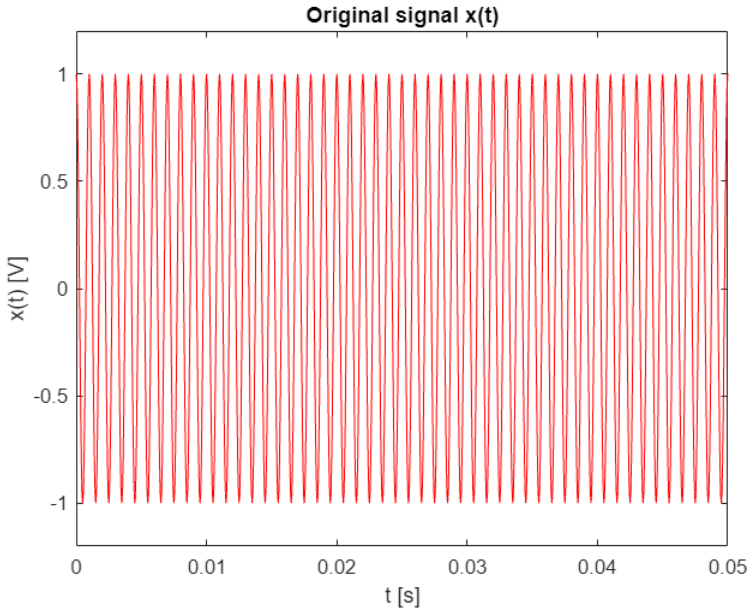
White Gaussian Noise Generator

```
close all % close all plots
clear all % clear all variables
clc % clear the screen
```

Generation of the sinusoid and plot

The following code segment generated sinusoidal signal with parameters defined below and plotting.

```
%% generation of the sinusiod and plot
A=1; % Amplitude of the sinusoid  [V]
f0=1000; % frequency of the sinusoid
fs=20000; % sampling frequency
duration=0.05; % singal duration in seconds
[t,x,N]=SinusoidalSource2023(A,f0,duration,fs); % generation of the singal

plot(t,x,'r')
xlabel('t [s]')
ylabel('x(t) [V]')
title('Original signal x(t)')
axis([min(t) max(t) 1.2*min(x) 1.2*max(x)])
```



The original signal is a sinusoidal waveform with an Amplitude of 1V, frequency of 1000hz and duration is 0.05seconds.

```
signal_power=0.5*A^2;
```

```
fprintf('sinusoid power [V^2]=%f' , signal_power)
```

```
sinusoid power [V^2]=0.500000
```

Noise Generation

The following code segment generated Gaussian noise signal with parameter defined below and added to the original sinusoidal signal. The power of noise and SNR is calculated. Finally, the original signal and the noisy signal are compared.

```
%% Noise generation
sigma=0.6; % std deviation of the noise. The noise power is sigma ^2 [v^2]
noise=sigma*randn(1,N); % generation the noise
fprintf('Noise power[V^2]=%f', sigma^2)
```

```
Noise power[V^2]=0.360000
```

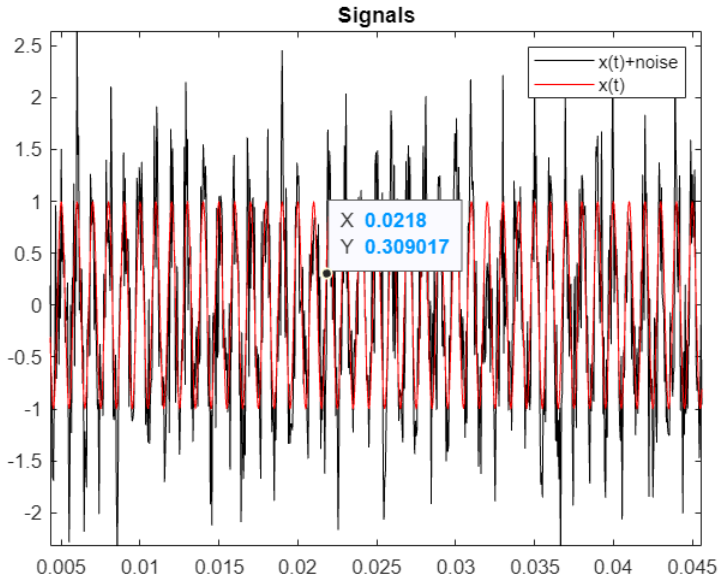
```
x_noisy=x+noise; % add Gussian noise to sinusoid

signal_to_noise_ratio_dB=10*log10(signal_power/(sigma^2));
fprintf('SNR [dB]=%f',signal_to_noise_ratio_dB)
```

```
SNR [dB]=1.426675
```

```
figure
plot(t,x_noisy,'k')
hold on
plot(t,x,'r')
legend('x(t)+noise','x(t)')
title('Signals');

xlim([0.0043 0.0456])
ylim([-2.32 2.64])

xt = findobj(gcf, "DisplayName", "x(t)")
```

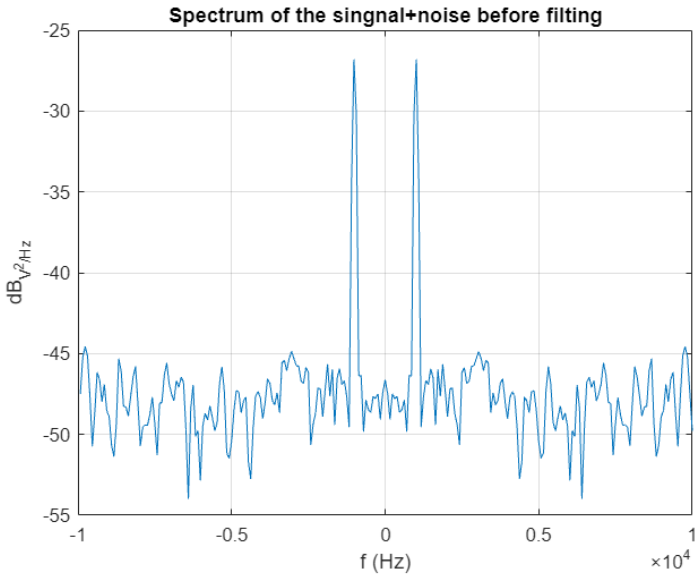
```
xt =
  Line (x(t)) with properties:

            Color: [1 0 0]
        LineStyle: '-'
        LineWidth: 0.5000
           Marker: 'none'
       MarkerSize: 6
  MarkerFaceColor: 'none'
            XData: [0 5.0000e-05 1.0000e-04 1.5000e-04 2.0000e-04 2.5000e-04 3.0000e-04
3.5000e-04 4.0000e-04 4.5000e-04 5.0000e-04 5.5000e-04 6.0000e-04 6.5000e-04 7.0000e-04
7.5000e-04 8.0000e-04 8.5000e-04 9.0000e-04 9.5000e-04 … ] (1×1001 double)
            YData: [1 0.9511 0.8090 0.5878 0.3090 6.1232e-17 -0.3090 -0.5878 -0.8090 -
0.9511 -1 -0.9511 -0.8090 -0.5878 -0.3090 -1.8370e-16 0.3090 0.5878 0.8090 0.9511 1 0.9511
0.8090 0.5878 0.3090 3.0616e-16 -0.3090 -0.5878 -0.8090 … ] (1×1001 double)

  Show all properties
```

```
datatip(xt,0.0218,0.309);
```



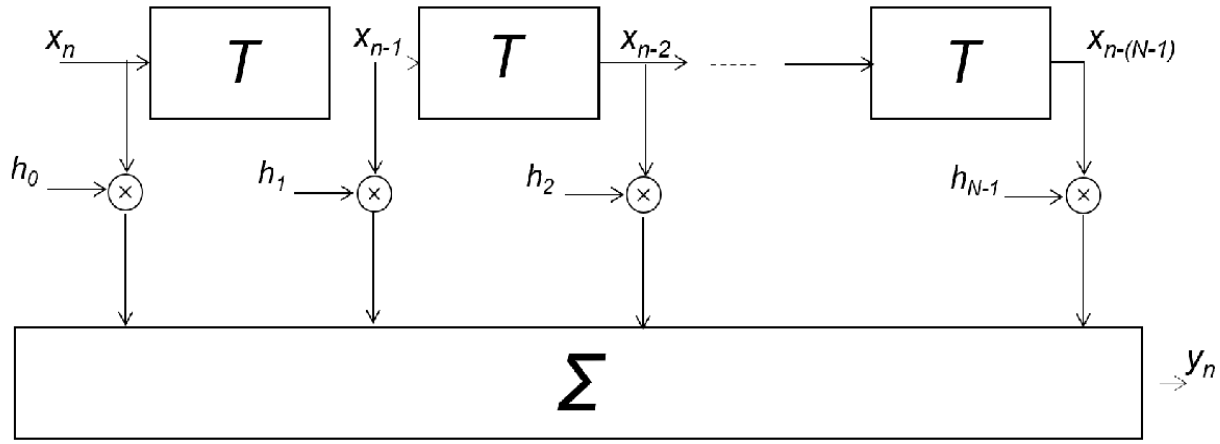
The original signal and noisy signal are plotted in one picture. In general, this picture shows that noisy signal amplitude is between 1.6 and 0.4 which is the amplitude plus sigma and amplitude minus sigma, the noise only affects amplitude, because of the AWGN feture: Additive, White, zero mean, Gaussian, stationary and uncorrelated.

```
figure
PlotSpectrum(x_noisy,fs);
title('Spectrum of the singnal+noise before filting');
```



The noise is white, this picture shows noise have a constant power spectral density across all frequency.

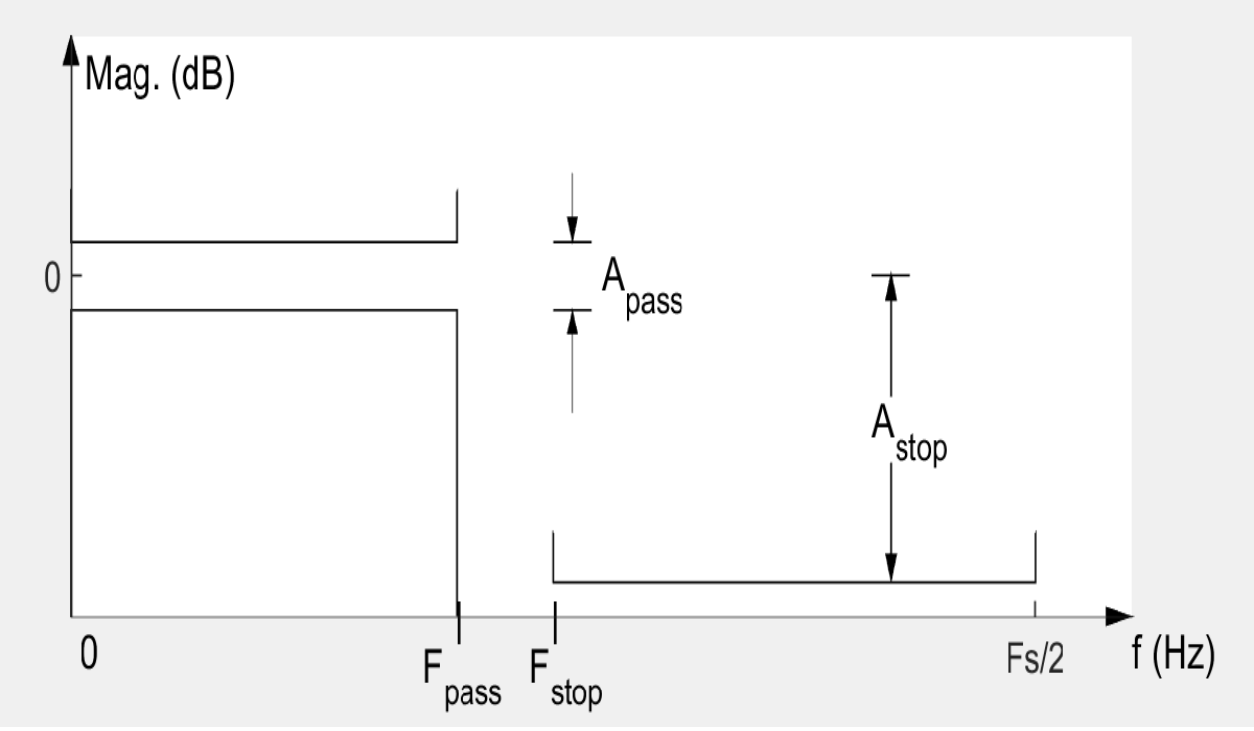
FIR Filter



This is a digital filter used to process signals by convolution operation. The input signal is convolved with filter coefficient.

```
Nf=400; % Number of FIR filter taps
```

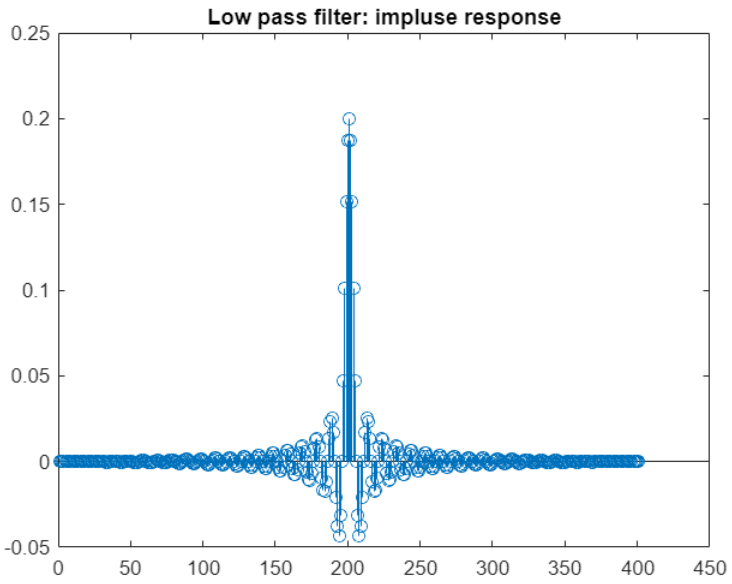
Low pass filter design



This is low pass filter, which allows signals with frequency lower than Fpass to pass through.

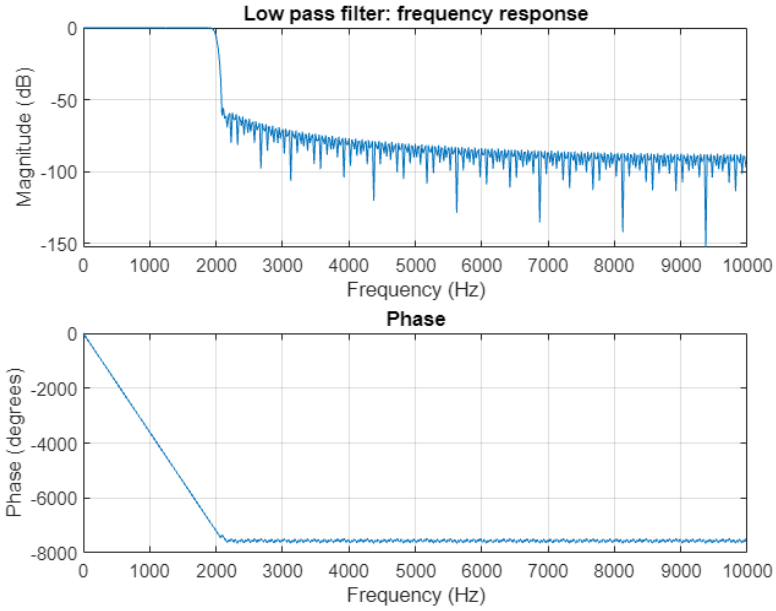
The following code segment generated a low pass filter with a 3dB cut off frequency Fpass is set to 2000 hz.

```
%% *Low pass filter design *
Fpass=2000; % 3dB cut frequency
h_lowpass=fir1(Nf, Fpass/(0.5*fs)); % filter design

%% *Filter impluse response and frequency respond (transfer function)*
stem(h_lowpass) %filter taps (coefficients), that is ,filter impluse resopnse
title('Low pass filter: impluse response')
```



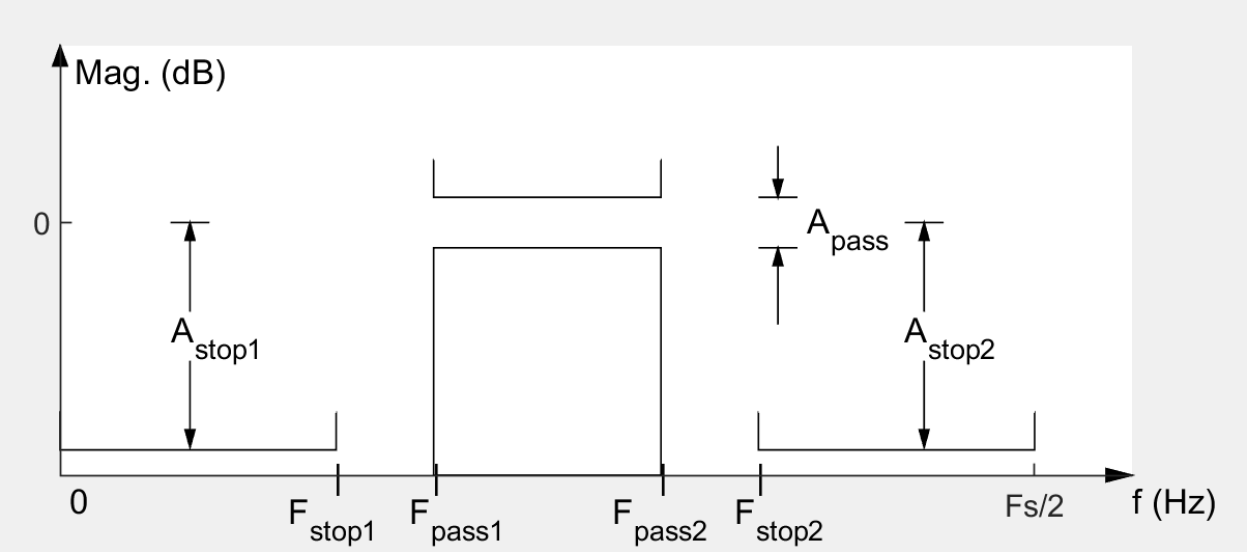
This picture shows the impulse response of low pass filter, in time domain, it is a sinc function.

```
freqz(h_lowpass,1,[],fs); % plot the frequency response
title('Low pass filter: frequency response')
```



In magnitude and frequency, the filter allows signals below Fpass=2000Hz to pass through truncating signals above 2000Hz. In phase and frequency, signals below 2000hz are linear which do not do not change the original information.

Passband filter design

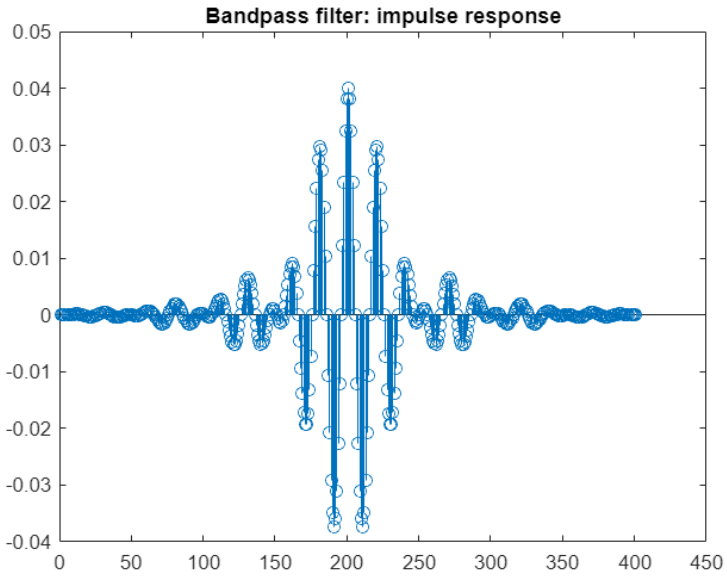


This is band pass filter, which allows signals with frequency in pass band to pass through.

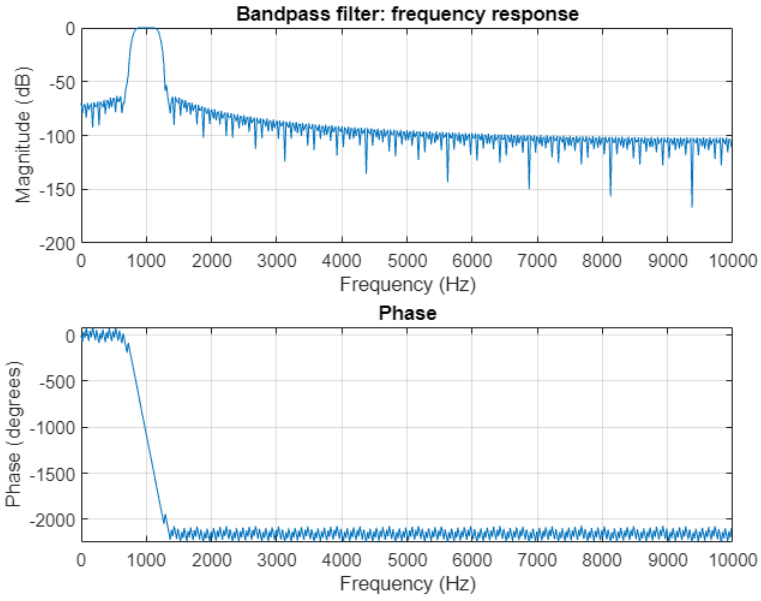
The following code segment generated a band pass filter with passband between 800hz-1200hz.

```
%% *Passband filter design*
Fpass1=800; % low cut frequency of the filter
```

```
 Fpass2=1200; % hign cut frequenct of the filter
 h_bandpass=fir1(Nf, [Fpass1/(0.5*fs) Fpass2/(0.5*fs)],'bandpass');% filer
design

 %% *filter impluse response and frequency response (transfer function)*
 stem(h_bandpass) % filter taps (coefficients), that is, filter impulse response
 title('Bandpass filter: impulse response')
```

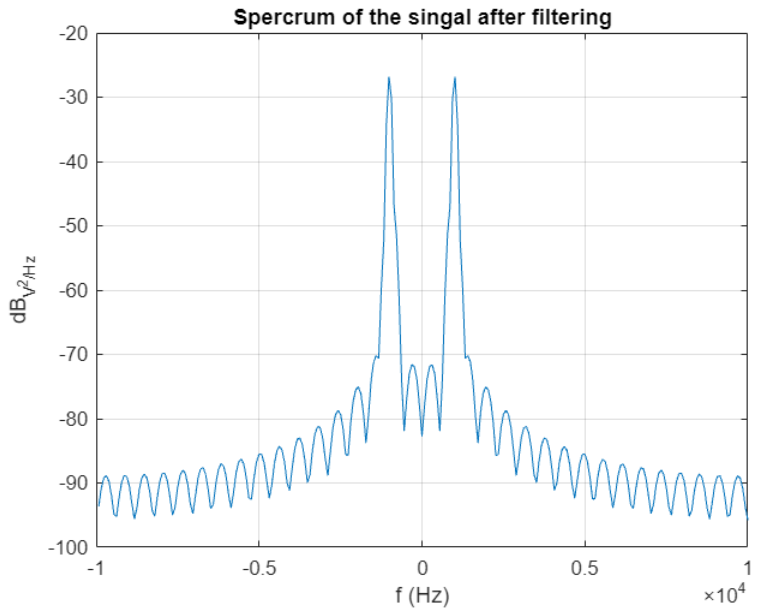



```
 freqz(h_bandpass,1,[],fs); %plot the frequency resionse
 title('Bandpass filter: frequency response')
```



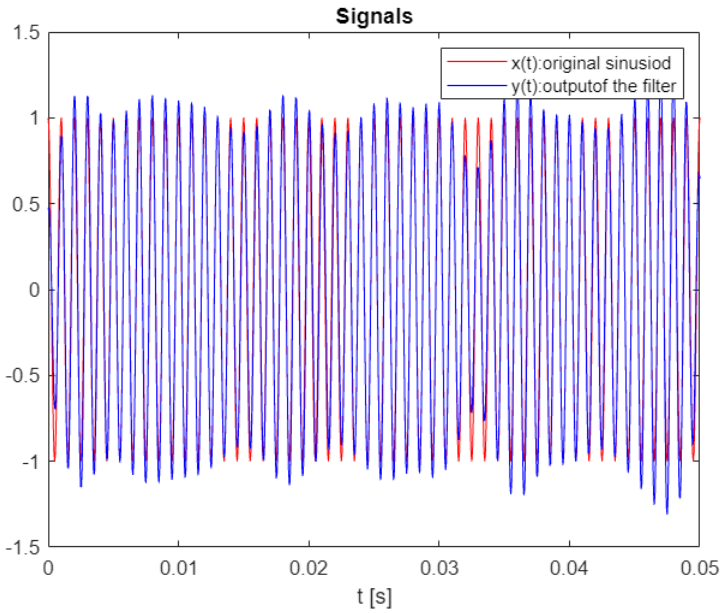
By replacing the low pass filter with a bandpass filter, it can be concluded that in the pass band the signal will pass, and signals below pass band or above will be truncated. Similar to low pass filter the effect of the filter on phase of signals is also linear.

```
%% *Filtering*
y=conv(x_noisy,h_bandpass,'same'); % filter the singal with the bandpass filter
%y=conv(x_noise,h_lowpass,'same'); % filter the singal with the lowpass filter

%% *Plots*
figure
PlotSpectrum(y,fs);
title('Spercrum of the singal after filtering');
```



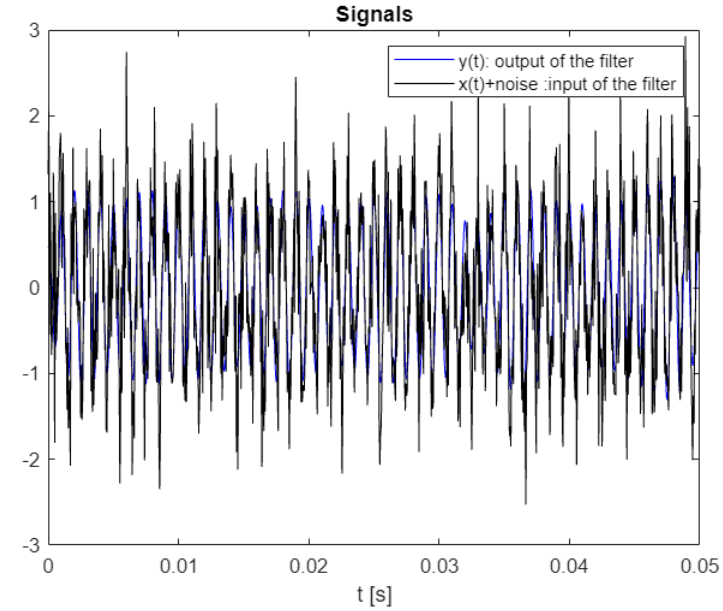
It can be concluded that after passing the filter, most of the signal energy is concentrated in the passband, and energy outside the passband is suppressed by filter.

```
figure
plot(t,x,'r')

hold on
plot(t,y,'b')
xlabel('t [s]')
legend('x(t):original sinusiod','y(t):outputof the filter')
title('Signals');
```



Comparing the signal passing through the filter and the original signal, their waveform are basically similar and do not affect the extraction of information.

```
figure
plot(t,y,'b')
hold on
plot(t,x_noisy,'k')
xlabel('t [s]')
legend('y(t): output of the filter','x(t)+noise :input of the filter')
title('Signals')
```



Filters can effectively filter out clutter and improve signal quality.

QAM modular

```
close all;
clear all;
clc;
```

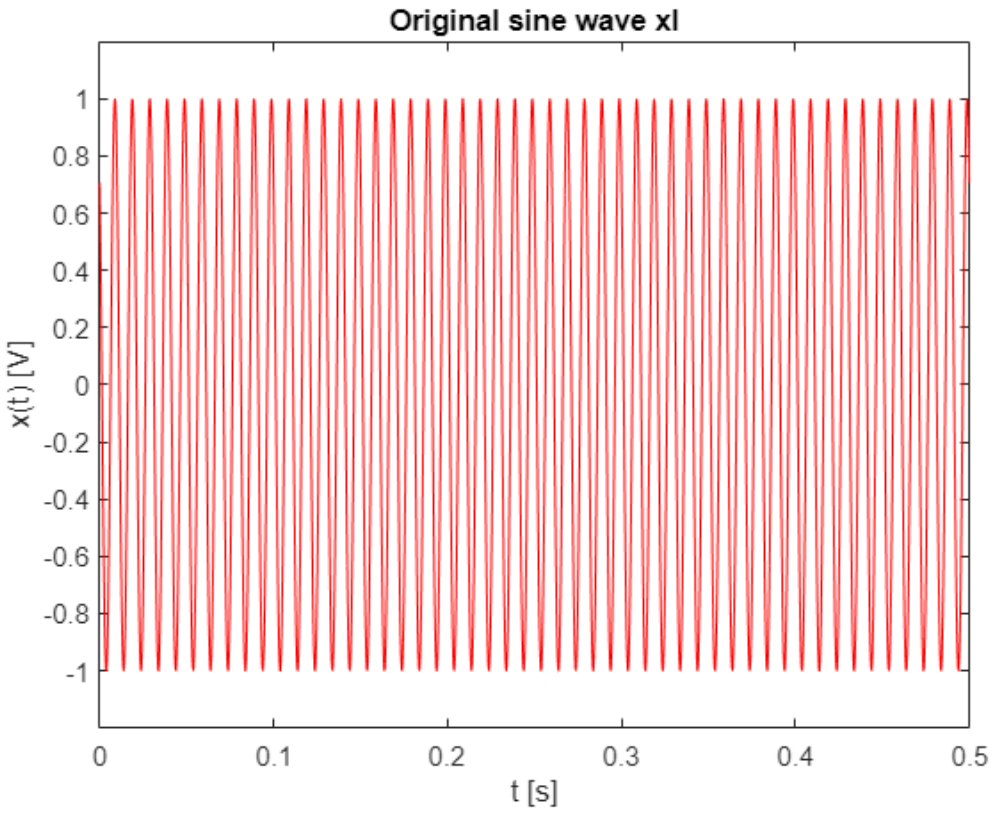
The following code segment define the basic parameter of QAM modular

```
fs=30000; % sampling frequency
A=1; % amplitude of the modulating in-phase and quadrature sine waves
T=0.5; % duration (sec) of the in-phase and quadrature sine waves
f1=100; % frequency of the modulating sine wave xI in the in-phase path
f2=300; % frequency of the modulating sine wave xQ in the quadrature path
fc=3000; % carrier frequency
```

generate (and plot) the modulating sine waves xI and xQ

The following code segment generate the modulating sine waves xI and xQ

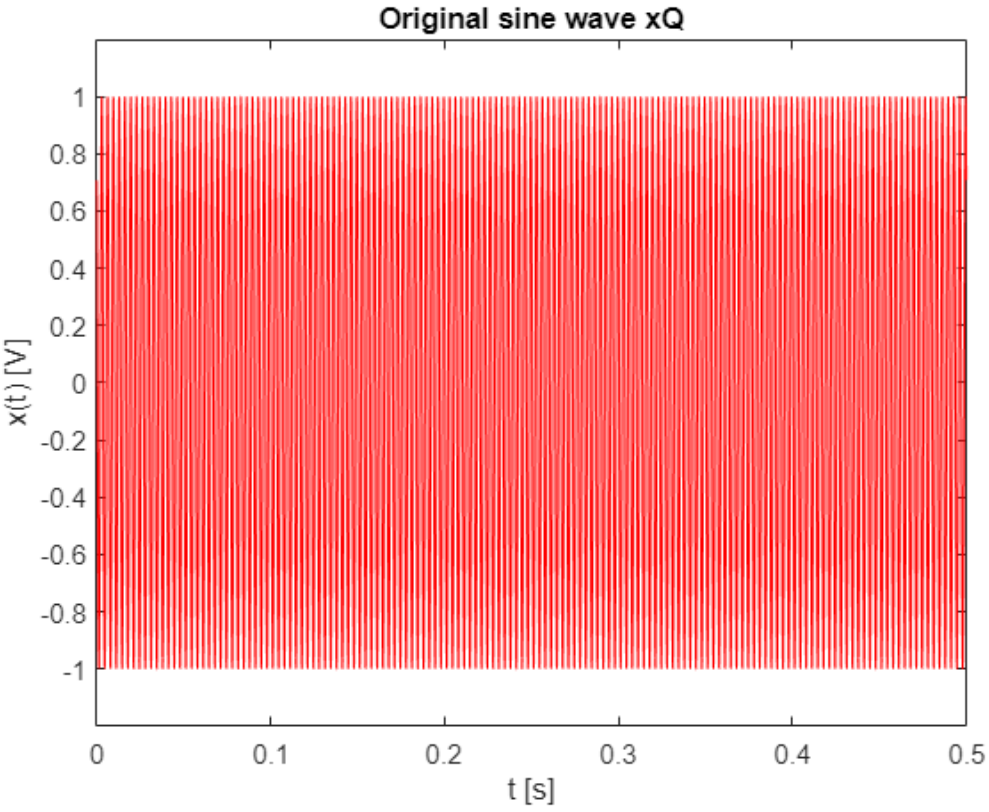
```
% generate (and plot) the modulating sine waves xI and xQ
% by means of the SinusoidalSource_2023 function
[t,xI,N]=SinusoidalSource_2023(A,f1,T,fs); % generation of the singal
figure
plot(t,xI,'r')
xlabel('t [s]')
ylabel('x(t) [V]')
title('Original sine wave xI')
axis([min(t) max(t) 1.2*min(xI) 1.2*max(xI)])
```




```
signal_power=0.5*A^2;
fprintf('XI power [V^2]=%f' , signal_power)
```

```
XI power [V^2]=0.500000
```

```
[t,xQ,N]=SinusoidalSource_2023(A,f2,T,fs); % generation of the singal
figure
plot(t,xQ,'r')
xlabel('t [s]')
ylabel('x(t) [V]')
title('Original sine wave xQ')
axis([min(t) max(t) 1.2*min(xQ) 1.2*max(xQ)])
```

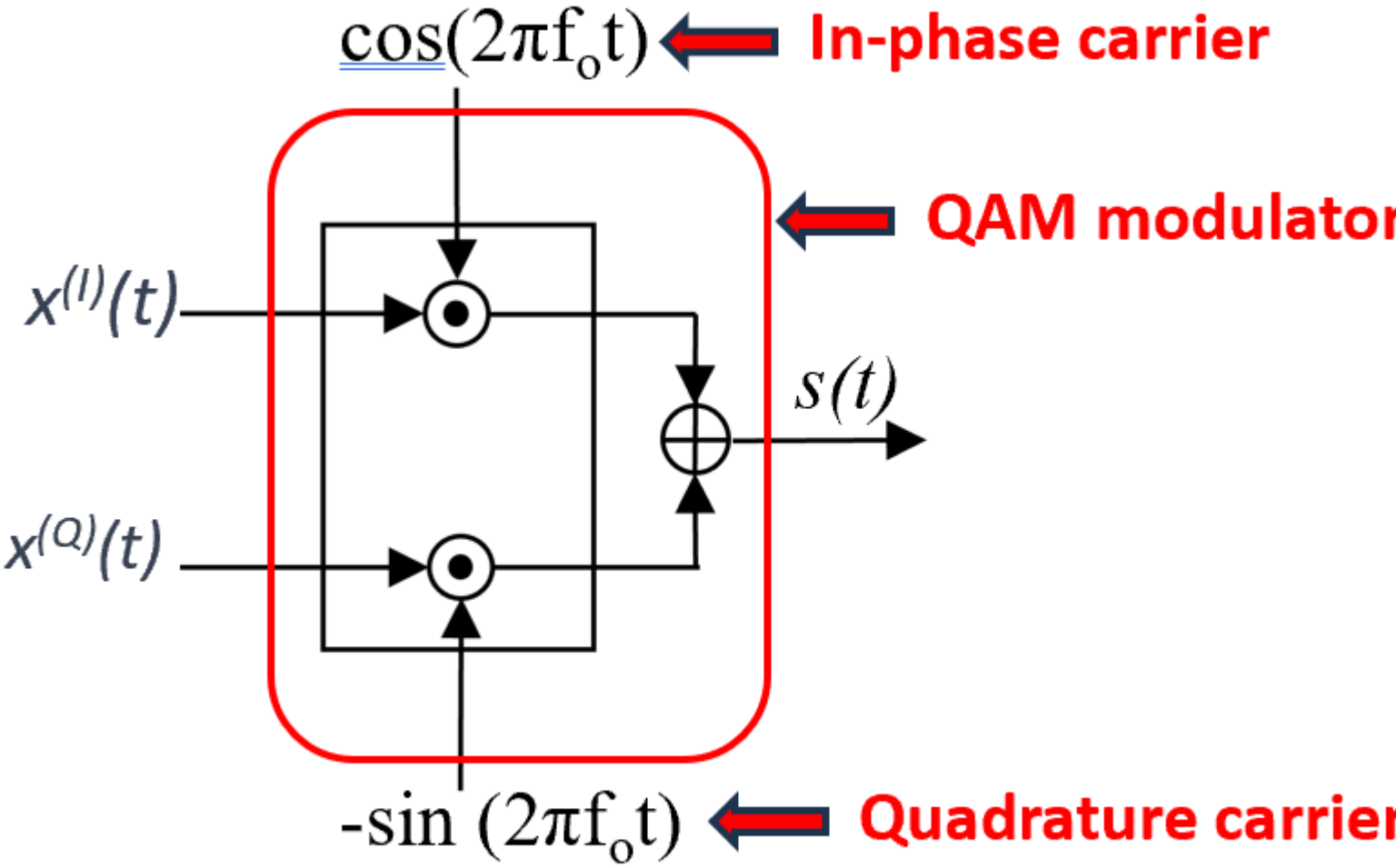



```
signal_power=0.5*A^2;
fprintf('sinusoid power [V^2]=%f' , signal_power)
```

```
sinusoid power [V^2]=0.500000
```

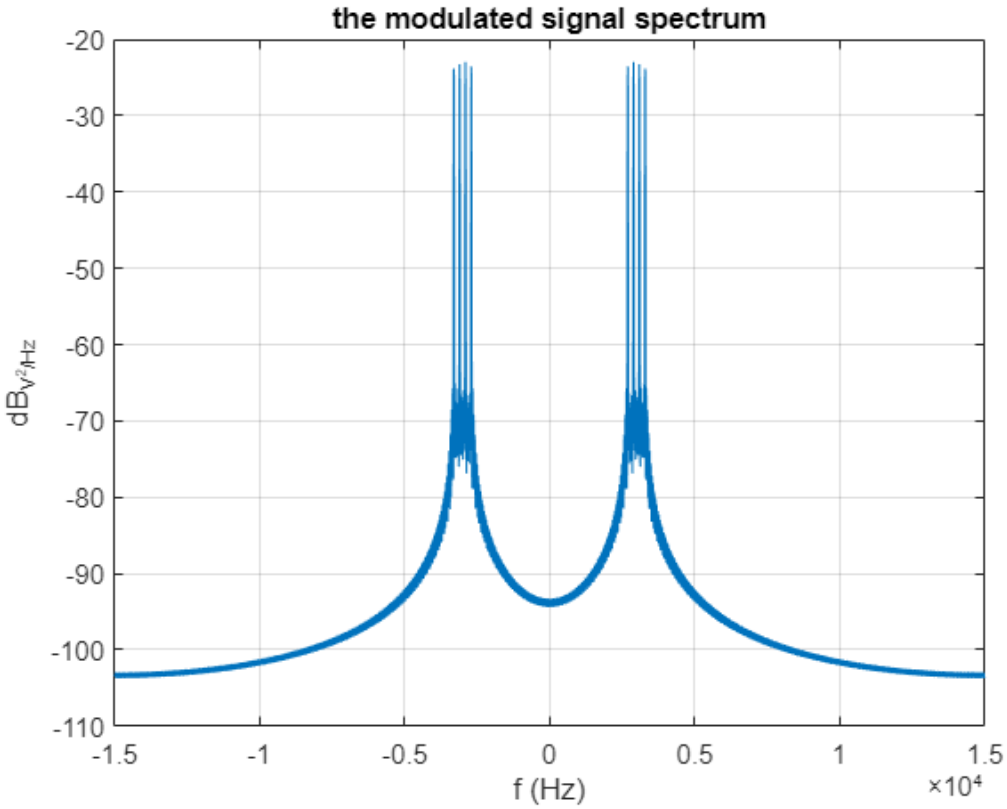
QAM modulation xI and xQ with carrier frequency

QAM modulator combines the multiplication of inputs signal and carrier to modulate signals with carrier frequecy..



The following code segment generate the QAM modulation signals carrying low-frequency signals to high frequencies and plotting the modulated signal spectrum.

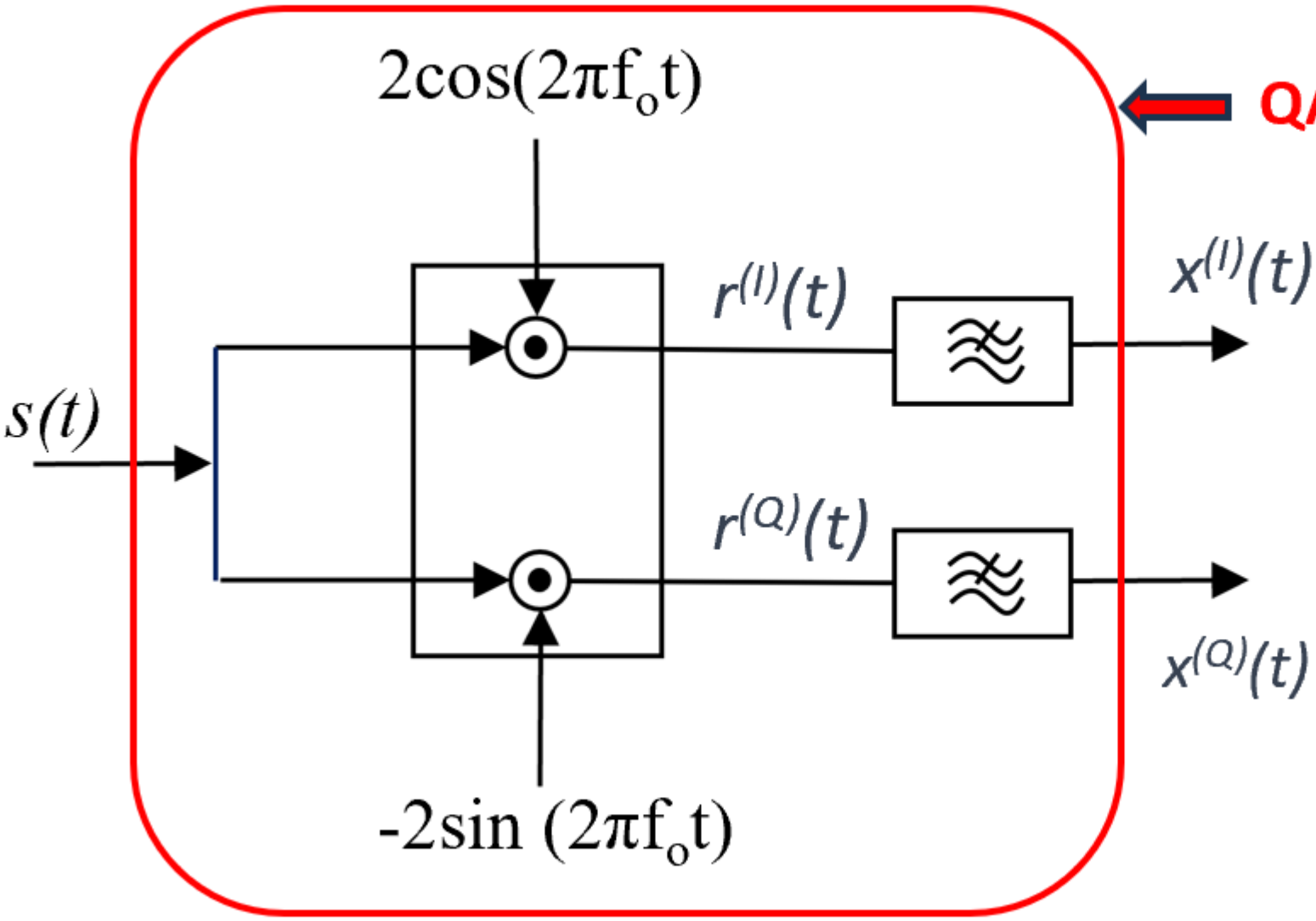
```
% QAM modulation xI and xQ with carrier frequency
% plot the modulated signal and its spectrum
% (use the function PlotSpectrum_2023(s,fs))
s = ModQAM_2023(xI,xQ,fc,T,fs);
figure
PlotSpectrum_2023(s,fs);
title('the modulated signal spectrum')
```



This pricure shows the QAM modulated signal sepctrum with input signals 'xI' , 'xQ', the carrier frequency is 3000hz. The signals pectrum is symmetries because the original signal is real signals.

QAM demodulation

QAM demodulator multiplying inputs signal and carrier, then Convolve of a low-pass filter to demodulate the original signals.



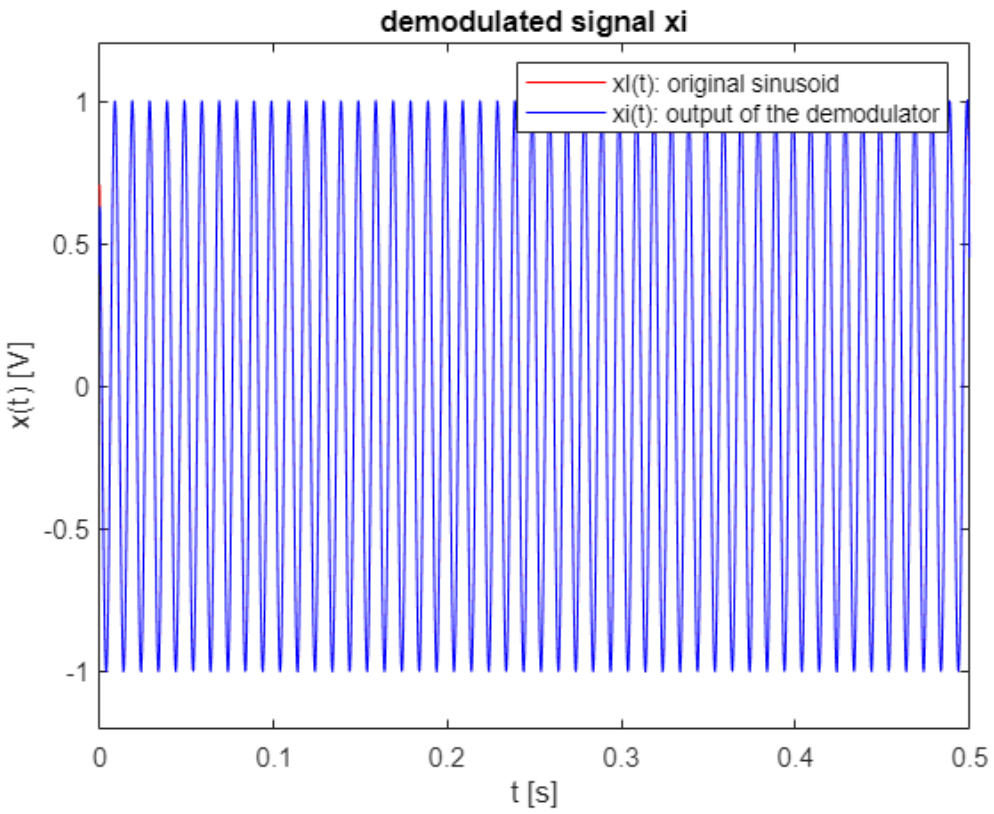
The following code segment demodulate signals from high frequency bands to lower frequencies and plotting.

```
% QAM demodulation
% plot the demodulated in-phase and quadrature signals and their spectra
[xi,xq,Delay] = DeModQAM_2023(s,fc,T,fs,0);
```

```
Xi = 1×15001
    1.4142    0.2771   -0.2310    0.4567    1.3290    1.2586    0.4246   -0.0852 ···
Xq = 1×15001
         0   -0.2013    0.7109    1.4055    0.9656   -0.0000   -0.3085    0.2623 ···
```

```
figure
plot(t,xI,'r')
hold on
plot(t,xi,'b')
legend ('xI(t): original sinusoid','xi(t): output of the demodulator')

xlabel('t [s]')
ylabel('x(t) [V]')
```

```
title('demodulated signal xi')
axis([min(t) max(t) 1.2*min(xi) 1.2*max(xi)])
```

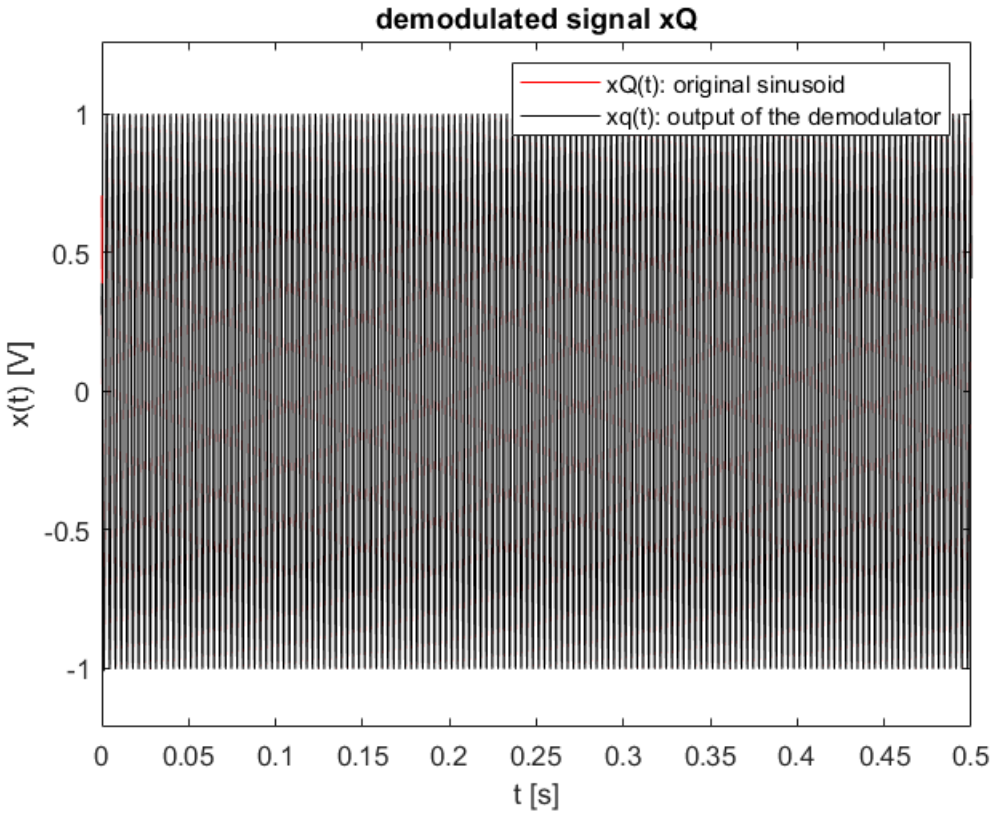


Output signal xI can be demodulated by demodulation function. It is as same as the original xI signal in amplitude, frequency and phase.

```
signal_power=0.5*A^2;
fprintf('sinusoid power [V^2]=%f' , signal_power)
```

```
sinusoid power [V^2]=0.500000
```

```
figure
plot(t,xQ,'r')
hold on
plot(t,xq,'k')
legend ('xQ(t): original sinusoid','xq(t): output of the demodulator')
xlabel('t [s]')
ylabel('x(t) [V]')
title('demodulated signal xQ')
axis([min(t) max(t) 1.2*min(xq) 1.2*max(xq)])
```

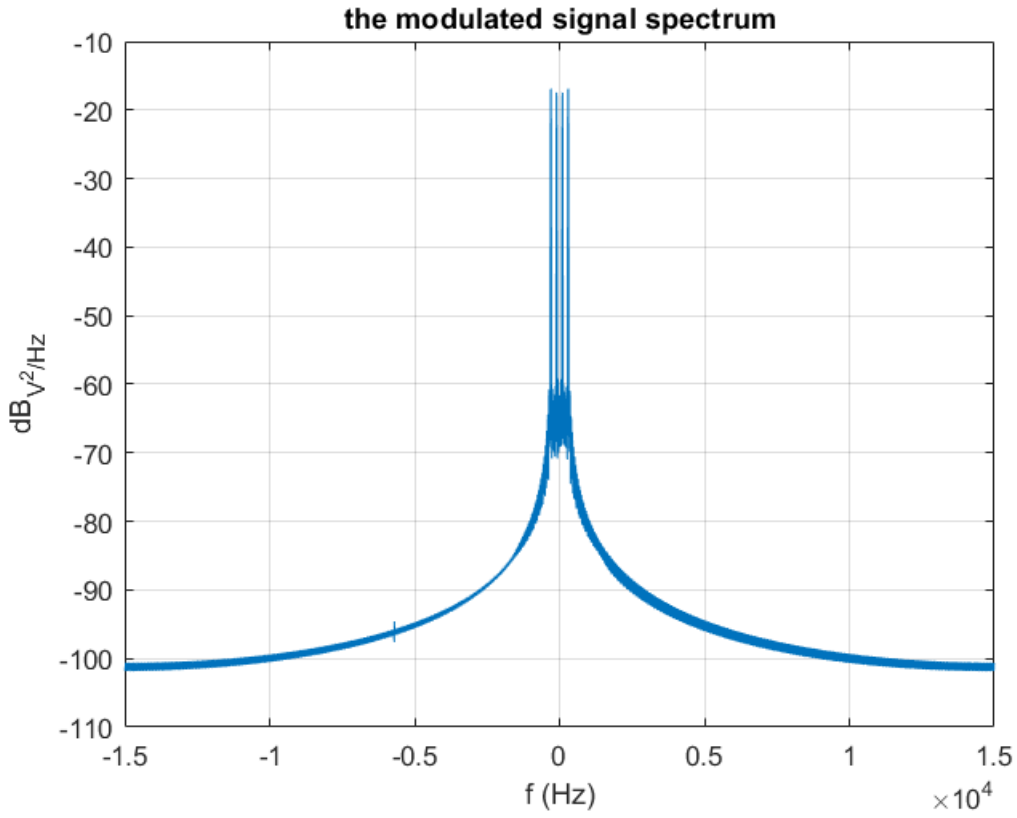


This picture gives us the information of the comparsion about original xq signal and demodulated signal xq. The conculsion is same with xI signals.

```
signal_power=0.5*A^2;
fprintf('sinusoid power [V^2]=%f' , signal_power)
```

```
sinusoid power [V^2]=0.500000
```

```
s_demod=xi+xq*j;
figure
PlotSpectrum_2023(s_demod,fs);
title('the modulated signal spectrum')
```



This picture gives the information about spectrum of demodulated signals s = xi+xq, comparing with the spectrum of modulated signals it shifted 3000hz in frequency domian. So the modulation is a process of moving signals from low frequency to high frequency for transmissoin by production a sin or cos function with carrier frequency. The demodulation is a process of moving signals from high frequency to low frequency for extracting information by convex opeartion.The whole process are linera time invariant and do not change any information.

Pluse Shaper

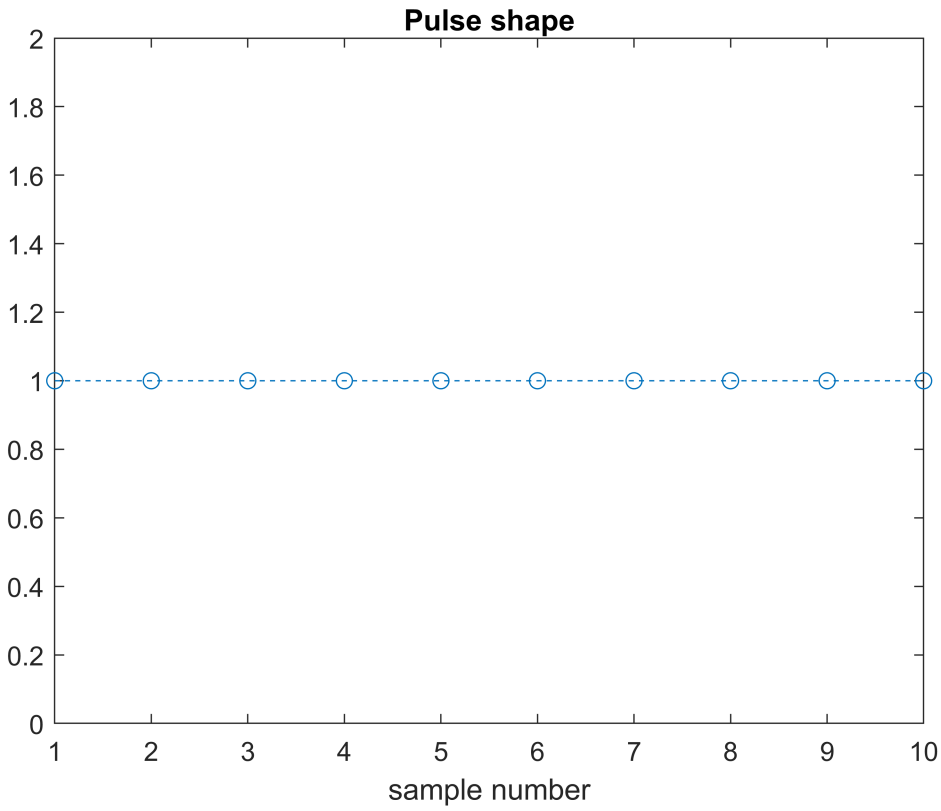
The following code segment defines a set of symbols representing complex numbers and initializes parameters for a pulse amplitude modulation (PAM) system, generating a modulated signal (x) and the corresponding pulse shape. Then plotting pulse shape and signals

```
clear all
symbols=[1+j -1-j -1+j 1-j -1-j 1+j 1+j -1-j 1+j -1+j -1+j 1-j 1+j -1-j 1+j 1-j];
Nf=200; % number of samples of the basic pulse
nsps=10; % number of samples of the PAM signal in a symbol time
rolloff=0.2; % rool-off factor for the root raised cosine pulse
pulsetype="RECT"; % choose the pulse type by uncommenting the corresponding line
```

PAMmodulator rectangular pluse

The following code segment chooses a rectangular pluse as a basic pluse, mapping from bits to symbols by convolution opeartion.

```
[x,h,Delay]=PAMmodulator_2023(symbols,nsps,Nf,pulsetype,rolloff);
plot(h,'--o')
xlabel('sample number')
title('Pulse shape')
```



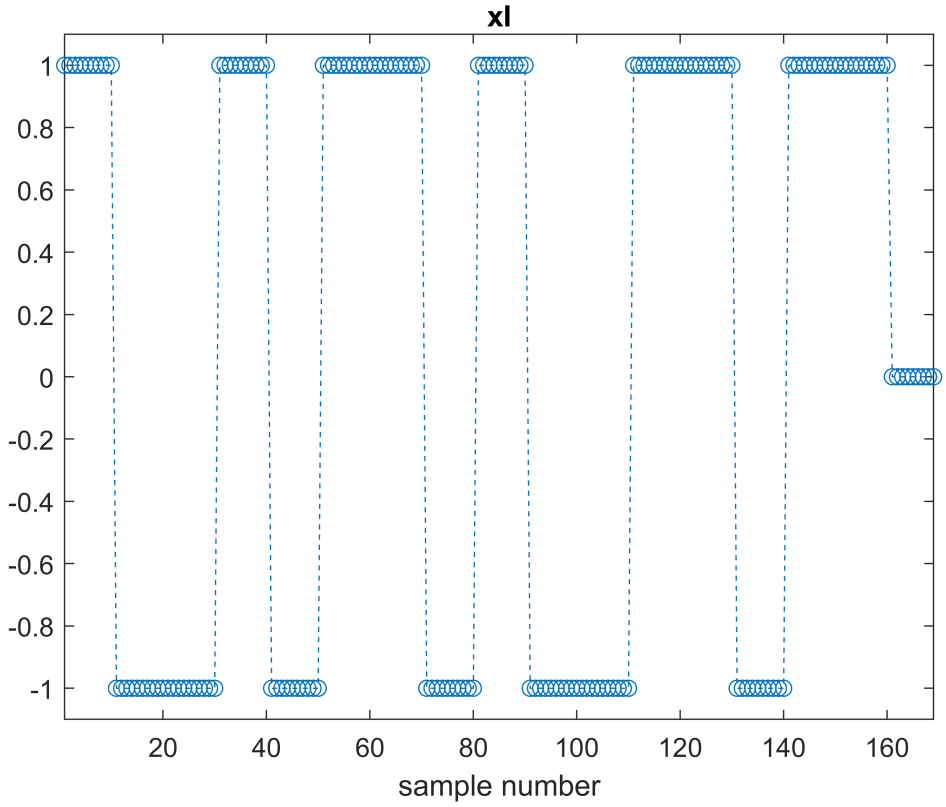
One time continuous waveform is represented by means of nsps=10 samplings with amplitude is 1.

```
figure
plot(real(x),'--o')
xlabel('sample number')
```

```
title('xI')
real(symbols)
```

```
ans = 1×16
     1    -1    -1     1    -1     1     1    -1     1    -1    -1     1     1 ···
```

```
axis([1 length(x) 1.1*min(real(x)) 1.1*max(real(x))])
```

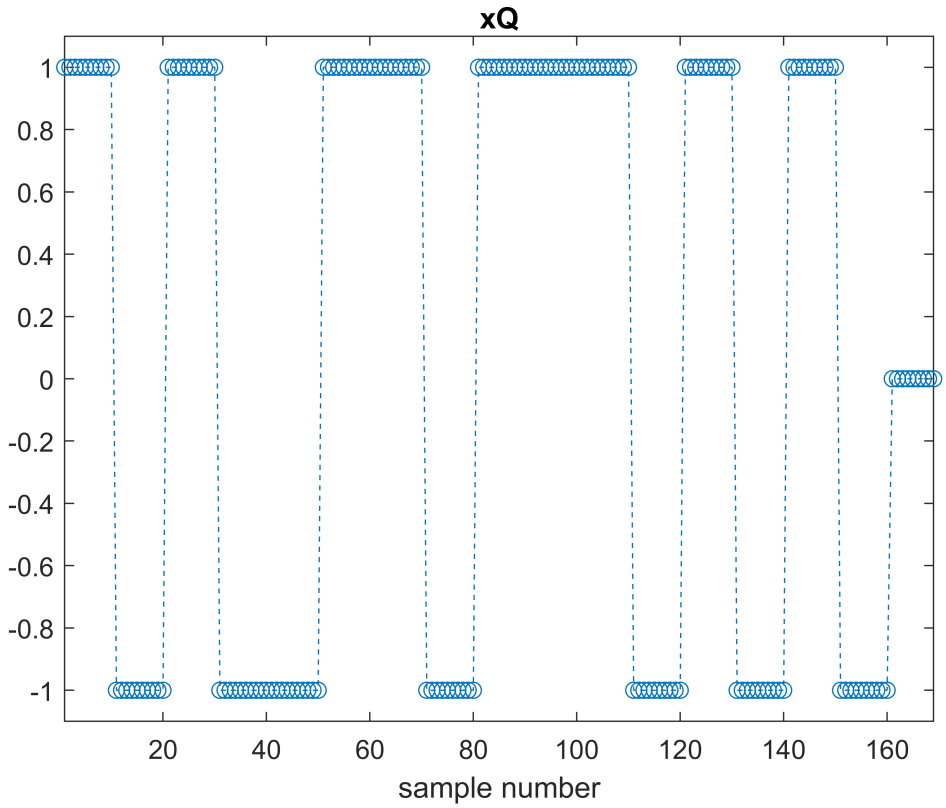


This picture shows the result of PAM signal with rectangular pulses corresopend to in phase components of signals.

```
figure
plot(imag(x),'--o')
xlabel('sample number')
title('xQ')
imag(symbols)
```

```
ans = 1×16
     1    -1     1    -1    -1     1     1    -1     1     1     1    -1     1 ···
```

```
axis([1 length(x) 1.1*min(imag(x)) 1.1*max(imag(x))])
```

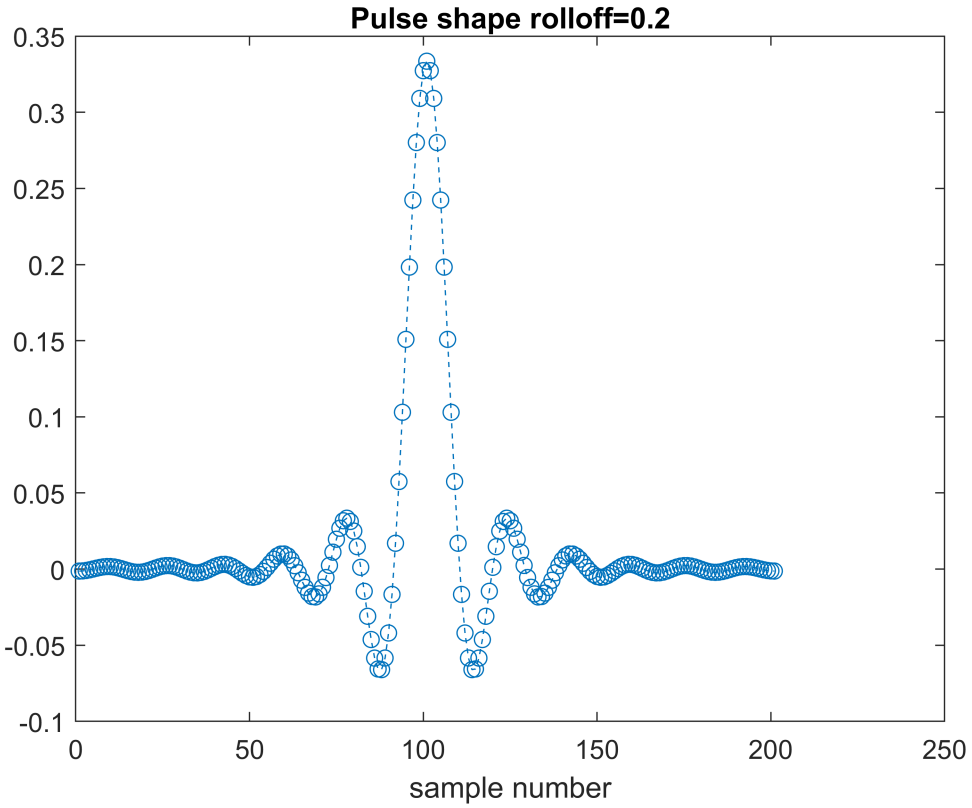


This picture shows the result of PAM signal with rectangular pulses corresopend to in *quadrature* components of signals.

PAMmodulator (rootraisedcosine pluse)

Similar to the previous code, the following code chooses rootraisdconsine pluse shape as basic pluse and compared with different rolloff.

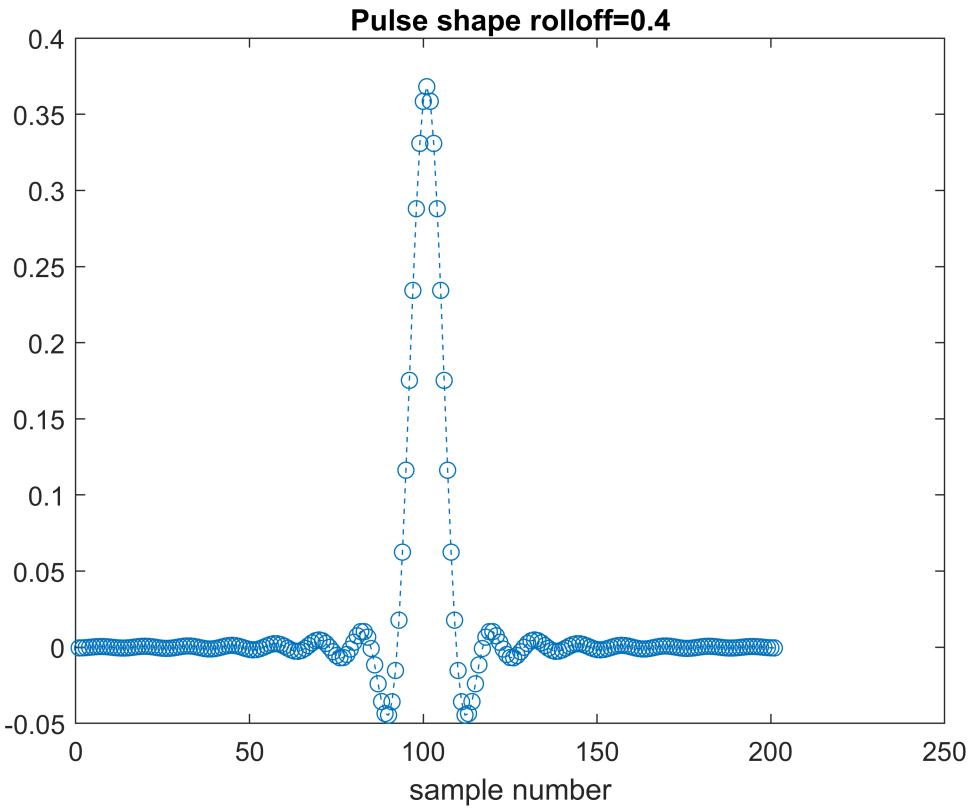
```
pulsetype="ROOTRAISEDCOSINE"; % choose the pulse type by uncommenting the corresponding line
[x,h,Delay]=PAMmodulator_2023(symbols,nsps,Nf,pulsetype,rolloff);
[x1,h1,Delay]=PAMmodulator_2023(symbols,nsps,Nf,pulsetype,rolloff*3);
[x2,h2,Delay]=PAMmodulator_2023(symbols,nsps,Nf,pulsetype,rolloff*5);
plot(h,'--o')
xlabel('sample number')
title('Pulse shape rolloff=0.2')
```



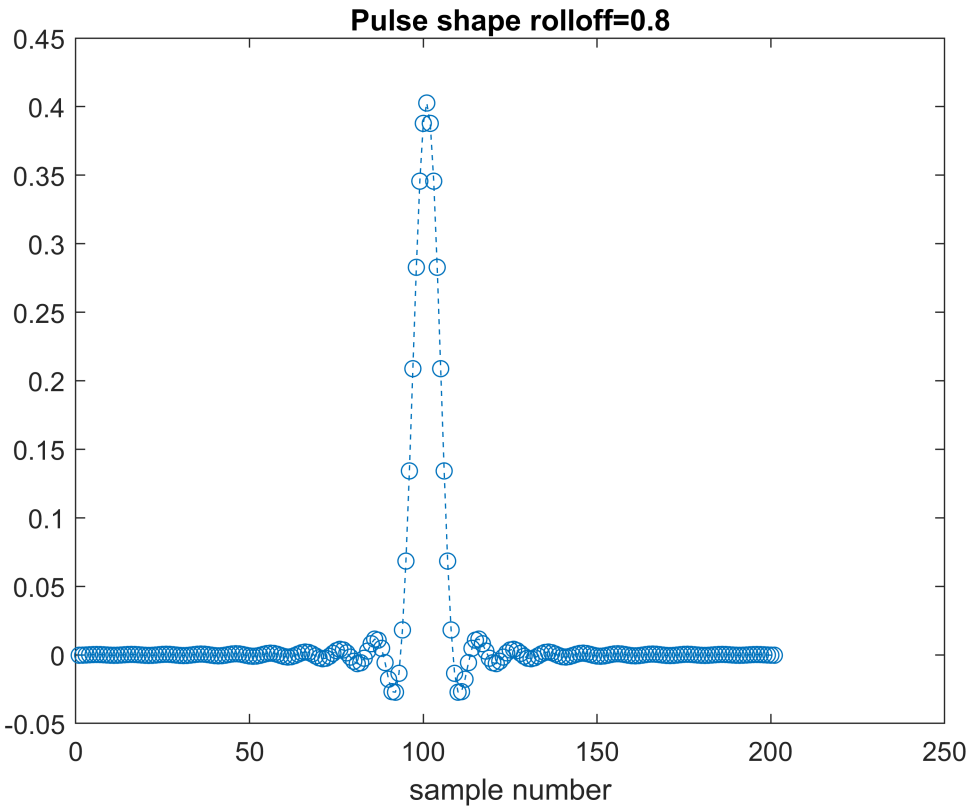

```
real(symbols)
```

```
ans = 1×16
     1    -1    -1     1    -1     1     1    -1     1    -1    -1     1     1 ···
```

```
plot(h1,'--o')
xlabel('sample number')
title('Pulse shape rolloff=0.4')
```




```
plot(h2,'--o')
xlabel('sample number')
title('Pulse shape rolloff=0.8')
```

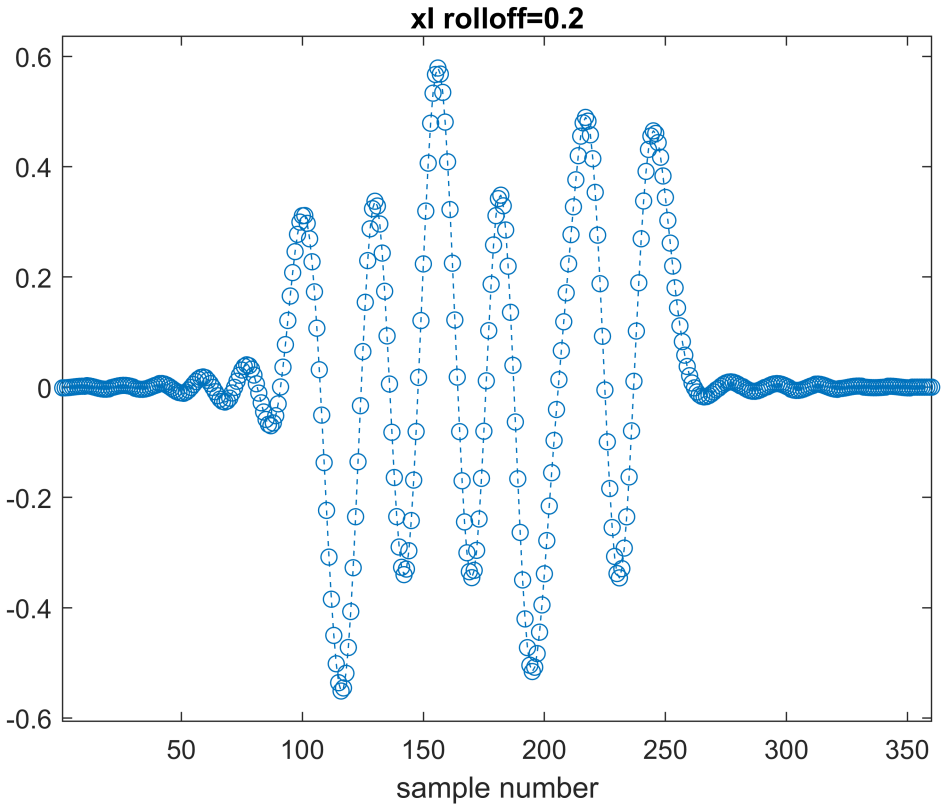


Comparing to three different rolloff It can be concluded the larger rolloff the larger, the greater the amplitude of the signal and the faster the attenuation,

```
figure
plot(real(x),'--o')
xlabel('sample number')
title('xI rolloff=0.2')
real(symbols)
```

```
ans = 1×16
     1    -1    -1     1    -1     1     1    -1     1    -1    -1     1     1 ···
```

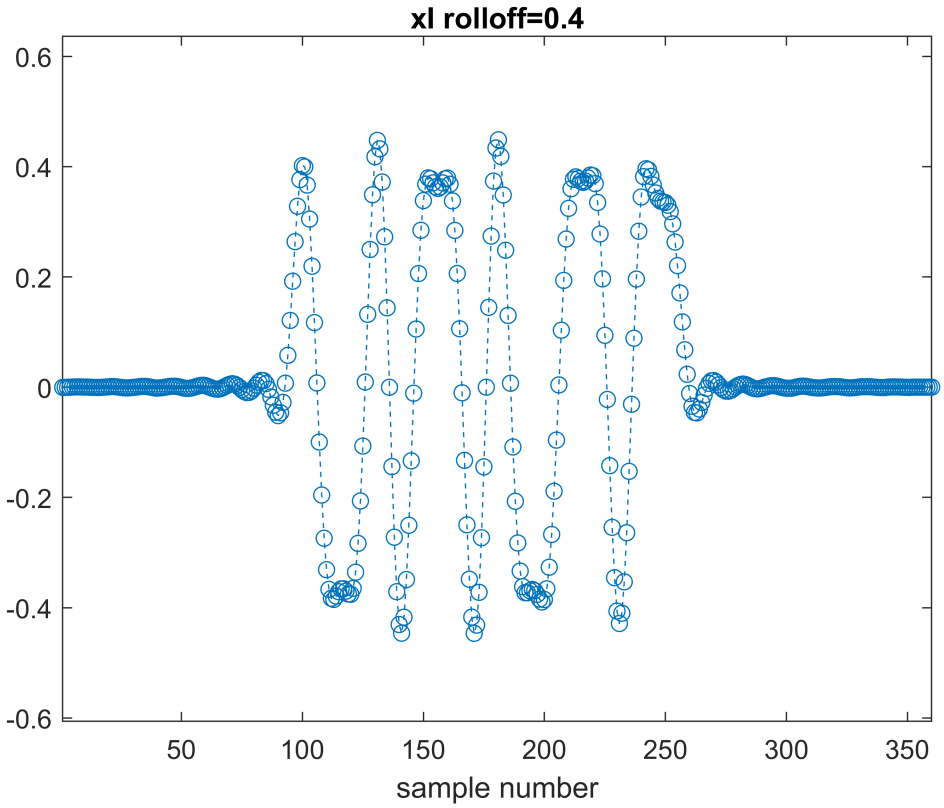
```
axis([1 length(x) 1.1*min(real(x)) 1.1*max(real(x))])
```




```
figure
plot(real(x1),'--o')
xlabel('sample number')
title('xI rolloff=0.4')
real(symbols)
```

```
ans = 1×16
     1    -1    -1     1    -1     1     1    -1     1    -1    -1     1     1 ···
```

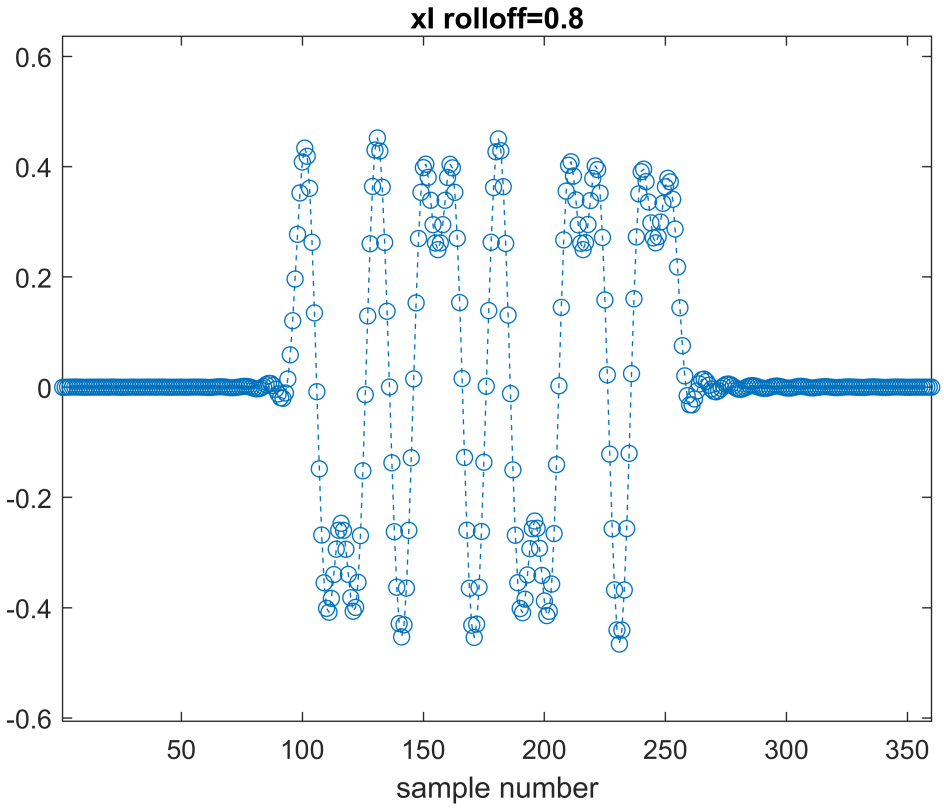
```
axis([1 length(x) 1.1*min(real(x)) 1.1*max(real(x))])
```




```
figure
plot(real(x2),'--o')
xlabel('sample number')
title('xI rolloff=0.8')
real(symbols)
```

```
ans = 1×16
     1    -1    -1     1    -1     1     1    -1     1    -1    -1     1     1 ···
```

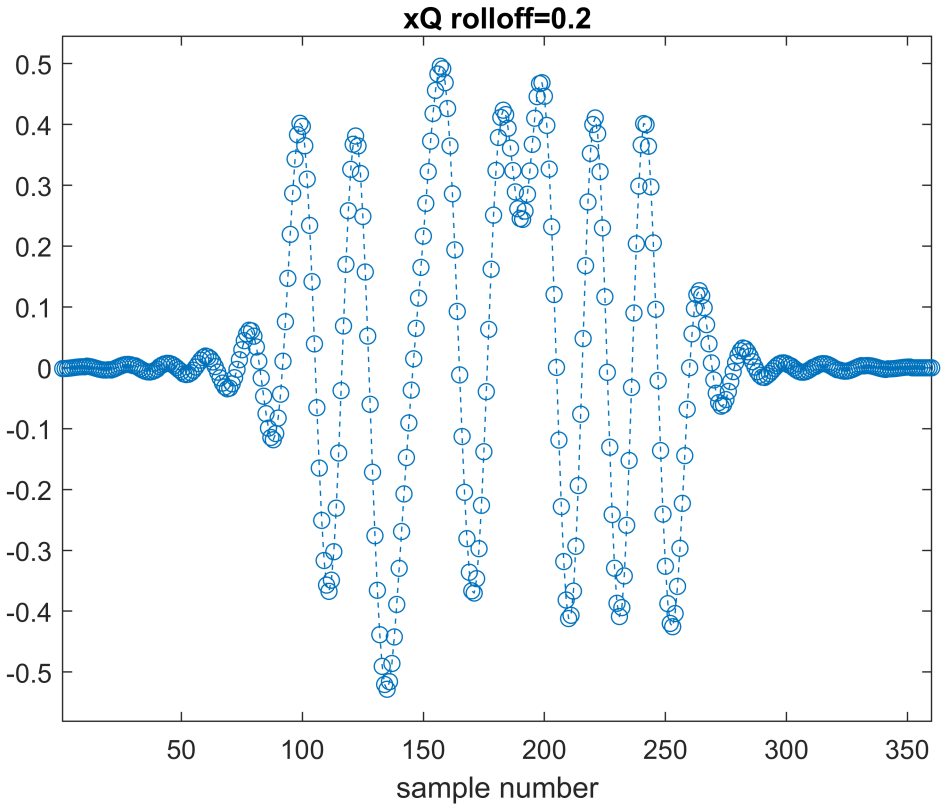
```
axis([1 length(x) 1.1*min(real(x)) 1.1*max(real(x))])
```




```
figure
plot(imag(x),'--o')
xlabel('sample number')
title('xQ rolloff=0.2')
imag(symbols)
```

```
ans = 1×16
     1    -1     1    -1    -1     1     1    -1     1     1     1    -1     1 ···
```

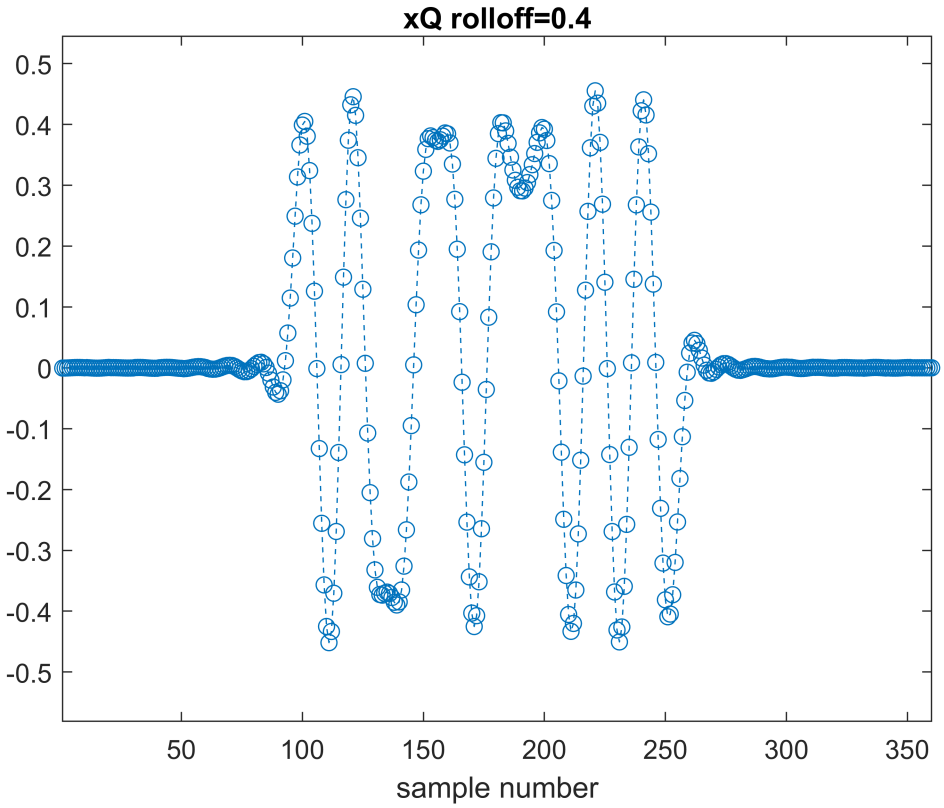
```
axis([1 length(x) 1.1*min(imag(x)) 1.1*max(imag(x))])
```




```
figure
plot(imag(x1),'--o')
xlabel('sample number')
title('xQ rolloff=0.4')
imag(symbols)
```

```
ans = 1×16
     1    -1     1    -1    -1     1     1    -1     1     1     1    -1     1 ···
```

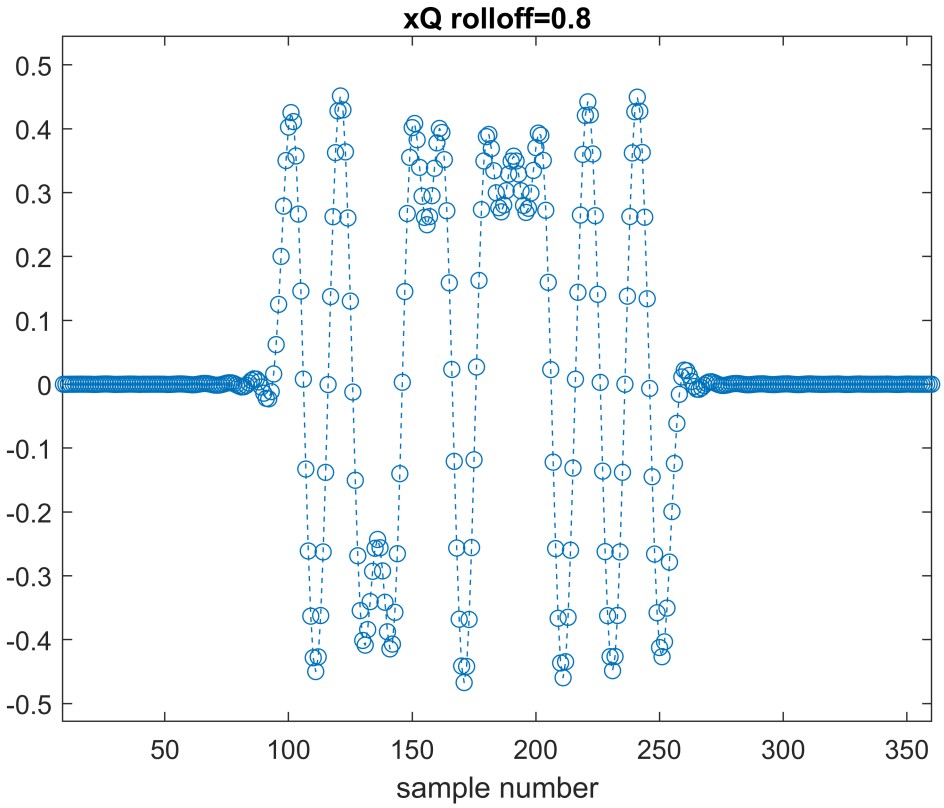
```
axis([1 length(x) 1.1*min(imag(x)) 1.1*max(imag(x))])
```




```
figure
plot(imag(x2),'--o')
xlabel('sample number')
title('xQ rolloff=0.8')
imag(symbols)
```

```
ans = 1×16
     1    -1     1    -1    -1     1     1    -1     1     1     1    -1     1 ···
```

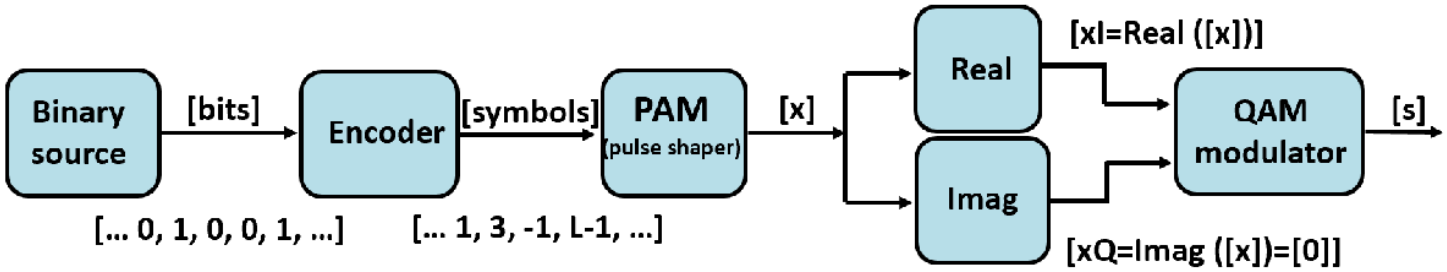
```
axis([1 length(x) 1.1*min(imag(x)) 1.1*max(imag(x))])
```



Comparing to symbols sequence with different rolloff, It can be concluded the larger rolloff, the more the waveform tends to be in a rectangular pluse, and the smaller rolloff, the more waveform tends to be in a sinc pluse.

Amplititude-shift keying trasmitter

```
clear all % clear all varables
close all % close all plots
clc % clear the screen in the command window
warning('off','all')
```



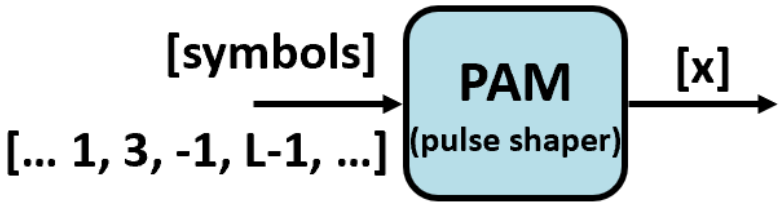
ASK is an amplitude modulation that tranform digital data to fixed-amplitude carrier wave at fixed frequency.The picture above shows the inplemented system process. Binary source is encoded form bits to symbols by encouder and shaped into viarous amplitude pluse by PAM. L-ASK singals do not have any in quadrature components, So take the output of PAM and feed it into the QAM in-phase signal input of the QAM modulation system.This is the inplemented modulation system.

Setting signal specifications

```
%------------------------------------------------------------------
% *Signal specifications*
bits_per_symbol=4; % number of bits that correspond to a symbol
L=2^bits_per_symbol; % number of levels of the modulation alphabet
nbits=1000*bits_per_symbol; % number of bits to be transmitted
Br=1000; % [bits/s] bit rate
fs=50000; % [samples/s] sampling frequency
fc=3000; % [Hz] carrier frequency
Bs=Br/log2(L); % [symbols/s] Symbol rate
nsps=fs/Bs; % samples per symbol
%------------------------------------------------------------------
```

The Pluse shaper

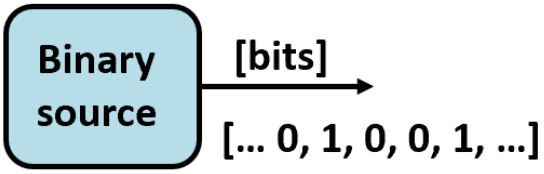
we have two different plusetypes **ROOTRAISEDCOSINE** and **RECT**




```
% *Pulse shaper specifications*
Nf=5*nsps; % number of filter coefficients (=five time the symbol duration)
pulsetype='RECT'; % pulsetype: 'RECT' or 'ROOTRAISEDCOSINE'
switch pulsetype
```

```
case 'ROOTRAISEDCOSINE'
rolloff=0.8; % roll-off factor for root raised cosine pulses
bandwidth_Hz=0.5*Bs*(1+rolloff) % bandwidth of the baseband signal
case 'RECT'
rolloff=[];
bandwidth_first_lobe_Hz=Bs;
end
```

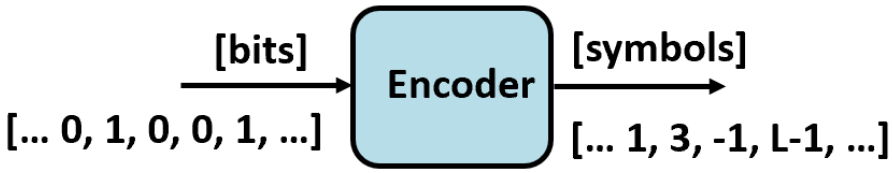
Binary source



The function BinarySource_2023() using Matlab function randi generate 1 by nbits matrix to describe the binary source

```
%------------------------------------------------------------------
% *Source bits generation*
source_bits=BinarySource_2023(nbits); % vector with the source bits
```

Encoder



The Encoder_2023() function given as input a victior of bit source bits, generate as output a victior of symbols using the multi-level symmetric alphabet {-1,1,-3,3... -L+1,L-1} of L=nlevels levels(L=2,4,8),according the Gray rule.

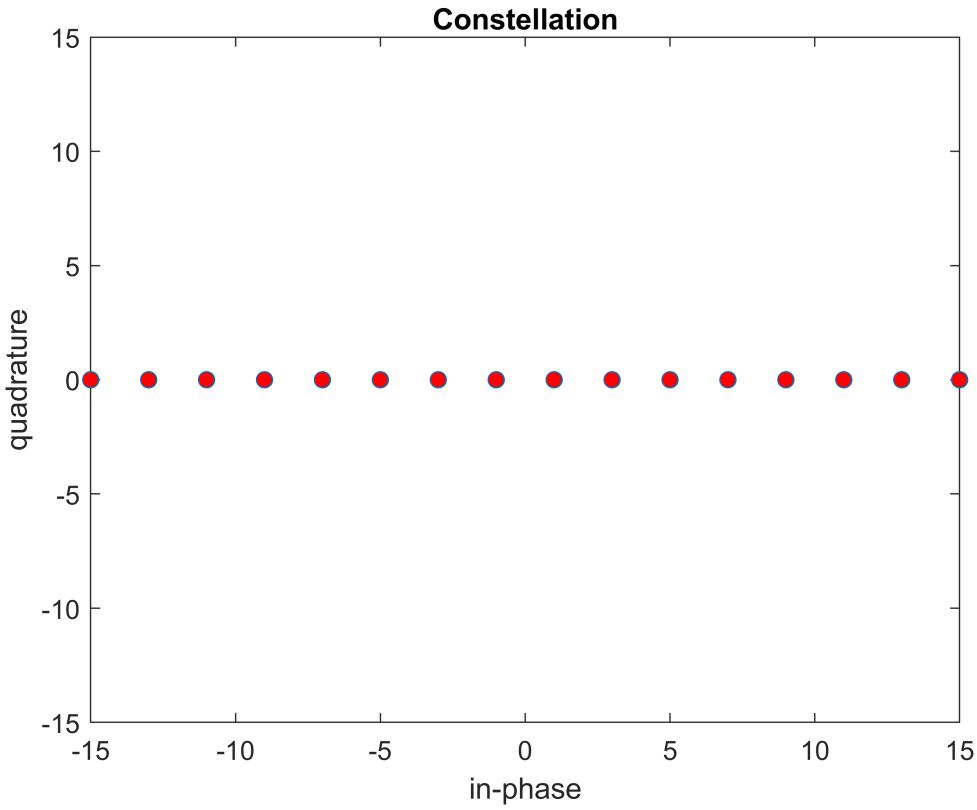
```
%------------------------------------------------------------------
% *Symbols generation with Gray mapping*
symbols=Encoder_2023(source_bits, L);
```

```
bit_table = 16×4
     0     0     0     0
     0     0     0     1
     0     0     1     1
     0     0     1     0
     0     1     1     0
     0     1     1     1
     0     1     0     1
     0     1     0     0
     1     1     0     0
     1     1     0     1
     ⋮

symbol_table = 16×1
   -15
```

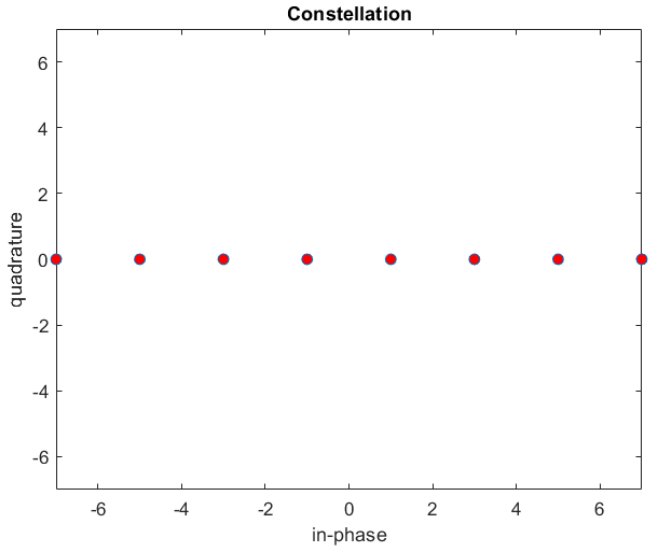
```
-13
-11
 -9
 -7
 -5
 -3
 -1
  1
  3
  ⋮
```

Plot of the constellation

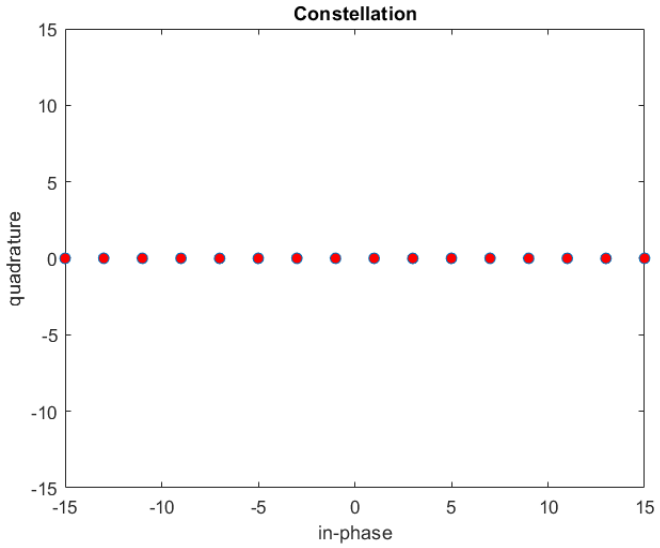
```
%-----------------------------------------------------------------
% *Plot of the constellation*
figure
plot(real(symbols),imag(symbols),'o','MarkerFaceColor','r'),
axis ([-L+1 L-1 -L+1 L-1]);
title('Constellation');
xlabel('in-phase');
ylabel('quadrature');
```



This picture is plotteed with parameter L = 4, ROOTRAISEDCOSINE pluse or RECT pluse, rolloff=0.8 or 0.2



This picture is plotteed with parameter L = 8, ROOTRAISEDCOSINE pluse or RECT pluse, rolloff=0.8 or 0.2



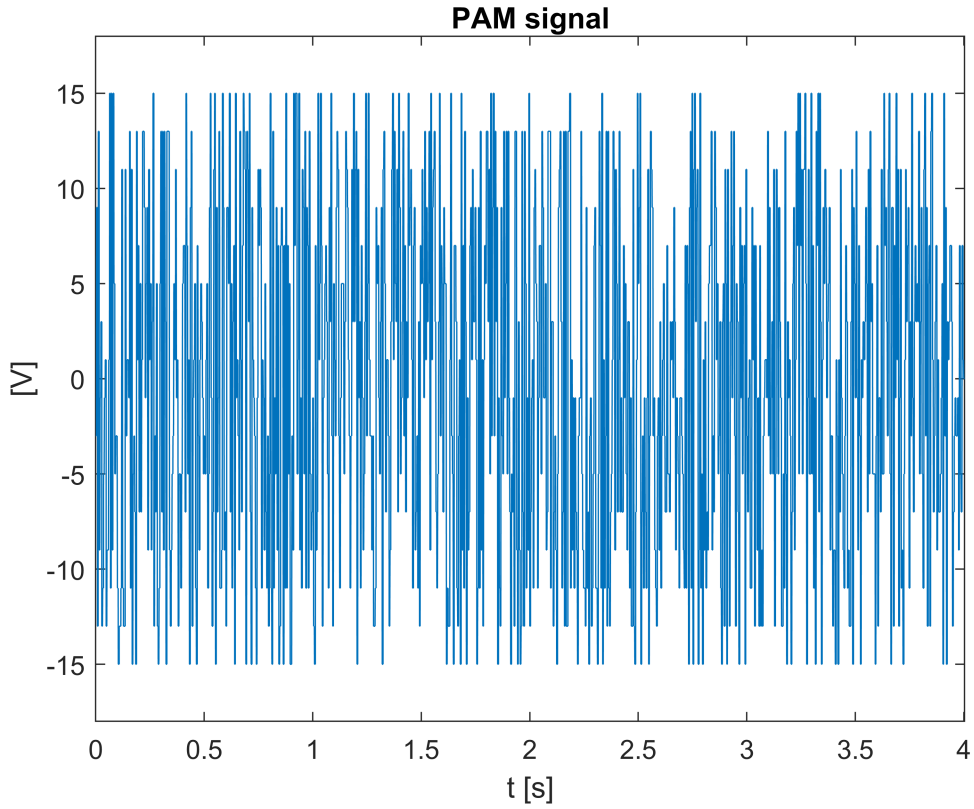
plotteed with parameter L = 16, ROOTRAISEDCOSINE pluse or RECT pluse, rolloff=0.8 or 0.2

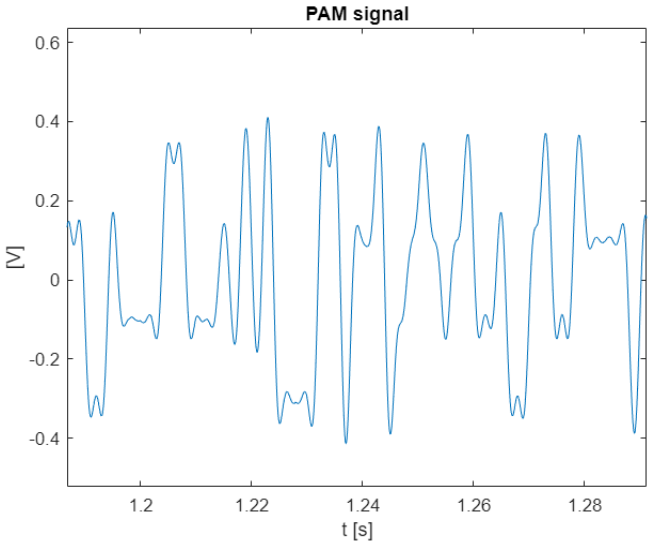
These pictures above show the constellation of QAM signals, with x-axis representing in phase signals and y-axis representing quadrature signals, which leads to the conclusion that L-ASK signals do not have any quadrature components.It can be concluded the constellation demostrates the change of L.

Generation of the baseband PAM signal

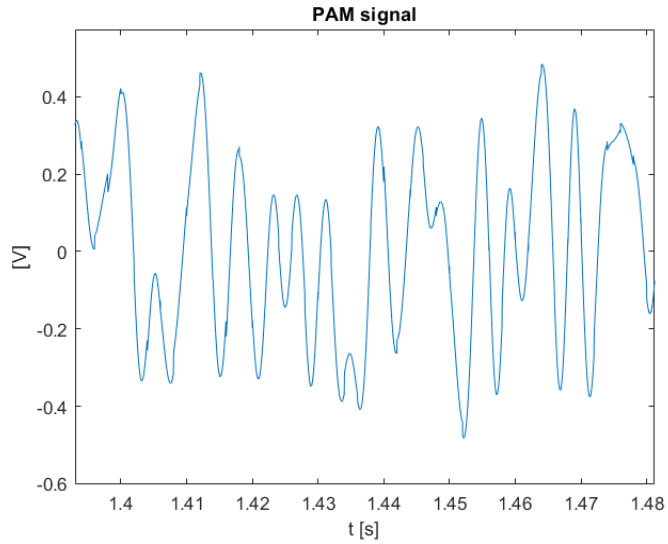
```
%------------------------------------------------------------------
% *Generation of the baseband PAM signal*
x=PAMmodulator_2023(symbols,nsps,Nf,pulsetype,rolloff);
Ts=1/fs;% sampling interval
T=Ts*(length(x)-1); % signal duration
%------------------------------------------------------------------
% *Plot of the baseband PAM signal and of its spectrum*
```

```
figure
t=0:1/fs:T;
plot(t,x)
xlabel('t [s]')
ylabel ('[V]')
title('PAM signal')
axis([min(t) max(t) 1.2*min(x) 1.2*max(x)])
```

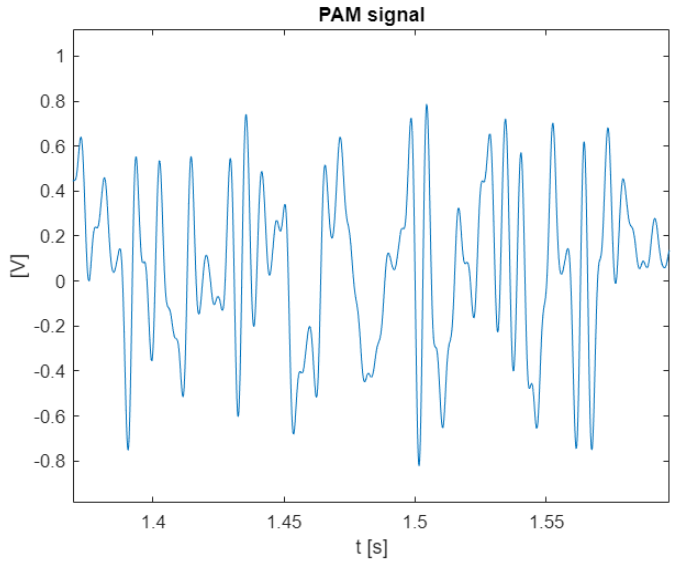




This picture is plotteed with parameter L = 4, ROOTRAISEDCOSINE pluse , rolloff=0.8

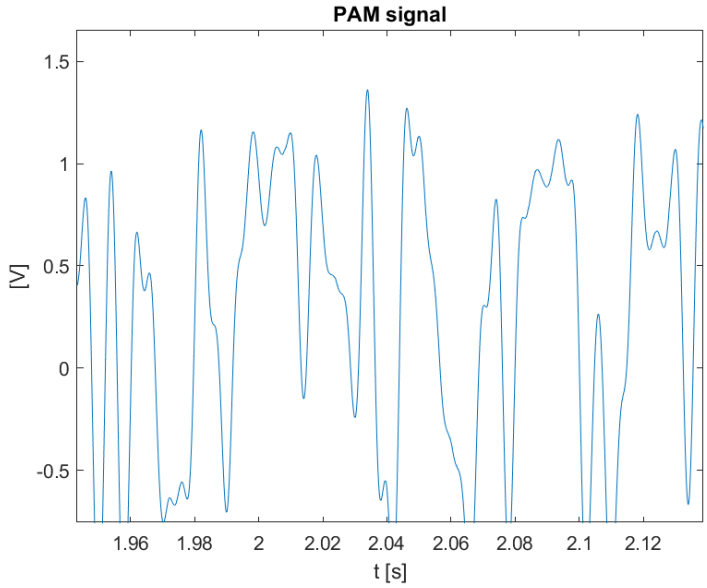


This picture is plotteed with parameter L = 4 ROOTRAISEDCOSINE pluse, rolloff=0.2

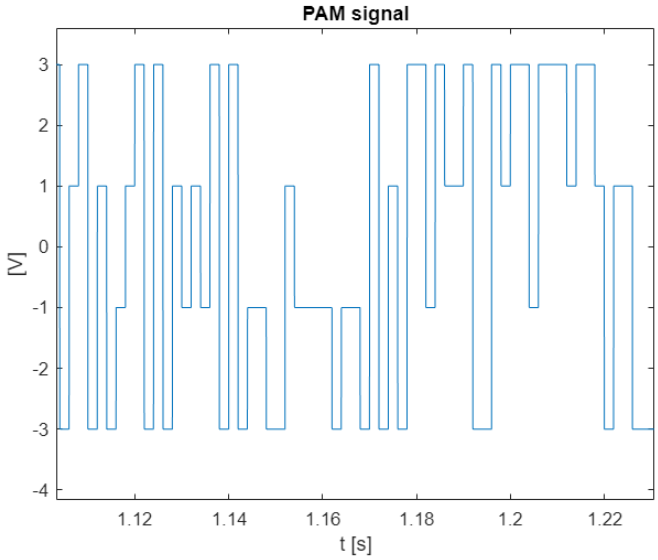


This picture is plotteed with parameter L = 8, ROOTRAISEDCOSINE pluse , rolloff=0.8

l=8 root 0.8



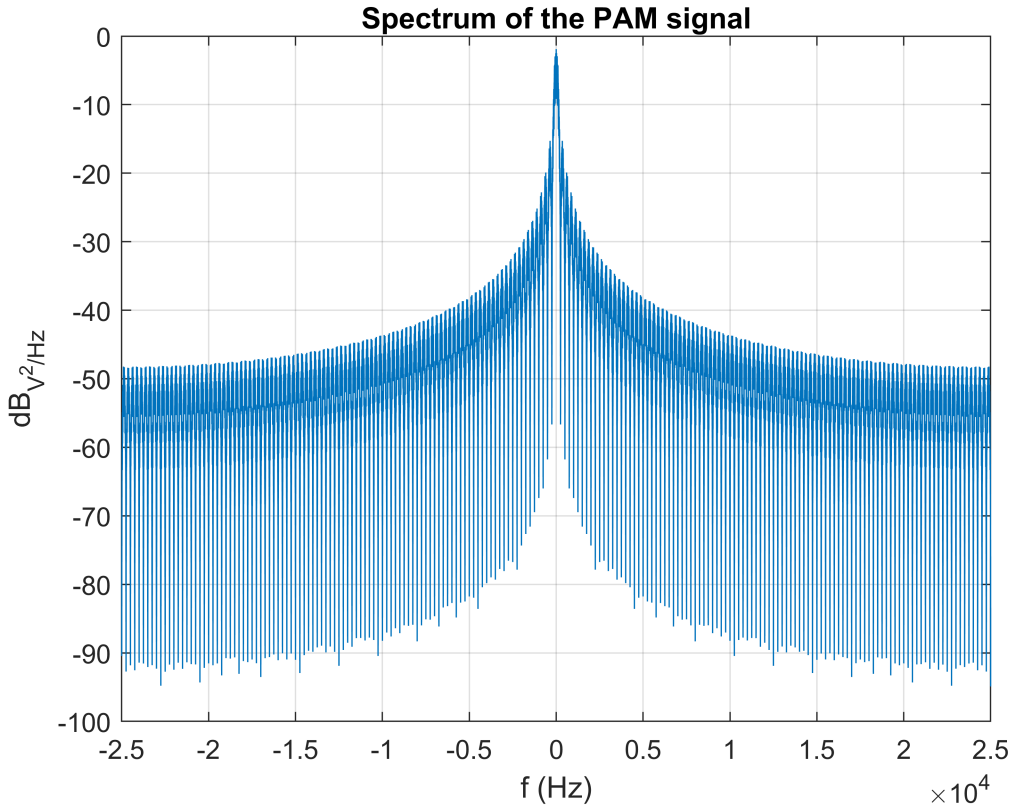
This picture is plotteed with parameter L = 16, ROOTRAISEDCOSINE pluse , rolloff=0.8

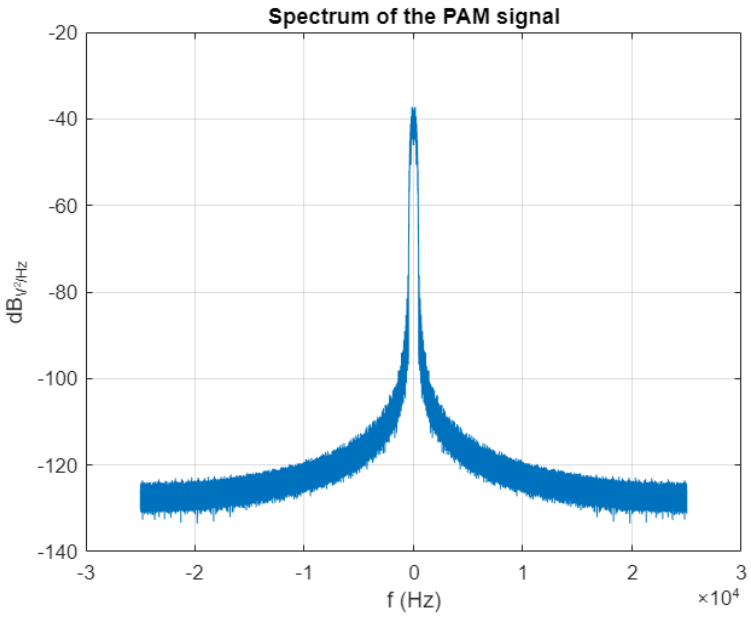


This picture is plotteed with parameter L = 4, RECT pluse

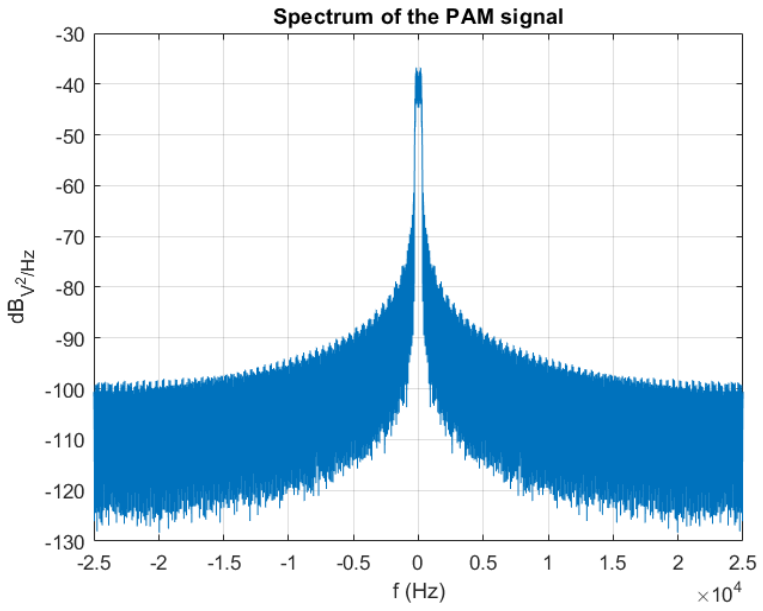
These picture illustrated the symbols sequences with different parameters L=4,8,16, RECT or ROOTRAISEDCOSINE pluse shape and different values of the roll-off factor (0< α < 1) for the ROOTRAISEDCOSINE pulse. In the case L parameter is different and the other parameters are the same, the larger value L is, the larger magnitude of a symbol is. In the case rolloff parameter is different and the other parameters are the same, the larger value rolloff is, the more pluse shape tends to rectgular.

```
figure
PlotSpectrum_2023(x,fs);
title('Spectrum of the PAM signal')
```

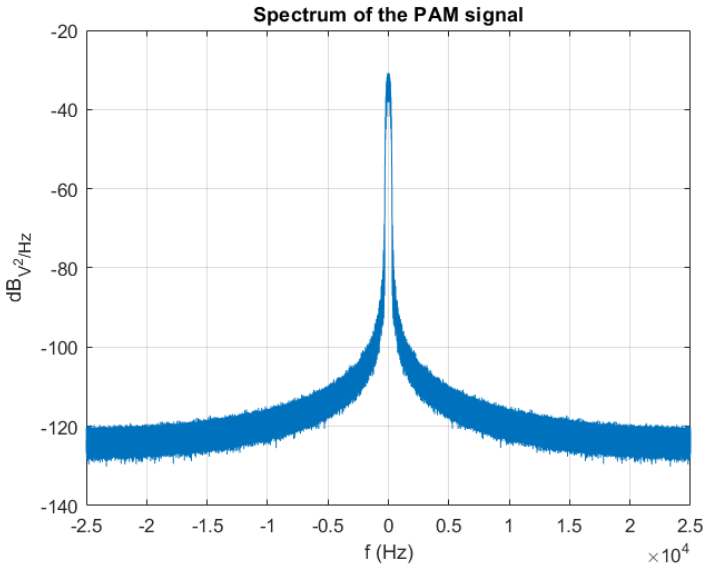




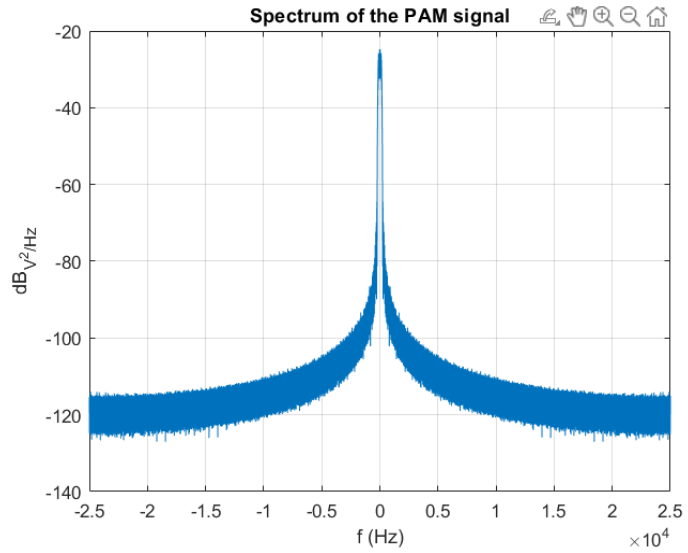
This picture is plotteed with parameter L = 4, ROOTRAISEDCOSINE pluse , rolloff=0.8

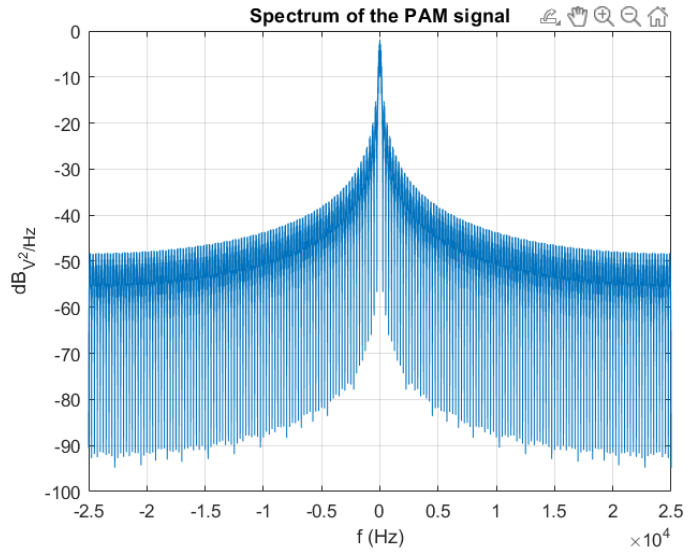


This picture is plotteed with parameter L = 4, ROOTRAISEDCOSINE pluse , rolloff=0.2

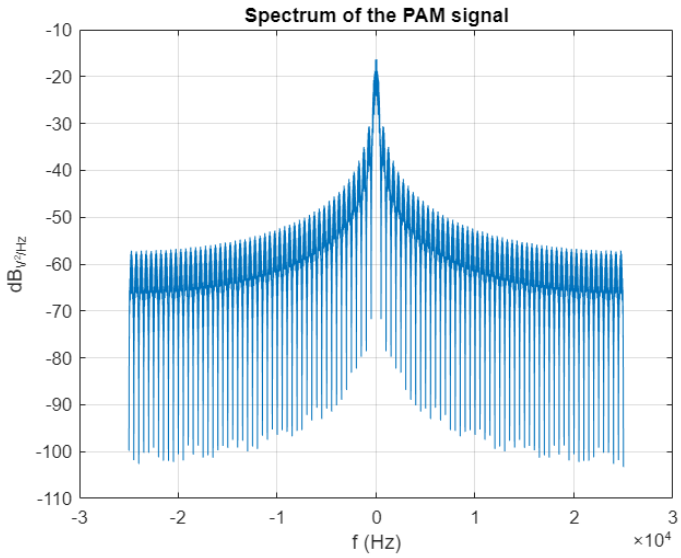


This picture is plotteed with parameter L = 8, ROOTRAISEDCOSINE pluse , rolloff=0.8





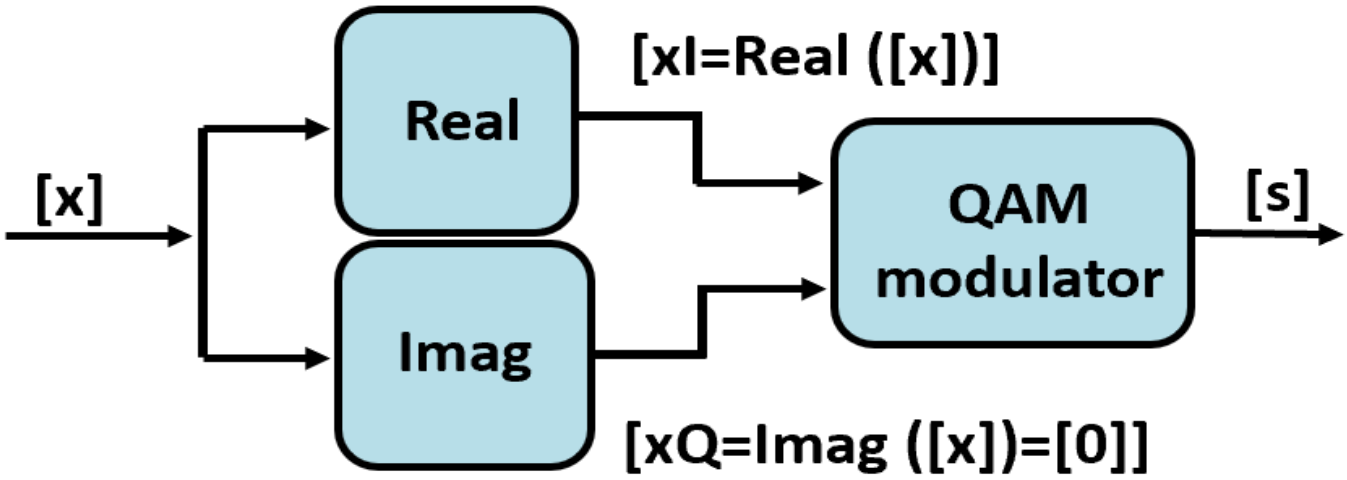
This picture is plotteed with parameter L = 16, ROOTRAISEDCOSINE pluse , rolloff=0.8



This picture is plotteed with parameter L = 4, RECT pluse

In frequency domain, the value of L have impact on signal gain, the larger value L is, the lower modulation gain is. This is beacuse symbols have a higher value need more energy. Raising the value of rolloff can effectively decerase the gain in the other frequency.The RECT pluse have a lower gain at 0 frequency and higher gain at other frequency.

QAM modulator




```
% *L-ASK modulation through the QAM modulator*
xI=x % in-phase signal
```

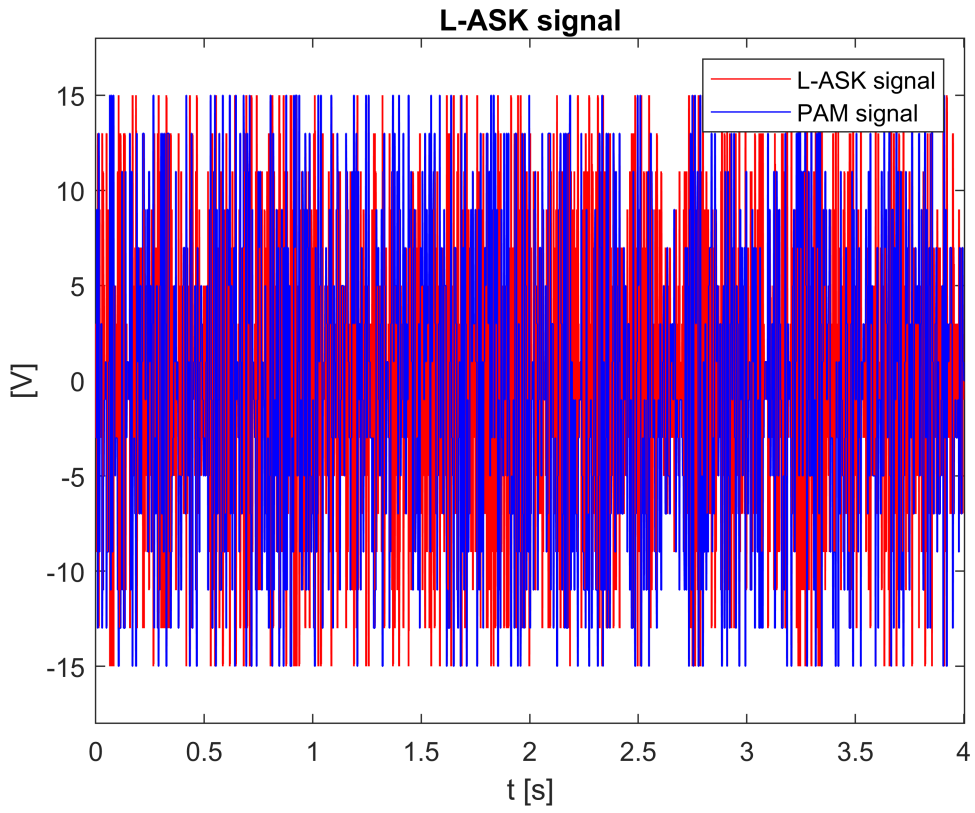
```
xI = 1×200199
    -3    -3    -3    -3    -3    -3    -3    -3    -3    -3    -3    -3    -3 ···
```

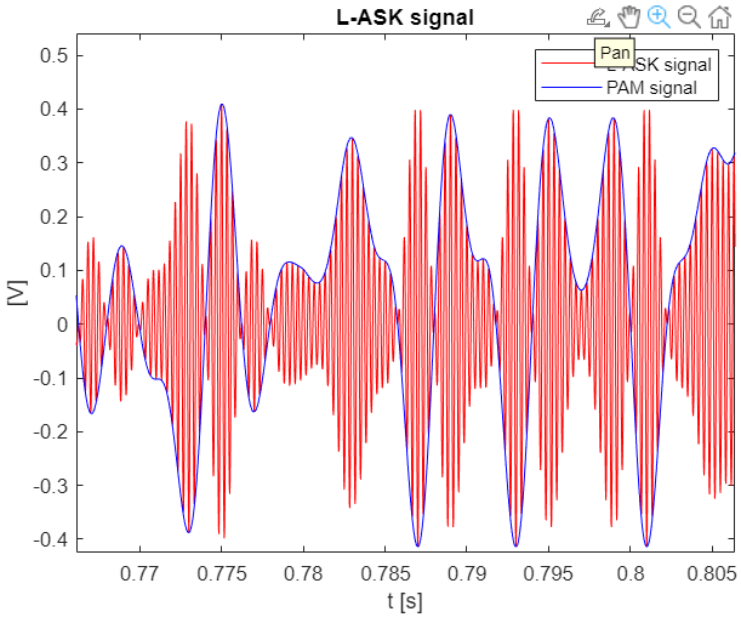
```
xQ=zeros(1,length(x)) % the quadrature component is zero
```

```
xQ = 1×200199
```

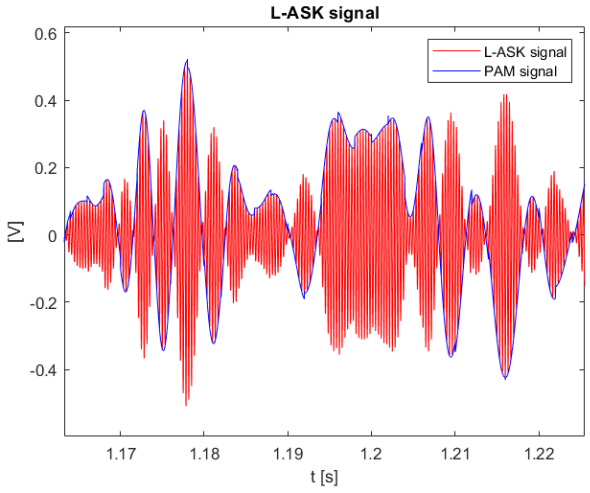
```
     0     0     0     0     0     0     0     0     0     0     0     0     0···
```

```
s=ModQAM_2023(xI,xQ,fc,T,fs);
%-----------------------------------------------------------------
% *Plot of the L-ASK signal and of its spectrum*
figure
plot(t,s,'r')
hold on
plot(t,x,'b')
axis([min(t) max(t) 1.2*min(x) 1.2*max(x)])
xlabel('t [s]')
ylabel ('[V]')
title('L-ASK signal')
legend('L-ASK signal','PAM signal')
```

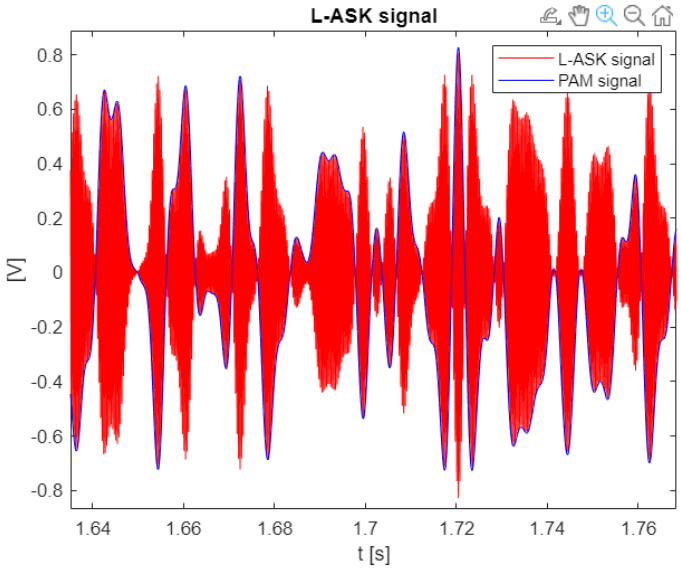




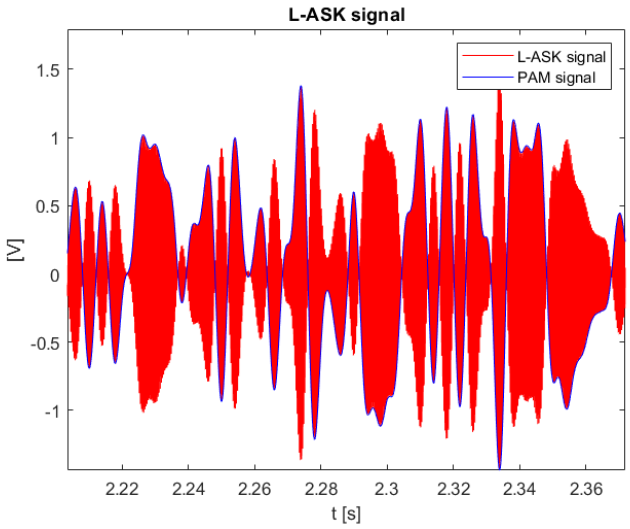
This picture is plotteed with parameter L = 4, ROOTRAISEDCOSINE pluse , rolloff=0.8



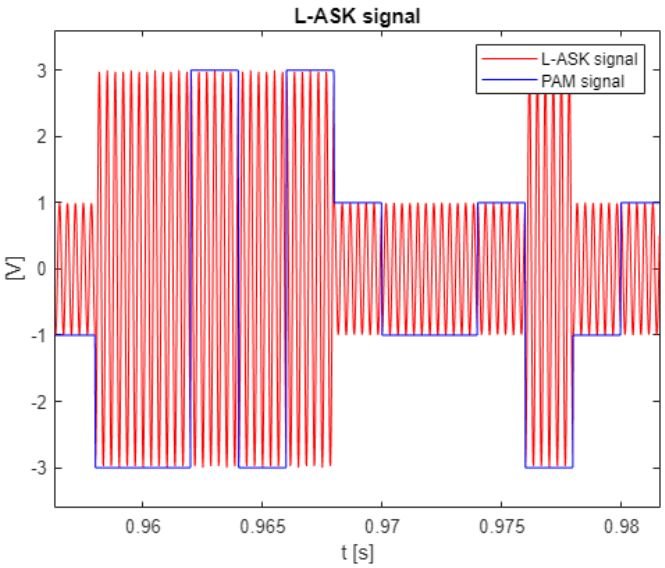
This picture is plotteed with parameter L = 4, ROOTRAISEDCOSINE pluse , rolloff=0.2



This picture is plotteed with parameter L = 8, ROOTRAISEDCOSINE pluse , rolloff=0.8



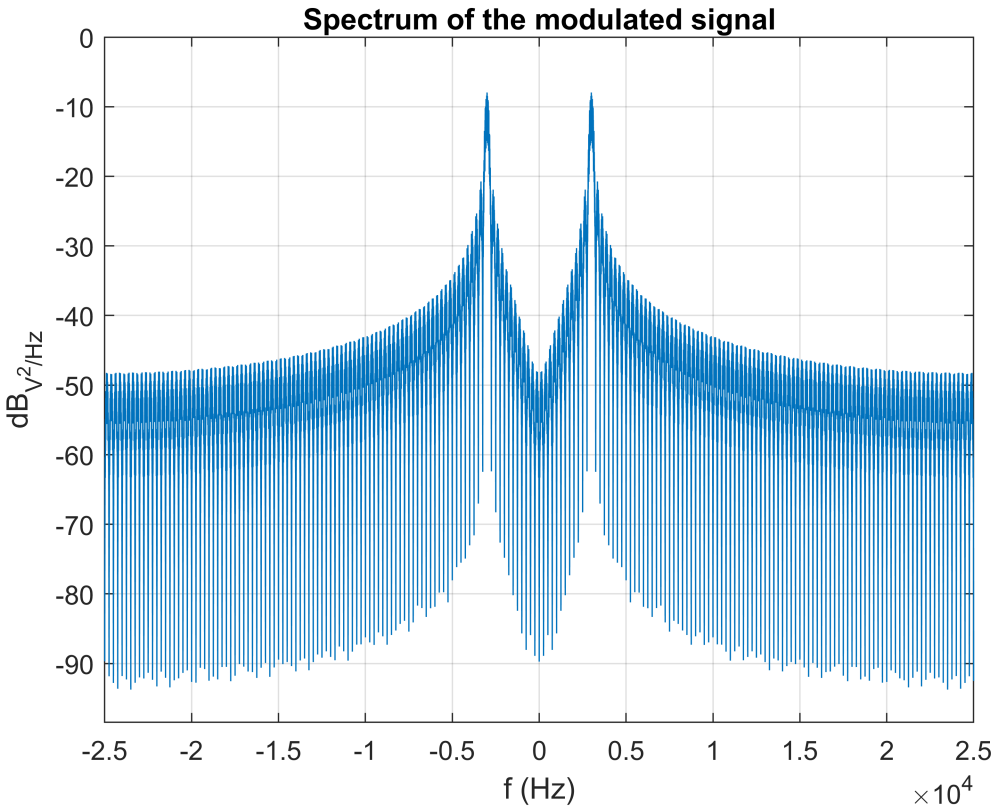
This picture is plotteed with parameter L = 16, ROOTRAISEDCOSINE pluse , rolloff=0.8

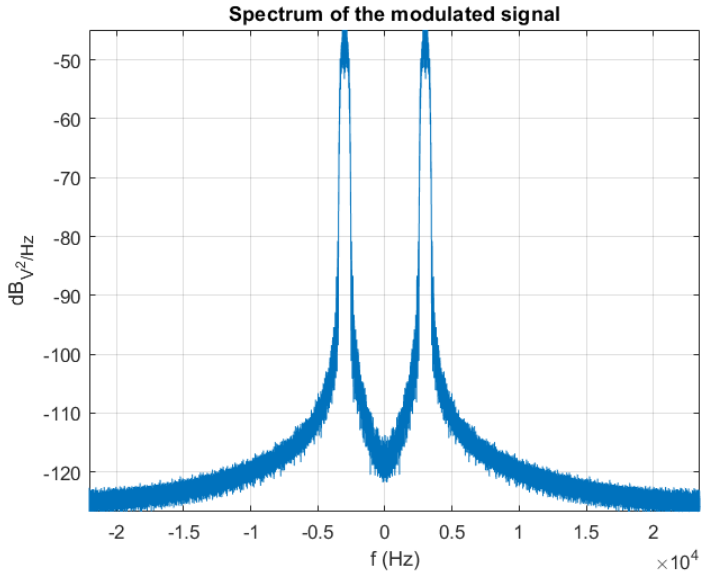


This picture is plotteed with parameter L = 4, RECT pluse

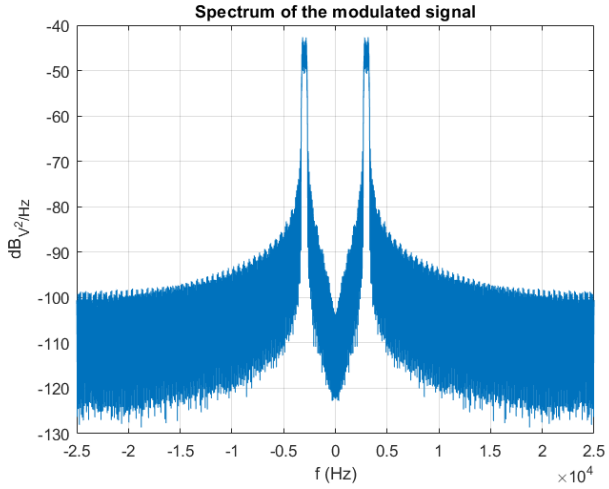
These pictures show the symbols sequence which is the combination of the PAM signals, and the results are similar to PAM signals.

```
figure
PlotSpectrum_2023(s,fs);
title('Spectrum of the modulated signal')
```

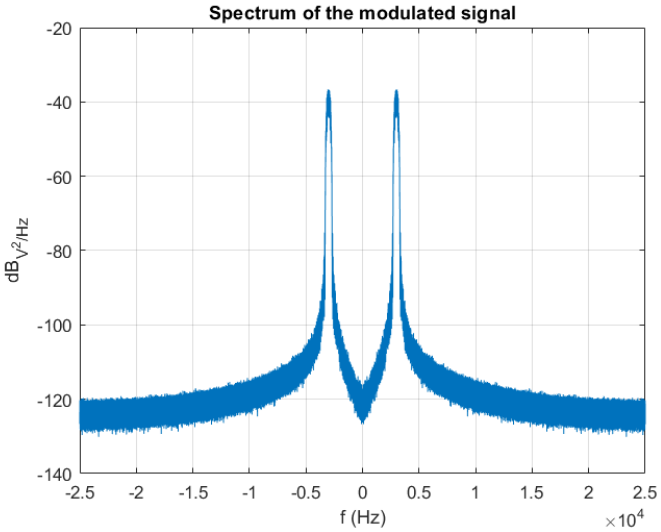




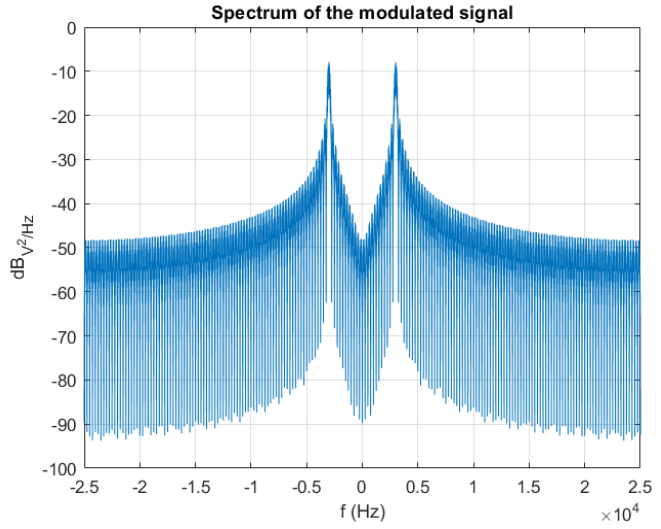
This picture is plotteed with parameter L = 4, ROOTRAISEDCOSINE pluse , rolloff=0.8



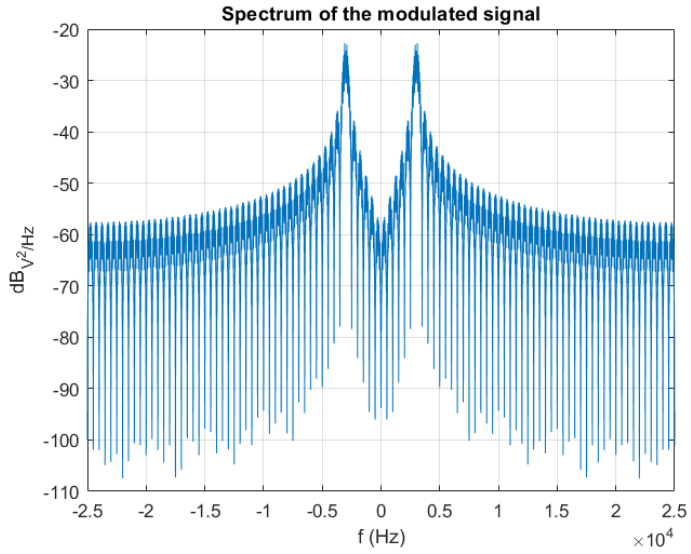
This picture is plotteed with parameter L = 4, ROOTRAISEDCOSINE pluse , rolloff=0.2



This picture is plotteed with parameter L = 8, ROOTRAISEDCOSINE pluse , rolloff=0.8



This picture is plotteed with parameter L = 16, ROOTRAISEDCOSINE pluse , rolloff=0.8



This picture is plotteed with parameter L = 4, RECT pluse

These pictures show the similar result to spectrum of the PAM signals, but the modulated signal frequency is different.

```
%-----------------------------------------------------------------
% output the signal to the audio card
sound(s,fs)
```

2-ASK communication system

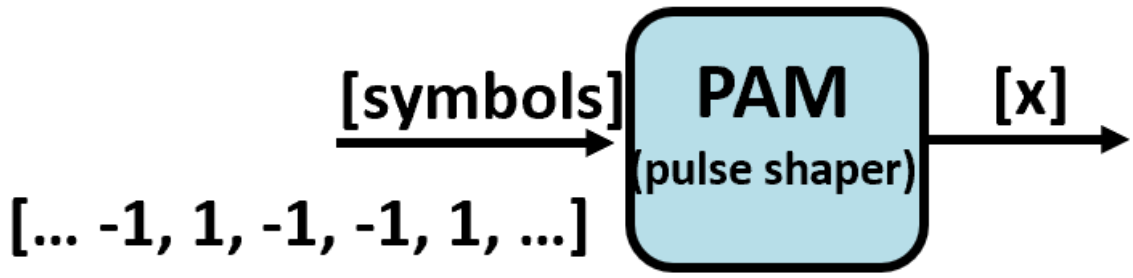
```
% ********* Initialization of the environment
clear all % clear all variables
close all % close all plots
clc % clear the screen in the command window
```

Signal specifications

The following code segment initializes signla parameters for a 2-ASK communication system.On the basic of previous activities, the code adds noise to the signals and the demodulator component.

```
% ********* Signal specifications
bits_per_symbol=1; % number of bits that correspond to a symbol
L=2^bits_per_symbol; % number of levels of the modulation alphabet
nbits=100000*bits_per_symbol; % number of bits to be transmitted
Br=500; % [bits/s] bit rate
fs=50000; % [samples/s] sampling frequency
Ts=1/fs; % sampling time[s].
fc=3000; % [Hz] carrier frequency
Bs=Br/log2(L); % [symbols/s] Symbol rate
nsps=fs/Bs; % samples per symbol
```

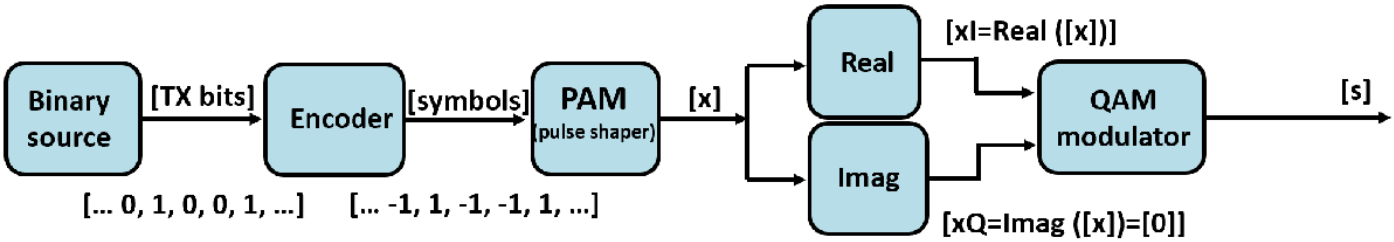
Pulse shaper specifications



The following code segment initializes the pluse shaper parameters.

```
% ********* Pulse shaper specifications*
Nf=5*nsps; % number of filter coefficients for the pulse shaper
pulsetype='RECT';
switch pulsetype
case 'ROOTRAISEDCOSINE' % root raised cosine pulse
rolloff=0.8 % roll-off factor for root raised cosine pulses
bandwidth_Hz=0.5*Bs*(1+rolloff); % bandwidth of the baseband signal
case 'RECT' % rectangular pulse,
rolloff=[];
bandwidth_first_lobe_Hz=Bs;
end
```

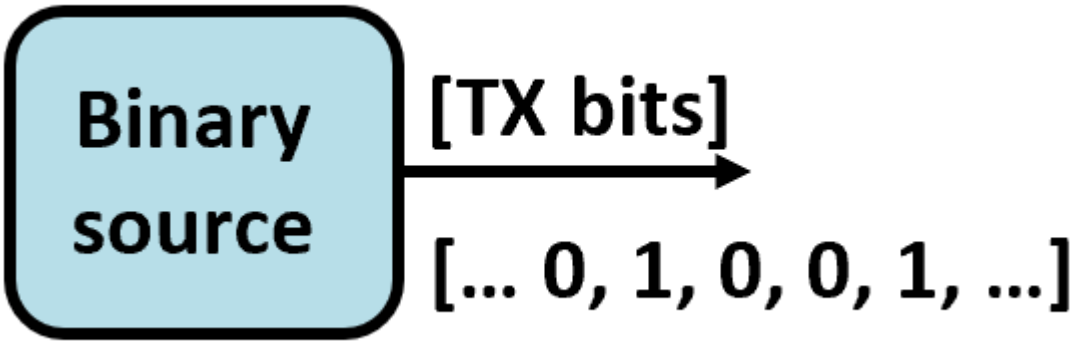
Modulator specifications



The transmitter has been already implemente in activities.

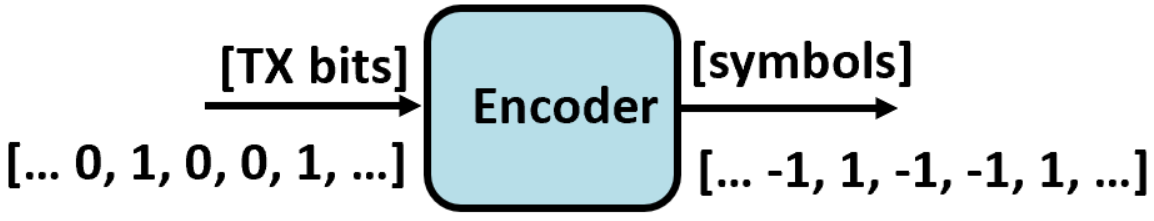
```
% ********* Modulator specifications
psi=0; % carrier offset at the receiver
% ********* Monte Carlo simulation
index=1;
for EbN0_dB=1:10 % Run a Monte Carlo simulation for 10 different values of SNR
```

Source bits generation



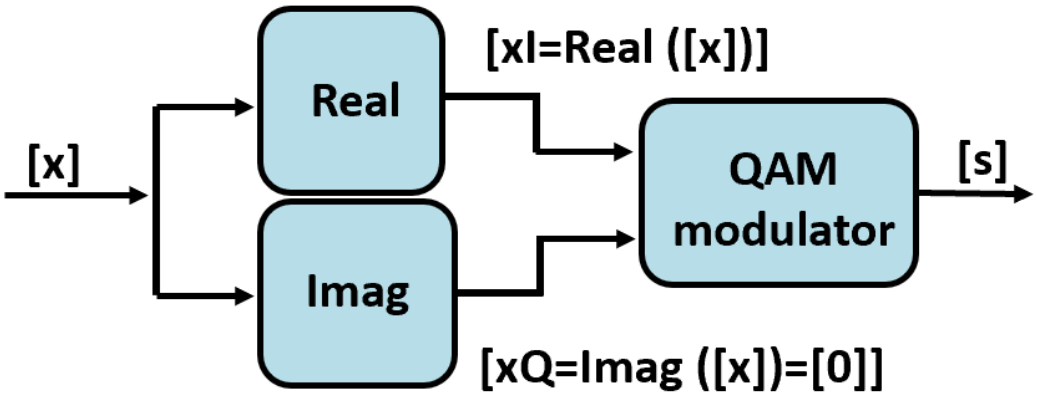

```
% ********* Source bits generation
source_bits=BinarySource_2023(nbits); % vector with the source bits
```

Symbols generation with Gray mapping



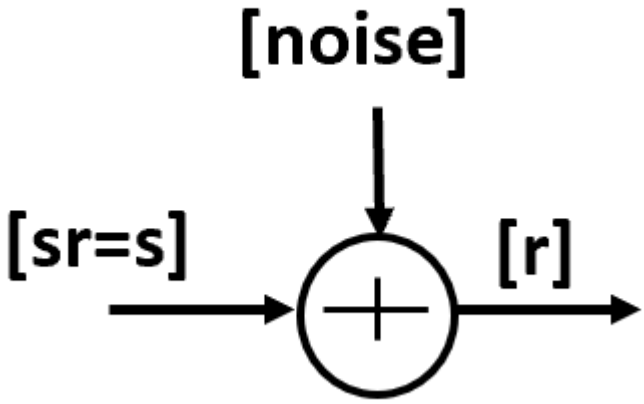

```
% ********* Symbols generation with Gray mapping
symbols=Encoder_2023(source_bits, L);
```

2-ASK modulation through the QAM modulator




```
% ********* Generation of the baseband PAM signal
[x,h,DelayPAM]=PAMmodulator_2023(symbols,nsps,Nf,pulsetype,rolloff);
T=Ts*(length(x)-1); % signal duration
% ********* 2-ASK modulation through the QAM modulator
xI=x; % in-phase signal
xQ=zeros(1,length(x)); % the quadrature component is zero
s=ModQAM_2023(xI,xQ,fc,T,fs);
```

AWGN Channel

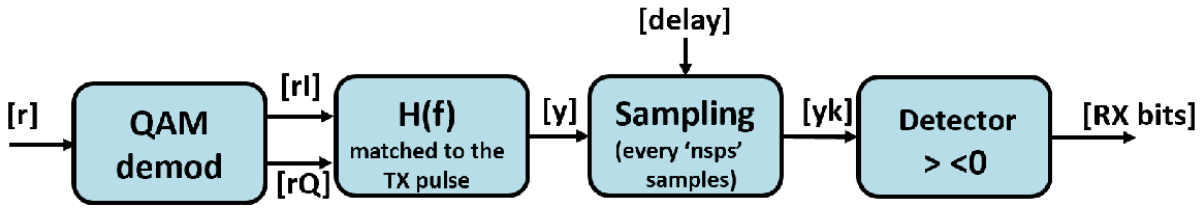


The signal is characterized by its energy Eb in a bit interval Ts or, equivalently, by its power P=Eb/Ts.
The noise is characterized by its (on-sided) power spectral density N0 or equivalently, by its power Pnoise=N0*fs/2 in the band [0, fs/2].At the output of the channel, the signal is characterized by the Signal-to-Noise Ratio.

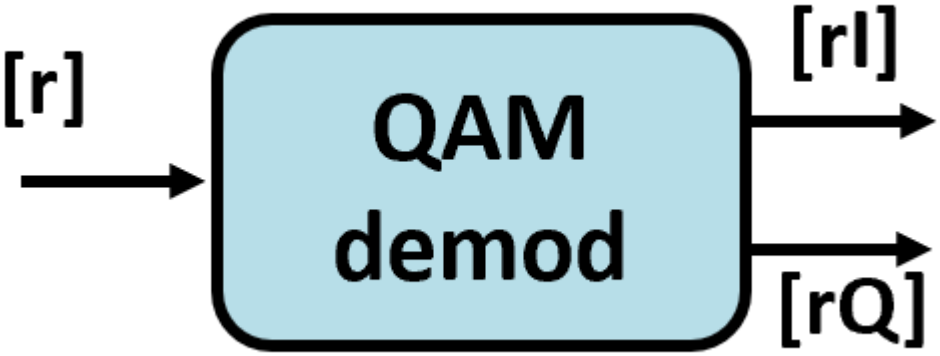
```
% ********* Noise generation and sum with the received signal
```

```
figure
sr=s;
P=(1/length(sr))*sum(sr.^2); % Estimation of the mean power of the signal
EbN0=10^(0.1*EbN0_dB); % SNR expressed in linear units
sigman=sqrt(P*fs/(2*Br*EbN0)); % calculation of the noise's standard deviation
noise=sigman.*randn(1,length(sr)); % generation of the Gaussian noise
r=sr+noise; % addition of the noise to the received signal
```

Demodulator specifications

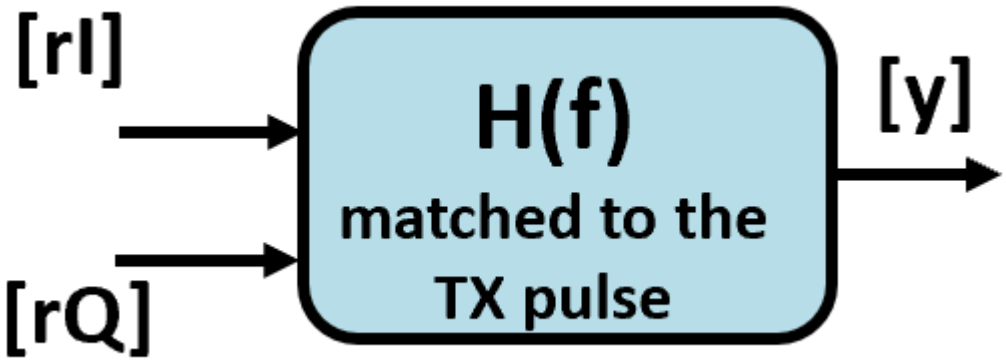


QAM demodulator



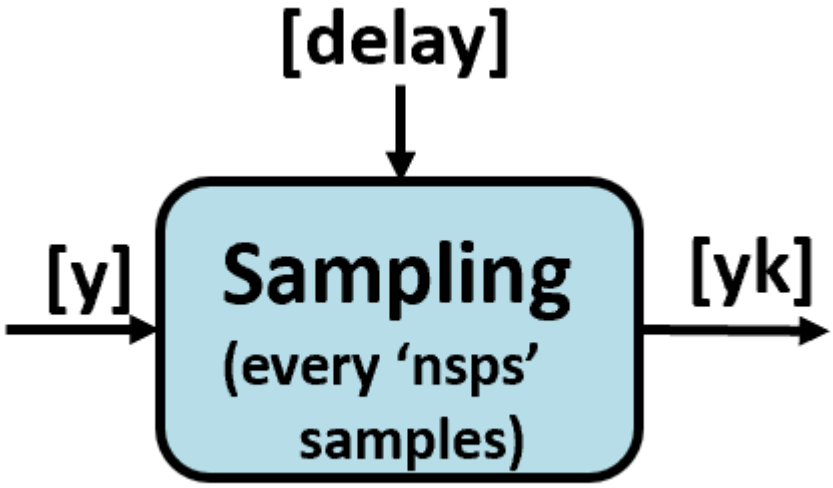

```
% ********* QAM demodulator
[rI,rQ,DelayQAM]=DeModQAM_2023(r,fc,T,fs,psi); % demodulate the signal
% ********* Matched filter
r=rI+1i*rQ;
```

filtering with the matched filter



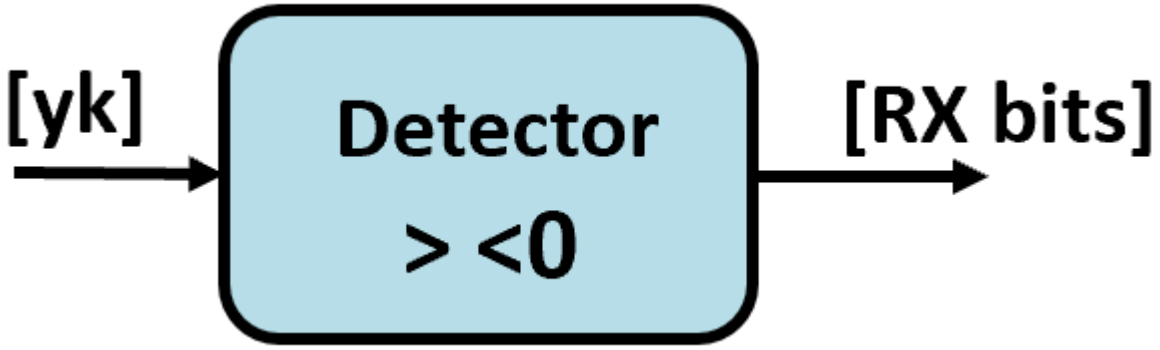

```
y=conv(r,h)/nsps; % filtering with the matched filter
```

Sampler



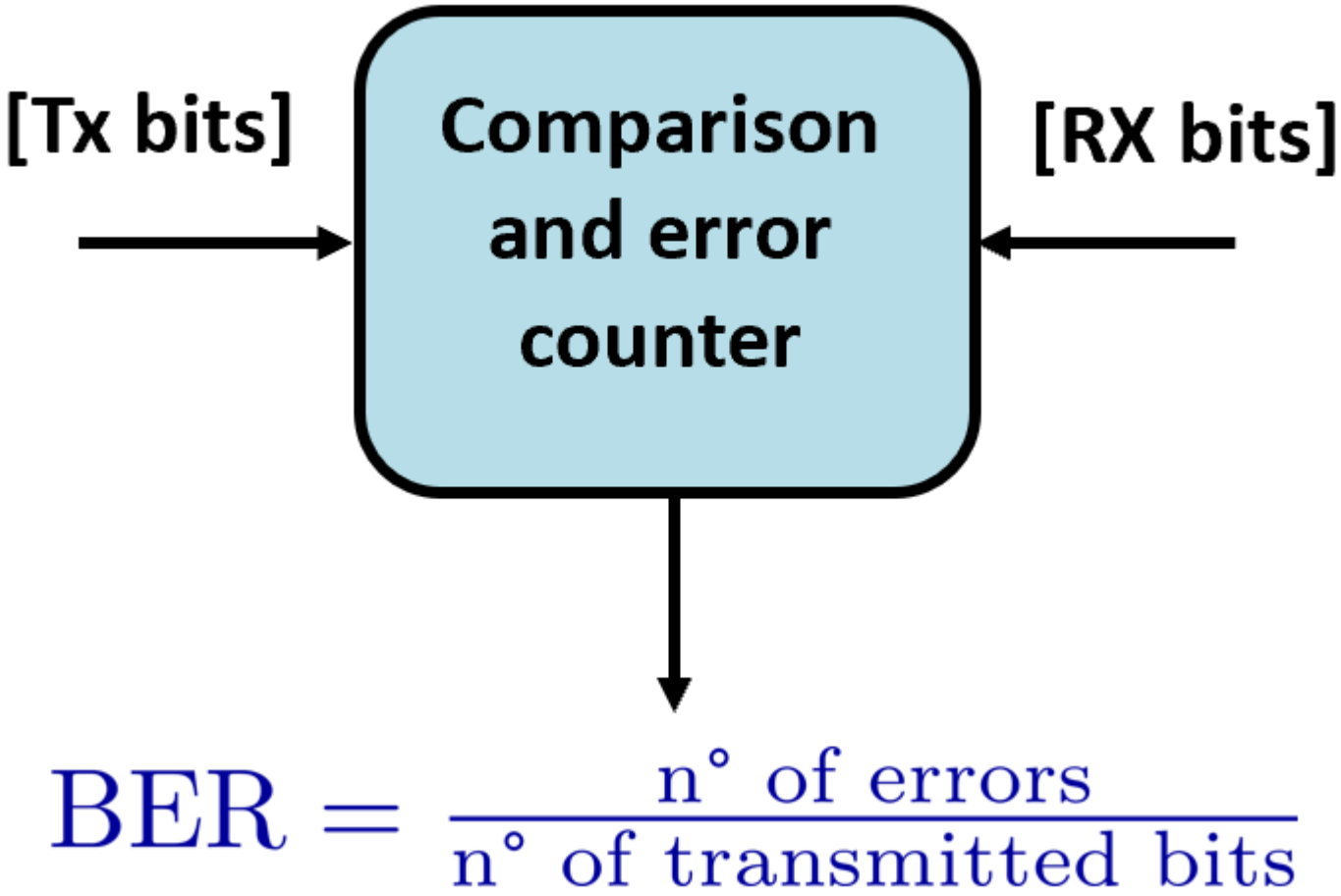

```
% ********* Sampler
% compute the overall delay introduced by the TX and RX filters
delay=round (2*DelayPAM+DelayQAM);
% sample the signal every symbol time accounting for the delay
yk = y(delay:nsps:nsps*nbits+delay-1);
```

Detector




```
% ********* Detector
RXdata=real(yk) > 0; % take only the real part
```

Bit Error Rate (BER) estimate

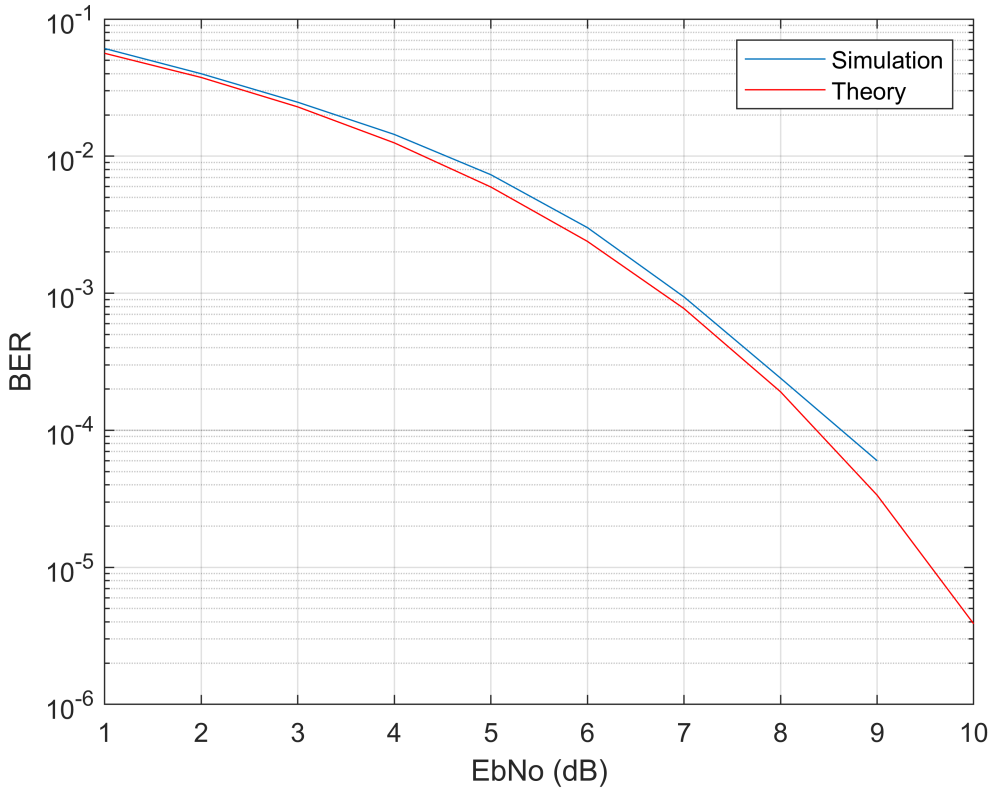



```
% ********* Bit Error Rate (BER) estimate
```

```
noe=sum(abs(source_bits-RXdata)); % Errors count
nod=length(source_bits);
SNRdB(index)=EbN0_dB;
ber(index)=noe/nod; % Estimate the BER
fprintf('EbNo=%.1f (dB) BER=%g\n',SNRdB(index),ber(index));
index=index+1;
% ********* End of Monte Carlo simulation
end
```

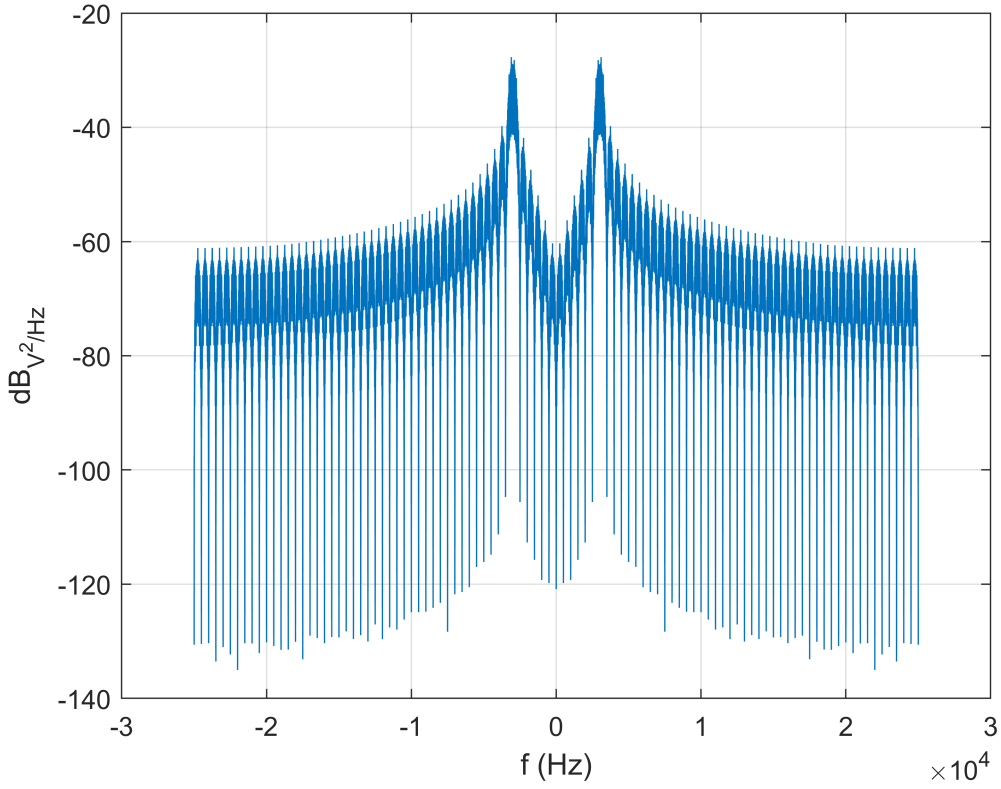
```
EbNo=1.0 (dB) BER=0.06102
EbNo=2.0 (dB) BER=0.03993
EbNo=3.0 (dB) BER=0.02474
EbNo=4.0 (dB) BER=0.0144
EbNo=5.0 (dB) BER=0.00732
EbNo=6.0 (dB) BER=0.00301
EbNo=7.0 (dB) BER=0.00094
EbNo=8.0 (dB) BER=0.00024
EbNo=9.0 (dB) BER=6e-05
EbNo=10.0 (dB) BER=0
```

```
% ********* BER vs EbN0 plot
figure
semilogy(SNRdB,ber);
xlabel('EbNo (dB)');
ylabel('BER');
grid on
hold on
SNR=10.^(0.1*SNRdB);
Pb=0.5*erfc(sqrt(SNR));
plot(SNRdB,Pb,'r')
legend('Simulation', 'Theory')
```

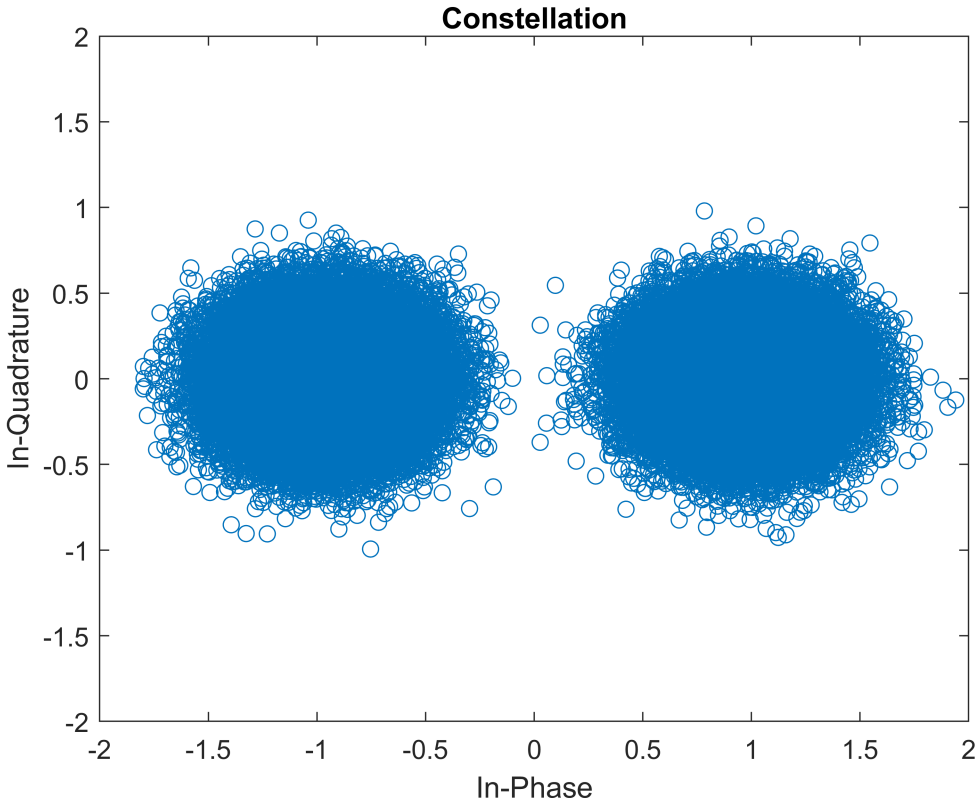


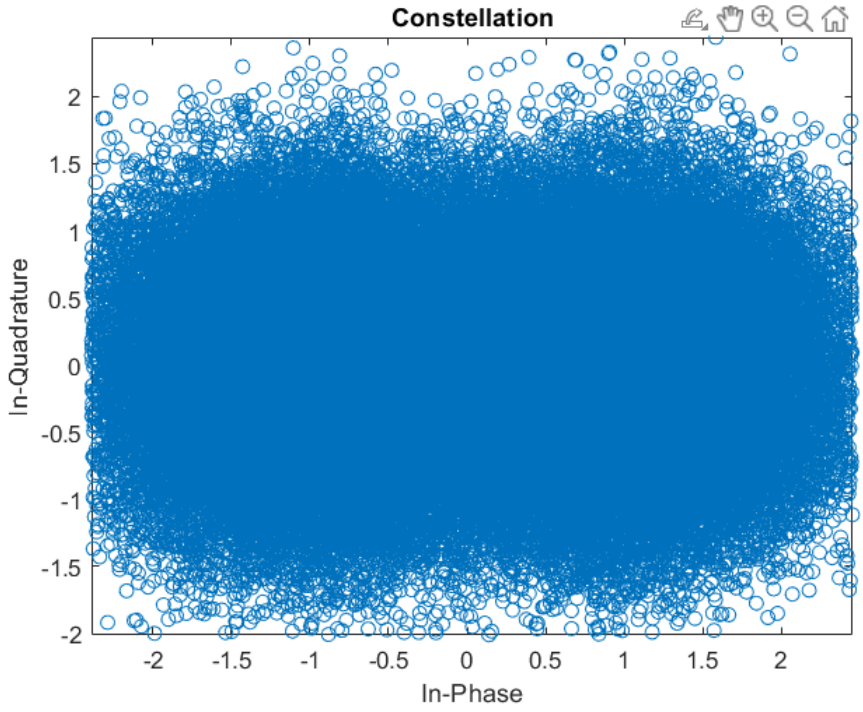
This picture above gives the information of the relationship between EbN0 and BER, It can be concluded that with the incraesing value of EbN0, the number BER gets lower and lower.Which means a signals with higher EbN0 represent a higher quality of signals. The theoretical results are slightly lower than the simulation results, but this lower part of the value can be ignored. Because of the limitation of computing power this activity can not run a large number of bits, the BER is 0 when EbN0=9 or 10.

```
% ********* Spectrum plot
figure
PlotSpectrum_2023(s,fs);
```

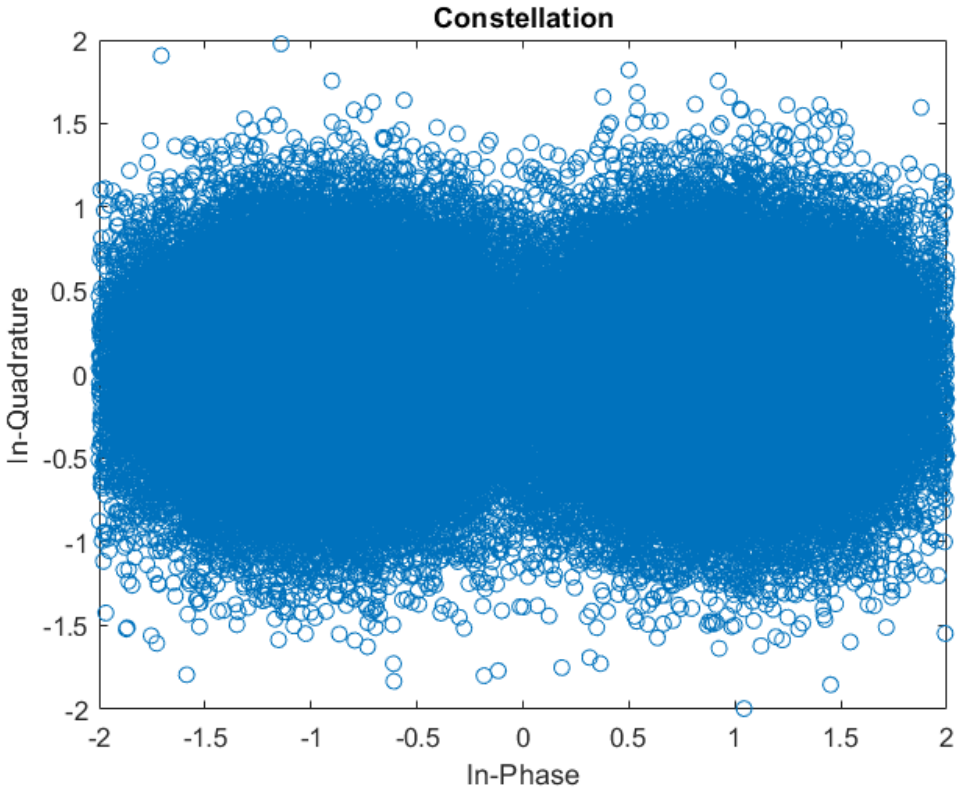



```
% ********* Constellation
x=real(yk);
y=imag(yk);
figure
plot(x,y,'o'), axis ([-2 2 -2 2]);
title('Constellation');
xlabel('In-Phase');
ylabel('In-Quadrature');
```

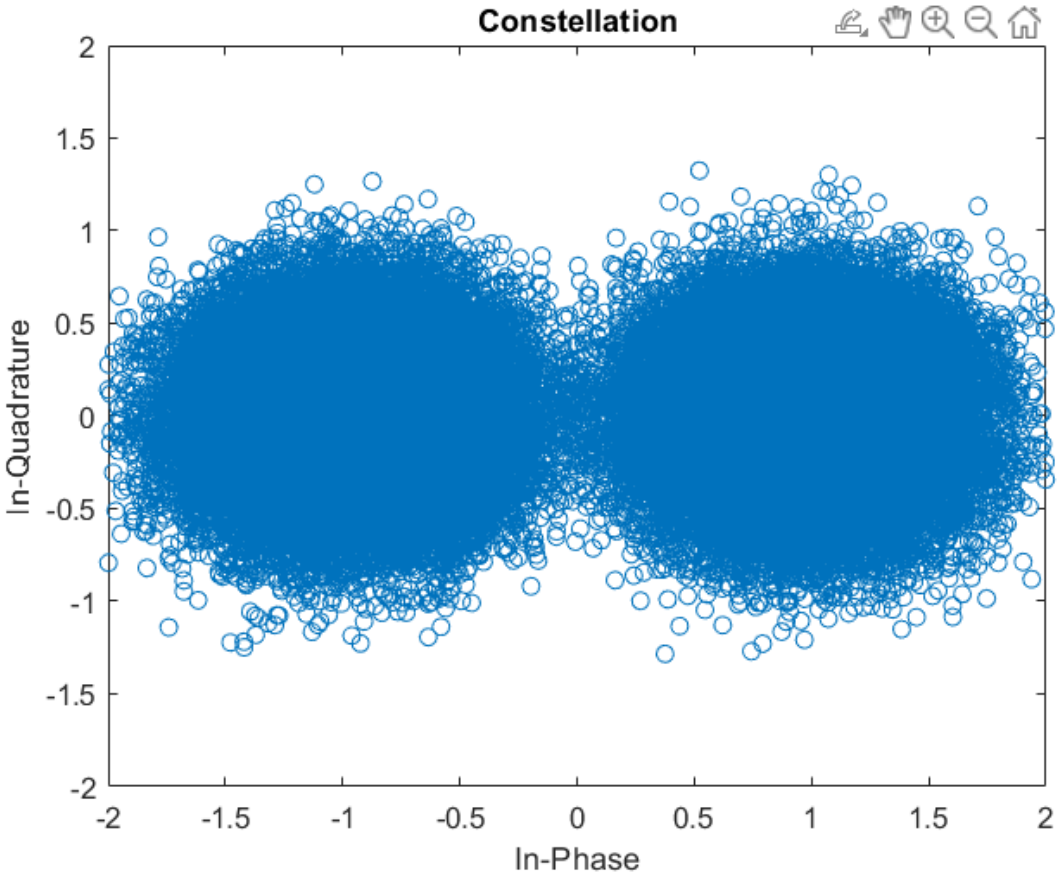




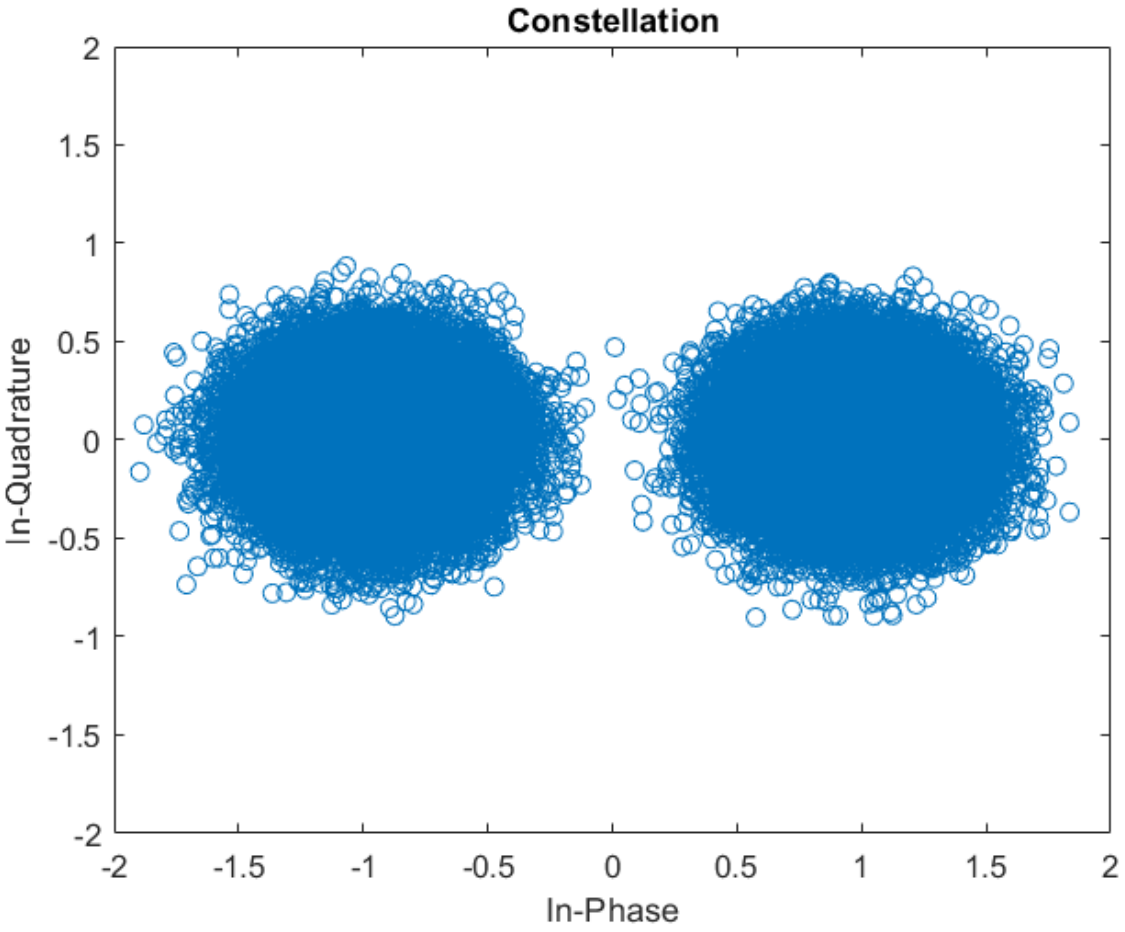
This picture is plotteed with parameter nbits=100000*bits_per_symbol,SNR=1



This picture is plotteed with parameter nbits=100000*bits_per_symbol,SNR=4



This picture is plotteed with parameter nbits=100000*bits_per_symbol,SNR=7

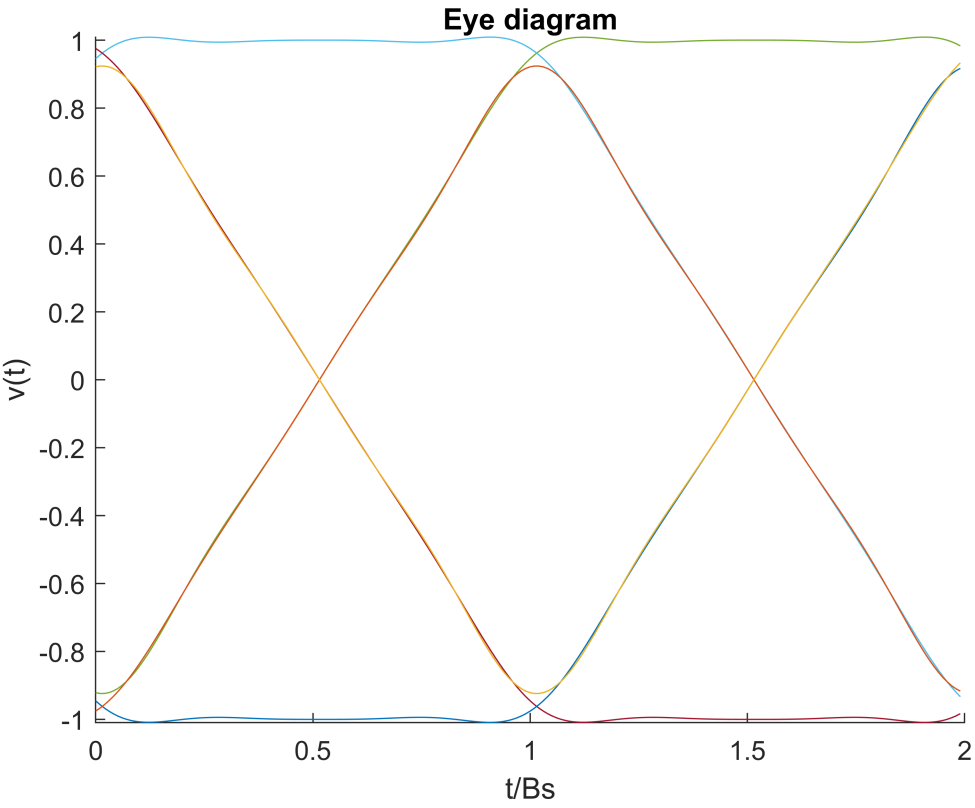


This picture is plotteed with parameter nbits=100000*bits_per_symbol,SNR=10

These pictures above are constellation for different values of the SNR.The higher value of SNR, the larger average distance between 1 and -1, so It is much easier to detect the correct information. This is correspond with the result of BbN0-BER.

```
% ********* Eye diagram
H=2; % number of observation symbols
[yi,yq,DelayQAM]=DeModQAM_2023(sr,fc,T,fs,psi); % demodulate the signal
vt0=conv(yi,h)/nsps;
mm=max(abs(vt0));
x1=0:H*nsps-1;
x2=x1./nsps;
figure;
hold on
for iii=1:10
x=x1+iii*nsps+delay-1;
y=real(vt0(x)); % plot the in-phase via
plot (x2,y), axis([0 H -mm mm]);
end
title('Eye diagram');
```

```
xlabel('t/Bs');
ylabel('v(t)');
```

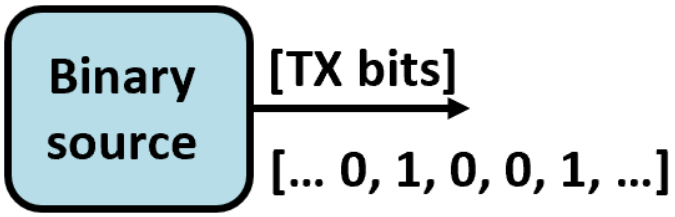


This is the eye diagram with no noise. It can easily detect the signal of 1 and -1.

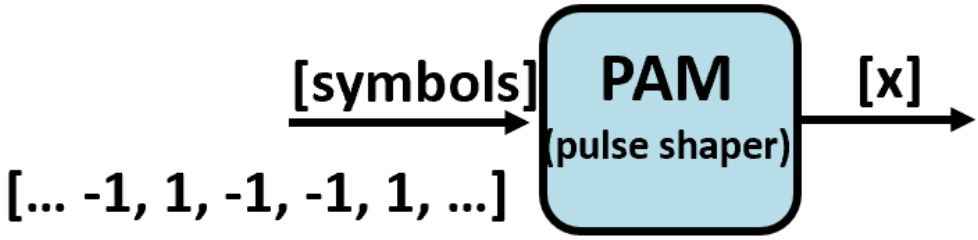
Hamming Code

Based on the previous activities, the following code adds the 7-4hamming to the system to decrease the BER.

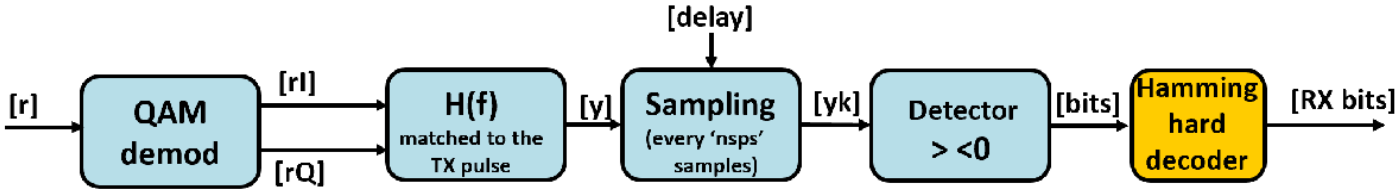
```
%% *2-ASK system (both transmitter and receiver) with Hamming coding*
% ************* Initialization of the environment*
clear all % clear all variables
close all % close all plots
warning off
```



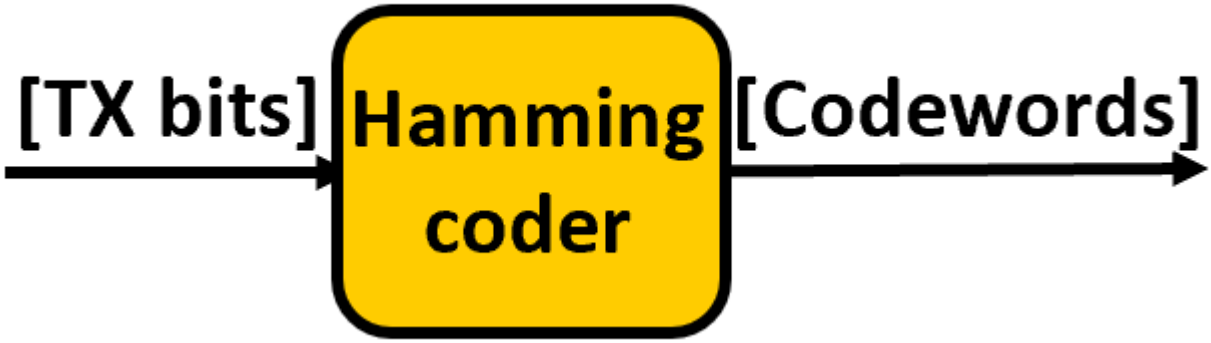

```
% ************* Signal specifications*
bits_per_symbol=1; % number of bits that correspond to a symbol
L=2^bits_per_symbol; % number of levels of the modulation alphabet
nbits=10000*bits_per_symbol; % number of bits to be transmitted
Br=500; % [bits/s] bit rate
fs=50000; % [samples/s] sampling frequency
Ts=1/fs; % sampling time[s].
fc=3000; % [Hz] carrier frequency
Bs=Br/log2(L); % [symbols/s] Symbol rate
nsps=fs/Bs; % samples per symbol
```



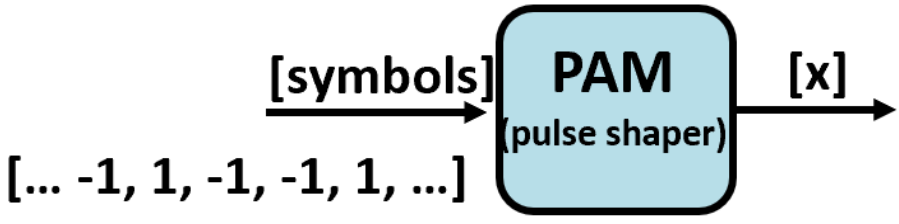

```
% ************* Pulse shaper specifications*
Nf=5*nsps; % number of filter coefficients for the pulse shaper
pulsetype='RECT';
switch pulsetype
case 'ROOTRAISEDCOSINE'
rolloff=0.8 % roll-off factor for root raised cosine pulses
bandwidth_Hz=0.5*Bs*(1+rolloff); % bandwidth of the baseband signal
case 'RECT' % rectangula pulse
rolloff=[];
bandwidth_first_lobe_Hz=Bs;
end
```




```
% ************* Demodulator specifications*
psi=0; % carrier offset at the receiver
% ************* Check on the number of bits to by coded by the Hamming encoder*
if rem(nbits,4)~=0 % The number of bits must be a multiple of 4
disp('The number of bits to e transmitted must be a multiple of 4')
return % The execution is stopped
end
% ************* Monte Carlo simulation
index=1;
for EbN0_dB=1:10% Run a Monte Carlo simulation for 10 different values of SNR
% ************* Source bits generation*
source_bits=BinarySource_2023(nbits); % vector with the source bits
% ************* Hamming coding*
```

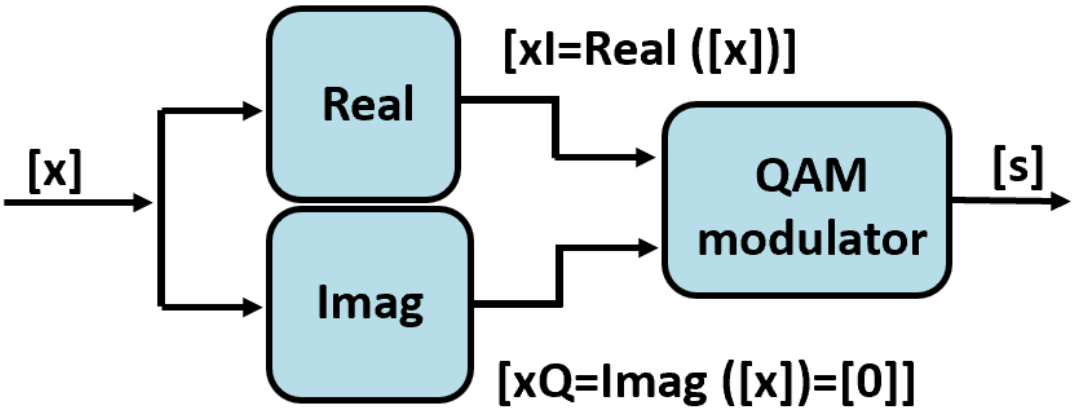



```
codedbits=HammingCoder_2023(source_bits); % Hamming coder
nCodedBits=length(codedbits);
% ************* Symbols generation with Gray mapping*
symbols=Encoder_2023(codedbits, L);
% ************* Generation of the baseband PAM signal*
```

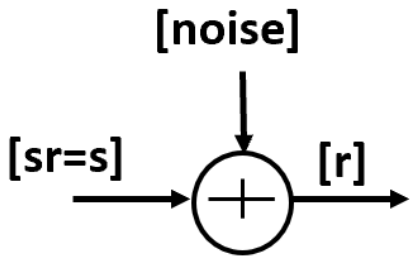



```
[x,h,DelayPAM]=PAMmodulator_2023(symbols,nsps,Nf,pulsetype,rolloff);
T=Ts*(length(x)-1); % signal duration
```

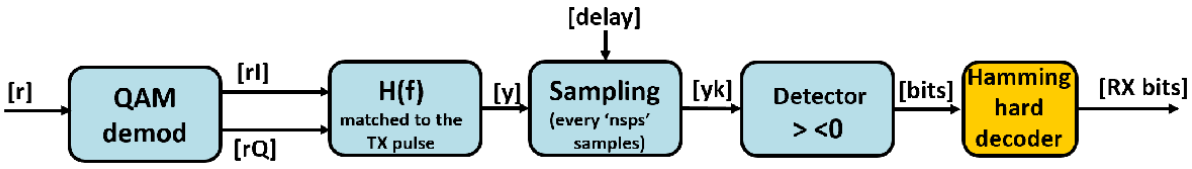
```
% ************ 2-ASK modulation through the QAM modulator*
```



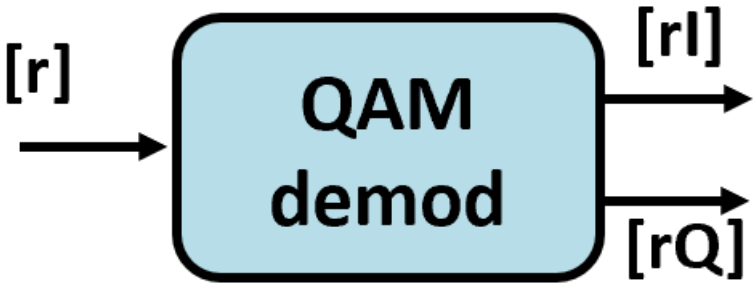

```
xI=x; % in-phase signal
xQ=zeros(1,length(x)); % the quadrature component is zero
s=ModQAM_2023(xI,xQ,fc,T,fs);
% ************ Noise generation and sum with the received signal*
```



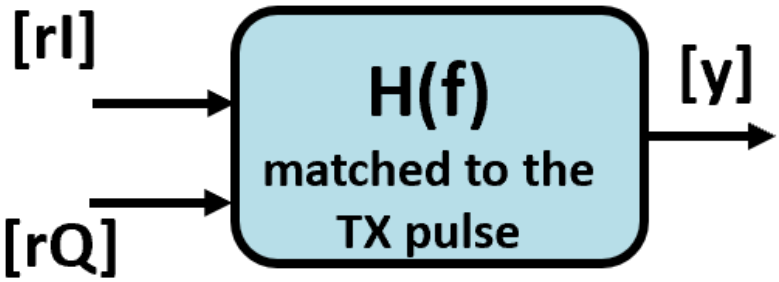

```
sr=s;
P=(1/length(sr))*sum(sr.^2); % Estimation of the mean power of the signal
EbN0=10^(0.1*EbN0_dB); % SNR expressed in linear units
sigman=sqrt(P*fs/(2*Br*EbN0)); % calculation of the noise's standard deviation
noise=sigman.*randn(1,length(sr)); % generation of the Gaussian noise
r=sr+noise; % addition of the noise to the received signal
```



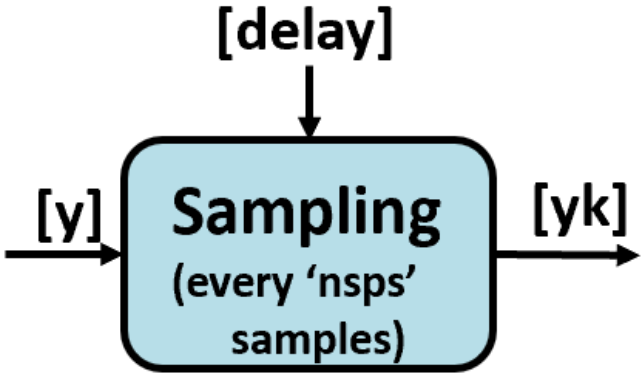

```
% ************ QAM demodulator*
```



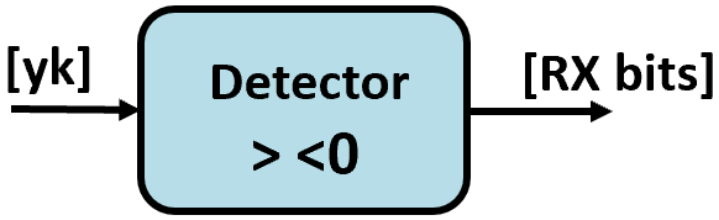

```
[rI,rQ,DelayQAM]=DeModQAM_2023(r,fc,T,fs,psi); % demodulate the signal
% ************* Matched filter*
```



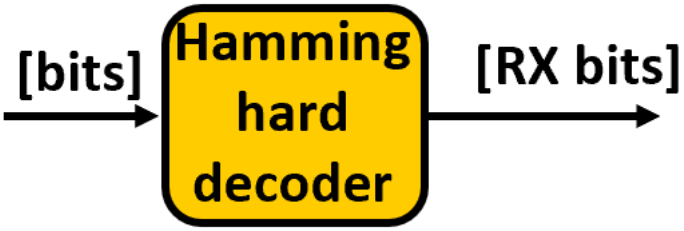

```
r=rI+1i*rQ;
y=conv(r,h)/nsps; % filtering with the matched filter
% ************* Sampler*
```




```
% compute the overall delay introduced by the TX and RX filters
delay=round (2*DelayPAM+DelayQAM);
% sample the signal every symbol time accounting for the delay
yk = y(delay:nsps:nsps*nCodedBits+delay-1);
% ************* Detector*
```




```
b=real(yk) > 0; % take only the real part
% ************* Hamming decoder*
```

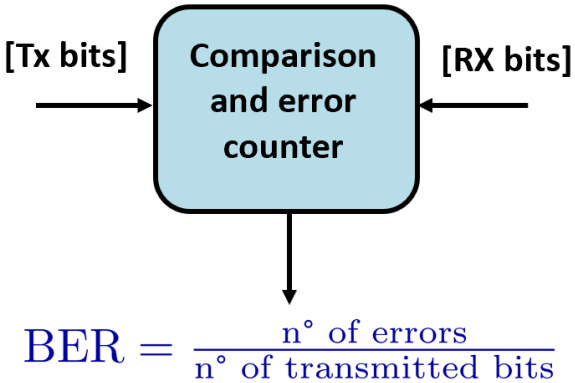


Compare the received (possibly wrong) codewords and check the table selecting the one with the minimum Hamming distance.

```
RXdata=HammingHardDecoder_2023(b); % hard decoding
```

Estimate the BER

```
% ************* Bit Error Rate (BER) estimate*
noe=sum(abs(source_bits-RXdata)); % Errors count
nod=length(source_bits);
SNRdB(index)=EbN0_dB;
```

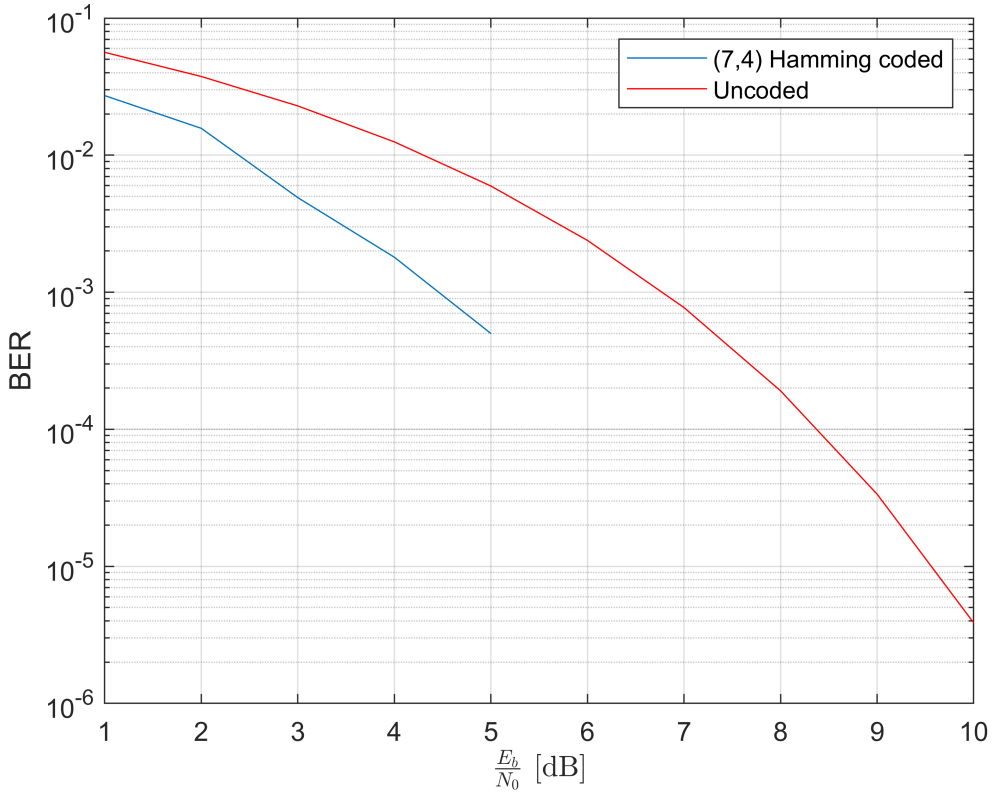


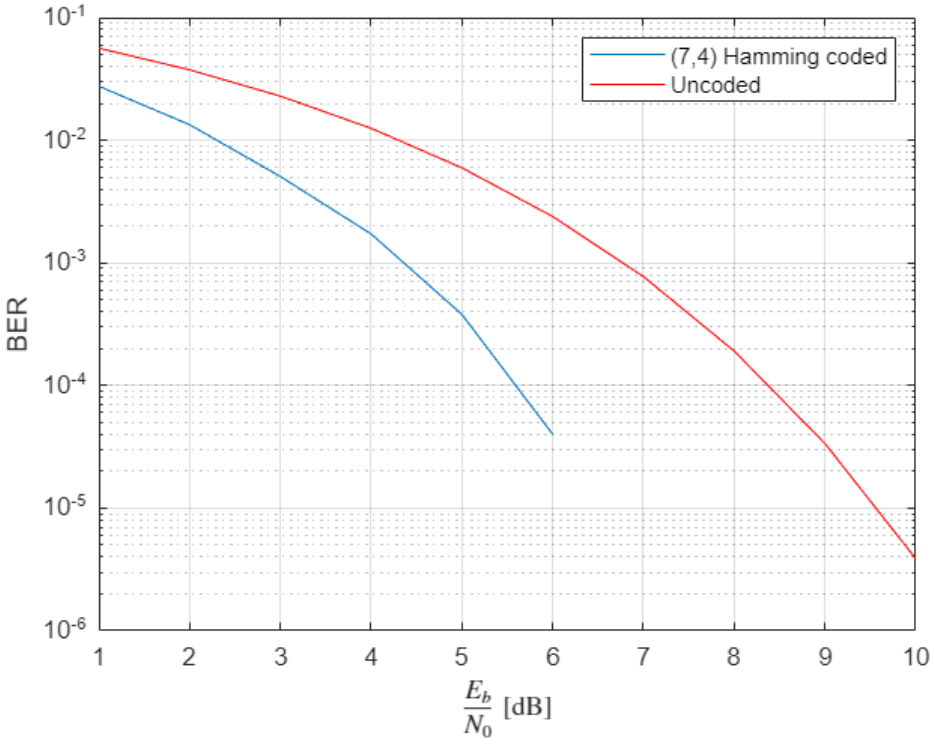

```
ber(index)=noe/nod; % Estimate the BER
fprintf('EbNo=%.1f (dB) BER=%g\n',SNRdB(index),ber(index));
index=index+1;
```

```
% ************* End of Monte Carlo simulation*
end
```

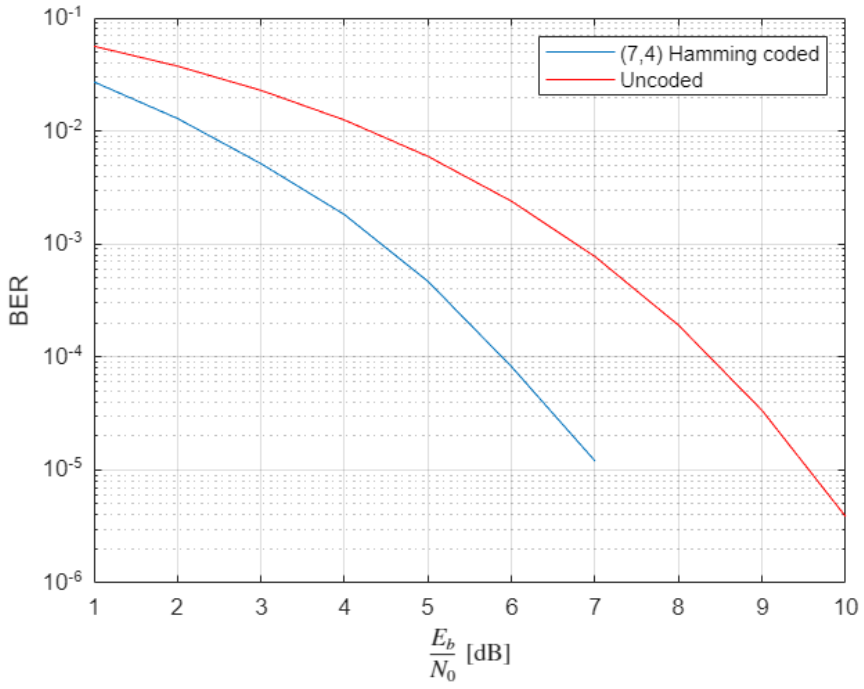
```
EbNo=1.0 (dB) BER=0.0272
EbNo=2.0 (dB) BER=0.0157
EbNo=3.0 (dB) BER=0.0049
EbNo=4.0 (dB) BER=0.0018
EbNo=5.0 (dB) BER=0.0005
EbNo=6.0 (dB) BER=0
EbNo=7.0 (dB) BER=0
EbNo=8.0 (dB) BER=0
EbNo=9.0 (dB) BER=0
EbNo=10.0 (dB) BER=0
```

```
% ************* BER vs EbN0 plot*
figure
semilogy(SNRdB,ber);
xlabel('$\frac{E_b}{N_0}$ [dB]','HorizontalAlignment','center',...
'Interpreter','latex');
ylabel('BER');
grid on
hold on
SNR=10.^(0.1*SNRdB);
Pb=0.5*erfc(sqrt(SNR));
plot(SNRdB,Pb,'r')
legend('(7,4) Hamming coded', 'Uncoded')
```





This picture is plotted with parameter nbits=50000*bits_per_symbol;

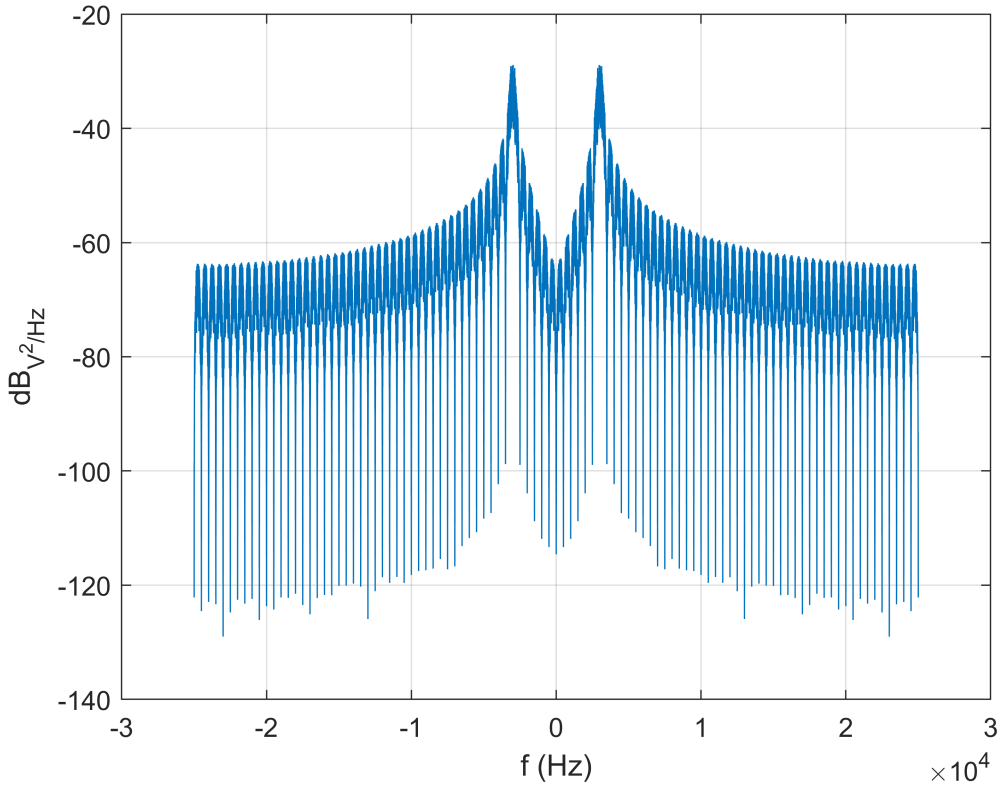


This picture is plotted with parameter nbits=100000*bits_per_symbol;

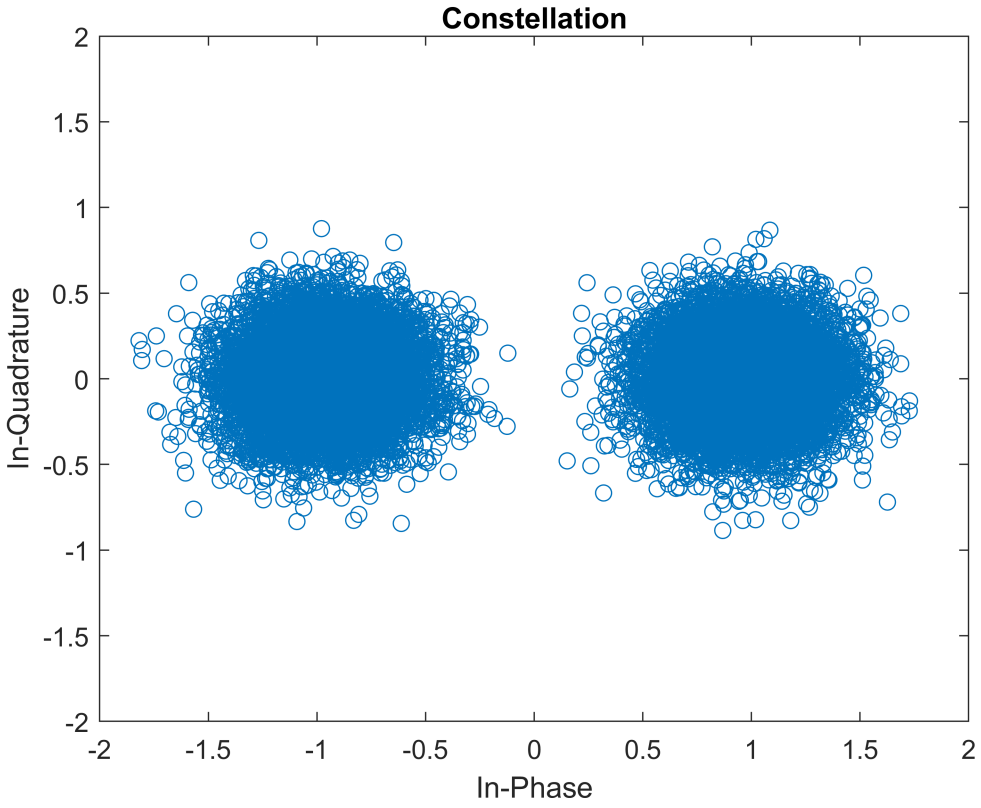
Overall, the error bit rate of 74 Hamming code and uncoded decreases with the increase of SNR, and 74 Hamming code can improve the SNR by 3db for the same case of error bit rate.The higher the signal-to-noise ratio, the more significant the signal quality improvement of Hamming code is.After simulating different number

of bits, it was found that the simulation results were partially missing because the number of bits simulated was too low

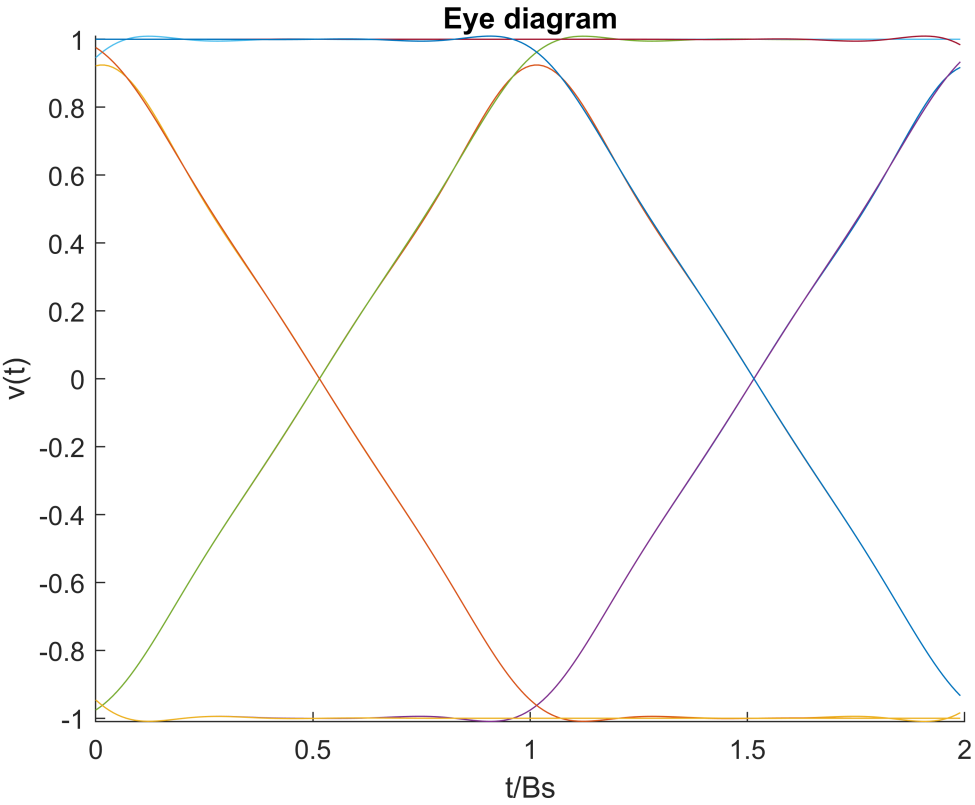
```
% ********* Spectrum plot
figure
PlotSpectrum_2023(s,fs);
```




```
% ********* Constellation
x=real(yk);
y=imag(yk);
figure
plot(x,y,'o'), axis ([-2 2 -2 2]);
title('Constellation');
xlabel('In-Phase');
ylabel('In-Quadrature');
```




```
% ********* Eye diagram
H=2; % number of observation symbols
[yi,yq,DelayQAM]=DeModQAM_2023(sr,fc,T,fs,psi); % demodulate the signal
vt0=conv(yi,h)/nsps;
mm=max(abs(vt0));
x1=0:H*nsps-1;
x2=x1./nsps;
figure;
hold on
for iii=1:10
x=x1+iii*nsps+delay-1;
y=real(vt0(x)); % plot the in-phase via
plot (x2,y), axis([0 H -mm mm]);
end
title('Eye diagram');
xlabel('t/Bs');
ylabel('v(t)');
```


Eye diagram
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
v(t)
0
0.5
1
1.5
2
t/Bs