数字通信实验报告

实验题目：Digital Modulated Symbol Transmission over Channels

实验要求：

1. 生成不同顺序的 BPSK/QAM 信号（BPSK~64-QAM）

（1）学习 Matlab 中的“qammod”和“qamdemod”函数;

（2）尝试绘制具有不同 QAM 的信号星座图.

2. 模拟不同顺序的 BPSK/QAM 传输

通过 AWGN 信道（高斯信道）进行传输.

3. 尝试提高 8-QAM 的性能

（1）在 Matlab 中用“qammod”函数绘制 8-QAM 信号星座;

（2）试着阅读材料，并思考如何提高 8-QAM 的性能.

实验过程及结果：

1. 生成不同顺序的 BPSK/QAM 信号（BPSK~64-QAM）

（1）学习 Matlab 中的“qammod”和“qamdemod”函数;

- **Y = qammod(X,M,⋯,Name,Value)**
- 输出使用正交幅度调制消息信号 X 的复包络.
- M 是字母表大小，必须是 2 的整数幂，表示调制阶数.
- 消息信号 X 必须由 0 到 M-1 之间的整数组成. 可以是标量、矢量、矩阵或三维数组。
- InputType：integer 或 bit 字符串之一.

  integer 表示消息信号是介于 0 和 M-1 之间的整数值.

  bit 表示消息信号为二进制（0 或 1）. 默认值为 integer.
- UnitAveragePower：逻辑标量值.

  如果为 true，QAM 星座将缩放为平均功率为 1（即 true 表示采用归一化的星座图）.

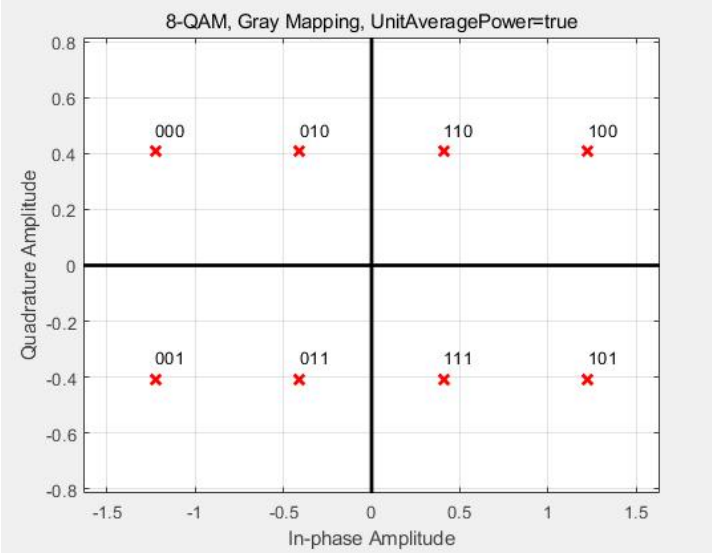
  如果为 false，则 QAM 星座点之间最小距离为 2.默认值为 false.
- OutputDataType 将定点类型输出为有符号、无标度的 numerictype 对象.

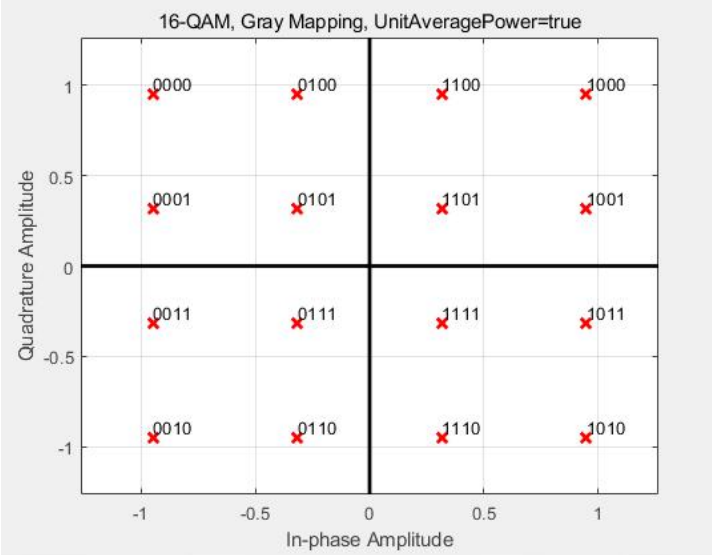
  当输入为定点时，必须指定此参数.
- PlotConstellation 是一个逻辑标量值.

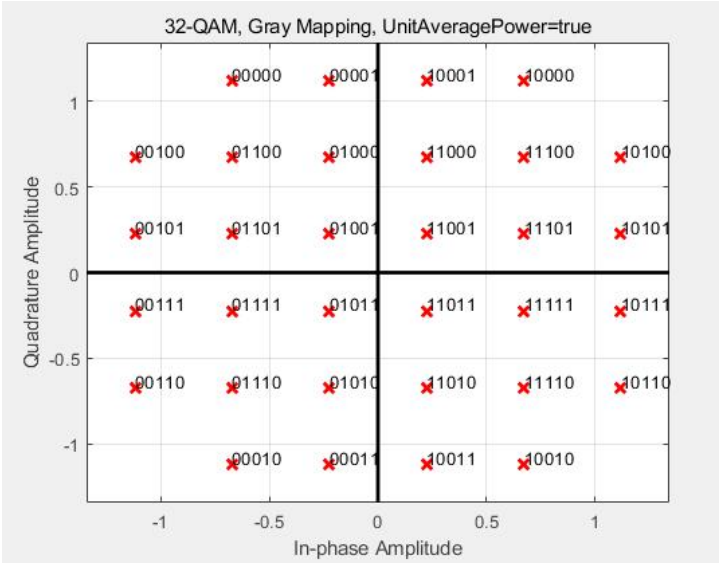
  如果为 true，则绘制 QAM 星座图. 默认值为 false.
- SYMBOL_ORDER：gray 或 bin，默认为 gray.

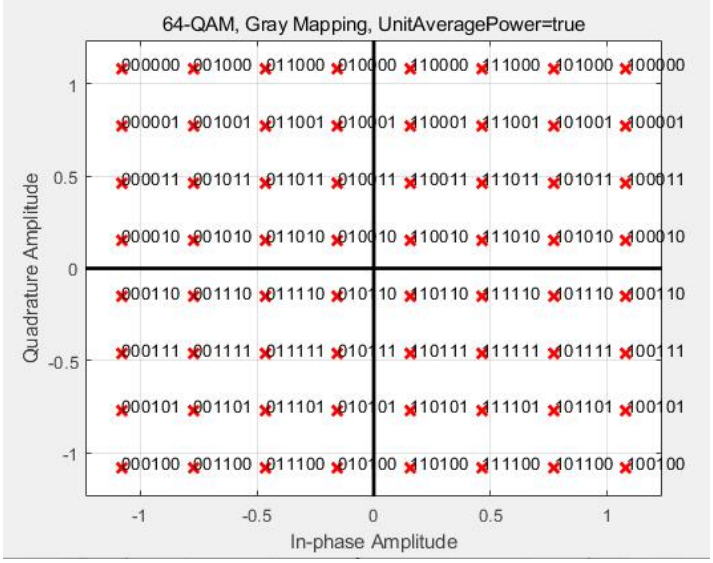
（2）尝试绘制具有不同 QAM 的信号星座图.

利用 Matlab 中的“qammod”和“qamdemod”函数，我们绘制出了如下星座图：


8-QAM, Gray Mapping, UnitAveragePower=true
000
010
110
100
001
011
111
101
Quadrature Amplitude
In-phase Amplitude


16-QAM, Gray Mapping, UnitAveragePower=true
0000
0100
1100
1000
0001
0101
1101
1001
0011
0111
1111
1011
0010
0110
1110
1010
Quadrature Amplitude
In-phase Amplitude


32-QAM, Gray Mapping, UnitAveragePower=true
Quadrature Amplitude
In-phase Amplitude

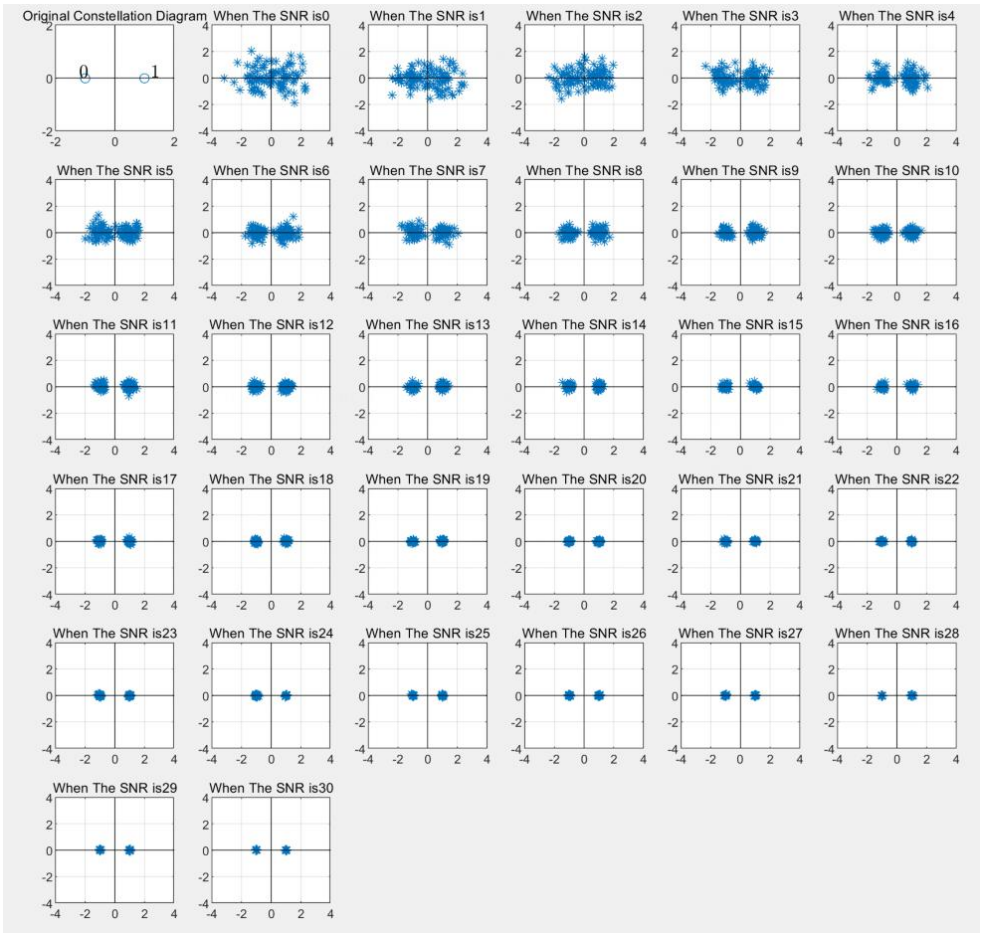


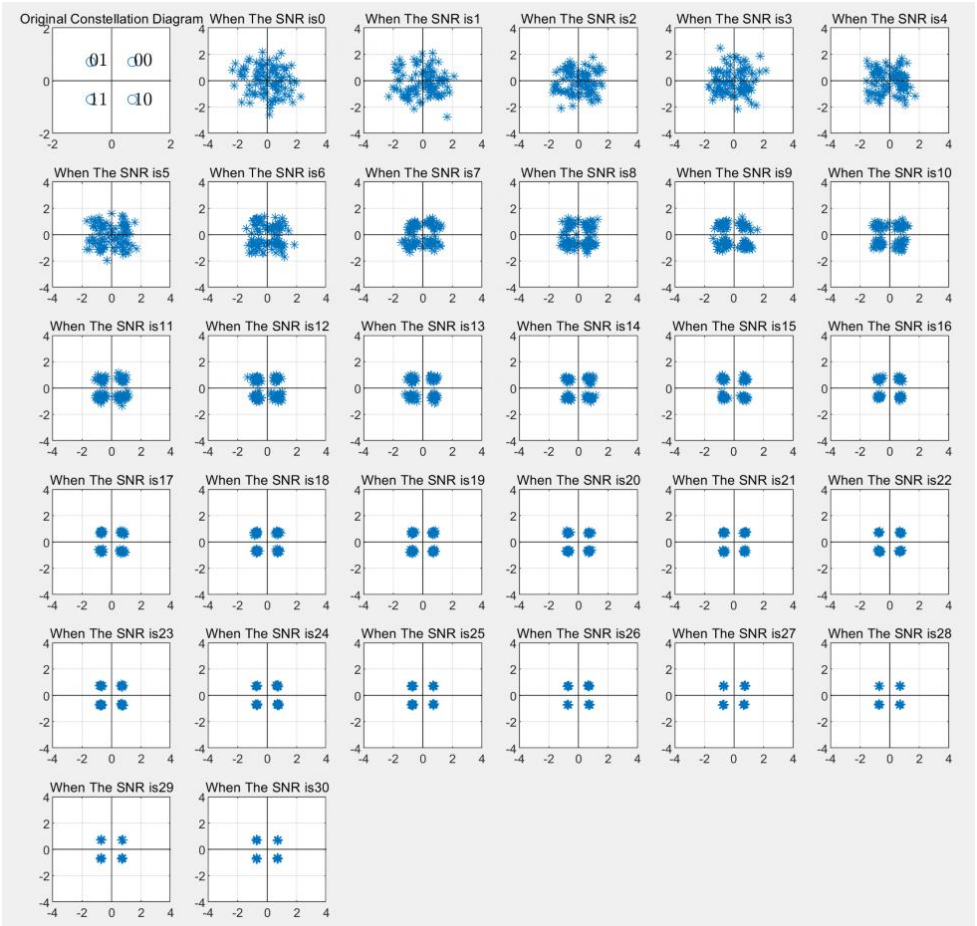
2. 模拟不同顺序的 BPSK/QAM 传输（AWGN 信道）

本实验需要使用蒙特卡罗仿真的原理进行实验仿真，Monte Carlo 仿真方法是通过大量的计算机模拟来检验系统的动态特性并归纳出统计结果的一种随机分析方法，它包括伪随机数的产生，Monte Carlo 仿真设计以及结果解释等内容，其作用在于用数学方法模拟真实物理环境，并验证系统的可靠性与可行性.

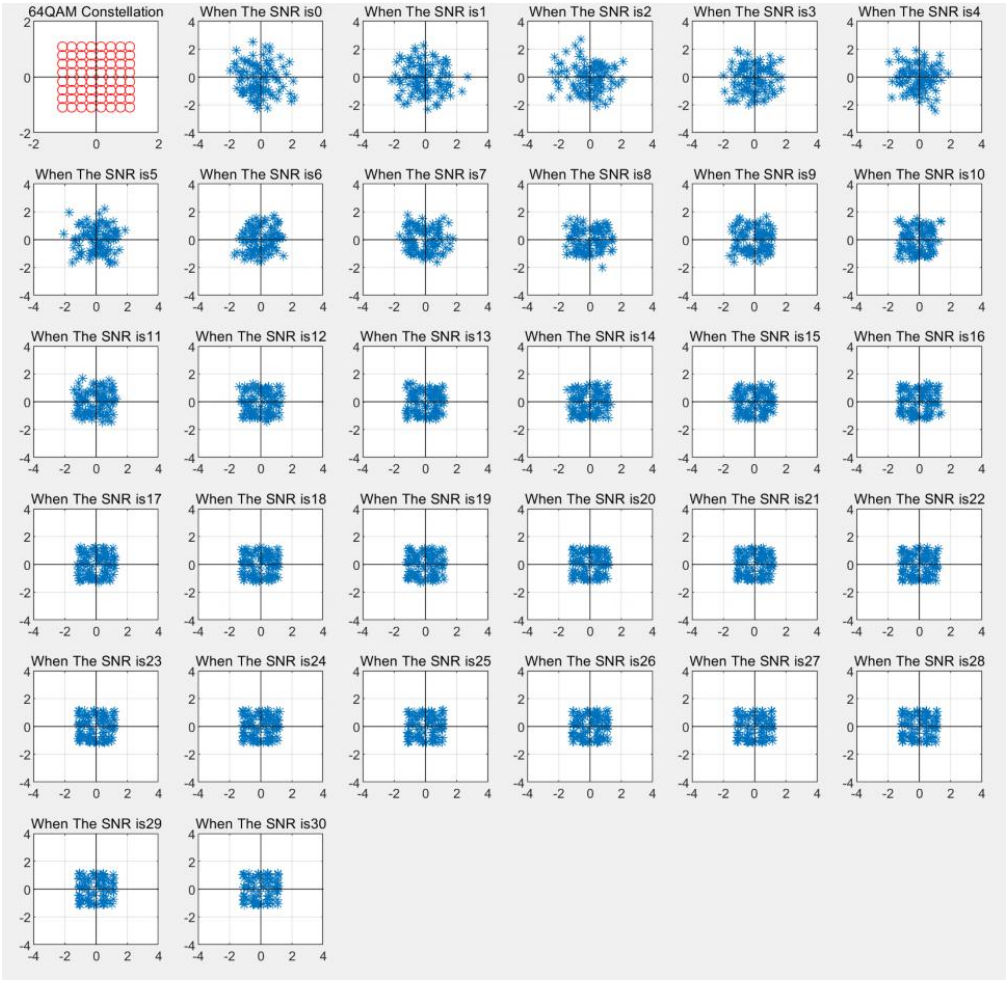
首先建立与描述该问题相似的概率模型，然后对模型进行随机模拟或统计抽样，再利用所得到的结果求出特征的统计估计值作为原问题的近似解，并对解的精度做出某些估计.

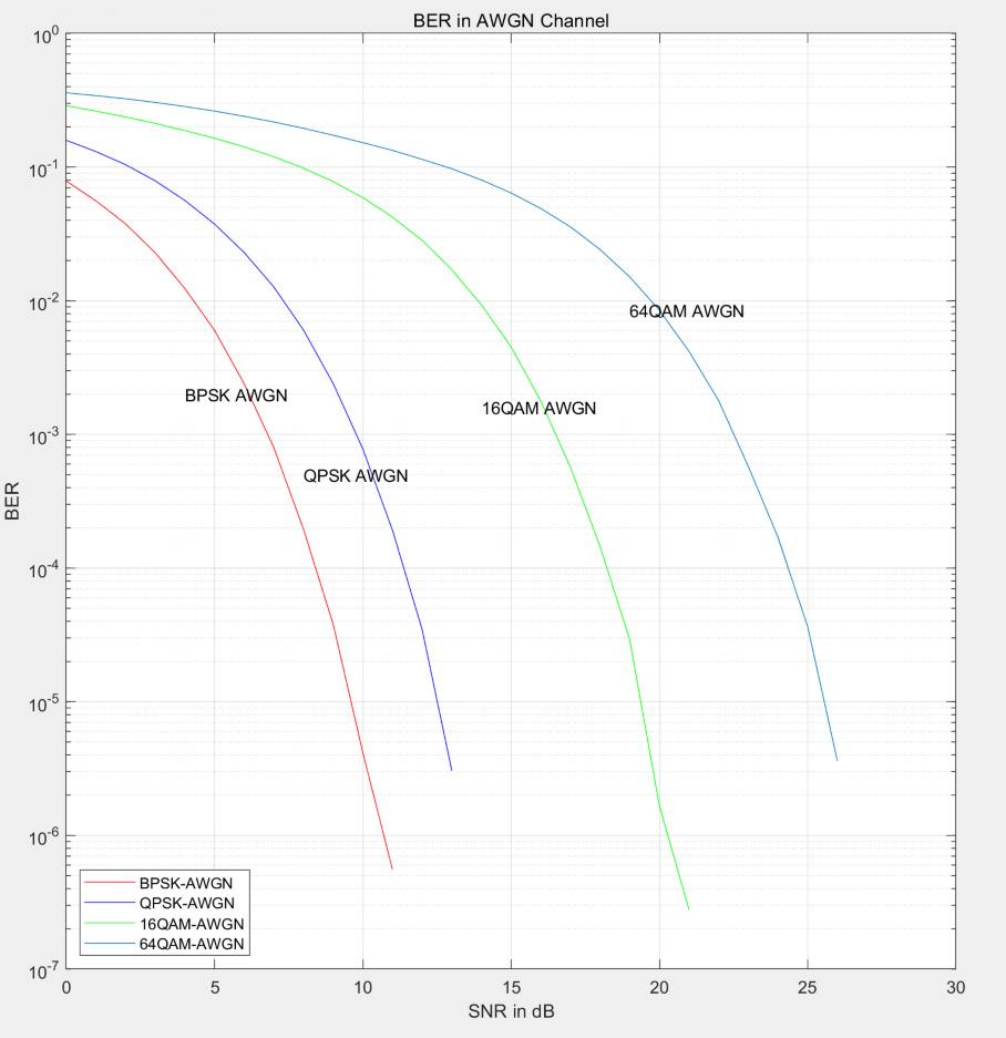
我们选择了 BPSK，QPSK，16QAM 以及 64QAM 四种调制解调方式进行了实验仿真，实验仿真结果图如下：


Original Constellation Diagram
0
1
When The SNR is0
When The SNR is1
When The SNR is2
When The SNR is3
When The SNR is4
When The SNR is5
When The SNR is6
When The SNR is7
When The SNR is8
When The SNR is9
When The SNR is10
When The SNR is11
When The SNR is12
When The SNR is13
When The SNR is14
When The SNR is15
When The SNR is16
When The SNR is17
When The SNR is18
When The SNR is19
When The SNR is20
When The SNR is21
When The SNR is22
When The SNR is23
When The SNR is24
When The SNR is25
When The SNR is26
When The SNR is27
When The SNR is28
When The SNR is29
When The SNR is30


Original Constellation Diagram
01
00
11
10
When The SNR is0
When The SNR is1
When The SNR is2
When The SNR is3
When The SNR is4
When The SNR is5
When The SNR is6
When The SNR is7
When The SNR is8
When The SNR is9
When The SNR is10
When The SNR is11
When The SNR is12
When The SNR is13
When The SNR is14
When The SNR is15
When The SNR is16
When The SNR is17
When The SNR is18
When The SNR is19
When The SNR is20
When The SNR is21
When The SNR is22
When The SNR is23
When The SNR is24
When The SNR is25
When The SNR is26
When The SNR is27
When The SNR is28
When The SNR is29
When The SNR is30


16QAM Constellation
When The SNR is0
When The SNR is1
When The SNR is2
When The SNR is3
When The SNR is4
When The SNR is5
When The SNR is6
When The SNR is7
When The SNR is8
When The SNR is9
When The SNR is10
When The SNR is11
When The SNR is12
When The SNR is13
When The SNR is14
When The SNR is15
When The SNR is16
When The SNR is17
When The SNR is18
When The SNR is19
When The SNR is20
When The SNR is21
When The SNR is22
When The SNR is23
When The SNR is24
When The SNR is25
When The SNR is26
When The SNR is27
When The SNR is28
When The SNR is29
When The SNR is30


64QAM Constellation
When The SNR is0
When The SNR is1
When The SNR is2
When The SNR is3
When The SNR is4
When The SNR is5
When The SNR is6
When The SNR is7
When The SNR is8
When The SNR is9
When The SNR is10
When The SNR is11
When The SNR is12
When The SNR is13
When The SNR is14
When The SNR is15
When The SNR is16
When The SNR is17
When The SNR is18
When The SNR is19
When The SNR is20
When The SNR is21
When The SNR is22
When The SNR is23
When The SNR is24
When The SNR is25
When The SNR is26
When The SNR is27
When The SNR is28
When The SNR is29
When The SNR is30



结果分析：为了实现 Monte Carlo 仿真方法，我们将每一个调制方式独立仿真了 36 万次，以此来寻找各种调制方式误码率在 10^{-3} 和 10^{-5} 时稳定的误码率.

我们对每增加 1dB 高斯白噪声都绘制一次图像，最终展现出每种调制方式从 0dB 噪声到 30dB 噪声的全过程星座图，直观明显地展现信噪比增加的过程中星座图的变化，最后绘制出一个高斯信道下的误比特率 BER 曲线，更加直观地展现出信噪比与误码率的关系.

经过对误比特率曲线图像的观察，我们可以看出 BPSK 方式的抗噪声性能强于 QPSK 方式，16QAM 的抗噪声性能强于 64QAM.

3. 尝试提高 8-QAM 的性能

首先我们先对星座图性能评价指标进行学习：

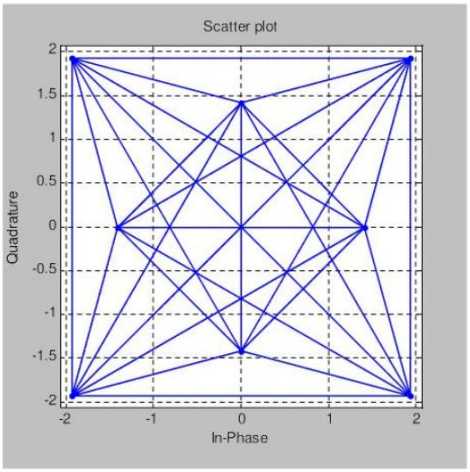
最小欧式距离：即信号星座图上星座点间的最小距离，该参数反映信号抗高斯白噪声的能力，其值越大，系统抗干扰能力越强.

最小相位偏移：即信号星座点相位的最小偏移，该参数反映信号抗相位抖动能力和对时钟恢复精确度的敏感性，其值越大，传输性能越好.

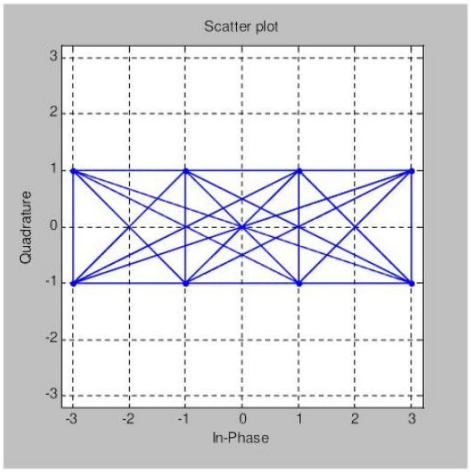
误比特率 BER 曲线：曲线越靠近左下，系统抗噪性能越好.

峰值——均值比：即星座图的峰值信噪比与均值信噪比的比值，值小一些更好，可以充分利用功放的输出功率.

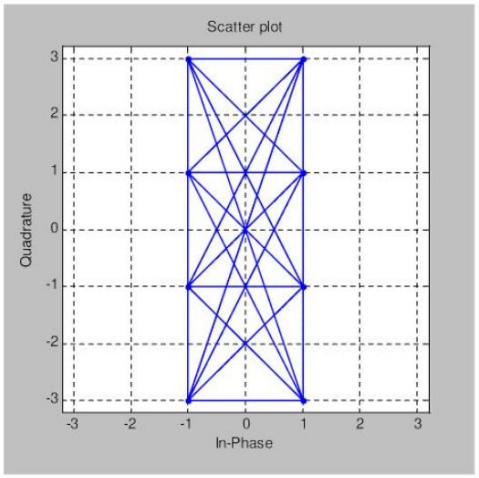
由于矩形星座图、圆形星座图具有通用性和代表性，所以我们取有代表性的三种作对比，此外我们定义了一个“最优”的 8-QAM 星座图并对它们进行了仿真，如下图：



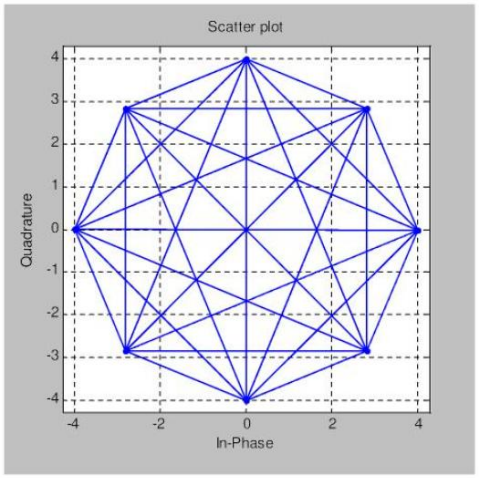
(a)最优



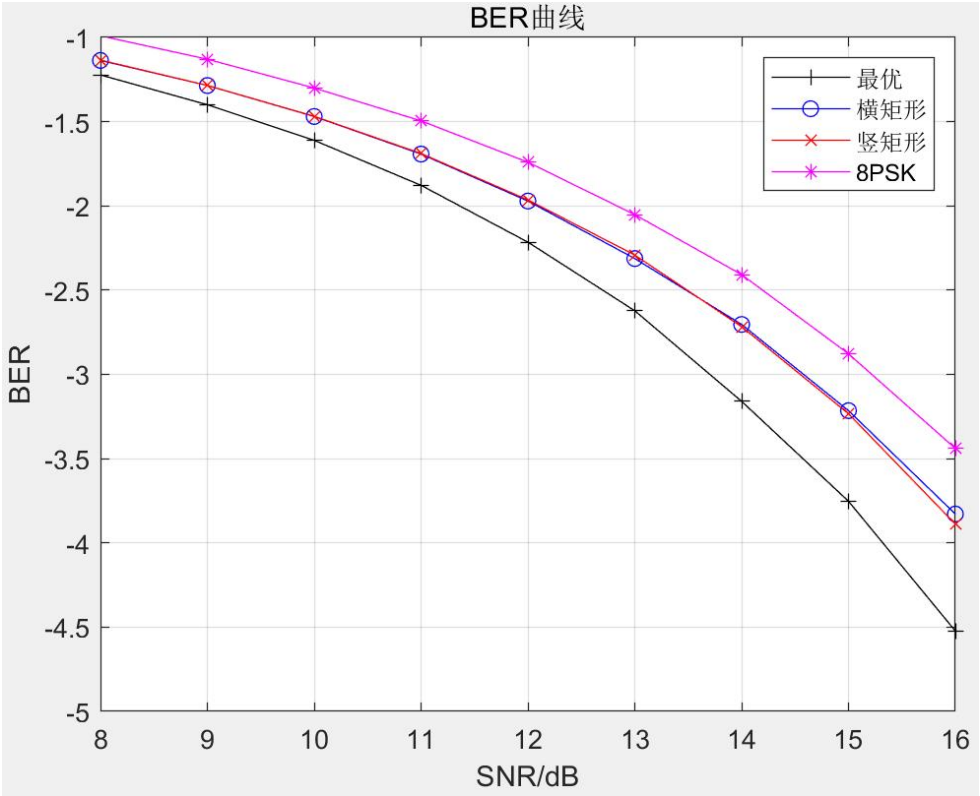
(b)横矩形



(c)竖矩形



(d)8PSK



结果分析：从误比特率 BER 图可以看出最优星座图的 BER 曲线最偏左下，横矩形星座图和竖矩形星座图的 BER 性能相近，圆形(8PSK)的 BER 性能最差.

通过三种常见的 8-QAM 星座图与最优星座图调制信号在 AWGN 信道中的误比特率曲线的对比，我们发现最优星座图的抗噪性能优于其他三种常见的 8-QAM 星座图.当然，我们也可以随机产生 8 个星座点和最优星座图对比，方法和之前的过程是类似的.

小组分工：

模型建立与代码实现：王晨阳，吴桐
结果测试与分析：马运聪，唐翠霜，夏汉宁
实验汇报与报告撰写：吴桐，马运聪，李政君

实验代码：

1. 生成不同顺序的 BPSK/QAM 信号（BPSK~64-QAM）

```
%可视化 8-QAM 调制的星座，使用 Gary 映射、位输入和星座功率归一化。
x1=randi([0,1],log2(8),1);
y1=qammod(x1,8,'InputType','bit','UnitAveragePower',true,'PlotConstellation
',true);
%可视化 16-QAM 调制的星座，使用 Gary 映射、位输入和星座功率归一化。
x2=randi([0,1],log2(16),1);
y2=qammod(x2,16,'InputType','bit','UnitAveragePower',true,'PlotConstellatio
n',true);
%可视化 32-QAM 调制。默认值：整数输入，Gary 编码，星座点间最小距离为 2
```

```
x3=randi([0,1],log2(32),1);
y3=qammod(x3,32,'InputType','bit','UnitAveragePower',true,'PlotConstellatio
n',true);
%可视化 64-QAM 调制的星座，使用 Gary 映射、位输入和星座功率归一化。
x4=randi([0,1],log2(64),1);
y4=qammod(x4,64,'InputType','bit','UnitAveragePower',true,'PlotConstellatio
n',true);
```

2. 模拟不同顺序的 BPSK/QAM 传输（AWGN 信道）

```
%%
clear;clc;
% Generation of Bits stream
N=3600000; %Number of Bits
Bits_Sequence=randi([0 1],1,N); % original bits stream
%%
%BPSK Modulation
BPSK_Sequence=2.*Bits_Sequence-1; % uni-polar to bi-polar
% Adds Gaussian Noise to BPSK signal and Calculate the Bit Error Rate
SNR_dB = [0:1:30];
BPSK_BER_list=[];
% Received BPSK Signals
figure('name','BPSK','Position',[800,100,1000,1000]);
subplot(6,6,1);
plot(real(BPSK_Sequence),imag(BPSK_Sequence),'o');
grid on;
hold on;
line([-3,3],[0,0],'Color','k');
line([0,0],[-3,3],'Color','k');
text(1.2,0.2,'1','FontSize',12,'Interpreter','latex');
text(-1.2,0.2,'0','FontSize',12,'Interpreter','latex');
title('Original Constellation Diagram');
xlim([-2,2]);
ylim([-2,2]);
hold off;
len = length(BPSK_Sequence);
for i=1:1:size(SNR_dB,2)
N0 = 1/10^(SNR_dB(i)/10); % Noise variance
Noise = sqrt(N0/2)*((randn(1,len)+1i*randn(1,len))); % AWGN Noise
Rx_BPSK_sig = BPSK_Sequence + Noise; % Received signal
subplot(6,6,i+1)
plot(real(Rx_BPSK_sig(1:100)),imag(Rx_BPSK_sig(1:100)),'*');
grid on
hold on
line([-5,5],[0,0],'Color','k');
```

```
line([0,0],[-5,5],'Color','k');
title(['When The SNR is',num2str(SNR_dB(i))]);
xlim([-4,4]);
ylim([-4,4]);
hold off
% Demodulation
De_BPSK_Bits = (real(Rx_BPSK_sig)>0); %Demodulated Sequence
Error_bits_BPSK = Bits_Sequence - De_BPSK_Bits; %Calculate Bit Errors
BPSK_BER_list(end+1) = sum(abs(Error_bits_BPSK))/N; % BER
end
%%
%QPSK AWGN
% Split the bits into two streams
Bits1 = Bits_Sequence(1:2:end); % Odd order (shape 1*5000)
Bits2 = Bits_Sequence(2:2:end); % Even order
% QPSK pi/4 radians constellation
qpsk_sig =(((Bits1==0).*(Bits2==0)*(exp(1i*pi/4))+(Bits1==0).*(Bits2==1)...
*(exp(3*1i*pi/4))+(Bits1==1).*(Bits2==1)*(exp(5*1i*pi/4))...
+(Bits1==1).*(Bits2==0)*(exp(7*1i*pi/4))));
% Adds Gaussian Noise to QPSK signal
SNR_dB = [0:1:30];
QPSK_BER_list=[];
figure('name','QPSK','Position',[800,100,1000,1000]);
subplot(6,6,1)
plot(real(qpsk_sig),imag(qpsk_sig),'o');
grid on
hold on
line([-3,3],[0,0],'Color','k');
line([0,0],[-3,3],'Color','k');
text(0.75,0.75,'00','FontSize',12,'Interpreter','latex');
text(-0.75,0.75,'01','FontSize',12,'Interpreter','latex');
text(-0.75,-0.75,'11','FontSize',12,'Interpreter','latex');
text(0.75,-0.75,'10','FontSize',12,'Interpreter','latex');
title('Original Constellation Diagram');
xlim([-2,2]);
ylim([-2,2]);
hold off
len = length(qpsk_sig);
for i=1:1:size(SNR_dB,2)
N0 = 1/10^(SNR_dB(i)/10); % Noise variance
Noise = sqrt(N0/2)*((randn(1,len)+1i*randn(1,len))); % AWGN Noise
Rx_QPSK_sig = qpsk_sig + Noise; % Received signal
subplot(6,6,i+1)
```

```
plot(real(Rx_QPSK_sig(1:100)),imag(Rx_QPSK_sig(1:100)),'*'); % Select some of
the points
grid on
hold on
line([-5,5],[0,0],'Color','k');
line([0,0],[-5,5],'Color','k');
title(['When The SNR is',num2str(SNR_dB(i))]);
xlim([-4,4]);
ylim([-4,4]);
hold off
% Demodulation
Bits4 = (real(Rx_QPSK_sig)<0);
Bits3 = (imag(Rx_QPSK_sig)<0);
Demod_qpsk_bits = zeros(1,2*length(Rx_QPSK_sig)); % Demodulated bits
Demod_qpsk_bits(1:2:end) = Bits3;
Demod_qpsk_bits(2:2:end) = Bits4;
Error_bits_QPSK = Bits_Sequence - Demod_qpsk_bits; %Calculate Bit Errors
QPSK_BER_list(end+1) = sum(abs(Error_bits_QPSK))/N; % BER
end
%%
%16-QAM AWGN
% Split the bits into four streams
%Source
%rand() generate 1*num_bits shape matrix with number randomly from (0-1)
QAM16_Bits = reshape(Bits_Sequence,4,length(Bits_Sequence)/4);
Bits1 = QAM16_Bits(1,:);
Bits2 = QAM16_Bits(2,:);
Bits3 = QAM16_Bits(3,:);
Bits4 = QAM16_Bits(4,:);
% normalizing factor
normalizer = sqrt(1/10); % (2*4+10*8+18*4)/16=10
% bit mapping
qam16_sig =
normalizer*(1j.*(2.*Bits2-1).*(-2.*Bits3+3)+(2.*Bits1-1).*(-2.*Bits4+3));
% Adds Gaussian Noise to QPSK signal
SNR_dB=[0:1:30];
QAM16_BER_list=[];
figure('name','16QAM','Position',[800,100,1000,1000]);
subplot(6,6,1)
plot(real(qam16_sig),imag(qam16_sig),'ro');
grid on
hold on
line([-3,3],[0,0],'Color','k');
line([0,0],[-3,3],'Color','k');
```

```
text(0.3562,0.3562,'1111','FontSize',12,'Interpreter','latex');
text(-0.3562,0.3562,'0111','FontSize',12,'Interpreter','latex');
text(-0.3562,-0.3562,'0011','FontSize',12,'Interpreter','latex');
text(0.3562,-0.3562,'1011','FontSize',12,'Interpreter','latex');
text(0.9987,0.9987,'1100','FontSize',12,'Interpreter','latex');
text(-0.9987,0.9987,'0100','FontSize',12,'Interpreter','latex');
text(-0.9987,-0.9987,'0000','FontSize',12,'Interpreter','latex');
text(0.9987,-0.9987,'1000','FontSize',12,'Interpreter','latex');
text(0.3562,0.9987,'1101','FontSize',12,'Interpreter','latex');
text(-0.3562,0.9987,'0101','FontSize',12,'Interpreter','latex');
text(-0.3562,-0.9987,'0001','FontSize',12,'Interpreter','latex');
text(0.3562,-0.9987,'1001','FontSize',12,'Interpreter','latex');
text(0.9987,0.3562,'1110','FontSize',12,'Interpreter','latex');
text(-0.9987,0.3562,'0110','FontSize',12,'Interpreter','latex');
text(-0.9987,-0.3562,'0010','FontSize',12,'Interpreter','latex');
text(0.9987,-0.3562,'1010','FontSize',12,'Interpreter','latex');
title('16QAM Constellation');
xlim([-2,2]);
ylim([-2,2]);
hold off
len = length(qam16_sig);
for i=1:1:size(SNR_dB,2)
N0 = 1/10^(SNR_dB(i)/10); % Noise variance
Noise = sqrt(N0/2)*((randn(1,len)+1i*randn(1,len))); % AWGN Noise
Rx_qam16_sig = qam16_sig + Noise; % Received signal
subplot(6,6,i+1)
plot(real(Rx_qam16_sig(1:100)),imag(Rx_qam16_sig(1:100)),'*'); % Select some
of the points
grid on
hold on
line([-5,5],[0,0],'Color','k');
line([0,0],[-5,5],'Color','k');
title(['When The SNR is',num2str(SNR_dB(i))]);
xlim([-4,4]);
ylim([-4,4]);
hold off
% Demodulation
ab = 1/sqrt(10);
Bits6 = imag(Rx_qam16_sig)>0;
Bits7 = (imag(Rx_qam16_sig)<2*ab) & (imag(Rx_qam16_sig)>-2*ab);
Bits5 = real(Rx_qam16_sig)>0;
Bits8 = (real(Rx_qam16_sig)<2*ab) & (real(Rx_qam16_sig)>-2*ab);
% Combine into single stream
comb = [Bits5; Bits6; Bits7; Bits8];
```

```
Demod_qam16_bits = reshape(comb,1,4*length(comb));
Error_bits_qam16 = Bits_Sequence - Demod_qam16_bits; % Calculate Bit Errors
QAM16_BER_list(end+1) = sum(abs(Error_bits_qam16))/N; % BER
end

%%
%64-QAM AWGN
% Split the bits into four streams
%Source
QAM64_Bits = reshape(Bits_Sequence,6,length(Bits_Sequence)/6);
Bits1 = QAM64_Bits(1,:);
Bits2 = QAM64_Bits(2,:);
Bits3 = QAM64_Bits(3,:);
Bits4 = QAM64_Bits(4,:);
Bits5 = QAM64_Bits(5,:);
Bits6 = QAM64_Bits(6,:);
QAM64_Bits = zeros(1,length(Bits_Sequence)/6);
for i=1:1:length(Bits_Sequence)/6
QAM64_Bits(i)=32*Bits1(i)+16*Bits2(i)+8*Bits3(i)+4*Bits4(i)+2*Bits5(i)+Bits
6(i);
end
% normalizing factor
normalizer = sqrt(1/42); %
(2*4+10*8+18*4+26*8+34*8+50*4+50*8+58*8+74*8+98*4)/64=42
% bit mapping
qam64_sig = normalizer*qammod(QAM64_Bits,64);
% Adds Gaussian Noise to QPSK signal
SNR_dB=[0:1:30];
QAM64_BER_list=[];
% Constellation of 64 QAM
figure('name','64QAM','Position',[800,100,1000,1000]);
subplot(6,6,1)
plot(real(qam64_sig),imag(qam64_sig),'ro'); grid on;hold on;
line([-10,10],[0,0],'Color','k');
line([0,0],[-10,10],'Color','k');
title('64QAM Constellation');
xlim([-2,2]);
ylim([-2,2]);
hold off;
len = length(qam64_sig);
for i=1:1:size(SNR_dB,2)
N0 = 1/10^(SNR_dB(i)/10); % Noise variance
Noise = sqrt(N0/2)*((randn(1,len)+1i*randn(1,len))); % AWGN Noise
Rx_qam64_sig = qam64_sig + Noise; % Received signal
```

```
subplot(6,6,i+1)
plot(real(Rx_qam64_sig(1:100)),imag(Rx_qam64_sig(1:100)),'*'); % Select some
of the points
grid on;
hold on;
line([-5,5],[0,0],'Color','k');
line([0,0],[-5,5],'Color','k');
title(['When The SNR is',num2str(SNR_dB(i))]);
xlim([-4,4]);
ylim([-4,4]);
hold off
% Demodulation
de_qam64_sym=qamdemod(Rx_qam64_sig/normalizer,64);
de_Bits1=zeros(1,length(de_qam64_sym));
de_Bits2=zeros(1,length(de_qam64_sym));
de_Bits3=zeros(1,length(de_qam64_sym));
de_Bits4=zeros(1,length(de_qam64_sym));
de_Bits5=zeros(1,length(de_qam64_sym));
de_Bits6=zeros(1,length(de_qam64_sym));
for j=1:1:length(de_qam64_sym)
buf=de_qam64_sym(j);
de_Bits1(j)=floor(buf/32);
de_Bits2(j)=floor((buf-32*de_Bits1(j))/16);
de_Bits3(j)=floor((buf-32*de_Bits1(j)-16*de_Bits2(j))/8);
de_Bits4(j)=floor((buf-32*de_Bits1(j)-16*de_Bits2(j)-8*de_Bits3(j))/4);
de_Bits5(j)=floor((buf-32*de_Bits1(j)-16*de_Bits2(j)-8*de_Bits3(j)-4*de_Bit
s4(j))/2);
de_Bits6(j)=floor((buf-32*de_Bits1(j)-16*de_Bits2(j)-8*de_Bits3(j)-4*de_Bit
s4(j)-2*de_Bits5(j)));
end
% Combine into single stream
comb = [de_Bits1; de_Bits2; de_Bits3; de_Bits4;de_Bits5;de_Bits6];
Demod_qam64_bits = reshape(comb,1,6*length(comb));
Error_bits_qam64 = Bits_Sequence - Demod_qam64_bits; % Calculate Bit Errors
QAM64_BER_list(end+1) = sum(abs(Error_bits_qam64))/N; % BER
end
%%
figure('name','BER in different Channel','Position',[500,100,900,900]);
semilogy(SNR_dB,BPSK_BER_list,'r');text(4,0.002,'BPSK AWGN');
grid on;hold on;
semilogy(SNR_dB,QPSK_BER_list,'b');text(8,0.0005,'QPSK AWGN');
semilogy(SNR_dB,QAM16_BER_list,'g');text(14,0.0016,'16QAM AWGN');
semilogy(SNR_dB,QAM64_BER_list);text(19,0.0085,'64QAM AWGN');
```

```
legend('BPSK-AWGN','QPSK-AWGN','16QAM-AWGN','64QAM-AWGN','Location','SouthWest');
title('BER in AWGN Channel');xlabel('SNR in dB');ylabel('BER');
```

3. 尝试提高 8-QAM 的性能

```
M=8; %8QAM
k=log2(M); %每个码元对应的 bit 数
n=1.2e6; %比特序列长度
samp=1; %过采样率
coderate=1; %编码码率
x=randi([0 1],1,n); %产生 1 行 n 列即 n 个取值为 0 或 1 的随机数组
xreshape=reshape(x, n/k, k); %将二进制比特流数组转换为 n/k 行 k 列的符号数组
xsym=bi2de (xreshape,'left-msb') ;
xsym=bi2de(xreshape,'left-msb');

h_mod1=modem.genqammod('Constellation',[-(2+3^0.5)^0.5+1j*(2+3^0.5)^0.5,(2+3^0.5)^0.5+1j*(2+3^0.5)^0.5,1j*2^0.5,2^0.5,-(2+3^0.5)^0.5-1j*(2+3^0.5)^0.5,(2+3^0.5)^0.5-1j*(2+3^0.5)^0.5,-2^0.5,-1j*2^0.5,], 'InputType', 'integer');
h_demod1 =
modem.genqamdemod('Constellation',[-(2+3^0.5)^0.5+1j*(2+3^0.5)^0.5,(2+3^0.5)^0.5+1j*(2+3^0.5)^0.5,1j*2^0.5,2^0.5,-(2+3^0.5)^0.5-1j*(2+3^0.5)^0.5,(2+3^0.5)^0.5-1j*(2+3^0.5)^0.5,-2^0.5,-1j*2^0.5,],'OutputType', 'integer');
h_mod2 = modem.genqammod('Constellation',
[ -3+1j*1,-3-1j*1,-1+1j*1,-1-1j*1,1+1j*1,1-1j*1,3+1j*1,3-1j*1,],'InputType',
'integer');
h_demod2 =
modem.genqamdemod('Constellation',[ -3+1j*1,-3-1j*1,-1+1j*1,-1-1j*1,1+1j*1,
1-1j*1,3+1j*1,3-1j*1,],'OutputType','integer');
h_mod3 = modem.genqammod('Constellation',
[ -1+1j*3,-1+1j*1,-1-1j*1,-1-1j*3,1+1j*3,1+1j*1,1-1j*1,1-1j*3,],'InputType',
'integer');
h_demod3 = modem.genqamdemod( 'Constellation',
[ -1+1j*3,-1+1j*1,-1-1j*1,-1-1j*3,1+1j*3,1+1j*1,1-1j*1,1-1j*3,],'OutputType',
'integer');
h_mod4 =
modem.genqammod('Constellation',4*exp(1j*2*pi*[0:M-1]/M),'InputType','integer');
h_demod4 =
modem.genqamdemod('Constellation' ,4*exp(1j*2*pi*[0:M-1]/M) ,'OutputType','inteGer');

y1=modulate(h_mod1, xsym);
y2=modulate(h_mod2, xsym);
y3=modulate(h_mod3, xsym);
```

```
y4=modulate(h_mod4, xsym);
snr = 7:1:25;
num=length(snr);
num_of_errors=zeros(4,num);
bit_error_rate=zeros(4,num) ;

for m=1:num
yn1=awgn(y1, snr(m), 'measured');
yn2=awgn(y2, snr(m), 'measured');
yn3=awgn(y3, snr(m), 'measured');
yn4=awgn(y4, snr(m), 'measured');

zsym1=demodulate(h_demod1, yn1);
zsym2=demodulate(h_demod2, yn2);
zsym3=demodulate(h_demod3, yn3);
zsym4=demodulate(h_demod4, yn4);

z1 = reshape(de2bi(zsym1,'left-msb'),1,n);
z2 = reshape(de2bi(zsym2,'left-msb'),1,n);
z3 = reshape(de2bi(zsym3,'left-msb'),1,n);
z4 = reshape(de2bi(zsym4,'left-msb'),1,n);

[num_of_errors(1,m),bit_error_rate(1,m)]=biterr(x,z1);
[num_of_errors(2,m),bit_error_rate(2,m)]=biterr(x,z2);
[num_of_errors(3,m),bit_error_rate(3,m)]=biterr(x,z3);
[num_of_errors(4,m),bit_error_rate(4,m)]=biterr(x,z4);
end

plot(snr, log10(bit_error_rate(1,1:num)),'-+k');
xlabel('SNR/dB'),ylabel('BER');
title( 'BER 曲线');
axis([8 17 -6.5 -1]);
grid on
hold on
plot(snr,log10(bit_error_rate(2, 1:num)),'-ob');
hold on
plot(snr,log10(bit_error_rate(3, 1:num)),'-xr');
hold on
plot(snr,log10(bit_error_rate(4, 1:num)) ,'-*m');
legend('最优','横矩形','竖矩形','8PSK');
```