# 通信原理第四次实验

## 一、实验目的

1. 掌握低通信号的抽样及重建过程；

2. 掌握PCM的编码及译码过程；

3. 掌握汉明码的编码及译码过程；

4. 掌握卷积码的编码及译码过程。

## 二、实验内容

1. 设低通信号。

（1）画出该低通信号的波形；

（2）画出抽样速率为的抽样序列；

（3）画出抽样序列恢复出的原始信号。

2.设输入信号为对信号进行抽样、量化和A律PCM编码，经过传输后，接收端进行PCM译码，过载电平。

（1）画出经过PCM编码、译码后的波形与未编码波形的对比图；

（2）设信道没有误码，画出不同幅度情况下（dB，），PCM译码后的量化信噪比。此时仿真框图可以表示为：

形状

中度可信度描述已自动生成

仿真中噪声功率可以用mean((x-y).^2)来计算，其中x表示信号的抽样值序列，y表示量化信号值序列。

3.设消息比特个数为1000000，仿真进行（7，4）汉明编码的QPSK调制通过AWGN信道后的误比特率性能。

4.设消息比特个数为1000000，仿真BPSK调制在AWGN信道下使用卷积码的误比特率性能，其中卷积码约束长度为7，生成多项式为[171,133]，码率为二分之一，译码分别采用硬判决译码和软判决译码。

## 三、实验程序

实验1：

clear all;

t=0:0.01:20;

%模拟生成连续信号

xt=cos(0.15\*pi\*t)+sin(2.5\*pi\*t)+cos(4\*pi\*t);

subplot(3,1,1);

plot(t,xt);

title('低通信号的波形');

%对信号已4Hz进行采样

t\_=0:1/4:20;

xt\_=cos(0.15\*pi\*t\_)+sin(2.5\*pi\*t\_)+cos(4\*pi\*t\_);

subplot(3,1,2);

stem(t\_,xt\_);

title('fs=4Hz的抽样序列波形');

T=1/4;

x=zeros(1,2001);

%通过插值重构得到原信号

for i=1:length(t\_)

x=x+xt\_(i)\*sinc((t-i\*T+T)/T);

end

subplot(3,1,3);

plot(t,x);

title('通过序列恢复的原始信号');

xlabel('t/s');

ylabel('U/V');

实验2.1

dt=0.01;

t=0:dt:10-dt;

xt=sin(2\*pi\*t); %绘画出原信号

hold on;

plot(t,xt);

dt\_=0.1;

t\_=0:dt\_:10-dt\_; %对原信号进行抽样，抽样频率是10Hz

xt\_=sin(2\*pi\*t\_);

delta=1/4096; %量化的最小分度

qxt=xt\_/delta;

decode=zeros(1,length(qxt));

for i=1:length(qxt) %进行译码

temp=qxt(i);

if temp>=0 && temp<32

temp=floor(temp/2)\*2+1;

elseif temp>=32 && temp<64

temp=floor(temp/2)\*2+1;

elseif temp>=64 && temp<128

temp=floor(temp/4)\*4+2;

elseif temp>=128 && temp<256

temp=floor(temp/8)\*8+4;

elseif temp>=256 && temp<512

temp=floor(temp/16)\*16+8;

elseif temp>=512 && temp<1024

temp=floor(temp/32)\*32+16;

elseif temp>=1024 && temp<2048

temp=floor(temp/64)\*64+32;

elseif temp>=2048 && temp<4096

temp=floor(temp/128)\*128+64;

elseif temp==4096

temp=4032;

end

if temp<0

temp=-temp;

if temp>=0 && temp<32

temp=floor(temp/2)\*2+1;

elseif temp>=32 && temp<64

temp=floor(temp/2)\*2+1;

elseif temp>=64 && temp<128

temp=floor(temp/4)\*4+2;

elseif temp>=128 && temp<256

temp=floor(temp/8)\*8+4;

elseif temp>=256 && temp<512

temp=floor(temp/16)\*16+8;

elseif temp>=512 && temp<1024

temp=floor(temp/32)\*32+16;

elseif temp>=1024 && temp<2048

temp=floor(temp/64)\*64+32;

elseif temp>=2048 && temp<4096

temp=floor(temp/128)\*128+64;

elseif temp==4096

temp=4032;

end

temp=-temp;

end

decode(i)=temp;

end

decode=decode\*delta;

plot(t\_,decode);

xlabel('t/s');

ylabel('U/v');

title('原始信号和经过PCM调制的信号');

legend('原始信号','采用PCM编码、译码之后的结果');

实验2.2

G=-70:0.01:0;

y=zeros(1,length(G));

A0=1;

for i =1:length(G)

%因为A0是1，所以可以直接这样转换

Ac=sqrt(db2pow(G(i)))\*A0;

SN=quantityOnce(Ac);

SN=pow2db(SN);

y(i)=SN;

end

plot(G,y);

title('信噪比和量噪比的关系')

xlabel('$\frac{Ac^2}{A\_0^2}/dB$','interpreter','latex');

ylabel('量噪比qSNR/dB');

其中quantityOnce：

function [ qSNR ] = quantityOnce( Ac )

%QUANTITYONCE ，获得不同Ac的情况，返回量噪比

% 输入 Ac ，输出量噪比

t=0:0.01:10;

xt=Ac\*sin(2\*pi\*t);

delta=1/4096;

qxt=xt/delta;

decode=zeros(1,length(qxt));

for i=1:length(qxt) %进行译码

temp=qxt(i);

if temp>=0 && temp<32

temp=floor(temp/2)\*2+1;

elseif temp>=32 && temp<64

temp=floor(temp/2)\*2+1;

elseif temp>=64 && temp<128

temp=floor(temp/4)\*4+2;

elseif temp>=128 && temp<256

temp=floor(temp/8)\*8+4;

elseif temp>=256 && temp<512

temp=floor(temp/16)\*16+8;

elseif temp>=512 && temp<1024

temp=floor(temp/32)\*32+16;

elseif temp>=1024 && temp<2048

temp=floor(temp/64)\*64+32;

elseif temp>=2048 && temp<4096

temp=floor(temp/128)\*128+64;

end

if temp<0

temp=-temp;

if temp>=0 && temp<32

temp=floor(temp/2)\*2+1;

elseif temp>=32 && temp<64

temp=floor(temp/2)\*2+1;

elseif temp>=64 && temp<128

temp=floor(temp/4)\*4+2;

elseif temp>=128 && temp<256

temp=floor(temp/8)\*8+4;

elseif temp>=256 && temp<512

temp=floor(temp/16)\*16+8;

elseif temp>=512 && temp<1024

temp=floor(temp/32)\*32+16;

elseif temp>=1024 && temp<2048

temp=floor(temp/64)\*64+32;

elseif temp>=2048 && temp<4096

temp=floor(temp/128)\*128+64;

end

temp=-temp;

end

decode(i)=temp;

end

decode=decode\*delta;

varNoise=mean((decode-xt).^2); %计算量化噪声

varSignal=var(xt); %计算信号噪声

qSNR =varSignal/varNoise; %计算量噪比

end

实验3：

clear all;

n = 7; % Codeword length

k = 4; % Message length

r=-15:0.5:10;

bitlen=100000000;

data = randi([0 1],1,bitlen);

BER=zeros(1,length(r));

for i=1:length(r)

encData = encode(data,n,k,'hamming/binary'); %进行海明编码

decModuleData=qpsk(r(i),encData); %经过信噪比 r的 AWGN信道，得到的QPSK解调之后的比特串

decData = decode(decModuleData,n,k,'hamming/binary'); %进行海明编码的译码

numerr = biterr(data,decData); %计算比特错误率

ber=numerr/bitlen;

BER(i)=ber;

end

hold on;

plot(r,BER);

r\_db=-15:0.5:10;

r\_pow=db2pow(r\_db);

%通过公式计算理论的信噪比和误码率的函数

Pe\_pow=0.5\*erfc(sqrt(r\_pow/2) ) ;

plot(r\_db,Pe\_pow);

legend('采用Haming编码的误码率','不采用Haming编码的理论的误码率');

xlabel('SNR /dB');

ylabel('Pe');

其中qpsk：

function [ newblist] = qpsk(rb,bitlist)

%SIM\_Q2 输入信噪比rb/db，输入比特序列，得到解调后的序列

r=db2pow(rb); %将dB->power

bitlen=length(bitlist); %比特序列长度

baudlen=bitlen/2; %码元数

fs=100; %采样频率

dt=1/fs; %参数可以调，调细一点，取值更加合理，跑的更慢

t=0:dt:baudlen-dt; %码元持续的时间

%进行QPSK调制

st=zeros(1,length(t));

for i=1 :baudlen

%串并转换

a=bitlist(2\*i-1);

b=bitlist(2\*i);

if a==0 %将0的比特转换成 -1,方便调制

a=-1;

end

if b==0

b=-1;

end

st(1+(i-1)\*(1/dt):i\*(1/dt) )=a\*cos(2\*pi\*20\*t(1+(i-1)\*(1/dt):i\*(1/dt)))-b\*sin(2\*pi\*20\*t(1+(i-1)\*(1/dt):i\*(1/dt)));

end

%模拟AWGN信道，添加高斯白噪声，并通过带通滤波器

%信噪比为r db，A=sqrt(2) 噪声功率是1

P=1/r;

st=st+sqrt(P\*(1/dt/2/4)).\*randn(1,length(t)); %模拟信道的噪声的功率由解调器的信号功率推出 ， fs/2=50 ， B=4 所以功率是50/4

st=bandpass(18,22,1/dt,st);

%----------------------------------------

%用两个载波进行解调，分为上下之路

st\_up=st.\*cos(2\*pi\*20\*t);

st\_low=st.\*-sin(2\*pi\*20\*t);

%-------------------------------------

%低通滤波器， 因为数字基带信号的频率是 0-fB

st\_up=lowpass(2,1/dt,st\_up); %0-2 Hz的低通滤波器

st\_low=lowpass(2,1/dt,st\_low);

%------------------------

%对每一个支路进行抽样判决

newblist(bitlen)=0;

for i =1:baudlen

if st\_up((1/dt)\*(i)-(1/5)\*(1/dt) )>0

a=1;

else

a=0;

end

if st\_low((1/dt)\*(i)-(1/5)\*(1/dt))>0

b=1;

else

b=0;

end

%并串转换

newblist(2\*i-1)=a;

newblist(2\*i)=b;

end

end

实验4：

clear all;

%这里是硬编码

trellis = poly2trellis(7,[171 133]); %生成多项式，约束是7

tb = 30;

r=-15:0.5:10;

bitlen=1000;

data = randi([0 1],1,bitlen);

BER=zeros(1,length(r));

for i=1:length(r)

[encData,fstate\_a] = convenc(data,trellis);

decModuleData=bpsk(r(i),encData);

decData = vitdec(decModuleData,trellis,tb,'trunc','hard');

numerr = biterr(data,decData);

ber=numerr/bitlen;

BER(i)=ber;

end

subplot(2,1,1);

hold on;

plot(r,BER);

r\_db=-15:0.5:10;

r\_pow=db2pow(r\_db);

%通过公式计算理论的信噪比和误码率的函数

Pe\_pow=0.5\*erfc(sqrt(r\_pow));

plot(r\_db,Pe\_pow);

legend('采用卷积编码(硬解码)的误码率','不采用卷积编码的理论的误码率');

xlabel('SNR /dB');

ylabel('Pe');

%采用软编码

r=-15:0.5:10;

data = randi([0 1],1,bitlen);

BER=zeros(1,length(r));

for i=1:length(r)

[encData,fstate\_a] = convenc(data,trellis);

decModuleData=bpsk(r(i),encData);

decData = vitdec(decModuleData,trellis,tb,'trunc','soft',1);

numerr = biterr(data,decData);

ber=numerr/bitlen;

BER(i)=ber;

end

subplot(2,1,2);

hold on;

plot(r,BER);

r\_db=-15:0.5:10;

r\_pow=db2pow(r\_db);

%通过公式计算理论的信噪比和误码率的函数

Pe\_pow=0.5\*erfc(sqrt(r\_pow));

plot(r\_db,Pe\_pow);

legend('采用卷积编码（软解码）的误码率','不采用卷积编码的理论的误码率');

xlabel('SNR /dB');

ylabel('Pe');

其中bpsk：

function [ newblist] = bpsk(rb,bitlist)

%输入信噪比r/db，得到一次2PSK的误码率,Pe

% 此处显示详细说明

bitlen=length(bitlist);

r=db2pow(rb);

%---------------------------

%生成数字信号

dt=0.01; %参数可以调，调细一点，取值更加合理，跑的更慢

t=0:dt:bitlen-dt;

xt=zeros(1,length(t));

%对于每1秒中的取值赋值

for i=1 :bitlen

if bitlist(i)==1

xt(1+(i-1)\*(1/dt):i\*(1/dt) )=1;

else

xt(1+(i-1)\*(1/dt):i\*(1/dt) )=-1;

end

end

%--------------------------------

%生成载波并调制载波

carrier=cos(2\*pi\*20\*t);

st=xt.\*carrier;

%信噪比为0，A=1 噪声功率是1/2

%------------

%模拟AWGN信号，加上高斯白噪声，通过带通滤波器

P=0.5/r;

st=st+sqrt(P\*(1/dt/2/4)).\*randn(1,length(t)); %模拟信道的噪声 ， fs/2=50 ， B=4 所以功率是50/4

st=bandpass(18,22,1/dt,st);

%------------------------------

%用载波进行解调，并通过低通滤波器，

st=st.\*carrier;

st=lowpass(2,1/dt,st);

%----------------------------

%定时抽样

newblist(bitlen)=0;

for i =1:bitlen

if st((1/dt)\*(i)-(1/5)\*(1/dt))>0

newblist(i)=1;

else

newblist(i)=0; %可以是0，也可以是1

end

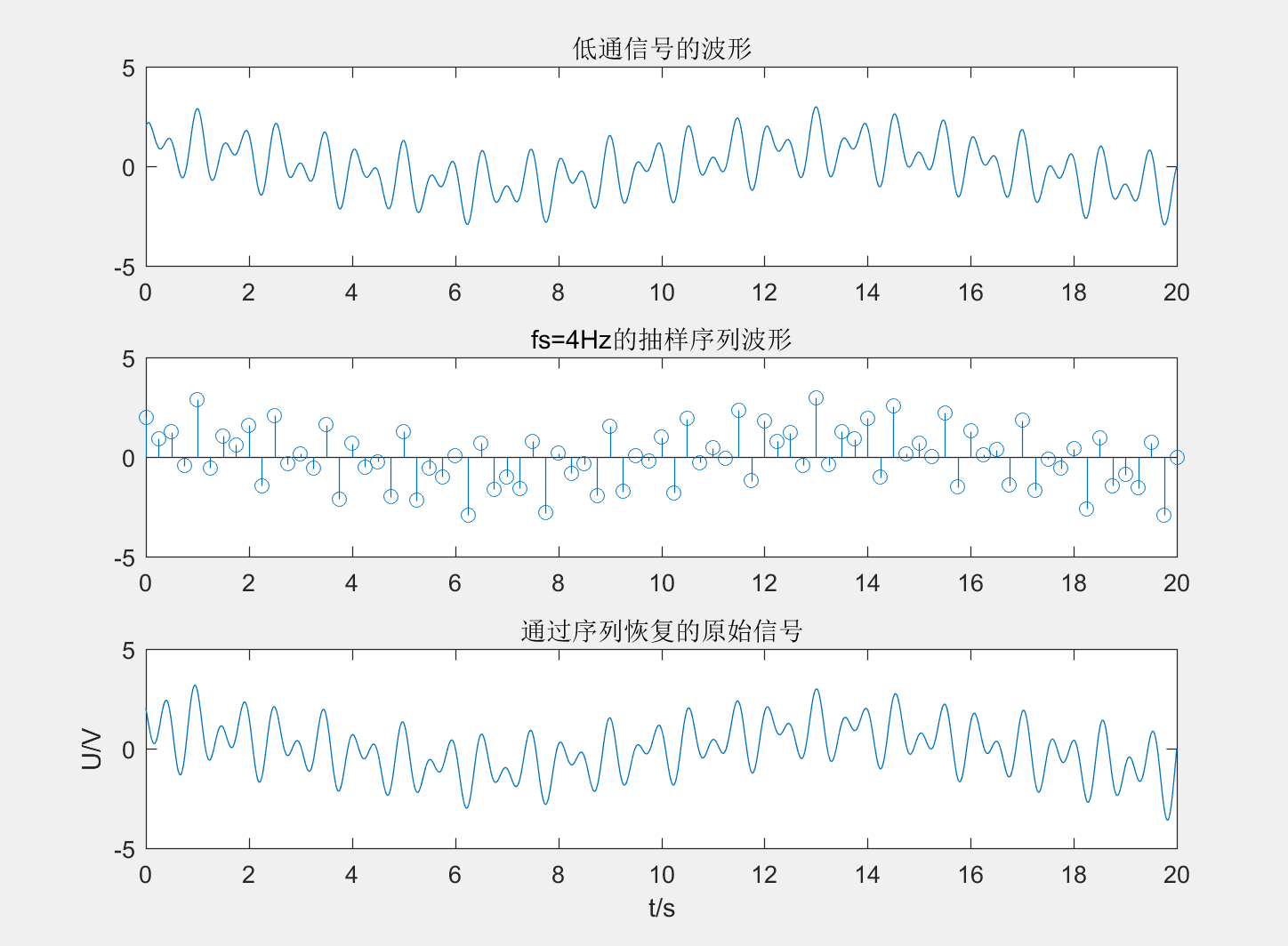
end

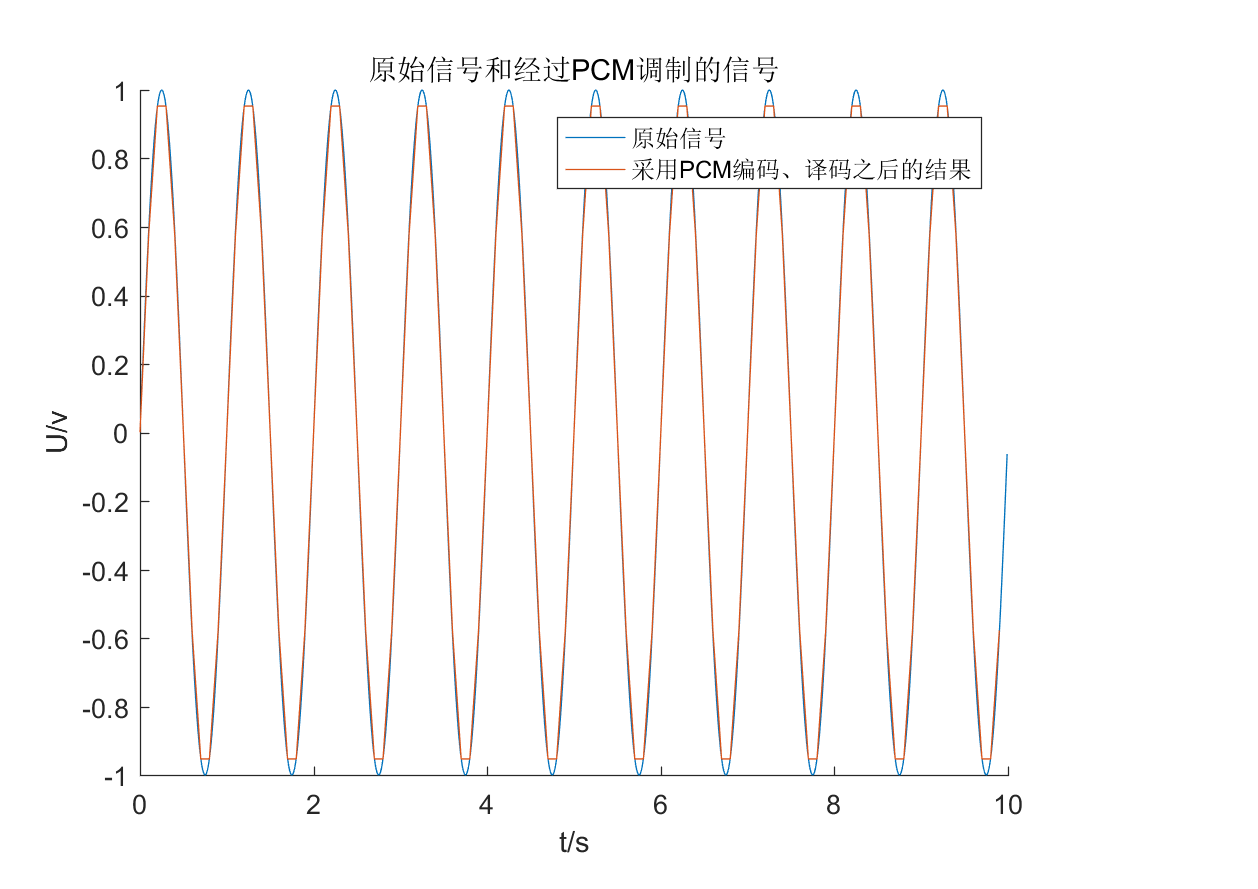
%------------------------------

end

## 四、实验结果

实验1：

实验2.1：

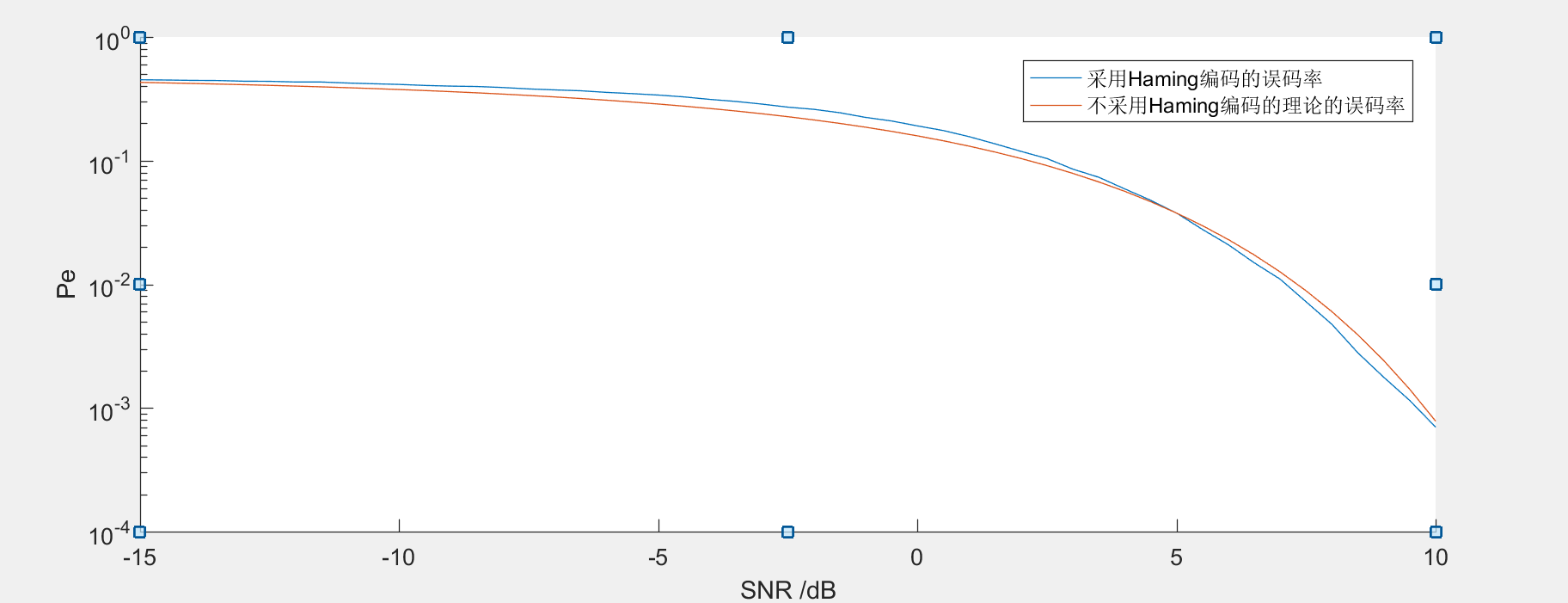


实验2.2：

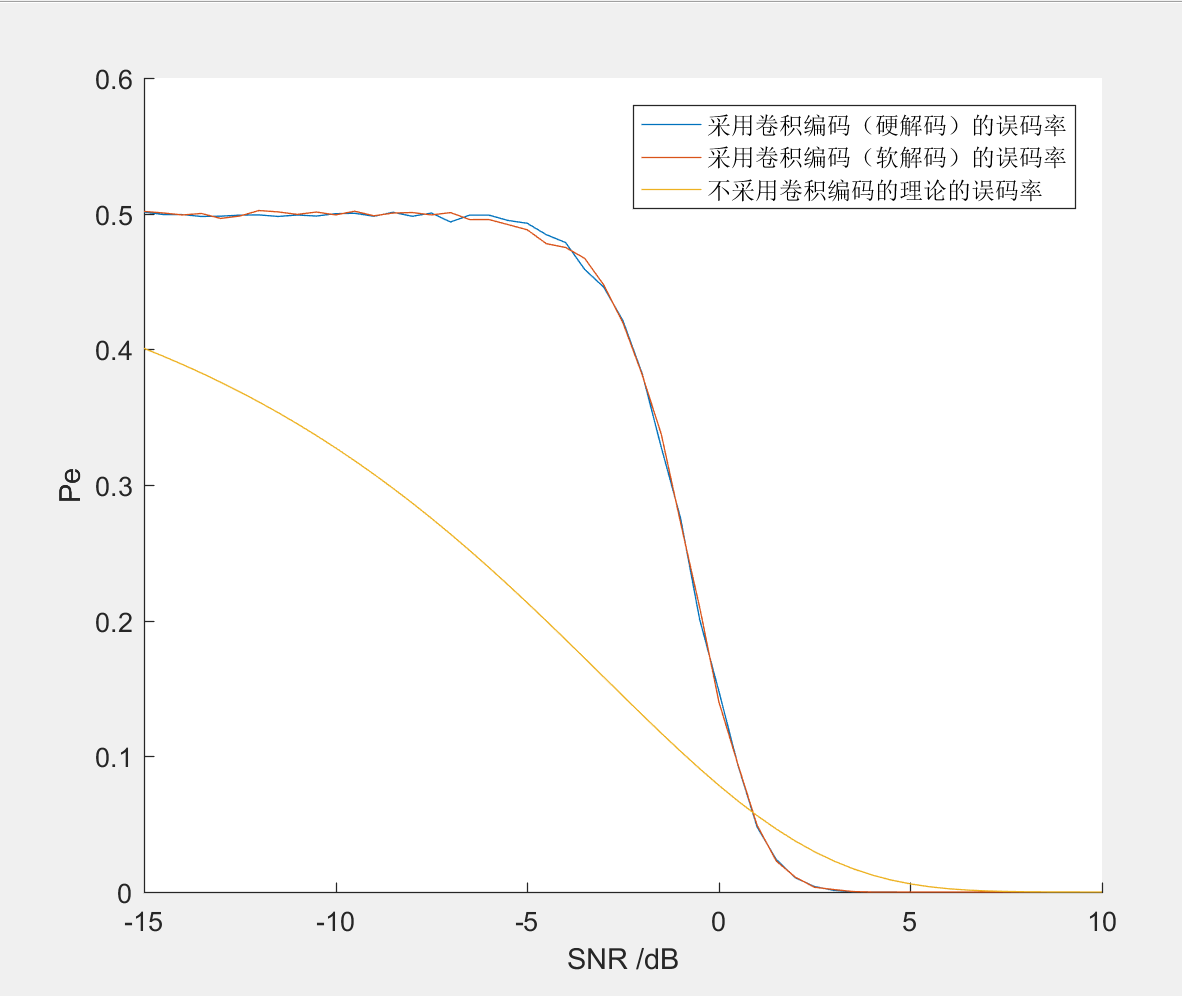
图表, 折线图

描述已自动生成

实验3：



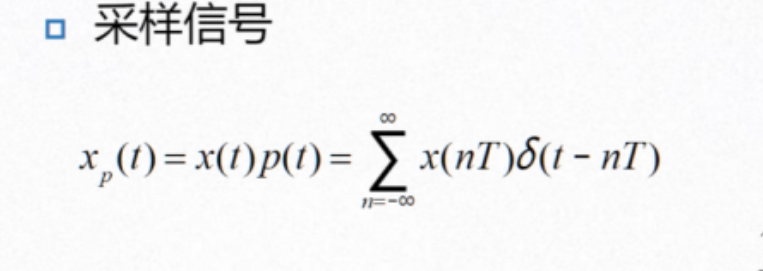
实验4：



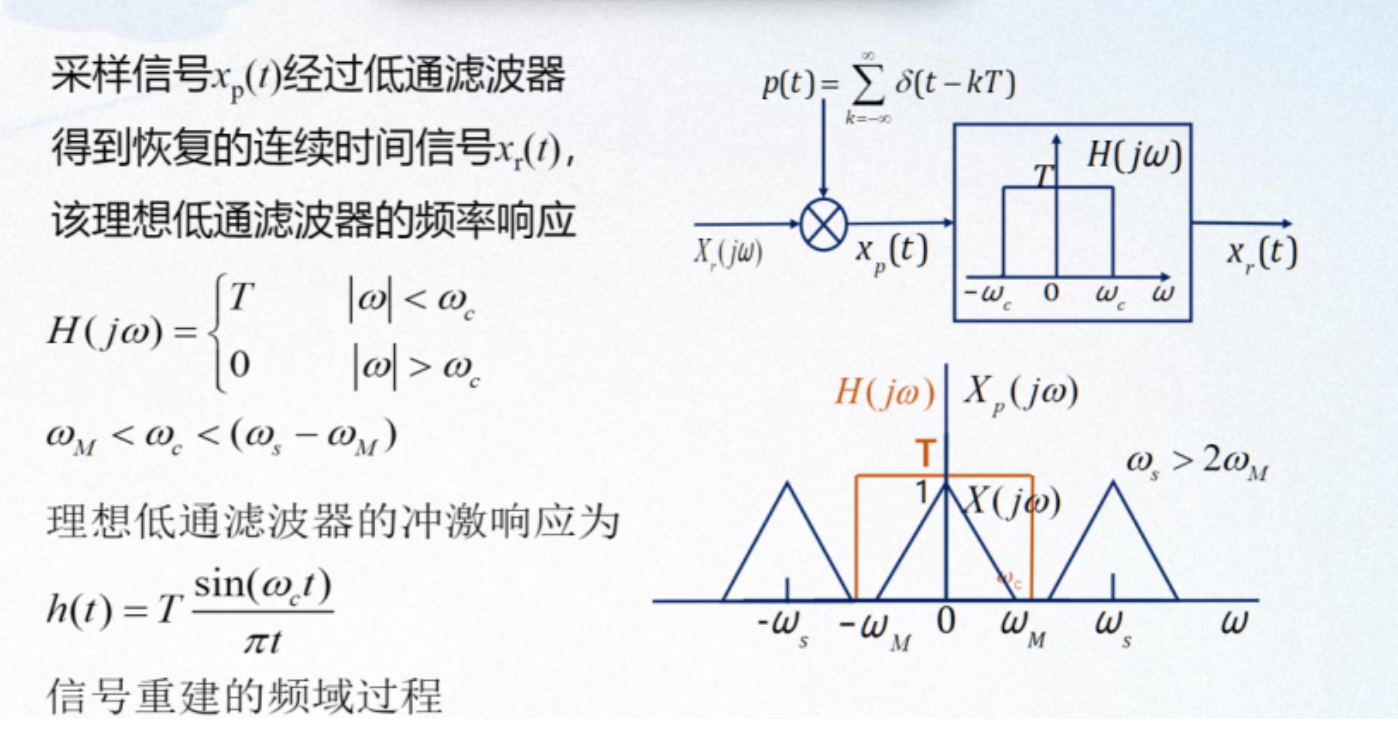
## 五、实验分析

1、实验一插值重构的公式

采样信号的公式



通过低通滤波器的频域表示和时域表示



插值重构的式子，所以最后通过时域插值重构的方法得到原来的信号：

图示

描述已自动生成

2、实验二：

可以看到PCM编码、译码之后的效果和很好，和原信号基本看不到区别点在哪里。

一个原因是这次采用的最小分度是Δ=1/4096，已经算精度很高了，基本和原信号吻合。

3、实验三：

**实现海明编码的方式是调用encode函数**

调用的方式是 :

encode（未编码比特串，一个码元比特总长，一个码元的信息位长，海明编码方式）；

**实现海明编码解码的方式是调用decode函数**

调用的方式是 :

decode（编码的比特串，一个码元比特总长，一个码元的信息位长，海明编码方式）；

图像可以看到，这里的误码率区别并没有教材P337的效果那么好，误码率只是稍稍低于不采用Haming编码的理论误码率，原因就是因为本身仿真的QPSK，误码率就比理论的误码率要高，所以相当于Haming编码使得误码率的降低，因为仿真的原因，又使得误码率增大。

4、实验四：

实验四生成的函数**调用convenc(data,trellis)**

其中包括两个参数，data是输入的比特序列，

**trell**是用网格的方式表示卷积码的一个结构体

通过trellis = poly2trellis(7,[171 133])；得到题目要求的网格

意思是约束长度是7，生成多项式是[171 133]

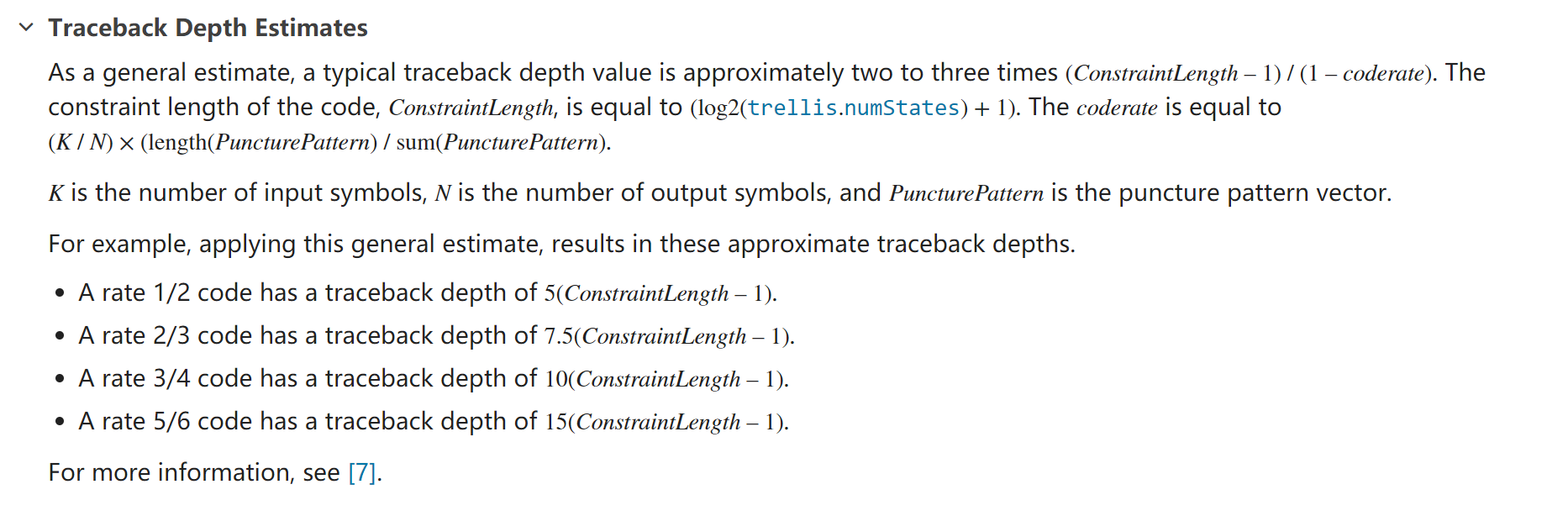
解码用到的是vitdec(decModuleData,trellis,tb,'trunc','hard');

表示解码的方式使用 Viterbi 算法，

decModuleData 待解码的比特序列

trellis 是生成的网格，和编码用的多项式相同，

tb代表回溯深度，其中对于不同信息率（码元信息比特位/码元总比特位）计算公式是



第四个参数opmode代表工作模式

* ‘Cont’代表上一次解码和下一次解码是有连续的
* ‘term’代表即将结束译码，也就是不再使用寄存器，这里要保证输入到convenc的信息后面有足够的0，使得所有在寄存器中的信息都能得到
* ‘trunc’是截断型工作模式，开始时0开始，结束直接截断，不会影响下一次编码。
* 这里使用截断型编码时合适的。

最后一个参数表示解码的方式是硬解码还是软解码