目录

简介

本次实验使用MATLAB为工具，进行通信系统仿真，仿真期间使用不少现成的库文件以及自己编写的函数，达到了目的。

引言

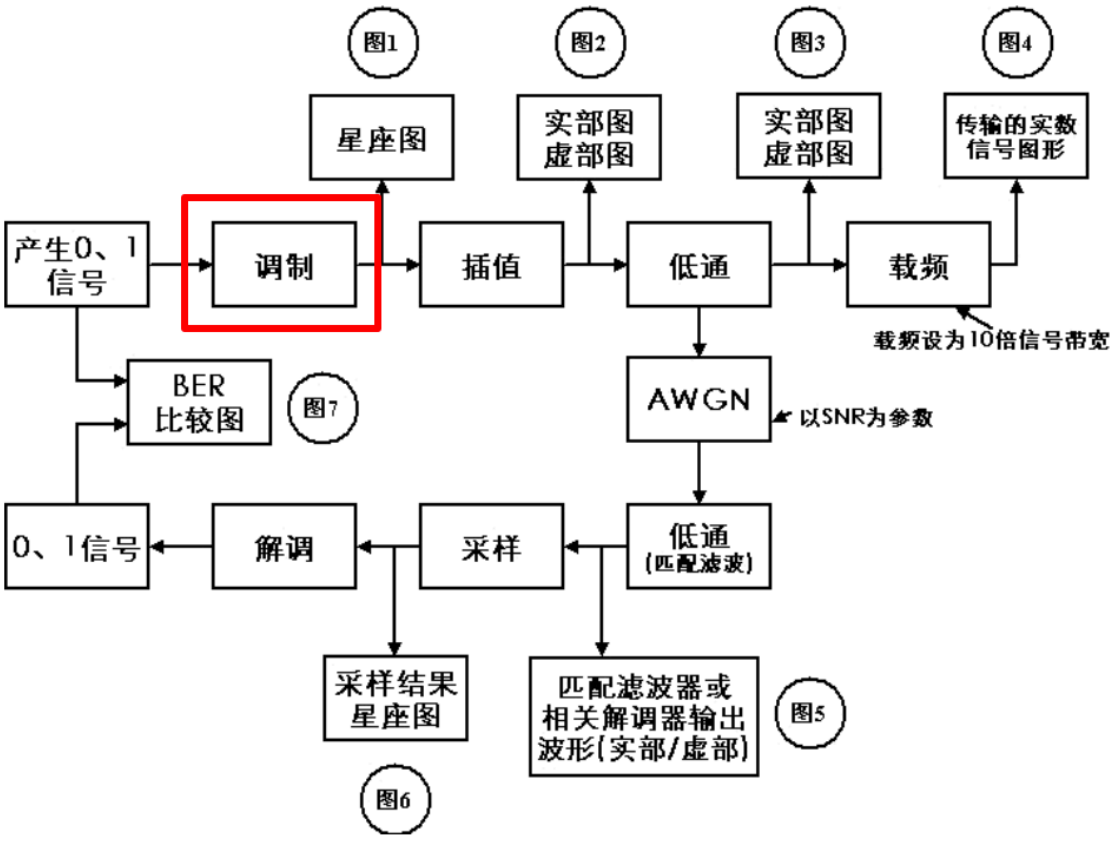
这次实验我并没有那么认真去做，所以有些能做到的也没有做到，查阅了网上很多资料，也获得了很多启发。希望有错之处，宽容海涵。

运行环境是：matlab2020

这一个版本没有了randint，我用的random所以一般都是在各版本都通用的。

仿真结果展示图

实验总体框图简介



产生0、1离散信号序列

我在这里使用了$rand(m,n)$函数和$round(a)$函数用来模拟产生的二进制序列，由于本身$rand(m,n)$是均匀分布0～1之间的数，再配合$round(a)$进行四舍五入，可以遇见获得随机均匀分布的0、1二进制序列。实验结果：

```
randomZeroOne =

     0     0     1     1     1     0     1     1     0     1     0     0     1     1     1     1     0     1     1     1

```

调制

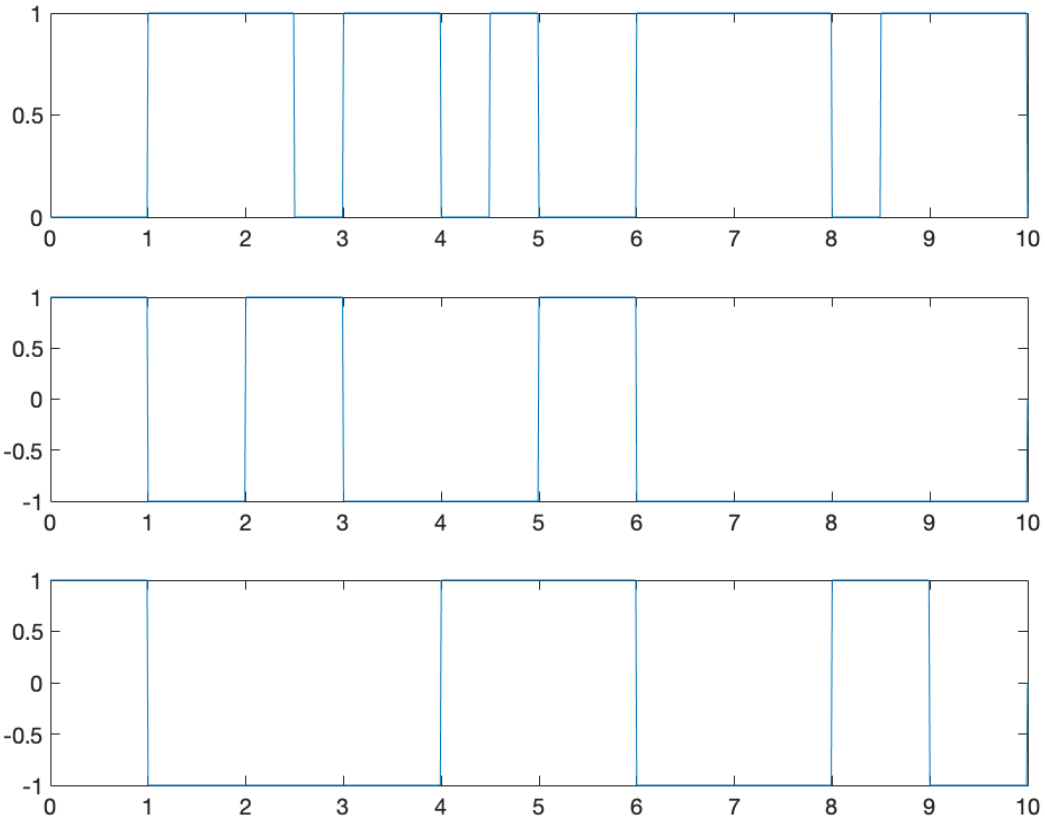
调制信号，顾名思义将二进制数变为QPSK的编码，其实实际上是在**四进制**与**I路Q路**之间作出的**映射**，排列按照格雷码排列。如下图：

$$
\begin{aligned}
&00 \rightarrow I:+1 \quad Q:+1 \\
&01 \rightarrow I:-1 \quad Q:+1 \\
&11 \rightarrow I:-1 \quad Q:-1 \\
&10 \rightarrow I:+1 \quad Q:-1
\end{aligned} \tag{1}
$$

那么在期间，我并没有使用$genqammod()$函数来进行，而是通过自己的理解来编写了由最原始的2进制数变为4进制数的函数$b2f()$，顾名思义binarytoFour，以及由4进制变为I路和Q路的映射函数$ZeroOne2IQ()$。

由于接下来的操作时采样，但是由于为了更加逼近真实的情况，我将用原先产生的随机0、1序列来产生**连续模拟**的0、1信号。并且通过自己编写的$generateContinuousZeroOne()$，借用系统自带的脉冲构造函数$pulstran()$构造了由原先随机0、1**离散**序列构成的**连续模拟**0、1信号序列。实验图展示了随机连续0、1序列，和相对应调制的I路和Q路。实验结果如下：

其中顶部为随机信号序列的连续模拟信号，中部的是调制的I路连续模拟信号，最底部的是调制的Q路连续模拟信号。



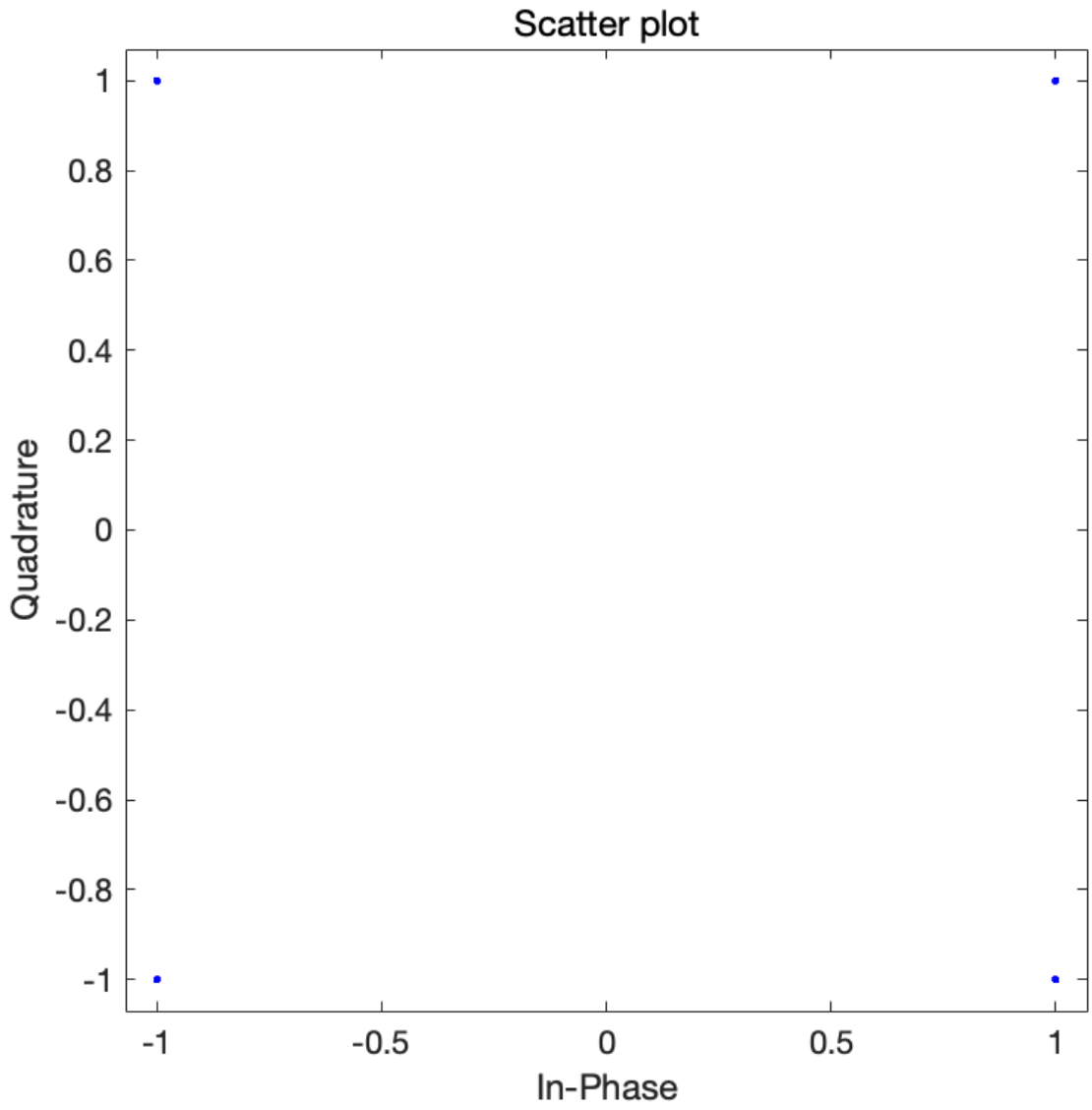
I、Q路的星座图

首先，关于星座图，我在上面并没有的到复数指，但是却获得了组成星座图的两个坐标，一个是I路，一个是Q路。我将上面的到的两个路的值进行合并，如下操作

```
I + Q * 1i
```

将I路的值放在实轴，将Q路的值放在虚轴，构成星座图。

实验图结果如下：



可以看到在$\pi/4$、$3\pi/4$、$5\pi/4$、$7\pi/4$有着4个幅度为$\sqrt{2}$的点。

插值

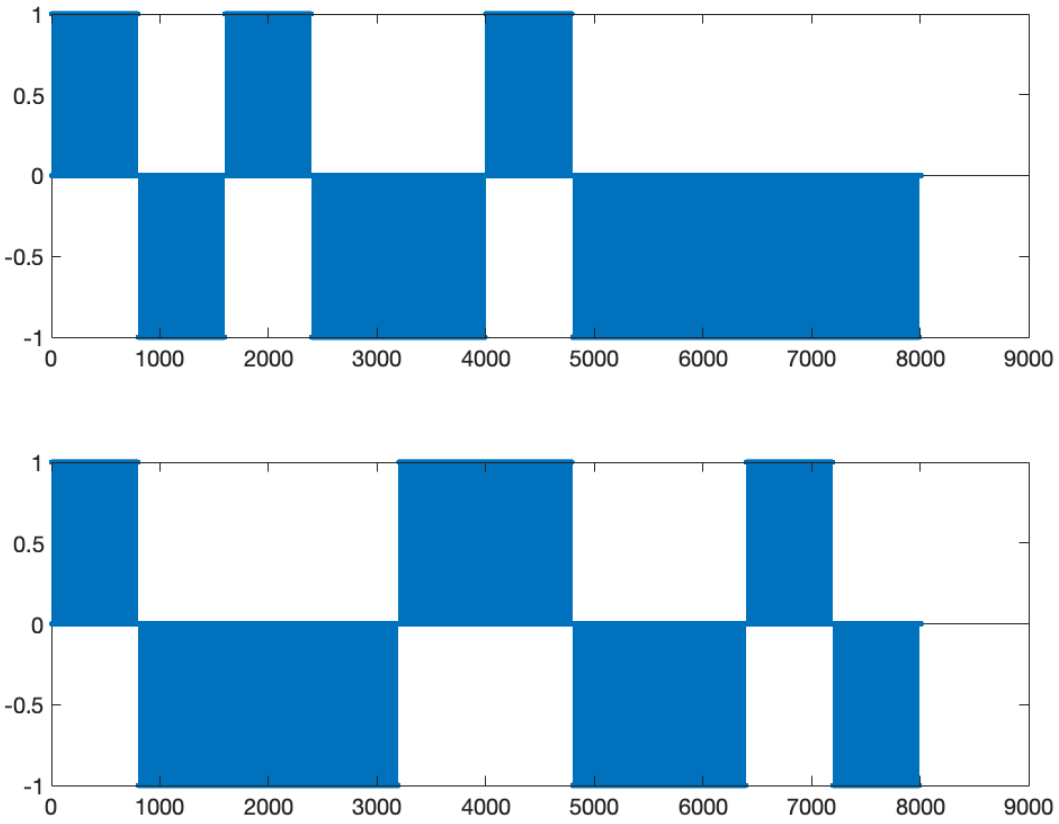
实际上，我对其也有了思路，不过碍于时间，就直接使用$upsample$。可叙述如下：

1. 对要进行采样的连续信号进行扩充，如果要采样8倍的点，那么就在时间轴上扩大8倍。调用$zeros(1, 8 * length(coninuousSignal))$来声明空间
2. 对上述声明空间进行每次进行迭代，迭代8步，然后获取原先的值

```
Zt(1:8:end) = x
```

这直接调用$upsample()$，实验结果如下：

其中顶部是上采样的I路，底部是上采样的Q路。

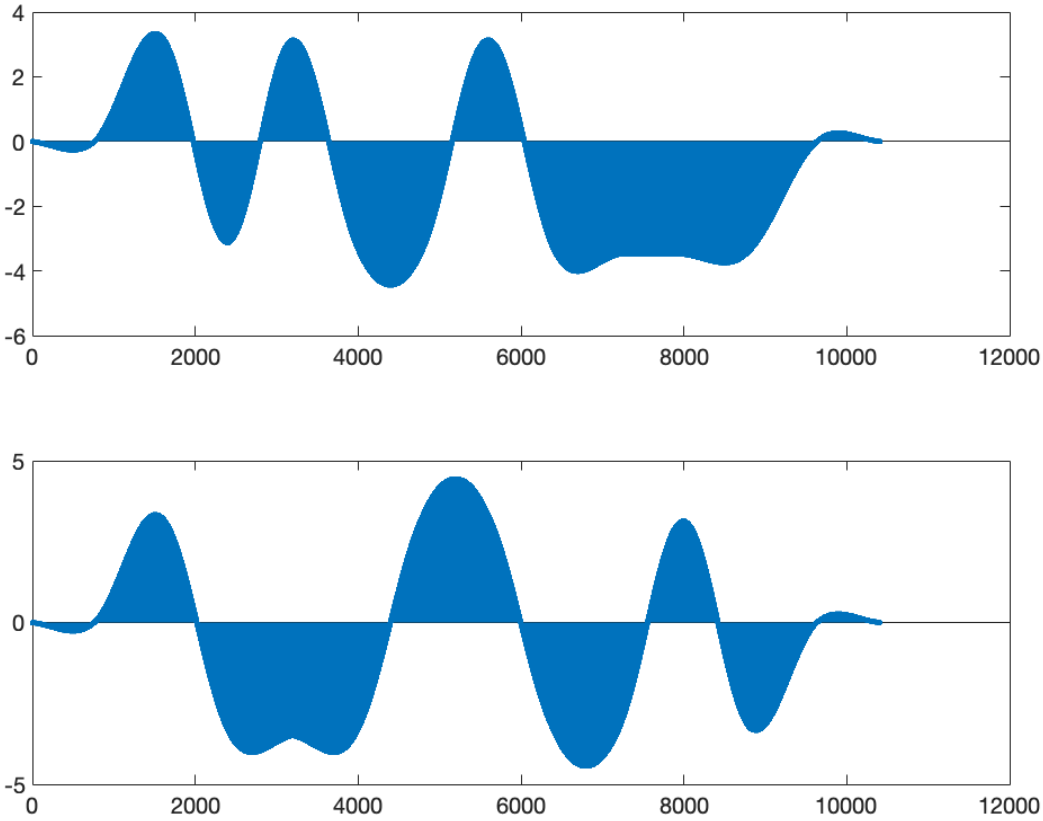


低通

一半我们在通信系统中，经常在发送端使用成型滤波器，在接收端使用匹配滤波，也就是相关接收。

这里我们用的是$rcosdesign()$，这里碍于时间关系，并没有自己写，不过有思路，还未完全想明白，待以后填坑。在这里用的是根升余弦，将高频成分去除，在实验图中可以看到，关于1的连续信号棱角都被磨平了。实验图如下：

其中顶部是低通成型滤波的I路，底部是低通成型滤波的Q路。



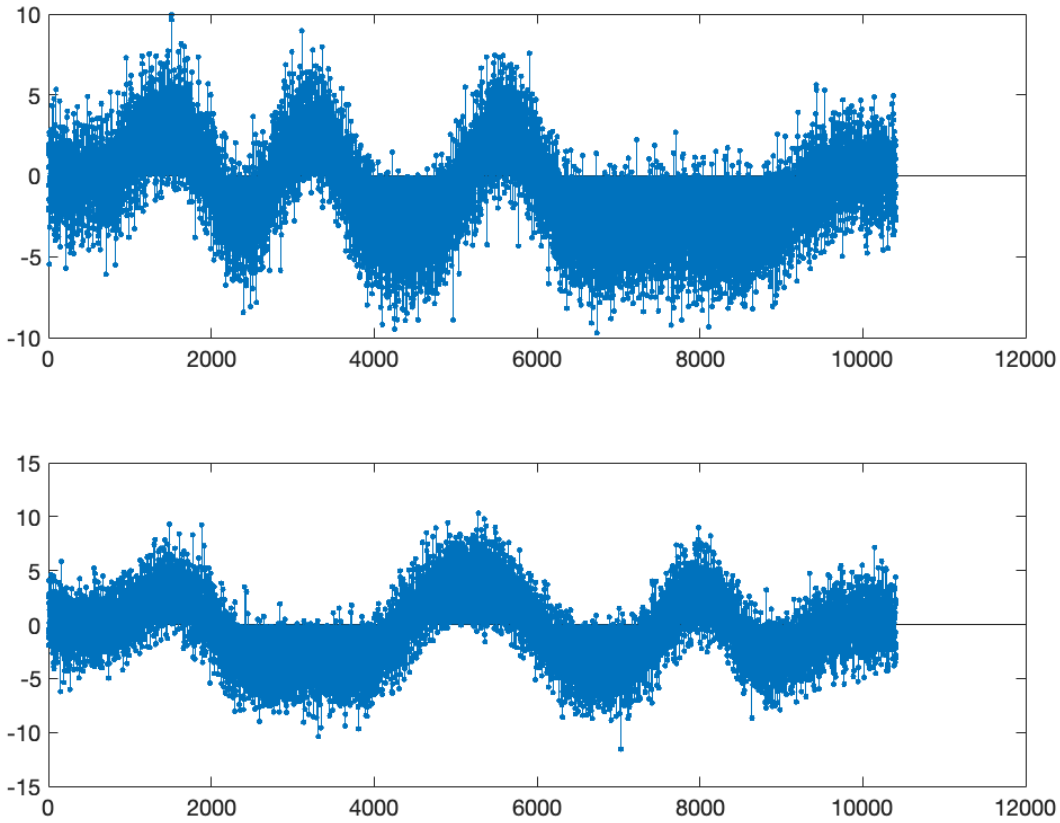
Awgn

百度百科上讲，*AWGN*又称加性高斯白噪声(Additive White Gaussian Noise)，是最基本的噪声与干扰模型。所以我们在这里用来模拟信号在信道之间传输的干扰。

自己如果书写也有一定思路，但碍于时间。思路来自$randn()$用来产生随机数，关于$N(0,1)$标准正态分布。

在这里需要注意的是SNR指的是信号噪声比，也就是说，如果SNR很大，说明噪声很小，如果SNR很小，说明噪声已经严重影响信号。我在这里取$SNR=3$，较为适中。实验结果如下：

下图的顶部是通过信道AWGN的I路，底部是通过信道AWGN的Q路



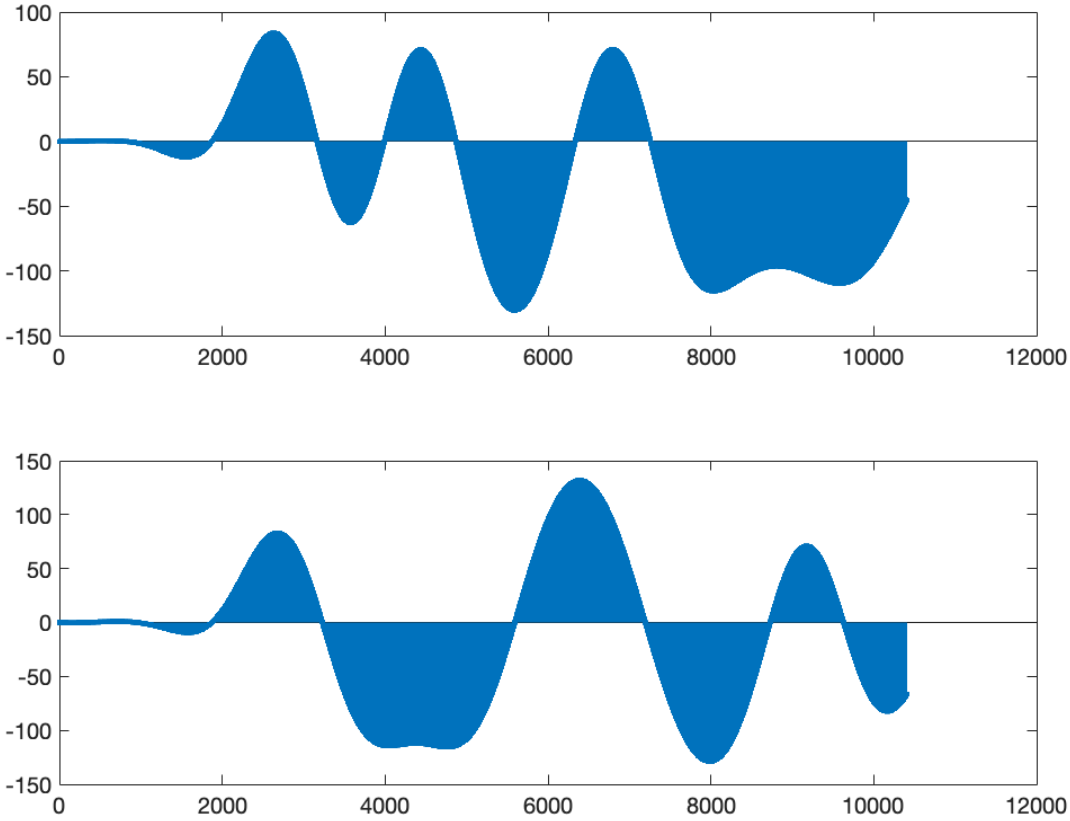
匹配滤波

在上面已经讲到，通信系统一般使用的是匹配滤波来进行接受相关信息。

可以看到实际上发送端的低通滤波是为了减少高频噪声，来获得更大的有效发送功率，那么匹配滤波就是为了使得接受的的信号信噪比最大，也就是说信号非常好，大家一致都是为了信号的可靠性。

那么在这里我们将用到的是$filter()$函数和$rcosdesign()$所返回的系统$h(t)$，将其用于从信道传输过来的信号，可以获得具有最高信噪比的发送信号。实验结果如图所示：

下图的顶部是接受匹配滤波的I路，底部是接受匹配滤波的Q路



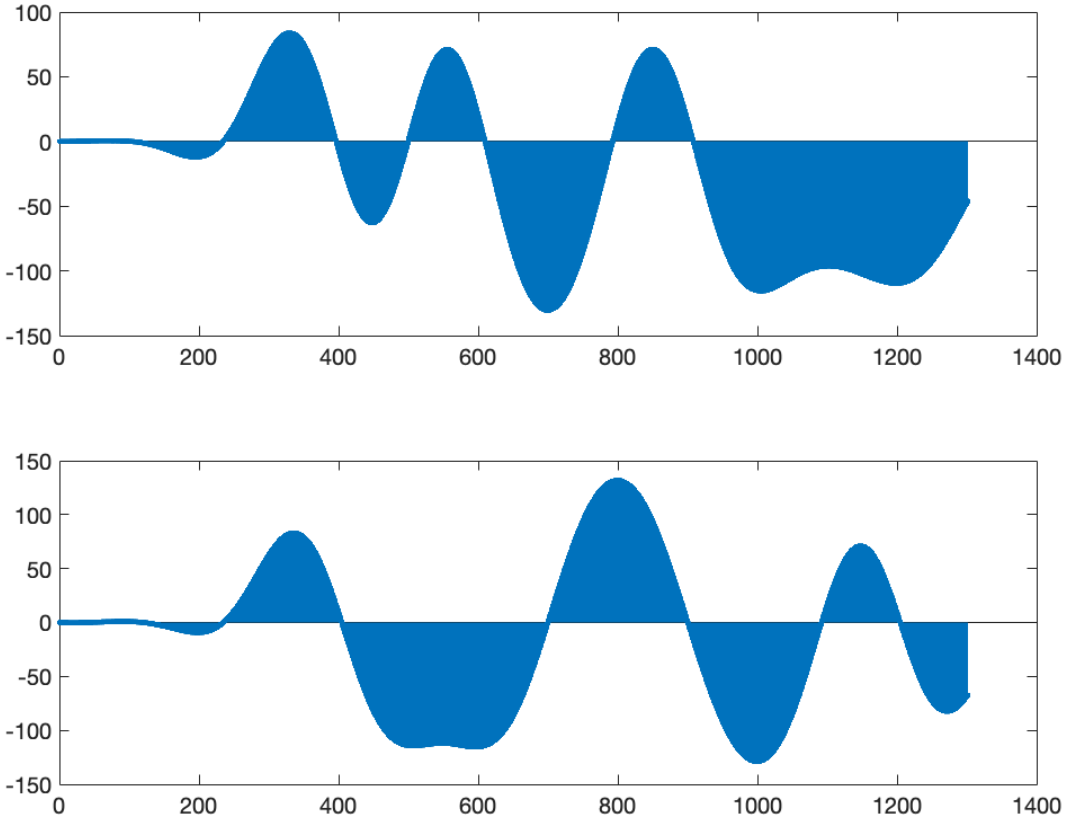
下采样

在上面已经提到过上采样，实际上下采样就是上采样的一个逆过程，用于减少传输的数据量，个人理解。那么实际上就是:

```
xt = x(1:8:end)
```

将原始信息缩减了8倍，但是却具有一样的信息量。实验结果如下，可以看到下标发生了变化：

下图的顶部是接受信号下采样的I路，底部是接受信号下采的Q路



抽样检查

实际上，抽样检查，是包括在采样里的，但是我将其拿出来，然后自己编写了一个抽样检查，其函数名为$sampling()$。主要功能是将I路和Q路经过下采样数据量依旧很大，所以进行了抽样检查，进行还原I路和Q路的离散序列信号。

其中的困难有：

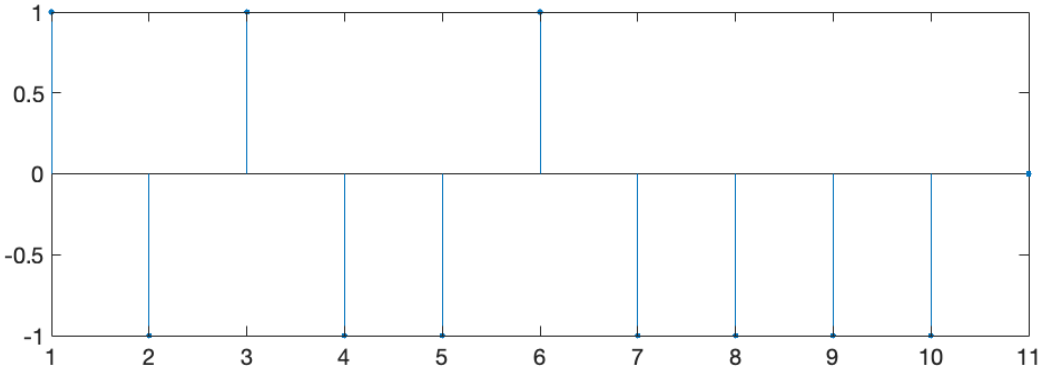
1. 传输过后，信号的还原中存在时延长现象。不好把控长度的切片。
2. 传输过程中有噪音，容易误判。

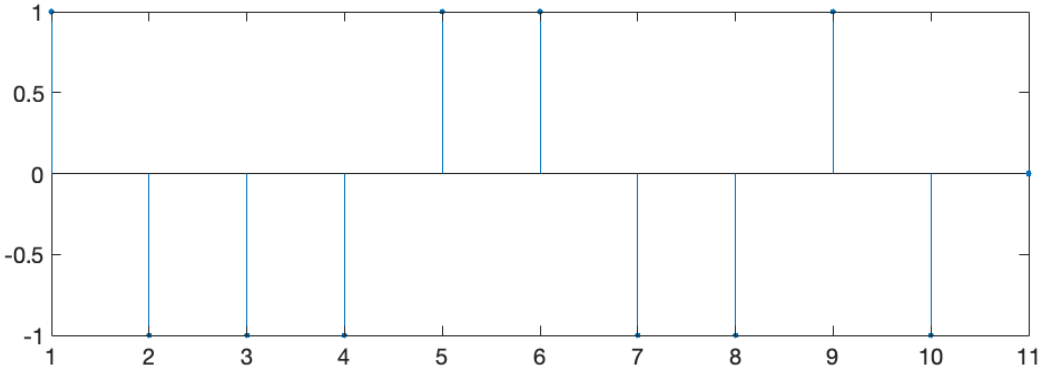
好在，差不多都克服了。思路如下：

1. 不管时延现象，直接进行对于大于0的值为1，反之则为零的判断
2. 获得抽样的信号的周期，由于我这是100个点为一个信号，我就将其设为$100*i+50$，i为信号周期有多少个，每次要检查在中心的点的值。
3. 最后通过裁剪前面无用的时延判断结果，来获得正确的信号值，当然我在这里末尾并没有进行裁剪，所以可能存在一些误差。

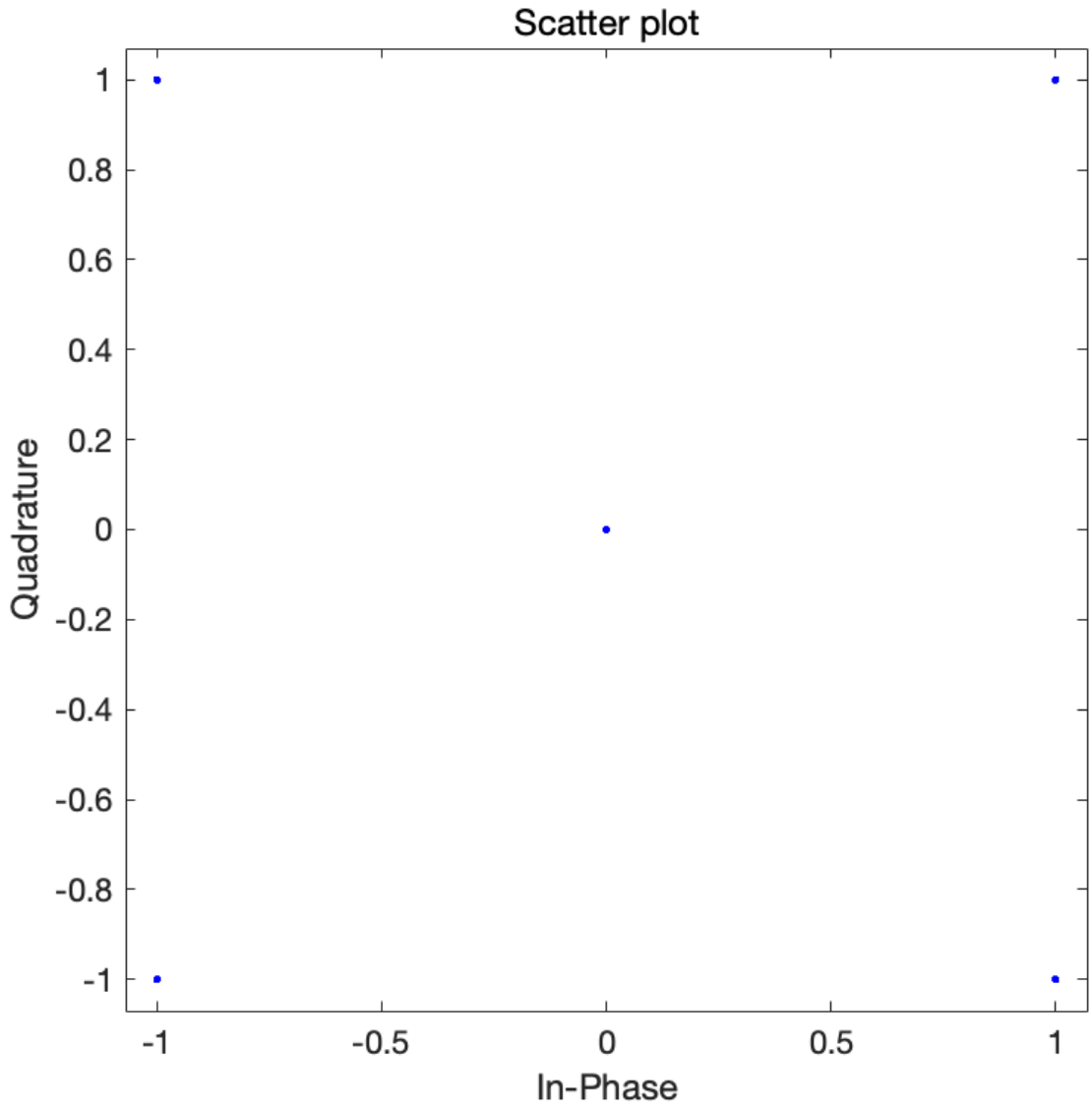
实验结果如下：

下图的顶部是下采样I路抽样的值，底部是下采样Q路抽样的值


1
0.5
0
-0.5
-1
1
2
3
4
5
6
7
8
9
10
11


1
0.5
0
-0.5
-1
1
2
3
4
5
6
7
8
9
10
11

星座图如下：



为什么会在$(0,0)$处出现点，因为在抽样检查的时候我会将未知的，时延的的无用点作为(0,0)处理。

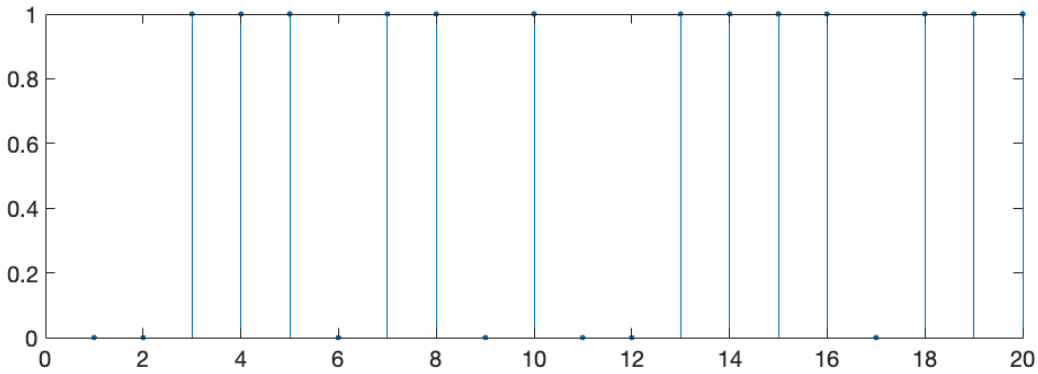
解调

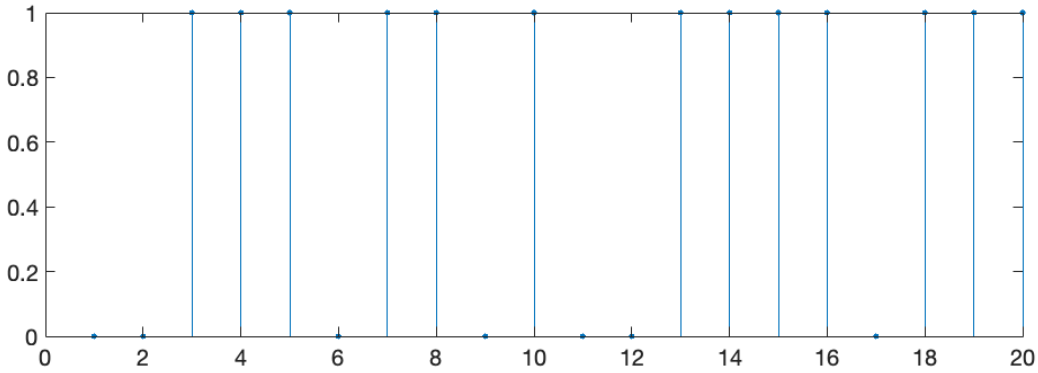
在上面，已经叙述过调制的问题是如何解决的，实际上解调就是调制的逆过程。

我的思路是，先将获得的I路、Q路信号进行转换，转换成4进制值，用的是自己编写的$IO2Four()$函数，其次在通过上述（1）式的映射规则来获得真实的二进制离散序列，这里使用的是自己编写的$f2b()$函数。

实验结果如下：

下图中顶部的是原始0、1离散信号，底部是解调而来的0、1离散信号



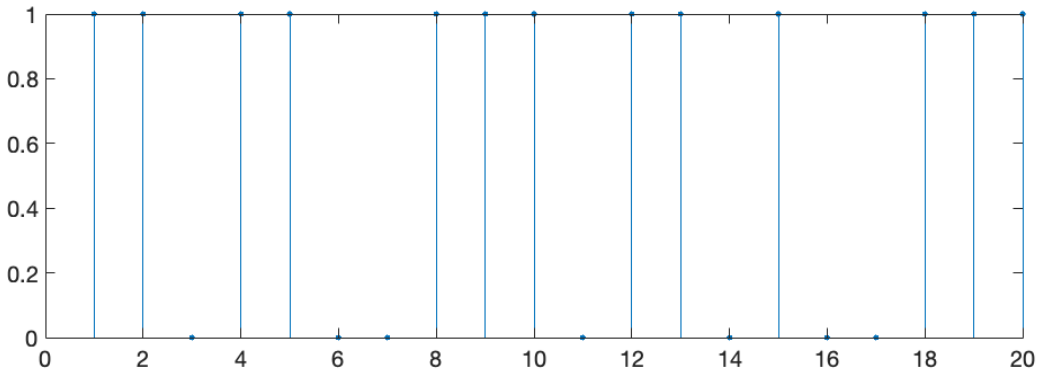


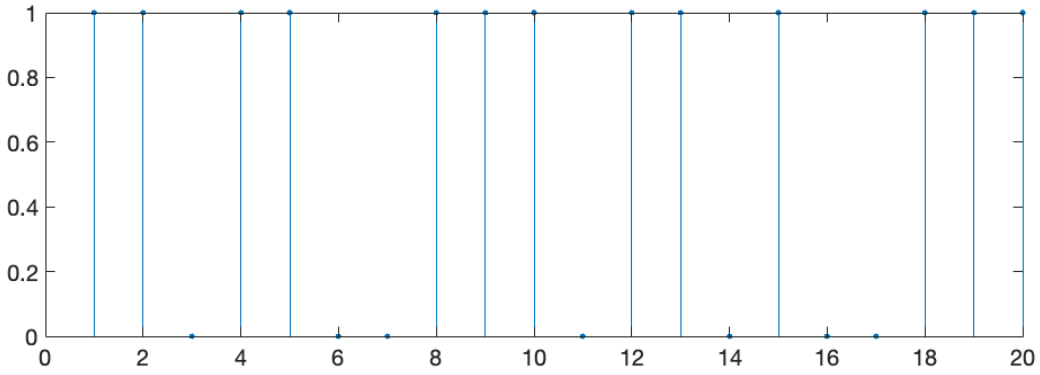
可以看到误码率为0。

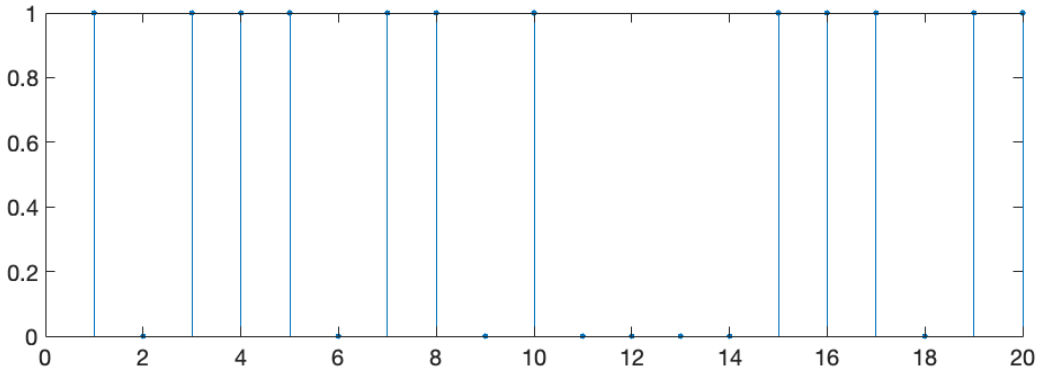
BER比较图

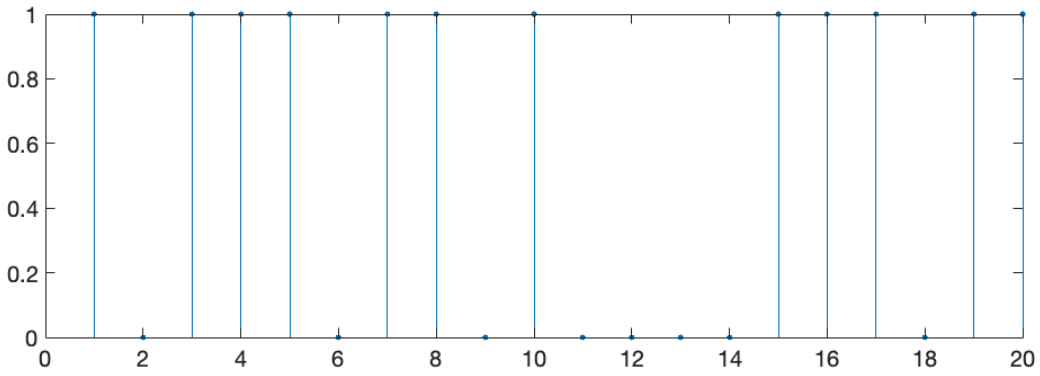
在这里我将会进行，5次20个码元共计100个码元的传输接受模拟，虽然可能过于少量，但也有一定实验价值。图如下：

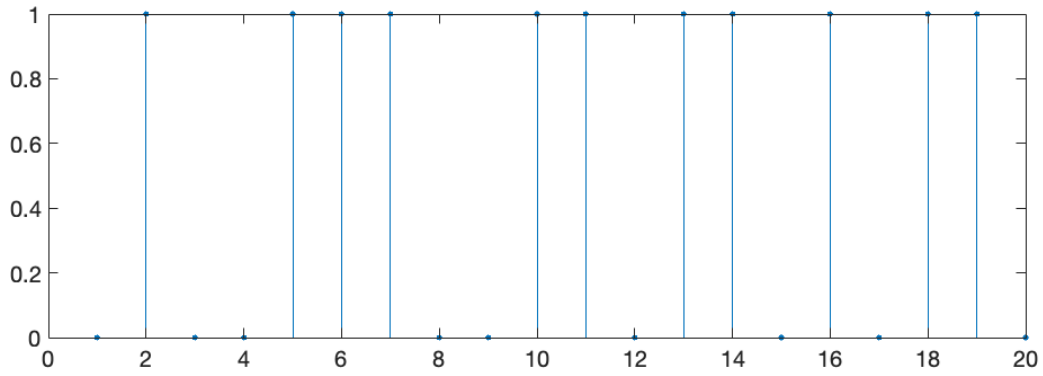
100个码元传输完全没有问题，误码率为0。

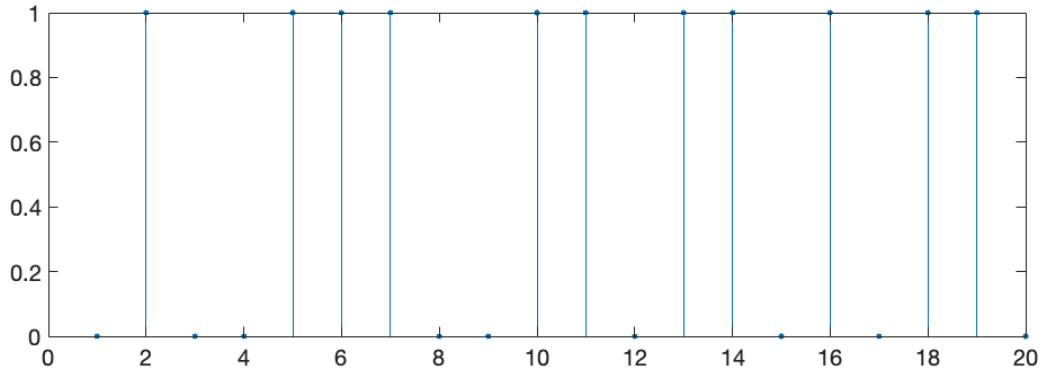

1
0.8
0.6
0.4
0.2
0
0 2 4 6 8 10 12 14 16 18 20

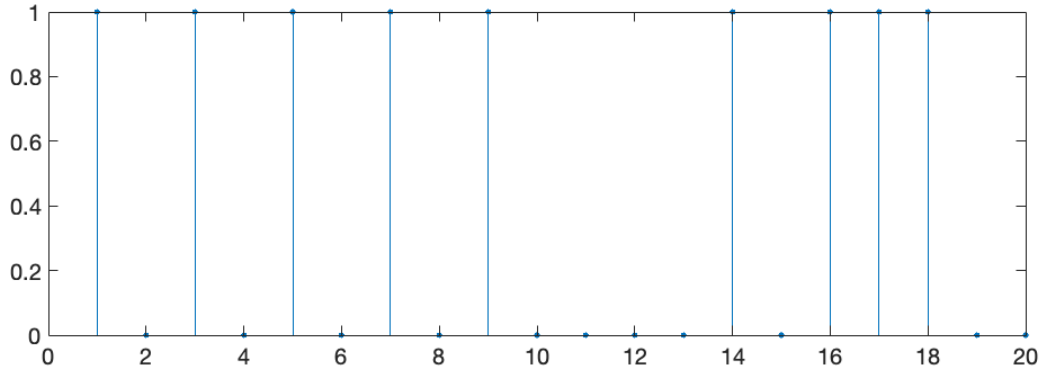

1
0.8
0.6
0.4
0.2
0
0 2 4 6 8 10 12 14 16 18 20

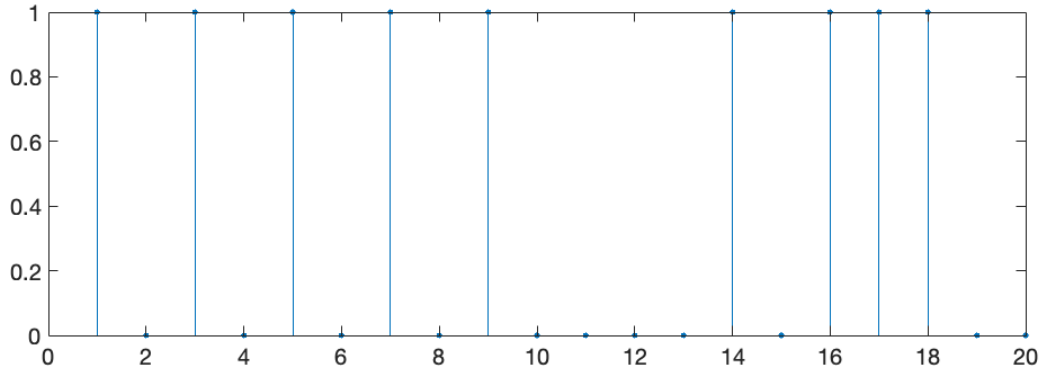

1
0.8
0.6
0.4
0.2
0
0 2 4 6 8 10 12 14 16 18 20

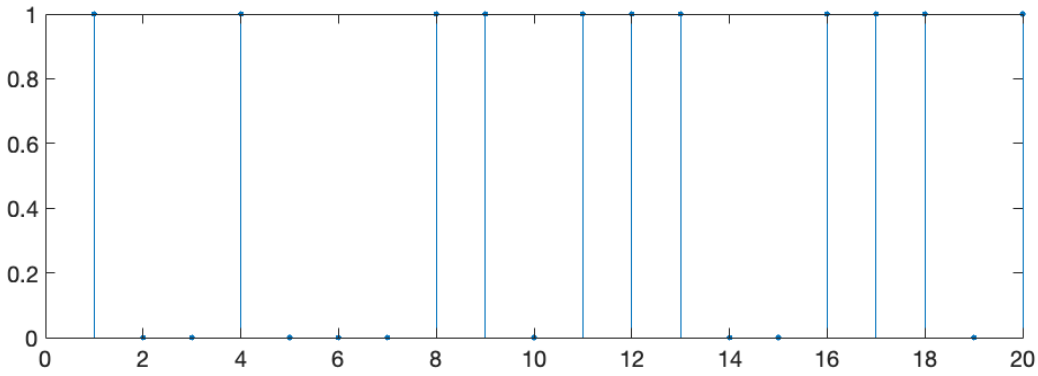

1
0.8
0.6
0.4
0.2
0
0 2 4 6 8 10 12 14 16 18 20

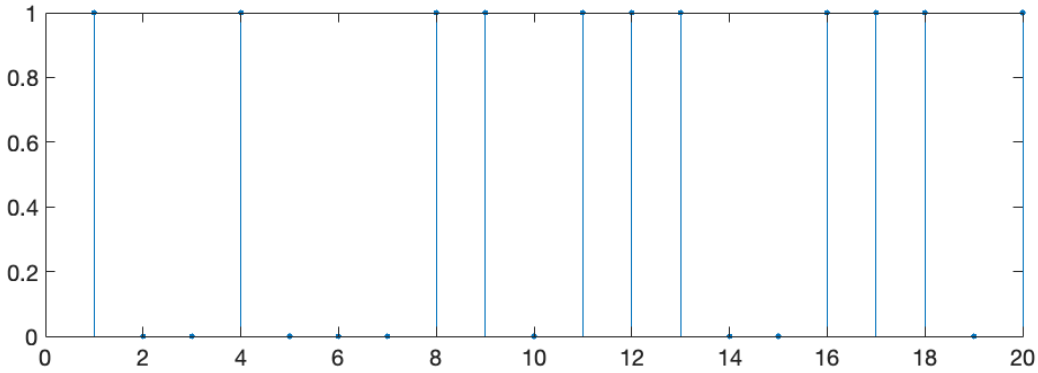

1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
14
16
18
20


1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
14
16
18
20


1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
14
16
18
20


1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
14
16
18
20





总代码表

Main.m

```
clear;
close all;
clc;
% 产生随机10序列
randomZeroOne = getZeroOne(1, 20)
% 产生连续模拟10序列信号
Time = 0:0.01:10;
timeSlot = 0:0.5:9.5;
timeSlotRandomZeroOne = [timeSlot;randomZeroOne].';
continuousZeroOne = generateContinuousZeroOne(Time,
timeSlotRandomZeroOne, 1/2);
figure("Name", "Order and IQ");
subplot(311);
plot(Time, continuousZeroOne);
%QPSK调制
% 2进制序列转4进制序列
arrayFour = b2f(randomZeroOne);
% I路和Q路转换
[I,Q] = ZeroOne2IQ(arrayFour);
```

```
% 产生IQ序列图像

timeSlot = 0:9; % 应为I路Q路的时间间隔是原信号的2倍
timeSlotRandomZeroOne = [timeSlot;I].';
continuousZeroOneI = generateContinuousZeroOne(Time,
timeSlotRandomZeroOne, 1);
subplot(312);
plot(Time,continuousZeroOneI);
timeSlot = 0:9; % 应为I路Q路的时间间隔是原信号的2倍
timeSlotRandomZeroOne = [timeSlot;Q].';
continuousZeroOneQ = generateContinuousZeroOne(Time,
timeSlotRandomZeroOne, 1);
subplot(313);
plot(Time,continuousZeroOneQ);
% 星座图
scatterplot(I+Q*1i);
% 产生连续模拟10序列信号 用于I路 Q路
figure("Name", "IQ路序列图像");
subplot(211);
stem(I, '.');
subplot(212);
stem(Q, '.');
% 上采样 原信号2Hz IQ信号1Hz 采样用800Hz采样
M = 8;
IUpsampling = upsample(continuousZeroOneI, M);
QUpsampling = upsample(continuousZeroOneQ, M);
figure("Name","upsampling");
subplot(211);
stem(IUpsampling, '.');
subplot(212);
stem(QUpsampling, '.');
% 低通滤波
Fs = 800; %通过10s采集8000样本点
Fd = 1; % 因为是对IQ路做采样
rolloff = 0.5;
filterRcosdesign = rcosdesign(rolloff, 3, Fs/Fd);
IFilterLow = conv(IUpsampling, filterRcosdesign);
QFilterLow = conv(QUpsampling, filterRcosdesign);
figure("Name","低通滤波");
subplot(211);
stem(IFilterLow, ".");
subplot(212);
stem(QFilterLow, ".");
% AWGN
SNR = 3;
IAwgn = awgn(IFilterLow, SNR, "measured");
QAwgn = awgn(QFilterLow, SNR, "measured");
figure("Name", "AWGN 模拟信道")
subplot(211);
```

```
stem(IAwgn, '.');
subplot(212);
stem(QAwgn, '.');
% 匹配滤波
IMatchedFilter = filterRcosdesign;
QMatchedFilter = filterRcosdesign;
IMatchedFilterResult = filter(IMatchedFilter,1, IAwgn);
QMatchedFilterResult = filter(QMatchedFilter,1, QAwgn);
figure("Name", "Result of MatchedFilter");
subplot(211);
stem(IMatchedFilterResult, '.');
subplot(212);
stem(QMatchedFilterResult, '.');
% 下采样
Idownsampling=downsample(IMatchedFilterResult,M);
Qdownsampling=downsample(QMatchedFilterResult,M);
figure("Name", "downsampling");
subplot(211);
stem(Idownsampling, '.');
subplot(212);
stem(Qdownsampling, '.');
% 抽样检查
[ISampling, QSampling] = sampling(Idownsampling, Qdownsampling);
ISampling = ISampling(4:end);
QSampling = QSampling(4:end);
figure("Name", "抽样检查后的01序列");
subplot(211);
stem(ISampling, '.');
subplot(212);
stem(QSampling, '.');
% 抽样后的星座图
scatterplot(ISampling+QSampling*1i);
% 解调
demodulationArrayIO2Four = IQ2Four(ISampling, QSampling);
demodulationArrayZeroOne = f2b(demodulationArrayIO2Four);
figure("Name", "发送信号序列和接受信号序列对比图");
subplot(211);
stem(randomZeroOne,'.');
subplot(212);
stem(demodulationArrayZeroOne, '.');
```

b2f.m

```
function arrayFour = b2f(arrayZeroOne)
    bm = [0,1,2,3];
    arrayFour = zeros(1, length(arrayZeroOne) / 2);
    for i = 1:length(arrayFour)
        twoSum = 0;
        twoSum = twoSum + arrayZeroOne((i-1) * 2 +1) * 2 + arrayZeroOne((i-1) * 2 +2);
        arrayFour(i) = bm(twoSum+1);
    end
end
```

f2b.m

```
function arrayZeroOne = f2b(arrayFour)
    n = length(arrayFour);
    for i = 1:n
        if arrayFour(i) == 4
            arrayFour = [arrayFour(1:i-1),arrayFour(i+1:end)];
        end
    end
    n = length(arrayFour);
    arrayZeroOne = zeros(1, n * 2);
    for i = 1:n
        switch arrayFour(i)
            case 0
                arrayZeroOne((i-1)*2+1) = 0;
                arrayZeroOne((i-1)*2+2) = 0;
            case 1
                arrayZeroOne((i-1)*2+1) = 0;
                arrayZeroOne((i-1)*2+2) = 1;
            case 2
                arrayZeroOne((i-1)*2+1) = 1;
                arrayZeroOne((i-1)*2+2) = 0;
            case 3
                arrayZeroOne((i-1)*2+1) = 1;
                arrayZeroOne((i-1)*2+2) = 1;
        end
    end


end
```

generateContinuousZeroOne.m

```
function continuousZeroOne = generateContinuousZeroOne(Time,
timeSlotRandomZeroOne, width)
    continuousZeroOne = pulstran(Time-width/2,
timeSlotRandomZeroOne,'rectpuls', width);
end
```

getZeroOne.m

```
function b = getZeroOne(m, n)
    a = rand(m, n);
    b = round(a);
end
```

IO2Four.m

```
function arrayFour = IQ2Four(I, Q)
    n = length(I);
    arrayFour = zeros(1, n);
    for i = 1: n
        if I(i) == 0 || I(i) == 0
            arrayFour(i) = 4;
            continue;
        end

        if I(i) == 1
            if Q(i) == 1
                arrayFour(i) = 0;
            else
                arrayFour(i) = 2;
            end
        else
            if Q(i) == 1
                arrayFour(i) = 1;
            else
                arrayFour(i) = 3;
            end
        end
    end
end
```

sampling.m

```
function [I, Q] = sampling(IDownsampling, QDownsampling)
    n = length(IDownsampling);
    head = 1;
```

```
    tail = round(n / 100);
    I = zeros(1, tail+1);
    Q = zeros(1, tail+1);
    for i = head: tail
        if IDownsampling((i-1)*100+50) >= 0
            I(i) = 1;
        else
            I(i) = -1;
        end
        if QDownsampling((i-1)*100+50) >= 0
            Q(i) = 1;
        else
            Q(i) = -1;
        end
    end
end


```

ZeroOne2IQ.m

```
function  [I,Q] = ZeroOne2IQ(arrayFour)
    n = length(arrayFour);

    I = zeros(1,n);
    Q = zeros(1,n);
    for i = 1: n
        switch arrayFour(i)
            case 0
                I(i) = 1;
                Q(i) = 1;
            case 1
                I(i) = -1;
                Q(i) = 1;
            case 2
                I(i) = 1;
                Q(i) = -1;
            case 3
                I(i) = -1;
                Q(i) = -1;
        end
    end
end


```