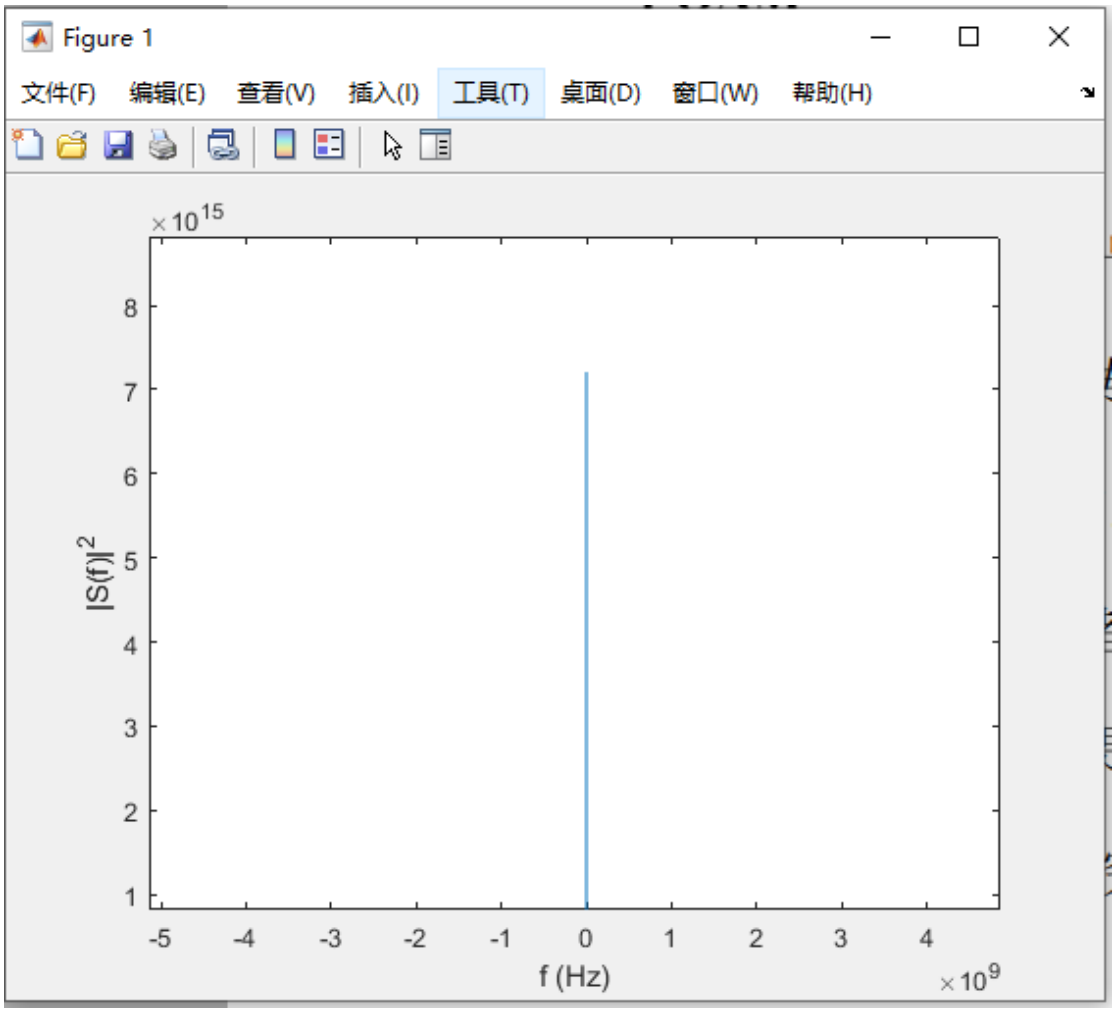
信号与系统实践任务三报告

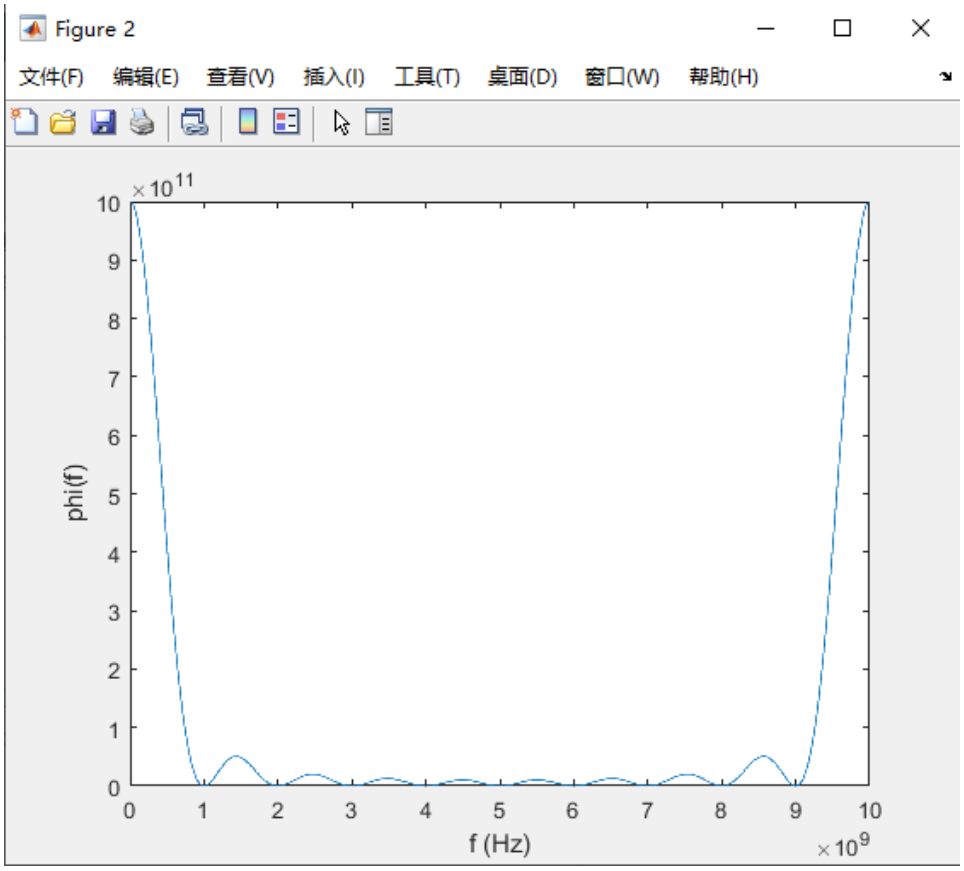
刘正浩 2019270103005

1. 数字调制信号的频谱分析

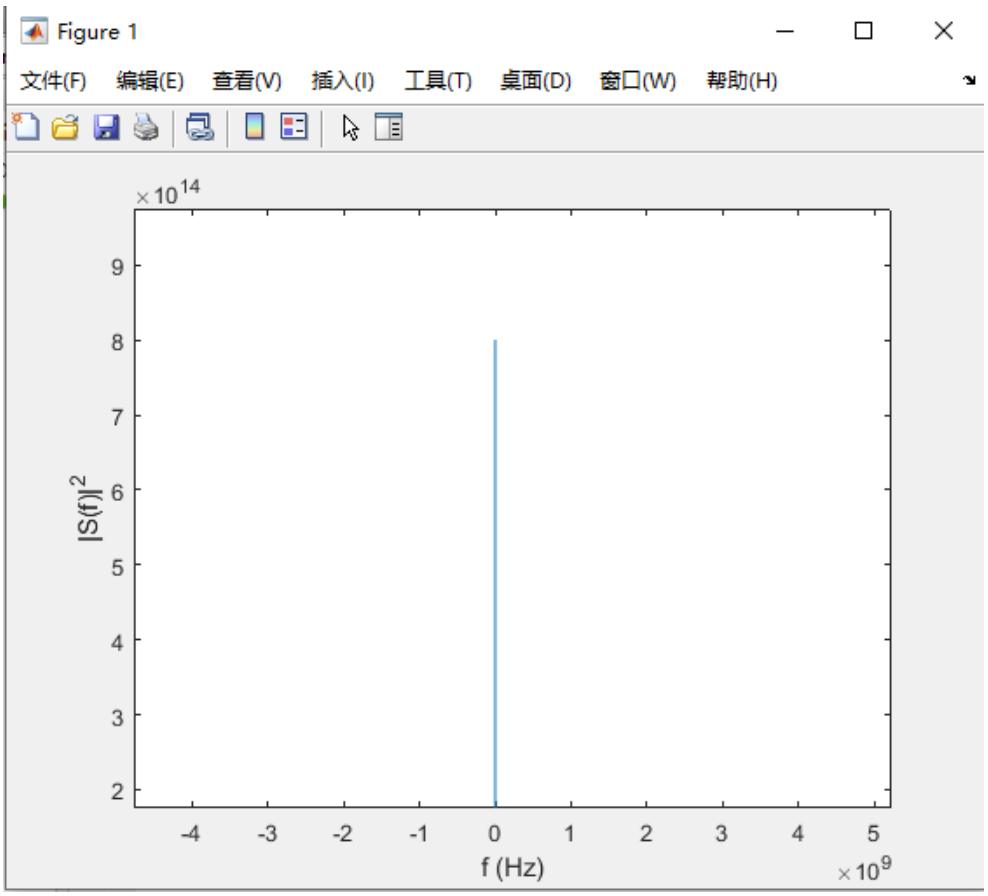
本任务的目的在于，假设一个通信系统使用宽度为 T=1ns，幅度归一化为 1 的方波信号 g(t)，对应的符号传输速率为 1G 波特率。随机生成 4-QAM 和 16-QAM 的I_n序列，在 MATLAB 中对基带信号 s(t)进行傅里叶变换得到 S(f)，分别绘图$|S(f)|^2$和$\Phi_S(f)$，观察两者的异同。画出的 16-QAM$|S(f)|^2$图为



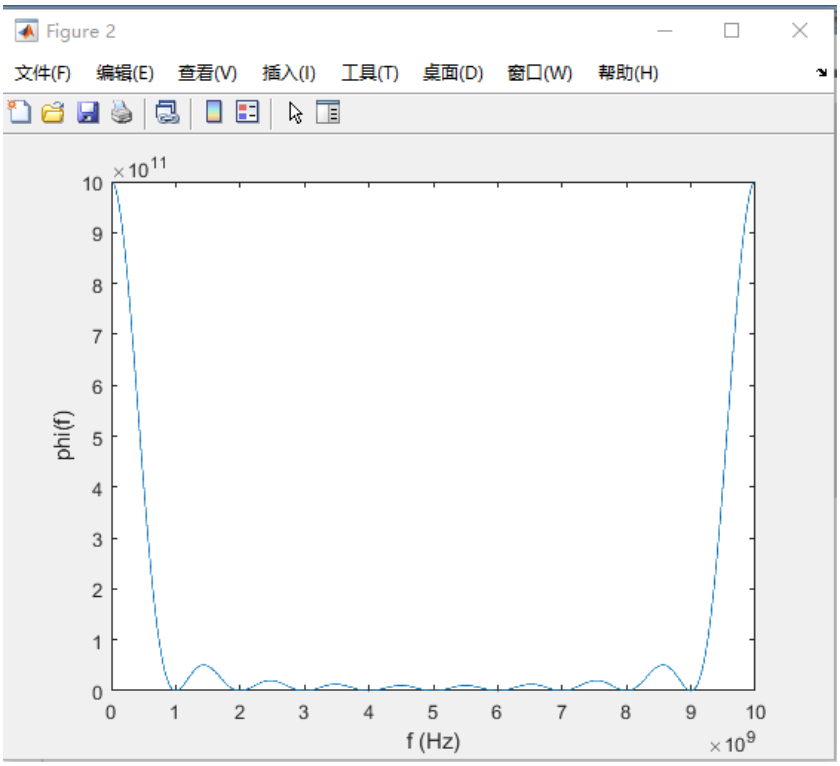
$\Phi_S(f)$图为



画出的 4-QAM$|S(f)|^2$图为



$\Phi_s(f)$图为



附：源码

```
%数字调制信号的频谱分析

fs = 10^10;
t = -10^-6:1/fs:10^-6;
w = 10^-9;
delay = 0.5 * w;

%产生矩形脉冲
gt = rectpuls(t-delay,w);
M = 16; %填 4 或 16 来选择 4-QAM 或 16-QAM
gt_original = gt; %存储下原始的脉冲信号
st = qammod(gt,M);

%产生冲激串信号
for i = 1:1:999
  st=st+qammod(rectpuls(t-delay-i*w,w),M);
end
L_1 = length(st); %信号的总时长
L_2 = length(gt_original);
```

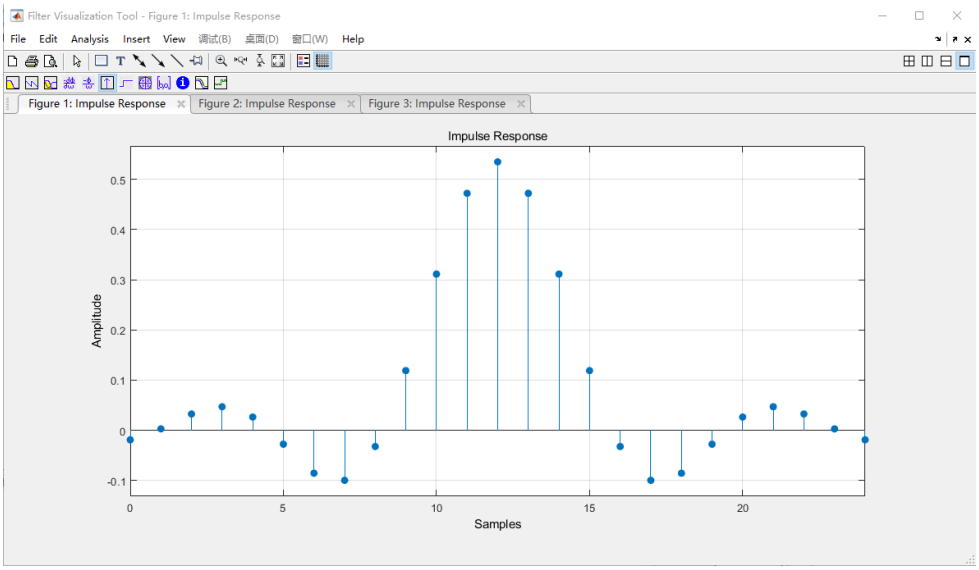
```
Sf = fft(st);
y_SF = (abs(Sf)).^2;
f_1 = (0:L_1-1)*fs/L_1;

%画出|S(f)|^2
figure(1)
plot(f_1,y_SF)
xlabel('f (Hz)')
ylabel('|S(f)|^2')

%画出Phi(f)
Gf = fft(gt_original);
y_PhiF = (10^10)*(abs(Gf)).^2;
f_2 = (0:L_2-1)*fs/L_2;
figure(2)
plot(f_2,y_PhiF)
xlabel('f (Hz)')
ylabel('phi(f)')
```

2. 奈奎斯特准则

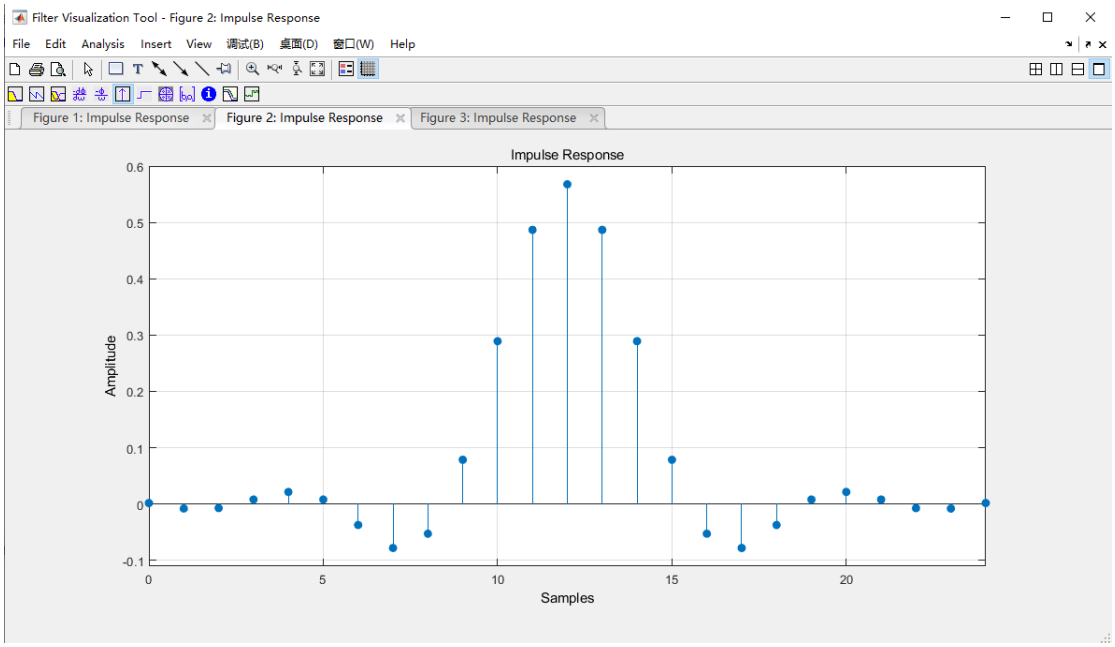
本任务的目的是，在 MATLAB 中产生不同滚降系数的升余弦脉冲，画出其时域波形、频域频谱，并与升余弦函数的正确波形、频谱比较正确性进行验证。在仿真过程中，使用函数 rcosdesign()产生离散采样后的升余弦脉冲。

仿真结果如下：

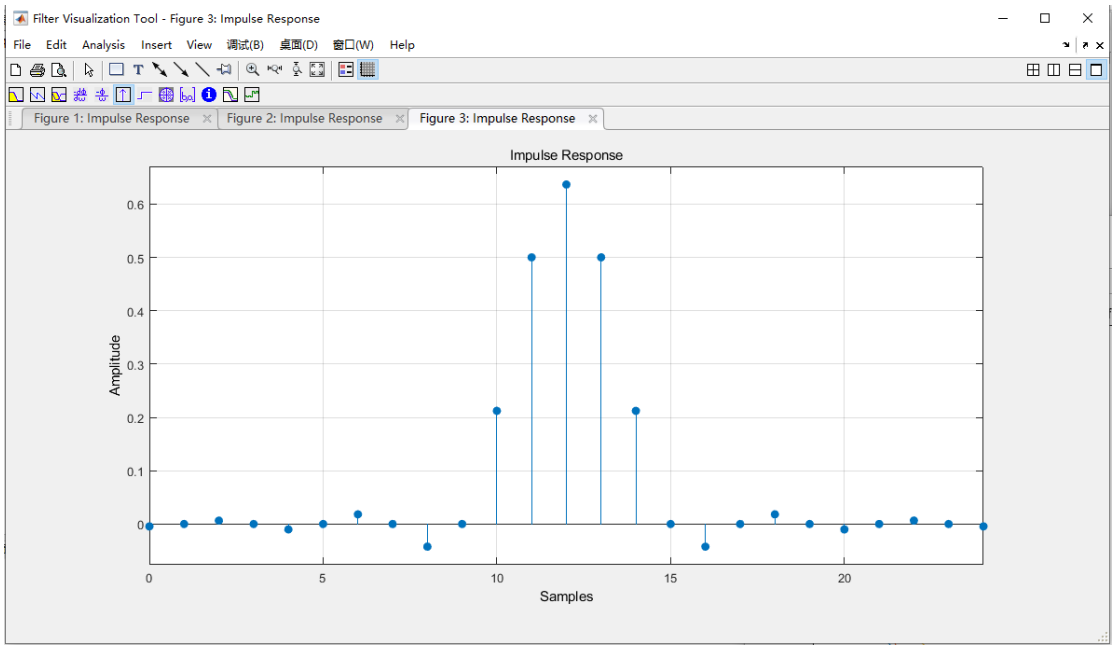
滚降系数为 0.25 的升余弦函数



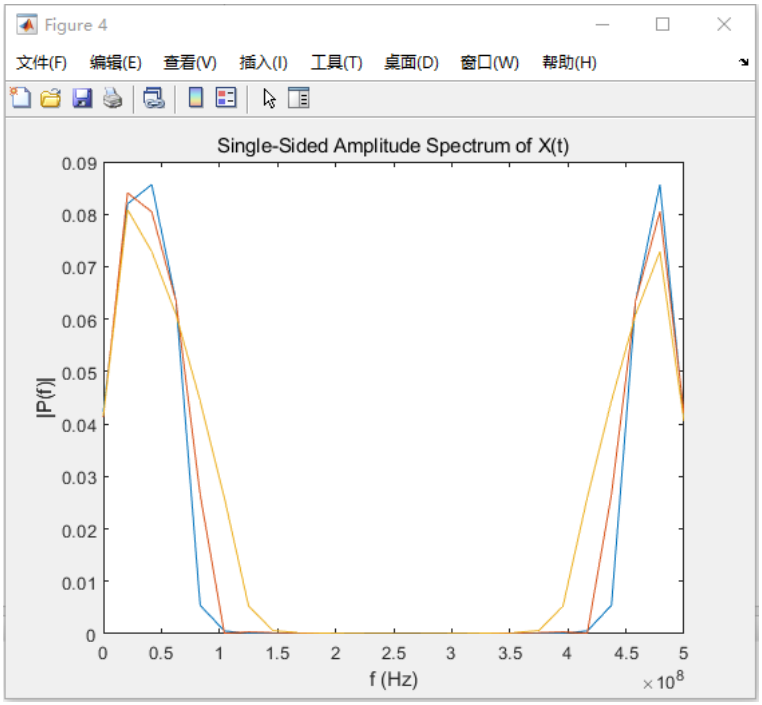
滚降系数为 0.5 的升余弦函数：



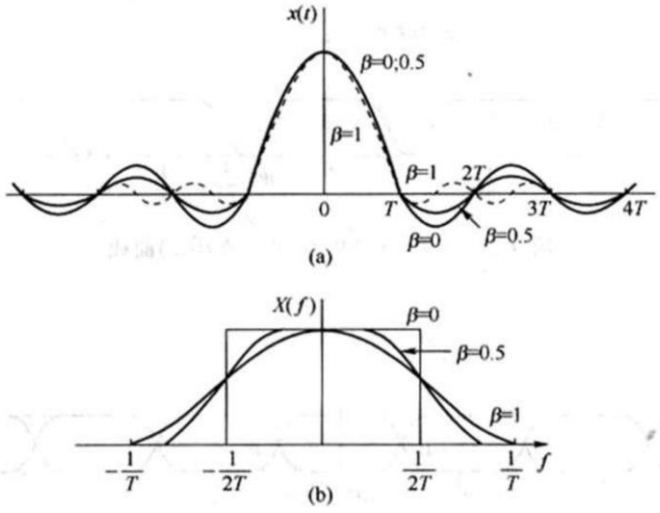
滚降系数为 1 的升余弦函数



三个升余弦函数的频谱



经过对比，与标准的升余弦函数形状和频谱（下图）基本一致。



附：源码

```
%奈奎斯特准则

Fs = 10^9;  % 采样频率
T = 1/Fs;  %采样周期
L = 48;  %信号长度
t = (0:L-1)*T;  %时间

%生成三个滚降系数不同的升余弦函数
h_1 = rcosdesign(0.25,6,4);  %滚降系数为0.25
fvtool(h_1,'Analysis','impulse');
```

```
h_2 = rcosdesign(0.5,6,4);  %滚降系数为0.5
fvtool(h_2,'Analysis','impulse');
h_3 = rcosdesign(1,6,4);  %滚降系数为1
fvtool(h_3,'Analysis','impulse');

%画h_1 的频谱
Y_1 = fft(h_1);
P2_1 = abs(Y_1/L);
P1_1 = P2_1(1:L/2+1);
P1_1(2:end-1) = 2*P1_1(2:end-1);
f = Fs*(0:(L/2))/L;
plot(f,P1_1)
title('Single-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)')
hold on

%画h_2 的频谱
Y_2 = fft(h_2);
P2_2 = abs(Y_2/L);
P1_2 = P2_2(1:L/2+1);
P1_2(2:end-1) = 2*P1_2(2:end-1);
f = Fs*(0:(L/2))/L;
plot(f,P1_2)
title('Single-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)')
hold on

%画h_3 的频谱
Y_3 = fft(h_3);
P2_3 = abs(Y_3/L);
P1_3 = P2_3(1:L/2+1);
P1_3(2:end-1) = 2*P1_3(2:end-1);
f = Fs*(0:(L/2))/L;
plot(f,P1_3)
title('Single-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)')
ylabel('|P(f)|')
hold on
```

3. 基于 RRC 脉冲的数字调制信号的性能仿真和频谱分析

本任务的目的是，使用 RRC 脉冲完成 4-QAM 和 16-QAM 的数字通信系统的性能仿真，并证明误码性能与 RRC 脉冲的滚降系数无关，只与 SNR 有关。

在验证方面，我找到了专门用于验证的源码，但由于水平有限并没有完全看懂。以下是源码：

```
%基于RRC 脉冲的数字调制信号的性能仿真和频谱分析
%以下均为参考代码

T_start=0;%开始时间
T_stop=10;%截止时间
T=T_stop-T_start;%仿真持续时间
T_sample=1/1000;%采样间隔
f_sample=1/T_sample;% 采样速率
N_sample=T/T_sample;% 采样点数
n=0:N_sample-1;
r_s=100;%码元传输速率
alpha=0.25;%df=alpha*rs=25Hz
NumBits=T*r_s;%传输符号个数
NumCoff=10;%number of coefficients of RRC
Tb=f_sample/r_s;
%--------------------------------
%Transmitter
%--------------------------------
g_T=boxcar(Tb);%发送滤波器--升余弦滤波器


b=rand(1,4000);b(b>0.5)=1;b(b<=0.5)=0;
a_2n=b(:,2:2:end);
a_2n_1=b(:,1:2:end);
a_2n_r1=reshape(a_2n,2,length(a_2n)/2);
a_2n_r2=reshape(a_2n_1,2,length(a_2n_1)/2);
b_r2n_1=rand(2,1000);
x2=rand(2,1000);
for i=1:1000
  sym m;
  m=a_2n_r1(1,i);
  b_r2n_1(1,i)=m*(2.^m);
```

```
end
for i=1:1000
  sym m;
  m=a_2n_r1(2,i);
  b_r2n_1(2,i)=m;
end
b_2n=sum(b_r2n_1);
for i=1:1000
  sym m;
  m=a_2n_r2(1,i);
  x2(1,i)=m*(2.^m);
end
for i=1:1000
  sym m;
  m=a_2n_r2(2,i);
  x2(2,i)=m;
end
b_2n_1=sum(x2);
b_t(2,:)=b_2n;%b1 的值的赋给 b2 的第一行从开始到结束。
b_t(1,:)=b_2n_1;
b_t1=reshape(b_t,1,length(b)/2);
%b_2n
b2=zeros(f_sample/r_s,NumBits);
b2(1,:)=b_2n;%b1 的值的赋给 b2 的第一行从开始到结束。
b3=reshape(b2,1,f_sample/r_s*NumBits);%reshape 函数变换特定矩阵，此处是把
b2 转换成了一维矩阵
s_2n=conv(b3,g_T);%发送信号 1
fc=100;
k=0:N_sample-1+NumCoff-1;
c1=cos(2*pi*fc*k*T_sample);
s_2n=s_2n.*c1;
%--------------------------------
%b_2n_1

b2=zeros(f_sample/r_s,NumBits);
b2(1,:)=b_2n_1;%b1 的值的赋给 b2 的第一行从开始到结束。
b3=reshape(b2,1,f_sample/r_s*NumBits);%reshape 函数变换特定矩阵，此处是把
b2 转换成了一维矩阵
```

```
s_2n_1=conv(b3,g_T);%发送信号1
fc=100;
k=0:N_sample-1+NumCoff-1;
c2=-sin(2*pi*fc*k*T_sample);
s_2n_1=s_2n_1.*c2;
%---------------------------------
s=s_2n+s_2n_1;
%AWGN channel
%---------------------------------
N_0=10^(-5);
noise_w=wgn(1,length(s),N_0*f_sample,'linear');%产生一个m 行n 列的高斯白噪声的矩阵，p 以dBW 为单位指定输出噪声的强度。
r=s+noise_w;
%figure(1)
%plot(s)
%hold on
%plot(r)
%---------------------------------
%receiver
%---------------------------------(1:length(y1))
g_R=boxcar(Tb);%接收滤波器

%2n
x_2n=r.*c1;
x_2n=conv(x_2n,g_R);



%2n-11
x_2n_1=r.*c2;
x_2n_1=conv(x_2n_1,g_R);



NumCoff=10;%number of coefficients of RRC
SamplingIns=NumCoff/2;
SamplingStart=NumCoff;

sample1=zeros(f_sample/r_s,NumBits);%抽样间隔为f_sample/r_s 个点数
```

```
sample1(1,:)=ones(1,NumBits);
sample2=reshape(sample1,1,f_sample/r_s*NumBits);
sample3=zeros(1,length(x_2n));
sample3(NumCoff+1:NumCoff+f_sample/r_s*NumBits)=sample2;
%plot(sample3)
%2n 抽样
y_2n=x_2n.*sample3;
%y3=y2;
y_2n(:,all((y_2n==0),1))=[];%B = all(A, 1)返回一个行向量，可以认为all(A, 1)
等价于all(A)
b_r2n=rand(1,1000);
for i=1:1000
  sym m;
  sym n;
  m=y_2n(i);
  if(m>11.75)
    b_r2n(i)=3;
   else if(m>7.5)
     b_r2n(i)=2;
    else if(m>2.6)
     b_r2n(i)=1;
      else
      b_r2n(i)=0;
      end
    end
  end

end
%2n-1
y_2n_1=x_2n_1.*sample3;
%y3=y2;
y_2n_1(:,all((y_2n_1==0),1))=[];%B = all(A, 1)返回一个行向量，可以认为
all(A, 1)等价于all(A)

b_r2n_1=rand(1,1000);
for i=1:1000
  sym m;
  sym n;
```

```
 m=y_2n_1(i);
 if(m>12.5)
   b_r2n_1(i)=3;
  else if(m>7.5)
    b_r2n_1(i)=2;
   else if(m>2.5)
    b_r2n_1(i)=1;
     else
     b_r2n_1(i)=0;
     end
    end
 end

end
%b_t=(sign(b1)+1)*0.5;
%b_r=(sign(y2)+1)*0.5;
%BER=length(find(b_t~= b_r))/NumBits;

%并串转换
b_r(2,:)=b_r2n;%b1 的值的赋给 b2 的第一行从开始到结束。
b_r(1,:)=b_r2n_1;
b_r=reshape(b_r,1,length(b)/2);
%---------
b_t=sign(b);
BER1=length(find(b_t1~=b_r))/2000;
fprintf('误码率=%f',BER1);
```

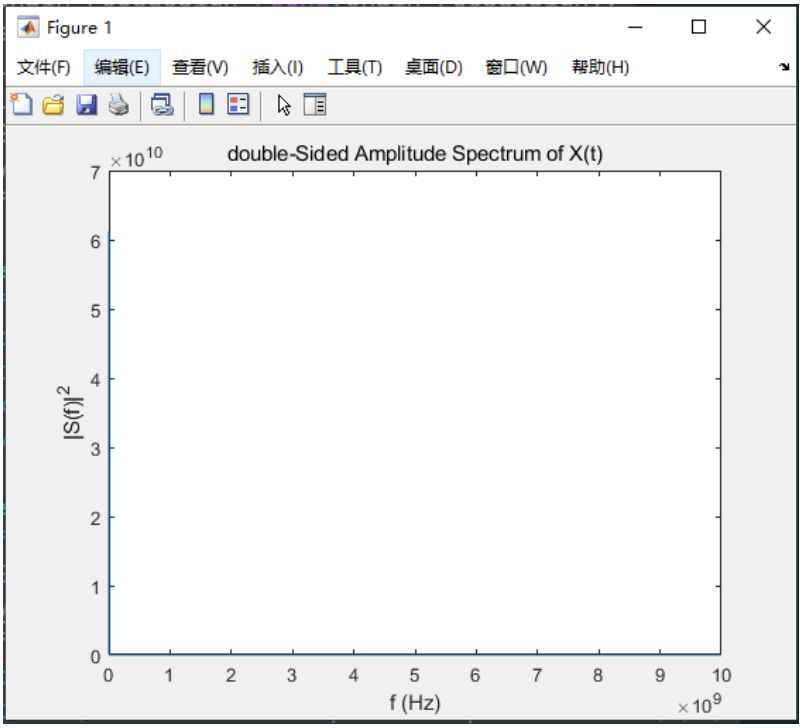
同时，为观察频谱特性，我还找到一个用于画频谱特性曲线的程序：

```
%画频谱
%主要思路与任务一中的思路相同，这里只是把方波载波变成了 RRC 脉冲

%产生 RRC 脉冲
h_1 = rcosdesign(0.25,40,60);
M = 16;%更改参数决定是 4QAM 还是 14QAM 的信号
h_1_origin = h_1; %保存下原始的 gt 信号
fs = 10^10;

%新建一个空的一维数组，可以容纳 1000*2401 个元素，用于构建冲激串
chain_rcosdesign=zeros(1,58345);
```

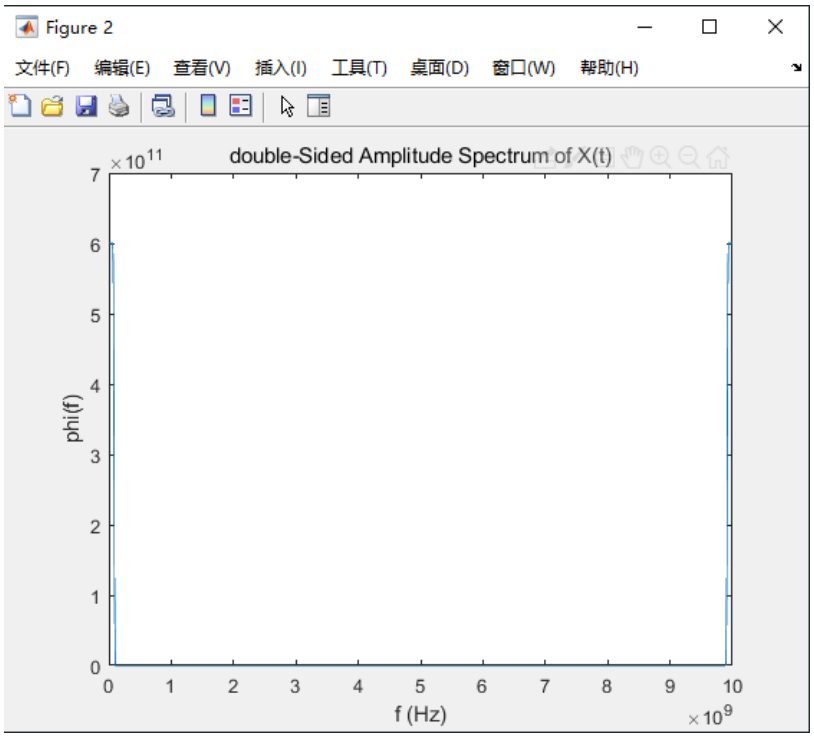
```
for i= 0:1:999
  for j=1:1:2401
   chain_rcosdesign(i*56+j)=chain_rcosdesign(i*56+j)+h_1(j);
  end
end
%构建好了类似于冲激串的函数,但是需要对于信号进行四舍五入，否则无法使用qammod 函数
chain_rcosdesign=round(chain_rcosdesign);
st = qammod( chain_rcosdesign,M);
L_1=length(st); %信号的总时长
L_2=length(h_1_origin);
Sf=fft(st);
y_SF=(abs(Sf)).^2;
f_1=(0:L_1-1)*fs/L_1;

%画出|S(f)|^2
figure(1)
plot(f_1,y_SF)
title('double-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)')
ylabel('|S(f)|^2')

%画出Phi_s
figure(2)
Gf=fft(h_1_origin);
y_phiF=(10^10)*(abs(Gf)).^2;
f_2=(0:L_2-1)*fs/L_2;
plot(f_2,y_phiF)
title('double-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)')
ylabel('phi(f)')
```

产生的幅频特性曲线和相频特性曲线如下：

幅频特性：



相频特性：



然而，在分析误码率与滚降系数之后，我却没有得到误码率与滚降系数无关的结论。因为即使不改变滚降系数，因为每次运行产生的随机序列都不同，所以连续运行几次仿真得到的误码率都不同。因为参考程序在连续运行时会出现因为变量改变而无法赋值的情况，所以在每次运行之前我都会清空工作区，这样就出

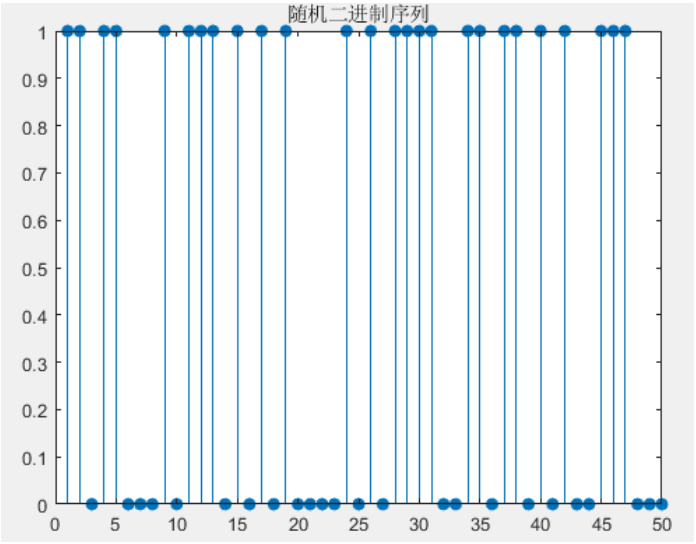
现一个问题——怎样在保证每次随机生成的信号都是相同的情况下改变滚降系数。但是我目前并没有想到如何克服这个问题。所以验证工作最后是失败的。

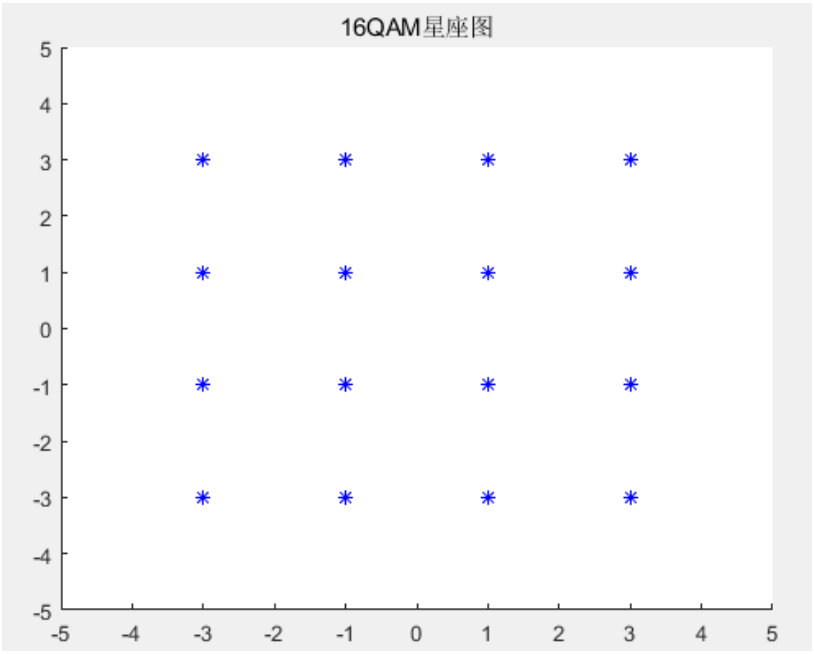
```
>> RRC_reference
误码率=0.032000>> RRC_reference
误码率=0.033000>> RRC_reference
误码率=0.042000>> RRC_reference
误码率=0.033500>>
```

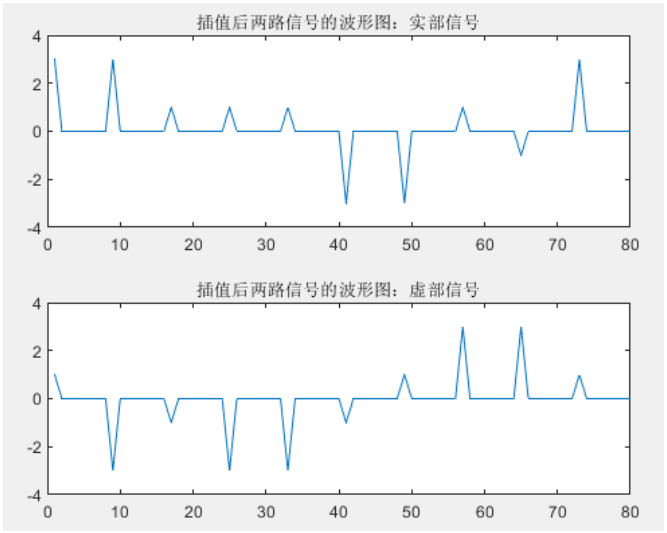
4. 有 ISI 时的数字调制信号的性能仿真

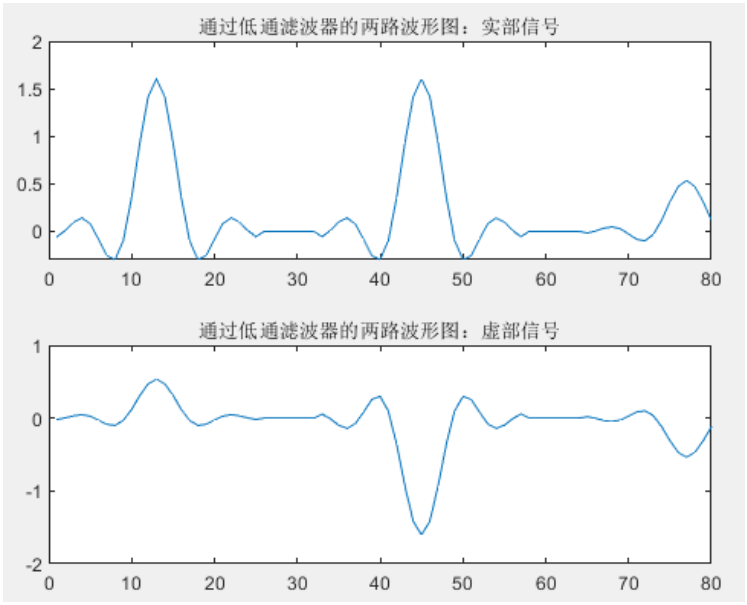
因为没有想到如何模拟仿真时刻错误，我从网上找到了相关的源代码并下载下来进行仿真、学习。

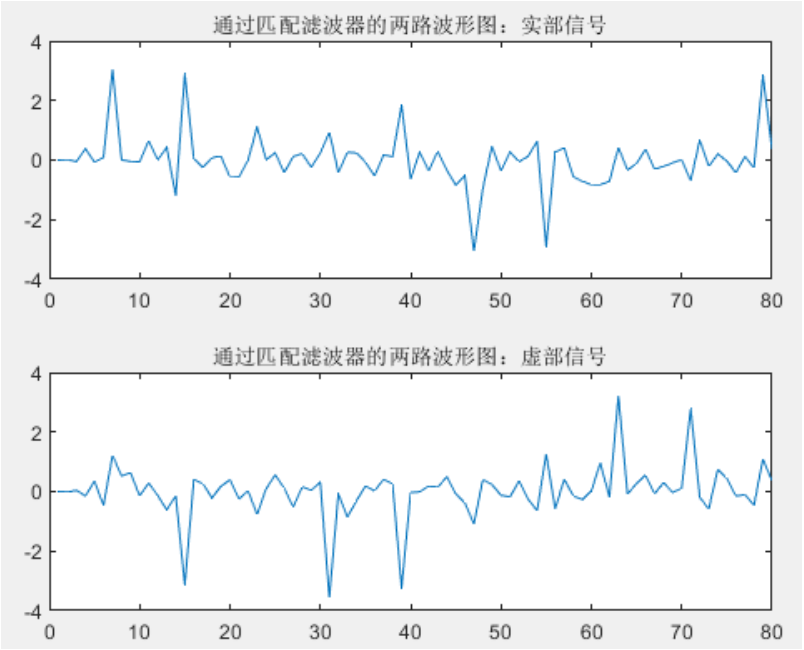
运行相关代码后得到的图像：

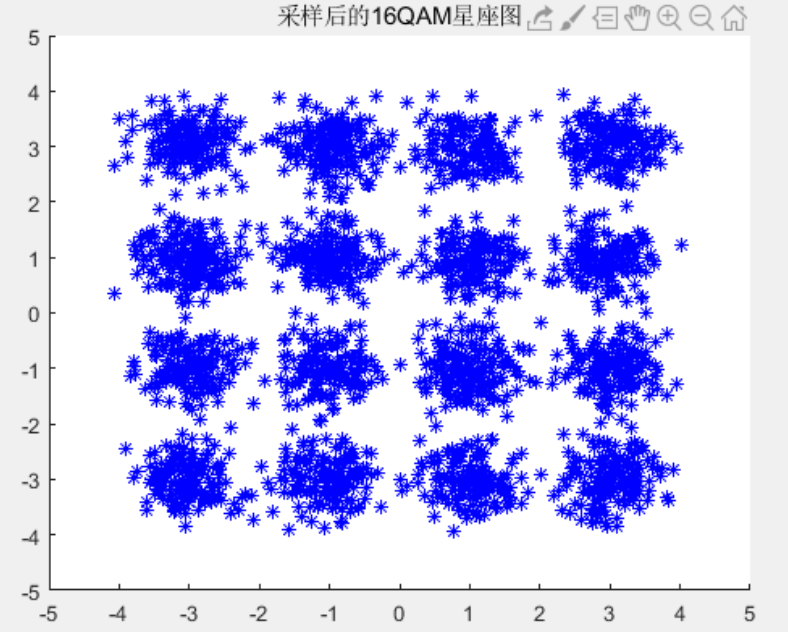


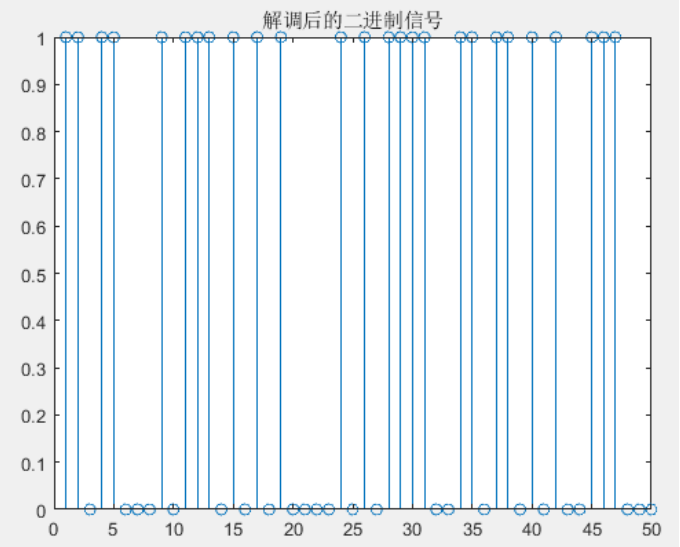


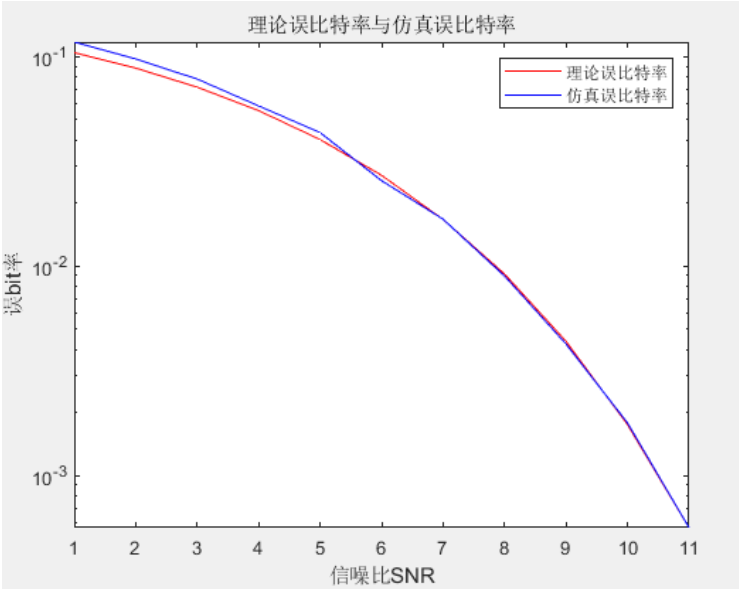

插值后两路信号的波形图：实部信号
插值后两路信号的波形图：虚部信号


通过低通滤波器的两路波形图：实部信号
通过低通滤波器的两路波形图：虚部信号


通过匹配滤波器的两路波形图：实部信号
通过匹配滤波器的两路波形图：虚部信号


采样后的16QAM星座图
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5 -4 -3 -2 -1 0 1 2 3 4 5


解调后的二进制信号
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 5 10 15 20 25 30 35 40 45 50


理论误比特率与仿真误比特率
理论误比特率
仿真误比特率
误bit率
10^{-1}
10^{-2}
10^{-3}
1 2 3 4 5 6 7 8 9 10 11
信噪比SNR

参考源码：

```
clc
clear variables
close all

%% 随机信号的生成
%=====定义待仿真序列的维数
global N
N=10000;
%=====定义产生“1”的概率为 p
global p
p=0.5;
%=========================
%首先产生随机二进制序列
source=randsrc(1,N,[1,0;p,1-p]);
%=========================
%画出二进制序列
figure(1)
stem(source(1,1:50),'filled');
title('随机二进制序列');
%============================
%对产生的二进制序列进行 QAM 调制
[source1,source2]=Qam_modulation(source);
%============================
%画出星座图
figure(2)
scatter(source1,source2,'b*');
title('16QAM 星座图');
axis([-5 5 -5 5])
%============================
%对两路信号进行插值
sig_insert1=insert_value(source1,8);
sig_insert2=insert_value(source2,8);
%==============================
%画出插值后两路信号的波形图
figure(3)
subplot(2,1,1);
plot(sig_insert1(1,1:80));
title('插值后两路信号的波形图：实部信号');
subplot(2,1,2);
plot(sig_insert2(1,1:80));
title('插值后两路信号的波形图：虚部信号');
%==============================
%通过低通滤波器
rolloff = 0.25;  % Filter rolloff
span = 6;        % Filter span
sps = 4;         % Samples per symbol
[sig_rcos1,sig_rcos2]=rise_cos(sig_insert1,sig_insert2,rolloff,span,sps);
%==============================
%画出通过低通滤波器的两路波形图
figure(4)
subplot(2,1,1);
plot(sig_rcos1(1,1:80));
title('通过低通滤波器的两路波形图：实部信号');
subplot(2,1,2);
plot(sig_rcos2(1,1:80));
title('通过低通滤波器的两路波形图：虚部信号');
%================================
```

```
%10 倍载波调制
[t,sig_modulate]=modulate_to_high(sig_rcos1,sig_rcos2,0.25,2);
%================================
%画出传输的实信号图形
figure(5)
plot(t(1:500),sig_modulate(1:500));
title('传输的实信号图形');
%================================
%将滤波后的信号加入高斯白噪声
snr=10;
[x1,x2]=generate_noise(sig_rcos1,sig_rcos2,snr);
sig_noise1=x1.';
sig_noise2=x2.';
%================================
%画出加入高斯白噪声的波形
figure(6)
subplot(2,1,1);
plot(sig_noise1(1:80));
title('加入高斯白噪声的信号波形:实部信号');
subplot(2,1,2);
plot(sig_noise2(1:80));
title('加入高斯白噪声的信号波形:虚部信号');
%================================
%经过匹配滤波器
[sig_match1,sig_match2]=filt_match(sig_noise1,sig_noise2,rolloff,span,sps);
%================================
%画出通过匹配滤波器的两路波形图
figure(7)
subplot(2,1,1);
plot(sig_match1(1:80));
title('通过匹配滤波器的两路波形图：实部信号');
subplot(2,1,2);
plot(sig_match2(1:80));
title('通过匹配滤波器的两路波形图：虚部信号');
%================================
%采样
% sample=8;
[x1,x2]=pick_sig(sig_match1,sig_match2,8);
sig_pick1=x1.';
sig_pick2=x2.';
%================================
%画出采样后的星座图
figure(8)
scatter(sig_pick1,sig_pick2,'b*');
title('采样后的 16QAM 星座图');
axis([-5 5 -5 5])
%================================
%解调
signal=demodulate_sig(sig_pick1,sig_pick2);
%================================
%画出解调后的二进制信号
figure(9)
stem(signal(1,1:50));
title('解调后的二进制信号');
% %计算误 bit 率
% error_b=length(find(signal~=source))/N;


%% 误码率计算
```

```
% =======理论误 bit 率
snr=1:1:11;
error_theory=(1-(1-(2*(1-1/sqrt(16))*1/2*erfc(1/sqrt(2)*sqrt(3*4*10.^(snr/10)/(16-
1))))).^2)/4;
% =======用理论的误 bit 率来决定需要仿真的点数
n=floor(1./error_theory)*1000+100;
n(n<5000)=5000;
% 开始仿真
for i=1:length(n)
% 首先产生随机二进制序列
source=randsrc(1,n(i),[1,0;p,1-p]);
% 对产生的二进制序列进行 QAM 调制
[source1,source2]=Qam_modulation(source);
% 对两路信号进行插值
sig_insert1=insert_value(source1,8);
sig_insert2=insert_value(source2,8);
% 通过低通滤波器
rolloff = 0.25;  % Filter rolloff
span = 6;        % Filter span
sps = 4;         % Samples per symbol
[sig_rcos1,sig_rcos2]=rise_cos(sig_insert1,sig_insert2,rolloff,span,sps);
% 将滤波后的信号加入高斯白噪声
[x1,x2]=generate_noise(sig_rcos1,sig_rcos2,snr(i));
sig_noise1=x1.';
sig_noise2=x2.';
% 经过匹配滤波器
[sig_match1,sig_match2]=filt_match(sig_noise1,sig_noise2,rolloff,span,sps);
% 采样
[x1,x2]=pick_sig(sig_match1,sig_match2,8);
sig_pick1=x1.';
sig_pick2=x2.';
% 解调
signal=demodulate_sig(sig_pick1,sig_pick2);
% 计算误 bit 率
error_bit(i)=length(find(signal-source)~=0)/n(i);
end
figure(10)
semilogy(snr,error_theory,'r-');
hold on
semilogy(snr,error_bit,'b-');
xlabel('信噪比 SNR')
ylabel('误 bit 率')
title('理论误比特率与仿真误比特率')
legend('理论误比特率','仿真误比特率')


%% 星座图映射
function[y1,y2]=Qam_modulation(x)
%Qam_modulation
%==================================
%对产生的二进制序列进行 QAM 调制
    %=======首先进行串并转换，将原二进制序列转换成两路信号
N=length(x);
a=1:2:N;
y1=x(a);
y2=x(a+1);
    %=======分别对两路信号进行 16QAM 调制
        %======对两路信号进行 2-4 电平变换
a=1:2:N/2;
```

```
temp1=y1(a);
temp2=y1(a+1);
y11=temp1*2+temp2;

temp1=y2(a);
temp2=y2(a+1);
y22=temp1*2+temp2;
        %======对两路信号分别进行相位调制
a=1:N/4;
y1=(y11*2-1-4)*1.*cos(2*pi*a);
y2=(y22*2-1-4)*1.*cos(2*pi*a);
        %======按照格雷码的规则进行映射
y1(y11==0)=-3;
y1(y11==1)=-1;
y1(y11==2)=3;
y1(y11==3)=1;

y2(y22==0)=-3;
y2(y22==1)=-1;
y2(y22==2)=3;
y2(y22==3)=1;
end

%% 插值
function y=insert_value(x,ratio)
%============================
%=======x 是待插值序列，ratio 是插值比例
%=======两路信号进行插值
%=======首先产生一个长度等于 ratio 倍原信号长度的零向量
y=zeros(1,ratio*length(x));
%========再把信号放在对应的位置
a=1:ratio:length(y);
y(a)=x;
end

%% 波形成型
%========x1、x2 是两路输入信号
function[y1,y2]=rise_cos(x1,x2,rolloff,span,sps)
%========生成平方根升余弦滤波器
rrcFilter=rcosdesign(rolloff,span,sps,'sqrt');
%=======对两路信号进行滤波
y1=upfirdn(x1,rrcFilter,sps);
y2=upfirdn(x2,rrcFilter,sps);
end

%% 10 倍载波调制
%=====x1，x2 代表两路输入信号，f 是输入信号的频率，hf 是载波频率
function [t,y]=modulate_to_high(x1,x2,f,hf)
%=====产生两个中间变量，用来存储插值后的输入信号
yo1=zeros(1,length(x1)*hf/f*10);
yo2=zeros(1,length(x2)*hf/f*10);
n=1:length(yo1);
%=======对输入信号分别进行插值，相邻的两个点之间加入 9 个点，且这 9 个点的值同第 0 个点的值
yo1(n)=x1(floor((n-1)/(hf/f*10))+1);
yo2(n)=x2(floor((n-1)/(hf/f*10))+1);
%======生成输出信号的时间向量
t=(1:length(yo1))/hf*f/10;
%======生成载波调制信号
y=yo1.*cos(2*pi*hf*t)-yo2.*sin(2*pi*hf*t);
```

```
end

%% 加入高斯白噪声
%======对输入的两路信号加高斯白噪声，返回处理后的两路信号，信息点等效 bit 信噪比为 snr 的值
function[y1,y2]=generate_noise(x1,x2,snr)
%=======snr1 代表 snr 对于的符号信噪比
snr1=snr+10*log10(4);
%=======算出所有信号的平均功率
ss=var(x1+1i*x2,1);
%=======加入高斯白噪声
y=awgn(x1+1i*x2,snr1+10*log10(ss/10),'measured');
y1=real(y);
y2=imag(y);
end

%% 匹配滤波器
%========x1、x2 是两路输入信号
function[y1,y2]=filt_match(x1,x2,rolloff,span,sps)
%========生成平方根升余弦滤波器
rrcFilter=rcosdesign(rolloff,span,sps,'sqrt');
%=======对两路信号进行滤波
y1=upfirdn(x1,rrcFilter,1,sps);
y2=upfirdn(x2,rrcFilter,1,sps);
end


%% 采样
function[y1,y2]=pick_sig(x1,x2,ratio)
y1=x1(3*2+1:ratio:length(x1)-ratio);
y2=x2(3*2+1:ratio:length(x2)-ratio);
end

%% 判决解调
function y=demodulate_sig(x1,x2)
%=======对 x1 路信号进行判决
xx1(x1>2)=3;
xx1((x1<2)&(x1>=0))=1;
xx1((x1>=-2)&(x1<0))=-1;
xx1(x1<-2)=-3;
%=======对 x2 路信号进行判决
xx2(x2>2)=3;
xx2((x2<2)&(x2>=0))=1;
xx2((x2>=-2)&(x2<0))=-1;
xx2(x2<-2)=-3;
%========将 x1 路信号按格雷码规则还原成 0,1 信号
temp1=zeros(1,length(xx1)*2);
temp1(find(xx1==-1)*2)=1;
temp1(find(xx1==1)*2-1)=1;
temp1(find(xx1==1)*2)=1;
temp1(find(xx1==3)*2-1)=1;
%========将 x2 路信号按格雷码规则还原成 0,1 信号
temp2=zeros(1,length(xx2)*2);
temp2(find(xx2==-1)*2)=1;
temp2(find(xx2==1)*2-1)=1;
temp2(find(xx2==1)*2)=1;
temp2(find(xx2==3)*2-1)=1;
%========将两路信号合成 1 路
y=zeros(1,length(temp1)*2);
y(1:2:length(y))=temp1;
```

```
y(2:2:length(y))=temp2;
end
```

参考文献

[1] Matlab-16QAM 调制与解调 16-QAM 星座点图 16-QAM 在 AWGN 信道下的误码率和误比特率性能，仿真值与理论值曲线对比图 君琴 的博客-CSDN 博客

[2] BPSK、QPSK、MPSK、QAM、16QAM 的调制解调 Matlab 实现_ICT_Liang 的博客-CSDN 博客_matlab qam