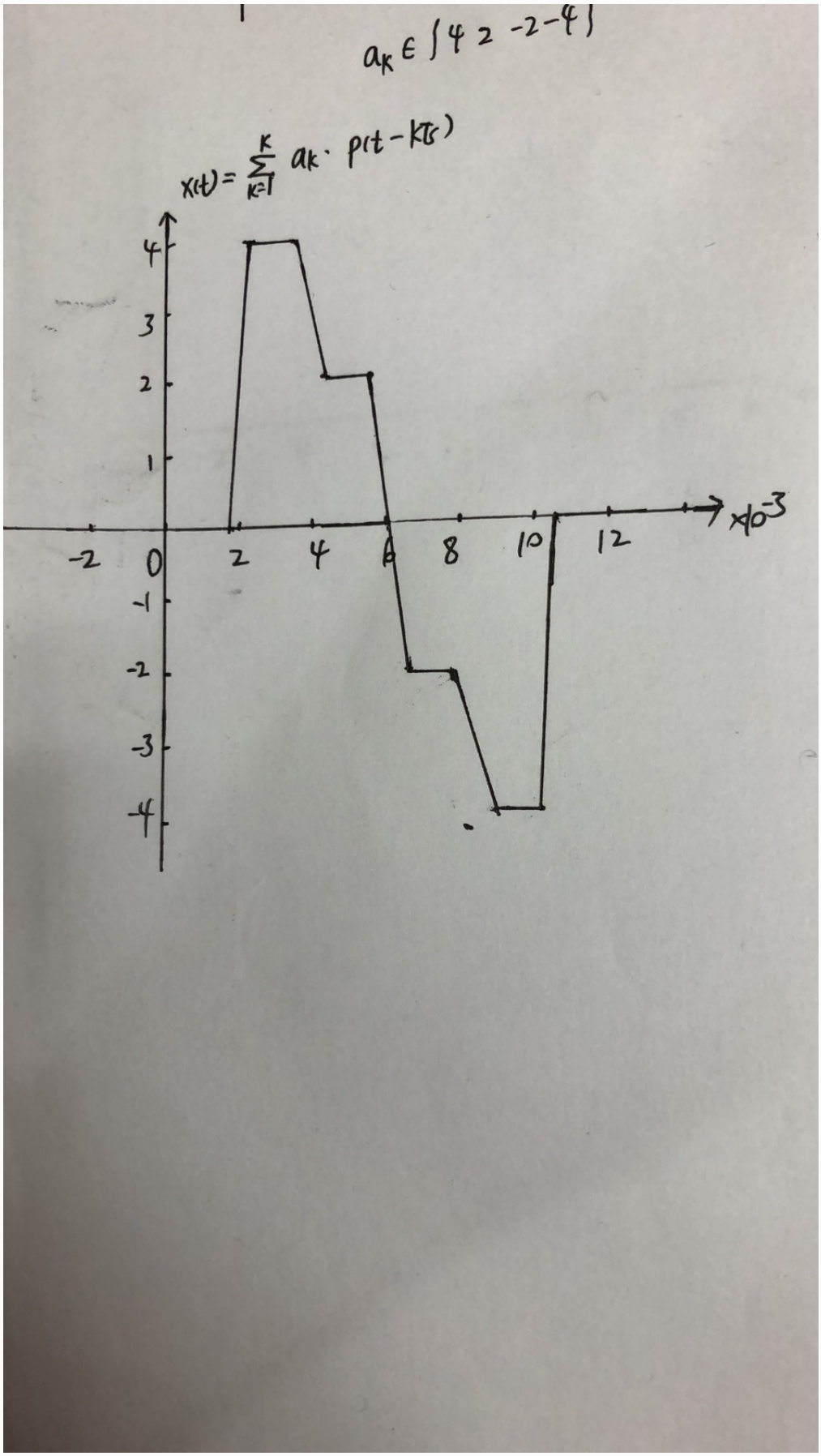
Problem 3 (4 points)

Assume a sequence of K modulation symbols: $a_1, a_2, \ldots, a_K$. The corresponding transmit signal is given by

$$x(t) = \sum_{k=1}^{K} a_k \cdot p(t - kT_s).$$

Sketch by hand the transmit signal for a sequence a_1, a_2, a_3, a_4 of four chosen modulation symbols.

Sketch



Analysis

I choose $4, 2, -2, -4$ as modulation symbols and 00011110 as bit sequence. The sketch is a sequence of signal and corresponding to theoretical curve.

Problem 4 (8 points)

Use the same modulation symbols as in the previous question. Write a Matlab script to plot the signal $x_1(t) = a_1 \cdot p(t - T_s)$ with MATLAB. You may use the following code:

```
t = 0:T0:6*Ts;
p = @(t) rect( (t-Ts/2)/Ts );
x1 = a1 * p(t-1*Ts);
plot(t,x1);
```

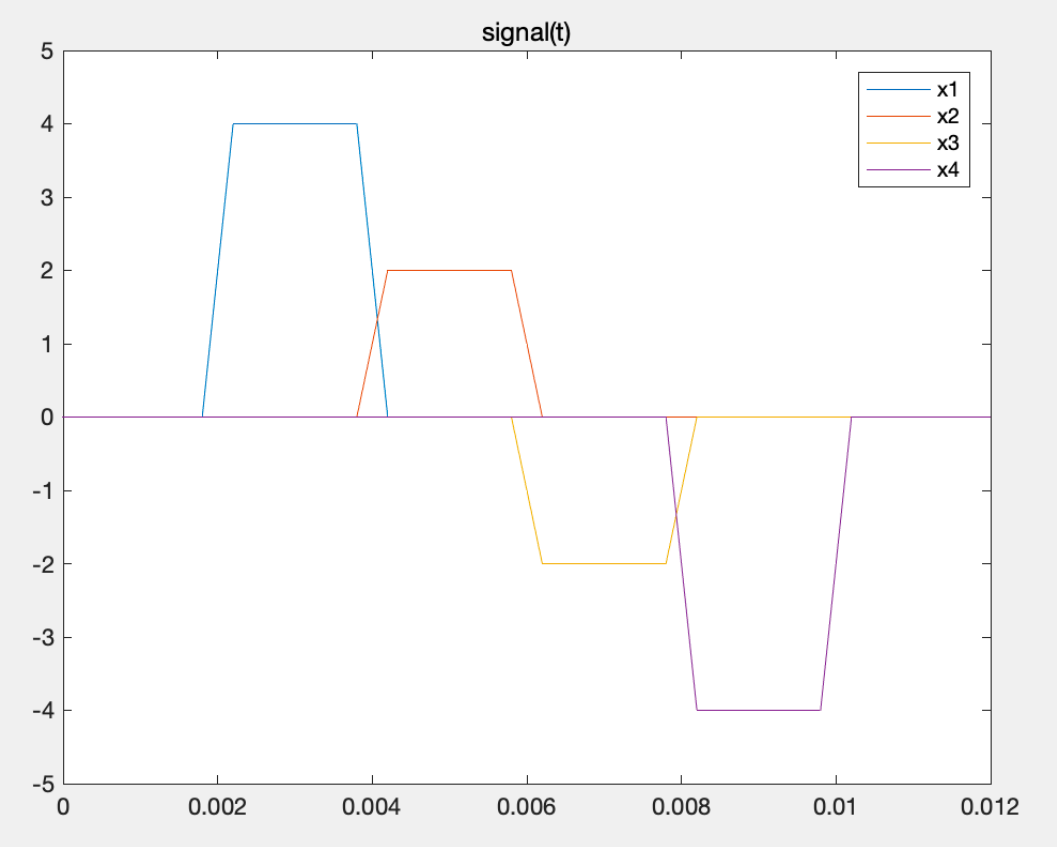
In a similar way, plot the signals $x_2(t)$,$x_3(t)$, and $x_4(t)$ in the same figure.

Code

```
clear
clc

a_1 = 4;
a_2 = 2;
a_3 = -2;
a_4 = -4;

T_s = 2/1000;
T_0 = T_s/10;
t = 0:T_0:6*T_s;
p = @(t) rect((t-T_s/2)/T_s);

x_1 = a_1 * p(t-1*T_s);
x_2 = a_2 * p(t-2*T_s);
x_3 = a_3 * p(t-3*T_s);
x_4 = a_4 * p(t-4*T_s);

X = x_1 +x_2+x_3+x_4;

plot(t,x_1)
axis([0*T_s, 6*T_s, -5 5])
title('signal(t)')
hold on

plot(t,x_2)
axis([0*T_s, 6*T_s, -5 5])
hold on

plot(t,x_3)
axis([0*T_s, 6*T_s, -5 5])
hold on

plot(t,x_4)
axis([0*T_s, 6*T_s, -5 5])
```

```
legend('x1','x2','x3','x4')
```

Result



Analysis

I use k to shift signal at T_s interval.

Problem 5 (8 points)

Write a Matlab function that implements the modulator, i.e. takes a sequence of modulation symbols as modulation symbols as the input and produces the transmit signal as the output. Test your function by comparing it to the result of the previous two questions.

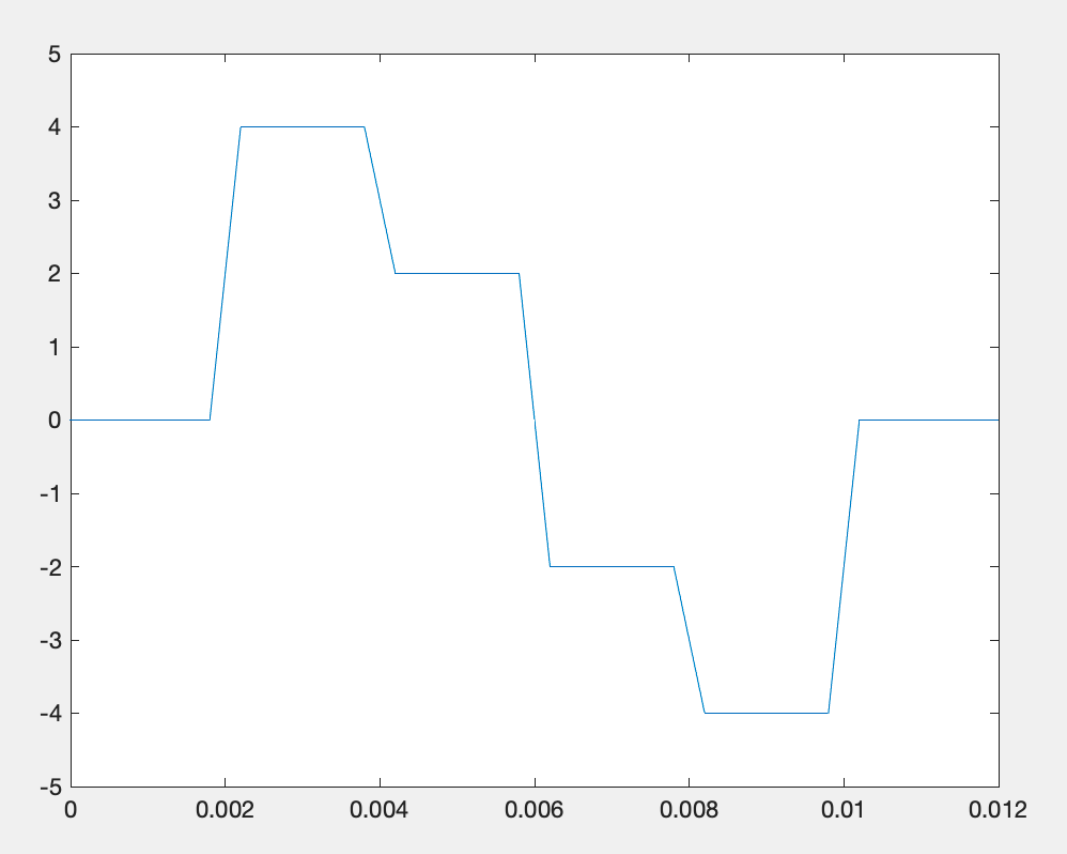
Code

```
function transmit_signal = modulator(sequence)

transmit_signals = []

% decode sequence
decode_value = decode_binary(sequence)

value = split(decode_value,',')

a_k = str2num(char(value))
```

```
length_ak = length(a_k)

T_s = 2/1000;
T_0 = T_s/10;
t = -2*T_s:T_0:6*T_s;
p = @(t) rect((t-T_s/2)/T_s);

for i = 1:length_ak
    transmit_signal = a_k(i) * p(t-i*T_s);
    transmit_signals = [transmit_signals;transmit_signal];
end

transmit_signal = sum(transmit_signals)

end
```

```
clear
clc

T_s = 2/1000;
T_0 = T_s/10;
t = 0:T_0:6*T_s;

% generate rand sequence
% @ parameter the length of sequence
% sequence = create_test_sequence(8);
% default sequence
sequence = '00011110';

transmit_signal = modulator(sequence)

plot(t, transmit_signal)

axis([0, 6*T_s, -5 5])
```

Result

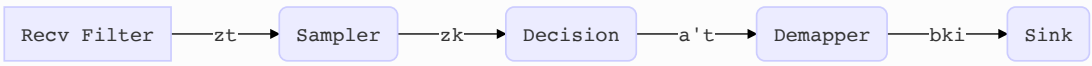


Analysis

The transmit signal is corresponding to previous 2 problems. However the signal is $x_k(t) = a_k \cdot p(t - T_s)$ in problem 4, and the signal is $x(t) = \sum_{k=1}^{K} a_k \cdot p(t - kT_s)$ in problem 5. Therefore, there is a little difference between them.Because $x(t)$ is the sum of $x_k(t)$. But essence of these problem is same.

Digital demodulator

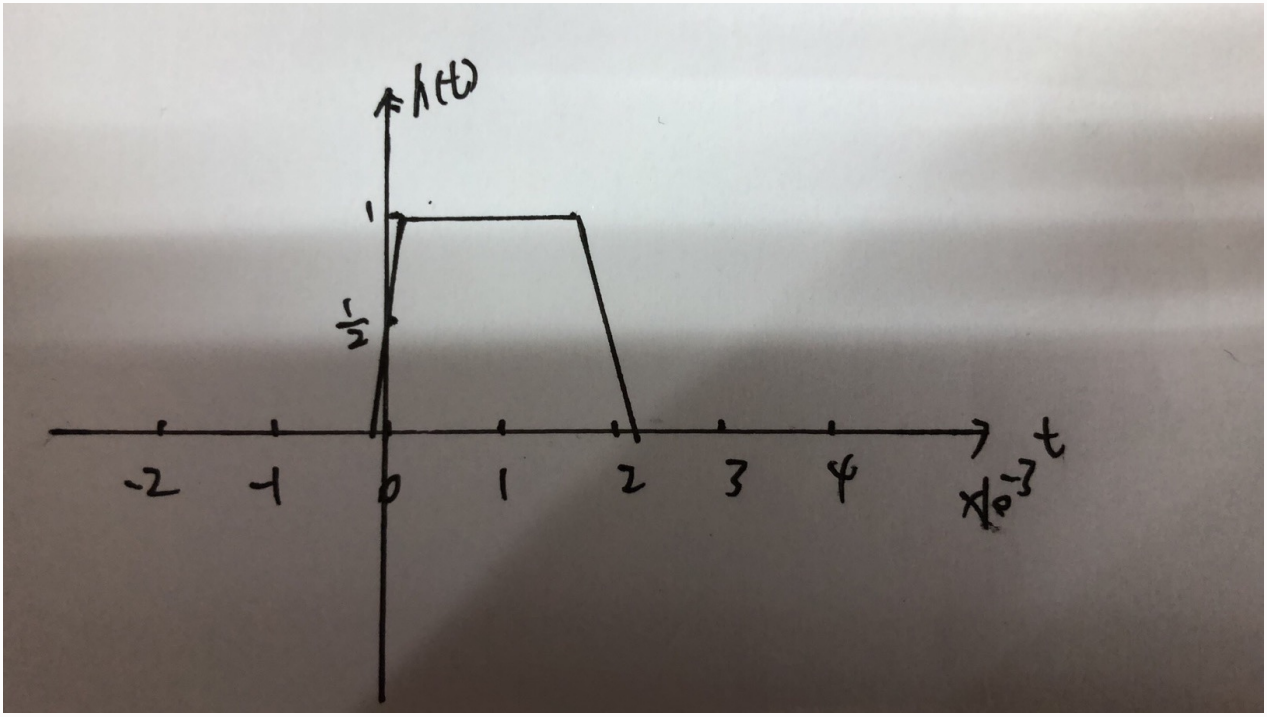
The figure shows a block diagram of a digital receiver.



Problem 6 (4 points)

Assume that the receive filter is a matched filter, $i.e.$, the impulse response of the receiver filter is $h(t) = p(T_s - t)$. Sketch by hand the impulse response of the receive filter (with the correct time shift).

Sketch

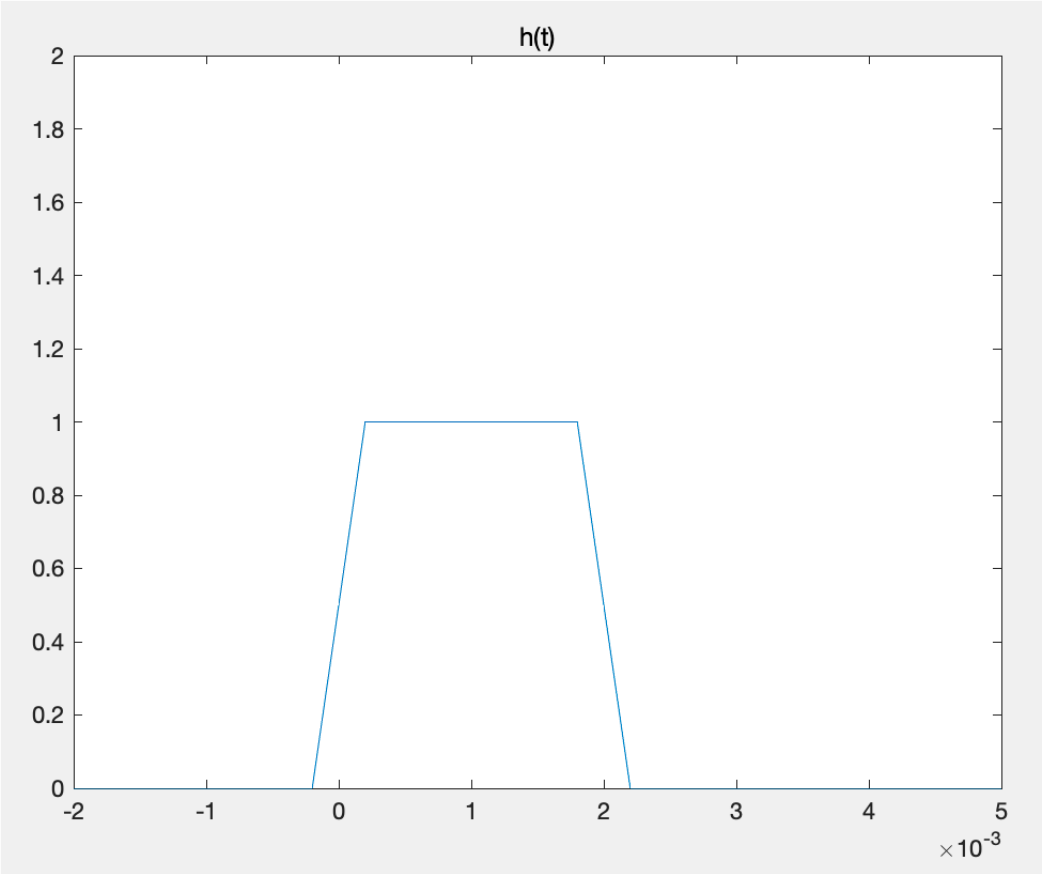


Code

```
function h_t = match_filter(t)
    T_s = 2/1000;
    p = @(t) rect( (t-T_s/2)/T_s );
    h_t = p(T_s - t)
end
```

```
clear
clc

T_s = 2/1000
T_0 = T_s /10
t = -2*T_s:T_0:6*T_s

h_t = match_filter(t);

plot(t, h_t)
title('h(t)')
axis([-2/1000, 5/1000, 0 , 2])
```

Result



Analysis

As we can see from result, the figure of $h(t)$ is same as the figure of $p(t)$.Because $p(t) = rect(\frac{t-T_s/2}{T_s})$ and $h(t) = p(T_s - t) = rect(\frac{T_s/2-t}{T_s})$. Furthermore, the $rect(t)$ is a even function, so the figure of $h(t)$ is same as the figure of $p(t)$.

Problem 7 (4 points)

For the time being, assume noise-free transmission. In this case, the received signal is $y(t) = x(t)$. The output signal of the receive filter is $z(t) = y(t) * h(t)$. Show analytically (by a short calculation) that

$$z(t) = \sum_{k=1}^{k} a_k \cdot [p(t - kT_s) * h(t)].$$

Which properties do you use to show that?

Calculation

$$z(t) = \sum_{k=1}^{k} a_k \cdot [p(t - kT_s) * h(t)]$$

Because $p(t)$ and $h(t)$ are the same function and are two rectangle function. Moreover, the convoluation of two rectangle function is a triangle function (shape is triangle) . I use $\Delta(t)$ symbol to represent the triangle function. The equation is by

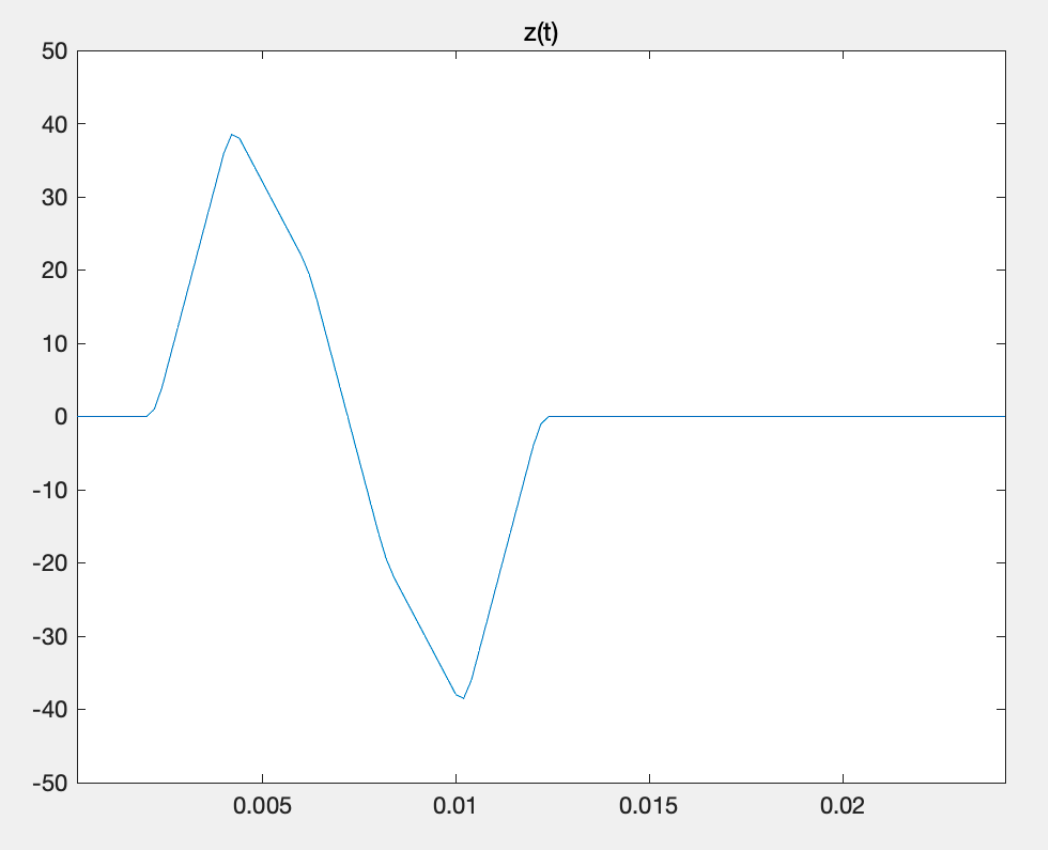
$$z(t) = \sum_{k=1}^{k} a_k \cdot [p(t - kT_s) * h(t)]$$
$$= \sum_{k=1}^{k} a_k \cdot \Delta(t - kT_s)$$

Therefore it is a triangle function with different amplitude and different time shift.

Code

```
clear
clc

T_s = 2/1000
T_0 = T_s /10
T_0_conv = T_0/2
t = 0:T_0:6*T_s
t_conv = 0:T_0_conv:6*T_s

h_t = match_filter(t);

signals_points = generate_defort_signals();

z_t = conv(signals_points,h_t);
tz = T_0*(1:length(z_t))
plot(tz,z_t)
title('z(t)')
axis([min(tz) max(tz), -50,50])
```

Result



Analysis

The length of signal is two times as much as the original signal and it is a triangle function (shape is triangle) with different amplitude and different time shift.

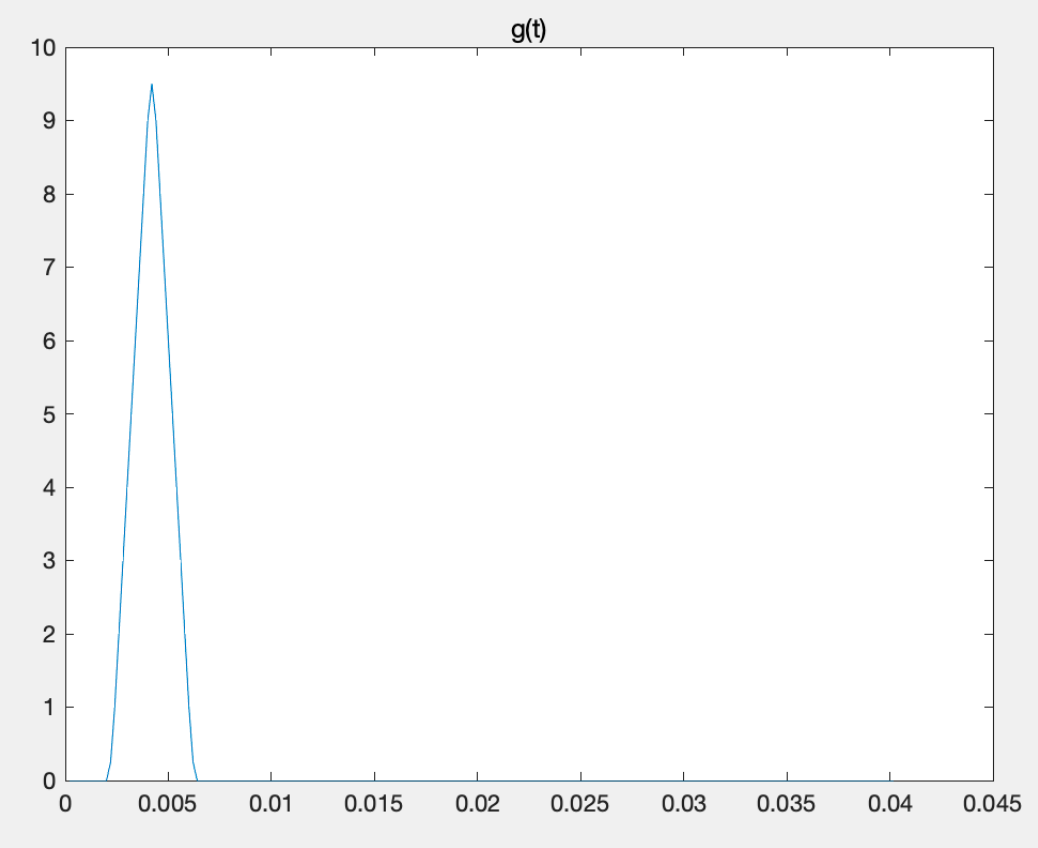
Problem 8 (4 points)

Use Matlab to compute and plot the signal $g(t) = p(t - kTs) * h(t)$ for $k = 1$. Discuss if the result is expected.

Code

```
clear
clc

T_s = 2/1000;
T_0 = T_s/10;
T_0_conv = T_0/2
t = 0*T_s:T_0:10*T_s;
p = @(t) rect((t-T_s/2)/T_s);

h = match_filter(t)

g = conv(p(t-T_s*1),h)

zt = T_0 * (1:length(g))
plot(zt, g)
title('g(t)')
```

Result



Analysis

The result is expected. It is a triangle function (shape is triangle) because of the convoluation of two rectangle function.

Problem 9 (10 points)

Assume the transmit signal $x_1(t)$ from Problem 4. Compute and plot the corresponding signal $z(t)$.

You may use the following code:

```
tp = 0:T0:(Ts-T0);
h = p(tp);
z = T0/Ts * conv(y,h);
tz = T0 * (1:(length(z)));
plot(tz, z)
```

Relate your result to $g(t)$ from the previous question.

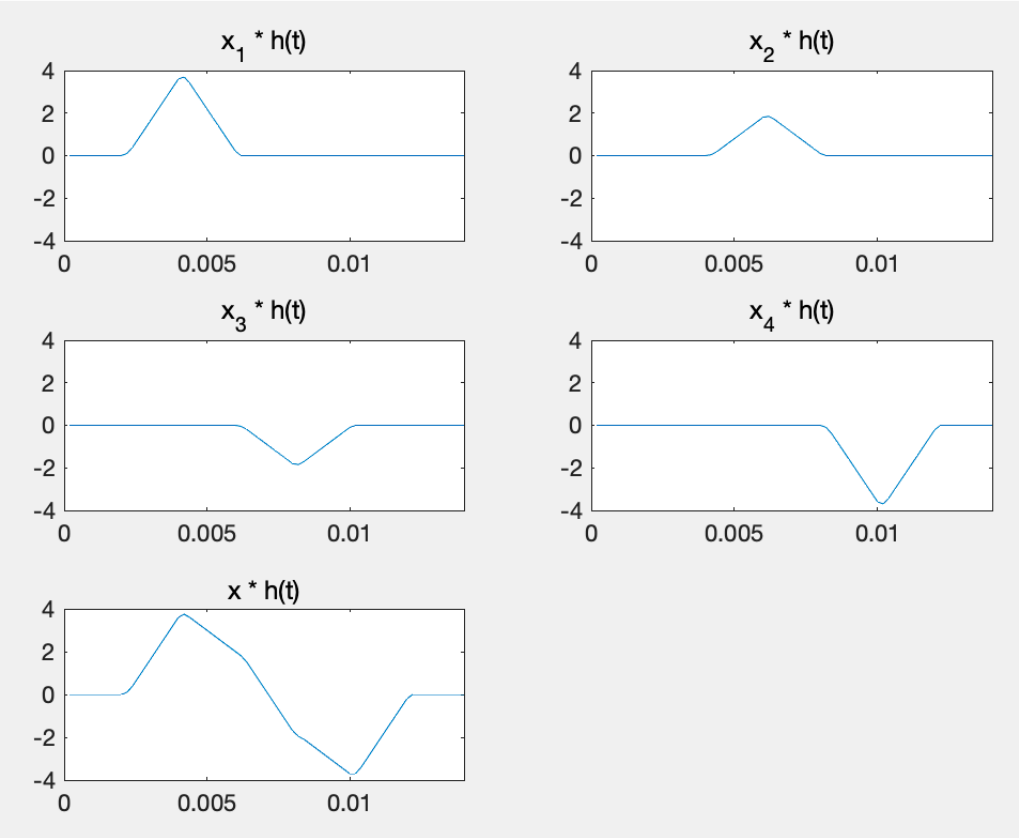
Repeat this with $x_2(t)$. Is the result as expected? Now repeat this with the full transmit signal $x(t)$ and interpret the result.

Code

```
clear
clc
```

```
a_1 = 4;
a_2 = 2;
a_3 = -2;
a_4 = -4;

T_s = 2/1000;
T_0 = T_s/10;
t = 0:T_0:6*T_s;
p = @(t) rect( (t-T_s/2)/T_s );

x_1 = a_1 * p(t-1*T_s);
x_2 = a_2 * p(t-2*T_s);
x_3 = a_3 * p(t-3*T_s);
x_4 = a_4 * p(t-4*T_s);
% X = x_1 +x_2+x_3+x_4;

tp = 0:T_0:(T_s-T_0);
h = p(tp)

z_1 = T_0/T_s * conv(x_1,h);
z_2 = T_0/T_s * conv(x_2,h);
z_3 = T_0/T_s * conv(x_3,h);
z_4 = T_0/T_s * conv(x_4,h);
Z = z_1+z_2+z_3+z_4

tz = T_0 * (1:(length(z_1)));

subplot(3,2,1)
plot(tz, z_1)
title('x_1 * h(t)')
axis([0,T_0*length(z_1),-4,4])

subplot(3,2,2)
plot(tz,z_2)
title('x_2 * h(t)')
axis([0,T_0*length(z_2),-4,4])

subplot(3,2,3)
plot(tz,z_3)
title('x_3 * h(t)')
axis([0,T_0*length(z_3),-4,4])

subplot(3,2,4)
plot(tz,z_4)
title('x_4 * h(t)')
axis([0,T_0*length(z_4),-4,4])

subplot(3,2,5)
plot(tz,Z)
title('x * h(t)')
axis([0,T_0*length(Z),-4,4])
```

Result



Analysis

The convoluation of two rectangle function is a triangle function (shape is triangle) . Therefore, the result is a series of 'triangle function' and is submitted to this formula.

$$z(t) = \sum_{k=1}^{k} a_k \cdot [p(t - kT_s) * h(t)]$$
$$= \sum_{k=1}^{k} a_k \cdot \Delta(t - kT_s)$$

Problem 10 (8 points)

Based on the results of the previous problem, discuss what are the optimal sampling times for $z(t)$, to determine the samples z_k. Write a Matlab function (or script) that obtains the samples z_k from $z(t)$.

Code

```
function zk = generate_zk(Z,num)

% @paraam Z means sequence
% @param num means the number of symbol

zk = []
step = 10
stop_value = 11 + num * step

for i = 21:step:stop_value
    zk = [zk Z(i)]
end
end
```

```
% follow by previous step
zk = generate_zk(Z,4)
```

Result

```
zk =

    3.7500    1.8000   -1.9500   -3.7000
```

Analysis

I use the 10 magnitude of step or 0.002 in time to obtain z_k from $z(t)$, because we can find the interval of $z_k(t)$ is 0.002, and we use 10 samping points per period. Therefore, I use 10 samping length to get z_k.

Problem 11 (12 points)

Assume now transmission over an AWGN channel: $y(t) = x(t) + w(t)$. With Matlab you can obtain a noisy receive signal with the following commands:

```
varnoise = 1;
y = x + sqrt(varnoise) * randn(size(x));
```

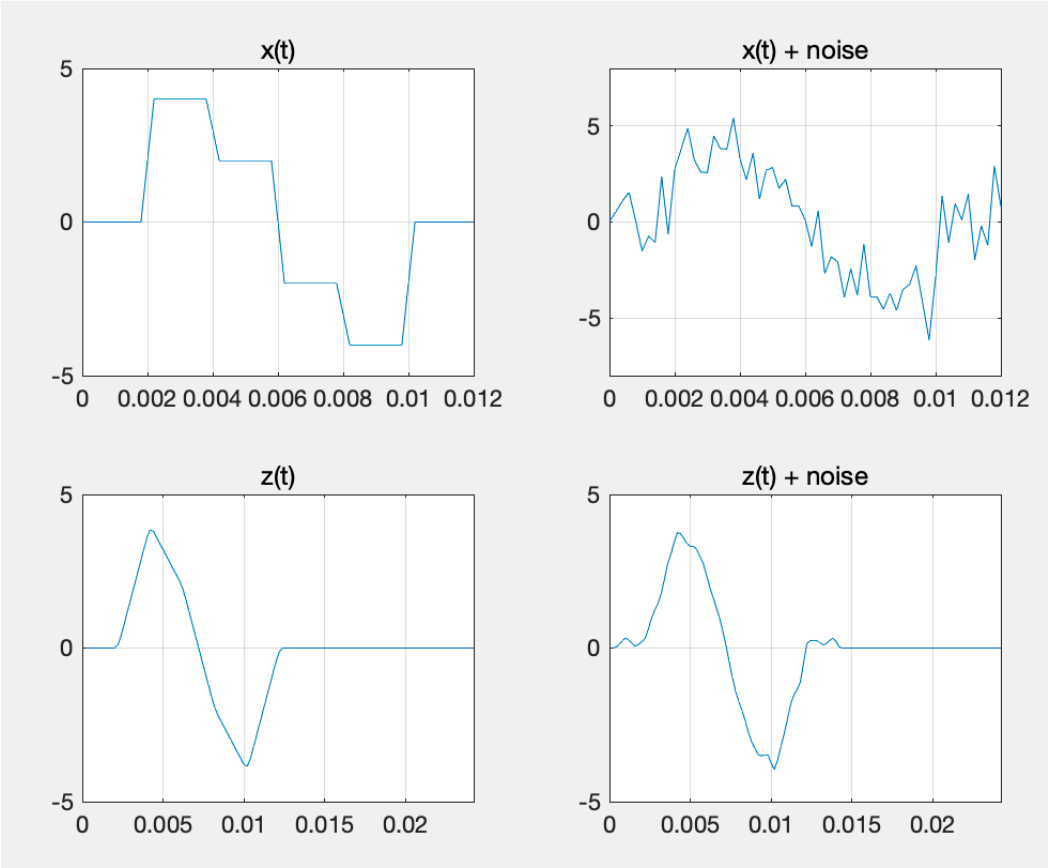
The value of **varnoise** denotes the variance of the additive white Gaussian noise. Write a Matlab script that plots (in subplots) the transmit signal, the noisy receive signal, the signal at the matched filter output, and prints out the samples z_k.

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

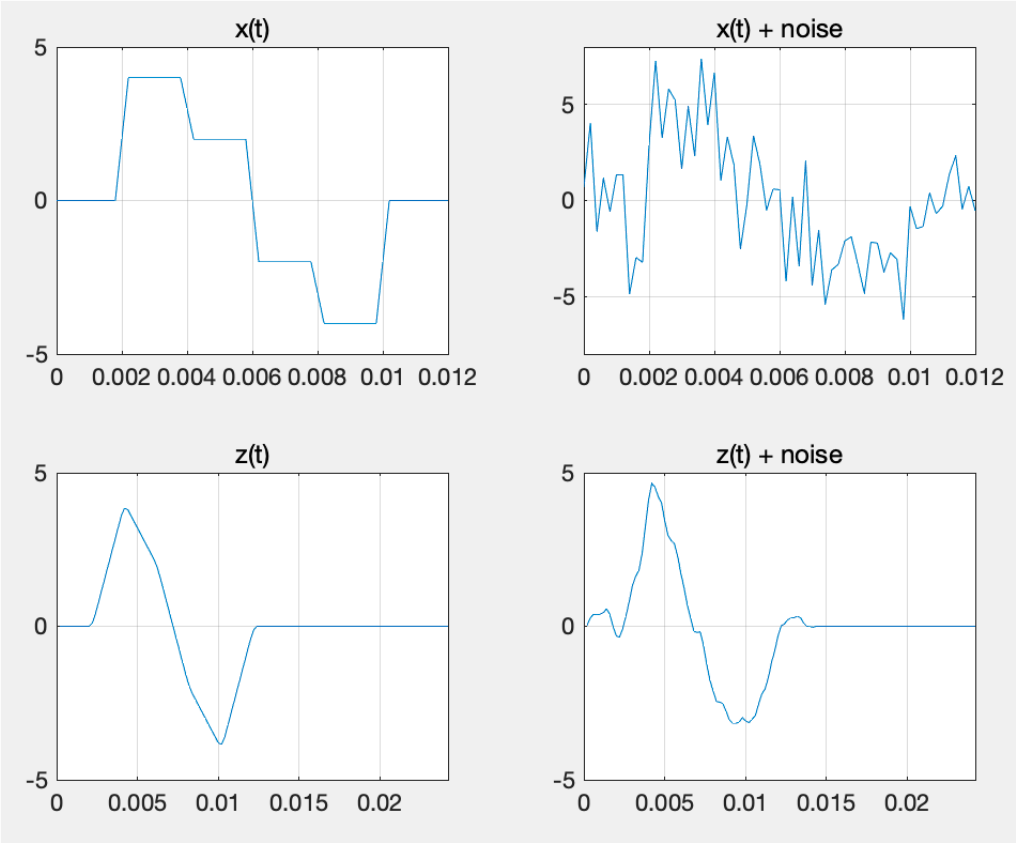
Code

```
clear
clc

T_s = 2/1000
T_0 = T_s /10
t = 0:T_0:6*T_s

singals_points = generate_defort_signals()

[m,n] = size(singals_points)

subplot(2,2,1)
plot(t,singals_points)
grid on
axis([0 max(t) -5 5])
title('x(t)')

signal_noise_y = add_noise_parameters(singals_points,0,1)
subplot(2,2,2)
plot(t,signal_noise_y)
grid on
axis([0 max(t) -8 8])
title('x(t) + noise')

h_t = match_filter(t);

z_t = T_0/T_s*conv(singals_points,h_t);
subplot(2,2,3)
tz = T_0*(1:length(z_t))
plot(tz,z_t)
grid on
axis([0 max(tz) -5 5])
title('z(t)')

z_t_noise =  T_0/T_s*conv(signal_noise_y, h_t)
subplot(2,2,4)
plot(tz,z_t_noise)
grid on
axis([0 max(tz) -5 5])
title('z(t) + noise')

zk = generate_zk(z_t_noise,4)
```

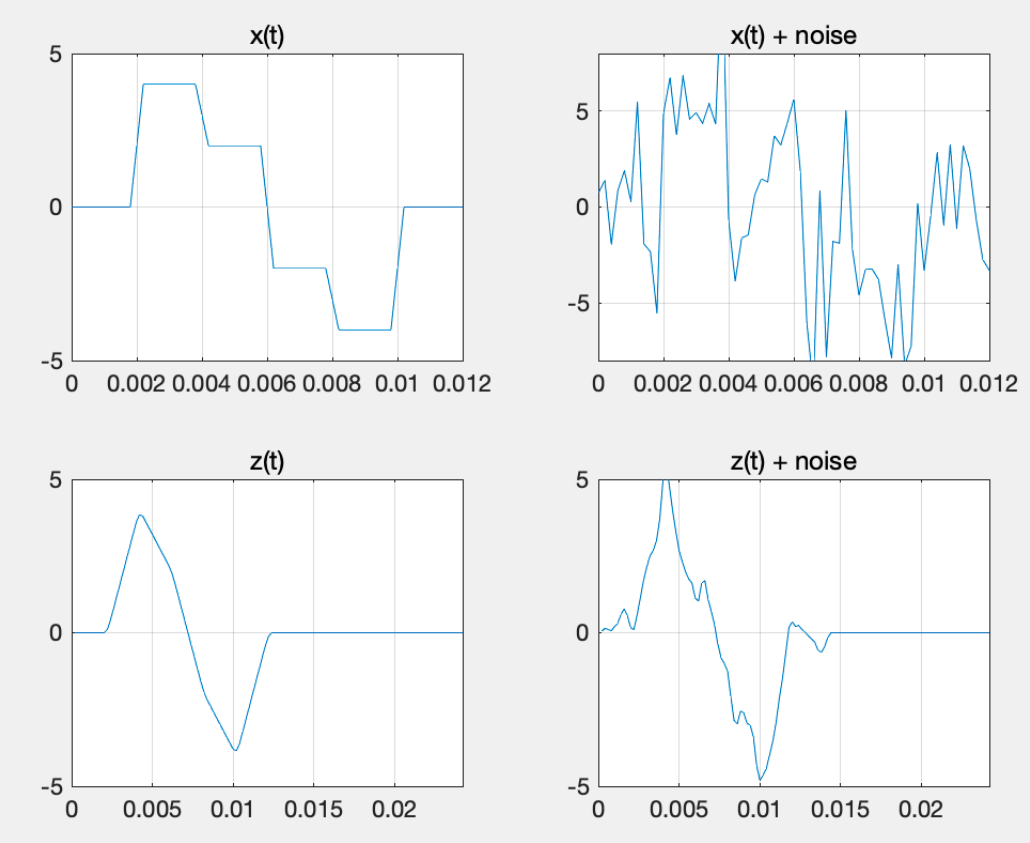
Result



noise with 1 variance



noise with 5 varian



noise with 10 variance

Analysis

As we can see from result, I choose various variance noise to experiment. Noise damage the signal on $y(t)$ and $z(t)$ to some extend. However noise has a bigger impact on $y(t)$ than $z(t)$. Noise almost destroy shape of $y(t)$, but $z(t)$ can be distinguished when variance of noise is 10. Therefore, it is easier to estimate the transmitted modulation symbol a_k from $z(t)$.

Problem 12 (6 points)

Consider now the decision block. Determine the decision regions for each modulation symbol. Then write a MATLAB function that implements the decision function, $i.e.$, that maps z_k to $\hat{a}_k$.

Code

```
function ak_sequence = map_zk_to_ak(zk)

length_value = length(zk);
ak_sequence = [];

for i=1:length_value
a_k = [4,2,-2,-4];

distance = abs(a_k - zk(i));

% key-value
```

```
alphabet = containers.Map(distance, a_k);

% sort distance
distance_sorted = sort(distance);

ak = alphabet(distance_sorted(1));

ak_sequence = [ak_sequence ak];
end
end
```

```
ak = map_zk_to_ak(zk)
```

Result

```
zk =

    3.4651    1.7040   -2.0827   -4.0790


ak =

     4     2    -2    -4
```

Analysis

I use the distance to get the closest point. For example, when we receive 2.1 , we can calculate the distance to vector $[4, 2, -2, -4]$. Then the result is $[1.9, 0.1, 4.1, 6.1]$. Therefore the shortest distance is 0.1 and 0.1 is the absolute value of $2.1 - 2$. Finally, the answer is 2. In addition, I use hashmap which is called dictionary in python to implement the mapping from distance to modulation symbol.

Problem 13 (6 points)

Write a Matlab function that implements the Demapper. (This is very similar to your Mapper, written before.)

Code

```
function bk = demapper(ak)

length_value = length(ak)

alphabet = containers.Map({4, 2, -2,-4},...
    {'00', '01', '11','10'});

binary_sequence = []

for i =1 :length_value
    binary_value = alphabet(ak(i))
    binary_sequence = [binary_sequence binary_value]
```

```
end
    bk = join(binary_sequence,',')
end
```

```
 bk = demapper(ak)
```

Result

```
ak =

     4     2    -2    -4

>> bk = demapper(ak)

bk =

    '00011110'
```

Analysis

Because I use the hashmap (dictionary) in the form of $00-4$ in previous problems. In this problem, I use the hashmap in the form of $-4-00$ to map which is similar to previous function.

Digital transmission system

In the previous questions we have investigated the individual components required for a baseband digital communication system. We now put all the components together.

Problem 14 (10 points)

Write a Matlab script that implements the transmitter, the channel and the receiver.

The start is a sequence of data bits (you may choose them or pick them randomly), and the end is the sequence of estimated data bits.

Code

```
function bk = match_filter_test(binary_sequence, mu, variance)

% @ param mu is mean of noise
% @ param variance is the variance of noise

% generate original signal
signals_points = modulator(binary_sequence);

% add noise to original signal
xt_noisy = add_noise_parameters(signals_points,mu,variance);
```

```
% create h(t) signal
T_s = 2/1000;
T_0 = T_s /10;
t = 0:T_0:6*T_s;

ht = match_filter(t);

% convolute x(t) and h(t)
Z = T_0/T_s * conv(xt_noisy,ht);

% samping zk from zt
zk = generate_zk(Z,4);

% map zk to ak
ak = map_zk_to_ak(zk);

% map ak to bk
bk = demapper(ak);

end
```

```
function str = create_test_sequence(length)

% length means the length of binary sequence

if(mod(length,2) == 1)
    length = length +1;
end

str = '';
for i =1:length
    if(rand>0.5)
        str(i) = '1';
    else
        str(i) = '0';
    end

end
end
```

```
original_sequence = create_test_sequence(8);
bk = match_filter_test(original_sequence,0,1);
```

Result

```
original_sequence =

    '01010010'
bk =

    '01010010'
```

Analysis

As we can see from result, when noise with variance of 1, signal can be recover successfully. It is expected.

Problem 15 (6 points)

Test your script for very low noise and for higher noise. Does the system react as expected?

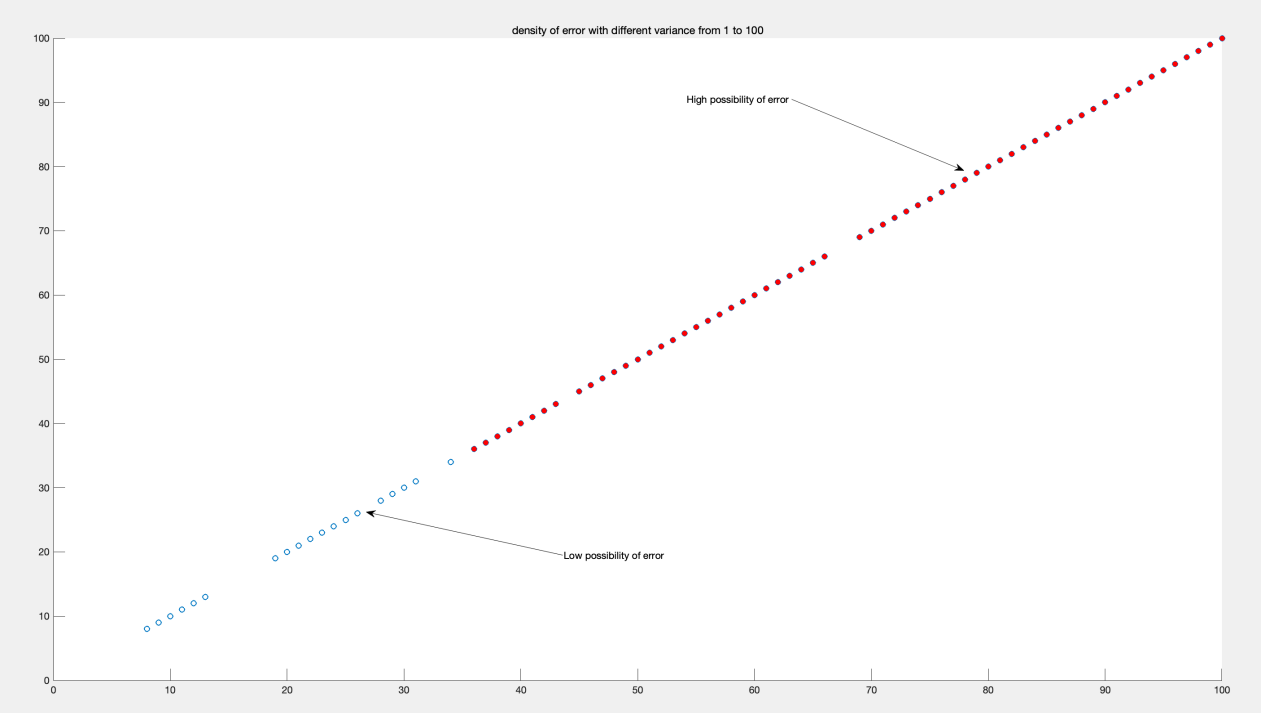
Code

```
original_sequence_list = []
bk_list = []

% increase variance to test function
for i = 1:100
    original_sequence = create_test_sequence(8);
    bk = match_filter_test(original_sequence,0,i);

    original_sequence_list = [original_sequence_list;original_sequence]
    bk_list = [bk_list;bk]
end

% check different index
error_position = original_sequence_list ~= bk_list

[m,n] = size(error_position)

% find the variance when error happen
variance_position = []
for i = 1:m
    if(ismember(1,error_position(i,:)))
        sprintf('When variance is %d, function cause error.',i)
        variance_position = [variance_position,i]
    end
end

% show the density of error
scatter(variance_position,variance_position)
```

Result

```
error_position =

  50×8 logical array

   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
   % 0   1   0   0   1   0   0   0
   0   0   0   0   0   0   0   0
```

```
% 0   0   0   1   0   0   0   0
% 0   0   0   1   0   0   0   0
% 0   0   0   0   0   1   0   0
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
% 0   0   1   0   0   0   0   1
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
% 0   1   0   0   0   1   0   1
0   0   0   0   0   0   0   0
% 0   1   0   1   1   0   0   0
0   0   0   0   0   0   0   0
% 0   1   0   0   0   0   0   1
% 0   1   0   0   0   0   0   0
% 0   0   0   1   0   0   0   0
0   0   0   0   0   0   0   0
% 0   0   0   1   0   0   0   1
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
% 0   1   0   0   0   0   0   0
0   0   0   0   0   0   0   0
% 0   0   0   1   0   1   0   1
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
% 0   0   0   0   0   1   0   0
% 0   0   0   0   1   0   0   0
% 0   0   0   0   0   1   0   1
0   0   0   0   0   0   0   0
% 0   0   0   1   0   1   0   0
% 0   1   0   1   0   0   0   1
% 0   0   0   0   0   0   0   1
% 1   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
% 0   0   0   1   1   0   0   1
% 1   0   1   0   1   0   0   0
% 0   0   0   0   0   1   0   0
% 0   0   1   0   0   0   0   0
% 0   1   0   0   0   1   0   1
% 0   1   0   0   0   0   0   1
% 0   1   0   0   0   0   0   0
% 0   0   0   0   0   1   0   0
```



Analysis

I test my function by changing vatiance from 1 to 100. Firstly, I write a function to find the error position of different variance. I find the frequency of error become larger when varance become larger. They are positively correlated. Therefore, I plot the figure about this position when error happen. It is obvious that when noise become larger, the function can't recover signal successfully. However, when variance is smaller than 20. The system has a good performance. Therefore, the result is expected.