MATLAB 大作业 语音通信系统

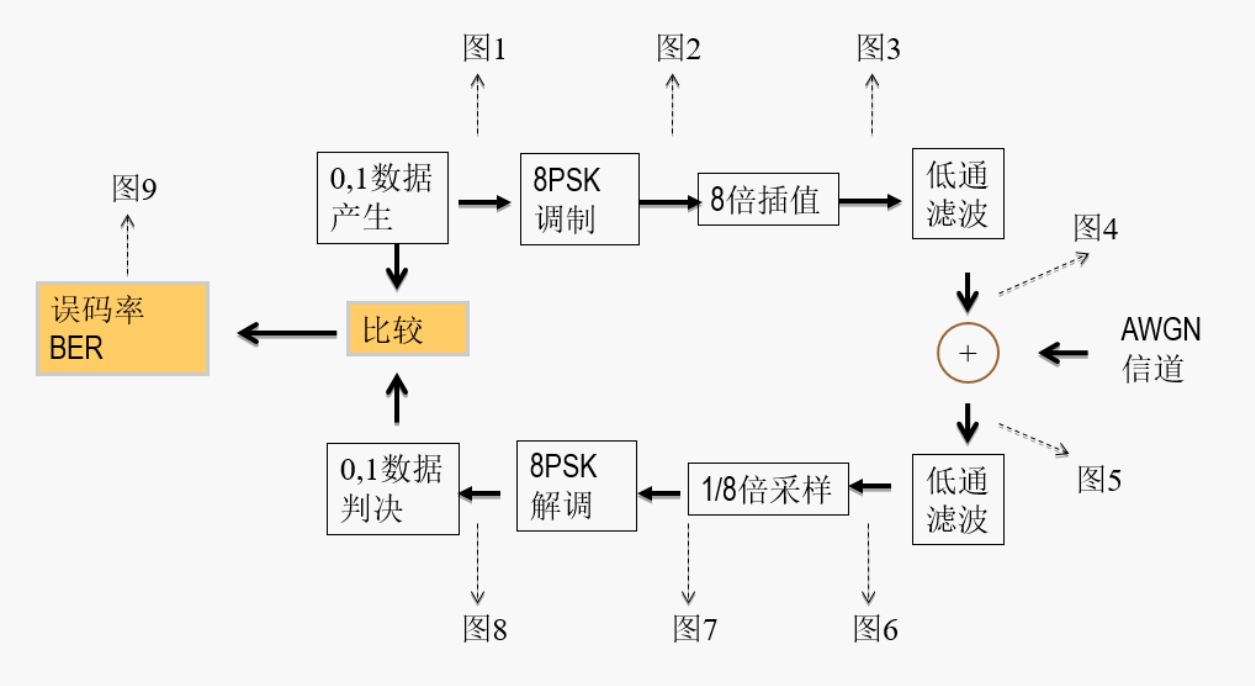
杨凯欣　1500012805　Tel:18811318106

2018 年 6 月

目录

§1 实验要求

1. 通信系统框图如下：

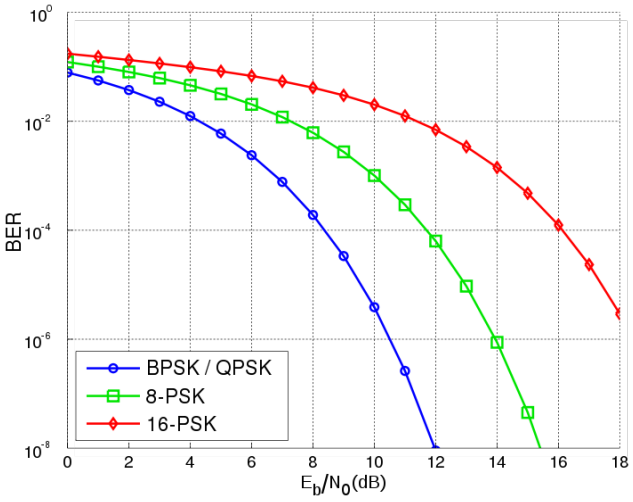


2. 给出原理表达式和原理框图；
3. 给出仿真程序；
4. 给出仿真结果并分析、讨论、比较；
5. 画出理论与仿真误码率曲线；
6. 将自己设计的滤波器同平方根升余弦低通滤波器比较，分析不同。

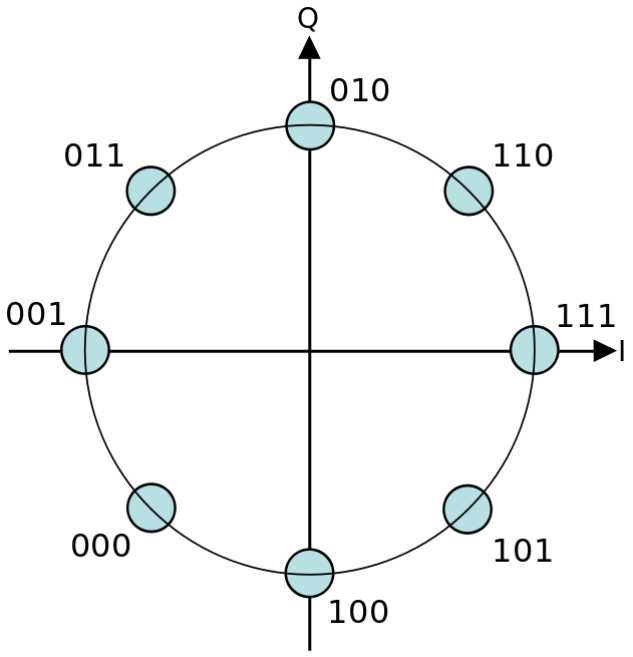
§2 实验原理

2.1 MPSK

1. MPSK - multiple phase shift keying 多进制数字相位调制，又称多相制，是二相制的推广。它是利用载波的多种不同相位状态来表征数字信息的调制方式。多进制数字相位调制也有绝对相位调制（MPSK）和相对相位调制（MDPSK）两种。



2. 8PSK (8 Phase Shift Keying 8 移相键控) 是一种相位调制算法。相位调制（调相）是频率调制（调频）的一种演变，载波的相位被调整用于把数字信息的比特编码到每一词相位改变（相移）。因为 8PSK 拥有 8 种状态，所以 8PSK 每个符号 (symbol) 可以编码 3 个比特 (bits)。8PSK 抗链路恶化的能力（抗噪能力）不如 QPSK，但提供了更高的数据吞吐容量。



2.2 AWGN

$awgn(s_filter, EbN0 - 10 * log10(T), 'measured');$

$$snr(dB) = \frac{E_s}{N_0}(dB) - 10 * lg(Nsamp) = \frac{E_b}{N_0}(dB) + 10 * lg(k) - 10 * lg(Nsamp)$$

Nsamp 为过采样率，k 为每符号比特数。

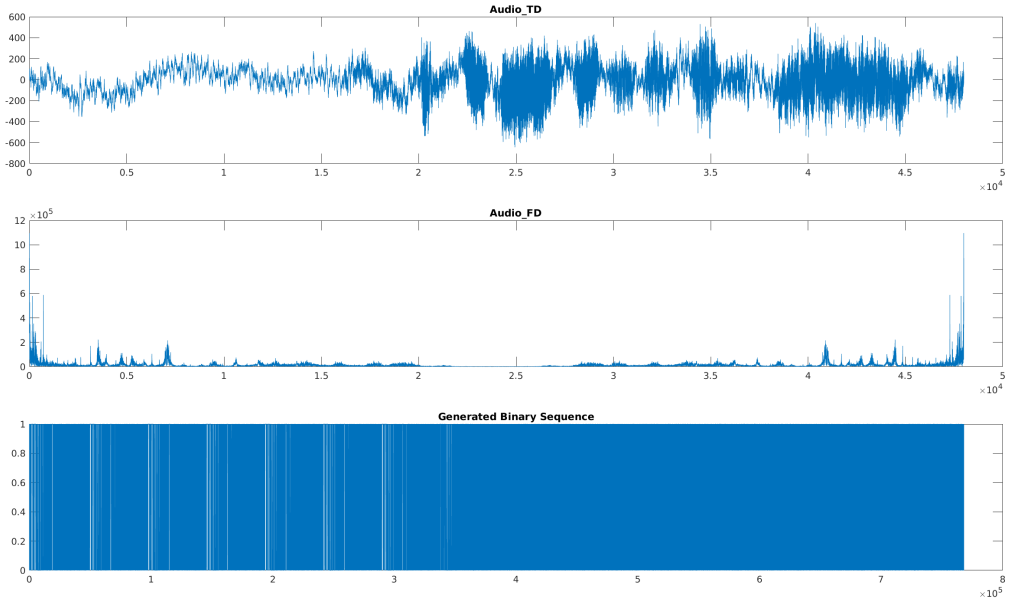
§3 代码及仿真结果

以自己设计的等波纹滤波器，在 SNR=10 处的结果为例：

3.1 0/1 数据产生

```
% Audio Input

clear
load /home/ykx/Audio_Communication_System/ber_standard.mat
M = 8;
K = log2(M);

% sample: datapoints of the audio
% fs: frequency of samplerate
% cnt_point: number of datapoints
% cnt_track: number of tunnels
basepath = '/home/ykx/Audio_Communication_System/';
wav = [basepath 'music.wav'];
[sample, fs] = audioread(wav, 'native');
info = audioinfo(wav);
[cnt_point, cnt_track] = size(sample);

% reshape to represent every symbol with 3 bits
sample = reshape(sample, [], 1);
bits = reshape(double(dectobin(sample, 16)) - 48, 3, []);
Ns = length(bits);
BER = [];
theory = [];
```

```
24
25 % display and save
26 figure(1)
27 subplot(3,1,1)
28 plot(sample)
29  title ('Audio\_TD');
30 subplot(3,1,2)
31 plot(abs( fft (sample)))
32  title ('Audio\_FD');
33 subplot(3,1,3)
34 plot(reshape(bits, 1, []) )
35  title ('Generated Binary Sequence');
36 saveas(1, [basepath 'figure/audio.png']);
```

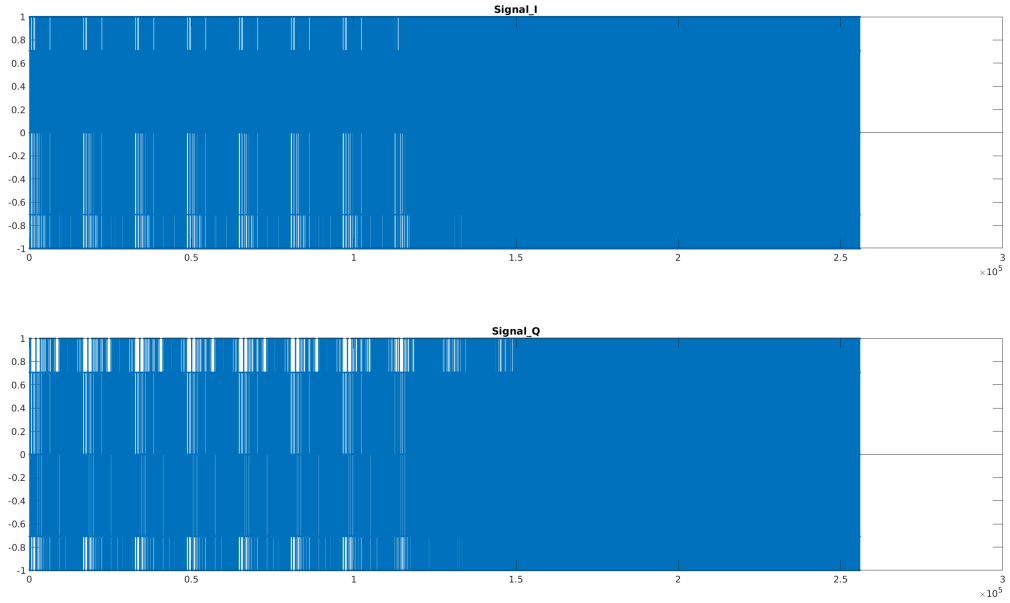


读取`music.wav` 文件，受计算能力的限制，音频文件选取采样率为 8000Hz，量化位数为 16bits，时长为 6s，总点数为 48000，量化后比特数为 768000，以 3bits 一组，构成 3*256000 数组。

上图显示的依次是音频的时域、频域图像，以及量化成比特序列后的图像。

3.2 8PSK 调制

```
% Generate 8PSK Signal

%8PSK symbols
sym_map=[1;(1+1i)/sqrt(2);1i;(-1+1i)/sqrt(2);-1;(-1-1i)/sqrt(2);-1i;(1-1i)/sqrt(2)];

s_mpsk = [];
s_I = [];
s_Q = [];

k = 4 * bits(1, :) + 2 * bits(2, :) + bits(3, :) + 1;
s_mpsk = [s_mpsk sym_map(k)];
s_I = [s_I real(sym_map(k))]; % the I branch of 8PSK
s_Q = [s_Q imag(sym_map(k))]; % the Q branch of 8PSK
s = [s_I s_Q];

% display and save
figure(2)
subplot(2,1,1)
stem(s_I, '.')
title('Signal\_I')
subplot(2,1,2)
stem(s_Q, '.')
title('Signal\_Q')
saveas(2, [basepath 'figure/8psk.png']);
```

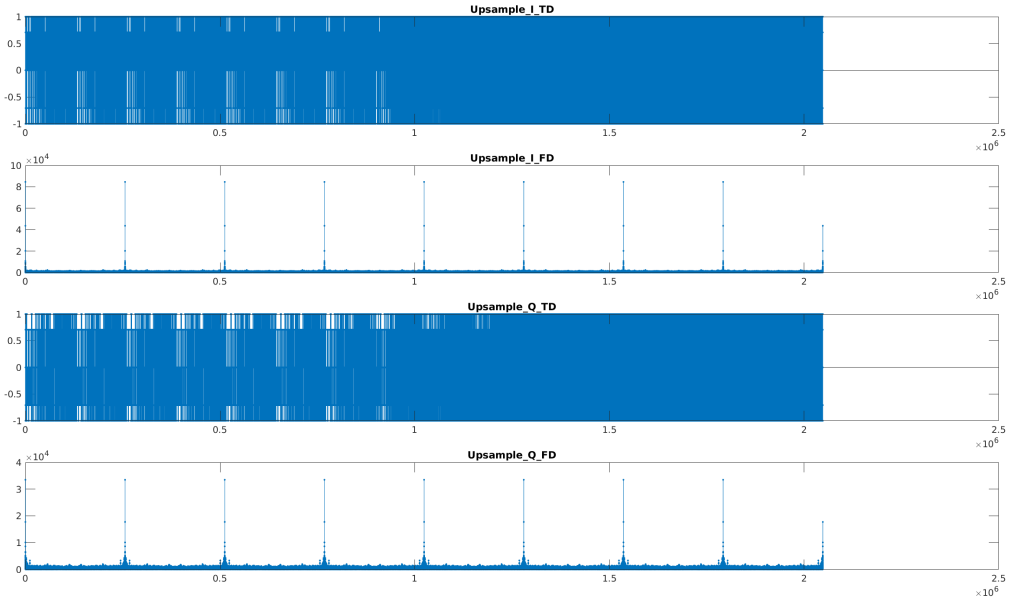


3bits 一组做 8PSK 调制，映射成复数信号 s_mpsk，分离成两路实信号 s_I，s_Q。上图分别显示了 s_I 和 s_Q 的时域波形。

3.3 8 倍插值

```
% Upsample

s_upsample = upsample(s, 8);

% display and save
figure(3)
subplot(4,1,1)
stem(s_upsample(:,1), '.');
 title ('Upsample\_I\_TD')
subplot(4,1,2)
stem(abs(fft(s_upsample(:,1))), '.');
 title ('Upsample\_I\_FD')
subplot(4,1,3)
stem(s_upsample(:,2), '.');
 title ('Upsample\_Q\_TD')
subplot(4,1,4)
```

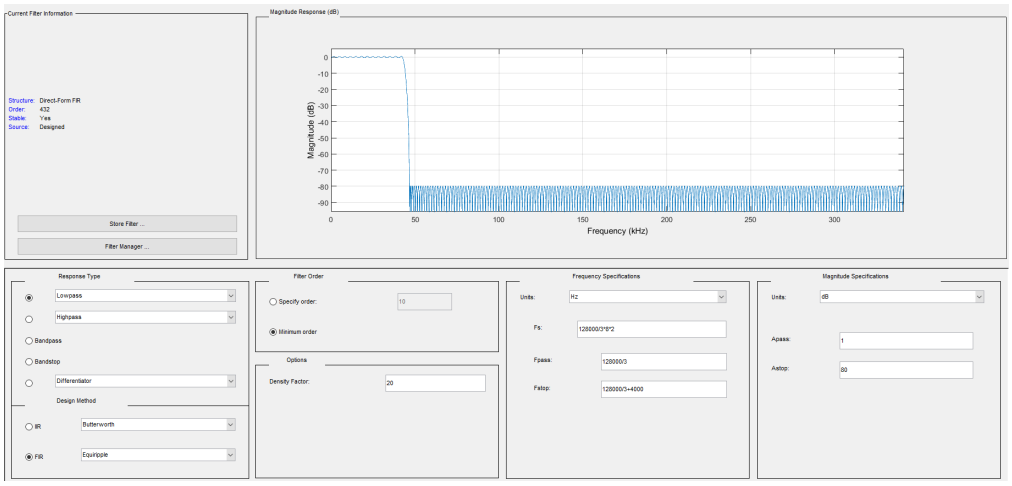
```
17 stem(abs(fft(s_upsample(:,2))), '.');
18  title ('Upsample\_Q\_FD')
19 saveas(3, [basepath 'figure/upsample.png']);
```

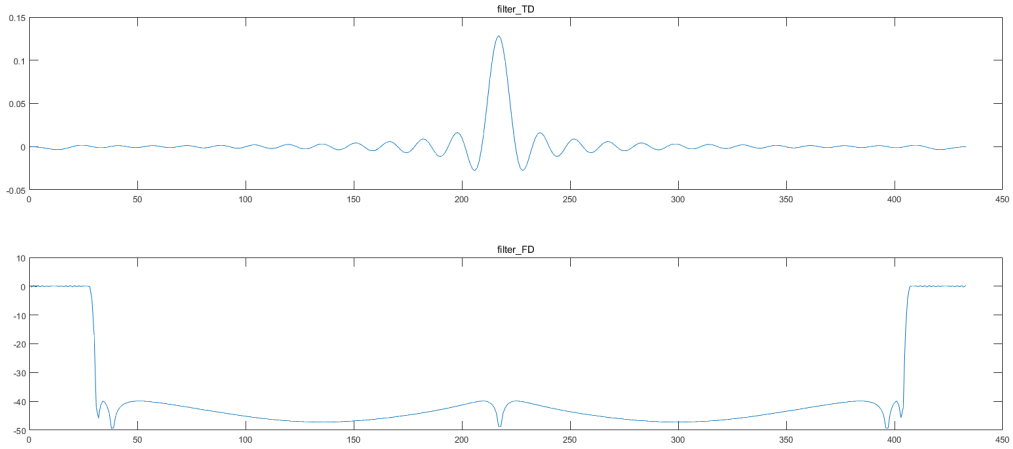


分别对两支路做 8 倍上采样得到时域与频域波形如上，可以看到数字频域在 0 到 2π 范围内以 $2\pi/8$ 的间隔重复 8 次。

3.4 成型滤波

首先设计了一个等波纹滤波器,参数设置上,F_s 选取为 2 倍信号频率,即 $2*128000*8/3 \approx 682666Hz$，通带频率为 $128000/3 \approx 42666Hz$，阻带频率为 $128000/3+4000 \approx 46666Hz$，通带和阻带波纹分别为 1dB 和 80dB。

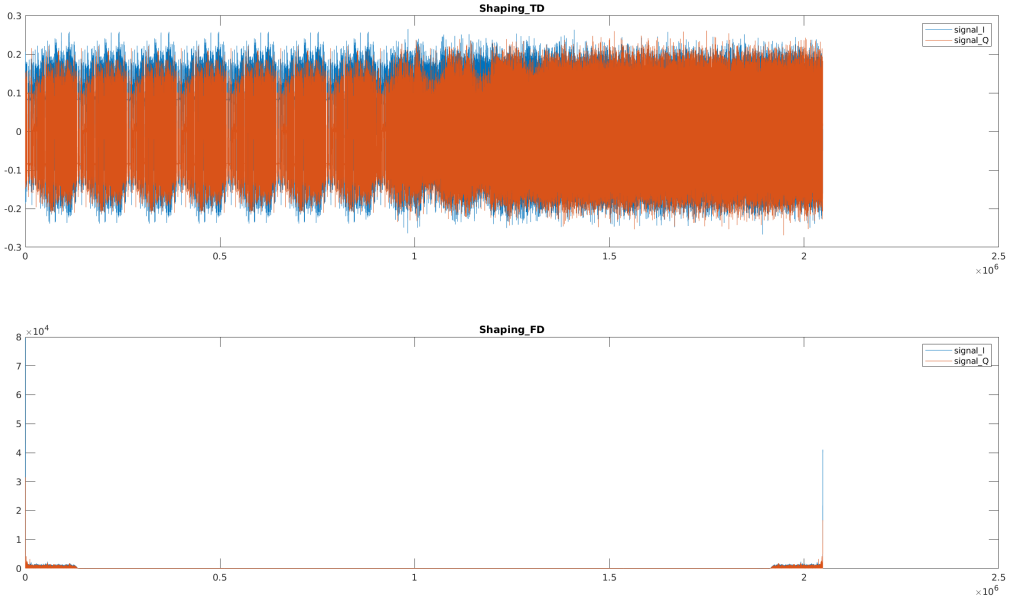




接下来利用该滤波器实现成型滤波：

```
% Shaping Filter

hd = HD;
myfilter  = hd.Numerator;
% add zeros after signals
s_transmit =  filter (myfilter, 1, [s_upsample; zeros((length(myfilter) - 1) / 2, 2)]);
% cut off the prefix zeros
s_transmit = s_transmit(((length(myfilter)-1)/2+1): end, 1:2);

% display and save
figure(4)
subplot(2,1,1)
```

```
13 plot(s_transmit);
14 legend('signal\_I','signal\_Q');
15  title ('Shaping\_TD')
16 subplot(2,1,2)
17 plot(abs(fft(s_transmit)));
18 legend('signal\_I','signal\_Q');
19  title ('Shaping\_FD')
20 saveas(4, [basepath 'figure/shaping.png']);
21
22 % combine two branches
23 s_transmit = s_transmit(:,1) + 1i * s_transmit(:,2) ;
```



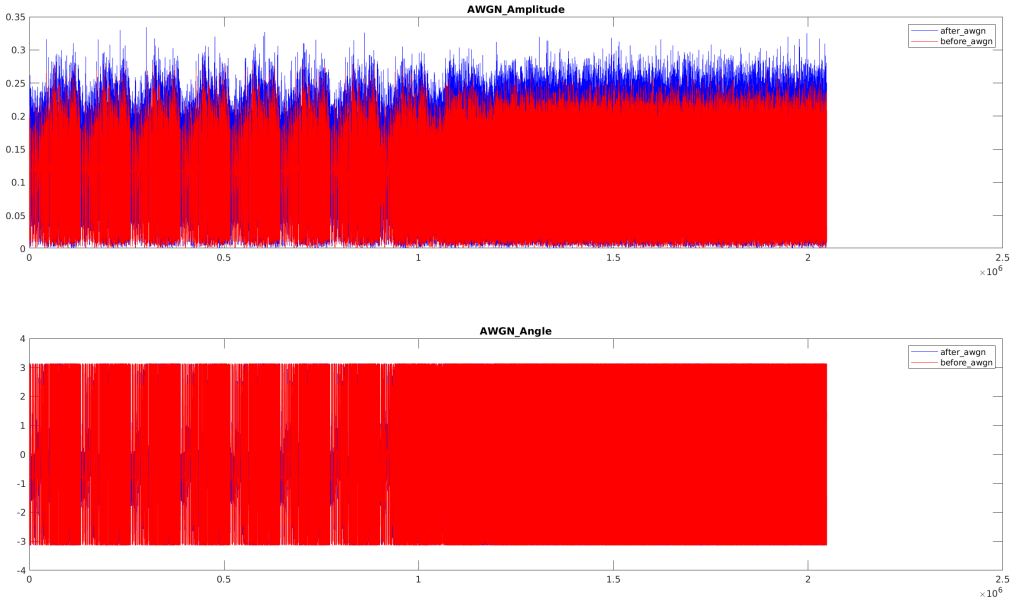
由于数字滤波器存在延时效应，所以在滤波前要先在序列末尾补零，滤波后舍去序列中的前导零。

最后将两支路合并成复数信号，准备通过 AWGN 信道。

3.5 AWGN 信道

```
1 % AWGN Channel
2
```

```
Es_range = 10.^([[-7] [8:1:22]]/10) ; % Energy per symbol
for item = 1:16
    Es = Es_range(item);
    Eb = Es/K;                  % Energy per bit
    N0 = 2;
    SNR = 10 * log10((K * Eb/N0) / 8);
    s_awgn = awgn(s_transmit, SNR, 'measured');

    % display and save
    figure(5)
    subplot(2,1,1)
    plot(abs(s_awgn), 'b');
    hold on;
    plot(abs(s_transmit), 'r');
    legend('after\_awgn','before\_awgn');
     title ('AWGN\_Amplitude');
    hold off;
    subplot(2,1,2)
    plot(angle(s_awgn), 'b');
    hold on;
    plot(angle(s_transmit), 'r');
    legend('after\_awgn','before\_awgn');
     title ('AWGN\_Angle');
    hold off;
    saveas(5, fullfile (basepath, 'figure/', string(item), '/awgn.png'));
```

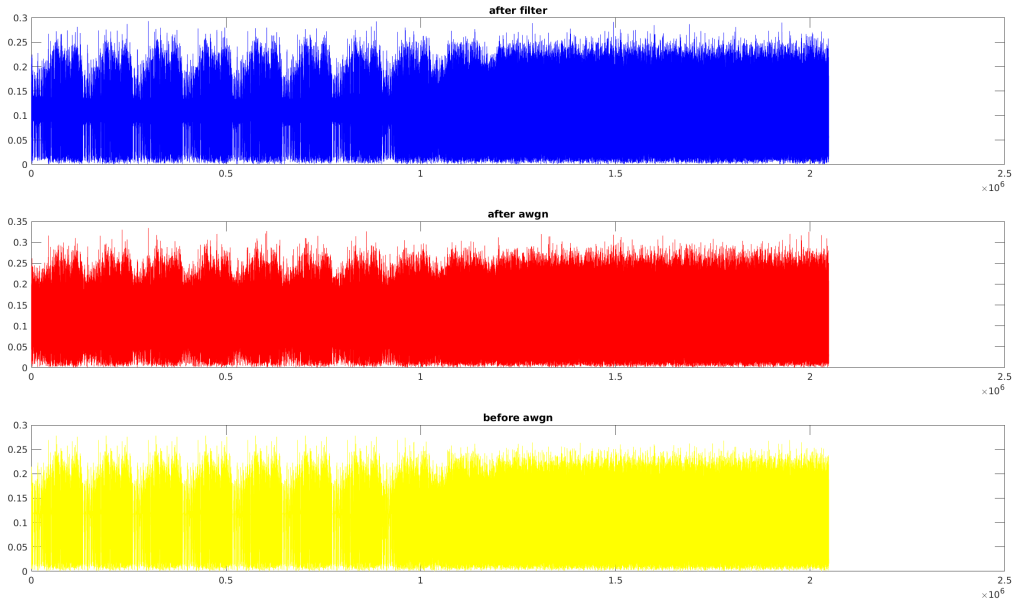


设计一系列的符号功率值，循环以便最终得到 BER-EbN0 图像。从幅值上可以看到通过 AWGN 信道后，有用信号上附加了一定的噪声信号。

3.6 低通滤波

```
% Match Filter

s_receive = filter(myfilter, 1, [s_awgn; zeros((length(myfilter) - 1) / 2, 1)]);
s_receive = s_receive(((length(myfilter)-1)/2+1): end, 1);

% display and save
figure(6)
subplot(3,1,1)
plot(abs(s_receive), 'b');
 title('after filter');
subplot(3,1,2)
plot(abs(s_awgn), 'r');
 title('after awgn');
subplot(3,1,3)
plot(abs(s_transmit), 'y');
 title('before awgn');
```

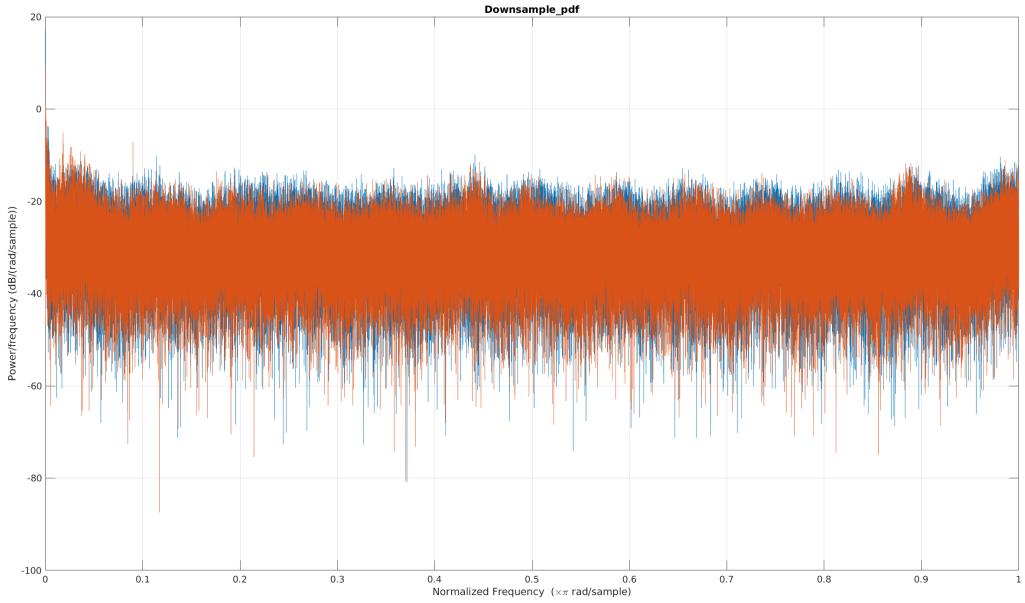
```
17      saveas(6, fullfile(basepath,'figure/',string(item),'/matchedfilter.png'));
```



做一次低通滤波，由三条时域图像可以粗略看到，低通滤波后的波形更接近通过 AWGN 信道前的波形，部分噪声被滤除。

3.7 1/8 下采样

```
1       % Downsample
2
3       s_downsample = downsample(s_receive, 8);
4       % divide into two branches
5       s_downsample = [real(s_downsample) imag(s_downsample)];
6
7       %display and save
8       figure(7)
9       periodogram(s_downsample);
10       title('Downsample\_pdf');
11      saveas(7, fullfile(basepath,'figure/',string(item),'/downsample.png'));
```

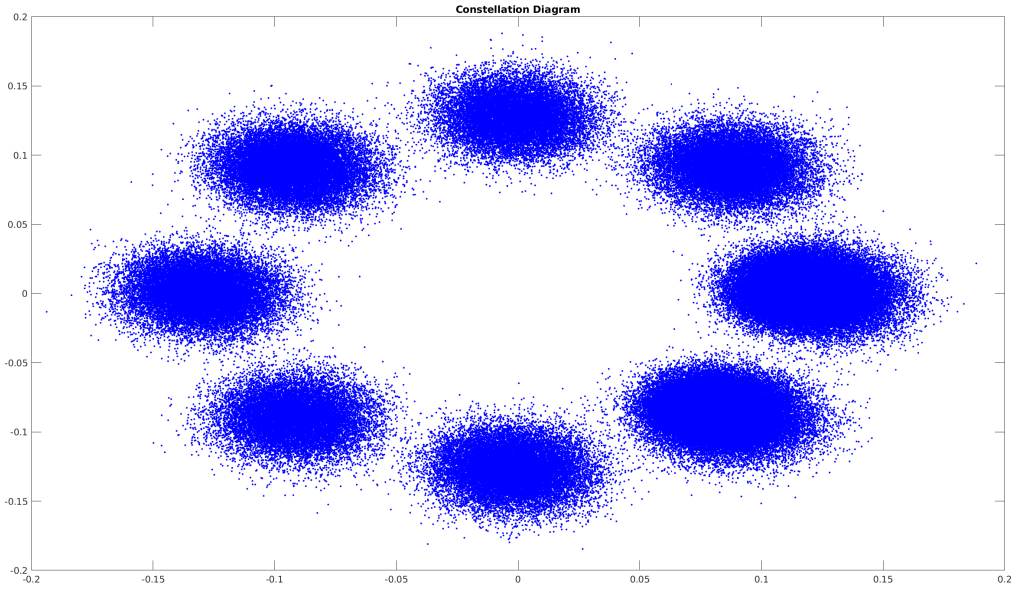


做 1/8 下采样，并将复数信号按照实部和虚部分离成两支路，上图为两支路的功率谱密度函数图像。

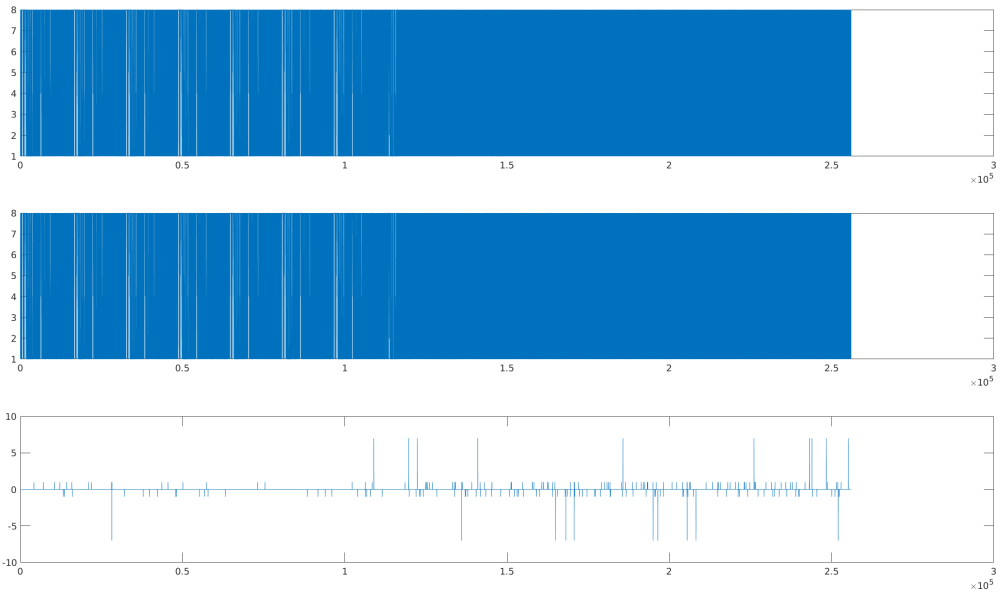
3.8 8PSK 解调及判决

```
% 8PSK Judgement

s_demodulate = s_downsample;

% display and save
figure(8)
plot(s_demodulate(:,1), s_demodulate(:,2), 'b.')
 title ('Constellation Diagram');
saveas(8, fullfile (basepath, 'figure/', string(item), '/constellation.png'));

% calculate the distance to 8 phase points
s_result = s_demodulate(:,1) - 1i * s_demodulate(:,2);
s_result = s_result';
distance = abs(repmat(s_result, [M, 1]) - repmat(sym_map, [1, Ns]));

% get the minimun distance and the order
```

```
17      [min_dis, min_pos] = min(uint32(distance .* 10000));
18
19      % display and save
20      figure(9)
21      subplot(3,1,1)
22      plot(k)
23      subplot(3,1,2)
24      plot(min_pos)
25      subplot(3,1,3)
26      plot(k-min_pos)
27      saveas(9, fullfile(basepath,'figure/',string(item),'/error.png'));
28
29      % demodulate from symbols to bits sequence
30      min_pos = min_pos - 1;
31      bits_result = [];
32      bits_result = [bits_result sign(bitand(min_pos,4))] ;
33      bits_result = [bits_result; sign(bitand(min_pos,2))];
34      bits_result = [bits_result; mod(min_pos,2)];
35
36      % display and save
37      figure(11)
38      subplot(2,1,1)
39      plot(reshape(bits, 1, []),'b');
40      title('transmitted\_bits');
41      subplot(2,1,2)
42      plot(reshape(bits_result, 1, []),'r');
43      title('received\_bits');
44      saveas(11, fullfile(basepath,'figure/',string(item),'/bits.png'));
```



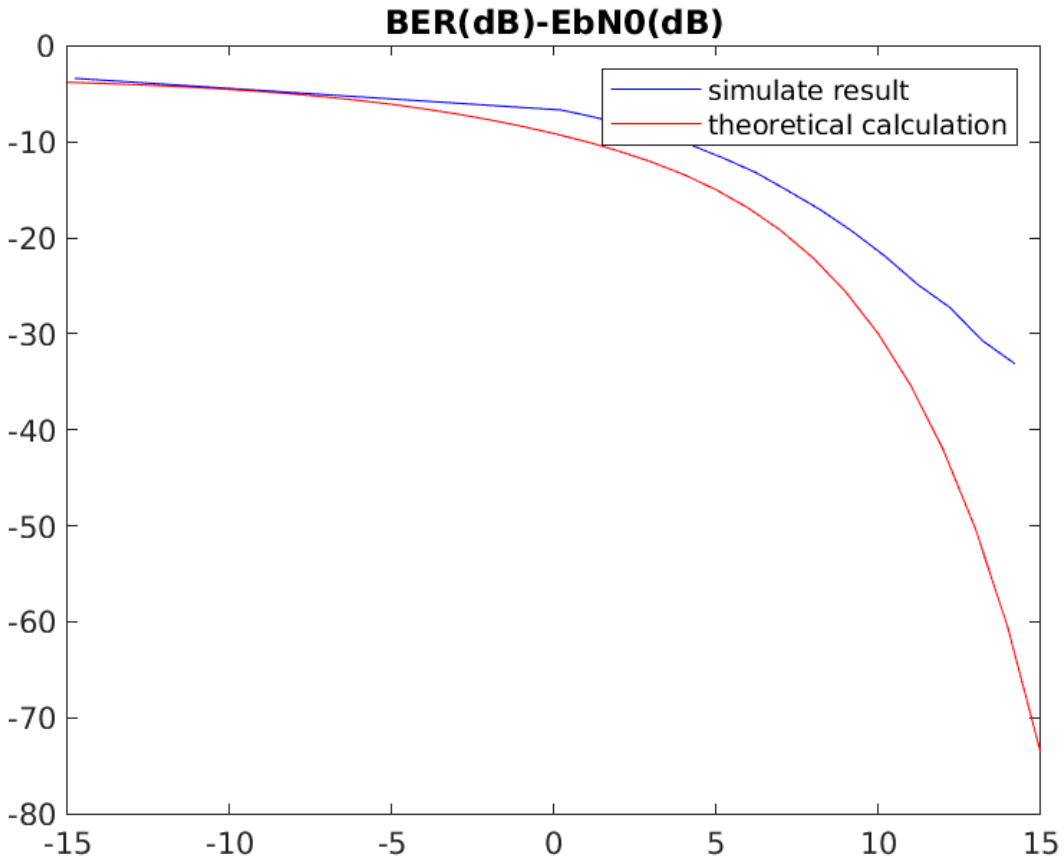
符号判决采用距离比较的方法，将接收到的各个符号同 8PSK 的 8 个相位点求距离，取距离最小的点作为判决结果，得到星座图。由于是 SNR=10 时的图像，可以看到星座图较为明显地分为了 8 个密集区，对应 8PSK 的 8 个相位点。



同时做出发送比特序列和接收比特序列的图像，由于过于密集不够直观，故做出第三条图像，为二者差值，可以看到存在少量的错判。

3.9 误码率计算

```
    % Calculate

    bits_error = abs(bits_result - bits);
    BER = [BER biterr(bits, bits_result)];

end

% display and save
figure(10)
hold off;
BE = BER;
BER = BER / (3 * length(bits));
plot(10 * log10(Es_range/(K * N0)), 10 * log10(BER), 'b');
hold on;
% plot the theoretical curve
plot(bers0.data{1,1}, 10 * log10(bers0.data{1,2}), 'r');
 title ('BER(dB)-EbN0(dB)')
legend('simulate result','theoretical calculation');
saveas(10, [basepath 'figure/BER-EBN0.png']);
```



最后的误码率曲线采用对数坐标，可以看到当 EbN0 较低时，仿真结果与理论曲线吻合较好，但当 EbN0 大于 10dB 时，两条曲线有数量级的差距，仿真结果明显差于理想曲线。

结合滤波器的时域、频域曲线分析可知，在这种设计下的滤波器存在较强的符号间干扰，随着噪声的逐渐减小，符号间干扰的作用越来越明显，显著影响到判决结果。

§4 比较与分析

以上是对整个流程的简单介绍，下面针对不同信噪比、不同滤波器得到的结果进行比较和分析。

4.1 信噪比的影响

由星座图可以比较明显直观地看到 SNR 对于信道性能的影响。

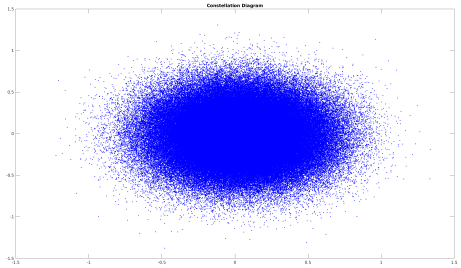


Figure 1: SNR=-19dB

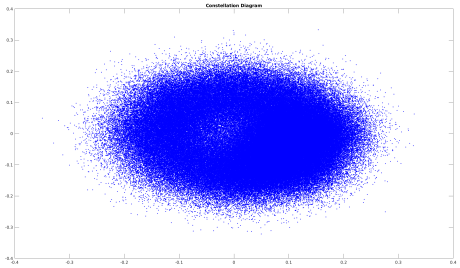


Figure 2: SNR=-4dB

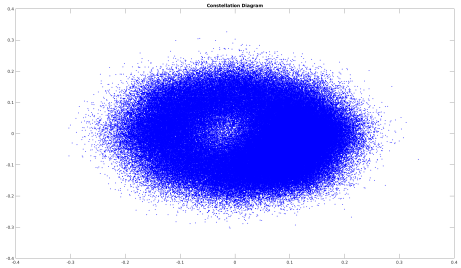


Figure 3: SNR=-3dB

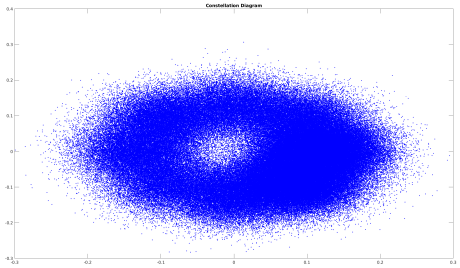


Figure 4: SNR=-2dB

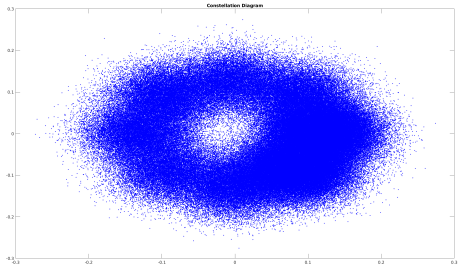


Figure 5: SNR=-1dB

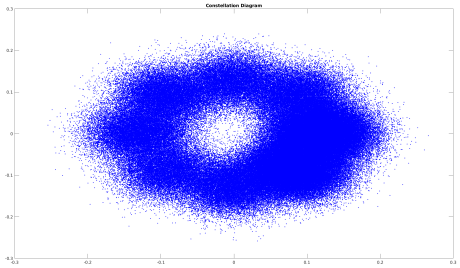


Figure 6: SNR=0dB

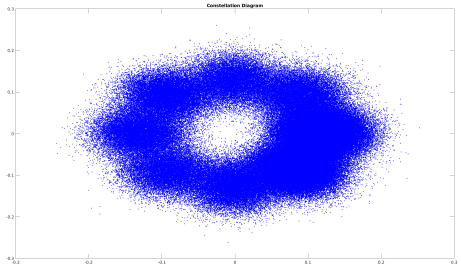


Figure 7: SNR=1dB

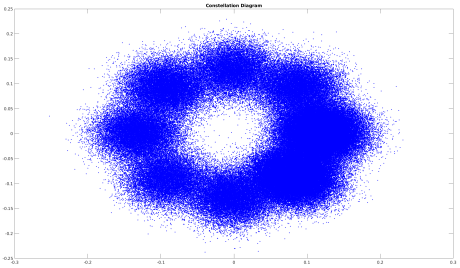


Figure 8: SNR=2dB

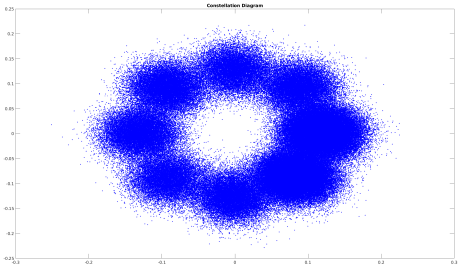


Figure 9: SNR=3dB

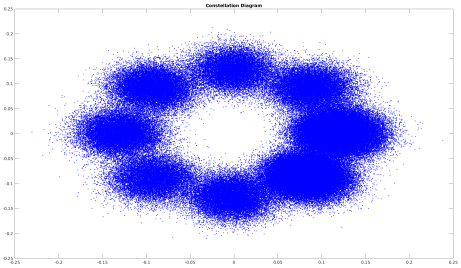


Figure 10: SNR=4dB

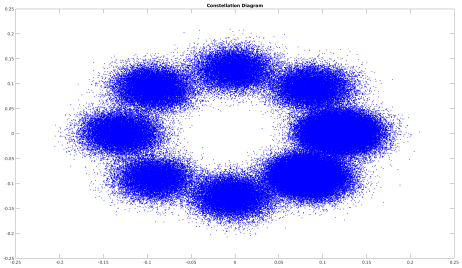


Figure 11: SNR=5dB

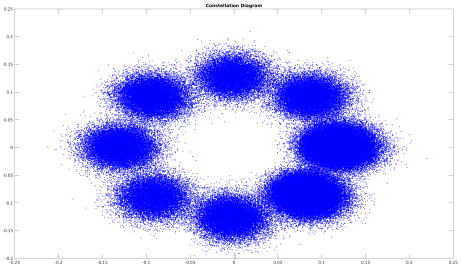


Figure 12: SNR=6dB

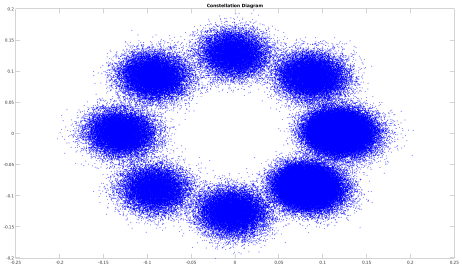


Figure 13: SNR=7dB

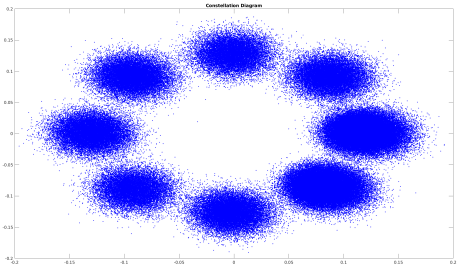


Figure 14: SNR=8dB

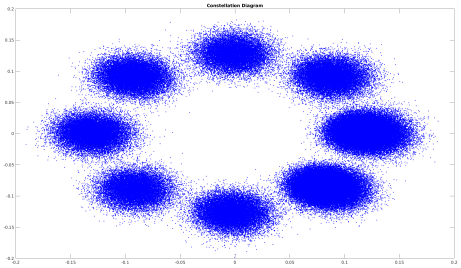


Figure 15: SNR=9dB

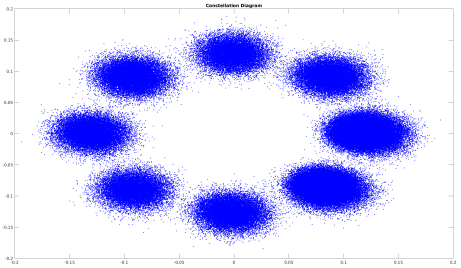
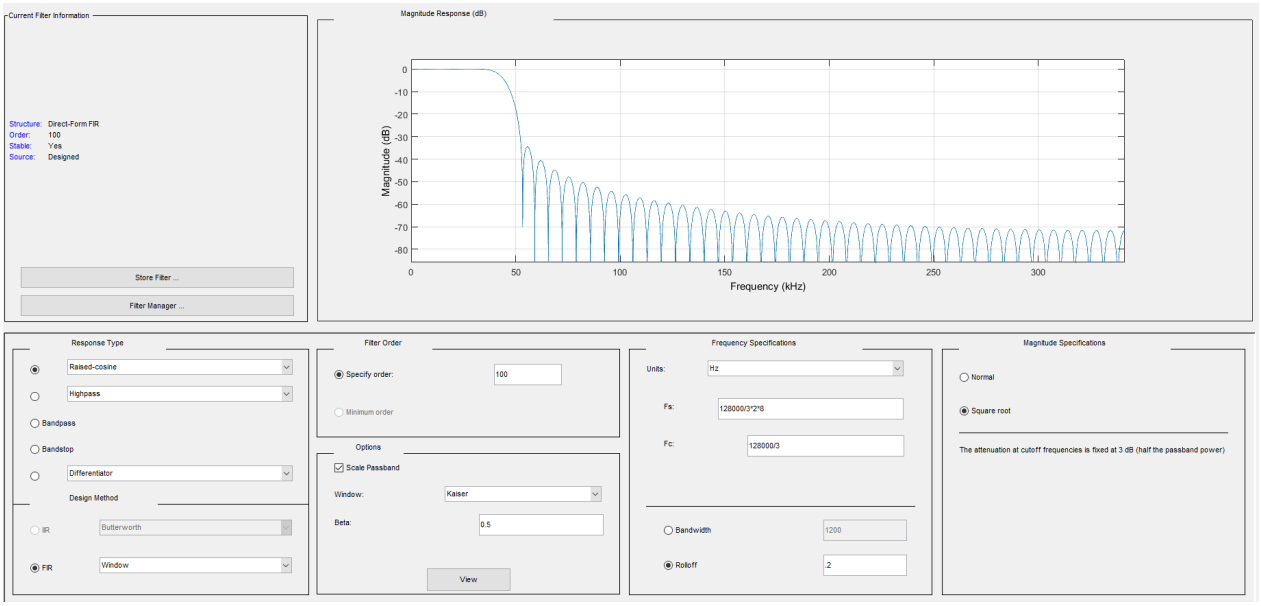


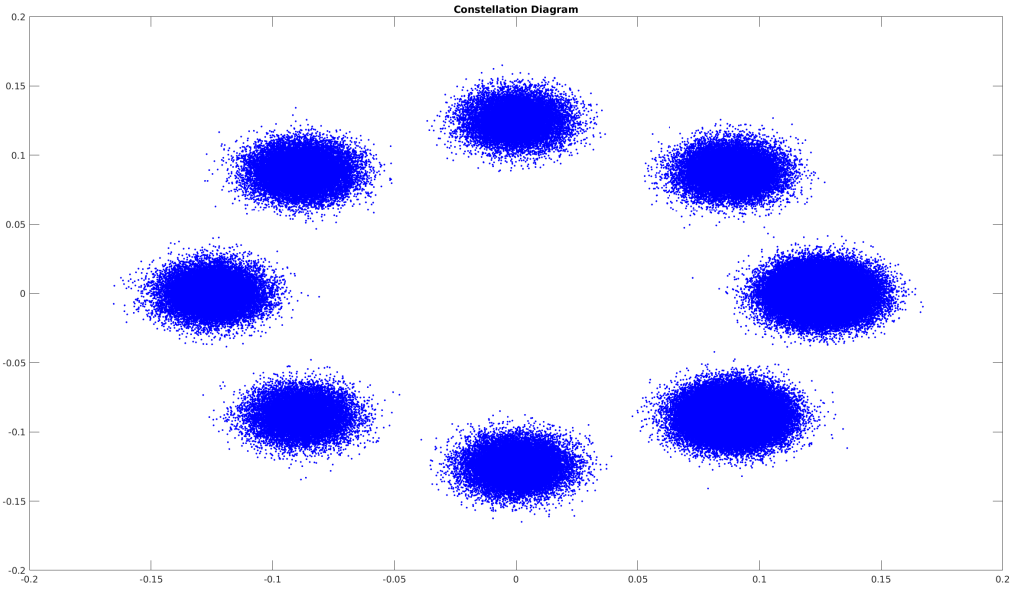
Figure 16: SNR=10dB

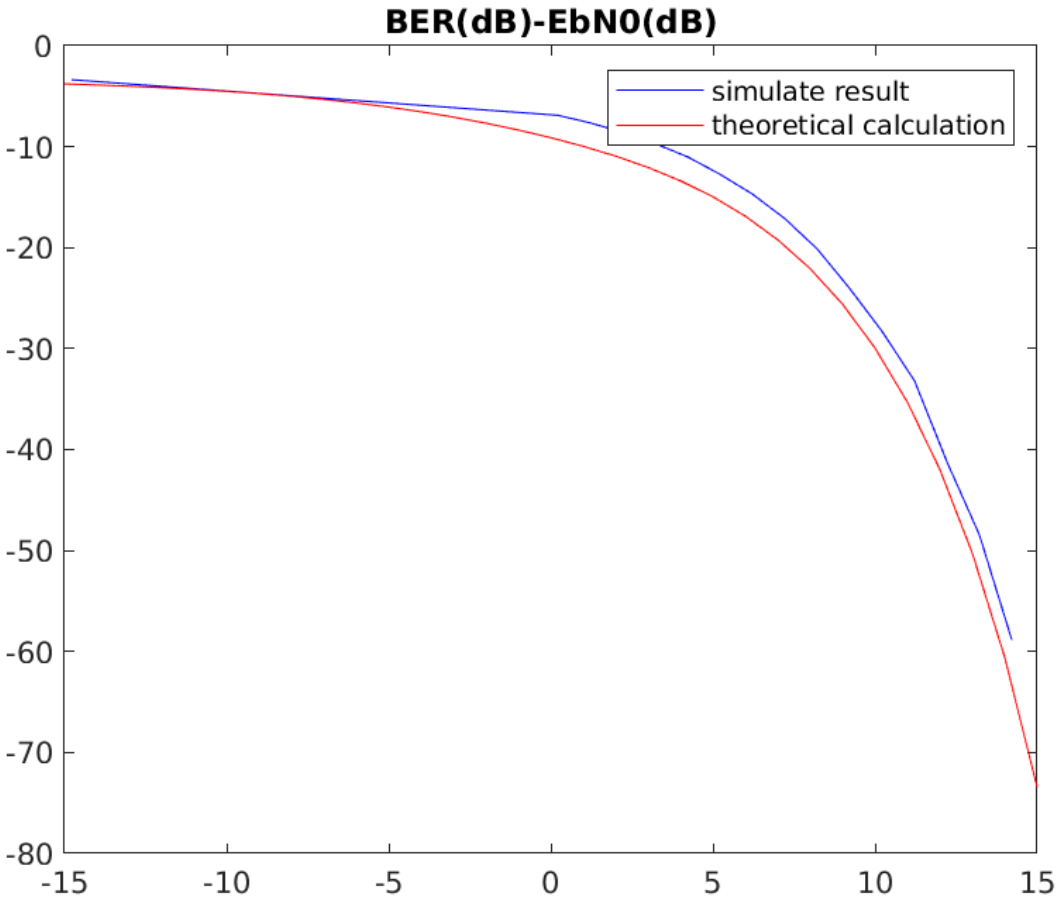
可以看到随着 SNR 的提升，星座图 8 个相位点逐渐清晰并彼此分离，表示此时判决的误码率不断下降。

4.2 滤波器的影响

使用等波纹方法设计的低通滤波器在 EbN0 较高时性能较差，与理想曲线相距较远，故尝试使用平方根升余弦滤波器，参数设置为凯泽窗，Beta=0.5，阶数为 100 阶，$F_s = 128000/3 * 8 * 2 \approx 682666Hz, F_c = 128000/3 \approx 42666Hz$，参数设计、得到的星座图及最后的误码率曲线如下：




Constellation Diagram
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.2
-0.15
-0.1
-0.05
0
0.05
0.1
0.15
0.2


BER(dB)-EbN0(dB)
simulate result
theoretical calculation
0
-10
-20
-30
-40
-50
-60
-70
-80
-15
-10
-5
0
5
10
15

可以看到升余弦滤波器星座图得到的散点更集中于 8 个相位点附近，误码率与理想曲线吻合更好，说明平方根升余弦滤波器在解决符号间干扰的问题上性能突出。

§5 附 0：辅助代码

5.1 dectobin.m

实现十进制整数（包含正负）到二进制的转化。

```
function bin = dectobin(dec, bit)
    bin = dec2bin(abs(dec), bit);
    for j = 1:length(dec)
        if dec(j,1) < 0
            for i = 1:bit
                if bin(j, i) == 48
                    bin(j, i) = 49;
                else
                    bin(j, i) = 48;
                end
            end
            bin(j, bit) = bin(j, bit) + 1;
            for i = bit:-1:2
                if bin(j, i) == 50
                    bin(j, i - 1) = bin(j, i - 1) + 1;
                    bin(j, i) = 48;
                end
            end
            if bin(j, 1) == 50
               bin(j, 1) = 48;
            end
        end
    end
end
```

5.2 HD.m

自己设计的等波纹滤波器参数内容。

```
function Hd = HD
%HD Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.4 and DSP System Toolbox 9.6.
% Generated on: 20-Jun-2018 00:42:54

% Equiripple Lowpass filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 682666.66667;  % Sampling Frequency

Fpass = 42666.666667;     % Passband Frequency
Fstop = 46666.666667;     % Stopband Frequency
Dpass = 0.057501127785; % Passband Ripple
Dstop = 0.0001;           % Stopband Attenuation
dens  = 20;               % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fpass, Fstop]/(Fs/2), [1 0], [Dpass, Dstop]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

% [EOF]
```

5.3 SRC.m

自己设计的平方根升余弦滤波器参数内容。

```
function Hd = SRC
%SRC Returns a discrete-time filter object.
```

```
3
4  % MATLAB Code
5  % Generated by MATLAB(R) 9.4 and DSP System Toolbox 9.6.
6  % Generated on: 19-Jun-2018 23:52:49
7
8  % FIR Window Raised-cosine filter designed using the FIRRCOS function.
9
10 % All frequency values are in Hz.
11 Fs = 682666.66667;  % Sampling Frequency
12
13 N    = 100;          % Order
14 Fc   = 42666.666667; % Cutoff Frequency
15 TM   = 'Rolloff';    % Transition Mode
16 R    = 0.2;          % Rolloff
17 DT   = 'sqrt';       % Design Type
18 Beta = 0.5;          % Window Parameter
19
20 % Create the window vector for the design algorithm.
21 win = kaiser(N+1, Beta);
22
23 % Calculate the coefficients using the FIR1 function.
24 b  = firrcos(N, Fc/(Fs/2), R, 2, TM, DT, [], win);
25 Hd = dfilt.dffir(b);
26
27 % [EOF]
```

5.4 ber_standard.mat

8PSK 理想误码率曲线数据内容。