Beijing-Dublin International College

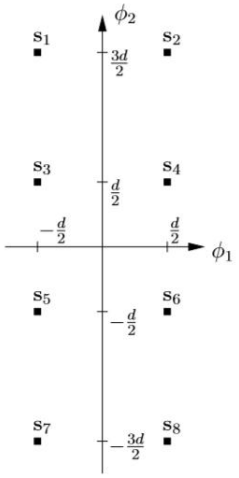
Assignment 1

Simulation of an 8-ary Digital Communication System

Name: ChengLitao	**UCD ID:** **17206018**
Honesty Pledge: I clearly understand the Academic Rules of Beijing University of Technology and University College Dublin and I am aware of the punishment for violating the Rules of Beijing University of Technology and University College Dublin. I hereby promise to abide by the relevant rules and promise the originality of my work. If found violating the rules, I would accept the punishment thereof. **Date:** 30/10/2020	

1 Introduction

I aimed to create a simulation of a digital communication system. And this system uses 8-ary modulation (there are M = 8 equiprobable transmit symbols)

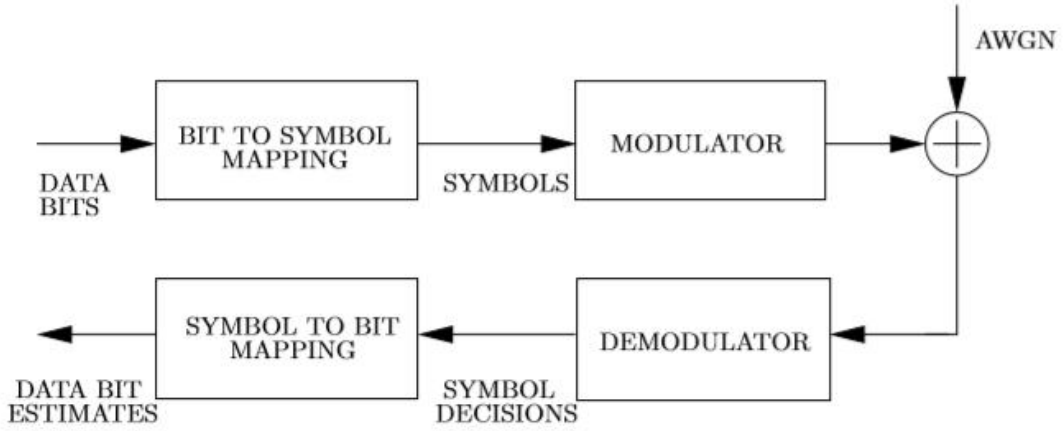


The whole process contains some modules to implement some special functions.

So firstly we should create some bits array randomly, and each 3bits mapped to symbols, then use a special coding method changing to the vector form, then additive white Gaussian noise (AWGN) channel will applied.

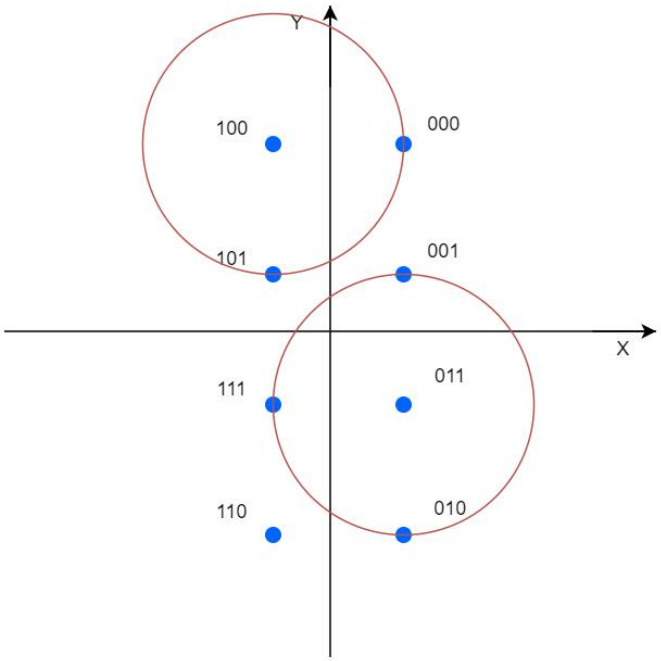
Then the optimum receiver will demodulate the vector, mapping to the symbol and

corresponding bits form.



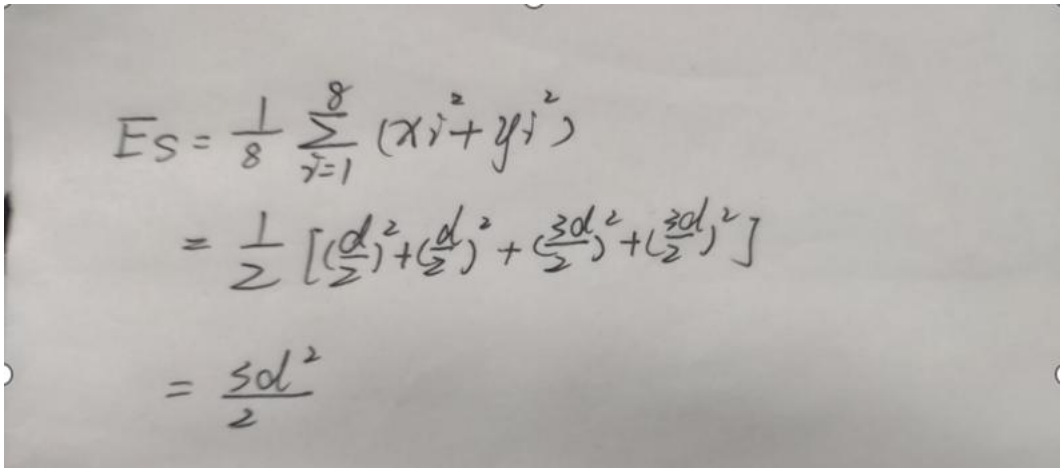
2 explanation for code design

Firstly, we design the bits map to vector method. I choose the Gray Code ,which any two adjacent codes differ by only one binary number, surround the whole eight points in clockwise(actually in any direction).



In this way, we can find that each adjacent point differ only 1 binary number(each points in cycle), so if there are some bit error happened, it is more likely to revise by randomly noise.

Then we the try to create 1million symbols to verify this system performance, and record the symbol vector in sx and sy, we can calculate the average energy of symbol(Es), and the average bit energy (Eb) is the one third of Es.




```
symbolNum=1000000
d=1;
Es=3*d*d/2;%enger of signal power(d^2)
sx=[-0.5,0.5,-0.5,0.5,-0.5,0.5,-0.5,0.5];
sy=[1.5,1.5,0.5,0.5,-0.5,-0.5,-1.5,-1.5];
SER=[];
BER=[];

for sigma=0.1:0.01:0.5
```

For each sigma of noise, I simulate the whole process of transmission in some steps below:

1 create bits randomly

```
%1CreatbitSignal
bit_in=rand(symbolNum,3)<0.5;
```

2 change to decimal form and use index() function map to the special Gray Code(in decimal too)

```
%2MapToSymbol
BinToDec=bit_in(:,1)*4+bit_in(:,2)*2+bit_in(:,3);
index=zeros(1,symbolNum);
index(BinToDec==0)=2;
index(BinToDec==1)=4;
index(BinToDec==2)=8;
index(BinToDec==3)=6;
index(BinToDec==4)=1;
index(BinToDec==5)=3;
index(BinToDec==6)=7;
index(BinToDec==7)=5;
symbol_in=index;
```

3modulate to the vector defined before.

```
%3modulate
index(symbol_in==1)=sx(1);
index(symbol_in==2)=sx(2);
index(symbol_in==3)=sx(3);
index(symbol_in==4)=sx(4);
index(symbol_in==5)=sx(5);
index(symbol_in==6)=sx(6);
index(symbol_in==7)=sx(7);
index(symbol_in==8)=sx(8);
index1=index';

index(symbol_in==1)=sy(1);
index(symbol_in==2)=sy(2);
index(symbol_in==3)=sy(3);
index(symbol_in==4)=sy(4);
index(symbol_in==5)=sy(5);
index(symbol_in==6)=sy(6);
index(symbol_in==7)=sy(7);
index(symbol_in==8)=sy(8);
index2=index';

v=[index1 index2];
```

4

Add the AWGN sigama Variance(multiply the sigama)

```
%4 add noise
noise=sigma.*randn(symbolNum,2);
v=v+noise;
```

5

Use the optimum receiver to decide the symbol by judge the min-distance.

```
%5 demodulate (optimum)
s=[sx' sy'];
ds=zeros(symbolNum,8);
for i=1:8
    ds(:,i)=(v(:,1)-sx(i)).^2+(v(:,2)-sy(i)).^2;
end
[min_d,symbol_out]=min(ds,[],2);
symbol_out;
```

6 map to bits, which is the Reverse procedure of the step 2.

```
%6 MapToBits
index(symbol_out==1)=1;
index(symbol_out==2)=0;
index(symbol_out==3)=1;
index(symbol_out==4)=0;
index(symbol_out==5)=1;
index(symbol_out==6)=0;
index(symbol_out==7)=1;
index(symbol_out==8)=0;
b1=index';
```

…

```
bits_out=[b1 b2 b3];
```

7

SER=1- (number of correct symbols in transmission/total number of symbol transmitted) *100%

BER=1- (number of correct bits in transmission/total number of bits transmitted) *100%

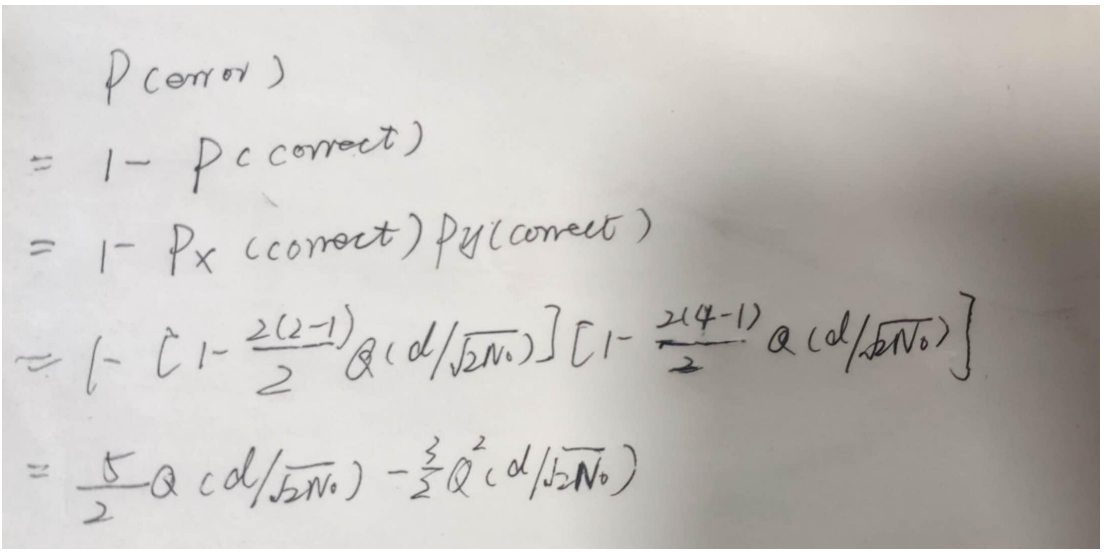
```
%7calculate SER & BER
a= (symbol_in'==symbol_out);
SER=[SER,1-(sum(a(:)==1)/symbolNum)]


b=bit_in==bits_out;
BER=[BER,1-(sum(b(:)==1)/(3*symbolNum))]
```

3 plot the results and answer the questions

```
sigma=0.1:0.01:0.5;
N0=2*(sigma.^2);% En=noise RMS value^2=N0/2
theory_value=2.5*qfunc(d./(sqrt(2*N0)))-1.5.*(qfunc(d./(sqrt(2.*N0)))).^2;
```

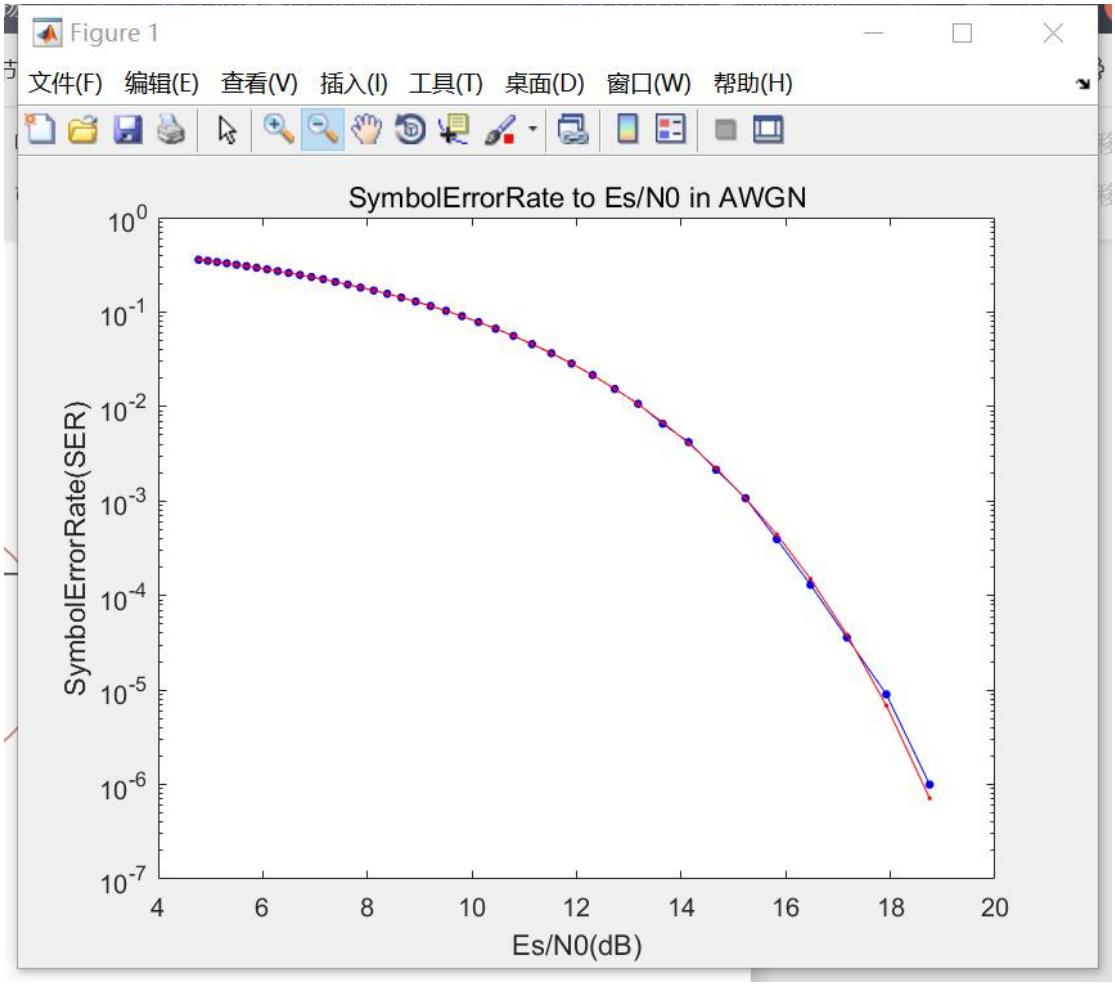
The Derivation of theory value:

To find the error rate, we have to find the correct situation when both two basis function coefficient are correct. In this case, we can change the symbol to 2-ASK and 4-ASK case, and calculate the and event(happen in the same time) probability:



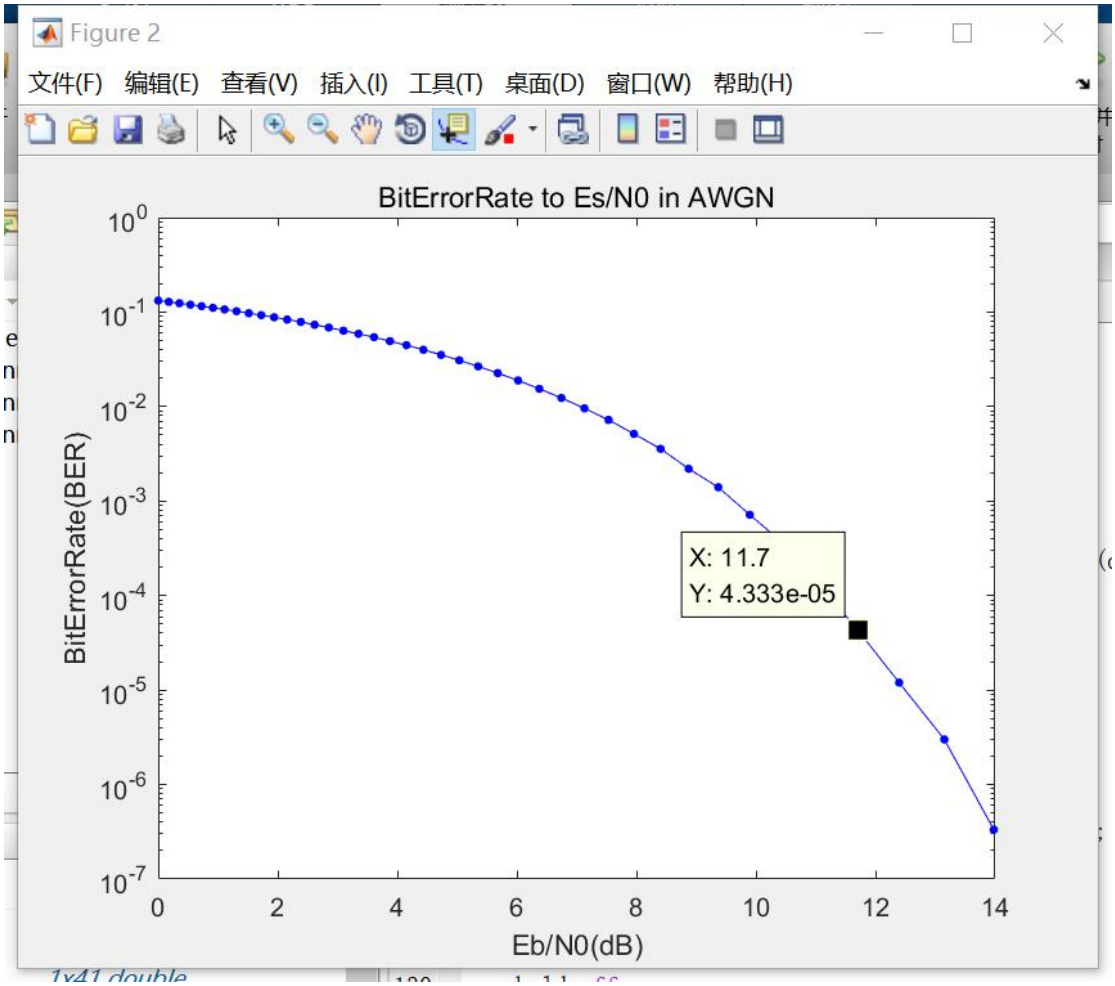

```
%plot SER to Es/N0
x=10.*log10(Es./N0);
figure(1);
semilogy(x,SER,'-b.','MarkerSize',10);
hold on;
semilogy(x,theory_value,'-r.','MarkerSize',5);
title('SymbolErrorRate to Es/N0 in AWGN');
xlabel('Es/N0(dB)');
ylabel('SymbolErrorRate(SER)');
hold off;


%plot BER to Eb/N0
figure(2);
semilogy(10.*log10(Es./(N0.*3)),BER,'-b.','MarkerSize',10);
title('BitErrorRate to Es/N0 in AWGN');
xlabel('Eb/N0(dB)');
ylabel('BitErrorRate(BER)');
```

SER to Es/N0:



Red line is the theory value, and blue line is the experimental value. It is obvious that they are quite similar.

BER to Eb/N0:



From the picture， we can find that when Eb/N0>11.2, the BER is less than 10^-4.

4 conclusion

The whole system simulation is quite closed to the theory value, and the speed performance of this code is acceptable,which is less than 10s.