# 第三次实验报告

## 实验目的

1. 掌握 BPSK 调制解调原理；

2. 掌握 QPSK 调制解调原理；

3. 掌握 BPSK 调制解调原理；

4. 理解噪声对 BPSK，QPSK，QAM 相干解调性能的影响。

## 实验1

### **1.1实验内容**

用MATLAB产生256bit独立等概率的二进制信源，码元持续时间为1s，仿真信号在AWGN信道中的BPSK调制解调过程，其中载波幅值为1，频率为 20Hz，判决门限为 0。

（1）当解调器输入信噪比为0dB时，分别画出调制信号、BPSK已调信号、低通滤波器输出信号和解调信号；

（2）画出解调器输入信噪比在-15dB到5dB时的接收机理论误码率曲线与实际误码率曲线(20次以上平均)；

### **1.2实验程序**

clear;

N = 256; % 产生码元数

Ts = 1; % 码元持续时间

A = 1; % 载波幅值

fc = 20; % 载波频率

d = 0; % 判决门限

L = 200; % 每个码元的采样点数

dt = Ts/L; % 采样间隔

T = N\*Ts; % 总时间

t = 0:dt:T-dt; % 时间

% 生成调制信号

b = randi([0,1],1,N); % 二进制随机码

s = signal\_expand(b,L); % 调制信号

subplot(511)

plot(t,s);

title('调制信号');

xlabel('t');

ylabel('幅度');

axis([0,10,0,1.2]);

grid on;

% BPSK

c = A\*cos(2\*pi\*fc\*t); % 载波

s\_BPSK = c.\*(2\*s-1);

subplot(512)

plot(t,s\_BPSK);

title('BPSK已调信号');

xlabel('t');

ylabel('幅度');

axis([0,10,-1.2,1.2]);

% 噪声

SNRdb = 0; % 信噪比

SNR = 10^(SNRdb/10);

B = 1; % 基带带宽

B\_bpf = 2\*(B+fc); % 理想带通滤波器带宽

P = (norm(s\_BPSK(1:L)).^2)./length(s\_BPSK(1:L)); % 平均功率

n0 = (P/SNR)/B\_bpf; % 噪声功率谱密度

u = sqrt(n0\*L\*fc/2)\*randn(1,L\*N); % 噪声

s\_BPSK\_u = s\_BPSK+u; % 经过信道的已调信号

% 滤波器

[f,S\_BPSK\_u] = T2F(t,s\_BPSK\_u); % 频域

S\_BPSK\_u\_BPF = BPF(f,-(fc+B),fc+B,1).\*S\_BPSK\_u; % 经过带通滤波器

[t,s\_BPSK\_u\_BPF] = F2T(f,S\_BPSK\_u\_BPF); % 时域

s\_BPSK\_u\_BPF\_c = s\_BPSK\_u\_BPF.\*c; % 与载波相乘

[f,S\_BPSK\_u\_BPF\_c] = T2F(t,s\_BPSK\_u\_BPF\_c); % 频域

S\_BPSK\_u\_BPF\_c\_LPF = LPF(B,f,1).\*(S\_BPSK\_u\_BPF\_c); % 经过低通滤波器

[t,s\_BPSK\_u\_BPF\_c\_LPF] = F2T(f,S\_BPSK\_u\_BPF\_c\_LPF); % 时域

subplot(513)

plot(t,s\_BPSK\_u\_BPF\_c\_LPF);

title('低通滤波器输出信号');

xlabel('t');

ylabel('幅度');

axis([0,10,-1.5,1.5]);

grid on;

% 解调

s\_BPSK = BPSK\_SJ(s\_BPSK\_u\_BPF\_c\_LPF,L,d); % 抽样判决

subplot(514)

plot(t,s\_BPSK);

title('解调信号');

xlabel('t');

ylabel('幅度');

axis([0,10,0,1.2]);

% 信噪比

SNR1 = -15; % 最小信噪比

SNR2 = 5; % 最大信噪比

T = 30; % 次数

n = SNR1:SNR2; % 信噪比横轴

Pe\_theory = 0.5\*erfc(sqrt(10.^(n/10))); % 理论误码率

Pe\_practice = zeros(1,length(n)); % 实际误码率

for SNR = SNR1:SNR2

    for i = 1:T

        b = randi([0,1],1,N); % 二进制随机码

        s = signal\_expand(b,L); % 调制信号

        out = BPSK(t,s,A,fc,L,d,B,SNR);  % 进行调制解调

        out = out - s;  % 信号差

        Pe\_practice(-SNR1+SNR+1) = Pe\_practice(-SNR1+SNR+1) + (sum(abs(out),'double')/L/N);  % 误码累计

    end

    Pe\_practice(-SNR1+SNR+1)=Pe\_practice(-SNR1+SNR+1)/T; % 平均误码率

end

subplot(515)

plot(n,Pe\_theory);

hold on;

plot(n,Pe\_practice);

legend('理论误码率','实际误码率');

title('误码率曲线');

xlabel('信噪比');

ylabel('误码率');

### **1.3实验结果**

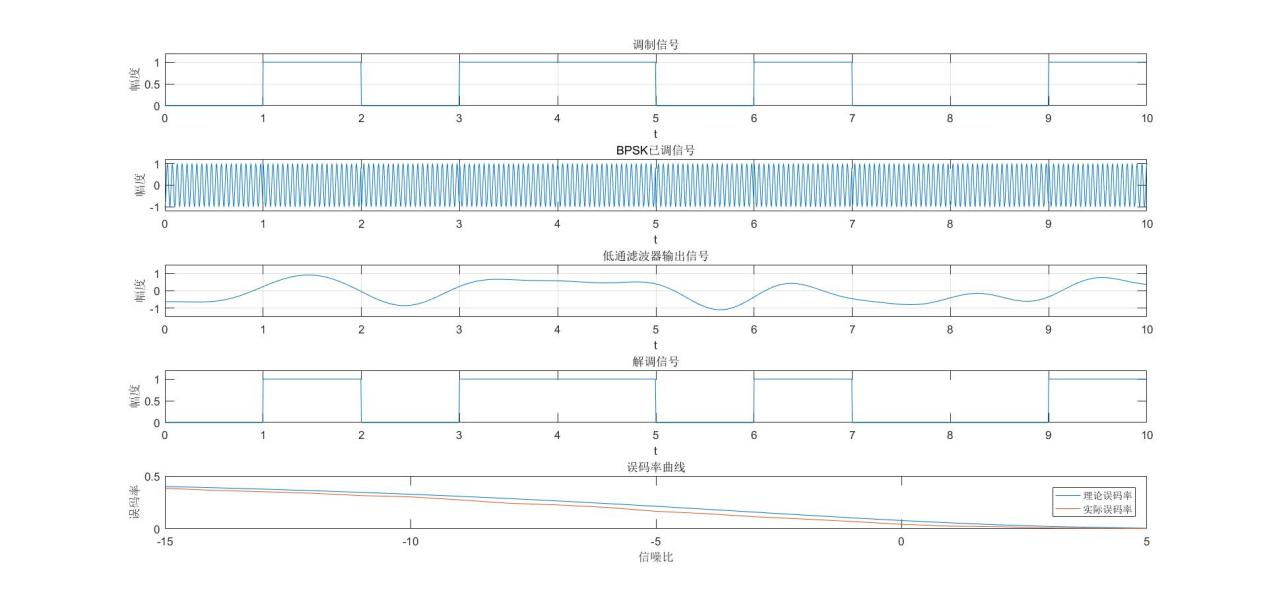


图 1 实验1结果图片

### **1.4实验分析**

本次实验需要将256bit二进制信源进行扩展以与载波达到同一维度，就可以直接与载波相乘完成调制，由于题目给出的是解调器输入信噪比，所以在计算噪声的功率谱密度的公式应为。最终的抽样判决也是选取中间时间的值作为依据，与判决门限进行比较。

BPSK理论误码率为，在计算实际误码率时，可以将上面的过程封装成函数进行调用，计算多次误码率求平均当作最后的实际误码率。

## 实验2

### **2.1实验内容**

用MATLAB产生256bit 独立等概率的二进制信源，码元持续时间为1s，仿真信号在AWGN信道中的QPSK调制解调过程，其中载波幅值为1，频率为20Hz，判决门限为0。

（1）当解调器输入信噪比为0dB时，分别画出调制信号、QPSK已调信号、低通滤波器输出信号和解调信号；

（2）画出解调器输入信噪比在-15dB到5dB时的接收机理论误码率曲线与实际误码率曲线(20次以上平均)；

### **2.2实验程序**

clear;

N = 256; % 产生码元数

Ts = 1; % 码元持续时间

A = 1; % 载波幅值

fc = 20; % 载波频率

d = 0; % 判决门限

L = 1000; % 每个码元的采样点数

dt = Ts/L; % 采样间隔

T = N\*Ts; % 总时间

t = 0:dt:T-dt; % 时间

t\_ = 0:dt:N/2-dt;

% 生成调制信号

b = randi([0,1],1,N); % 二进制随机码

s = signal\_expand(b,L); % 调制信号

s1\_ = zeros(1,N/2); % \_0和\_1

s2\_ = zeros(1,N/2); % 0\_和1\_

for i = 2:2:N % 串并转换

    if b(i-1)==0

        s1\_(round(i/2)) = -1;

    else

        s1\_(round(i/2)) = 1;

    end

    if b(i)==0

        s2\_(round(i/2)) = -1;

    else

        s2\_(round(i/2)) = 1;

    end

end

s1 = signal\_expand(s1\_,L);

s2 = signal\_expand(s2\_,L);

subplot(511)

plot(t,s);

title('调制信号');

xlabel('t');

ylabel('幅度');

axis([0,10,0,1.2]);

grid on;

% QPSK

c1 = A\*cos(2\*pi\*fc\*t\_); % 载波1

c2 = -A\*sin(2\*pi\*fc\*t\_); % 载波2

s\_QPSK = s1.\*c1+s2.\*c2;

subplot(512)

plot(t\_,s\_QPSK);

title('QPSK已调信号');

xlabel('t');

ylabel('幅度');

axis([0,10,-1.5,1.5]);

% 噪声

SNRdb = 0; % 信噪比

SNR = 10^(SNRdb/10);

B = 2; % 基带带宽

B\_bpf = 2\*(B+fc); % 理想带通滤波器带宽

P = (norm(s\_QPSK(1:L)).^2)./length(s\_QPSK(1:L)); % 平均功率

n0 = (P/SNR)/B\_bpf; % 噪声功率谱密度

u = sqrt(n0\*L\*fc/2)\*randn(1,L\*N/2); % 噪声

s\_QPSK\_u = s\_QPSK+u; % 经过信道的已调信号

% 滤波器

[f,S\_QPSK\_u] = T2F(t\_,s\_QPSK\_u); % 频域

S\_QPSK\_u\_BPF = BPF(f,-(fc+B),fc+B,1).\*S\_QPSK\_u; % 经过带通滤波器

[~,s\_QPSK\_u\_BPF] = F2T(f,S\_QPSK\_u\_BPF); % 时域

s1\_QPSK\_u\_BPF\_c = s\_QPSK\_u\_BPF.\*c1; % 与载波相乘

s2\_QPSK\_u\_BPF\_c = s\_QPSK\_u\_BPF.\*c2;

[~,S1\_QPSK\_u\_BPF\_c] = T2F(t\_,s1\_QPSK\_u\_BPF\_c); % 频域

[~,S2\_QPSK\_u\_BPF\_c] = T2F(t\_,s2\_QPSK\_u\_BPF\_c);

S1\_QPSK\_u\_BPF\_c\_LPF = LPF(B,f,2).\*(S1\_QPSK\_u\_BPF\_c); % 经过低通滤波器

S2\_QPSK\_u\_BPF\_c\_LPF = LPF(B,f,2).\*(S2\_QPSK\_u\_BPF\_c);

[~,s1\_QPSK\_u\_BPF\_c\_LPF] = F2T(f,S1\_QPSK\_u\_BPF\_c\_LPF); % 时域

[~,s2\_QPSK\_u\_BPF\_c\_LPF] = F2T(f,S2\_QPSK\_u\_BPF\_c\_LPF);

subplot(513)

plot(t\_,s1\_QPSK\_u\_BPF\_c\_LPF);

hold on;

plot(t\_,s2\_QPSK\_u\_BPF\_c\_LPF);

title('低通滤波器输出信号');

xlabel('t');

ylabel('幅度');

legend('同向分量','正交分量');

axis([0,10,-4,4]);

grid on;

% 解调

s1\_QPSK = QPSK\_SJ(s1\_QPSK\_u\_BPF\_c\_LPF,L,d); % 抽样判决

s2\_QPSK = QPSK\_SJ(s2\_QPSK\_u\_BPF\_c\_LPF,L,d);

s\_ = zeros(1,N); % 初始化

for i = 1:1:round(N/2) % 串并转换

    s\_(2\*i-1:2\*i) = [s1\_QPSK(i),s2\_QPSK(i)];

end

s = signal\_expand(s\_,L);

subplot(514)

plot(t,s);

title('解调信号');

xlabel('t');

ylabel('幅度');

axis([0,10,0,1.2]);

% 信噪比

SNR1 = -15; % 最小信噪比

SNR2 = 5; % 最大信噪比

T = 30; % 次数

n = SNR1:SNR2; % 信噪比横轴

Pe\_theory = 0.5\*erfc(sqrt(10.^(n/10)/2)); % 理论误码率

Pe\_practice = zeros(1,length(n)); % 实际误码率

for SNR = SNR1:SNR2

    for i = 1:T

        b = randi([0,1],1,N); % 二进制随机码

        out = QPSK(N,t\_,b,A,fc,L,d,B,SNR);  % 进行调制解调

        out = out - b;  % 信号差

        Pe\_practice(-SNR1+SNR+1) = Pe\_practice(-SNR1+SNR+1) + (sum(abs(out),'double')/N);  % 误码累计

    end

    Pe\_practice(-SNR1+SNR+1)=Pe\_practice(-SNR1+SNR+1)/T; % 平均误码率

end

subplot(515)

semilogy(n,Pe\_theory);

hold on;

semilogy(n,Pe\_practice);

legend('理论误码率','实际误码率');

title('误码率曲线');

xlabel('信噪比');

ylabel('误码率');

axis([-15 5 0.001 1]);

### **2.3实验结果**

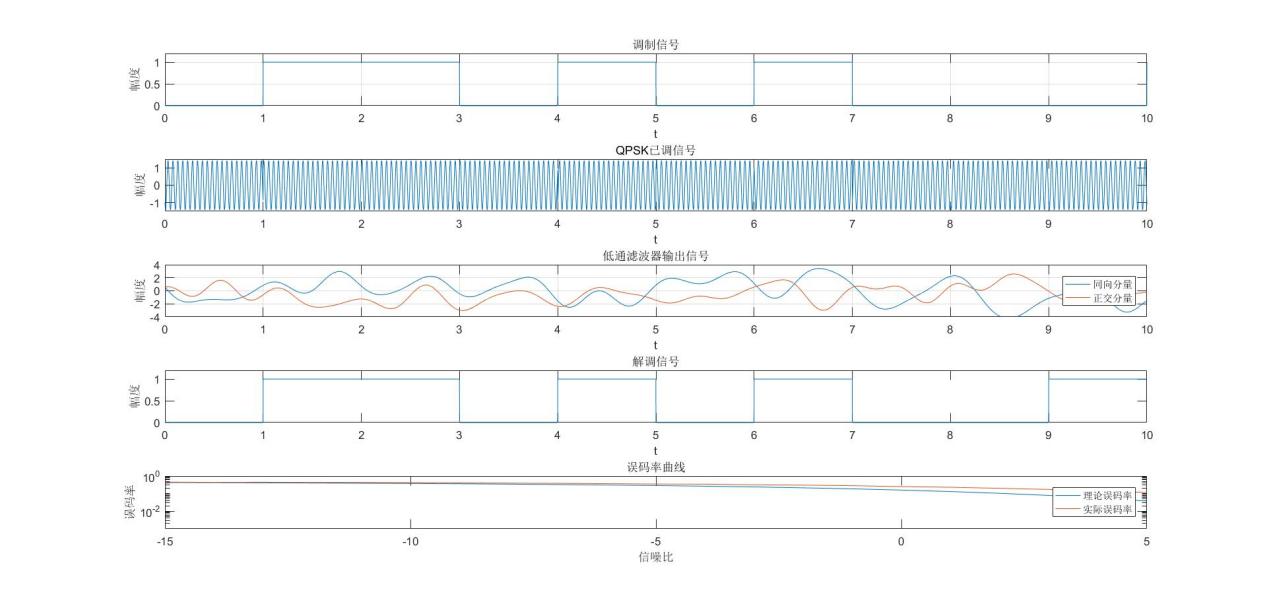


图 2 实验2结果图片

### **2.4实验分析**

本实验可以先将256bit二进制信源进行串并转换，乘上正交的载波生成已调信号，相加后变为1码元包含2bit的信号。相干解调时，由于cos的载波与sin的载波相乘并不会保留低频分量，而是全部搬移到高频，故可以通过低通滤波器过滤掉高频的分量，然后将低通滤波器输出的信号进行判决得到最后结果。

对于误码率，从码元为2bit来看，由于QPSK码间距离减小，故误码率也大于BPSK，从最终解码来看，相当于每比特传输用的功率为平均功率的一半，故误码率也大于BPSK，其误码率公式为。

## 实验3

### **3.1实验内容**

用MATLAB产生256bit 独立等概率的二进制信源，码元持续时间为1s，仿真信号在AWGN信道中的16QAM调制解调过程，其中载波幅值为1或2，频率为20Hz，判决门限为0。

（1）当解调器输入信噪比为0dB时，分别画出调制信号、16QAM星座图、16QAM已调信号、低通滤波器输出信号和解调信号；

（2）画出解调器输入信噪比在-15dB到5dB时的接收机理论误码率曲线与实际误码率曲线(20次以上平均)；

### **3.2实验程序**

clear;

N = 256; % 产生码元数

Ts = 1; % 码元持续时间

A = 1; % 载波幅值

fc = 20; % 载波频率

L = 1000; % 每个码元的采样点数

dt = Ts/L; % 采样间隔

T = N\*Ts; % 总时间

t = 0:dt:T-dt; % 时间

t\_ = 0:dt:T/4-dt;

% 生成调制信号

b = randi([0,1],1,N); % 二进制随机码

s = signal\_expand(b,L);

subplot(321)

plot(t,s);

title('调制信号');

xlabel('t');

ylabel('幅度');

axis([0,6,0,1.2]);

grid on;

% QAM

% 产生I、Q路码元

I = zeros(1,N/4);

Q = zeros(1,N/4);

for i = 4:4:N % 串并转换和2-4电平转换

    if b(i-3)==0 && b(i-1)==0

        I(round(i/4)) = -3;

    elseif b(i-3)==0 && b(i-1)==1

        I(round(i/4)) = -1;

    elseif b(i-3)==1 && b(i-1)==0

        I(round(i/4)) = 1;

    else

        I(round(i/4)) = 3;

    end

    if b(i-2)==0 && b(i)==0

        Q(round(i/4)) = -3;

    elseif b(i-2)==0 && b(i)==1

        Q(round(i/4)) = -1;

    elseif b(i-2)==1 && b(i)==0

        Q(round(i/4)) = 1;

    else

        Q(round(i/4)) = 3;

    end

end

I = signal\_expand(I,L);

Q = signal\_expand(Q,L);

c1 = A\*cos(2\*pi\*fc\*t\_); % 载波1

c2 = -A\*sin(2\*pi\*fc\*t\_); % 载波2

s\_QAM = I.\*c1+Q.\*c2;

subplot(323)

plot(t\_,s\_QAM);

title('QAM已调信号');

xlabel('t');

ylabel('幅度');

axis([0,6,-5,5]);

% 噪声

SNRdb = 0; % 信噪比

SNR = 10^(SNRdb/10);

B = 4; % 基带带宽

B\_bpf = 2\*(B+fc); % 理想带通滤波器带宽

P = (norm(s\_QAM(1:L)).^2)./length(s\_QAM(1:L)); % 平均功率

n0 = (P/SNR)/B\_bpf; % 噪声功率谱密度

u = sqrt(n0\*L\*fc/2)\*randn(1,L\*N/4); % 噪声

s\_QAM\_u = s\_QAM+u; % 经过信道的已调信号

% 滤波器

[f,S\_QAM\_u] = T2F(t\_,s\_QAM\_u); % 频域

S\_QAM\_u\_BPF = BPF(f,-(fc+B),fc+B,1).\*S\_QAM\_u; % 经过带通滤波器

[~,s\_QAM\_u\_BPF] = F2T(f,S\_QAM\_u\_BPF); % 时域

s1\_QAM\_u\_BPF\_c = s\_QAM\_u\_BPF.\*c1; % 与载波相乘

s2\_QAM\_u\_BPF\_c = s\_QAM\_u\_BPF.\*c2;

[~,S1\_QAM\_u\_BPF\_c] = T2F(t\_,s1\_QAM\_u\_BPF\_c); % 频域

[~,S2\_QAM\_u\_BPF\_c] = T2F(t\_,s2\_QAM\_u\_BPF\_c);

S1\_QAM\_u\_BPF\_c\_LPF = LPF(B,f,2).\*(S1\_QAM\_u\_BPF\_c); % 经过低通滤波器

S2\_QAM\_u\_BPF\_c\_LPF = LPF(B,f,2).\*(S2\_QAM\_u\_BPF\_c);

[~,s1\_QAM\_u\_BPF\_c\_LPF] = F2T(f,S1\_QAM\_u\_BPF\_c\_LPF); % 时域

[~,s2\_QAM\_u\_BPF\_c\_LPF] = F2T(f,S2\_QAM\_u\_BPF\_c\_LPF);

subplot(324)

plot(t\_,s1\_QAM\_u\_BPF\_c\_LPF);

hold on;

plot(t\_,s2\_QAM\_u\_BPF\_c\_LPF);

title('低通滤波器输出信号');

xlabel('t');

ylabel('幅度');

legend('同向分量','正交分量');

axis([0,6,-10,10]);

grid on;

% 解调

s1\_QAM = QAM\_SJ(s1\_QAM\_u\_BPF\_c\_LPF,L); % 抽样判决

s2\_QAM = QAM\_SJ(s2\_QAM\_u\_BPF\_c\_LPF,L);

s1\_QAM\_ = sample(s1\_QAM\_u\_BPF\_c\_LPF,L);

s2\_QAM\_ = sample(s2\_QAM\_u\_BPF\_c\_LPF,L);

% 16QAM星座点

constellation = [-(3+3i), -(3+1i), -(3-3i), -(3-1i), ...

                 -(1+3i), -(1+1i), -(1-3i), -(1-1i), ...

                 (3+3i), (3+1i), (3-3i), (3-1i), ...

                 (1+3i), (1+1i), (1-3i), (1-1i)];

% 绘制星座图

subplot(322)

scatter(real(constellation), imag(constellation), 'filled');

hold on;

scatter(s1\_QAM\_, s2\_QAM\_);

grid on;

axis([-7,7,-3.5,3.5]);

title('16QAM 星座图');

xlabel('I');

ylabel('R');

ax = gca;  % 获取当前坐标轴对象

ax.XAxisLocation = 'origin';  % 设置X轴显示在中心

ax.YAxisLocation = 'origin';  % 设置Y轴显示在中心

b = zeros(1,N); % 初始化

for i = 1:1:round(N/4)  % 串并转换

    if s1\_QAM(i) == -3

        b(4\*i-3) = 0;

        b(4\*i-1) = 0;

    elseif s1\_QAM(i) == -1

        b(4\*i-3) = 0;

        b(4\*i-1) = 1;

    elseif s1\_QAM(i) == 1

        b(4\*i-3) = 1;

        b(4\*i-1) = 0;

    else

        b(4\*i-3) = 1;

        b(4\*i-1) = 1;

    end

    if s2\_QAM(i) == -3

        b(4\*i-2) = 0;

        b(4\*i) = 0;

    elseif s2\_QAM(i) == -1

        b(4\*i-2) = 0;

        b(4\*i) = 1;

    elseif s2\_QAM(i) == 1

        b(4\*i-2) = 1;

        b(4\*i) = 0;

    else

        b(4\*i-2) = 1;

        b(4\*i) = 1;

    end

end

s = signal\_expand(b,L);

subplot(325)

plot(t,s);

title('解调信号');

xlabel('t');

ylabel('幅度');

axis([0,6,0,1.2]);

% 信噪比

SNR1 = -15; % 最小信噪比

SNR2 = 5; % 最大信噪比

T = 30; % 次数

n = SNR1:SNR2; % 信噪比横轴

Pe\_theory = 1-(1-(3\*(0.5\*erfc(sqrt(10.^(n/10))/sqrt(40)))-9/4\*(0.5\*erfc(sqrt(10.^(n/10))/sqrt(40))).^2)).^(1/4); % 理论误码率

Pe\_practice = zeros(1,length(n)); % 实际误码率

for SNR = SNR1:SNR2

    for i = 1:T

        b = randi([0,1],1,N); % 二进制随机码

        out = QAM(N,t\_,b,A,fc,L,B,SNR);  % 进行调制解调

        out = out - b;  % 信号差

        Pe\_practice(-SNR1+SNR+1) = Pe\_practice(-SNR1+SNR+1) + (sum(abs(out),'double')/N);  % 误码累计

    end

    Pe\_practice(-SNR1+SNR+1)=Pe\_practice(-SNR1+SNR+1)/T; % 平均误码率

end

subplot(326)

semilogy(n,Pe\_theory);

hold on;

semilogy(n,Pe\_practice);

legend('理论误码率','实际误码率');

title('误码率曲线');

xlabel('信噪比');

ylabel('误码率');

axis([-15 5 0.1 1]);

### **3.3实验结果**

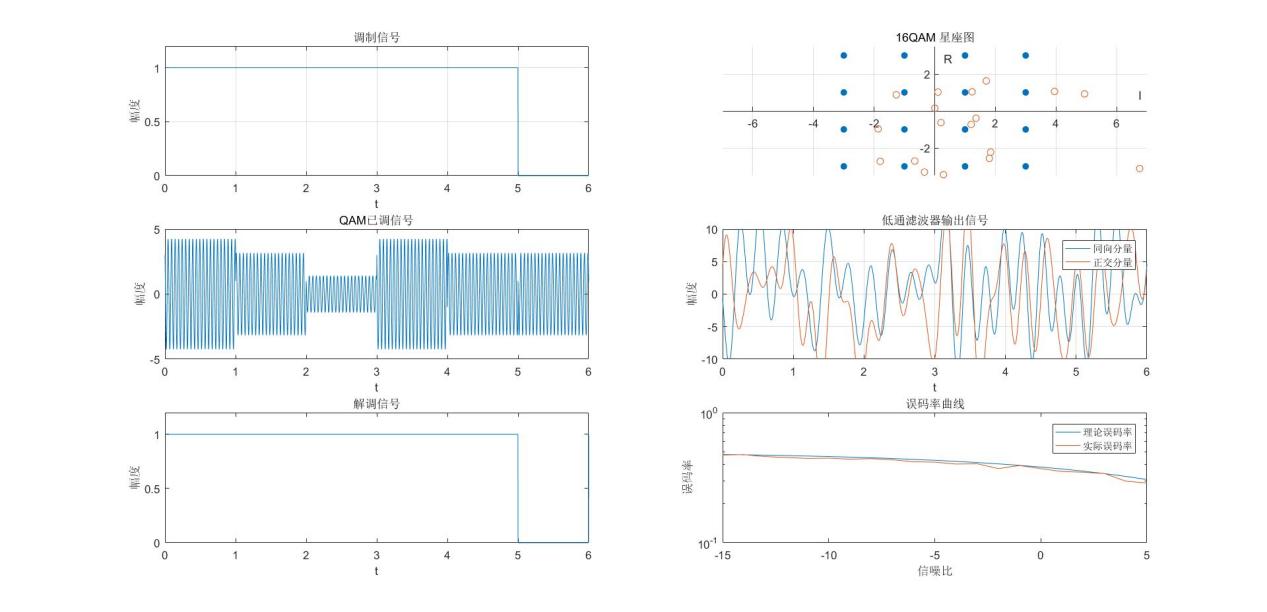


图 3 实验3结果图片

### **3.4实验分析**

首先将信号进行交并转换为I、Q两路信号，进行2-4电频转换后，变为映射为的信号，然后乘上相应的载波。经过信道后，首先进入带通滤波器过滤，然后乘上载波再经过低通滤波器得到两个正交信号，最后将低通滤波器输出输入到抽样判决器，根据抽样判别与逆映射可以还原原始信号，判决门限为-2，0，2。由于16QAM调制的噪声容限小于BPSK与QPSK，故在相同的信噪比下得到的解调结果误码率更高。

关于16QAM理论误码率的计算公式，参考https://zhuanlan.zhihu.com/p/266910750，得到了码元为4bit的16QAM误码率为，其中关于d的求解为：得出，再进一步化简，，由于本次实验最终解调后码元为1bit，所以根据（1减去4bit全部正确的概率即为误码率）和Q函数与互补误差函数之间的关系：即可计算得到解调后码元为1bit时的理论误码率公式，推导过程如下所示。

*附录：提供部分关键函数清单*

#### 信号扩展函数

function [out]=signal\_expand(s,L)

% 信号拓展

%*------------------------输入参数*

% s：原信号

% L：信号拓展后一个码元长度

%*---------------------输出(返回)参数*

% out：拓展后信号

N=length(s);             %基带信号码元长度

out=zeros(1,N\*L);

for i = 1 : N

    out((i-1)\*L+1:i\*L) = repmat(s(i), 1, L);

end

end

#### **BPSK抽样判决函数**

function out = BPSK\_SJ(s,L,d)

% 抽样判决

%------------------------输入参数

% s：原始信号

% L：判断点数

% d:判决门限

%---------------------输出(返回)参数

% out：输出信号

N = length(s)/L;

out = zeros(1,N);

for i = 1 : N

    if s((i-1)\*L+L/2)>d

        out((i-1)\*L+1:i\*L) = ones(1,L);

    else

        out((i-1)\*L+1:i\*L) = zeros(1,L);

    end

end

end

#### **BPSK调制解调函数**

function out = BPSK(t,s,A,fc,L,d,B,SNRdb)

% bpsk调制解调

%------------------------输入参数

% t：时间

% s：原始信号

% A：载波幅度

% fc：载波频率

% L：每个码元的采样点数

% d：判决门限

% B：信号带宽

% SNRdb：信噪比

%---------------------输出(返回)参数

% out：输出信号

%%  BPSK调制

c = A\*cos(2\*pi\*fc\*t);  % 载波

s\_BPSK = c.\*(s\*2-1);

%% 经过噪声

SNR = 10^(SNRdb/10);  % 信噪比

B\_bpf = 2\*fc+2\*B;  % 理想带通滤波器带宽

P = (norm(s\_BPSK(1:L)).^2)./length(s\_BPSK(1:L));  % 求解平均功率

n0 = P/SNR/B\_bpf;

u = sqrt(n0\*L\*fc/2)\*randn(1,length(s));  % 噪声

s\_BPSK\_u = s\_BPSK+u;  % 信号与加性噪声

%% 滤波器接收

[f,S\_BPSK\_u] = T2F(t,s\_BPSK\_u);   % 转换到频域

S\_BPSK\_u\_BPF = BPF(f,-(fc+B),fc+B,1).\*S\_BPSK\_u;  % 经过带通滤波器后

[t,s\_BPSK\_u\_BPF] = F2T(f,S\_BPSK\_u\_BPF);  % 变换为时域

s\_BPSK\_u\_BPF\_c = s\_BPSK\_u\_BPF.\*c ;  % 乘上载波

[f,S\_BPSK\_u\_BPF\_c] = T2F(t,s\_BPSK\_u\_BPF\_c) ;  % 变换到频域

S\_BPSK\_u\_BPF\_c\_LPF = LPF(B,f,1).\*(S\_BPSK\_u\_BPF\_c);  % 经过低通滤波器

%% 解调信号——LPF对输出信号进行抽样判决

[~,s] = F2T(f,S\_BPSK\_u\_BPF\_c\_LPF);  % 变换到时域

out = BPSK\_SJ(s,L,d);

end

#### **QPSK抽样判决函数**

function out = QPSK\_SJ(s,L,d)

% 抽样判决

%*------------------------输入参数*

% s：原始信号

% L：判断点数

% d:判决门限

%*---------------------输出(返回)参数*

% out：输出信号

N = length(s)/L;

out = zeros(1,N);

for i = 1 : N

    if s((i-1)\*L+L/2)>d

        out(i) = 1;

    else

        out(i) = 0;

    end

end

end

#### **QPSK调制解调函数**

function out = QPSK(N,t,s,A,fc,L,d,B,SNRdb)

% bpsk调制解调

%------------------------输入参数

% N：产生码元数

% t：时间

% s：原始信号

% A：载波幅度

% fc：载波频率

% L：每个码元的采样点数

% d：判决门限

% B：信号带宽

% SNRdb：信噪比

%---------------------输出(返回)参数

% out：输出信号

%% 产生调制信号

s1\_ = zeros(1,N/2); % \_0和\_1

s2\_ = zeros(1,N/2); % 0\_和1\_

for i = 2:2:N % 串并转换

    if s(i-1)==0

        s1\_(round(i/2)) = -1;

    else

        s1\_(round(i/2)) = 1;

    end

    if s(i)==0

        s2\_(round(i/2)) = -1;

    else

        s2\_(round(i/2)) = 1;

    end

end

s1 = signal\_expand(s1\_,L);

s2 = signal\_expand(s2\_,L);

%% QPSK调制

c1 = A\*cos(2\*pi\*fc\*t);  % 载波

c2 = -A\*sin(2\*pi\*fc\*t);

s\_QPSK = c1.\*s1+c2.\*s2;

%% 经过噪声

SNR = 10^(SNRdb/10);  % 信噪比

B\_bpf = 2\*fc+2\*B;  % 理想带通滤波器带宽

P = (norm(s\_QPSK(1:L)).^2)./length(s\_QPSK(1:L));  % 求解平均功率

n0 = P/SNR/B\_bpf;

u = sqrt(n0\*L\*fc/2)\*randn(1,L\*N/2);  % 噪声

s\_QPSK\_u = s\_QPSK+u;  % 信号与加性噪声

%% 滤波器接收

[f,S\_QPSK\_u] = T2F(t,s\_QPSK\_u); % 频域

S\_QPSK\_u\_BPF = BPF(f,-(fc+B),fc+B,1).\*S\_QPSK\_u; % 经过带通滤波器

[~,s\_QPSK\_u\_BPF] = F2T(f,S\_QPSK\_u\_BPF); % 时域

s1\_QPSK\_u\_BPF\_c = s\_QPSK\_u\_BPF.\*c1; % 与载波相乘

s2\_QPSK\_u\_BPF\_c = s\_QPSK\_u\_BPF.\*c2;

[~,S1\_QPSK\_u\_BPF\_c] = T2F(t,s1\_QPSK\_u\_BPF\_c); % 频域

[~,S2\_QPSK\_u\_BPF\_c] = T2F(t,s2\_QPSK\_u\_BPF\_c);

S1\_QPSK\_u\_BPF\_c\_LPF = LPF(B,f,2).\*(S1\_QPSK\_u\_BPF\_c); % 经过低通滤波器

S2\_QPSK\_u\_BPF\_c\_LPF = LPF(B,f,2).\*(S2\_QPSK\_u\_BPF\_c);

[~,s1\_QPSK\_u\_BPF\_c\_LPF] = F2T(f,S1\_QPSK\_u\_BPF\_c\_LPF); % 时域

[~,s2\_QPSK\_u\_BPF\_c\_LPF] = F2T(f,S2\_QPSK\_u\_BPF\_c\_LPF);

%% 解调信号——LPF对输出信号进行抽样判决

s1\_QPSK = QPSK\_SJ(s1\_QPSK\_u\_BPF\_c\_LPF,L,d); % 抽样判决

s2\_QPSK = QPSK\_SJ(s2\_QPSK\_u\_BPF\_c\_LPF,L,d);

out = zeros(1,N); % 初始化

for i = 1:1:round(N/2) % 串并转换

    out(2\*i-1:2\*i) = [s1\_QPSK(i),s2\_QPSK(i)];

end

end

#### 16QAM抽样判决函数

function out = QAM\_SJ(s,L)

% 抽样判决

%*------------------------输入参数*

% s：原始信号

% L：判断点数

%*---------------------输出(返回)参数*

% out：输出信号

N = length(s)/L;

out = zeros(1,N);

d1 = -2;

d2 = 0;

d3 = 2;

for i = 1 : N

    if s((i-1)\*L+L/2)<d1

        out(i) = -3;

    elseif s((i-1)\*L+L/2)>=d1 && s((i-1)\*L+L/2)<d2

        out(i) = -1;

    elseif s((i-1)\*L+L/2)>=d2 && s((i-1)\*L+L/2)<d3

        out(i) = 1;

    elseif s((i-1)\*L+L/2)>=d3

        out(i) = 3;

    end

end

end

#### **16QAM调制解调函数**

function out = QAM(N,t,b,A,fc,L,B,SNRdb)

% bpsk调制解调

%------------------------输入参数

% N：产生码元数

% t：时间

% b：原始信号

% A：载波幅度

% fc：载波频率

% L：每个码元的采样点数

% B：信号带宽

% SNRdb：信噪比

%% 基带信号与载波

I = zeros(1,N/4);

Q = zeros(1,N/4);

for i = 4:4:N  % 串并转换 和 2-4电平转换

    if b(i-3)==0 && b(i-1)==0

        I(round(i/4)) = -3;

    elseif b(i-3)==0 && b(i-1)==1

        I(round(i/4)) = -1;

    elseif b(i-3)==1 && b(i-1)==0

        I(round(i/4)) = 1;

    else

        I(round(i/4)) = 3;

    end

    if b(i-2)==0 && b(i)==0

        Q(round(i/4)) = -3;

    elseif b(i-2)==0 && b(i)==1

        Q(round(i/4)) = -1;

    elseif b(i-2)==1 && b(i)==0

        Q(round(i/4)) = 1;

    else

        Q(round(i/4)) = 3;

    end

end

I = signal\_expand(I,L);

Q = signal\_expand(Q,L);

%% QAM调制

c1 = A\*cos(2\*pi\*fc\*t);  % 载波1

c2 = -A\*sin(2\*pi\*fc\*t);  % 载波2

s\_QAM = I.\*c1+Q.\*c2;

%% 噪声

SNR = 10^(SNRdb/10);

B\_bpf = 2\*(B+fc); % 理想带通滤波器带宽

P = (norm(s\_QAM(1:L)).^2)./length(s\_QAM(1:L)); % 平均功率

n0 = (P/SNR)/B\_bpf; % 噪声功率谱密度

u = sqrt(n0\*L\*fc/2)\*randn(1,L\*N/4); % 噪声

s\_QAM\_u = s\_QAM+u; % 经过信道的已调信号

%% 滤波器接收

[f,S\_QAM\_u] = T2F(t,s\_QAM\_u); % 频域

S\_QAM\_u\_BPF = BPF(f,-(fc+B),fc+B,1).\*S\_QAM\_u; % 经过带通滤波器

[~,s\_QAM\_u\_BPF] = F2T(f,S\_QAM\_u\_BPF); % 时域

s1\_QAM\_u\_BPF\_c = s\_QAM\_u\_BPF.\*c1; % 与载波相乘

s2\_QAM\_u\_BPF\_c = s\_QAM\_u\_BPF.\*c2;

[~,S1\_QAM\_u\_BPF\_c] = T2F(t,s1\_QAM\_u\_BPF\_c); % 频域

[~,S2\_QAM\_u\_BPF\_c] = T2F(t,s2\_QAM\_u\_BPF\_c);

S1\_QAM\_u\_BPF\_c\_LPF = LPF(B,f,2).\*(S1\_QAM\_u\_BPF\_c); % 经过低通滤波器

S2\_QAM\_u\_BPF\_c\_LPF = LPF(B,f,2).\*(S2\_QAM\_u\_BPF\_c);

[~,s1\_QAM\_u\_BPF\_c\_LPF] = F2T(f,S1\_QAM\_u\_BPF\_c\_LPF); % 时域

[~,s2\_QAM\_u\_BPF\_c\_LPF] = F2T(f,S2\_QAM\_u\_BPF\_c\_LPF);

%% 解调

s1\_QAM = QAM\_SJ(s1\_QAM\_u\_BPF\_c\_LPF,L); % 抽样判决

s2\_QAM = QAM\_SJ(s2\_QAM\_u\_BPF\_c\_LPF,L);

out = zeros(1,N);  % 初始化

for i = 1:1:round(N/4)  % 串并转换

    if s1\_QAM(i) == -3

        out(4\*i-3) = 0;

        out(4\*i-1) = 0;

    elseif s1\_QAM(i) == -1

        out(4\*i-3) = 0;

        out(4\*i-1) = 1;

    elseif s1\_QAM(i) == 1

        out(4\*i-3) = 1;

        out(4\*i-1) = 0;

    else

        out(4\*i-3) = 1;

        out(4\*i-1) = 1;

    end

    if s2\_QAM(i) == -3

        out(4\*i-2) = 0;

        out(4\*i) = 0;

    elseif s2\_QAM(i) == -1

        out(4\*i-2) = 0;

        out(4\*i) = 1;

    elseif s2\_QAM(i) == 1

        out(4\*i-2) = 1;

        out(4\*i) = 0;

    else

        out(4\*i-2) = 1;

        out(4\*i) = 1;

    end

end

end