数字通信第三次作业-均衡

姓名：王浩

学号：MG20230081

专业：通信与信息系统

## 一．基本概念

迫零均衡(ZFE): 是基于最大失真准则的线性均衡器。最大失真准则定义为均衡器输出端最大码间干扰的最优化。但虽然消除了码间干扰，但增大了加性噪声。仿真中接收端默认已知瑞利信道矩阵，且为了减小计算复杂度，信道矩阵H定义为20\*20的方阵，保证了HHH的逆矩阵始终存在。

## 二．运行结果

### 2.1误比特率-信噪比（Eb/N0）



图1.通过Rayleigh信道，QPSK和16QAM误比特率性能曲线

**结果分析：**从上图可以看出，由于迫零均衡虽然通过矩阵运算抵消了瑞利信道矩阵，消除了多径衰落，但变相的增加了加性噪声，两个系统的抗噪声性能相比高斯信道都变差了许多。基本上信噪比在50dB以上时，误码率才能到10的-6次方的数量级。在相同信噪比情况下，QPSK的误码率稍微比16QAM小。

### 2.2误码率-信噪比（Es/N0）



图2.通过Rayleigh信道，QPSK和16QAM误码率性能曲线

**结果分析：** 误码率与误比特率的关系这里近似为SER = BER \* log2（M），M为调制阶数。

## 三．附录代码

### 3.1误比特率-信噪比（Eb/N0）

clc;clear all;close all;

N = 10000000;

s = source(N); %信源产生，序列个数为N

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%16QAM

Eb = 2.5;%16QAM每个比特能量

mu = 0;

SNR = 0 : 5 : 70;

BER = zeros(1,length(SNR));

N0 = Eb./(power(10,SNR/10));

sigma = sqrt(N0/2); %计算噪声的标准差

for i = 1:length(sigma)

n = normrnd(mu,sigma(i),[2,N/4]); %产生服从高斯分布的双路噪声

n\_c = n(1,:);n\_s = n(2,:);

s1 = zeros(4,N/4);

for c = 1:N/4

s1(1,c) = s(4\*c-3);

s1(2,c) = s(4\*c-2);

s1(3,c) = s(4\*c-1);

s1(4,c) = s(4\*c);

end %将信源分解成四路信号

[s\_c1,s\_s1] = QAM(s1); %进行16QAM编码

% h1 = normrnd(0,sqrt(1/2),N/2,N/4); %产生瑞利乘性噪声

% h\_i = h1(1:N/4,:);h\_q = h1(N/4+1:N/2 ,:);

% H = h\_i + 1i\*h\_q;

% s\_r = H\*(s\_c1 + 1i\*s\_s1).';

%

% r1 = s\_r + (n\_c + 1i\*n\_s).';

%

% W = inv(H'\*H)\*(H');

%

% W1 = W\*H;

% r\_ZF = W \* r1; %均衡后输出信号

% r = zeros(1,N/2);

% for j= 1:size(r\_ZF,1)

% r(j) = r\_ZF(j,j);

% end

h1 = normrnd(0,sqrt(1/2),40,N/4); %产生瑞利乘性噪声

h\_i = h1(1:20,:);h\_q = h1(21:40 ,:);

H = h\_i + 1i\*h\_q;

S = (s\_c1 + 1i\*s\_s1).';

for j = 0 : N/80-1 %串并转换，每一列20bit，减小计算复杂度

s\_r(20\*j+1:20\*(j+1)) = H(:,20\*j+1:20\*(j+1))\*S(20\*j+1:20\*(j+1));

end

r1 = s\_r + n\_c + 1i\*n\_s;

W = zeros(20,N/4);

for j = 0 : N/80-1

h = H(:,20\*j+1:20\*(j+1));

W(:,20\*j+1:20\*(j+1)) = (h'\*h)\h';

end

% W1 = zeros(20,N/4);

% for j = 0 : N/80-1

% h = H(:,20\*j+1:20\*(j+1));

% W1(:,20\*j+1:20\*(j+1)) = W(:,20\*j+1:20\*(j+1))\*h; %测试最终是否为单位矩阵

% end

r\_ZF = zeros(1,N/4);

for j = 0 : N/80-1

r\_ZF(20\*j+1:20\*(j+1)) = W(:,20\*j+1:20\*(j+1))\*(r1(20\*j+1:20\*(j+1))).'; %均衡后输出信号

end

r\_c = real(r\_ZF);r\_s = imag(r\_ZF);

y = judgement\_16QAM(r\_c,r\_s); %%16QAM解码，判决输出

BER(i) = error\_rate(s,y); %%求误比特率

end

semilogy(SNR,BER,'b\*');

axis([0 60 10^-6 1]);

hold on;

grid on;

xlabel('SNR/dB');ylabel('BER');

title('BER-SNR,Rayleigh');

r = 4\*Eb./N0;

BER\_true = (3/2\*(1 - sqrt(r./(10 + r))) - 9/16\*(1 - sqrt(r./(10 + r)).\*(4/pi\*atan(sqrt((10+r)./r)))))/4; %16QAM理想误比特率

semilogy(SNR,BER\_true,'-m');

hold on

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%QPSK

Eb = 1/2;%QPSK每个比特能量为1/2

N0 = Eb./(power(10,SNR/10));

sigma = sqrt(N0/2); %计算噪声的标准差

for i = 1:length(sigma)

n = normrnd(mu,sigma(i),[2,N/2]); %产生服从高斯分布的双路噪声

n\_c = n(1,:);n\_s = n(2,:);

s1\_c = zeros(1,N/2);s1\_s = zeros(1,N/2);

for c = 1:N/2

s1\_c(c) = s(2\*c-1);

s1\_s(c) = s(2\*c);

end %将信源分解成双路信号

[s\_c,s\_s] = QPSK(s1\_c,s1\_s); %进行QPSK编码

h1 = normrnd(0,sqrt(1/2),40,N/2); %产生瑞利乘性噪声

h\_i = h1(1:20,:);h\_q = h1(21:40 ,:);

H = h\_i + 1i\*h\_q;

S = (s\_c + 1i\*s\_s).';

for j = 0 : N/40-1 %串并转换，每一列20bit，减小计算复杂度

s\_r(20\*j+1:20\*(j+1)) = H(:,20\*j+1:20\*(j+1))\*S(20\*j+1:20\*(j+1));

end

r1 = s\_r + n\_c + 1i\*n\_s;

W = zeros(20,N/2);

for j = 0 : N/40-1

h = H(:,20\*j+1:20\*(j+1));

W(:,20\*j+1:20\*(j+1)) = (h'\*h)\h';

end

% W1 = zeros(20,N/2);

% for j = 0 : N/40-1

% h = H(:,20\*j+1:20\*(j+1));

% W1(:,20\*j+1:20\*(j+1)) = W(:,20\*j+1:20\*(j+1))\*h; %测试最终是否为单位矩阵

% end

r\_ZF = zeros(1,N/2);

for j = 0 : N/40-1

r\_ZF(20\*j+1:20\*(j+1)) = W(:,20\*j+1:20\*(j+1))\*(r1(20\*j+1:20\*(j+1))).'; %均衡后输出信号

end

r\_c = real(r\_ZF);r\_s = imag(r\_ZF);

y = judgement\_QPSK(r\_c,r\_s); %%QPSK解码，判决输出

BER(i) = error\_rate(s,y); %%求误比特率

end

semilogy(SNR,BER,'rs');

hold on;

r = 2\*Eb./N0;

BER\_true = (1 - sqrt(r./(2 + r)) - 1/4\*(1 - sqrt(r./(2 + r)).\*(4/pi\*atan(sqrt((2+r)./r)))))/2;%QPSK理想误比特率

semilogy(SNR,BER\_true,'-y');

hold on

legend('16QAM simulated','16QAM theoretical','QPSK simulated','QPSK theoretical');

### 3.2误码率-信噪比（Es/N0）

clc;clear all;close all

N = 10000000;

s = source(N); %信源产生，序列个数为N

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%16QAM

Es = 10;%16QAM每个符号能量

mu = 0;

SNR = 0 : 5 :80;

SER = zeros(1,length(SNR));

N0 = Es./(power(10,SNR/10));

sigma = sqrt(N0/2); %计算噪声的标准差

for i = 1:length(sigma)

n = normrnd(mu,sigma(i),[2,N/4]); %产生服从高斯分布的双路噪声

n\_c = n(1,:);n\_s = n(2,:);

s1 = zeros(4,N/4);

for c = 1:N/4

s1(1,c) = s(4\*c-3);

s1(2,c) = s(4\*c-2);

s1(3,c) = s(4\*c-1);

s1(4,c) = s(4\*c);

end %将信源分解成四路信号

[s\_c,s\_s] = QAM(s1); %进行16QAM编码

h1 = normrnd(0,sqrt(1/2),40,N/4); %产生瑞利乘性噪声

h\_i = h1(1:20,:);h\_q = h1(21:40 ,:);

H = h\_i + 1i\*h\_q;

S = (s\_c + 1i\*s\_s).';

for j = 0 : N/80-1 %串并转换，每一列20bit，减小计算复杂度

s\_r(20\*j+1:20\*(j+1)) = H(:,20\*j+1:20\*(j+1))\*S(20\*j+1:20\*(j+1));

end

r1 = s\_r + n\_c + 1i\*n\_s;

W = zeros(20,N/4);

for j = 0 : N/80-1

h = H(:,20\*j+1:20\*(j+1));

W(:,20\*j+1:20\*(j+1)) = (h'\*h)\h';

end

W1 = zeros(20,N/4);

for j = 0 : N/80-1

h = H(:,20\*j+1:20\*(j+1));

W1(:,20\*j+1:20\*(j+1)) = W(:,20\*j+1:20\*(j+1))\*h; %测试最终是否为单位矩阵

end

r\_ZF = zeros(1,N/4);

for j = 0 : N/80-1

r\_ZF(20\*j+1:20\*(j+1)) = W(:,20\*j+1:20\*(j+1))\*(r1(20\*j+1:20\*(j+1))).'; %均衡后输出信号

end

r\_c = real(r\_ZF);r\_s = imag(r\_ZF);

y = judgement\_16QAM(r\_c,r\_s); %%16QAM解码，判决输出

SER(i) = symbol\_error\_16QAM(s,y); %%求误符号率

end

figure(2)

semilogy(SNR,SER,'b\*');

axis([0 70 10^-6 1]);

hold on;

grid on;

xlabel('SNR/dB');ylabel('BER');

title('SER-SNR,Rayleigh');

r = Es./N0;

SER\_true = 3/2\*(1 - sqrt(r./(10 + r))) - 9/16\*(1 - sqrt(r./(10 + r)).\*(4/pi\*atan(sqrt((10+r)./r)))); %16QAM理想误符号率

semilogy(SNR,SER\_true,'-m');

hold on

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%QPSK

Es = 1;%QPSK每个符号能量为1

N0 = Es./(power(10,SNR/10));

sigma = sqrt(N0/2); %计算噪声的标准差

for i = 1:length(sigma)

n = normrnd(mu,sigma(i),[2,N/2]); %产生服从高斯分布的双路噪声

n\_c = n(1,:);n\_s = n(2,:);

s1\_c = zeros(1,N/2);s1\_s = zeros(1,N/2);

for c = 1:N/2

s1\_c(c) = s(2\*c-1);

s1\_s(c) = s(2\*c);

end %将信源分解成双路信号

[s\_c,s\_s] = QPSK(s1\_c,s1\_s); %进行QPSK编码

% h = normrnd(0,sqrt(1/2),2,N/2); %产生瑞利乘性噪声

% h\_i = h(1,:);h\_q = h(2,:);

%

% H = h\_i + 1i\*h\_q;

% W = inv(H'\*H )\*(H'); %迫零均衡

% n\_ZF = W \* (n\_c + 1i\*n\_s);

% n = zeros(1,N/2);

% for j= 1:size(n\_ZF,1)

% n(j) = n\_ZF(j,j);

% end

%

% r = s\_c + 1i\*s\_s + n; %均衡后输出信号

%

% r\_c = real(r);r\_s = imag(r);

h1 = normrnd(0,sqrt(1/2),40,N/2); %产生瑞利乘性噪声

h\_i = h1(1:20,:);h\_q = h1(21:40 ,:);

H = h\_i + 1i\*h\_q;

S = (s\_c + 1i\*s\_s).';

for j = 0 : N/40-1 %串并转换，每一列20bit，减小计算复杂度

s\_r(20\*j+1:20\*(j+1)) = H(:,20\*j+1:20\*(j+1))\*S(20\*j+1:20\*(j+1));

end

r1 = s\_r + n\_c + 1i\*n\_s;

W = zeros(20,N/2);

for j = 0 : N/40-1

h = H(:,20\*j+1:20\*(j+1));

W(:,20\*j+1:20\*(j+1)) = (h'\*h)\h';

end

W1 = zeros(20,N/2);

for j = 0 : N/40-1

h = H(:,20\*j+1:20\*(j+1));

W1(:,20\*j+1:20\*(j+1)) = W(:,20\*j+1:20\*(j+1))\*h; %测试最终是否为单位矩阵

end

r\_ZF = zeros(1,N/2);

for j = 0 : N/40-1

r\_ZF(20\*j+1:20\*(j+1)) = W(:,20\*j+1:20\*(j+1))\*(r1(20\*j+1:20\*(j+1))).'; %均衡后输出信号

end

r\_c = real(r\_ZF);r\_s = imag(r\_ZF);

y = judgement\_QPSK(r\_c,r\_s); %%QPSK解码，判决输出

SER(i) = symbol\_error\_QPSK(s,y); %%求误符号率

end

semilogy(SNR,SER,'rs');

hold on;

r = Es./N0;

SER\_true = 1 - sqrt(r./(2 + r)) - 1/4\*(1 - sqrt(r./(2 + r)).\*(4/pi\*atan(sqrt((2+r)./r)))); %QPSK理想误符号率

semilogy(SNR,SER\_true,'-y');

hold on

legend('16QAM simulated','16QAM theoretical','QPSK simulated','QPSK theoretical');