数字通信第四次作业-编码

姓名：王浩

学号：MG20230081

专业：通信与信息系统

## 一．基本概念

汉明编码：汉明码是一种线性分组码。线性分组码是指将信息序列划分为长度为k的序列段，在每一段后面附加r位的监督码，且监督码和信息码之间构成线性关系，即它们之间可由线性方程组来联系。这样构成的抗干扰码称为线性分组码。本实验中采用（7，4）汉明码的编码方案，即k=4，r=3，最大纠错能力为1位，最大检错能力为2位。

## 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：** 由运行结果可以看出在信噪比小于6dB时，不编码的抗噪声性能要优于（7，4）汉明码硬判决，因为无论是哪种方案，在QPSK调制前每比特平均能量均为1/2，这样就导致了（7，4）汉明码在编码前每比特平均能量为7/8，这样7/8\*4/7=1/2，保证了调制前每比特平均能量均为1/2，控制了；两种方案在信噪比一样时发射相同信息量所需要的能量一致。但随着信噪比增大，每个码元（7bit）的错误比特数减小，（7，4）汉明码最大纠错能力为1位，其纠错能力得以体现，信噪比大于6dB时，不编码的抗噪声性能要弱于（7，4）汉明码硬判决软判决需要遍历所有可能情况，而且是浮点数计算，解码时计算量大大增加，但也换来了系统抗噪声性能的提升。

## 三．附录代码

**3.1主函数**

clc;clear all;close all;

N = 10000000;

s = source(N); %信源产生，序列个数为N

SNR = 0 : 1 : 15;

s1 = hamming\_encoding(s); %(7,4)汉明编码

N1 = length(s1);

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

%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

mu = 0;

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

r\_c = s\_c + n\_c;r\_s = s\_s + n\_s;

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,'-bd');

ylim([ 10^-6, 1])

hold on;

grid on;

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

%QPSK 硬判决

Eb = 7/8;%编码后QPSK每个比特能量为1/2，则编码前为7/8

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,N1/2]); %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,N1/2);s1\_s = zeros(1,N1/2);

for c = 1:N1/2

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

s1\_s(c) = s1(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

r\_c = s\_c + n\_c;r\_s = s\_s + n\_s;

y1 = judgement\_QPSK(r\_c,r\_s); %%QPSK解码，判决输出，硬判决，直接输出0或1

y = hamming\_decoding(y1); %最小汉明距离译码

BER(i) = error\_rate(s,y); %求误比特率

end

semilogy(SNR,BER,'-rs');

hold on;

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

for i = 1:length(sigma)

n = normrnd(mu,sigma(i),[2,N1/2]); %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,N1/2);s1\_s = zeros(1,N1/2);

for c = 1:N1/2

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

s1\_s(c) = s1(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

r\_c = s\_c + n\_c;r\_s = s\_s + n\_s;

for j = 1 : length(r\_c)

r(j\*2-1) = r\_c(j);

r(j\*2) = r\_s(j);

end

y = hamming\_decoding1(r); %最小汉明距离译码

BER(i) = error\_rate(s,y); %%求误比特率

end

semilogy(SNR,BER,'-kp');

hold on;

legend('Uncoded','Hard decision','Soft decision');

### 3.2（7，4）汉明编码

function b = hamming\_encoding(a)

Q = [1 1 1;1 1 0; 1 0 1;0 1 1 ];

G = [eye(4),Q];

N = length(a);

M = zeros(4,N/4);

for i = 1:N

M(i) = a(i); %matlab默认纵向填充

end

M = M.'; %M矩阵的转置

A = M \* G;

b = zeros(1, N/4\*7);

m = 1;

for i = 1:size(A,1) %A的行数

for j = 1:size(A,2) %A的列数

b(m) = A(i,j);

m = m+1;

end

end

b = mod(b,2); %模2处理

end

### 3.3 （7，4）汉明码软判决+最小欧式距离

function y = hamming\_decoding1(x)

A = [0 0 0 0 0 0 0;

0 0 0 1 0 1 1;

0 0 1 0 1 0 1;

0 0 1 1 1 1 0;

0 1 0 0 1 1 0;

0 1 0 1 1 0 1;

0 1 1 0 0 1 1;

0 1 1 1 0 0 0;

1 0 0 0 1 1 1;

1 0 0 1 1 0 0;

1 0 1 0 0 1 0;

1 0 1 1 0 0 1;

1 1 0 0 0 0 1;

1 1 0 1 0 1 0;

1 1 1 0 1 0 0;

1 1 1 1 1 1 1];

B = -sqrt(2).\*A + sqrt(2)/2;

for i = 0 : length(x)/7-1

m = x(7\*i+1:7\*i+7);

d = sum((m - B(1,:)).^2);

index = 1;

for j = 2 : 16

n = sum((m - B(j,:)).^2);

if (d > n)

d = n;

index = j;

end

end

y(4\*i+1:4\*i+4) = A(index,1:4);

end

### 3.3 （7，4）汉明码硬判决+最小汉明距离

function y = hamming\_decoding2(a)

N = length(a);

B = zeros(7,N/7);

for i = 1:N

B(i) = a(i); %matlab默认纵向填充

end

B = B.'; %M矩阵的转置

y = mhd(B); %最小汉明距离译码

function y = mhd(x)

y = zeros(1,size(x,1)\*4);

for i = 1 : size(x,1)

m = x(i,:);

n = zeros(1,16);

l = [0 0 0 0 0 0 0;

0 0 0 1 0 1 1;

0 0 1 0 1 0 1;

0 0 1 1 1 1 0;

0 1 0 0 1 1 0;

0 1 0 1 1 0 1;

0 1 1 0 0 1 1;

0 1 1 1 0 0 0;

1 0 0 0 1 1 1;

1 0 0 1 1 0 0;

1 0 1 0 0 1 0;

1 0 1 1 0 0 1;

1 1 0 0 0 0 1;

1 1 0 1 0 1 0;

1 1 1 0 1 0 0;

1 1 1 1 1 1 1];

for j = 1 : 16

n(j) = sum(abs(m - l(j,:)));

end

k = find(n == min(n));

switch k(1)

case 1

y(4\*i-3:4\*i) = [0 0 0 0];

case 2

y(4\*i-3:4\*i) = [0 0 0 1];

case 3

y(4\*i-3:4\*i) = [0 0 1 0];

case 4

y(4\*i-3:4\*i) = [0 0 1 1];

case 5

y(4\*i-3:4\*i) = [0 1 0 0];

case 6

y(4\*i-3:4\*i) = [0 1 0 1];

case 7

y(4\*i-3:4\*i) = [0 1 1 0];

case 8

y(4\*i-3:4\*i) = [0 1 1 1];

case 9

y(4\*i-3:4\*i) = [1 0 0 0];

case 10

y(4\*i-3:4\*i) = [1 0 0 1];

case 11

y(4\*i-3:4\*i) = [1 0 1 0];

case 12

y(4\*i-3:4\*i) = [1 0 1 1];

case 13

y(4\*i-3:4\*i) = [1 1 0 0];

case 14

y(4\*i-3:4\*i) = [1 1 0 1];

case 15

y(4\*i-3:4\*i) = [1 1 1 0];

case 16

y(4\*i-3:4\*i) = [1 1 1 1];

end

end

### 3.3 （7，4）汉明码硬判决+校验子检错

function y = hamming\_decoding(a)

Q = [1 1 1;1 1 0; 1 0 1;0 1 1 ];

H = [Q.',eye(3)];

N = length(a);

B = zeros(7,N/7);

for i = 1:N

B(i) = a(i); %matlab默认纵向填充

end

B = B.'; %M矩阵的转置

S = B \* (H.');

S = mod(S ,2); %模2运算

y1 = error\_detection(a,S); %纠错

for i = 0:length(y1)/7-1

y(i\*4+1:i\*4+4) = y1(i\*7+1:i\*7+4); %取码元（7bit）中前4bit为译码输出

end

function y = error\_detection(a,S)

for i = 1 : size(S,1)

if S(i,:) == [1 1 1]

a((i-1)\*7+1) = mod(a((i-1)\*7+1)+1 , 2);

elseif S(i,:) == [1 1 0]

a((i-1)\*7+2) = mod(a((i-1)\*7+2)+1 , 2);

elseif S(i,:) == [1 0 1]

a((i-1)\*7+3) = mod(a((i-1)\*7+3)+1 , 2);

elseif S(i,:) == [0 1 1]

a((i-1)\*7+4) = mod(a((i-1)\*7+4)+1 , 2);

elseif S(i,:) == [1 0 0]

a((i-1)\*7+5) = mod(a((i-1)\*7+5)+1 , 2);

elseif S(i,:) == [0 1 0]

a((i-1)\*7+6) = mod(a((i-1)\*7+6)+1 , 2);

elseif S(i,:) == [0 0 1]

a((i-1)\*7+7) = mod(a((i-1)\*7+7)+1 , 2);

end

end

y = a;