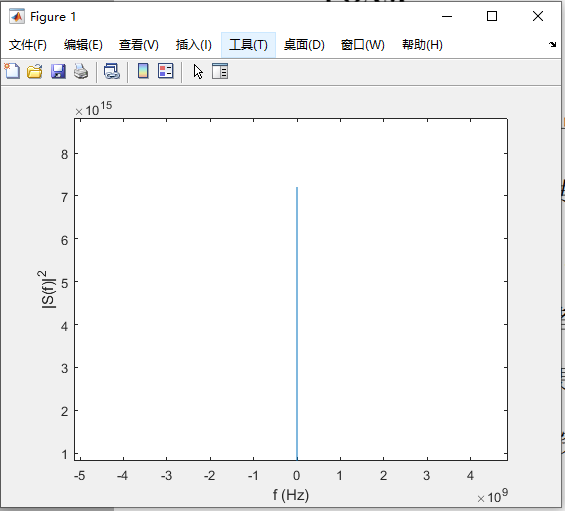
信号与系统实践任务三报告

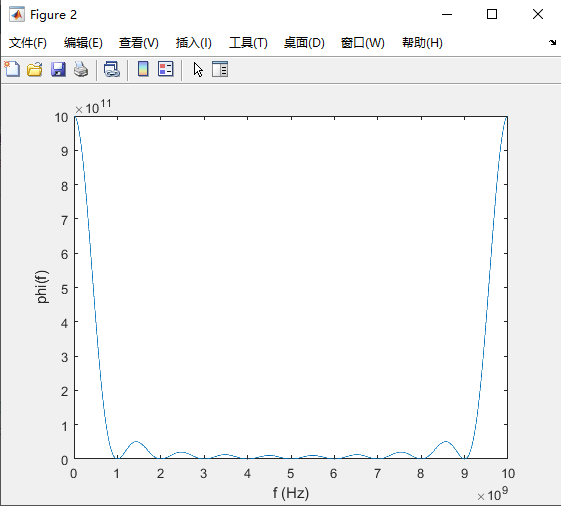
刘正浩 2019270103005

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)，分别绘图和，观察两者的异同。

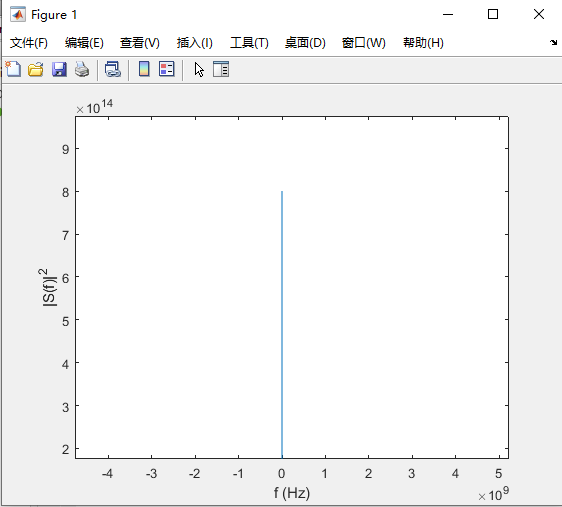
画出的16-QAM图为



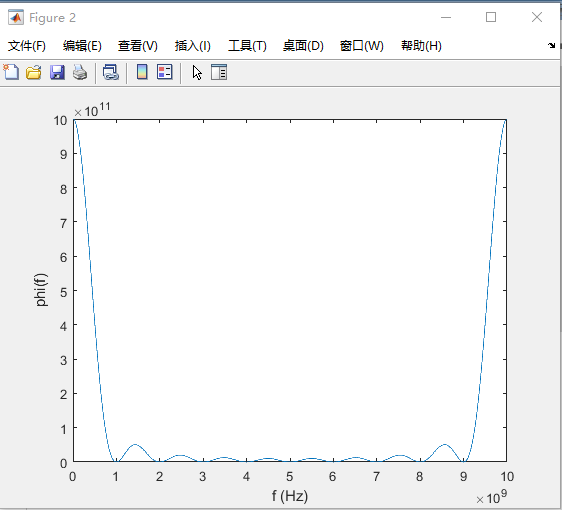
图为



画出的4-QAM图为



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)')

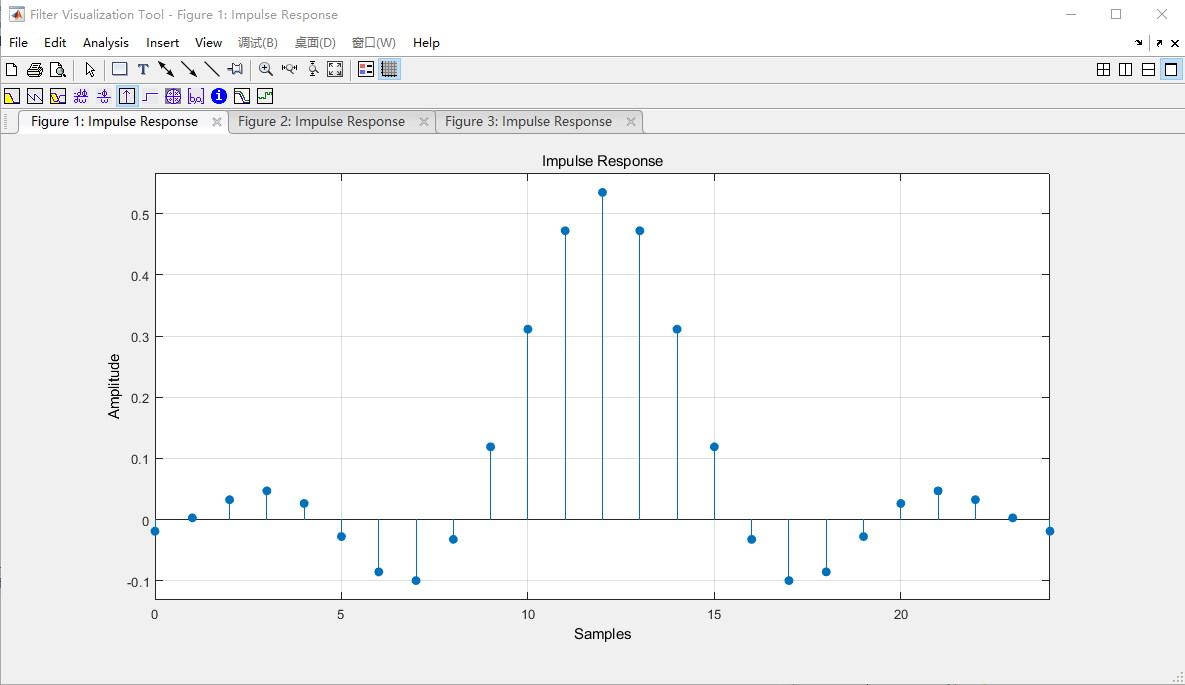
ylabel('phi(f)')

2. 奈奎斯特准则

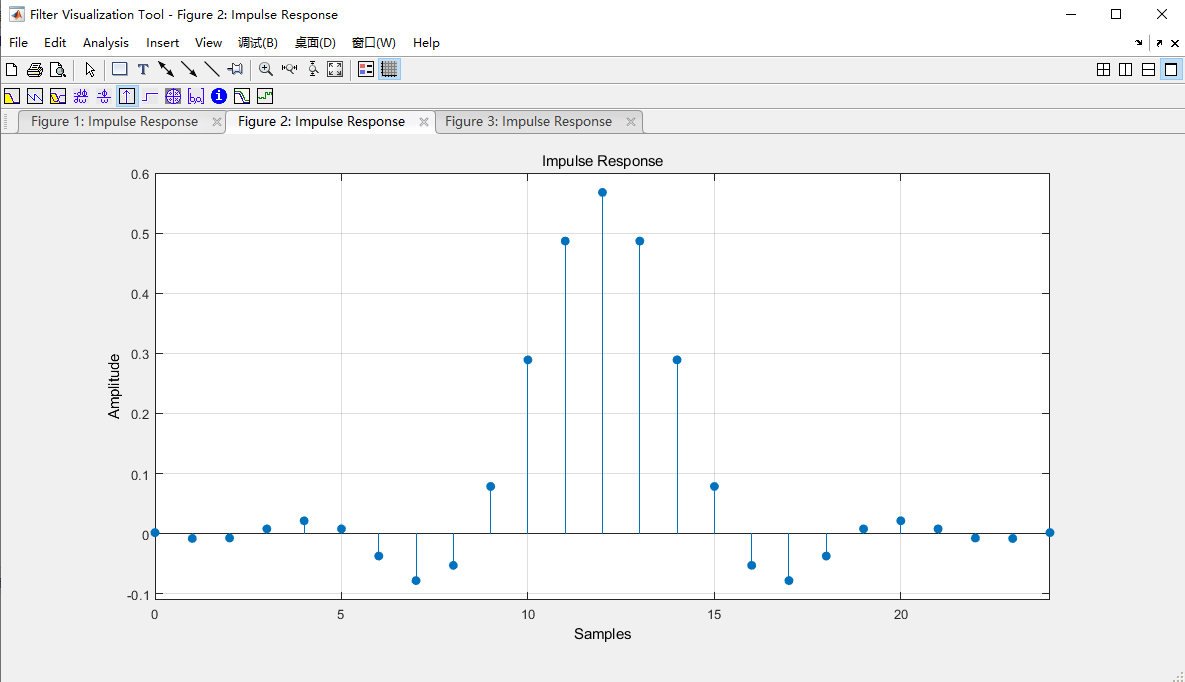
本任务的目的是，在MATLAB中产生不同滚降系数的升余弦脉冲，画出其时域波形、频域频谱，并与升余弦函数的正确波形、频谱比较正确性进行验证。在仿真过程中，使用函数rcosdesign()产生离散采样后的升余弦脉冲。

仿真结果如下：

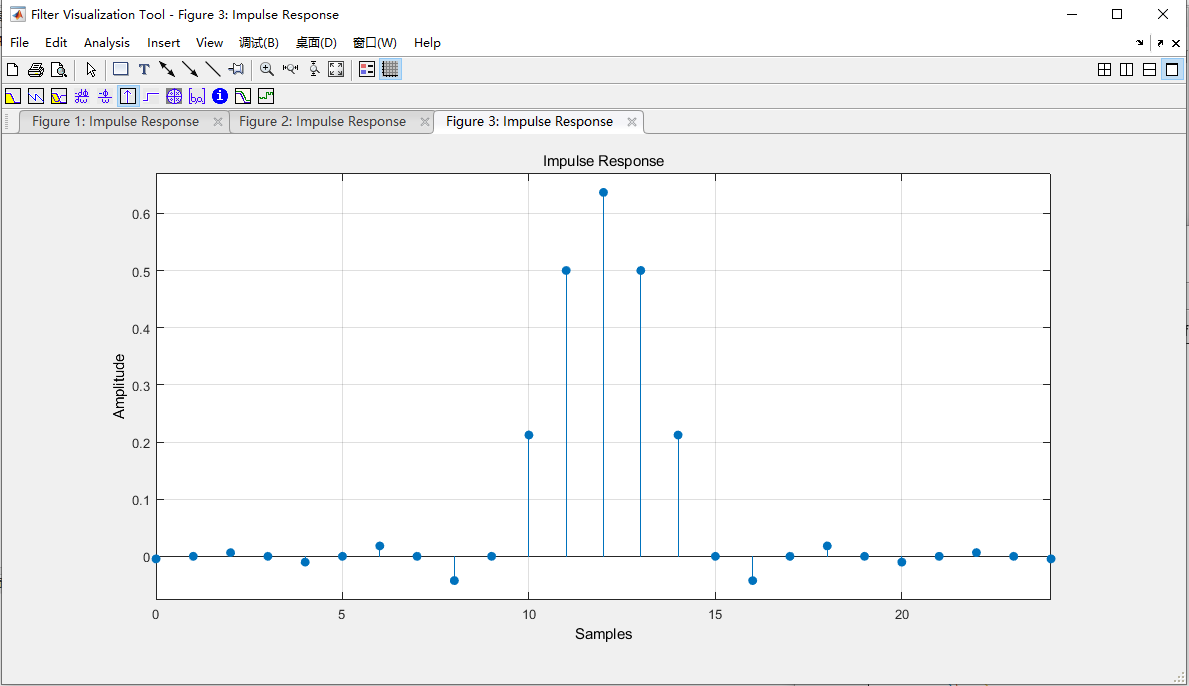
滚降系数为0.25的升余弦函数



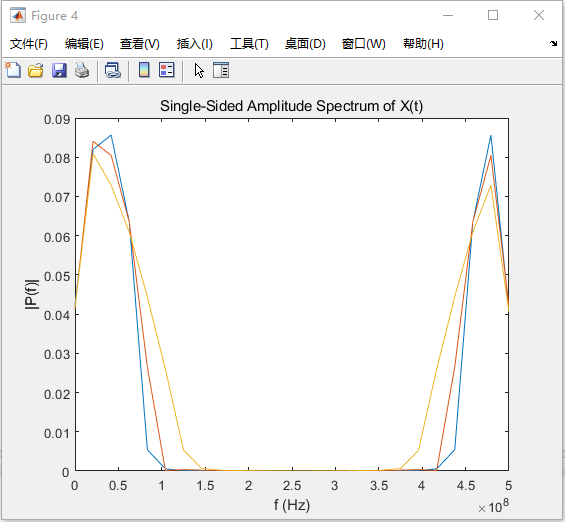
滚降系数为0.5的升余弦函数：



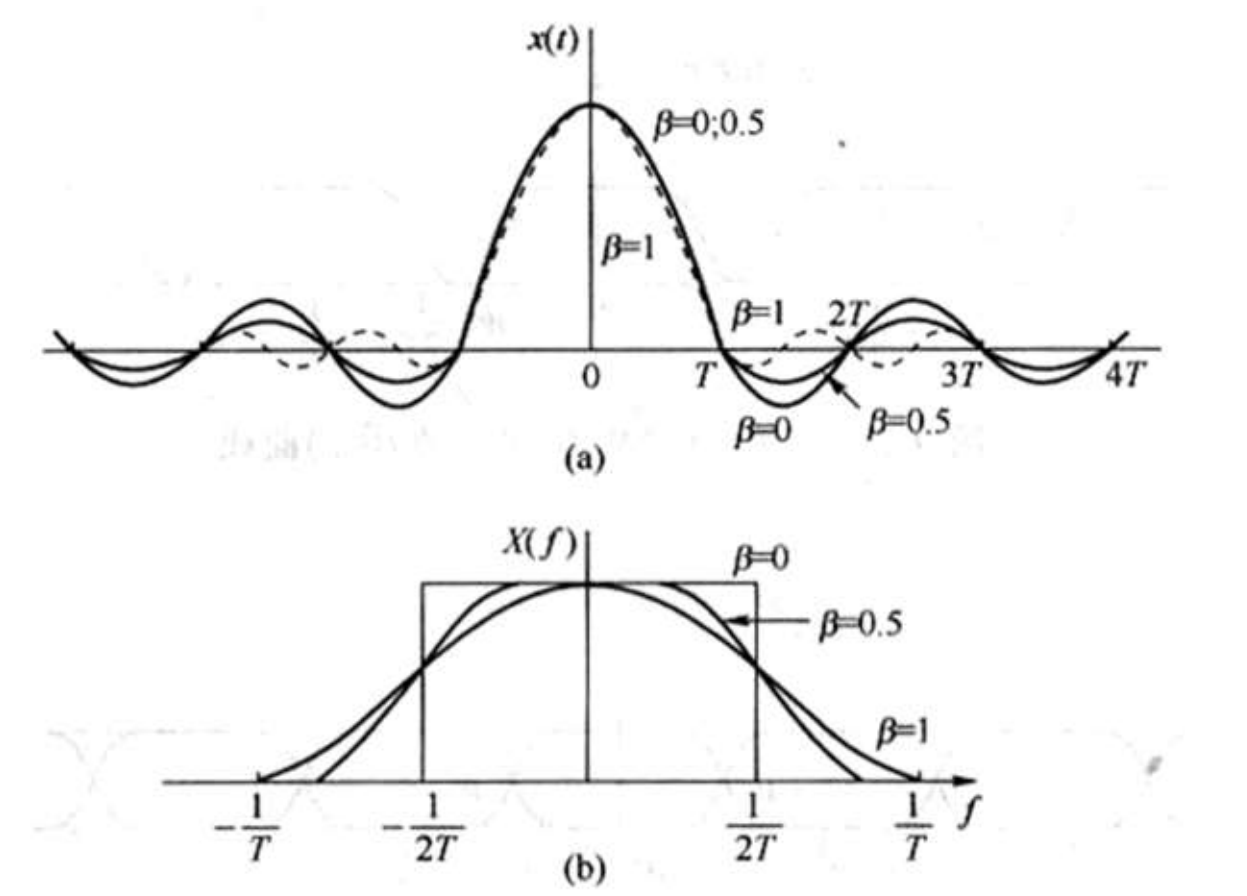
滚降系数为1的升余弦函数



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)

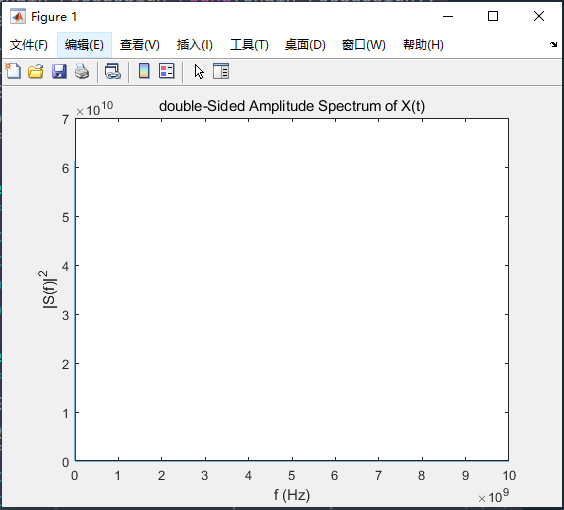
title('double-Sided Amplitude Spectrum of X(t)')

xlabel('f (Hz)')

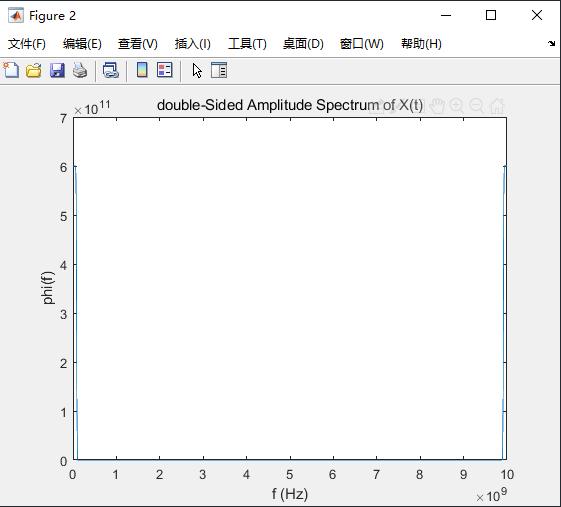
ylabel('phi(f)')

产生的幅频特性曲线和相频特性曲线如下：

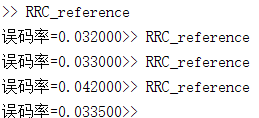
幅频特性：



相频特性：



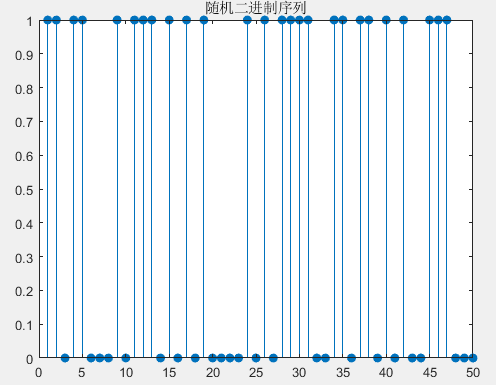
然而，在分析误码率与滚降系数之后，我却没有得到误码率与滚降系数无关的结论。因为即使不改变滚降系数，因为每次运行产生的随机序列都不同，所以连续运行几次仿真得到的误码率都不同。因为参考程序在连续运行时会出现因为变量改变而无法赋值的情况，所以在每次运行之前我都会清空工作区，这样就出现一个问题——怎样在保证每次随机生成的信号都是相同的情况下改变滚降系数。但是我目前并没有想到如何克服这个问题。所以验证工作最后是失败的。

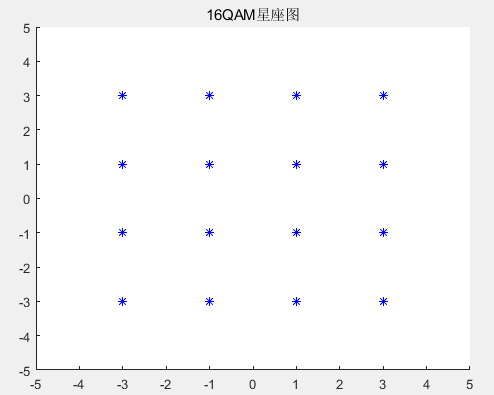


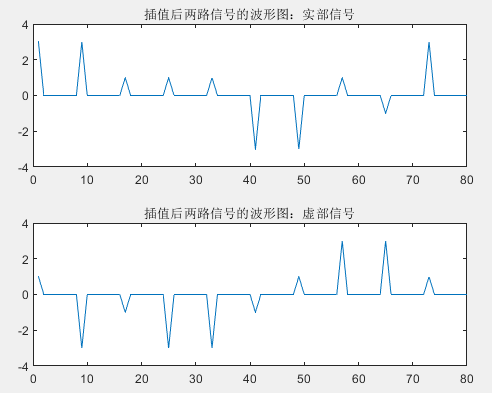
4. 有ISI时的数字调制信号的性能仿真

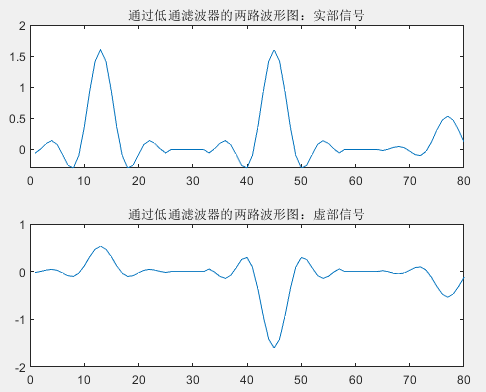
因为没有想到如何模拟仿真时刻错误，我从网上找到了相关的源代码并下载下来进行仿真、学习。

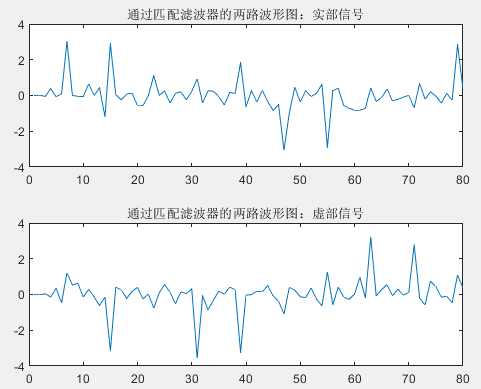
运行相关代码后得到的图像：

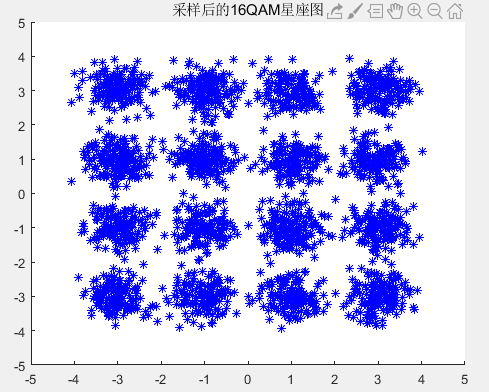


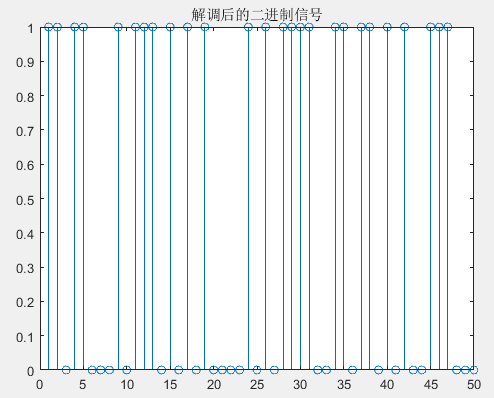


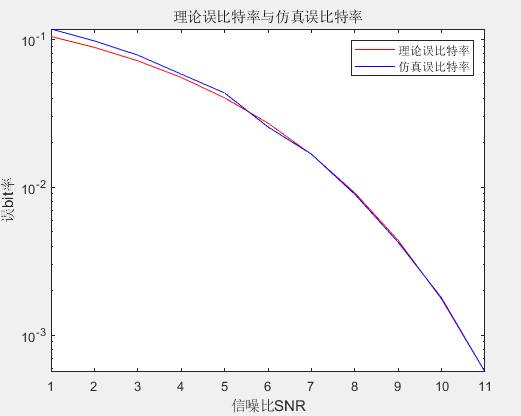












参考源码：

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](https://blog.csdn.net/weixin_44436677/article/details/106819257?biz_id=102&utm_term=%E8%B0%83%E5%88%B6%E8%84%89%E5%86%B2%EF%BC%8C%E4%BA%A7%E7%94%9F16-QAM%20%E7%9A%84%20%E8%B0%83%E5%88%B6%E4%BF%A1%E5%8F%B7&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-2-106819257&spm=1018.2118.3001.4449)

[2] [BPSK、QPSK、MPSK、QAM、16QAM的调制解调Matlab实现\_ICT\_Liang的博客-CSDN博客\_matlab qam](https://blog.csdn.net/weixin_41963233/article/details/90664747?utm_medium=distribute.pc_relevant_download.none-task-blog-BlogCommendFromBaidu-2.nonecase&depth_1-utm_source=distribute.pc_relevant_download.none-task-blog-BlogCommendFromBaidu-2.nonecas)