编码引论 第一次编程练习

小组成员：杨扬，何昊天，胡钰彬

报告人：胡钰彬

1. 总述

在本次编程练习中，我们三人共完成了如下内容：

- 针对题目中给出的两种信道情况，完成了信道的仿真。完成人：胡钰彬
- 对固定相位的信道情况，针对二、四、八电平完成了先发送一段序列来确定ϕ，之后再进行传输和判决的方法。针对二、四电平设计并完成了PSK+直流偏置的映射和判决方法，针对八电平设计并完成了MQAM去掉一个角点的映射和判决方法以及一种8QAM。映射均采用格雷映射。完成人：胡钰彬
- 对不固定相位的信道情况，对于接收电平的概率分布做了推导，基于最大化正确判决概率，完成了ASK的电平映射与判决方法设计。映射均采用格雷映射。完成人：杨扬
- 针对两种信道情况和相应的映射方式，完成了误比特率和信道信噪比关系的统计，画出了收端和发端的星座图。完成人：胡钰彬，杨扬
- 按照题目中给出的参数，完成了1/2效率和1/3效率卷积码编码器的设计，可选择收尾或是不收尾。完成人：胡钰彬
- 完成了1/2效率和1/3效率维特比译码器的编写，完成了硬判决译码和针对各种映射方式的软判决译码。添加译码器之后，统计了误比特率和信噪比的关系，并给出了典型误码图案。完成人：何昊天
- 使用CRC进行了冗余校验，统计了CRC汇报误块率和信噪比的关系，并比较了CRC汇报误块率和实际误块率之间的关系。完成人：胡钰彬
- 针对1200比特的实际传输任务，对不同的信道允许使用次数设计了不同的传输方案，并绘出不同方案下误比特率和信噪比的关系。完成人：何昊天

文件目录说明：

- 文件目录中的文件，以test开头的是在测试映射的参数，以plot开头的是在绘制各个比特映射下BER-SNR曲线，以modulate开头的是映射与调制函数，以judge为开头的是判决函数。其余脚本和函数功能基本和命名相符。

2. 信道仿真

- 信道仿真部分，主要完成了channel.m函数的编写。函数接收的参数如下：进入信道前的符号序列，信道的模式（1或2），信道噪声的标准差。
- 在函数中，利用random('unif') 随机出均匀分布的偏转角。固定相位时，随机出一个相位并将其重复；不固定相位时，随机出和符号序列等长的相位序列，将相位噪声加到原符号上之后，分别对实部和虚部产生噪声序列，合成之后叠加到原信号上
- 信道函数的代码见附录7.1：信道代码

3. 映射和判决方案

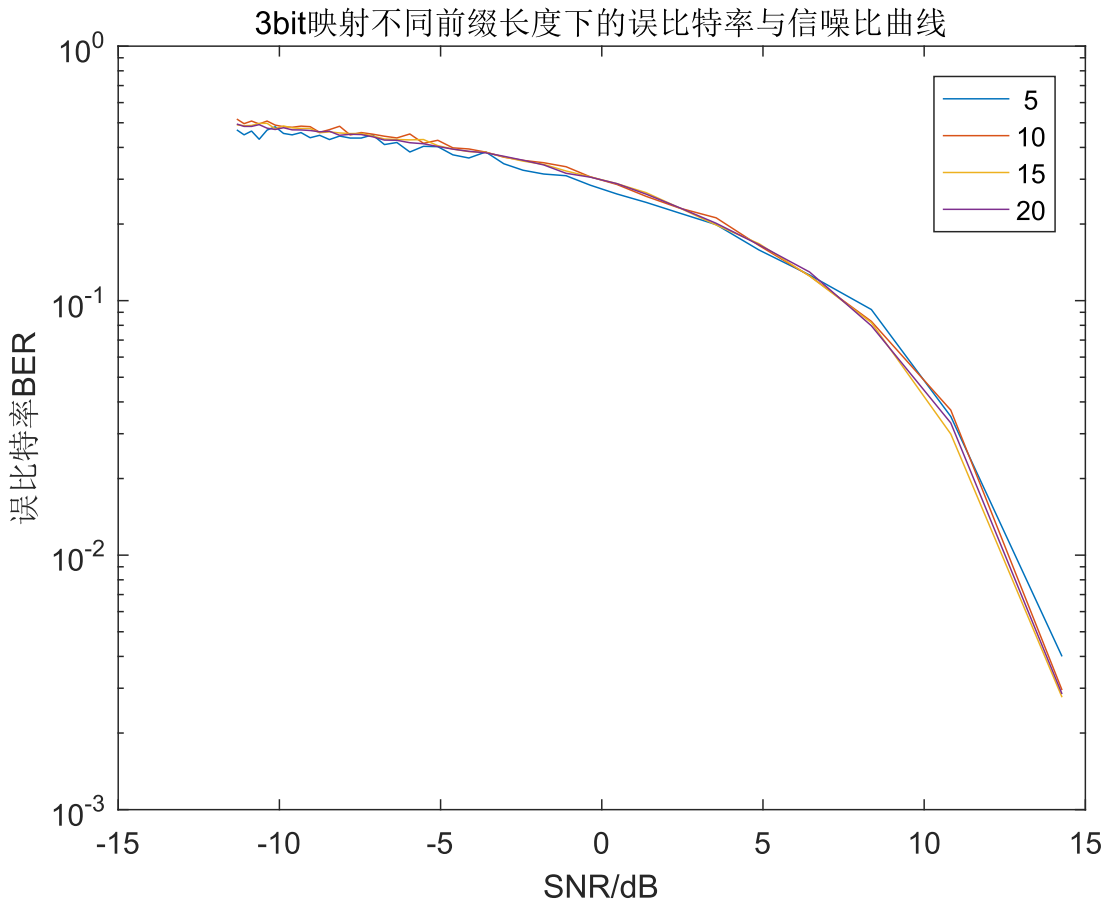
- 所有方案的设计都基于信源发送各符号的行为是等概的这一假设。

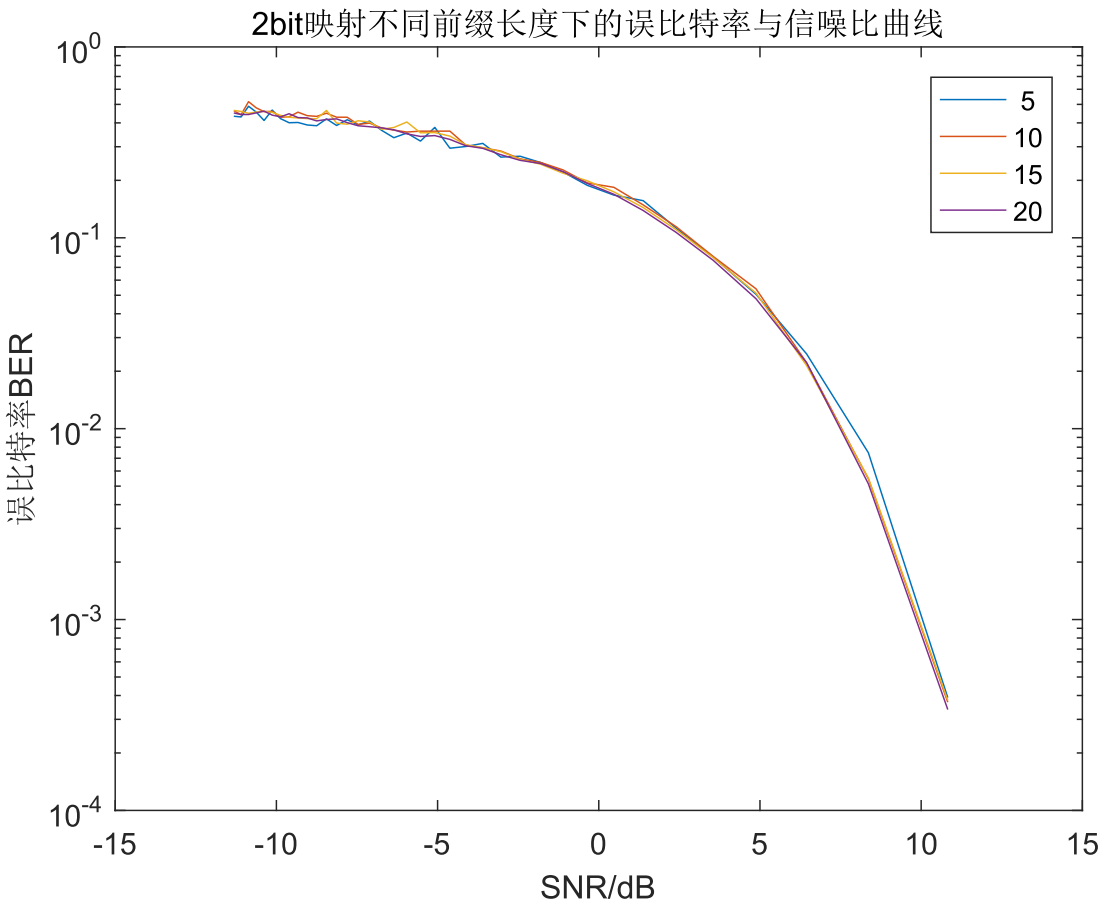
3.1 不固定相位下的映射判决方案-ASK

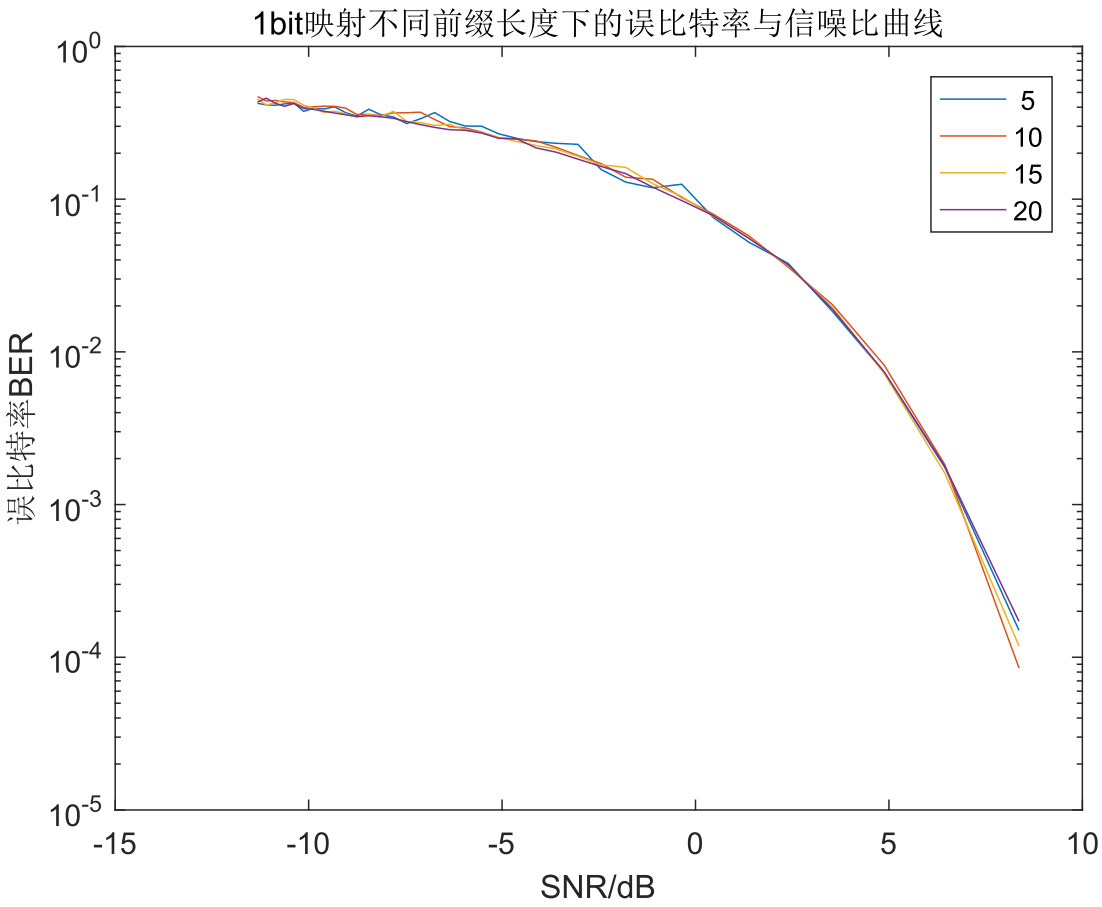
3.2 固定相位下的映射判决方案

3.2.1 sequence+PSK（s-PSK）

- sequence+PSK 的核心思路是PSK，即用相位进行调制，但是考虑到信道的固定相位噪声未知，我们决定先发送一段序列，用这些序列取平均来估计信道在相位上的噪声。
- 估计出来之后，将收到的复电平序列反向旋转一定角度，恢复出发端发射的复电平，之后再根据相位进行判决。（这一方法是在展示之前想出的）
- sequence 长度的确定
    - 我们选取四种长度：5，10，15，20。对每种长度用10000比特左右的序列进行仿真，对每种长度仿真10次，得到误比特率与信噪比的曲线如下：

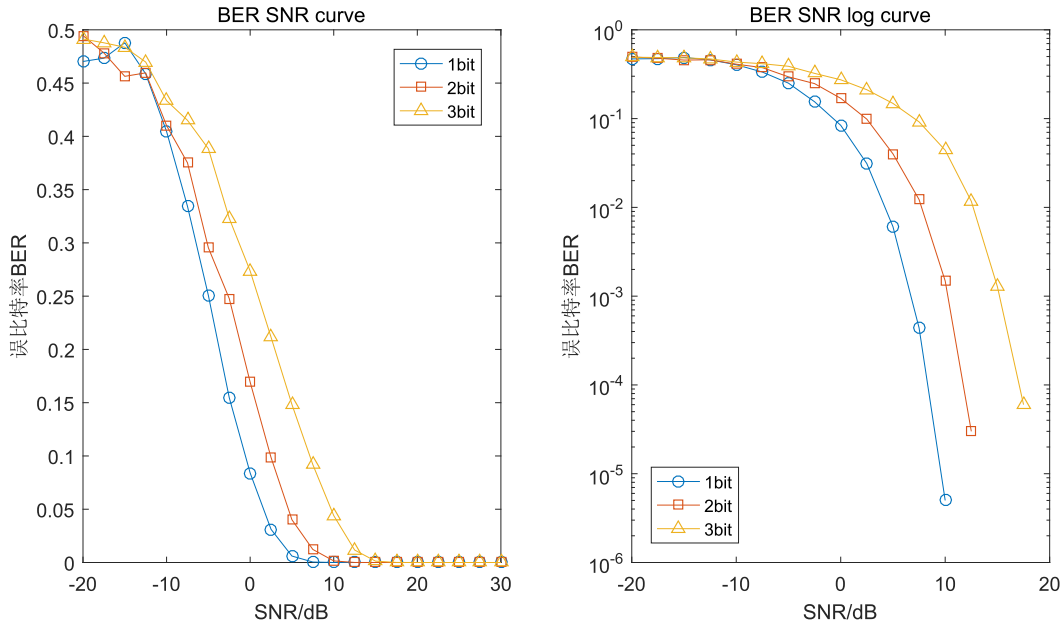



2bit映射不同前缀长度下的误比特率与信噪比曲线
误比特率BER
SNR/dB
5
10
15
20


1bit映射不同前缀长度下的误比特率与信噪比曲线
误比特率BER
SNR/dB
5
10
15
20

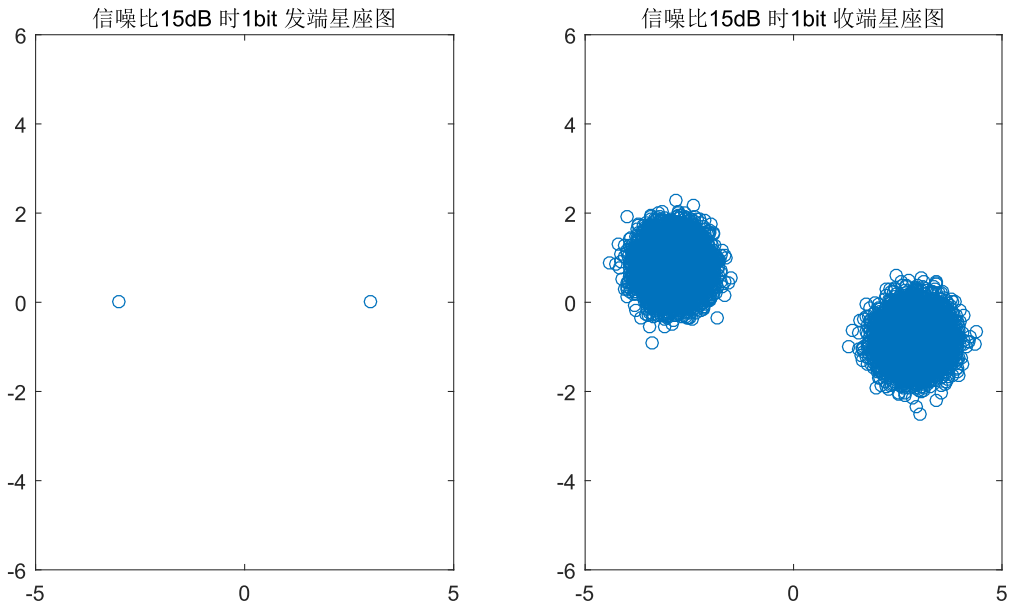
- 根据以上仿真结果，我们发现从对数坐标来看，不同的前缀长度对曲线的影响不是很大，但若长度太小（如5个符号），则不能很好地对相移进行估计，导致曲线发生抖动。考虑到这些因素，我们决定在序列前加上长度为10的前缀序列。

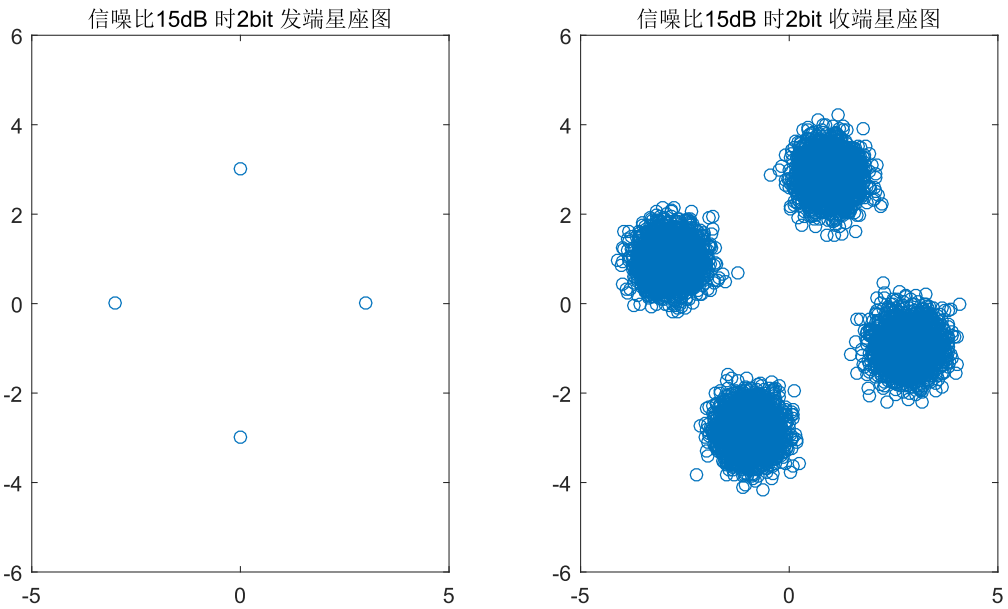
- 硬判决误比特率与复电平信道信噪比的关系

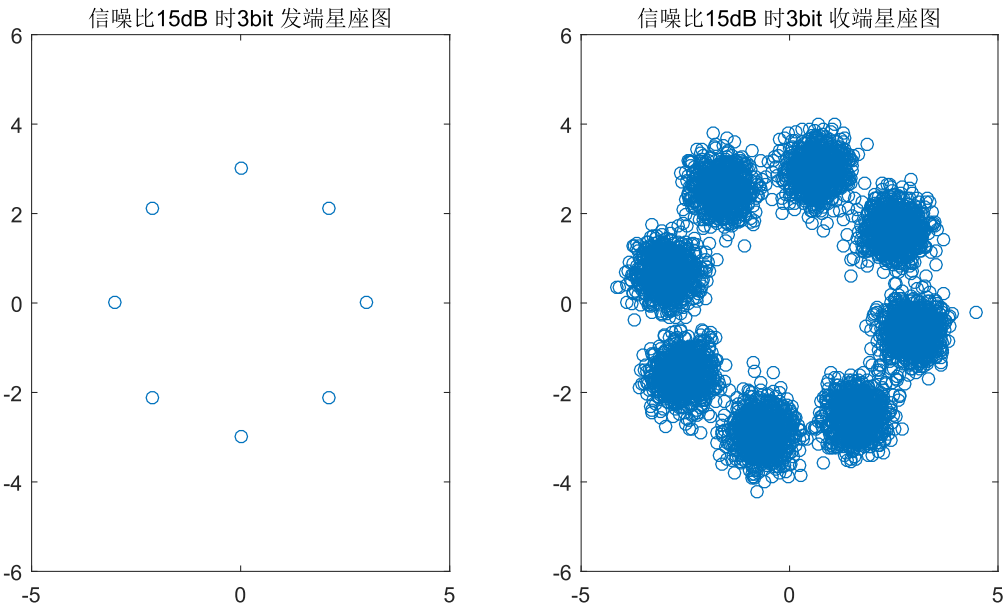


  - 上方左图为线性坐标，右图为对数坐标。
  - 总的来说，误比特率随着信噪比的提升而下降，这很容易理解。
  - 另一方面，在相同的信噪比下，随着电平映射比特数的增加，误比特率逐渐提升，这是因为采取了PSK调制的缘故。在信噪比固定的情况下，映射比特数越多，不同符号的调制电平在相位上会更难区分，使得判决错误概率增大。

- 发端星座图与收端星座图



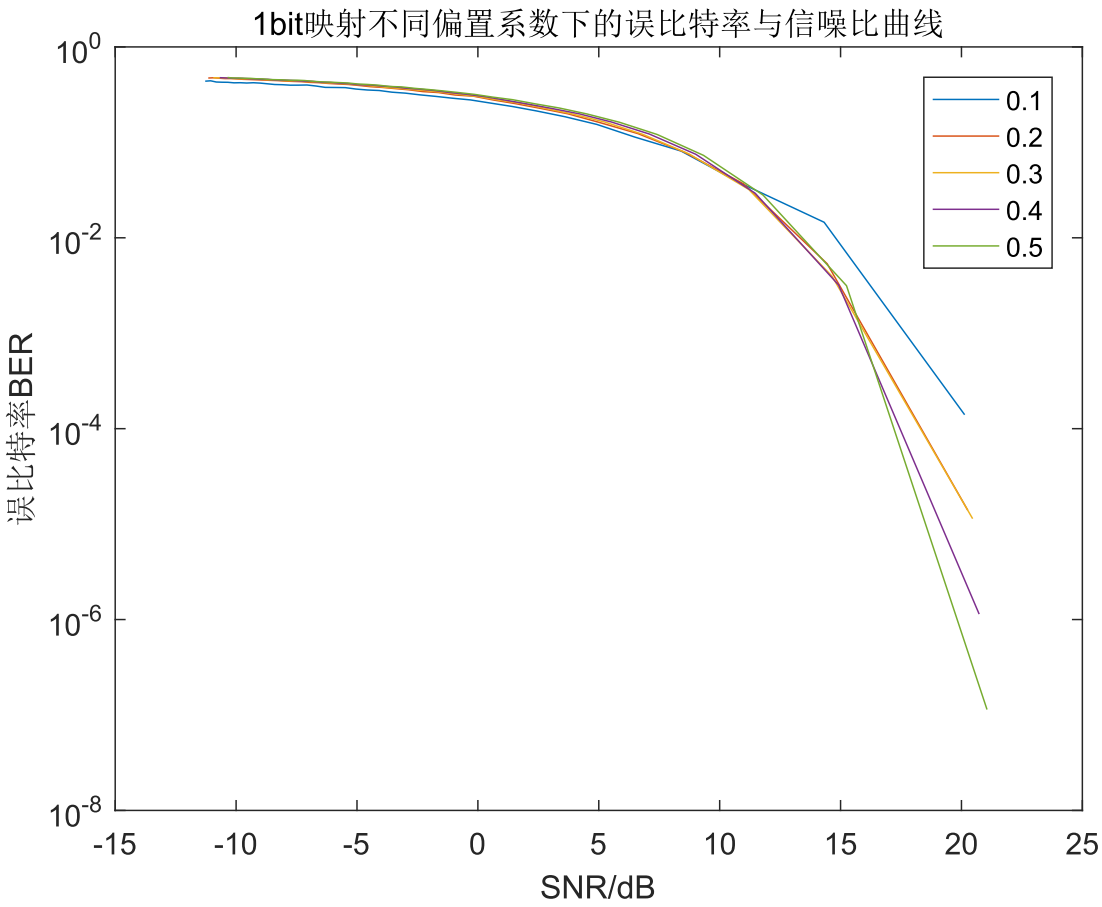


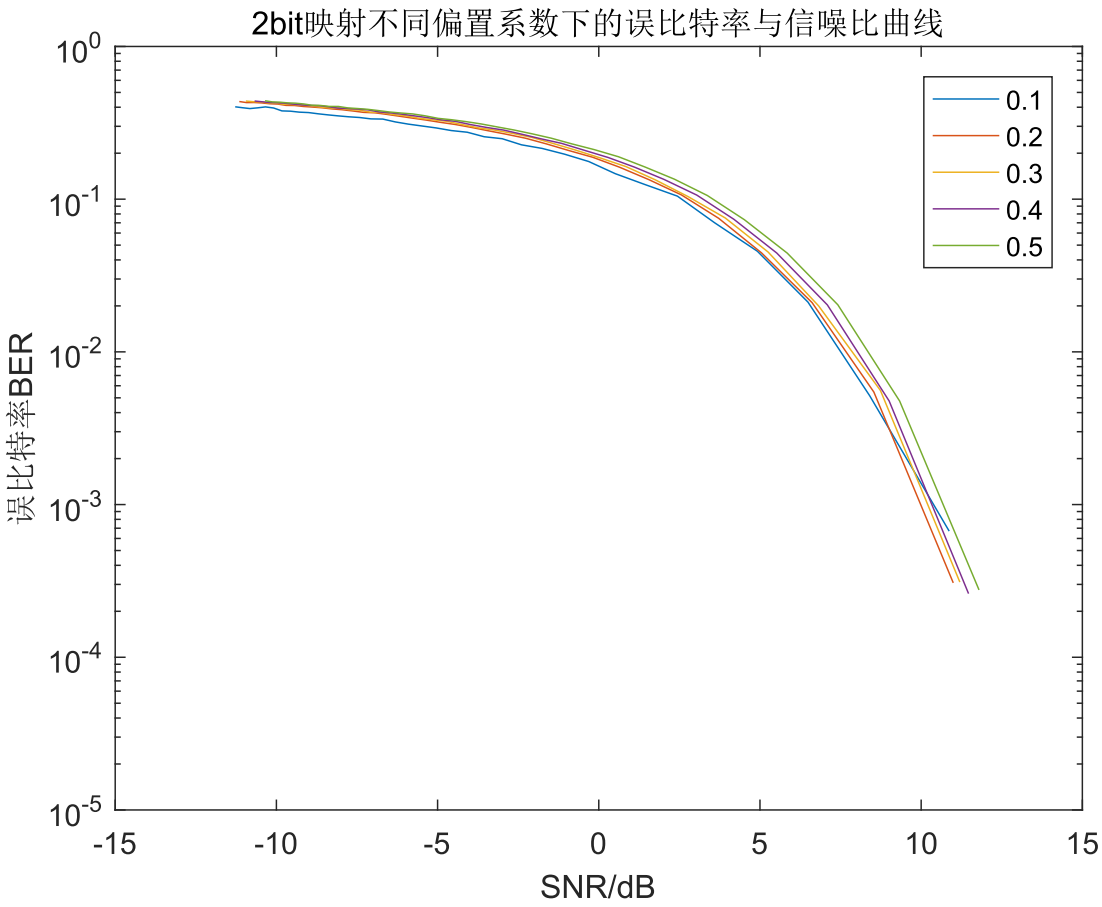


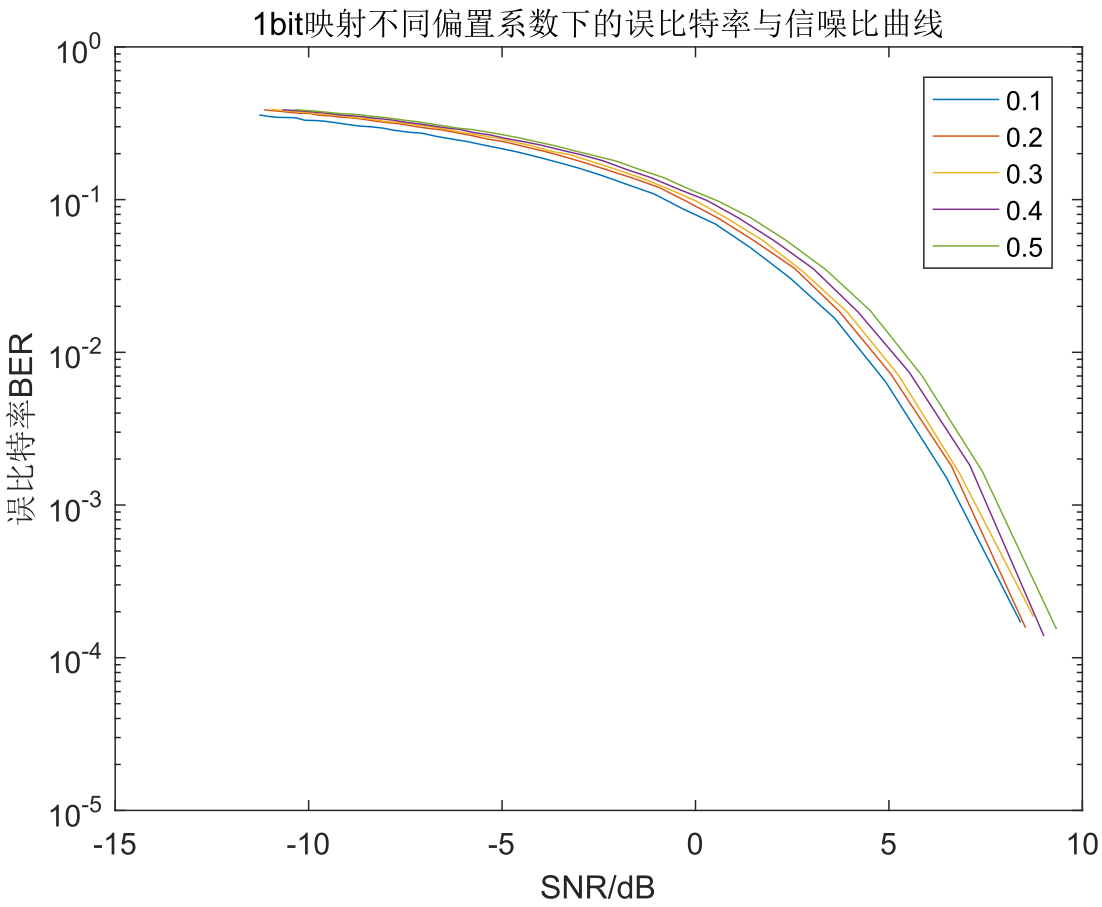
- 发端的星座图就是符号的的复电平，收端的星座图就是发送复电平加上复高斯噪声和固定相位噪声的结果，可以看出，发送电平在加入加性复高斯噪声之后整体旋转了一定角度。

3.2.2 Bias-PSK

- Bias-PSK是借鉴了徐泽来小组的方法，我们发现这种方法可以很好地应用于多电平调制的场景，因此决定尝试自己实现一下Bias-PSK。我们也为发送复电平加上一个实数偏置，将该偏置与PSK调制的模厂之比记作类似于前文中寻找前缀序列的合适长度，我们希望通过实际仿真观测Bias-ratio对误比特率的影响，从而确定一个比较合适的Bias-ratio。
- Bias ratio 的确定
  - 我们选取bias ratio 为0.1，0.2，0.3，0.4，0.5。对每个bias ratio用10000比特左右的序列进行仿真，每个bias ratio仿真10次对误比特率取平均，得到如下误比特率与信噪比的关系（图一标题标错了，应为3bit映射）

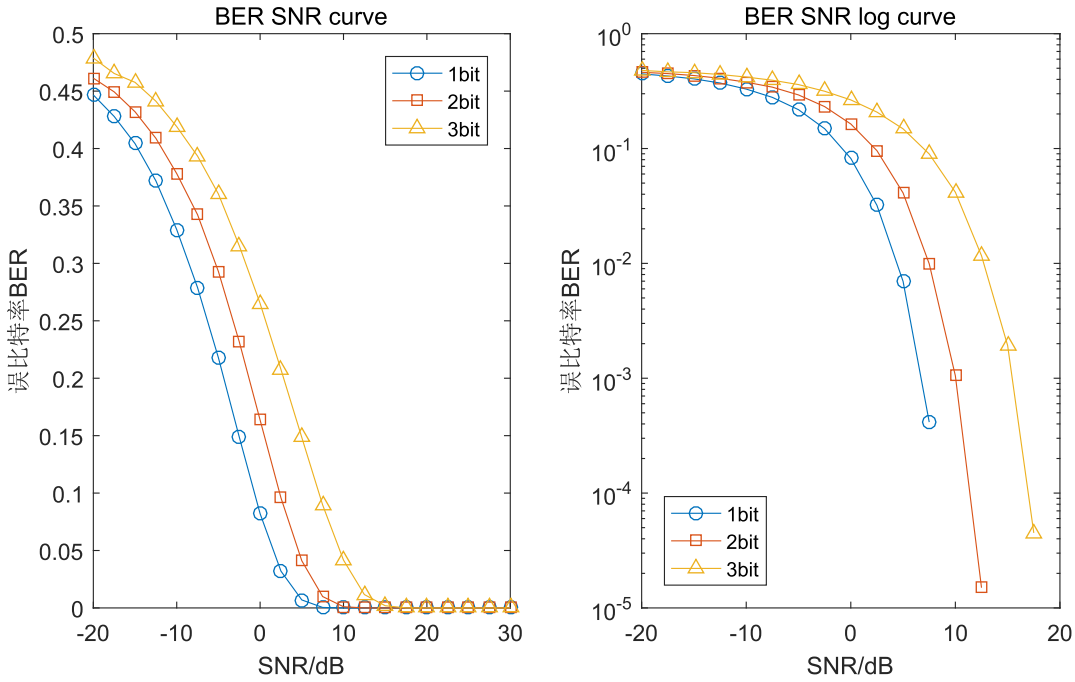

1bit映射不同偏置系数下的误比特率与信噪比曲线
误比特率BER
SNR/dB
0.1
0.2
0.3
0.4
0.5


2bit映射不同偏置系数下的误比特率与信噪比曲线
误比特率BER
SNR/dB
0.1
0.2
0.3
0.4
0.5

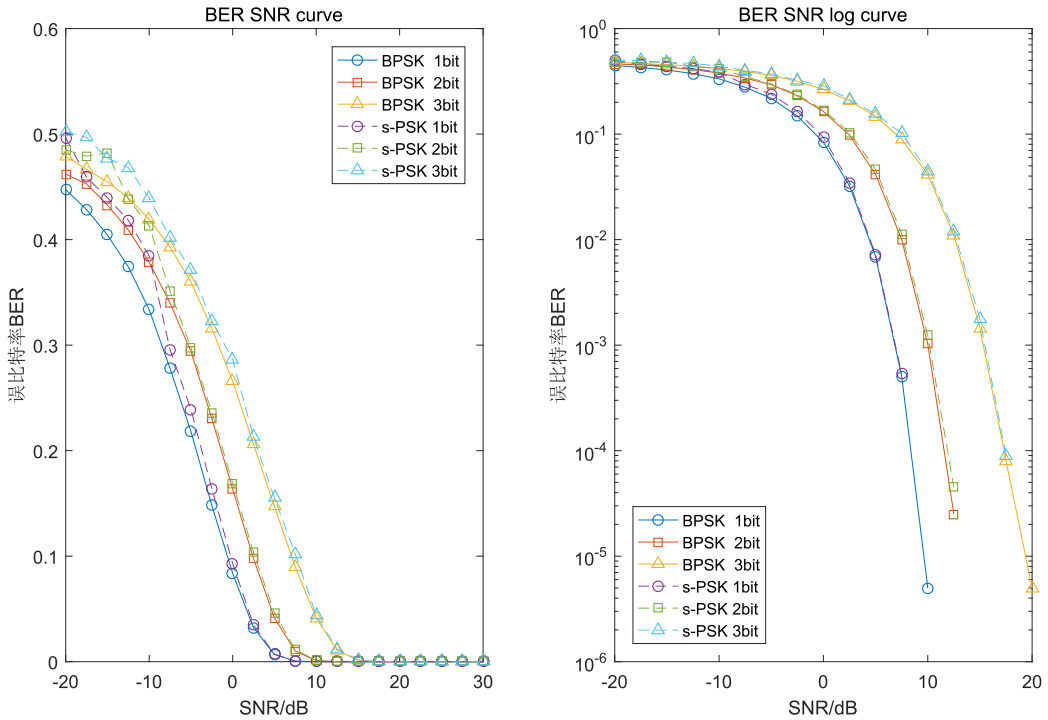


- 通过绘制不同bias-ratio下的误比特率-信噪比曲线，可以发现在1bit映射时，bias-ratio越大，曲线越靠右上方，与直观感受相反，我认为这是由于1bit映射时，只有两个互为相反数的调制电平，分辨度较高，对固定相位噪声的估计误差不会对判决产生较大的影响。相反，bias-ratio越大，会导致信号功率增大，在信噪比固定时，就会导致两个复电平的分辨度下降，误比特率增加。
- 而对于2bit映射和3bit映射，复电平之间的夹角逐渐减小，导致各符号之间的分辨度降低，对固定相位噪声的估计误差更为敏感，而bias-ratio越大，固定相位噪声受加性复高斯噪声的影响更小，我们对固定相位噪声的估计越精确，就可以在相同信噪比的情况下得到更低的误比特率。
- 综合上述考虑，我们决定将0.2作为bias-ratio，因为它能够兼顾较多和较少的比特映射时的不同情况。

- 硬判决误比特率与复电平信道信噪比的关系



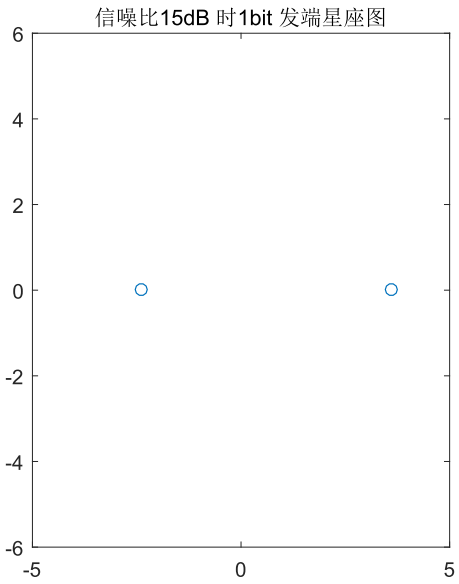
- 可以看到，整体曲线趋势同 s-PSK 映射方式的误比特率-信噪比曲线趋势十分相似，为了更好地对比这两种方法，我们将这两种映射方式得到的曲线放到一张图中进行比较：

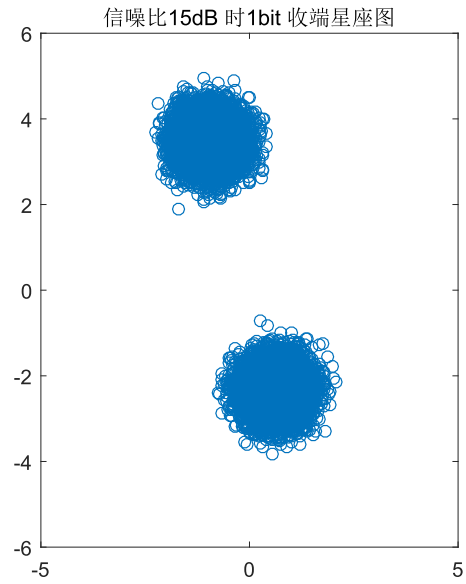


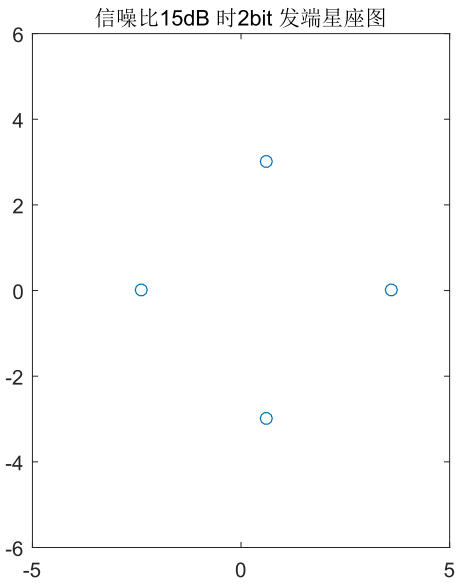
- 从这幅对比图中，我们可以看出s-PSK的曲线均稍稍靠右上方一点，这是因为在s-PSK中，我们只用了10位的前缀序列，对固定相位噪声的估计不太准确，使得整体误比特率更高。而从BPSK来看，我们采用的bias-ratio只有0.2，对信号功率的影响不是很大，也就是使用较少功率上的牺牲得到了较高误比特率上的提升。
- 经过这些分析，我们认为这幅图反映出来的优劣关系很大程度上取决于前缀序列长度和bias-ratio这两个参数的选取，不具有一般性。
- 同样我们可以预见，如果前缀序列更长，那么整体的误比特率会降低。而如果在一定幅度内提升bias-ratio，则会让1bit和2bit映射的效果变差，3bit映射的效果变好。在本次实验中我们采用0.2作为统一的

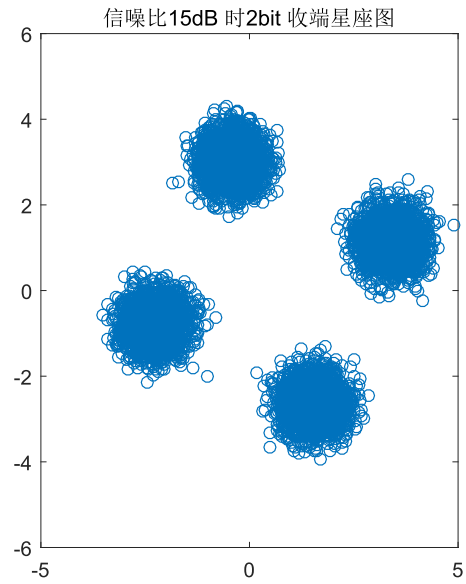
bias-ratio，实际情况中可以根据具体的映射情况进行优化。

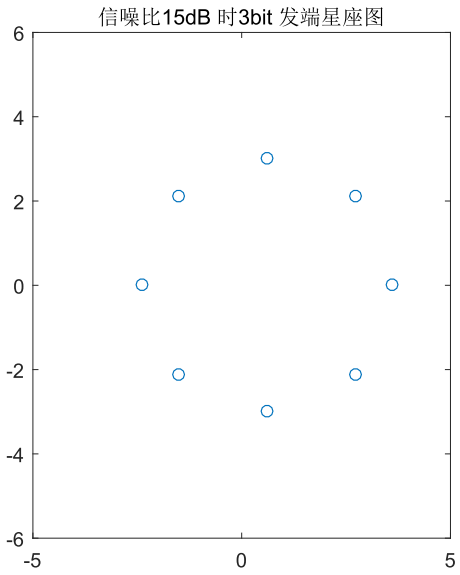
- 发端星座图与收端星座图

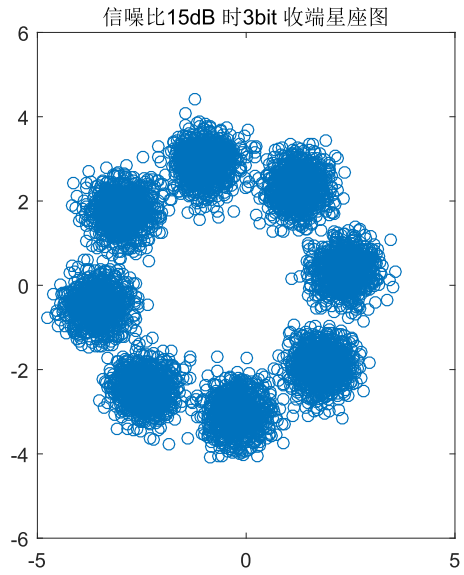








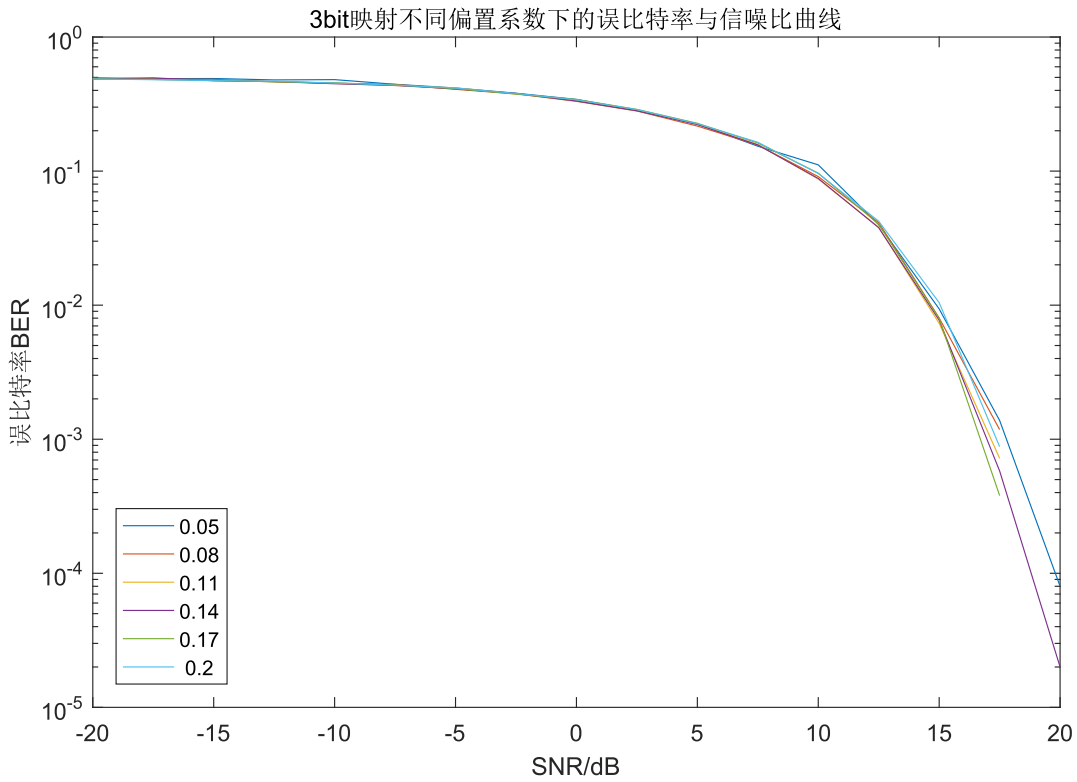




- 发端的星座图就是符号的的复电平加上偏置之后得到的复电平，收端的星座图就是发送复电平加上复高斯噪声和固定相位噪声的结果，可以看出，接收星座图的平均值向量同实轴正方向有一定夹角，这个夹角就是我们对固定相位噪声的估计。

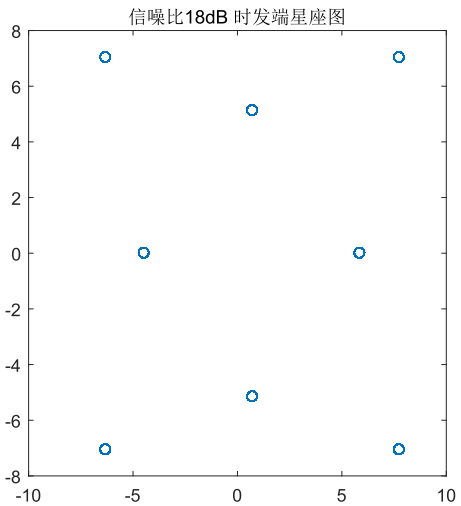
3.2.3 bias-8QAM

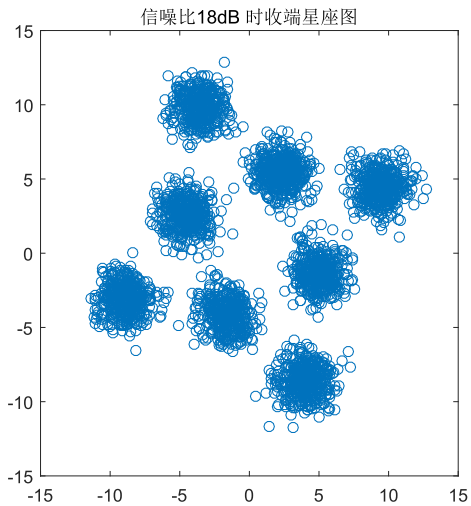
- 经过查找资料，我们找到了一种对3bit映射有效的QAM映射方法，这种QAM由两个幅度不同的2bitQAM结合而成，而且这两个QAM之间的相位相差$\pi/2$， 之所以希望选择QAM，是因为我们决定QAM对空间的利用率更高，会在信噪比误比特率曲线中有更好的表现。
- 由于这种8QAM是中心对称的，经过固定相位噪声的信道之后会损失掉相位信息，因此我们决定对这样的QAM加上偏置，同样用bias-ratio来刻画偏置的大小，这里的bias-ratio表示偏置量和发射复电平的最大模长之比。我们仍然希望找到一个较为合适bias-ratio使得信噪比误比特率曲线的表现最好，对于不同的bias-ratio用10000比特左右的序列进行仿真，每个bias ratio仿真10次对误比特率取平均，得到如下误比特率与信噪比的关系



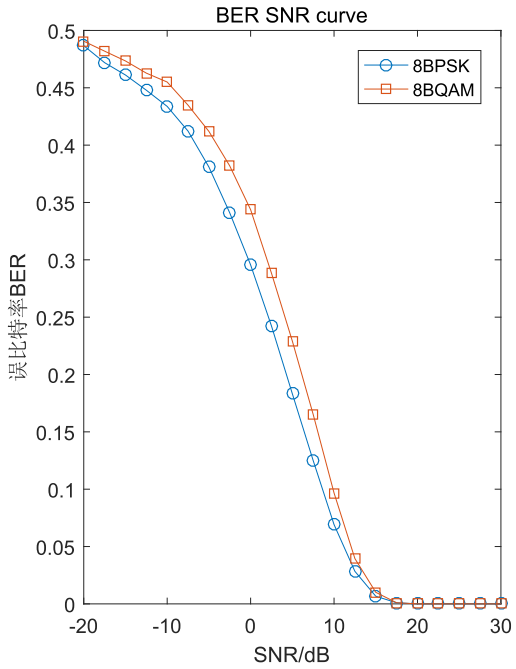
- 从这幅图像中，我们可以看出，随着bias ratio的上升，曲线并非一味地往下压，而是在下压到一定值时之后开始上抬，具体在图中，bias-ratio取0.05和0.2时曲线都在中间，在最下方的是bias-ratio取0.14时的图像。
- 这验证了我们在BPSK中对bias-ratio的影响的推测，即该比值对于3bit映射存在一个相对较好的取值。

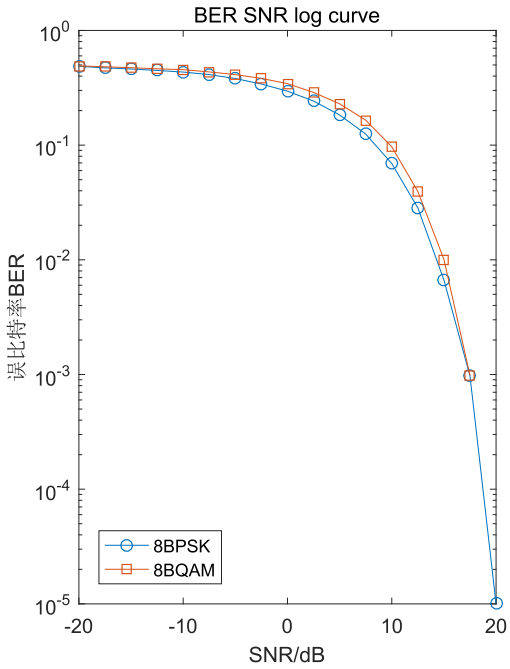
- 发端星座图和收端星座图如下：





- 为了和BPSK进行对比，我们将两个bias-ratio都取成0.2，绘制出这两种映射方法的误比特率-信噪比曲线如下：



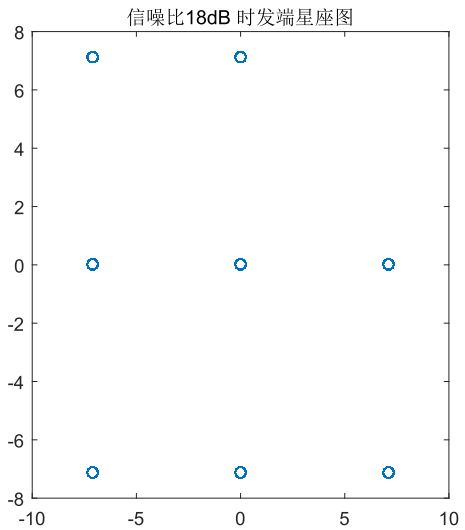


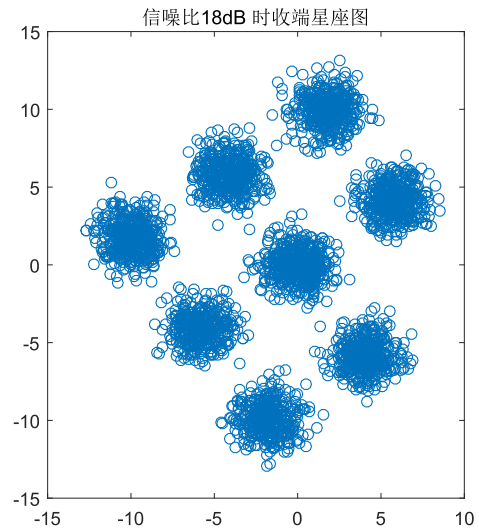
- 从上图中可以发现，在bias-ratio均取0.2的情况下，BPSK的效果要优于8BQAM。
- 从bias-ratio的角度看，在bias-ratio取相同值的时候，8QAM的平均信源功率应该小于PSK，这种感觉与结果显示相反，无法对结果进行显示。
- 从复电平的分布来看，8QAM的劣势在于它内圈的四个电平靠得比较近，这虽然带来了平均功率的下降，但是在加入噪声之后对误判概率的影响同样较大，因此在bias-ratio相同时，PSK的性能更好。

3.2.4 Partial QAM for 8 symbols

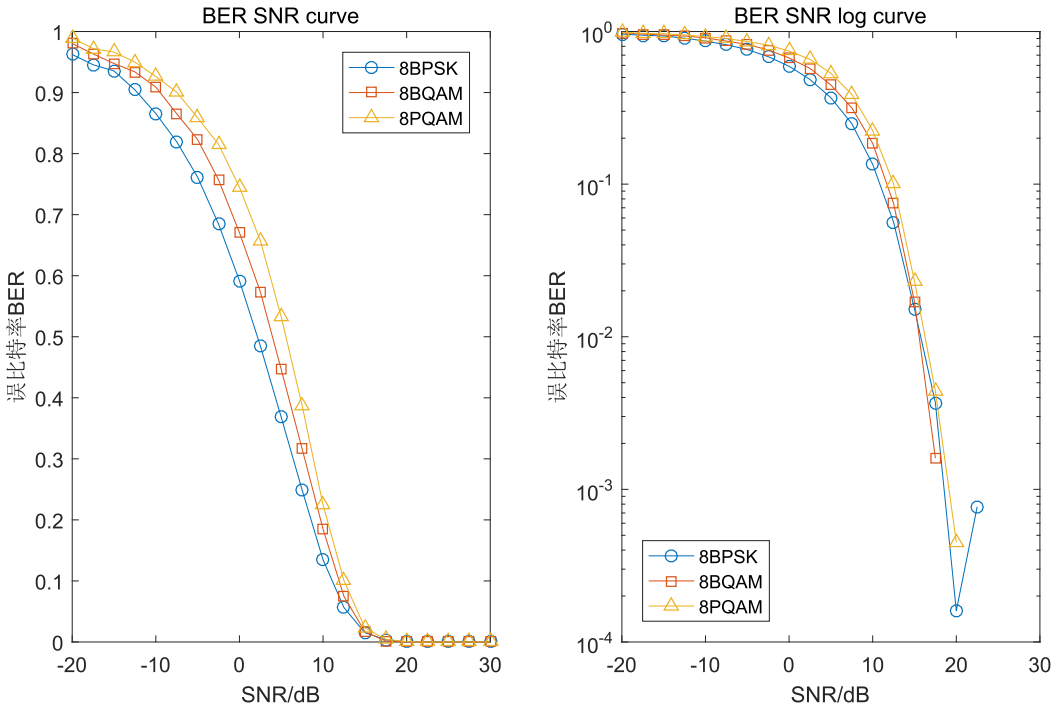
- 除去前面几种方法，我们还对3bit映射的QAM设计了另外一种映射方法，即将9QAM右上角的电平去掉，用剩下的8个复电平进行映射。因为此时的星座图是不对称的，因此如果对收端的星座图取平均，就能估计出右下角电平经过信道之后大致到达的位置，用这一位置来估计出固定相位噪声，用接收符号减去这一固定相位噪声，之后再进行判决。

- 发端星座图与收端星座图如下：





- 将这种Partial QAM与Bias-8QAM 以及 3bit映射的BPSK的误比特率曲线比较如下（后两者的bias ratio分别取0.15和0.2，都是对这两种映射方法来说较优）：



- 从图像中可以看出，在低信噪比的时候，BPSK的表现更好，而在高信噪比的时候，QAM系列的映射方法比BPSK的表现更好

3.2.5 固定相位映射方法总结

- 在这一部分中，我们共设计了四种映射方法，其中后两种是针对3bit映射的单独设计。
- 在信道使用次数有富余的情况下，可以考虑使用sequence-PSK来估计相位噪声并完成解调。
- 在信道使用次数较为严格但功率限制有富余的情况下，可以考虑使用两种加上bias-ratio的映射方法。

4. 卷积码编码和CRC校验

4.1 卷积码编码器

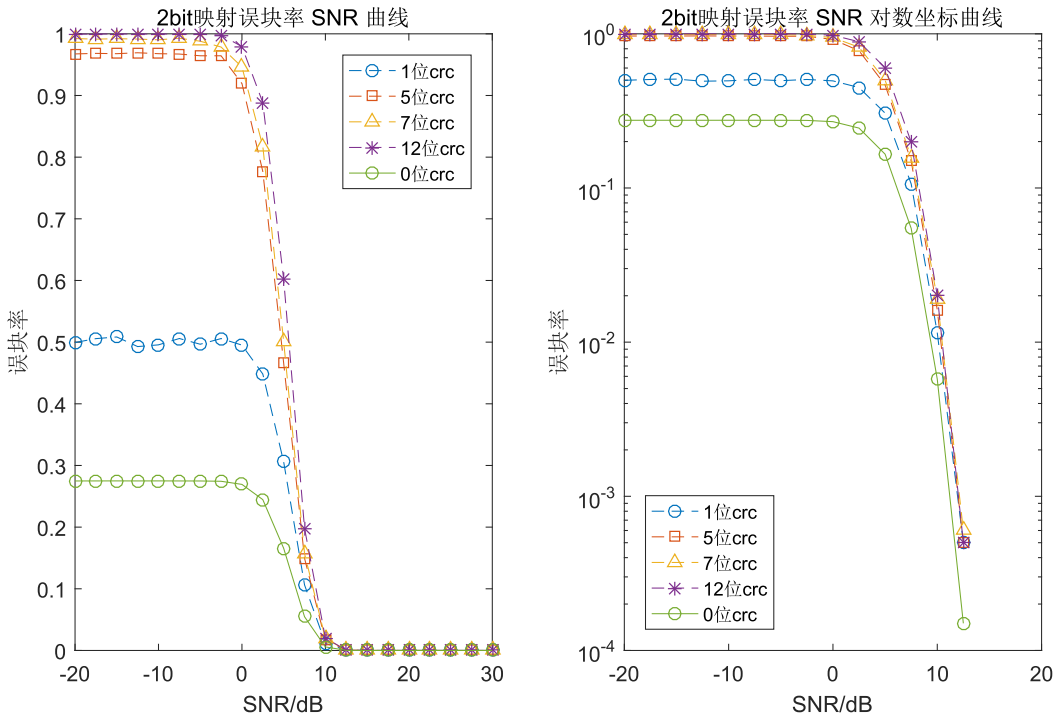
- 卷积码编码器实质上是将给定参数转化成二进制，然后用这个二进制串和输入序列做卷积。如果有n个参数，那么输出序列的长度就应当是原序列长度的n倍
- 对于每一个参数做卷积时，考虑到输出序列的长度，需要在序列最后一位进入卷积编码器的时候就停止输出，即除去卷积结果的最后$M-1$位，其中M为编码器的记忆长度。
- 若需要收尾，则需要在序列最后一位进入编码器的时候保证编码器内的M位都是0，因此需要向原序列末尾补上M个0之后再做卷积操作。
- 卷积码编码器的具体实现见附录7.2 卷积码代码

4.2 CRC校验部分

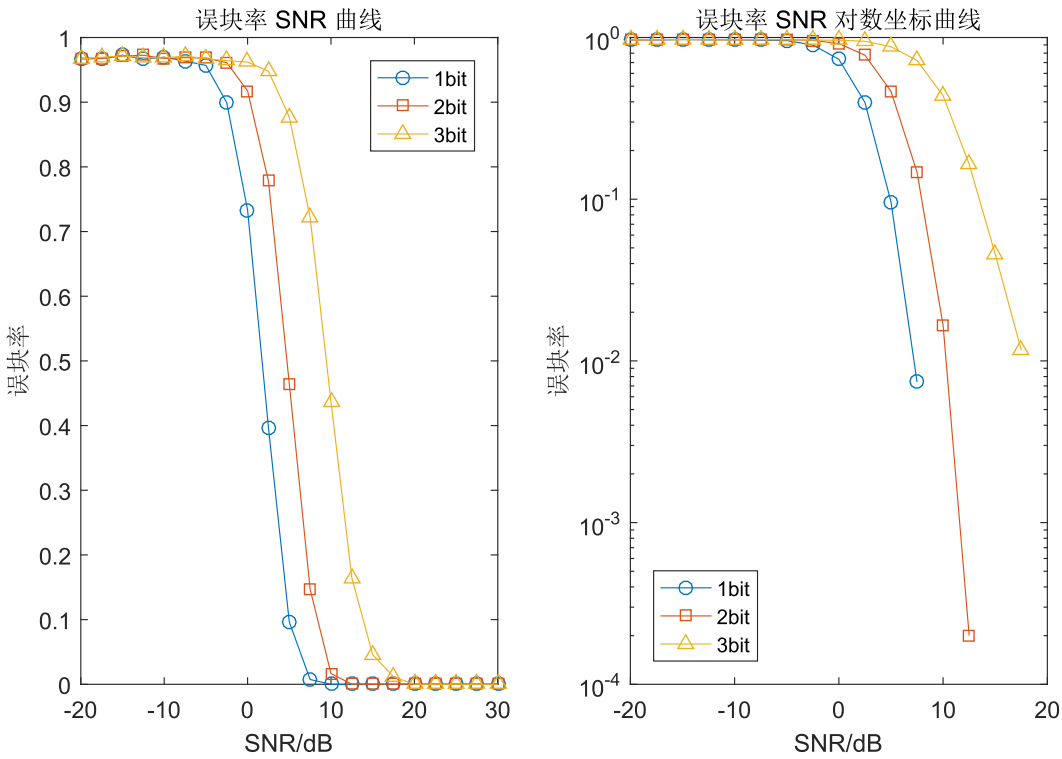
- CRC 校验码的添加本质上用到的是系数为0和1的多项式除法，因此可以转化为解卷积的问题，利用 $matlab$ 中的解卷积函数即可得到余式，对2取模就可以得到应当补在信息位之后的校验位。
- 用 CRC 做校验判断是否有误块时，只需要对收到的序列和生成多项式做解卷积操作，如果得到的余式模二全为0，则通过验证，否则报告错误。这部分代码见附录7.3 CRC代码
- 在这次编程作业中，我们实现了 $1,3,5,7,8,10,12,16$ 位的 CRC 冗余校验，根据ITU-IEEE规范，我们优先选取了用于 $CCITT$ 的生成多项式，在 3 位 CRC 冗余校验中，我们用了上课举例的生成多项式，选用的生成多项式如下：

校验位数	生成多项式	校验位数	生成多项式
1	$x+1$	8	$x^8+x^7+x^3+x^2+1$
3	x^3+x^2+1	10	$x^{10}+x^9+x^5+x^4+x+1$
5	x^5+x^3+x+1	12	$x^{12}+x^{11}+x^3+x^2+x+1$
7	x^7+x^3+1	16	$x^{16}+x^{12}+x^5+1$

- 根据实际要求，我们对25字节（100比特）的序列加上 CRC 冗余校验，尝试5位，7位，12位，在不编纠错码的情况下对通信过程进行仿真，得到三种 CRC 校验误块率与信噪比的关系，并与实际误块率进行比较。其中：CRC 误块率测10次取平均，真实误块率对30次测量取平均。由于 CRC 校验和映射比特数互不相关，因此我们选用2bit映射来测试在不同信噪比下 CRC 报错率和实际出错率的关系。

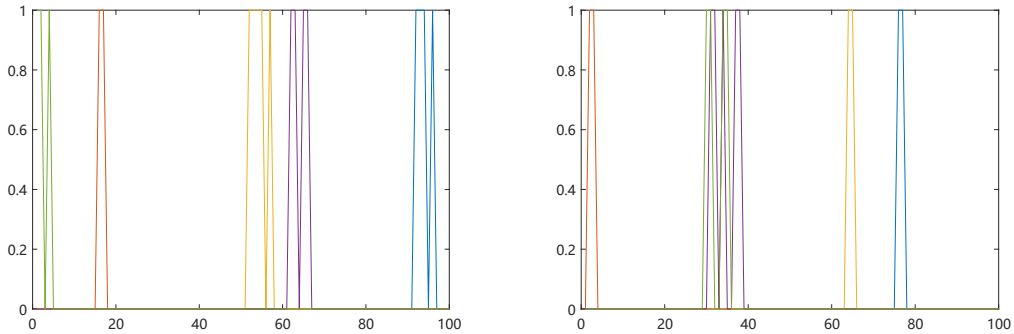


  - 可以看出，在低信噪比环境下，CRC 校验的虚警情况较为严重，这是因为CRC校验位本身也会出错，因此我们可以看到校验位长度越长，虚警情况越严重。但我们不能得出 CRC 校验位越长就越糟糕的结论，因为我们没有利用 CRC 来纠错，而如果CRC长度越长，那么其纠错能力应当越强，如果考虑到这一因素，则应当根据实际情况选择合适的 CRC 长度。
  - 我们大致做了一个判断，即：在低信噪比条件下，CRC 带来的弊端要大于其带来检错和纠错的好处，因此应该选择长度较短的 CRC ，此时由于所选的 CRC 长度较短，因此适合用于检错而非纠错；而在高信噪比条件下，CRC 带来的纠错方面的好处要大于其可能出错带来的坏处，因此可以选择长度较长的 CRC ，可以用作检错和纠错。
- 为了比较不同映射比特数之间的不同，我们针对5位 CRC 校验位，在不加纠错编码的情况下绘制出1，2，3bit调制下误块率和SNR曲线如下：



曲线和一般的误码率曲线特征相似，都是调制复电平数量越多，相同SNR下误块率越大。

- 加上卷积码编码和维特比硬判决之后，找到了信噪比在5dB下1bit映射的十个误码图案，绘制如下：



  - 其中颜色相同的为一个误码图案，可以看出由于卷积码的记忆性，这些发生错误的比特都是聚集在一起的。

5. 维特比译码

6. 传输方案的设计

7. 附录

7.1 信道代码

```
function [ output ] = channel( input ,mode, sigma )
%[ output ] = channel( input ,mode, sigma )
%   mode:    1 definite phi ; 2 different phi each time
%   sigma:   power of noise in real and imag part

```

```
switch mode
    case 1
        phi = random('unif',0,2*pi);
        phi = repmat(phi,1,length(input));
    case 2
        phi = random('unif',0,2*pi,1,length(input));
    otherwise
        error('模式错误')
end
n_r = sigma.*randn(1,length(input));
n_i = sigma.*randn(1,length(input));
n = n_r + i*n_i;
output = input.*exp(i.*phi)+n;
end
```

7.2 卷积码代码

```
function [ code ] = convcode( data, param, tail )
%CONVCODE 卷积编码 depend on matlab r2016a
%   convcode(data,rate,param,tail)
%       data:    0 1 序列，待编码内容
%       param:   列表，其中包括每个通道的8进制参数(数字格式)，
%                编码效率为1/n时，这里应该有n个元素
%       tail:    0表示不收尾，1表示收尾

rate = length(param);
param = num2str(reshape(param,rate,1));
param = base2dec(param,8);
param = dec2bin(param);
param = double(param == '1');
memory =size(param,2);
code = [];

if ~tail
    for k = [1:rate]
        code = [code;conv(data,param(k,:))];
    end
elseif tail
    data = [data,repmat(0,1,memory)];
    for k = [1:rate]
        code = [code;conv(data,param(k,:))];
    end
end
code = code(:,1:end-memory+1);
code = code(:);
code = mod(code,2)';

end
```

7.3 CRC 代码

```
function [ out ] = crc_encoder( data, num, block_len )
```

```
%CRC_ENCODER 此处显示有关此函数的摘要
%   此处显示详细说明
%[ out ] = crc_encoder( data, num, block_len )
%   out: 加好crc冗余的码字
%   data: 要加CRC的数据
%   num: CRC的位数，支持1, 3,5, 7, 8, 10, 12, 16
%   block_len: 每多少位加一次CRC
order = [1,3,5,7,8,10,12,16];
index = find(order==num);
table = cell(1,length(order));
table{1}=[1,1];
table{2}=[1,1,0,1];
table{3}=[1,0,1,0,1,1];
table{4}=[1,0,0,0,1,0,0,1];
table{5}=[1,1,0,0,0,1,1,0,1];
table{6}=[1,1,0,0,0,1,1,0,0,1,1];
table{7}=[1,1,0,0,0,0,0,0,0,1,1,1,1];
table{8}=[1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1];
block_num = length(data)/block_len;
out = zeros(1,block_num*(block_len+num));
for t=1:block_num
    [~,res]=deconv([data((t-
1)*block_len+1:t*block_len),zeros(1,num)],table{index});
    res = mod(res,2);
    out((t-1)*(block_len+num)+1:t*(block_len+num)) =...
        [data((t-1)*block_len+1:t*block_len),res(end-num+1:end)];
end

function [ flag ] = crc_judge( data,num )
%CRC_ 此处显示有关此函数的摘要
%   此处显示详细说明
%[ flag] = crc_judge( data,num )
%   flag: 数据是否正确
%   data: 用于判断的数据
%   num: crc长度，支持1, 3,5, 7, 8, 10, 12, 16

order = [1,3,5,7,8,10,12,16];
index = find(order==num);
table = cell(1,length(order));
table{1}=[1,1];
table{2}=[1,1,0,1];
table{3}=[1,0,1,0,1,1];
table{4}=[1,0,0,0,1,0,0,1];
table{5}=[1,1,0,0,0,1,1,0,1];
table{6}=[1,1,0,0,0,1,1,0,0,1,1];
table{7}=[1,1,0,0,0,0,0,0,0,1,1,1,1];
table{8}=[1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1];
[~,res]=deconv(data,table{index});
res = mod(res,2);
if sum(res)==0
    flag = 1;
else
    flag = 0;
```

```
54 end
55 end
```