2021春现代通信原理第二次实验报告

——模拟信号的数字传输

PB19071509 王瑞哲

>>>实验目的

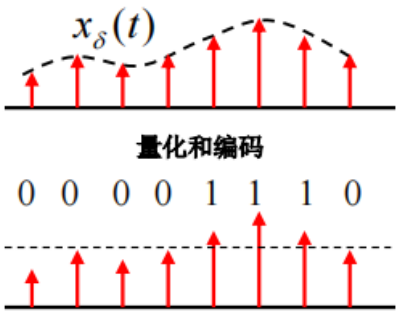
- 掌握低通信号抽样定理
- 理解 13 折线 A 率逐次比较型 PCM 编码仿真的思想
- 掌握 13 折线 A 律逐次比较型 PCM 编码、译码原理

>>>实验原理

一、脉冲编码调制

脉冲编码调制（Pulse Code Modulation），简称PCM。是对连续变化的模拟信号进行抽样、量化和编码产生的数字信号。

脉冲编码调制主要经过3个过程：**抽样**、**量化**和**编码**。抽样过程将连续时间模拟信号变为离散时间、连续幅度的抽样信号，量化过程将抽样信号变为离散时间、离散幅度的数字信号，编码过程将量化后的信号编码成为一个二进制码组输出。



实际使用中一般采用**非均匀量化**，这是一种对数形式的压缩特性，分为A律和U律。A律编码主要用于30/32路一次群系统，U律编码主要用于24路一次群系统。A律PCM用于欧洲和中国，U律PCM用于北美和日本。

二、低通抽样定理

一频带限制在$(0, f_H)$内的时间连续信号$m(t)$，若以$f_s \geq 2f_H$速率对$m(t)$等间隔$T_s = \frac{1}{f_s} \leq \frac{1}{2}f_H$抽样，则$m(t)$将被所得抽样函数$m_s(t)$完全确定。

抽样的过程是将输入的模拟信号与抽样信号相乘而得，通常抽样信号是一个周期为 Ts 的周期脉冲信号，抽样后得到的信号称为抽样序列。理想抽样信号定义如下：

$$\sigma_T(t) = \sum_n \delta(t - nT_s)$$

其中，$\delta(t) = \begin{cases} 1, & t = 0 \\ 0, & t \neq 0 \end{cases}$为抽样函数，$f_s = \frac{1}{T_s}$为抽样速率。因此抽样后的信号为：

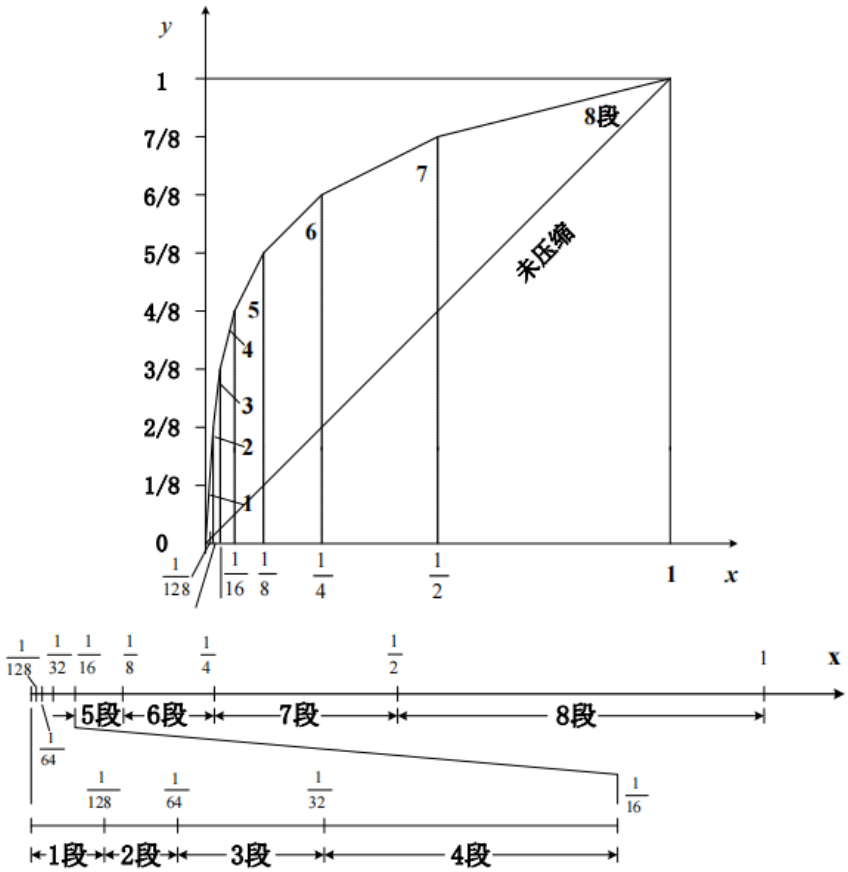
$$x_s(t) = x(t)\sigma_T(t) = \sum_{k=-\infty}^{\infty} x(kT_s)\delta(t - kT_s)$$

经带宽为f_H低通滤波器后即可恢复抽样信号。

三、逐次比较型PCM编码原理与方法

若采用均匀量化，则其量化信噪比随信号电平的减小而下降。产生这一现象的原因就是均匀量化时的量化级间隔Δ为固定值，而量化误差不管输入信号的大小均在(-Δ/2, Δ/2)内变化。故大信号时量化信噪比大，小信号时量化信噪比小。

对于语音信号来说，小信号出现的概率要大于大信号出现的概率，这就使平均信噪比下降。同时，为了满足一定的信噪比输出要求，输入信号应有一定范围(即动态范围)， 由于小信号信噪比明显下降，也使输入信号范围减小。**要改善小信号量化信噪比，可以采用量化间隔非均匀的方法，即非均匀量化。**



13折线A律PCM编码的码位安排如下：

极性码	段落码	段内码
M_1	$M_2M_3M_4$	$M_5M_6M_7M_8$

其中：

- 第一位M_1表示量化值的极性正负。$M_1 = 1$代表信号极性为正，$M_1 = 0$代表信号极性为负；后面7位分为段落码和段内码两部分，用于表示量化值的绝对值；
- $M_2M_3M_4$为段落码，分别对应下表段落编码
- $M_5M_6M_7M_8$为段内码，分别对应下表PCM编码的后四位

段落号	电平范围(q)	M_2	M_3	M_4	段落码对应起始电平(q)	量化间隔(q)	M_5对应电平(q)	M_6对应电平(q)	M_7对应电平(q)	M_8对应电平(q)
8	1024~2048	1	1	1	1024	**64**	512	256	128	64
7	512~1024	1	1	0	512	**32**	256	128	64	32
6	256~512	1	0	1	256	**16**	128	64	32	16
5	128~256	1	0	0	128	**8**	64	32	16	8
4	64~128	0	1	1	64	**4**	32	16	8	4
3	32~64	0	1	0	32	**2**	16	8	4	2
2	16~32	0	0	1	16	**1**	8	4	2	1
1	0~16	0	0	0	0	**1**	8	4	2	1

>>>实验内容

一、利用MATLAB验证低通抽样定理，若低通信号为$x(t) = 0.1\cos(0.15\pi t) + 0.5\cos(4\pi t)$，画出该低通信号的波形，画出抽样速率为$f_s = 4\text{Hz}$的抽样序列，再画出经低通滤波器恢复的波形。

编写MATLAB代码如下所示：

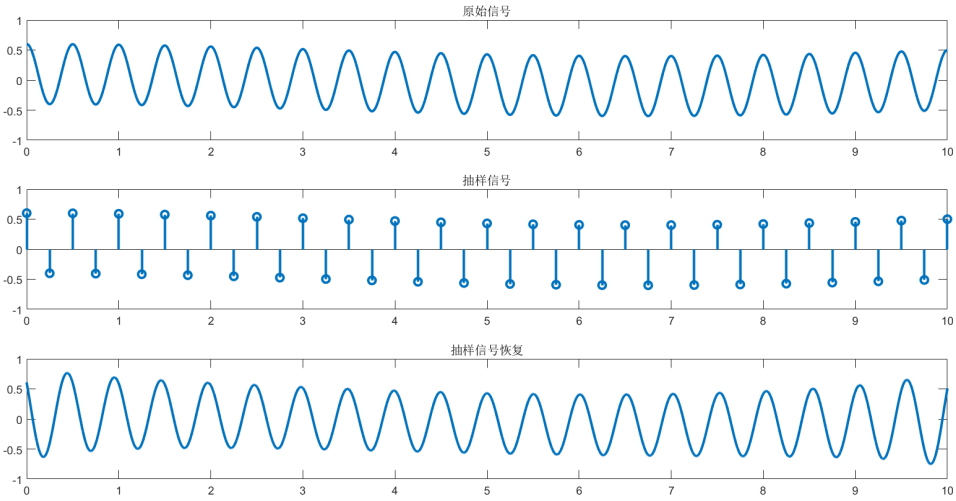
```
% 低通抽样定律
clear;
close all;
dt = 0.01;                % 时域步进
t = 0:dt:10;              % 时域范围：[0:10]，共取1001个点
xt = 0.1*cos(0.15*pi*t) + 0.5*cos(4*pi*t);
[f,xf] = T2F(t,xt);       % 求其频谱

% 抽样信号，抽样频率为4Hz
fs = 4;
sdt = 1/fs;
t1 = 0:sdt:10;
st = 0.1*cos(0.15*pi*t1) + 0.5*cos(4*pi*t1);
[f1,sf] = T2F(t1,st);

% 恢复原始信号
t2 = -50:dt:50;
gt = sinc(fs*t2);             % 抽样函数sinc，为频域门信号的傅里叶反变换
stt = sigexpand(st,sdt/dt);
xt_t = conv(stt,gt);          % 时域卷积恢复

figure(1);
subplot(311); plot(t,xt,'linewidth',2); title('原始信号'); axis([0 10 -1 1]);
subplot(312); stem(t1,st,'linewidth',2); title('抽样信号'); axis([0 10 -1 1]);
subplot(313);
```

```
t3= -50:dt:60+sdt-dt;
plot(t3,xt_t,'linewidth',2); title('抽样信号恢复'); axis([0 10 -1 1]);

% 计算均方误差
xt_t_c = xt_t(1,5001:6001);      % 截取长输出数组xt_t的0<t<10部分，以便和原信号xt对比
mse = mse(xt_t_c - xt)
```

其中，T2F.m 函数和 F2T.m 函数为傅里叶变换、反变换函数，与实验一中相同；sigexpand.m 函数可实现利用补零法对输入序列进行扩充，详细内容如下：

```
function [out] = sigexpand(d,M)
% 将输入d进行扩张，方法是在d的每个数据中插入M-1个零值。
%    如输入d=[1,2,3,4]; M=3;
%    输出out=[1,0,0,2,0,0,3,0,0,4,0,0]
    N = length(d);                % 基带信号码元长度
    out = zeros(M,N);             % 矩阵M为采样点   N为基带信号码元数量
    out(1,:) = d;                 % 将零矩阵第一列换成基带信号中的N个码元
    out = reshape(out,1,M*N);     % 重整为1行 m*n 列
end
```

运行结果如下图所示：



原信号$x(t) = 0.1\cos(0.15\pi t) + 0.5\cos(4\pi t)$为一个低频信号（0.075Hz）和一个高频信号（2Hz）的叠加，采样信号为4Hz，恰好满足奈奎斯特采样定理，因此上图中恢复信号与原信号基本相同。但注意到本次实验中原信号其实也是以离散形式展现的（采样点1001个），只能近似看作模拟信号，故所恢复的序列与原序列还是有微小的差别。为此本次实验中还设计了计算均方误差的代码，所计算的原信号与回复信号的均方误差为：

```
mse =

    0.0321
```

其中两序列x1，x2之间的均方误差定义为：

$$MSE(x_1, x_2) = \frac{1}{N}\sum_{i=1}^{N}(x_1 - x_2)^2$$

可见并不是无差别恢复，但在原信号的幅度约在0.5的情况下，0.0321的均方误差说明恢复效果已经相当好，误差可以近似忽略。

二、设输入一个样值 x∈[-2048，+2048]，对x进行A律PCM编码。要求编写成函数，该函数输入变量为样值，输出变量为A律13折线逐次比较的八位PCM编码，设码元宽度为1，画出其波形。

编写MATLAB代码如下所示：

```
clear;
x1 = -2*pi:0.4:2*pi;          x1_s = -2*pi:0.04:2*pi;
% 设置x1_s比x1采样点数多10倍，更接近模拟信号，单纯为了画图好看的目的，不用做信号处理
y1 = 2048*sin(x1);            y1_s = 2048*sin(x1_s);
w1 = A13_pcm_encoder(y1);                          % PCM编码
table1 = reshape(w1', 8, length(w1)/8)';           % 更方便地观看输出码字列表
y1_de = A13_pcm_decoder(w1,2048);                  % 解码输出
subplot(121);
plot(x1_s,y1_s,'b'); hold on                       % 绘制图像展示原模拟信号
stairs(x1,y1_de,'r','linewidth',2);                % 绘制阶梯图，更方便展示离散编码输出特
性
title('正弦信号PCM编解码');
xlabel('x'); ylabel('y');
legend('原始信号','PCM编解码信号','location','NorthEast');    % 修饰图像

x2 = -2.7:0.2:2.7;            x2_s = -2.7:0.02:2.7;
y2 = 2048*tanh(x2);           y2_s = 2048*tanh(x2_s);
w2 = A13_pcm_encoder(y2);                          % PCM编码
table2 = reshape(w2', 8, length(w2)/8)';           % 更方便地观看输出码字列表
y2_de = A13_pcm_decoder(w2,2048);                  % 解码输出
subplot(122);
plot(x2_s,y2_s,'b'); hold on
stairs(x2,y2_de,'r','linewidth',2);
title('双曲正切信号PCM编解码');
xlabel('x'); ylabel('y');
legend('原始信号','PCM编解码信号','location','SouthEast');
```

其中，A律13折线PCM编码函数 A13_pcm_encoder.m 如下所示：

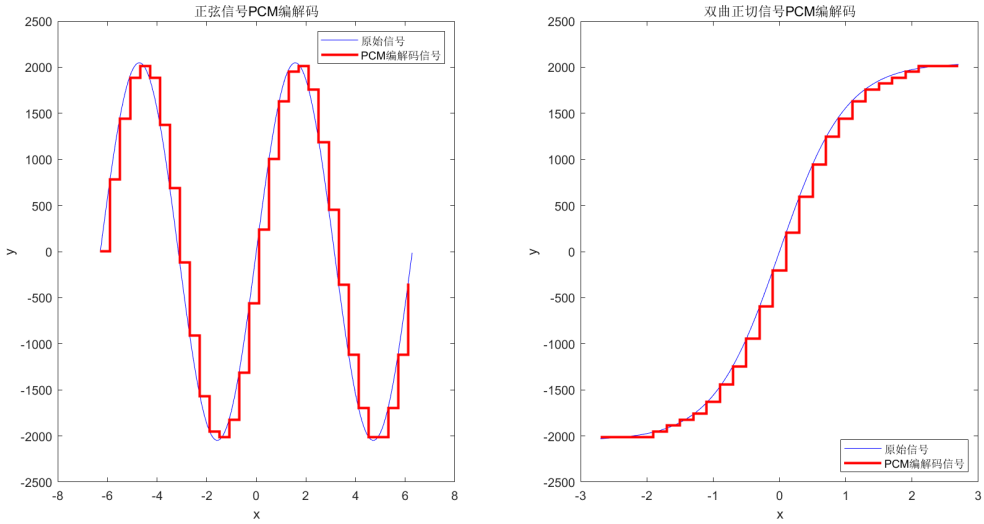
```
function [out] = A13_pcm_encoder(x)
% 对输入参数x进行A律13折线pcm编码，方法为逐次比较法
% 其中输入样值x的范围为[-2048, 2048]
    n = length(x);
    for i = 1:n
        % 编写极性码
        if x(i)>0
            out(i,1)=1;
        else
            out(i,1)=0;
        end

        % 编写段落码并计算量化间隔和量化起始电平
        if (0<=abs(x(i)) && abs(x(i))<16)
            % 段落码
            out(i,2)=0;out(i,3)=0;out(i,4)=0;
            % 量化间隔
            step = 1; %起始电平
            st = 0;
        elseif (16<=abs(x(i)) && abs(x(i))<32)
            out(i,2)=0; out(i,3)=0; out(i,4)=1; step=1; st=16;
```

```
        elseif (32<=abs(x(i)) && abs(x(i))<64)
            out(i,2)=0; out(i,3)=1; out(i,4)=0; step=2; st=32;
        elseif (64<=abs(x(i)) && abs(x(i))<128)
            out(i,2)=0; out(i,3)=1; out(i,4)=1; step=4; st=64;
        elseif (128<=abs(x(i)) && abs(x(i))<256)
            out(i,2)=1; out(i,3)=0; out(i,4)=0; step=8; st=128;
        elseif (256<=abs(x(i)) && abs(x(i))<512)
            out(i,2)=1; out(i,3)=0; out(i,4)=1; step=16; st=256;
        elseif (512<=abs(x(i)) && abs(x(i))<1024)
            out(i,2)=1; out(i,3)=1; out(i,4)=0; step=32; st=512;
        else
            out(i,2)=1; out(i,3)=1; out(i,4)=1; step=64; st=1024;
        end

        if (abs(x(i))>=2048) % 处理过载现象
            out(i,2:8)=[1 1 1 1 1 1 1];
        else
            % 计算段内码
            tmp = floor((abs(x(i))-st)/step);
            % 十进制转二进制数，
            % 如果不减48，最后4位是ASCII数49 48 48 48
            t = dec2bin(tmp,4)-48;
            out(i,5:8)=t(1:4);
       end
    end

    out = reshape(out',1,8*n);    % 输出：八位二进制码元长序列（一共n个码元，每个码元长8
位）
end
```

A律13折线PCM解码函数 A13_pcm_decoder.m 如下所示：

```
function [out] = A13_pcm_decoder(in,v)
% 对输入进行A律13折线pcm解码
% 输入in为8位二进制码，(-v,+v)为量化区间
    n = length(in);
    in = reshape(in', 8, n/8)';      % 重整原输入码序列--按每行8个二进制数字重整，一共n/8
个码字，每行即对应一个8位长码字
    slot = [0, 16, 32, 64, 128, 256, 512, 1024];         % 段落码解码手动打表
    step = [1, 1, 2, 4, 8, 16, 32, 64];                  % 段内码解码手动打表
    % 解码
    for i = 1:n/8
        ss = 2*in(i,1)-1;    % 解极性码
        tmp = in(i,2)*4+in(i,3)*2+in(i,4)+1;     % 解段落码
        st = slot(tmp);
        dt = (in(i,5)*8+in(i,6)*4+in(i,7)*2+in(i,8))*step(tmp)+0.5*step(tmp);
% 解段内码
        out(i) = ss*(st+dt)/2048*v;      % 重整为最终解码十进制数
    end
end
```

运行结果如下图所示：



其中，蓝线为原模拟信号，红线为在其上采样少量样值，经PCM编码再解码后的输出。原始模拟信号选取了正弦信号和tanh函数信号。为了更清楚地看到PCM编码过程，代码中还输出了编码表table。例如对于tanh函数信号，一共采样了28个样值，对应的8位PCM编码表为：

序号	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
1	0	1	1	1	1	1	1	1
2	0	1	1	1	1	1	1	1
3	0	1	1	1	1	1	1	1
4	0	1	1	1	1	1	1	1
5	0	1	1	1	1	1	1	0
6	0	1	1	1	1	1	0	1
7	0	1	1	1	1	1	0	0
8	0	1	1	1	1	0	1	1
9	0	1	1	1	1	0	0	1
10	0	1	1	1	0	1	1	0
11	0	1	1	1	0	0	1	1
12	0	1	1	0	1	1	0	1
13	0	1	1	0	0	0	1	0
14	0	1	0	0	1	0	0	1
15	1	1	0	0	1	0	0	1
16	1	1	1	0	0	0	1	0
17	1	1	1	0	1	1	0	1
18	1	1	1	1	0	0	1	1
19	1	1	1	1	0	1	1	0

（续表）

20	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**1**
21	1	1	1	1	1	0	0	1
22	1	1	1	1	1	1	0	0
23	1	1	1	1	1	1	0	1
24	1	1	1	1	1	1	1	0
25	1	1	1	1	1	1	1	1
26	1	1	1	1	1	1	1	1
27	1	1	1	1	1	1	1	1
28	1	1	1	1	1	1	1	1

仍以tanh函数信号为例，原采样值与解码输出值对比如下（一共28个采样值，仅选取部分正采样值列出，所有负采样值与之对称）：

原采样值	**...**	**204**	**597**	**946**	**1238**	**1467**	**...**	**2020**	**2029**
解码输出值	...	204	592	944	1248	1440	...	2016	2016
差值	...	0	5	2	10	27	...	4	4

两序列的平均绝对误差为：

```
>> mae(y2,y2_de)

ans =

   12.5636
```

其中两序列x1，x2之间的平均绝对误差定义为：

$$MAE(x_1,x_2)=\frac{1}{N}\sum_{i=1}^{N}|x_1-x_2|$$

>>>实验总结

通过本次实验，我对于低通信号采样定理与逐次比较法PCM编码原理有了更深层次的理解，同时亲自动手实践了信号抽样与恢复过程，根据PCM编码函数、解码函数实践了实际信号的编解码过程，并利用MATLAB展示之。

在本次实验中，我深刻体会到了误差计算在信号编解码中的重要作用。本次实验中，无论是低通信号的抽样与恢复，还是PCM编解码过程，都存在变换→恢复的过程。这也是模拟信号的数字传输中，不可避免的一个环节。本次实验中使用了均方误差MSE和平均绝对误差MAE来衡量前后序列的差别，利用MATLAB计算也非常便捷，这有助于我们对于信号变换过程原理性正确的进一步理解。

在本次实验中遇到的几个小问题仍然和矩阵运算基本问题有关。计算低通采样信号恢复序列的过程中，要特别注意卷积序列所导致的序列长度增加（N长序列与M长序列卷积结果长为N+M-1），这导致最终恢复序列比原始定义的序列要长；PCM编码函数生成的序列为1×（8×采样点个数）的行向量（长01序列），可以用reshape方法将其转变为 采样点个数×8 的矩阵，这样每行就恰好就代表一个采样点的8位码字。这也提醒我在MATLAB软件操作中，一定要格外注意有关于矩阵维数运算的各种方面，尽量避免出错。