clc;

clear;

%原理参考：https://zhuanlan.zhihu.com/p/57967971

%% 参数设置

N\_sc=52; %系统子载波数（不包括直流载波）、number of subcarrierA

N\_fft=64; % FFT 长度

N\_cp=16; % 循环前缀长度、Cyclic prefix

N\_symbo=N\_fft+N\_cp; % 1个完整OFDM符号长度

N\_c=53; % 包含直流载波的总的子载波数、number of carriers

M=4; %4PSK调制

SNR=0:1:20; %仿真信噪比

N\_frm=10; % 每种信噪比下的仿真帧数、frame

Nd=6; % 每帧包含的OFDM符号数

P\_f\_inter=8; %导频间隔 一共8个导频

data\_station=[]; %导频位置

L=7; %卷积码约束长度

tblen=6\*L; %Viterbi译码器回溯深度

stage = 3; % m序列的阶数

ptap1 = [1 3]; % m序列的寄存器连接方式

regi1 = [1 1 1]; % m序列的寄存器初始值

alpha=0.3 %

%% 基带数据数据产生

P\_data=randi([0 1],1,N\_sc\*Nd\*N\_frm);

%数据量=仿真帧数×每帧含OFDM符号数×子载波数

%% 信道编码（卷积码、或交织器）

%卷积码：前向纠错非线性码

%交织：使突发错误最大限度的分散化

trellis = poly2trellis(7,[133 171]); %(2,1,7)卷积编码

code\_data=convenc(P\_data,trellis);

%% qpsk调制

data\_temp1= reshape(code\_data,log2(M),[])'; %以每组2比特进行分组，M=4

data\_temp2= bi2de(data\_temp1); %二进制转化为十进制

modu\_data=pskmod(data\_temp2,M,pi/M); % 4PSK调制

scatterplot(modu\_data);

title('QPSK调制星座图') %星座图(也可以取实部用plot函数)

%% 扩频

%————————————————————————————————————————————————————————%

%扩频通信信号所占有的频带宽度远大于所传信息必需的最小带宽

%根据香农定理，扩频通信就是用宽带传输技术来换取信噪比上的好处，这就是扩频通信的基本思想和理论依据。

%扩频就是将一系列正交的码字与基带调制信号内积

%扩频后数字频率变成了原来的m倍。码片数量 = 2（符号数）\* m（扩频系数）

%————————————————————————————————————————————————————————%

code = mseq(stage,ptap1,regi1,N\_sc); % 扩频码的生成

code = code \* 2 - 1; %将1、0变换为1、-1

modu\_data=reshape(modu\_data,N\_sc,length(modu\_data)/N\_sc);

spread\_data = spread(modu\_data,code); % 扩频

spread\_data=reshape(spread\_data,[],1);

%% 插入导频

P\_f=3+3\*1i; %Pilot frequency

P\_f\_station=[1:P\_f\_inter:N\_fft];%导频位置

pilot\_num=length(P\_f\_station);%导频数量

for img=1:N\_fft %数据位置

if mod(img,P\_f\_inter)~=1 %mod(a,b)就是求的是a除以b的余数

data\_station=[data\_station,img];

end

end

data\_row=length(data\_station);

data\_col=ceil(length(spread\_data)/data\_row);

pilot\_seq=ones(pilot\_num,data\_col)\*P\_f;%将导频放入矩阵

data=zeros(N\_fft,data\_col);%预设整个矩阵

data(P\_f\_station(1:end),:)=pilot\_seq;%对pilot\_seq按行取

if data\_row\*data\_col>length(spread\_data)

data2=[spread\_data;zeros(data\_row\*data\_col-length(spread\_data),1)];%将数据矩阵补齐，补0是虚载频~

else data2=spread\_data;

end;

%% 串并转换

data\_seq=reshape(data2,data\_row,data\_col);

data(data\_station(1:end),:)=data\_seq;%将导频与数据合并

%% IFFT

ifft\_data=ifft(data);

figure(2)

subplot(3,1,1);

plot(abs(ifft\_data(:,1)),'b');

title('原始单个OFDM符号');

xlabel('Time');

ylabel('Amplitude');

%% 插入保护间隔、循环前缀

Tx\_cd=[ifft\_data(N\_fft-N\_cp+1:end,:);ifft\_data];%把ifft的末尾N\_cp个数补充到最前面

subplot(3,1,2);

plot(abs(Tx\_cd(:,1)));

xlabel('Time');

ylabel('Amplitude');

title('加CP的单个OFDM符号');

%% 加窗

Tx\_window=zeros(size(Tx\_cd));

% 通过矩阵点乘

Tx\_window=Tx\_cd.\*repmat(rcoswindow(alpha,size(Tx\_cd,1)),1,390);%390是矩阵Tx\_cd的维度(80\*390)

subplot(3,1,3);

plot(abs(Tx\_window(:,1)))

title('加窗后的单个OFDM符号')

xlabel('Time');

ylabel('Amplitude');

%% 信号频谱

%未加窗信号频谱

%figure % 归一化

figure(3);

orgin\_aver\_power = 20\*log10(mean(abs(fft(Tx\_cd'))));

subplot(2,1,1)

plot((1:length(orgin\_aver\_power))/length(orgin\_aver\_power),orgin\_aver\_power)

hold on

plot(0:1/length(orgin\_aver\_power):1 ,-35,'rd')

hold on

axis([0 1 -40 max(orgin\_aver\_power)])

grid on

title('未加窗信号频谱')

%加窗信号频谱

orgin\_aver\_power = 20\*log10(mean(abs(fft(Tx\_window'))));

subplot(2,1,2)

plot((1:length(orgin\_aver\_power))/length(orgin\_aver\_power),orgin\_aver\_power)

hold on

plot(0:1/length(orgin\_aver\_power):1 ,-35,'rd')

hold on

axis([0 1 -40 max(orgin\_aver\_power)])

grid on

title('加窗信号频谱')

%% 并串转换

Tx\_data=reshape(Tx\_window,[],1);%由于传输需要

%% 信道（信道均衡1、信道均衡2）

%预设好误码率的变量矩阵

Ber\_single=zeros(1,length(SNR)); %单径误码率矩阵设置 信道均衡1

Ber2\_single=zeros(1,length(SNR));

Ber\_mult=zeros(1,length(SNR));%多径误码率矩阵设置 信道均衡1

Ber2\_mult=zeros(1,length(SNR));

Ber\_spline=zeros(1,length(SNR));%单径误码率矩阵设置 信道均衡2

Ber2\_spline=zeros(1,length(SNR));

Ber\_offset=zeros(1,length(SNR)); %多径误码率矩阵设置 信道均衡2 有频率补偿

Ber2\_offset=zeros(1,length(SNR));

Tx\_data = reshape(Tx\_window,1,[]); % 变成时域一个完整信号，待传输

%signal\_origin = reshape(Tx\_window,1,[]); % 未加窗完整信号

mult\_path\_am = [1 0.2 0.1]; % 多径幅度

mult\_path\_time1=randi([1,50]); % 多径随机时延1-50

mult\_path\_time2=randi([1,50]);

path2 = 0.2\*[zeros(1,mult\_path\_time1) Tx\_data(1:end-mult\_path\_time1) ];

path3 = 0.1\*[zeros(1,mult\_path\_time2) Tx\_data(1:end-mult\_path\_time2) ];

Tx\_mult = Tx\_data + path2 + path3; % 多径信号

figure('menubar','none');

subplot(2,1,1);

plot(real(Tx\_mult) )

title('多径下OFDM信号')

xlabel('Time/samples')

ylabel('Amplitude')

subplot(2,1,2)

plot(real(Tx\_data) )

title('单径下OFDM信号')

xlabel('Time/samples')

ylabel('Amplitude')

%% 单径/多径信道后处理

for jj=1:length(SNR)

channe\_single=awgn(Tx\_data,SNR(jj),'measured');

channe\_mult=awgn(Tx\_mult ,SNR(jj),'measured');

%添加高斯白噪声

%y = awgn(x,SNR,SIGPOWER)

%如果SIGPOWER为'measured'，则函数将在加入噪声之前测定信号强度。

%% 串并转换

Rx\_data1\_single=reshape(channe\_single,N\_fft+N\_cp,[]);

Rx\_data1\_mult=reshape(channe\_mult,N\_fft+N\_cp,[]);

%% 去掉保护间隔、循环前缀

Rx\_data2\_single=Rx\_data1\_single(N\_cp+1:end,:);

Rx\_data2\_mult=Rx\_data1\_mult(N\_cp+1:end,:);

%% FFT

fft\_data\_single=fft(Rx\_data2\_single);

fft\_data\_mult=fft(Rx\_data2\_mult);

%% 信道估计与插值线性（信道均衡1）

%单径信道

data3\_single=fft\_data\_single(1:N\_fft,:);

Rx\_pilot\_single=data3\_single(P\_f\_station(1:end),:); %接收到的导频

h\_single=Rx\_pilot\_single./pilot\_seq;

H\_single=interp1( P\_f\_station(1:end)',h\_single,data\_station(1:end)','linear','extrap');

%分段线性插值：插值点处函数值由连接其最邻近的两侧点的线性函数预测。

%对超出已知点集的插值点用指定插值方法(线性)计算函数值

%多径信道

data3\_mult=fft\_data\_mult(1:N\_fft,:);

Rx\_pilota\_mult=data3\_mult(P\_f\_station(1:end),:); %接收到的导频

h\_mult=Rx\_pilota\_mult./pilot\_seq;

H\_mult=interp1( P\_f\_station(1:end)',h\_mult,data\_station(1:end)','linear','extrap');

%% 多途径信道样条插值估计（信道均衡2）

H\_mult\_spline=interp1( P\_f\_station(1:end)',h\_mult,data\_station(1:end)','spline','extrap');

%% 信道校正

data\_aftereq\_single=data3\_single(data\_station(1:end),:)./H\_single;

data\_aftereq\_mult=data3\_mult(data\_station(1:end),:)./H\_mult;

data\_aftereq\_spline=data3\_mult(data\_station(1:end),:)./H\_mult\_spline;

%% 并串转换

%单径信道(信道均衡方法1)

data\_aftereq\_single=reshape(data\_aftereq\_single,[],1);

data\_aftereq\_single=data\_aftereq\_single(1:length(spread\_data));

data\_aftereq\_single=reshape(data\_aftereq\_single,N\_sc,length(data\_aftereq\_single)/N\_sc);

%多径信道(信道均衡方法1)

data\_aftereq\_mult=reshape(data\_aftereq\_mult,[],1);

data\_aftereq\_mult=data\_aftereq\_mult(1:length(spread\_data));

data\_aftereq\_mult=reshape(data\_aftereq\_mult,N\_sc,length(data\_aftereq\_mult)/N\_sc);

%多径信道(信道均衡方法2)

data\_aftereq\_spline=reshape(data\_aftereq\_spline,[],1);

data\_aftereq\_spline=data\_aftereq\_spline(1:length(spread\_data));

data\_aftereq\_spline=reshape(data\_aftereq\_spline,N\_sc,length(data\_aftereq\_spline)/N\_sc);

%% 解扩

demspread\_data\_single = despread(data\_aftereq\_single,code); % 数据解扩

demspread\_data\_mult = despread(data\_aftereq\_mult,code);

demspread\_data\_spline = despread(data\_aftereq\_spline,code);

%% QPSK解调

demodulation\_data\_single=pskdemod(demspread\_data\_single,M,pi/M);

De\_data1\_single = reshape(demodulation\_data\_single,[],1);

De\_data2\_single = de2bi(De\_data1\_single);

De\_Bit\_single = reshape(De\_data2\_single',1,[]);

demodulation\_data\_mult=pskdemod(demspread\_data\_mult,M,pi/M);

De\_data1\_mult = reshape(demodulation\_data\_mult,[],1);

De\_data2\_mult = de2bi(De\_data1\_mult);

De\_Bit\_mult = reshape(De\_data2\_mult',1,[]);

demodulation\_data\_spline=pskdemod(demspread\_data\_spline,M,pi/M);

De\_data1\_spline= reshape(demodulation\_data\_spline,[],1);

De\_data2\_spline= de2bi(De\_data1\_spline);

De\_Bit\_spline = reshape(De\_data2\_spline',1,[]);

%% （解交织）

%% 信道译码（维特比译码）

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

rx\_c\_de\_single = vitdec(De\_Bit\_single,trellis\_single,tblen,'trunc','hard'); %硬判决

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

rx\_c\_de\_mult = vitdec(De\_Bit\_mult,trellis\_mult,tblen,'trunc','hard'); %硬判决

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

rx\_c\_de\_spline= vitdec(De\_Bit\_spline,trellis\_spline,tblen,'trunc','hard'); %硬判决

%% 计算误码率

[err,Ber2\_single(jj)] = biterr(De\_Bit\_single(1:length(code\_data)),code\_data);%译码前的误码率

[err, Ber\_single(jj)] = biterr(rx\_c\_de\_single(1:length(P\_data)),P\_data);%译码后的误码率

[err,Ber2\_mult(jj)] = biterr(De\_Bit\_mult(1:length(code\_data)),code\_data);%译码前的误码率

[err, Ber\_mult(jj)] = biterr(rx\_c\_de\_mult(1:length(P\_data)),P\_data);%译码后的误码率

[err,Ber2\_spline(jj)] = biterr(De\_Bit\_spline(1:length(code\_data)),code\_data);%译码前的误码率

[err, Ber\_spline(jj)] = biterr(rx\_c\_de\_spline(1:length(P\_data)),P\_data);%译码后的误码率

end

%% 原始信号不通过信道均衡

%信道编码、QPSK调制后的信号为modu\_data

%IFFT

ifft\_data\_original=ifft(modu\_data);

%插入保护间隔，循环前缀

Tx\_cd\_original=[ifft\_data\_original(N\_fft-N\_cp+1:end,:);ifft\_data\_original];

%加窗 （通过矩阵点乘）

Tx\_window\_original=zeros(size(Tx\_cd\_original));

Tx\_window\_original=Tx\_cd\_original.\*repmat(rcoswindow(alpha,size(Tx\_cd\_original,1)),1,60);%60是矩阵的维度

%并串转换

Tx\_data\_original=reshape(Tx\_window\_original,[],1);

%AWGN信道

Ber\_original=zeros(1,length(SNR));

Ber2\_original=zeros(1,length(SNR));

for ii=1:length(SNR)

channe\_original=awgn(Tx\_data\_original,SNR(ii),'measured');%添加高斯噪声

% 串并转换

Rx\_data1\_original=reshape(channe\_original,N\_fft+N\_cp,[]);

%去保护间隔、循环前缀

Rx\_data2\_original=Rx\_data1\_original(N\_cp+1:end,:);

% FFT

fft\_data\_original=fft(Rx\_data2\_original);

% 并串转换

data\_aftereq\_original=reshape(fft\_data\_original,[],1);

data\_aftereq\_original=data\_aftereq\_original(1:2652);

data\_aftereq\_original=reshape(data\_aftereq\_original,N\_sc,length(data\_aftereq\_original)/N\_sc);

%QPSK解调

demodulation\_data\_original=pskdemod(data\_aftereq\_original,M,pi/M);

De\_data1\_original= reshape(demodulation\_data\_original,[],1);

De\_data2\_original= de2bi(De\_data1\_original);

De\_Bit\_original= reshape(De\_data2\_original',1,[]);

% 信道译码（维特比译码）

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

rx\_c\_de\_original = vitdec(De\_Bit\_original,trellis\_original,tblen,'trunc','hard'); %硬判决

%误码率

[err,Ber2\_original(jj)] = biterr(De\_Bit\_original,code\_data(1:5304));%译码前的误码率

[err, Ber\_original(ii)] = biterr(rx\_c\_de\_original,P\_data(1:2652));%译码后的误码率

end

%% CFO估计(单径信道的频率估计和补偿)

FreqOffset=5\*10^3/(5\*10^9);%计算频偏比 载波频率为2GHz,频率偏移为5KHz

snr = 10^(-20/10);

snr1=2:2:20;

for jj=1:length(SNR)

channe\_CFO=awgn(Tx\_data,SNR(jj),'measured');

Rx = exp(i\*2\*pi\*FreqOffset\*(0:length(channe\_CFO)-1)./N\_fft).\*channe\_CFO ; %加频率偏移的信号

PHI\_sum = zeros(1,Nd\*(N\_fft+N\_cp)-N\_fft);

GM\_sum = zeros(1,Nd\*(N\_fft+N\_cp)-N\_fft);

%ML方法对于频偏估计,符号同步

for n = 64:Nd\*(N\_fft+N\_cp)-(N\_fft+N\_cp)

PHI=0;GM=0;

for m = n:n+N\_cp-1

PHI = PHI+ (Rx(m)\*conj(Rx(m)) + Rx(m+N\_fft)\*conj(Rx(m+N\_fft)));

GM = GM+ Rx(m)\*conj(Rx(m+N\_fft));

end

PHI\_sum(n) = abs(GM)- (snr/(+1))\*PHI;

GM\_sum(n) = -angle(GM)/(2\*pi);

end

%确定频偏

[d\_ml,deltaf\_ml]=ml\_estimate(Rx,1/4,N\_fft,SNR);

max\_peak=find\_peak(d\_ml,1/4,N\_fft);

deltaf\_ml1=FreqOffset;

for k=1:length(max\_peak)

if max\_peak(k)~=0

deltaf\_ml1=deltaf\_ml(k);

end

deltaf\_ml(k)=deltaf\_ml1;

end

deltaf\_ml\_add=zeros(1,(length(Rx)-length(deltaf\_ml)));

deltaf\_ml\_add(1:end)=deltaf\_ml(end);

deltaf\_ml=cat(2,deltaf\_ml,deltaf\_ml\_add);

%频率补偿

Rx\_offset=Rx\*exp(-i\*2\*pi\*deltaf\_ml(k)\*(k-1)/N\_fft);%频率补偿后的信号

%并串转换

Rx\_data1\_offset=reshape(Rx\_offset,N\_fft+N\_cp,[]);

%去保护间隔

Rx\_data2\_offset=Rx\_data1\_offset(N\_cp+1:end,:);

%FFT

fft\_data\_offset=fft(Rx\_data2\_offset);

%信道均衡

data3\_offset=fft\_data\_offset(1:N\_fft,:);

Rx\_pilot\_offset=data3\_offset(P\_f\_station(1:end),:); %接收到的导频

h\_offset=Rx\_pilot\_offset./pilot\_seq;

H\_offset=interp1( P\_f\_station(1:end)',h\_offset,data\_station(1:end)','linear','extrap');

%信道矫正

data\_aftereq\_offset=data3\_offset(data\_station(1:end),:)./H\_offset

%并串转换

data\_aftereq\_offset=reshape(data\_aftereq\_offset,[],1);

data\_aftereq\_offset=data\_aftereq\_offset(1:length(spread\_data));

data\_aftereq\_offset=reshape(data\_aftereq\_offset,N\_sc,length(data\_aftereq\_offset)/N\_sc);

%解扩

demspread\_data\_offset = despread(data\_aftereq\_offset,code);

%QPSK解调

demodulation\_data\_offset=pskdemod(demspread\_data\_offset,M,pi/M);

De\_data1\_offset = reshape(demodulation\_data\_offset,[],1);

De\_data2\_offset = de2bi(De\_data1\_offset);

De\_Bit\_offset = reshape(De\_data2\_offset',1,[]);

%信道译码

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

rx\_c\_de\_offset = vitdec(De\_Bit\_offset ,trellis\_offset ,tblen,'trunc','hard'); %硬判决

%计算误码率

[err,Ber2\_offset(jj)] = biterr(De\_Bit\_offset(1:length(code\_data)),code\_data);%译码前的误码率

[err, Ber\_offset(jj)] = biterr(rx\_c\_de\_offset(1:length(P\_data)),P\_data);%译码后的误码率

end

%% 误比特率曲线

%单径信道信道均衡1

figure(5);

subplot(3,1,1);

semilogy(SNR,Ber2\_single,'b-s');

hold on;

semilogy(SNR,Ber\_single,'r-o');

hold on;

legend('4PSK调制、卷积码译码前（有扩频）','4PSK调制、卷积码译码后（信道均衡1）');

hold on;

xlabel('SNR');

ylabel('BER');

title('AWGN单径信道（无频偏补偿、信道均衡1）下误比特率曲线');

%多径信道信道均衡1

subplot(3,1,2);

semilogy(SNR,Ber2\_mult,'b-s');

hold on;

semilogy(SNR,Ber\_mult,'r-o');

hold on;

legend('4PSK调制、卷积码译码前（有扩频）','4PSK调制、卷积码译码后（道信道均衡1）');

hold on;

xlabel('SNR');

ylabel('BER');

title('AWGN多径信道（无频偏补偿、信道均衡1）下误比特率曲线');

%多径信道信道均衡2

subplot(3,1,3);

semilogy(SNR,Ber2\_spline,'b-s');

hold on;

semilogy(SNR,Ber\_spline,'r-o');

hold on;

legend('4PSK调制、卷积码译码前（扩频）','4PSK调制、卷积码译码后（信道均衡2）');

hold on;

xlabel('SNR');

ylabel('BER');

title('AWGN多径信道（无频偏补偿、信道均衡2）下误比特率曲线');

figure(6);

%画出频偏的图像

subplot(3,1,1);plot(PHI\_sum);title('定时偏移估计');grid on;

subplot(3,1,2);plot(GM\_sum);title('频率偏移估计');grid on;

%频偏补偿

subplot(3,1,3);

semilogy(SNR,Ber2\_offset,'b-s');

hold on;

semilogy(SNR,Ber\_offset,'r-o');

legend('4PSK调制、卷积码译码前（无行信道均衡）','4PSK调制、卷积码译码后（无行信道估计）');

hold on;

xlabel('SNR');

ylabel('BER');

title('AWGN单径信道下（频偏补偿、信道均衡1）误比特率曲线');

%% 数据结果曲线

figure(7)

subplot(5,1,1);

x=0:1:30;

stem(x,P\_data(1:31));

ylabel('amplitude');

title('发送数据（以前30个数据为例)');

legend('4PSK调制、卷积译码、有扩频');

subplot(5,1,2);

x=0:1:30;

stem(x,rx\_c\_de\_single(1:31));

ylabel('amplitude');

title('接收数据单径信道(信道均衡1)');

legend('4PSK调制、卷积译码、扩频、单径');

subplot(5,1,3);

x=0:1:30;

stem(x,rx\_c\_de\_mult(1:31));

ylabel('amplitude');

title('接收数据多径信道(信道均衡1)');

legend('4PSK调制、卷积译码、扩频、多径');

subplot(5,1,4);

x=0:1:30;

stem(x,rx\_c\_de\_spline(1:31));

ylabel('amplitude');

title('接收数据多径信道(信道均衡2)');

legend('4PSK调制、卷积译码、扩频、多径');

subplot(5,1,5);

x=0:1:30;

stem(x,rx\_c\_de\_offset(1:31));

ylabel('amplitude');

title('接收数据单信道(信道均衡1、频率补偿)');

legend('4PSK调制、卷积译码、有扩频、多径、有频率补偿');

%% 星座图

%单径 无均衡

scatterplot(data\_aftereq\_original(:));

title('AWGN单径无信道均衡接收信号的星座图');

%单径 信道均衡1

scatterplot(demspread\_data\_single(:));

title('AWGN单径信道（信道均衡1）接收信号的星座图');

%多径 信道均衡1

scatterplot(demspread\_data\_mult(:));

title('AWGN多径信道（信道均衡1）接收信号的星座图');

%多径 信道均衡2

scatterplot(demspread\_data\_spline(:));

title('AWGN多径信道（信道均衡2）接收信号的星座图');

%单径 信道均衡1 有频率补偿

scatterplot(demspread\_data\_offset(:));

title('AWGN单径信道（信道均衡1、有频率补偿）接收信号的星座图');