QPSK 调制与解调（Matlab 仿真）

1. 一般在仿真的时候，大家都喜欢直接做等效基带仿真(类似于星座点的仿真)。

但实际要传，还是要传频带的波形信号。

2. 为了模拟真实的环境，先把基带信号经过一个自定义的信道，然后再做脉冲成型，上变频，加一点噪声 AWGN 进去。

3. 为了模拟同步，应该用专用的同步算法。但是这里的重点不在同步。所以用了很简单粗暴的办法。假装直接同步上了。

4. 为了造出不同步的结果，可以这样写 x_未同步 = [x(300:end); x; x]; 相当于循环发送，循环接收。

这是仿真。

Main

```
%%
% 单载波 QPSK 接收端
% 2017 年 5 月 17 日 18:02:56

clear;
close all;
clc

rand_seed = 0;
rand('seed',rand_seed);
randn('seed',rand_seed);

%%

% Set up parameters and signals.

M = 4; % Alphabet size for modulation
baud_rate = 100; % Baud rate
f_carrier1 = 75; % Carrier frequency
Nsym = 10000; % Number of symbols
```

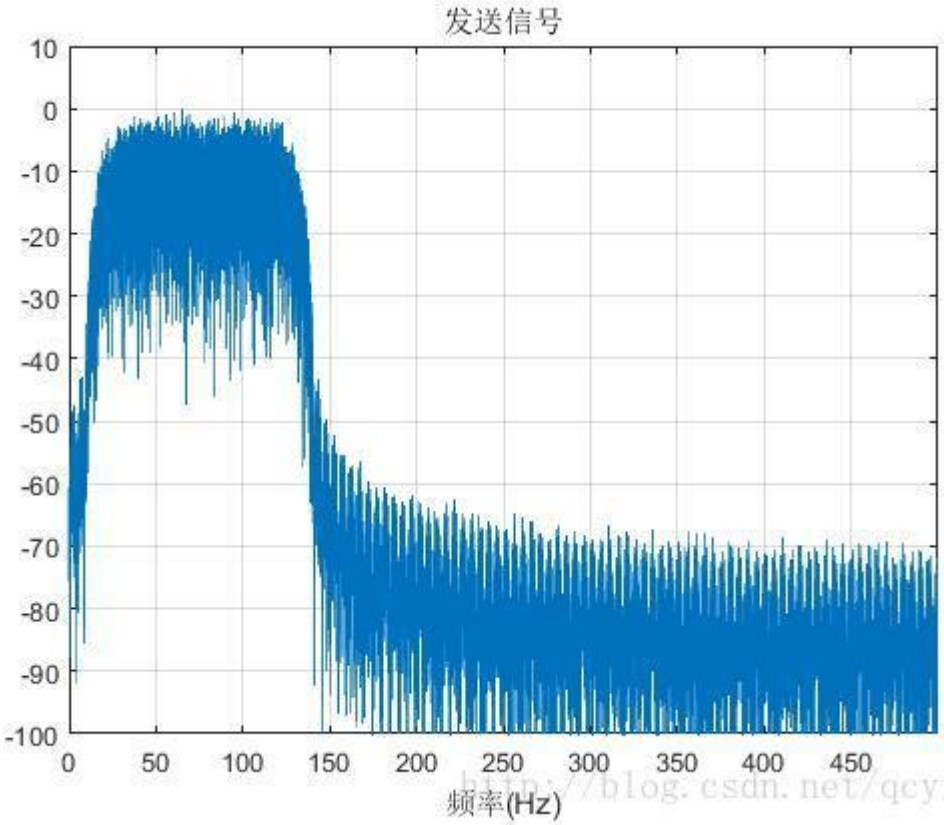
```
msg = randi([0 M-1],Nsym,1); % Random message
hMod = comm.RectangularQAMModulator(M);
modmsg = step(hMod,msg); % Modulate using QAM. % 映射后的基带信号
trainlen = 1000; % Length of training sequence

rolloff = .3; % 滚降因子
span = 20 ; % 截断长度
sps = 10;   % Samples per symbol
rrcFilter=rcosdesign(rolloff,span,sps,'sqrt'); %根升余弦滚降滤波器，‘sqrt’均方根升余弦；‘normal’升余弦

fs = baud_rate*sps; % 时间采样率,时间采样间隔为 1/fs 秒
Tsymbol=1/baud_rate;

% 2. 脉冲成型
% txSig = upfirdn(modmsg, rrcFilter, sps);   % 发送端的基带复波形信号

% chan = [1; .001];
chan = [.986; .845; .237; .123+.31i]; % Channel coefficients
% chan = [1 0.45 0.3+0.2i]; % Channel coefficients
filtmsg = filter(chan,1,modmsg); % Introduce channel distortion.（已经经过信道的畸变的基带复信号，星座点）
txSig = upfirdn(filtmsg, rrcFilter, sps);   % 发送端的基带复波形信号

txSig = awgn(txSig,20,'measured'); % Add AWGN

t = (0:1/fs:((length(txSig)-1)/fs)).';
T = t(end)+1/fs;
df = 1/T;
freq = -fs/2:df:fs/2-df;
cos1 = cos(2*pi*f_carrier1 * t);
sin1 = sin(2*pi*f_carrier1 * t);
x_upconv = real(txSig).* cos1 + imag(txSig) .* sin1;

%% === 接收端
x_training_wave = x_upconv;
x_training_msg = msg;
rxSig = [x_upconv(300:end) ; x_upconv];

% 1. 同步
x_resampled = resample(rxSig,1,1);
x_sync = sync_two_signals( x_resampled,x_training_wave,0);
```

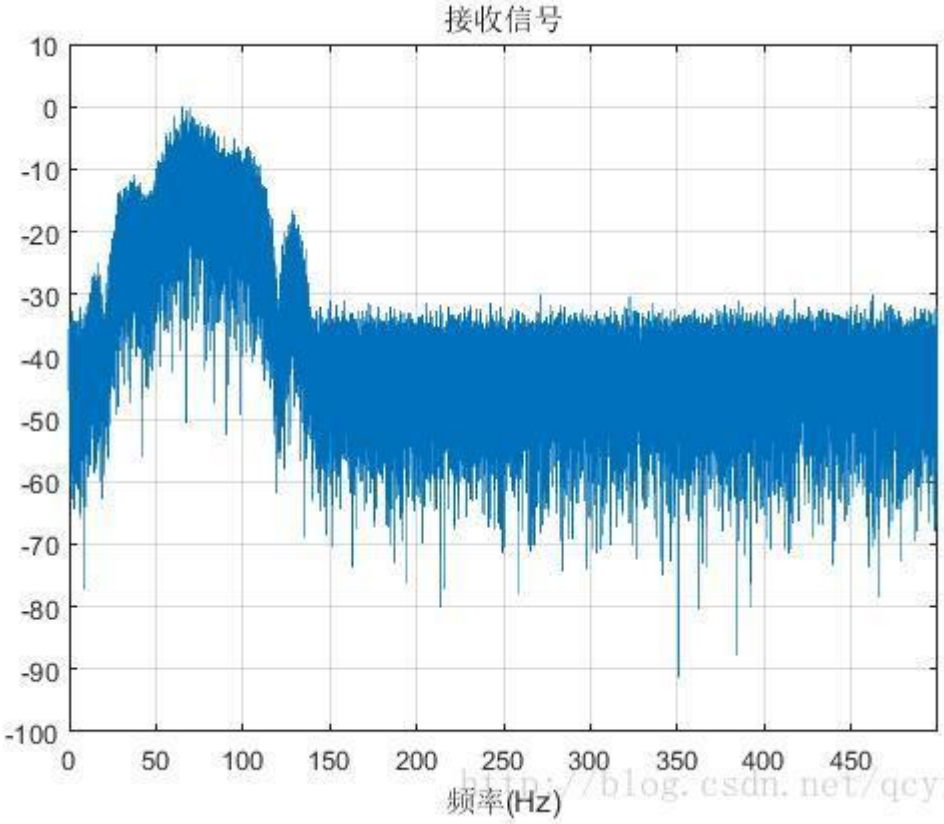
```
figure(2);
plot(freq,20*log10(abs(fftshift(fft(x_sync))/max(abs(fftshift(fft(x_sync)))))));
ylim([-100,10])
xlim([0,freq(end)])
grid on;
xlabel('频率(Hz)');
title('接收信号');

% 2. 下变频 + 匹配滤波
xi_dnconv = x_sync .* cos1;
xq_dnconv = x_sync .* sin1;
x_filtered = xi_dnconv + 1j * xq_dnconv;
rxFilt = upfirdn(x_filtered, rrcFilter, 1, sps);
rxFilt = rxFilt(span+1:end-span); % 这是接收端匹配滤波后的信号

% 3. 均衡
% eq1 = lineareq(6, lms(0.01)); % LMS
eq1 = lineareq(30, rls(0.99,0.01)); % Create an equalizer object. % 40 taps，RLS 算法，步长 0.99，自相关矩阵逆矩阵的初值 InvCorrInit 对角线上的元素
eq1.SigConst = step(hMod,(0:M-1)')'; % Set signal constellation. % 标准星座图
[symbolest,~] = equalize(eq1,rxFilt,x_training_msg(1:trainlen)); % Equalize. % 均衡器 obj，需要均衡的信号，训练序列

symbolest = symbolest ./ mean(abs(symbolest)) .* mean(abs(eq1.SigConst));

% Plot signals.
h = scatterplot(rxFilt,1,trainlen,'bx'); hold on;
scatterplot(symbolest,1,trainlen,'r.',h);
scatterplot(eq1.SigConst,1,0,'k*',h);
legend('Filtered signal','Equalized signal',...
    'Ideal signal constellation');
hold off;

% Compute error rates with equalization.
hDemod = comm.RectangularQAMDemodulator(M);
demodmsg = step(hDemod,symbolest); % Demodulate detected signal from equalizer.

% Create ErrorRate Calculator System object
serVec = step(comm.ErrorRate,msg(trainlen+1:end),demodmsg(trainlen+1:end));
srate = serVec(1)
snum = serVec(2)
% Convert integers to bits
hIntToBit = comm.IntegerToBit(log2(M));
Tx_bit = step(hIntToBit, msg(trainlen+1:end));
```

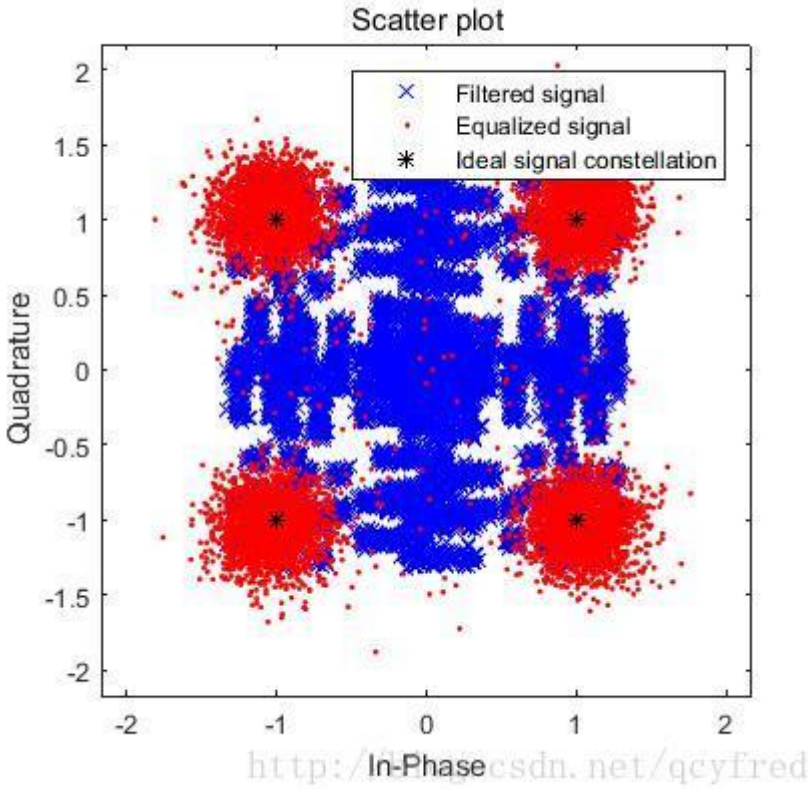
```
Rx_bit = step(hIntToBit, demodmsg(trainlen+1:end));
% Calculate BER
berVec = step(comm.ErrorRate,Rx_bit,Tx_bit);
brate = berVec(1)
bnum = berVec(2)
```

同步的代码

```
function x_sync = sync_two_signals( x_resampled,x_training_wave,idx )
% sync_two_signals( x_resampled,x_training_wave,idx )
% x_resampled：收到的信号
% x_training_wave：用发送的信号
% idx：要找同步上的第几段。idx = 1,2,3,...

[hicorrl,lagsil]=xcorr(x_resampled,x_training_wave);
[~,offsetindex]=max((hicorrl));
% offsetindex
for n = 2 : idx
    % figure;plot(lagsil,abs(hicorrl));
    hicorrl(offsetindex) = -inf;
    [~,offsetindex]=max(hicorrl);
end
% offsetindex
h=1;

x_sync=x_resampled(lagsil(offsetindex)+h:lagsil(offsetindex)+length(x_training_wave)+h-1);

end
```







误符号率

srate = 0.0041

错误符号数
snum = 37

误比特率

brate = 0.0021

错误比特数
bnum = 38