通信原理
MATLAB 软件作业二

PCM 均匀量化与 A 律 13 折线编码解码的 MATLAB 仿真实验

学　　院　　电子信息工程学院

作　　者　　何 沃 洲

学　　号　　13021264

2016 年 5 月

目　录

（一）PCM 原理

脉冲编码调制(PCM)，是一种将模拟信号的抽样量化值变换成代码的编码方式。编码调制的过程如图 1 所示。PCM 实现主要包括三个步骤：抽样、量化和编码。

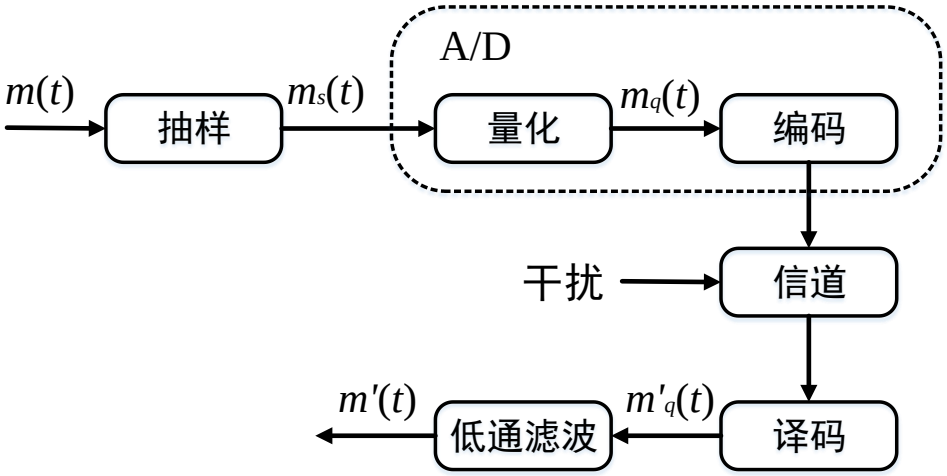

图 1　PCM 系统原理框图

所谓抽样，就是对模拟信号进行周期性扫描，把时间上连续的信号变成时间上离散的信号。该模拟信号经过抽样后还应当包含原信号中所有信息，也就是说能无失真的恢复原模拟信号。它的抽样速率的下限是由奈奎斯特抽样定理确定的。

量化把一个连续幅度值的无限数集合映射成一个离散幅度值的有限数集合，可分为均匀量化和非均匀量化。均匀量化存在的主要缺点是：无论抽样值大小如何，量化噪声的均方根值都固定不变。因此，当信号 $m(t)$较小时，则信号量化信噪比也就很小，这样，对于小信号时的量化信噪比就难以达到给定的要求。可见均匀量化时的信号动态范围将受到较大的限制。为了改善这个缺点，实际中往往采用非均匀量化。

非均匀量化是根据信号的不同区间来确定量化间隔的。对于信号取值小的区间，其量化间隔 Δ 也小；反之，量化间隔就大。它与均匀量化相比，改善了小信号的量化信噪比，实际相当于将抽样值通过了压缩再做均匀量化。通常使用的压缩器大多采用对数式压缩。根据 CCITT 的建议，A 律和 μ 律是广泛采用的两种对数压缩律。本文仿真中所涉及的 PCM 编码采用近似于 A 律规律的 13 折线（A=87.6）压扩特性

来进行编码，它基本上保持了连续压扩特性曲线的优点。

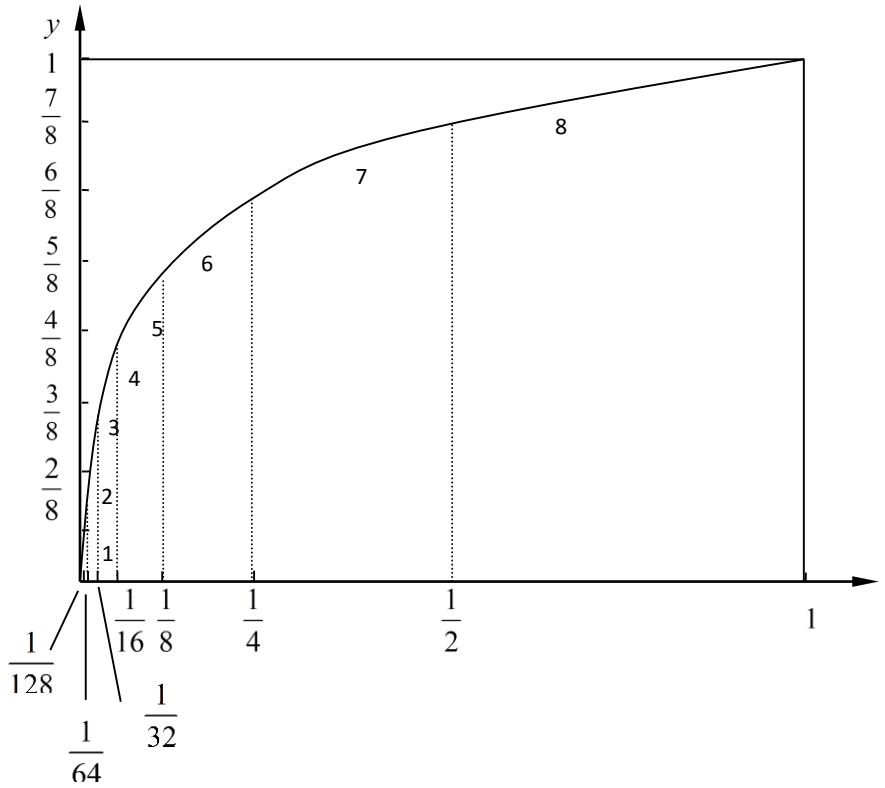

图 2　A 律 13 折线压扩特性

所谓编码就是把量化后的信号变换成代码，其相反的过程称为译码。在 A 律 13 折线法中，按 8 段折线（8 个段落）进行编码。第 1 位码表示极性（“1”代表正极性，“0”代表负极性）。第 2~4 位码表示段落码，它的 8 种可能状态分别代表 8 个段落的起始电平。第 5~8 位码表示段内码，16 种可能状态分别代表每一段落内 16 个等间隔划分的量化级。量化间隔随段落序号的增加以两倍递增。

极性码　段落码　　段内码

M_1　　$M_2M_3M_4$　$M_5M_6M_7M_8$

段落序号	段落码
8	111
7	110
6	101

量化级	段内码
15	1111
14	1110
13	1101
12	1100
11	1011
10	1010

5	100		9	1001
			8	1000
4	011		7	0111
			6	0110
3	010		5	0101
			4	0100
2	001		3	0011
			2	0010
1	000		1	0001
			0	0000

表 1　A 律 13 折线编码的段落码和段内码

设输入信号电平为$x\Delta$，在第 k 段内，

则段内码$(M_5M_6M_7M_8)_2 = (\frac{x\Delta-\text{第}k\text{段的起始电平}}{\text{第}k\text{段的量化间隔}\Delta_k})_{10}$

解码时，码字电平 = 段落起始电平 $+(8M_5+4M_6+2M_7+1M_8)\cdot\Delta_k$

解码电平 = 码字电平$+\Delta_k/2$

（二）MATLAB 程序及仿真结果

2.1 随机信号的产生

生成频率和幅度都在随机变化的输入信号数据序列（封装成函数 signal_generator）：

```
function [ signal ] = signal_generator( fm,dt )
flag = 1; %指示位
n =  20;   %拼接段
num = 1/(0.25*fm)/dt*n;        %原始信号序列的最高点数,由n段频率幅度随机的正弦构成
length = zeros(1,n);           %记录每段正弦的采样点数
array = zeros(1,round(num)); %原始信号数据序列
f = rand(1,n)*(fm-0.25*fm)+0.25*fm; %频率随机变化的范围（0.25*fm~fm）
A = rand(1,n)*32+1;  %平均功率动态范围30dB，折合幅度变化最大值约为最小值32倍
for i = 1:n %逐段产生原始信号序列
    time_scale = zeros(1,500); %每段正弦的最大采样点数
    length(i) = floor(1/(2*f(i))/dt);  %每段正弦的实际采样点数
    time_scale(1:length(i)) = dt:dt:(dt*length(i)); %每段正弦的时间尺度（多余补零）
    if (mod(i,2)==0) %对每段的正弦波形进行处理
```

```
        sine = A(i)*sin(2*pi*f(i)*time_scale);
    else
        sine = -A(i)*sin(2*pi*f(i)*time_scale);
    end
    array(flag:(flag+length(i)-1)) = sine(1:length(i)); %各段正弦拼接
    flag = flag+length(i); %跳到下一段拼接的位置
end
signal_length = round(1/(2*fm)/dt*n); %取原始信号序列前2500个点作为输入
signal = array(1:signal_length);
end
```

【注解】

该段程序利用若干段频率和幅值都在随机变化的半周期正弦拼接生成一段时长为 25ms 的随机信号，其中正弦的频率在 $0.25f_m \sim f_m$ (即 1kHz~4kHz)之间、幅值在 30dB 的功率动态范围中随机生成。

主要思路是先在预设范围内随机取每段正弦的频率和幅值。然后设置指示变量 flag，逐段向信号数据序列中填充随机的正弦信号，截取前 25ms 作为输入信号。

某次随机生成的输入信号波形如图 3 所示。

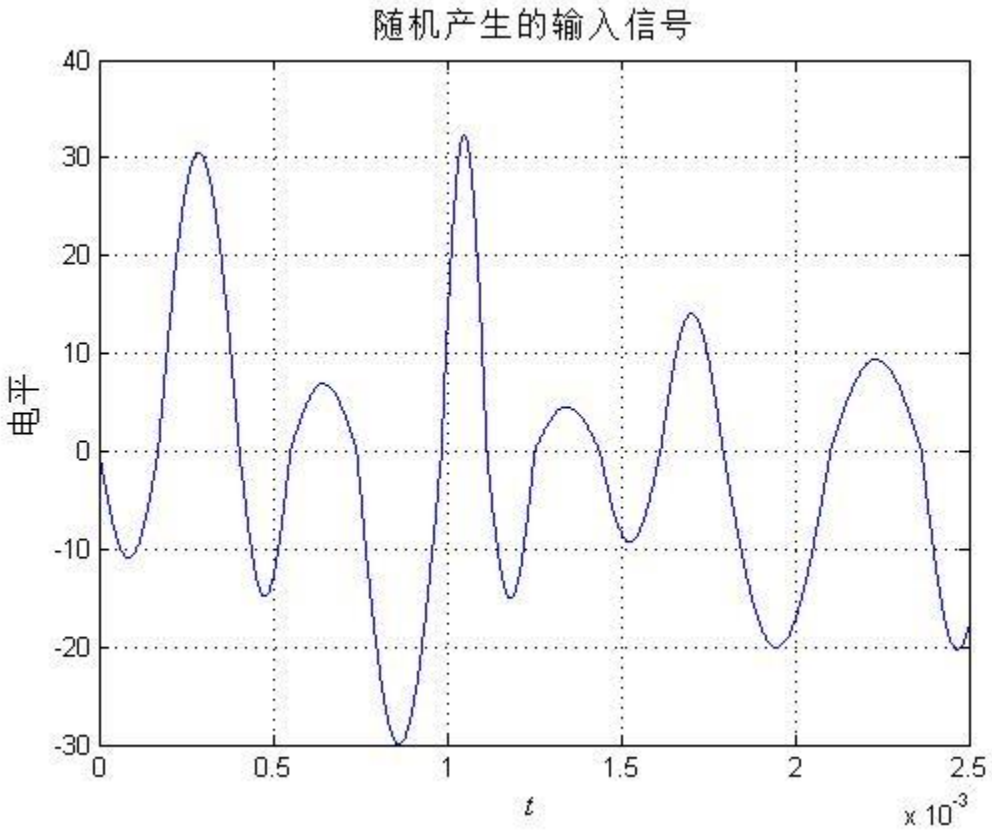

图 3　随机生成的输入信号

2.2 均匀量化器

输入原始信号序列，返回均匀量化信号序列及线性 PCM 编码结果（封装成函数 uni_quantizer）：

```
function [ output,code ] = uni_quantizer( input,n )
input_max = max(abs(input)); %求输入峰值
input_norm = input/input_max;%输入归一化
length = numel(input); %求输入的采样点数
output = zeros(1,length);
delta = 2/n; %归一化量化间隔
section = (0:(n-1))*delta-(n-1)/2*delta; %各量化区间的中点
bit = ceil(log2(n)); %均匀量化的比特数
code = zeros(length,bit); %编码结果矩阵
for i = 1:length  %逐点量化
    for k = 1:n
        if
(((input_norm(i))<=section(k)+delta/2)&&(input_norm(i)>=section(k)-
delta/2))
            output(i) = section(k)*input_max;
            %下面进行均匀量化编码
            m = k-2^(bit-1);
            %确定符号位
            if m>=0
                code(i,1) = 1;
            else
                code(i,1) = 0;
            end
                m_abs = abs(m);
            %从高位到低位逐位确定1或0
            for j = (bit-1):-1:1
                if (m_abs>=2^(j-1))
                    code(i,bit-j+1) = 1;
                    m_abs = m_abs-2^(j-1);
                else
                    code(i,bit-j+1) = 0;
                end
            end
        end
    end
end
end
```

【注解】

根据题目所给的条件，对正弦信号进行满载的均匀量化时

量化信噪比 $[S/\sigma_q^2]_{\max\ dB} = 6.02n + 1.76$

动态范围 $D = 30$dB，要求信噪比不低于 25dB，则 $6.02n + 1.76 \geq 30 + 25$ (dB)

可得均匀量化的位数$n \geq 8.84$，即最小量化位数应取为 9 才能达到要求。

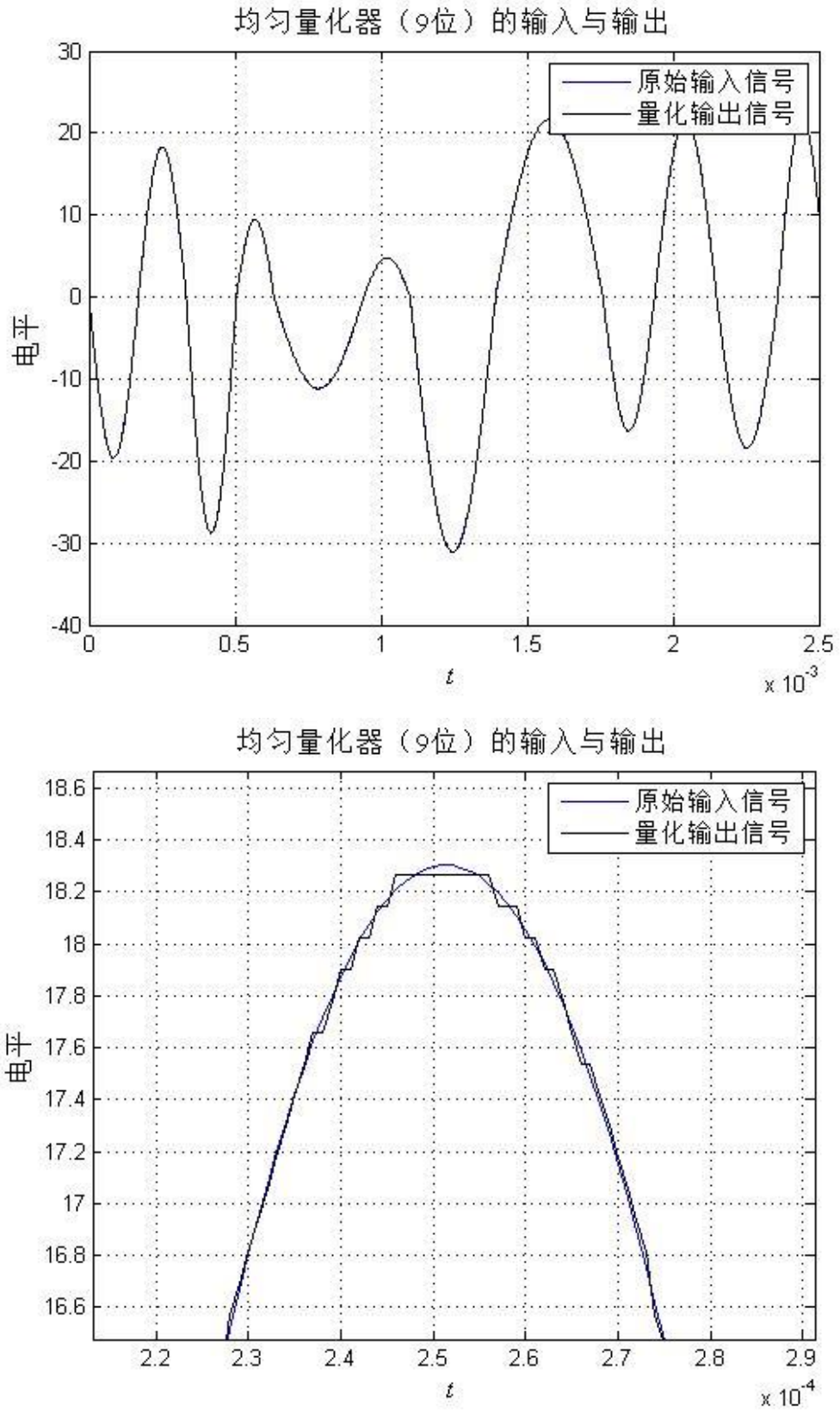

图 4　均匀量化器的输入与输出

该段程序进行输入信号的均匀量化。首先把信号范围归一化到区间[-1,1]，然后根据规定的均匀量化位数进行量化间隔的划分，计算每个量化区间对应的判决值。紧接着对抽样点逐个进行量化并记录对应点所属的量化阶。最后对量化阶进行编码（先确定符号位，然后对幅度值进行二进制编码）。函数返回均匀量化后的信号及编码结果（编码结果在主函数 PCM_main. m 中输出）。

原始输入信号和均匀量化输出信号的波形如上面的图 4 所示：

可以看出，由于量化位数足够多，粗略看量化输出信号已经与原输入信号基本重合，但放大看依然可以观察到量化“台阶”。在主函数中计算得到该次 9 位均匀量化的量化信噪比达到 52.2dB，平均量化误差为 0.0286（由于组成随机信号的每个正弦不一定都能取到动态范围中的最极端情况，所以实际信噪比远高于题目的最低要求 25dB）。

2.3 非均匀量化（A 律 13 折线 PCM 编码与解码）

该模块包括两个部分，编码器与解码器（分别封装成函数 `A13nonuni_quantizer` 和 `A13nonuni_decoder`）：

```
function [ code ] = A13nonuni_quantizer( input )
input_max = max(abs(input)); %求输入峰值
input_init = 4096*input/input_max; %输入归一化
length = numel(input); %求输入的采样点数
code = zeros(length,8);%编码矩阵
input_norm = zeros(1,length);
%取整
for i = 1:1:length if input_init(i) >= 4096
       input_norm(i) = 4096;
else
       input_norm(i) = round(input_init(i));
end
value = abs(input_norm(i));
%确定符号位
if (input_norm(i)>0)
    code(i,1) = 1;
else
    code(i,1) = 0;
end
%确定段落码
if (value>=0) && (value<32)
    code(i,2)=0;code(i,3)=0;code(i,4)=0;step=2;range=0;
elseif (value>=32) && (value<64)
    code(i,2)=0;code(i,3)=0;code(i,4)=1;step=2;range=32;
elseif (value>=64) && (value<128)
```

```
    code(i,2)=0;code(i,3)=1;code(i,4)=0;step=4;range=64;
elseif (value>=128) && (value<256)
    code(i,2)=0;code(i,3)=1;code(i,4)=1;step=8;range=128;
elseif (value>=256) && (value<512)
    code(i,2)=1;code(i,3)=0;code(i,4)=0;step=16;range=256;
elseif (value>=512) && (value<1024)
    code(i,2)=1;code(i,3)=0;code(i,4)=1;step=32;range=512;
elseif (value>=1024) && (value<2048)
    code(i,2)=1;code(i,3)=1;code(i,4)=0;step=64;range=1024;
else
     code(i,2)=1;code(i,3)=1;code(i,4)=1;step=128;range=2048;
end
%确定段内码
duan = floor((value-range)/step);
for j = 5:1:8
    if (duan>=2^(8-j))
       code(i,j) = 1;
       duan = duan-2^(8-j);
    else
       code(i,j) = 0;
    end
end
end
end
```

```
 ----------------------------------------------------------------------
function [ output ] = A13nonuni_decoder( code,maximum )
step = [2 2 4 8 16 32 64 128]; %各分段的量化间隔
range = [0 32 64 128 256 512 1024 2048]; %各分段的区间起点
length = size(code,1); %获取编码点数
output = zeros(1,length); %输出矩阵
section = zeros(1,length);
for i = 1:1:length
  sign = 2*code(i,1)-1; %符号位译码
  section(i) = code(i,2)*4+code(i,3)*2+code(i,4)*1+1; %段落码译码
  origin = range(section(i)); %区间起点
  stepsize = step(section(i));%区间步长
  output(i) =
sign*((code(i,5)*8+code(i,6)*4+code(i,7)*2+code(i,8)*1+0.5)*stepsize+or
igin)/4096*maximum; %段内码译码，转化为输出电平
end
end
```

【注解】

编码的主要思路是先把输入信号归一化到区间[-1,1]，然后等间隔划分为4096个量化区间 ($\Delta = 2/4096$)。根据抽样点的极性确定符号位M_1，再根据幅度值所在的区间确定段落码$M_2M_3M_4$，根据段落内等间隔划分的区间确定段内码$M_5M_6M_7M_8$。

译码实际上就是上面的逆过程。根据符号位确定极性，由段落码得到抽样点所在的区间起点和量化间隔，通过段内码得到恢复电平值。编码结果在主函数输出。

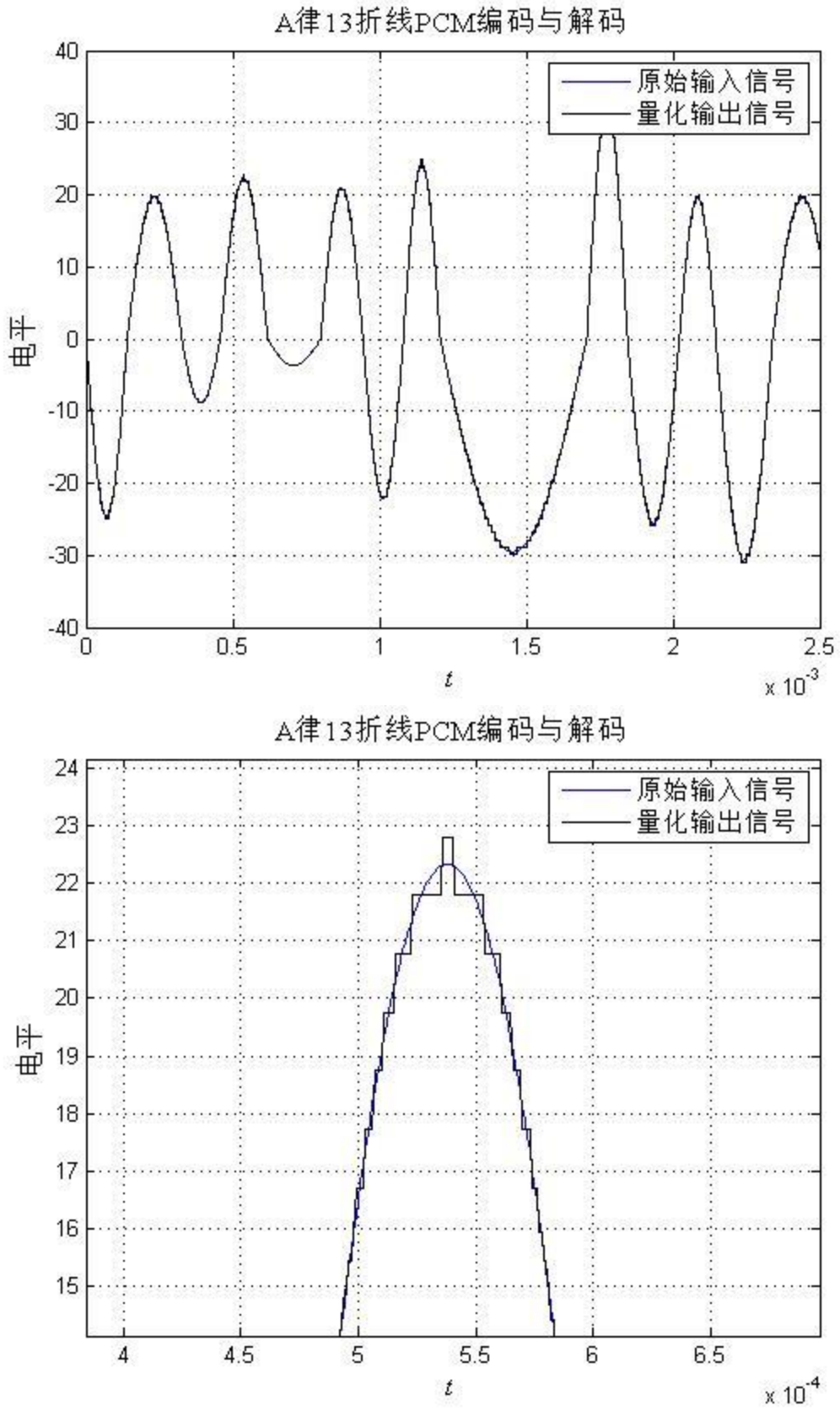

图 5　A 律 13 折线 PCM 译码后的波形

图 4 所示为原输入信号波形与 A 律 13 折线 PCM 译码后的波形对比。译码后的输出信号基本和原输入信号重合。在主函数中计算得到该次 A 律 13 折线量化的信噪比达到 37.8dB，译码后波形与原信号的平均量化误差为 0.135（每个抽样点量化误差的绝对值取平均）。由于该仿真的输入以幅度中等的信号为主，所以与相同位数的均匀量化器相比，A 律 13 折线 PCM 的优势并不明显。

（三）结论与收获

本次实验分别对 PCM 两种量化编码方式，均匀量化和 A 律 13 折线 PCM 进行了 MATLAB 仿真，结果与理论的分析基本一致，通过仿真更加巩固了我对 PCM 编码解码原理的认识和理解。

仿真的最大好处在于，为了完美地实现一个系统的功能，必须毫不含糊地掌握系统的每一个细节，每一步小的差错都可能导致结果的偏差。虽然在课上学习和课后计算 PCM 编码结果及误差的时候觉得自己差不多都掌握了，但是到自己仿真实现整个 PCM 系统的时候才发现其实很多细节都是很关键的。例如做 A 律 13 折线求段内码的时候应该向下取整得到对应的电平码字，但我开始时随意地取了四舍五入，结果解码时加上 $\Delta_k/2$ 反而放大了误差，令我费解了好一会。所以，做仿真既是对已有理论的检验论证，更是深入学习、清除盲点的一个过程。其实很多知识都是统一的，只有通过多动手多实践才能领会到它们内在的一致性。

附录

Main 函数 PCM_main. m :

```
%-------------------------------------------------------------------------
% 产生随机输入信号
%-------------------------------------------------------------------------
clear all
dt = 1e-6; %采样周期
fm = 4e3; %信号最高频率
signal = signal_generator(fm,dt); %输入信号产生
length = numel(signal); %输入信号的采样点数
t = dt:dt:(dt*length); %仿真2.5ms(采样频率为1MHz时)
input_max = max(abs(signal)); %求输入信号的最大值

%-------------------------------------------------------------------------
% 均匀量化
%-------------------------------------------------------------------------
```

```
n = 2^9; %均匀量化电平数
[uni_output,code] = uni_quantizer(signal,n);
%均匀量化器，输出量化信号及编码矩阵
uni_error = mean(abs(signal-uni_output)); %计算平均量化误差
uni_snr = 20*log10(norm(signal)./norm(signal-uni_output));
%计算均匀量化信噪比
disp('均匀量化编码为：');
disp(code);
%作图
figure(1);
subplot(2,1,1);
plot(t,signal,'-b');
hold on
plot(t,uni_output,'k');
hold on
legend('原始输入信号','量化输出信号');
grid
xlabel('\itt','FontName','Times New Roman','FontSize',12);
ylabel('µçÆ½','FontName','Times New Roman','FontSize',12);
title('均匀量化器的输入与输出','FontName','Times New Roman','FontSize',12);

%--------------------------------------------------------------------------
% 非均匀量化:A律13折线PCM编码与解码
%--------------------------------------------------------------------------
[A13_code] = A13nonuni_quantizer(signal); %A律13折线PCM编码
[A13nonuni_output] = A13nonuni_decoder(A13_code,input_max); %A律13折线
PCM解码
A13nonuni_error = mean(abs(signal-A13nonuni_output)); %求译码后的平均量化误
差
A13nonuni_snr = 20*log10(norm(signal)./norm(signal-A13nonuni_output));
%计算A律13折线PCM编码解码的量化信噪比
disp('A律13折线PCM编码为：');
disp(A13_code);
disp('均匀量化(9位)和A律13折线PCM编码结果如上');
%作图
subplot(2,1,2);
plot(t,signal,'-b');
hold on
plot(t,A13nonuni_output,'k');
hold on
legend('原始输入信号','量化输出信号');
grid
xlabel('\itt','FontName','Times New Roman','FontSize',12);
ylabel('电平','FontName','Times New Roman','FontSize',12);
title('A律13折线PCM编码与解码','FontName','Times New
Roman','FontSize',12);
%输出两者的译码后平均误差
disp(['均匀量化(9位)平均误差：',num2str(uni_error)]);
disp(['A律13折线译码平均误差：',num2str(A13nonuni_error)]);
%输出两者的量化信噪比
disp(['均匀量化(9位)信噪比为：',num2str(uni_snr),'dB']);
disp(['A律13折线量化信噪比为：',num2str(A13nonuni_snr),'dB']);
```