**OFDM系统仿真**



**学 院 电气自动化与信息工程**

**专 业 通信工程**

**年 级 2017级**

**姓 名 庞添天**

**学 号**  **3017234361**

**2020年 5 月 8 日**

目录

[模块代码 2](#_Toc39843980)

[Packet\_detection.m 2](#_Toc39843981)

[Frequency\_synchronization.m 4](#_Toc39843982)

[Fine\_time\_synchronization.m 7](#_Toc39843983)

[Channel\_estimation.m 9](#_Toc39843984)

[模块集成 12](#_Toc39843985)

[Integrated.m 12](#_Toc39843986)

[补充1 17](#_Toc39843987)

[整体仿真 18](#_Toc39843988)

[Complete.m 18](#_Toc39843989)

[补充2 24](#_Toc39843990)

[总结与体会 24](#_Toc39843991)

# 模块代码

## Packet\_detection.m

%% Preparation part

clear all;clc;

% sys para

gi=1/4;

fftlen = 64;

gilen = gi\*fftlen;

% training seq

ShortTrain = sqrt(13/6)\*[0,0,1+1i,0,0,0,-1-1i,0,0,0,1+1i,0,0,0,-1-1i,0,0,0,-1-1i,0,0,0,1+1i ...

0,0,0,0,0,0 ,-1-1i,0,0,0,-1-1i,0,0,0,1+1i,0,0,0,1+1i,0,0,0,1+1i,0,0,0,1+1i,0,0].';

NumberSTBlks = 10;

short\_train = tx\_freqd\_to\_timed(ShortTrain);

%plot(abs(short\_train));

short\_train\_blk = short\_train(1:16);

short\_train\_blks = repmat(short\_train\_blk,NumberSTBlks,1);

LongTrain = [1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,1,-1,-1,1,1 ,-1,1,-1,1,1,1,1,1,-1,-1,1,1,-1,1 ...

-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1].';

NumberLTBlks = 2;

long\_train = tx\_freqd\_to\_timed(LongTrain);

long\_train\_syms = [long\_train(fftlen-2\*gilen+1:fftlen,:);long\_train;long\_train];

%% Channel

h = zeros(gilen,1);

h(1) = 1;

h(5) = 0.5;

h(10) = 0.3;

start\_noise\_len = 500;

snr = 50;

%% Transmitter

tx = [short\_train\_blks;long\_train\_syms];

%% Pass channel

rx\_signal = filter(h,1,tx);

noise\_var = 1/(10^(snr/10))/2;

len = length(rx\_signal);

noise = sqrt(noise\_var) \* (randn(len,1) + 1i\*randn(len,1));

% add noise

rx\_signal = rx\_signal + noise;

start\_noise = sqrt(noise\_var) \* (randn(start\_noise\_len,1) + 1i\*randn(start\_noise\_len,1));

rx\_signal = [start\_noise;rx\_signal];

%% Receiver

search\_win = 700;

D = 16;

% Calculate delay correlation

delay\_xcorr = rx\_signal(1:search\_win).\*conj(rx\_signal(D+1:search\_win+D));

% Moving average of the delayed correlation

ma\_delay\_xcorr = abs(filter(ones(1,D),1,delay\_xcorr));

% Moving average of received power

ma\_rx\_pwr = filter(ones(1,D),1,abs(rx\_signal(D+1:search\_win+D)).^2);

% The decision variable

delay\_len = length(ma\_delay\_xcorr);

ma\_M = ma\_delay\_xcorr(1:delay\_len)./ma\_rx\_pwr(1:delay\_len);

% Remove delay samples

ma\_M(1:16) = [];

plot(ma\_M);grid;

threshold = 0.75;

%thres\_idx = find(ma\_M > threshold);

%if isempty(thres\_idx)

%thres\_idx = 1;

%else

%thres\_idx = thres\_idx(1)

%end

pre\_thres\_idx = find(ma\_M > threshold);

detect\_win = 12;

detect\_len = length(pre\_thres\_idx)-detect\_win;

for detect\_idx = 1:detect\_len

detect\_thres(detect\_idx) = mean(ma\_M(pre\_thres\_idx(detect\_idx):pre\_thres\_idx(detect\_idx+detect\_win)));

if detect\_thres(detect\_idx) > threshold

break

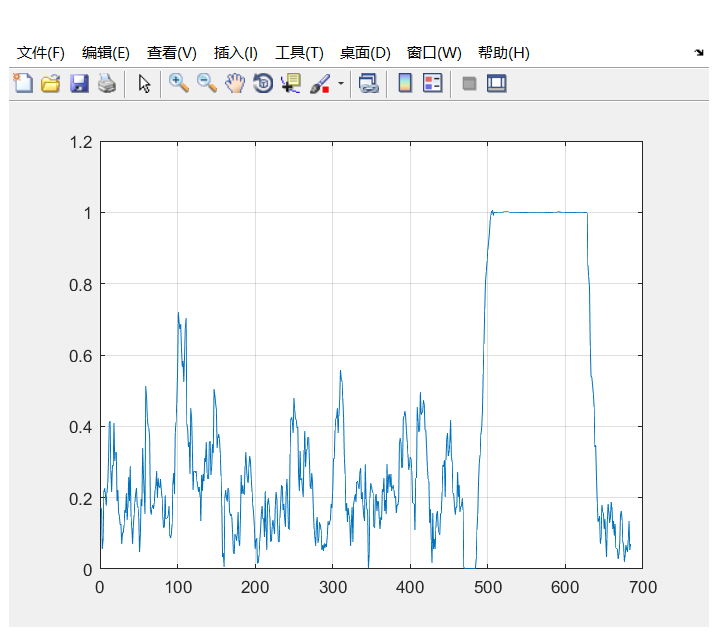
end

end

thres\_idx = pre\_thres\_idx(detect\_idx)

detected\_packet = rx\_signal(thres\_idx:length(rx\_signal));

在判断包头位置时，为排除起始噪声范围内可能的尖峰，采用了加窗平均的方式进行判决。下为 的曲线。可以看到已成功检测到包头位置在500左右（输出值为497）。



## Frequency\_synchronization.m

%% Preparation part

clear all;clc;

% sys para

gi=1/4;

fftlen = 64;

gilen = gi\*fftlen;

fs = 20e6;

% training seq

ShortTrain = sqrt(13/6)\*[0,0,1+1i,0,0,0,-1-1i,0,0,0,1+1i,0,0,0,-1-1i,0,0,0,-1-1i,0,0,0,1+1i ...

0,0,0,0,0,0 ,-1-1i,0,0,0,-1-1i,0,0,0,1+1i,0,0,0,1+1i,0,0,0,1+1i,0,0,0,1+1i,0,0].';

NumberSTBlks = 10;

short\_train = tx\_freqd\_to\_timed(ShortTrain);

%plot(abs(short\_train));

short\_train\_blk = short\_train(1:16);

short\_train\_blks = repmat(short\_train\_blk,NumberSTBlks,1);

LongTrain = [1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,1,-1,-1,1,1 ,-1,1,-1,1,1,1,1,1,-1,-1,1,1,-1,1 ...

-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1].';

NumberLTBlks = 2;

long\_train = tx\_freqd\_to\_timed(LongTrain);

long\_train\_syms = [long\_train(fftlen-2\*gilen+1:fftlen,:);long\_train;long\_train];

preamble = [short\_train\_blks;long\_train\_syms];

%% Channel

h = zeros(gilen,1);

h(1) = 1;

h(5) = 0.5;

h(10) = 0.3;

cfo = 0.2\*fs/fftlen

%% Loop start

snr = 5:5:30;

mse = zeros(1,length(snr));

pkt\_num = 1000;

for snr\_idx = 1:length(snr)

snr\_idx

est\_err = zeros(1,pkt\_num);

for pkt\_idx = 1:pkt\_num

%% Transmitter

tx = preamble;

%% Channel

rx\_signal = filter(h,1,tx);

noise\_var = 1/(10^(snr(snr\_idx)/10))/2;

len = length(rx\_signal);

noise = sqrt(noise\_var) \* (randn(len,1) + 1i\*randn(len,1));

% add noise

rx\_signal = rx\_signal + noise;

% add CFO

total\_length = length(rx\_signal);

t = [0:total\_length-1]/fs;

phase\_shift = exp(1i\*2\*pi\*cfo\*t).';

rx\_signal = rx\_signal.\*phase\_shift;

%% Receiver

pkt\_det\_offset = 10;

rlen = length(short\_train\_blks)-pkt\_det\_offset;

D = 16;

auto\_corr = rx\_signal(pkt\_det\_offset:pkt\_det\_offset+rlen-D).\*conj(rx\_signal(pkt\_det\_offset+D:pkt\_det\_offset+rlen));

% add all esti

auto\_corr\_mean = sum(auto\_corr);

freq\_est = -angle(auto\_corr\_mean)/(2\*D\*pi/fs);

est\_err(pkt\_idx) = (freq\_est-cfo)/cfo;

%radians\_per\_sample = 2\*pi\*freq\_est/fs;

%time\_base = 0:length(rx\_signal)-1;

%correction = exp(-1i\*(radians\_per\_sample)\*time\_base);

%out\_signal = rx\_signal.\*correction.';

end

mse(snr\_idx) = mean(abs(est\_err.^2));

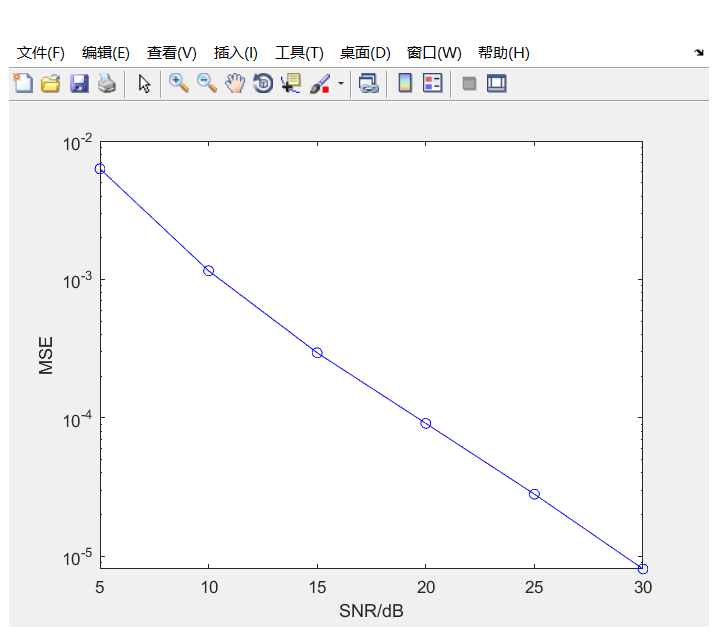
end

semilogy(snr,mse,'-bo');

xlabel('SNR/dB');

ylabel('MSE');

该步频域同步须在进行包检测后（即粗时间同步）开始，通过短训练序列的相关来完成检测并矫正频偏。初始频偏为0.2倍子载波带宽，下图为SNR-MSE曲线。可以看出，在信噪比较低的情况下也有较高精度。



## Fine\_time\_synchronization.m

%% Preparation part

clear all;clc;

% sys para

gi=1/4;

fftlen = 64;

gilen = gi\*fftlen;

fs = 20e6;

% training seq

ShortTrain = sqrt(13/6)\*[0,0,1+1i,0,0,0,-1-1i,0,0,0,1+1i,0,0,0,-1-1i,0,0,0,-1-1i,0,0,0,1+1i ...

0,0,0,0,0,0 ,-1-1i,0,0,0,-1-1i,0,0,0,1+1i,0,0,0,1+1i,0,0,0,1+1i,0,0,0,1+1i,0,0].';

NumberSTBlks = 10;

short\_train = tx\_freqd\_to\_timed(ShortTrain);

%plot(abs(short\_train));

short\_train\_blk = short\_train(1:16);

short\_train\_blks = repmat(short\_train\_blk,NumberSTBlks,1);

LongTrain = [1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,1,-1,-1,1,1 ,-1,1,-1,1,1,1,1,1,-1,-1,1,1,-1,1 ...

-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1].';

NumberLTBlks = 2;

long\_train = tx\_freqd\_to\_timed(LongTrain);

long\_train\_syms = [long\_train(fftlen-2\*gilen+1:fftlen,:);long\_train;long\_train];

preamble = [short\_train\_blks;long\_train\_syms];

%% Channel

h = zeros(gilen,1);

h(1) = 1;

h(5) = 0.5;

h(10) = 0.3;

%% Loop start

snr = -10:5:0;

mse = zeros(1,length(snr));

pkt\_num = 1000;

ideal\_start = 193;

for snr\_idx = 1:length(snr)

snr\_idx

est\_err = zeros(1,pkt\_num);

for pkt\_idx = 1:pkt\_num

%% Transmitter

tx = preamble;

%% Channel

rx\_signal = filter(h,1,tx);

noise\_var = 1/(10^(snr(snr\_idx)/10))/2;

len = length(rx\_signal);

noise = sqrt(noise\_var) \* (randn(len,1) + 1i\*randn(len,1));

% add noise

rx\_signal = rx\_signal + noise;

%% Receiver

start\_search = 150;

end\_search = 200;

time\_corr\_long = zeros(1,end\_search-start\_search+1);

for idx = start\_search:end\_search

time\_corr\_long(idx-start\_search+1) = sum((rx\_signal(idx:idx+63).\*conj(long\_train)));

end

[max\_corr,long\_search\_idx] = max(abs(time\_corr\_long));

fine\_time\_est = start\_search-1 + long\_search\_idx;

err\_est(pkt\_idx) = fine\_time\_est - ideal\_start;

end

mse(snr\_idx) = mean(abs(err\_est).^2);

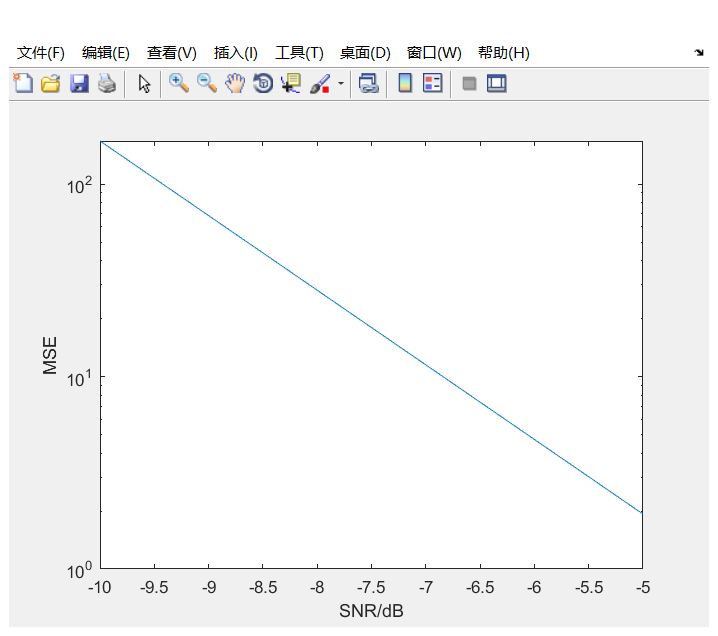
end

semilogy(snr,mse);

xlabel('SNR/dB');

ylabel('MSE');

精时间同步基于长训练序列进行处理，其范围较小，须在对CFO进行补偿后开始。但由下图可知，其精度几乎不受信噪比影响。



## Channel\_estimation.m

%% Preparation part

clear all;clc;

% sys para

gi=1/4;

fftlen = 64;

gilen = gi\*fftlen;

% training seq

ShortTrain = sqrt(13/6)\*[0,0,1+1i,0,0,0,-1-1i,0,0,0,1+1i,0,0,0,-1-1i,0,0,0,-1-1i,0,0,0,1+1i ...

0,0,0,0,0,0 ,-1-1i,0,0,0,-1-1i,0,0,0,1+1i,0,0,0,1+1i,0,0,0,1+1i,0,0,0,1+1i,0,0].';

NumberSTBlks = 10;

short\_train = tx\_freqd\_to\_timed(ShortTrain);

%plot(abs(short\_train));

short\_train\_blk = short\_train(1:16);

short\_train\_blks = repmat(short\_train\_blk,NumberSTBlks,1);

LongTrain = [1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,1,-1,-1,1,1 ,-1,1,-1,1,1,1,1,1,-1,-1,1,1,-1,1 ...

-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1].';

NumberLTBlks = 2;

long\_train = tx\_freqd\_to\_timed(LongTrain);

long\_train\_syms = [long\_train(fftlen-2\*gilen+1:fftlen,:);long\_train;long\_train];

preamble = [short\_train\_blks;long\_train\_syms];

%% Loop start

snr = -10:5:30;

mse = zeros(1,length(snr));

pkt\_num = 1000;

for snr\_idx = 1:length(snr)

snr\_idx

est\_err = zeros(1,pkt\_num);

for pkt\_idx = 1:pkt\_num

%% Transmitter

tx = preamble;

%% Channel

h = zeros(gilen,1);

h(1) = 1;

h(5) = 0.5;

h(10) = 0.3;

% index define

UsedSubcIdx = [7:32 34:59];

reorder = [33:64 1:32];

channel = fft(h,64);

channel(reorder) = channel;

channel = channel(UsedSubcIdx);

%% Transmitter

tx = preamble;

%% Pass channel

rx\_signal = filter(h,1,tx);

noise\_var = 1/(10^(snr(snr\_idx)/10))/2;

len = length(rx\_signal);

noise = sqrt(noise\_var) \* (randn(len,1) + 1i\*randn(len,1));

% add noise

rx\_signal = rx\_signal + noise;

%% Channel estimation

rx\_long = rx\_signal(161:end);

long\_tr\_syms = rx\_long(33:end);

long\_tr\_syms = reshape(long\_tr\_syms,64,2);

% to frequency domain

freq\_long\_tr = fft(long\_tr\_syms)/(64/sqrt(52));

freq\_long\_tr(reorder,:) = freq\_long\_tr;

freq\_tr\_syms = freq\_long\_tr(UsedSubcIdx,:);

channel\_est = mean(freq\_tr\_syms,2).\*conj(LongTrain);

err\_est(pkt\_idx) = mean(abs(channel\_est-channel).^2)/mean(abs(channel).^2);

end

mse(snr\_idx) = mean(abs(err\_est).^2);

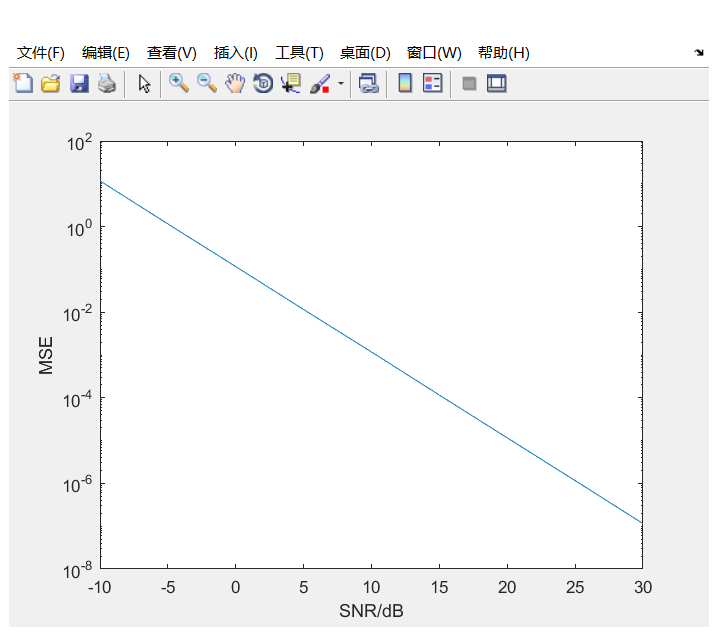
end

semilogy(snr,mse);

xlabel('SNR/dB');

ylabel('MSE');

信道估计同样基于长训练序列完成，但只对非零子载波部分做了估计（信道均衡时只对有用的子载波进行补偿）。下图是信道估计的SNR-MSE曲线，可以看到信道的估计同样无需较高信噪比就有很好的精度。



# 模块集成

## Integrated.m

%% Preparation part

clear all;clc;

% sys para

gi=1/4;

fftlen = 64;

gilen = gi\*fftlen;

fs = 20e6;

% training seq

ShortTrain = sqrt(13/6)\*[0,0,1+1i,0,0,0,-1-1i,0,0,0,1+1i,0,0,0,-1-1i,0,0,0,-1-1i,0,0,0,1+1i ...

0,0,0,0,0,0 ,-1-1i,0,0,0,-1-1i,0,0,0,1+1i,0,0,0,1+1i,0,0,0,1+1i,0,0,0,1+1i,0,0].';

NumberSTBlks = 10;

short\_train = tx\_freqd\_to\_timed(ShortTrain);

%plot(abs(short\_train));

short\_train\_blk = short\_train(1:16);

short\_train\_blks = repmat(short\_train\_blk,NumberSTBlks,1);

LongTrain = [1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,1,-1,-1,1,1 ,-1,1,-1,1,1,1,1,1,-1,-1,1,1,-1,1 ...

-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1].';

NumberLTBlks = 2;

long\_train = tx\_freqd\_to\_timed(LongTrain);

long\_train\_syms = [long\_train(fftlen-2\*gilen+1:fftlen,:);long\_train;long\_train];

preamble = [short\_train\_blks;long\_train\_syms];

%% Loop start

snr = 8:2:30;

pkt\_num = 1000;

freq\_mse = zeros(1,length(snr));

time\_mse = zeros(1,length(snr));

channel\_mse = zeros(1,length(snr));

for snr\_idx = 1:length(snr)

snr\_idx

freq\_est\_err = zeros(1,pkt\_num);

time\_est\_err = zeros(1,pkt\_num);

channel\_est\_err = zeros(1,pkt\_num);

for pkt\_idx = 1:pkt\_num

%% Transmitter

tx = preamble;

%% Channel

h = zeros(gilen,1);

h(1) = 1;

h(5) = 0.5;

h(10) = 0.3;

start\_noise\_len = 500;

cfo = 0.2\*fs/fftlen;

ideal\_start = 193;

% index define

UsedSubcIdx = [7:32 34:59];

reorder = [33:64 1:32];

channel = fft(h,64);

channel(reorder) = channel;

channel = channel(UsedSubcIdx);

rx\_signal = filter(h,1,tx);

noise\_var = 1/(10^(snr(snr\_idx)/10))/2;

len = length(rx\_signal);

noise = sqrt(noise\_var) \* (randn(len,1) + 1i\*randn(len,1));

start\_noise = sqrt(noise\_var) \* (randn(start\_noise\_len,1) + 1i\*randn(start\_noise\_len,1));

% add noise

rx\_signal = rx\_signal + noise;

rx\_signal = [start\_noise;rx\_signal];

% add CFO

total\_length = length(rx\_signal);

t = [0:total\_length-1]/fs;

phase\_shift = exp(1i\*2\*pi\*cfo\*t).';

rx\_signal = rx\_signal.\*phase\_shift;

%% Receiver

% Packet detect

search\_win = 700;

D = 16;

delay\_xcorr = rx\_signal(1:search\_win).\*conj(rx\_signal(D+1:search\_win+D));

ma\_delay\_xcorr = abs(filter(ones(1,D),1,delay\_xcorr));

ma\_rx\_pwr = filter(ones(1,D),1,abs(rx\_signal(D+1:search\_win+D)).^2);

delay\_len = length(ma\_delay\_xcorr);

ma\_M = ma\_delay\_xcorr(1:delay\_len)./ma\_rx\_pwr(1:delay\_len);

ma\_M(1:16) = [];

threshold = 0.75;

pre\_thres\_idx = find(ma\_M > threshold);

detect\_win = 12;

detect\_len = length(pre\_thres\_idx)-detect\_win;

for detect\_idx = 1:detect\_len

detect\_thres(detect\_idx) = mean(ma\_M(pre\_thres\_idx(detect\_idx):pre\_thres\_idx(detect\_idx+detect\_win)));

if detect\_thres(detect\_idx) > threshold

break

end

end

thres\_idx = pre\_thres\_idx(detect\_idx);

detected\_packet = rx\_signal(thres\_idx:length(rx\_signal));

% Frequency sync

pkt\_det\_offset = 10;

rlen = length(short\_train\_blks)-pkt\_det\_offset;

auto\_corr = detected\_packet(pkt\_det\_offset:pkt\_det\_offset+rlen-D).\*conj(detected\_packet(pkt\_det\_offset+D:pkt\_det\_offset+rlen));

auto\_corr\_mean = sum(auto\_corr);

freq\_est = -angle(auto\_corr\_mean)/(2\*D\*pi/fs);

freq\_est\_err(pkt\_idx) = (freq\_est-cfo)/cfo;

radians\_per\_sample = 2\*pi\*freq\_est/fs;

time\_base = 0:length(detected\_packet)-1;

correction = exp(-1i\*(radians\_per\_sample)\*time\_base);

out\_signal = detected\_packet.\*correction.';

% Fine Time sync

start\_search = 150;

end\_search = 200;

time\_corr\_long = zeros(1,end\_search-start\_search+1);

for idx = start\_search:end\_search

time\_corr\_long(idx-start\_search+1) = sum((out\_signal(idx:idx+63).\*conj(long\_train)));

end

[max\_corr,long\_search\_idx] = max(abs(time\_corr\_long));

fine\_time\_est = start\_search-1 + long\_search\_idx;

time\_err\_est(pkt\_idx) = fine\_time\_est - ideal\_start;

% Channel Estimation

long\_tr\_syms = out\_signal(fine\_time\_est:end);

long\_tr\_syms = reshape(long\_tr\_syms,64,2);

% to frequency domain

freq\_long\_tr = fft(long\_tr\_syms)/(64/sqrt(52));

freq\_long\_tr(reorder,:) = freq\_long\_tr;

freq\_tr\_syms = freq\_long\_tr(UsedSubcIdx,:);

channel\_est = mean(freq\_tr\_syms,2).\*conj(LongTrain);

channel\_err\_est(pkt\_idx) = mean(abs(channel\_est-channel).^2)/mean(abs(channel).^2);

end

freq\_mse(snr\_idx) = mean(abs(freq\_est\_err.^2));

time\_mse(snr\_idx) = mean(abs(time\_est\_err.^2));

channel\_mse(snr\_idx) = mean(abs(channel\_est\_err.^2));

end

subplot(1,3,1);

xlabel('SNR/dB');

ylabel('MSE');

plot(snr,freq\_mse)

title('freq mse')

subplot(1,3,2);

xlabel('SNR/dB');

ylabel('MSE');

plot(snr,time\_mse)

title('time mse')

subplot(1,3,3);

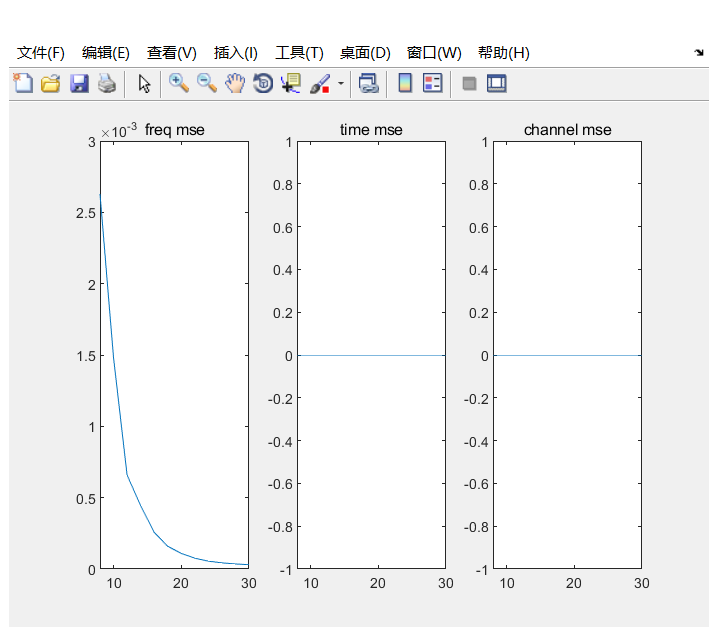
xlabel('SNR/dB');

ylabel('MSE');

plot(snr,channel\_mse)

title('channel mse')

上方代码为包检测、频域同步、时间精同步、信道估计四个模块的集合。在试验初期，我曾通过降低发包数和数据块个数的方式使代码得以运行。但那样只是误差积累没有达到解调的极限，并不可取。在通过加窗平均的方式对包检测进行改良之后得到下图仿真结果。可以看到，由于没有插入导频，频移还是会逐渐积累，对精度造成影响，但还在可接受的范围之内。



## 补充1

原有的包检测方式部分代码：

delay\_xcorr = rx\_signal(1:search\_win+2\*D).\*conj(rx\_signal(D+1:search\_win+3\*D));

ma\_delay\_xcorr = abs(filter(ones(1,2\*D),1,delay\_xcorr));

ma\_rx\_pwr = filter(ones(1,2\*D),1,abs(rx\_signal(D+1:search\_win+3\*D)).^2);

delay\_len = length(ma\_delay\_xcorr);

ma\_M = ma\_delay\_xcorr(1:delay\_len)./ma\_rx\_pwr(1:delay\_len);

ma\_M(1:16) = [];

threshold = 0.75;

thres\_idx = find(ma\_M > threshold);

thres\_idx = find(ma\_M > threshold);

if isempty(thres\_idx)

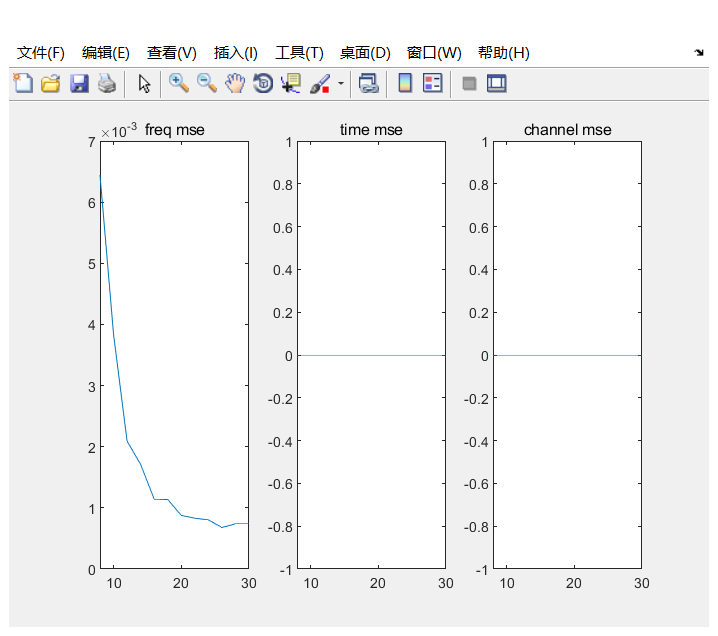
thres\_idx = 1;

else

thres\_idx = thres\_idx(1);

end

将发包数量从1000降到100，可得到如下仿真结果：



此时判定的thres\_idx为510，而修改后的代码判定为497。从上图与前面的对比也可以看出，修改包检测方式后频域同步误差下降明显。

# 整体仿真

## Complete.m

clear all; clc;

%% \*\*\*\*\*\*\*\*\*\*\*\*\*\* System parameters \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ml = 2; % Modulation level: 2--4QAM; 4--16QAM; 6--64QAM

NormFactor = sqrt(2/3\*(ml.^2-1));

gi = 1/4; % Guard interval

fftlen = 64;

gilen = gi\*fftlen; % Length of guard interval (samples)

blocklen = fftlen + gilen; % total length of the block with CP

non\_fftlen = 52;

% conv

trellis = poly2trellis(7,[133 171]);

code\_rate = 1/2;

tb = 7\*5;

bits\_per\_sym = code\_rate\*non\_fftlen\*ml;

NumSyms = 50;

TotalNumBits = bits\_per\_sym\*NumSyms;

fs = 20e6;

% training seq

ShortTrain = sqrt(13/6)\*[0,0,1+1i,0,0,0,-1-1i,0,0,0,1+1i,0,0,0,-1-1i,0,0,0,-1-1i,0,0,0,1+1i ...

0,0,0,0,0,0 ,-1-1i,0,0,0,-1-1i,0,0,0,1+1i,0,0,0,1+1i,0,0,0,1+1i,0,0,0,1+1i,0,0].';

NumberSTBlks = 10;

short\_train = tx\_freqd\_to\_timed(ShortTrain);

short\_train\_blk = short\_train(1:16);

short\_train\_blks = repmat(short\_train\_blk,NumberSTBlks,1);

LongTrain = [1,1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,1,-1,-1,1,1 ,-1,1,-1,1,1,1,1,1,-1,-1,1,1,-1,1 ...

-1,1,-1,-1,-1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,1,1,1].';

NumberLTBlks = 2;

long\_train = tx\_freqd\_to\_timed(LongTrain);

long\_train\_syms = [long\_train(fftlen-2\*gilen+1:fftlen,:);long\_train;long\_train];

preamble = [short\_train\_blks;long\_train\_syms];

% index define

%DataSubcPatt = [1:5 7:19 21:26 27:32 34:46 48:52]';

%DataSubcIdx = [7:11 13:25 27:32 34:39 41:53 55:59];

UsedSubcIdx = [7:32 34:59];

reorder = [33:64 1:32];

%% \*\*\*\*\*\*\*Loop start\*\*\*\*\*\*\*\*\*\*\*

snr = 10:5:30;

ber = zeros(1,length(snr));

pkt\_num = 1000;

for snr\_idx = 1:length(snr)

err = 0;

snr\_idx

for pkt\_idx = 1:pkt\_num

%% \*\*\*\*\*\*\*\*\*Transmitter\*\*\*\*\*\*\*\*\*

% Generate the information bits

inf\_bits = randn(1,TotalNumBits)>0;

% Encoding

coded\_bits = convenc(inf\_bits,trellis);

%info\_bits = 2\*coded\_bits-1;

% Modulate

paradata = reshape(coded\_bits,length(coded\_bits)/ml,ml);

mod\_ofdm\_syms = qammod(bi2de(paradata),2^ml)./NormFactor;

% IFFT

mod\_ofdm\_syms = reshape(mod\_ofdm\_syms,non\_fftlen,NumSyms);

mod\_ofdm\_syms = [zeros(6,NumSyms);mod\_ofdm\_syms(1:26,1:NumSyms);zeros(1,NumSyms);mod\_ofdm\_syms(27:52,1:NumSyms);zeros(5,NumSyms)];

mod\_ofdm\_syms = mod\_ofdm\_syms(reorder,:);

tx\_blks = sqrt(fftlen)\*ifft(mod\_ofdm\_syms);

% Add CP

tx\_frames = [tx\_blks(fftlen-gilen+1:fftlen,:); tx\_blks];

% P/S

trans\_bits = reshape(tx\_frames,NumSyms\*blocklen,1);

% Transmit

tx = [preamble;trans\_bits];

%% \*\*\*\*\*\*\*\*\*\*\*\*Channel\*\*\*\*\*\*\*\*\*\*\*\*\*

h = zeros(gilen,1);

h(1) = 1;

h(5) = 0.5;

h(10) = 0.3;

start\_noise\_len = 500;

cfo = 0.2\*fs/fftlen;

ideal\_start = 193;

rx\_signal = filter(h,1,tx);

noise\_var = 1/(10^(snr(snr\_idx)/10))/2;

len = length(rx\_signal);

noise = sqrt(noise\_var) \* (randn(len,1) + 1i\*randn(len,1));

start\_noise = sqrt(noise\_var) \* (randn(start\_noise\_len,1) + 1i\*randn(start\_noise\_len,1));

% add noise

rx\_signal = rx\_signal + noise;

rx\_signal = [start\_noise;rx\_signal];

% add CFO

total\_length = length(rx\_signal);

t = [0:total\_length-1]/fs;

phase\_shift = exp(1i\*2\*pi\*cfo\*t).';

rx\_signal = rx\_signal.\*phase\_shift;

%% \*\*\*\*\*\*\*\*\*\*\*\*Receiver\*\*\*\*\*\*\*\*\*\*\*\*

% Packet detect

search\_win = 700;

D = 16;

delay\_xcorr = rx\_signal(1:search\_win).\*conj(rx\_signal(D+1:search\_win+D));

ma\_delay\_xcorr = abs(filter(ones(1,D),1,delay\_xcorr));

ma\_rx\_pwr = filter(ones(1,D),1,abs(rx\_signal(D+1:search\_win+D)).^2);

delay\_len = length(ma\_delay\_xcorr);

ma\_M = ma\_delay\_xcorr(1:delay\_len)./ma\_rx\_pwr(1:delay\_len);

ma\_M(1:16) = [];

threshold = 0.75;

pre\_thres\_idx = find(ma\_M > threshold);

detect\_win = 12;

detect\_len = length(pre\_thres\_idx)-detect\_win;

for detect\_idx = 1:detect\_len

detect\_thres(detect\_idx) = mean(ma\_M(pre\_thres\_idx(detect\_idx):pre\_thres\_idx(detect\_idx+detect\_win)));

if detect\_thres(detect\_idx) > threshold

break

end

end

thres\_idx = pre\_thres\_idx(detect\_idx);

detected\_packet = rx\_signal(thres\_idx:length(rx\_signal));

% Frequency sync

pkt\_det\_offset = 10;

rlen = length(short\_train\_blks)-pkt\_det\_offset;

auto\_corr = detected\_packet(pkt\_det\_offset:pkt\_det\_offset+rlen-D).\*conj(detected\_packet(pkt\_det\_offset+D:pkt\_det\_offset+rlen));

auto\_corr\_mean = sum(auto\_corr);

freq\_est = -angle(auto\_corr\_mean)/(2\*D\*pi/fs);

radians\_per\_sample = 2\*pi\*freq\_est/fs;

time\_base = 0:length(detected\_packet)-1;

correction = exp(-1i\*(radians\_per\_sample)\*time\_base);

out\_signal = detected\_packet.\*correction.';

% Fine Time sync

start\_search = 150;

end\_search = 200;

time\_corr\_long = zeros(1,end\_search-start\_search+1);

for idx = start\_search:end\_search

time\_corr\_long(idx-start\_search+1) = sum((out\_signal(idx:idx+63).\*conj(long\_train)));

end

[max\_corr,long\_search\_idx] = max(abs(time\_corr\_long));

fine\_time\_est = start\_search-1 + long\_search\_idx;

% Channel Estimation

long\_tr\_syms = out\_signal(fine\_time\_est:fine\_time\_est+127);

long\_tr\_syms = reshape(long\_tr\_syms,64,2);

freq\_long\_tr = fft(long\_tr\_syms)/(64/sqrt(52)); % to frequency domain

freq\_long\_tr(reorder,:) = freq\_long\_tr;

freq\_tr\_syms = freq\_long\_tr(UsedSubcIdx,:);

channel\_est = mean(freq\_tr\_syms,2).\*conj(LongTrain);

% remove preamble

data\_syms = out\_signal(fine\_time\_est+128:end);

data\_syms = reshape(data\_syms,80,NumSyms);

% remove gi

data\_syms(1:16,:) = [];

freq\_data = fft(data\_syms)/(64/sqrt(52));

freq\_data(reorder,:) = freq\_data;

% select data carriers

freq\_data\_syms = freq\_data(UsedSubcIdx,:);

% channel equalization

chan\_corr\_mat = repmat(channel\_est,1,size(freq\_data\_syms,2));

freq\_data\_syms = freq\_data\_syms.\*conj(chan\_corr\_mat);

chan\_sq\_amp = abs(channel\_est).^2;

chan\_sq\_amp\_mtx = repmat(chan\_sq\_amp,1,size(freq\_data\_syms,2));

data\_syms\_out = freq\_data\_syms./chan\_sq\_amp\_mtx;

Data\_seq = reshape(data\_syms\_out,52\*NumSyms,1).\*NormFactor;

% demodulate

DemodSeq = de2bi(qamdemod(Data\_seq,2^ml),ml);

SerialBits = reshape(DemodSeq,size(DemodSeq,1)\*ml,1).';

% decode

hard\_decision = SerialBits > 0;

DecodedBits = vitdec(hard\_decision,trellis,tb,'trunc','hard');

err = err+sum(abs(DecodedBits-inf\_bits));

end

ber(snr\_idx) = err/(TotalNumBits\*pkt\_num);

end

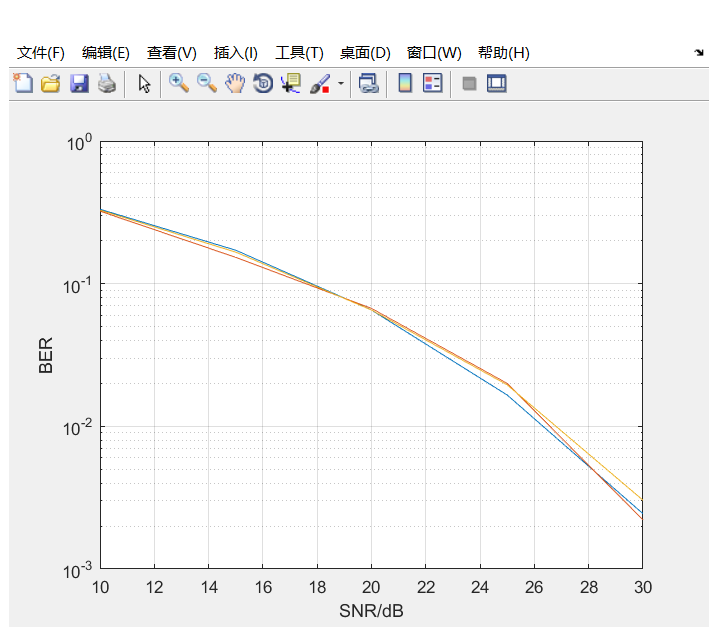
semilogy(snr,ber);

xlabel('SNR/dB');

ylabel('BER');

grid;hold on;

由于我将52个非零子载波全部用于承载数据，并未插入导频并及时进行相位补偿，所以频移误差还是会积累。下面为整体仿真误码率曲线，可以看出系统的可靠性应还需导频的加入和相位补偿来提升。



## 补充2

在完成实验后的尝试中加入交织与解交织，对误码率的改善并无明显作用。这可能是因为本次实验中构建的信道均为随机错误，而交织的主要作用是将突发错误转化为随机错误从而提高通信系统可靠性。

代码片段：

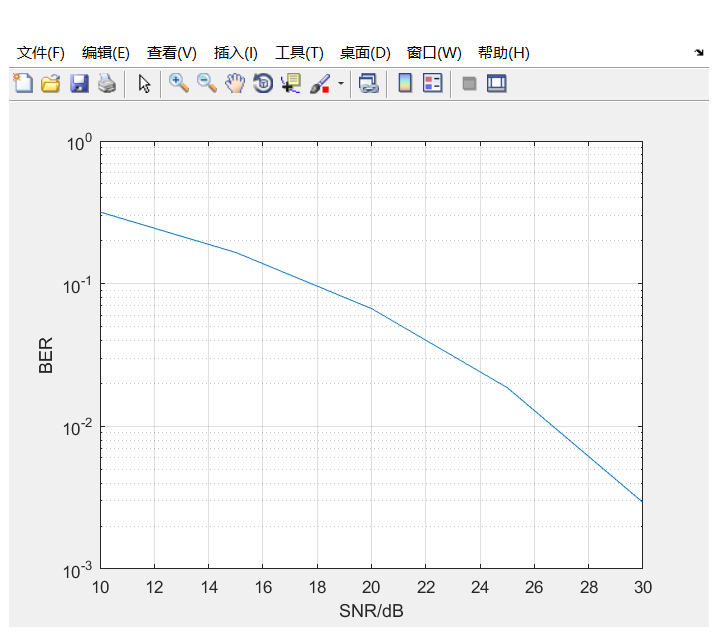
% Interleave （加在发射端编码和调制中间）

interleaved\_bits = tx\_interleaver(coded\_bits,non\_fftlen, ml);

% De-interleave （加在接收端解调和解码中间）

deint\_bits = rx\_deinterleave(SerialBits, non\_fftlen, ml);

仿真结果：



# 总结与体会

首先，我想感激为我解答无数问题的高镇老师。没有他的付出，我很难完成这次实验。本次OFDM仿真是对所学知识的一次总结。之前除了数字信号处理这门课外，几乎没有接触过Matlab语言编程，更多是用similink去仿真。不得不承认，Matlab对数据的操作和可视化是我接触过的语言里最好的，这点在调试代码、对数据进行查看时体现的尤为明显。实验中，信道的各项参数与对应的操作方法，模块的算法与前后的衔接整合等等，无不印证着我们在课上所学的理论知识，每一条曲线背后都是无数次的代码修改。真正动手去实践，令我对802.11a的帧结构有了更直观的体会，也对OFDM系统收发的步骤有了更深入的理解。