# 第四次实验报告

## 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。

## 实验1

### **1.1实验内容**

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。

### **1.2实验程序**

clear;

f = 1000; % 频率

dt = 1/f; % 时间间隔

t = -10:dt:10; % 时域

x = cos(0.15\*pi\*t)+sin(2.5\*pi\*t)+cos(4\*pi\*t); % 低通信号

fx = 4\*pi/(2\*pi); % 最高信号频率

subplot(311)

plot(t,x);

title('低通信号的波形');

xlabel('t');

ylabel('幅度');

grid on;

subplot(312)

fs = 4; % 抽样频率

x\_s = Sample(t,f,x,fs); % 对信号进行抽样

plot(t,x\_s);

title('抽样序列');

xlabel('t');

ylabel('幅度');

subplot(313)

t\_ = -20:dt:20; % 卷积所需时域

Sa = sinc(fs\*t); % 恢复信号

x\_ = conv(Sa,x\_s);

plot(t\_,x\_);

hold on;

plot(t,x);

title('恢复的信号与原信号比较');

legend('恢复信号','原信号');

xlabel('t');

ylabel('幅度');

axis([-10,10,-4,4])

### **1.3实验结果**

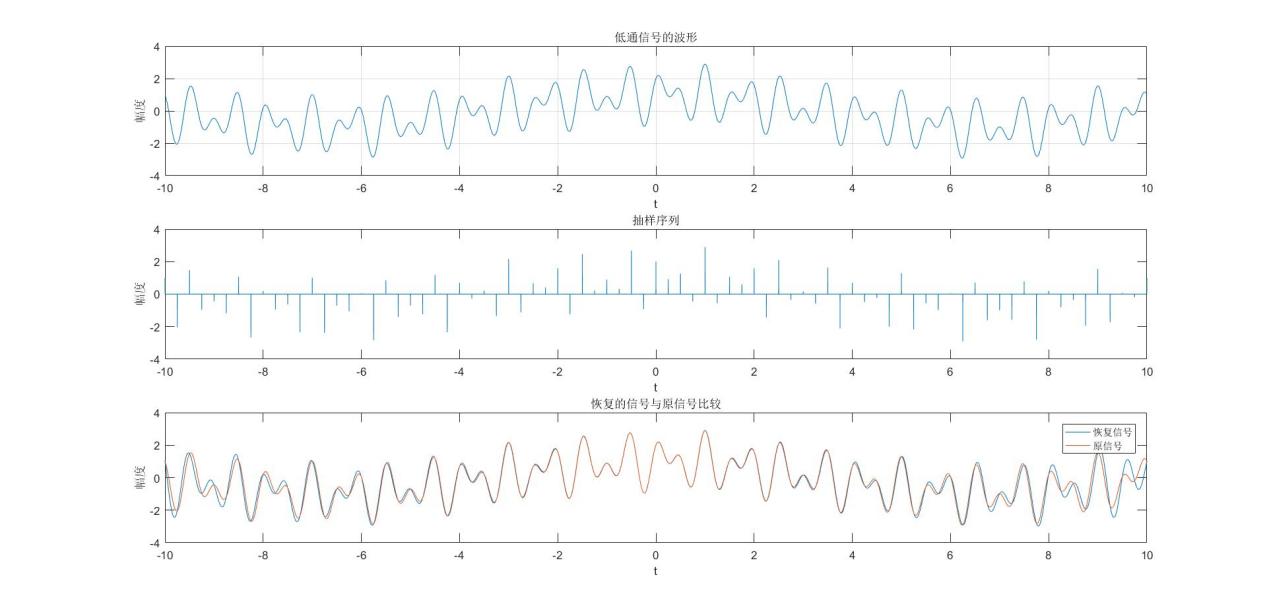


图 1 实验1结果图片

### **1.4实验分析**

根据所给信号可以得出信号的最大频率为2，故根据采样定理可以得到奈奎斯特采样频率为4Hz，这也正是题目（2）所给的采样频率，所以可以根据其采样得到的结果无失真的还原出原始信号，还原的公式为，其中为采样频率，为采样信号。但是由于我们的原本信号是由间隔很小的离散信号模拟的连续信号，所以在采样与还原过程中仍然存在少量失真，结果图片可以明显展示这一现象。

## 实验2

### **2.1实验内容**

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，PCM译码后的量化信噪比。

### **2.2实验程序**

clear;

f = 160; % 频率

dt = 1/f; % 时间间隔

t = -2:dt:2; % 时域

Ac = 1;

x = Ac\*sin(2\*pi\*t); % 输入信号

fs = 160;

x\_s = Sample(t,f,x,fs); % 抽样

[n,x\_PCM\_en] = PCM(t,x\_s,1,2048); % 编码

[~,x\_PCM\_de] = PCM(n,x\_PCM\_en,0,2048); % 译码

subplot(311)

plot(n,x\_PCM\_en);

title('PCM编码序列');

xlabel('N');

ylabel('幅度');

axis([0 100 0 1.2]);

grid on;

subplot(312)

plot(t,x\_PCM\_de);

hold on;

plot(t,x);

title('PCM译码与原信号对比图');

legend('PCM译码','原信号');

xlabel('t');

ylabel('幅度');

grid on;

Ac\_db = -70:0;

Ac = 10.^(Ac\_db/10);

s = zeros(1,length(Ac)); % 信号

u = zeros(1,length(Ac)); % 噪声

for i=1:1:length(Ac)

    x = sqrt(Ac(i))\*sin(2\*pi\*t);

    x\_s = Sample(t,f,x,fs);

    [n,x\_PCM\_en] = PCM(t,x\_s,1,2048); % PCM编码

    [~,x\_PCM\_de] = PCM(n,x\_PCM\_en,0,2048); % PCM译码

    u(i) = mean((x-x\_PCM\_de).^2);

    s(i) = mean(x\_PCM\_de.^2);

end

SNR = s./u;

subplot(313)

plot(Ac\_db,10\*log10(SNR));

title('不同幅度下PCM译码后的量化信噪比');

xlabel('A\_c');

ylabel('信噪比');

grid on;

### **2.3实验结果**

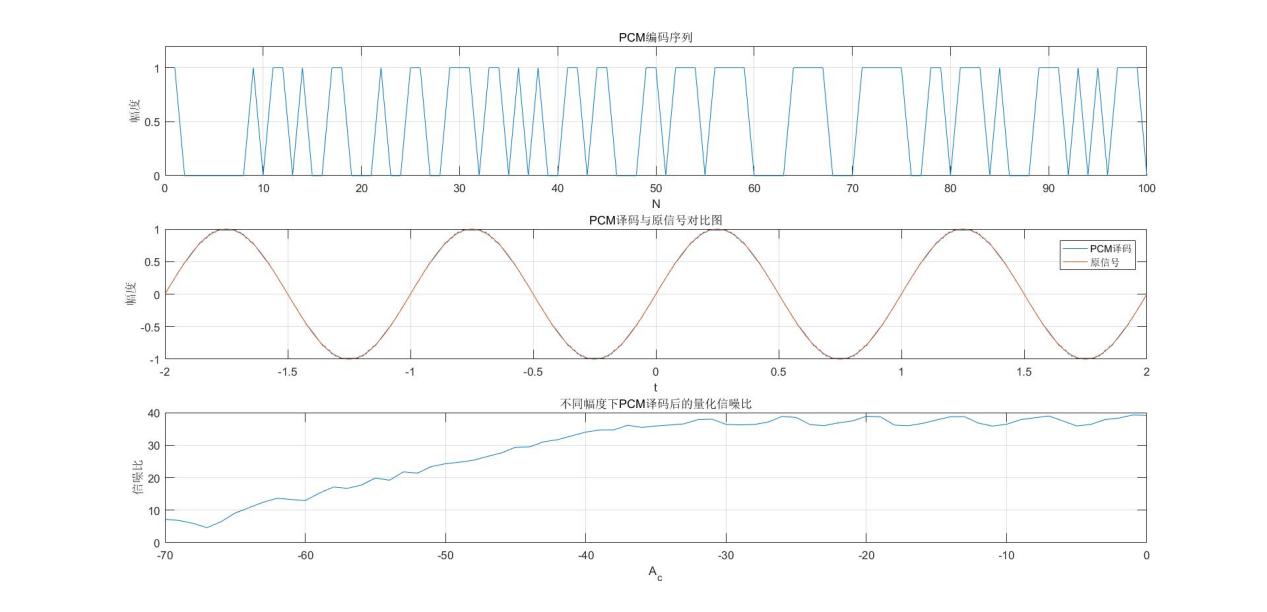


图 2 实验2结果图片

### **2.4实验分析**

使用抽样函数与PCM编码进行译码，在实验中要注意不论大小为多少，编码译码是都当作1V进行，否则得到的信噪比为定值。由图可以分析得到当从-70dB向0dB递增时，信噪比逐渐升高并最终趋于稳定。

## 实验3

### **3.1实验内容**

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，信噪比范围为0dB到10dB。

### **3.2实验程序**

clear;

N = 1000000; % 消息比特数

M = 4; % QPSK调制

n = 7; % 汉明编码码组长度

m = 3; % 汉明编码监督位长度

graycode = [0,1,3,2]; % 格雷编码规则

x = randi([0,1],N,n-m); % 消息

x1 = reshape(x',log2(M),N\*(n-m)/log2(M))'; % 消息重塑

x1\_de = bi2de(x1,'left-msb'); % 转成十进制

x1 = graycode(x1\_de+1); % 格雷编码

x1 = pskmod(x1,M); % QPSK调制

Eb1 = norm(x1).^2/(N\*(n-m)); % 计算比特能量

x2 = encode(x,n,n-m,'hamming/binary'); % 汉明编码

x2 = reshape(x2',log2(M),N\*n/log2(M))'; % 重塑编码后序列

x2 = bi2de(x2,'left-msb'); % 转成十进制

x2 = graycode(x2+1); % 格雷编码

x2 = pskmod(x2,M); % QPSK调制

Eb2 = norm(x2).^2/(N\*(n-m)); % 计算比特能量

SNR\_db = 0:10;

SNR = 10.^(SNR\_db/10);

for i=1:length(SNR\_db)

    sigma1 = sqrt(Eb1/(2\*SNR(i))); % 未编码的噪声标准差

    u1 = sigma1\*(randn(1,length(x1))+1j\*randn(1,length(x1))); % 噪声

    s1 = x1+u1;

    y1 = pskdemod(s1,M); % 未编码QPSK调制

    y1\_de = graycode(y1+1); % 未编码的格雷逆映射

    [e pe1(i)] = biterr(x1\_de',y1\_de,log2(M)); % 未编码的误比特率

    sigma2 = sqrt(Eb2/(2\*SNR(i))); % 编码的噪声标准差

    u2 = sigma2\*(randn(1,length(x2))+1j\*randn(1,length(x2))); % 噪声

    s2 = x2+u2;

    y2 = pskdemod(s2,M); % 编码QPSK调制

    y2\_de = graycode(y2+1); % 编码的格雷逆映射

    y2\_de = de2bi(y2\_de,'left-msb'); % 转换为二进制形式

    y2\_de = reshape(y2\_de',n,N)'; % 重塑

    y2\_de = decode(y2\_de,n,n-m,'hamming/binary'); % 译码

    [e pe2(i)] = biterr(x,y2\_de); % 未编码的误比特率

end

figure();

semilogy(SNR\_db,pe1,'-bo',SNR\_db,pe2,'-r\*');

title('(7,4)汉明编码的QPSK调制通过AWGN信道后的误比特率性能比较');

legend('未编码','(7,4)汉明编码');

xlabel('SNR(dB)');

ylabel('误比特率');

### **3.3实验结果**

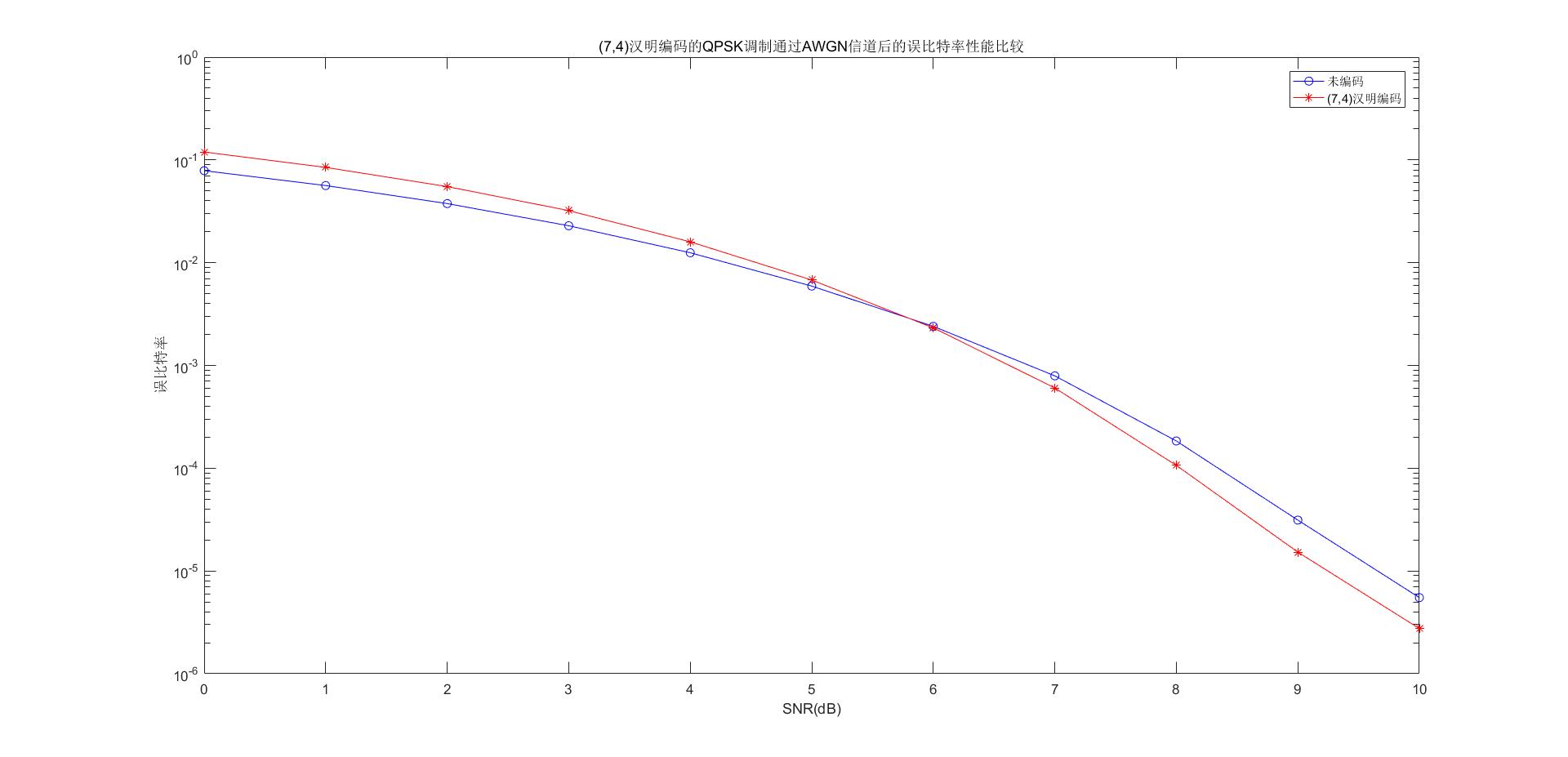


图 3 实验3结果图片

### **3.4实验分析**

（7，4）汉明编码具有检查2bit错误的能力与纠正1bit错误的能力, 信噪比范围为0dB到10dB时，可以看到当信噪比大于6dB的时候汉明编码系统比未编码系统的误比特律低，在信噪比较高时，汉明码的误比特率小于未编码的系统，而当信噪比较低时，往往会产生大于1bit的错误，所以导致汉明编码无法进行正确纠错，误比特率也相对的高。[MATLAB](https://so.csdn.net/so/search?q=MATLAB&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_43520653/article/details/_blank)提供了用Hamming码编码、解码的code和decode函数以及QPSK函数等，故本次实验中主要使用MATLAB自带的函数进行，以高效的运行。

## 实验4

### **4.1实验内容**

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，信噪比范围为0dB到10dB，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。

### **4.2实验程序**

clear;

SNR = 1:0.5:10; % 信噪比范围

N = 1000000; % 信息比特个数

M = 2; % BPSK调制

L = 7; % 约束长度

trel = poly2trellis(L,[171,133]); % 卷积码生成多项式

tblen = 6\*L; % Viterbi译码器回溯深度

x = randi([0,1],1,N); % 信息比特序列

x1 = convenc(x,trel); % 卷积编码

x\_ = pskmod(x1,M); % BPSK调制

for i = 1:length(SNR)

    % 加入高斯白噪声，因为码率为1/2，所以每一个符号的能量要比比特能量少3dB

    y = awgn(x\_,SNR(i)-3);

    y1 = pskdemod(y,M); % 硬判决

    y2 = vitdec(y1,trel,tblen,'cont','hard'); % Viterbi译码

    [err pe1(i)] = biterr(y2(tblen+1:end),x(1:end-tblen)); % 计算误比特率

    y3 = vitdec(real(y),trel,tblen,'cont','unquant'); % 软判决

    [err pe2(i)] = biterr(y3(tblen+1:end),x(1:end-tblen)); % 计算误比特率

end

ber = berawgn(SNR,'psk',2,'nodiff'); % BPSK调制理论误比特率

figure();

semilogy(SNR,ber,'-bd',SNR,pe1,'-go',SNR,pe2,'-r\*');

legend('BPSK理论误比特率','硬判决的误比特率','软判决的误比特率');

xlabel('SNR');

ylabel('误比特率');

### **4.3实验结果**

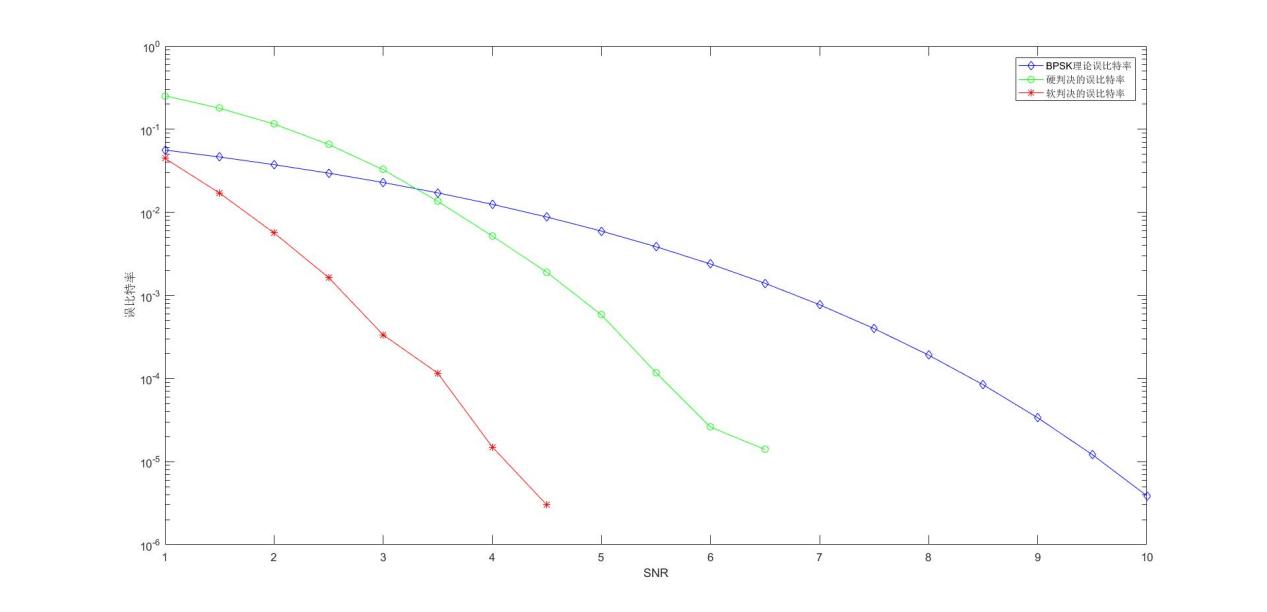


图 4 实验4结果图片

### **4.4实验分析**

本次实验的卷积编码解码和BPSK编码译码使用内置函数完成，并且可以通过结果观察到随着信噪比提升，误比特率在不断降低。在小信噪比情况下，编码的纠错能力超过了纠错门限以至于纠错码不再纠错，反而造成了更多的错误，所以硬判决误比特率先大于未编码误比特率，然后小于；维特比译码的软判决算法与硬判决算法相比，软判决译码算法的路径度量采用欧几里德距离而非汉明距离，计算复杂度更高但更加准确。此外，硬判决译码的判决过程损失了信道信息，所以软解码方式优于硬解码。

*附录：提供部分关键函数清单*

#### 采样函数

function out = Sample(t,f,s,fs)

% 抽样函数

%------------------------输入参数

% t：时域

% f：模拟频率

% s：输入信号

% fs：抽样频率

%---------------------输出(返回)参数

% out：抽样信号

gap = ceil(f/fs);

n = length(t);

out = zeros(1,n);

out(1:gap:n) = s(1:gap:n);

end

#### **PCM编码译码函数**

function [n,y] = PCM(t,s,mode,delta)

% A律13折线PCM编码

%*------------------------输入参数*

% t：时域

% s：信号

% mode：模式(1:编码,0:译码)

% delta：delta

%*---------------------输出(返回)参数*

% n：变换后序列号

% y：数字信号

if mode == 1

    %% 编码

    len = length(t);

    n = 1:1:8\*len;

    y = zeros(1,8\*len);

    for i = 1:8:8\*len

        y(i:i+7) = pcm\_e(s(floor(i/8)+1),delta);

    end

else

    %% 解码

    len = floor(length(t)/8);

    n = [];  % 空值

    y = zeros(1,len);

    for i = 1:1:len

        y(i) = pcm\_d(s((i-1)\*8+1:(i-1)\*8+8),delta);

    end

end

%% 编码子函数

    function y = pcm\_e(x,delta)

        % 数字编码

        y = zeros(1,8);

        %% 极性码

        if x > 0

            y(1) = 1;

        end

        x = abs(x);

        %% 段落码

        seg = floor(x\*delta);

        if 0 <= seg && seg < 16

            y(2)=0;y(3)=0;y(4)=0;step=1;st=0;

        elseif 16 <= seg && seg < 32

            y(2)=0;y(3)=0;y(4)=1;step=1;st=16;

        elseif 32 <= seg && seg < 64

            y(2)=0;y(3)=1;y(4)=0;step=2;st=32;

        elseif 64 <= seg && seg < 128

            y(2)=0;y(3)=1;y(4)=1;step=4;st=64;

        elseif 128 <= seg && seg < 256

            y(2)=1;y(3)=0;y(4)=0;step=8;st=128;

        elseif 256 <= seg && seg < 512

            y(2)=1;y(3)=0;y(4)=1;step=16;st=256;

        elseif 512 <= seg && seg < 1024

            y(2)=1;y(3)=1;y(4)=0;step=32;st=512;

        else

            y(2)=1;y(3)=1;y(4)=1;step=64;st=1024;

        end

        %% 段内码

        ise = floor((seg-st)/step);

        if ise < 16

            y(5:8) = (dec2bin(ise,4)-48);

        else

            y(5:8) = [1,1,1,1];

        end

    end

%% 译码子函数

    function y = pcm\_d(x,delta)

        %% 极性码

        pol = x(1)\*2-1;

        %% 段落码

        if x(2)==0&&x(3)==0&&x(4)==0

            step = 1;st=0;

        elseif x(2)==0&&x(3)==0&&x(4)==1

            step = 1;st=16;

        elseif x(2)==0&&x(3)==1&&x(4)==0

            step = 2;st=32;

        elseif x(2)==0&&x(3)==1&&x(4)==1

            step = 4;st=64;

        elseif x(2)==1&&x(3)==0&&x(4)==0

            step = 8;st=128;

        elseif x(2)==1&&x(3)==0&&x(4)==1

            step = 16;st=256;

        elseif x(2)==1&&x(3)==1&&x(4)==0

            step = 32;st=512;

        else

            step = 64;st=1024;

        end

        %% 段内码

        ise = x(5)\*8+x(6)\*4+x(7)\*2+x(8);

        seg = ise\*step+0.5\*step+st;

        y = pol\*seg/delta;

    end

end