MATLAB 实验　数字音频处理

杨凯欣　1500012805　Tel:18811318106

2017 年 5 月 30 日

目录

§1 实验要求

对给出的音频进行量化和均衡处理，并对结果进行分析.

具体步骤：

1. 示例音频 music.wav，读入 wav 文件：wavread()；

2. 做出示例音频的时域波形及频谱图；

3. 选择量化级数为 8(每符号 8bit, 共 2^8 个量化数目)，均匀量化，做出量化后的波形图和频谱图；

4. 同时对示例音频进行非均匀量化 (A 律、μ 律)，做出非均匀量化后的波形图和频谱图；

5. 解量化，比较量化后的语音效果，做出必要的波形图；

6. 计算解压后的信噪比；

7. 对以上各步处理后的音频写入 wav 文件，试听实际的语音效果.

§2 实验原理

2.1 脉冲编码调制

1. 脉冲编码调制 (Pulse-code modulation,PCM)，是对连续变化的模拟信号进行抽样、量化和编码后产生的数字信号.

   线性脉冲编码调制 (Linear pulse-code modulation,LPCM) 是一种采用线性均匀量化方式的 PCM，区别于幅度 (Amplitude) 以特定函数形式变化的 PCM 编码方式 (如:A 律 (A-law algorithm) 和 μ 律 (μ-law algorithm)).

   PCM 的两个重要参数是：采样率 (sampling rate) 和采样位数 (bit depth)，它们决定了采样后得到的数字信号的质量.

2. PCM 可以分为三步：抽样，量化，编码.

   抽样就是对模拟信号按一定的时间间隔周期性采样，从而将模拟信号变为时间离散的信号。抽样需要遵循奈奎斯特定理 (Nyquist theorem), 即采样频率大于信号最高频率两倍时完整保留了原始信号中的信息.

量化就是将抽样得到的信号在幅度上离散化，只可取有限个值. 量化可分为均匀量化和非均匀量化.

编码就是用一个二进制 (或多进制) 码元表示量化值. 量化编码的过程也就是一个模数转换 (A/D) 过程.

2.2 均匀量化和非均匀量化

1. 均匀量化指在量化过程中采用相等的量化间隔. 如：信号幅值的范围为 $A_{min} \sim A_{max}$，若将其分为 M 段，则：

$$\text{间隔大小: } \Delta A = \frac{A_{max} - A_{min}}{M}$$

$$\text{量化区间端点: } d_i = A_{min} + i\Delta A, i = 0, 1, ..., M$$

可以取区间的上端点、下端点或中间值作为量化值.

2. 非均匀量化指随着信号抽样值的不同，量化间隔也不断变化. 信号抽样值小时，量化间隔也小；抽样值大时，量化间隔也大.

非均匀量化的实现方法是：先将抽样值按函数规律压缩，经过均匀量化、编码、解码后，再按压缩函数的反函数规律扩张.

常用的两种非均匀量化方式为 μ 压缩律和 A 压缩律.

A 律压缩：

$$F(x) = sgn(x) \begin{cases} \frac{A|x|}{1+log(A)}, |x| < \frac{1}{A} \\ \frac{1+log(A|x|)}{1+log(A)}, \frac{1}{A} \leq |x| \leq 1 \end{cases}$$

A 律扩张：

$$F^{-1}(y) = sgn(y) \begin{cases} \frac{|y|(1+ln(A))}{A}, |y| < \frac{1}{1+ln(A)} \\ \frac{exp(|y|(1+ln(A))-1)}{A}, \frac{1}{1+ln(A)} \leq |y| < 1 \end{cases}$$

其中 A 的值常取为 $A = 87.6$.

A 律近似：采用 13 折线压缩特性近似.

μ 律压缩：

$$F(x) = sgn(x)\frac{ln(1+\mu|x|)}{ln(1+\mu)} - 1 \leq x \leq 1$$

μ 律扩张：

$$F^{-1}(y) = sgn(y)(1/\mu)((1+\mu)^{|y|} - 1) - 1 \leq y \leq 1$$

其中 μ 的值常取为 $\mu = 255$.

μ 律近似：采用 15 折线压缩特性近似.

2.3 信噪比

即信号与噪声的功率之比，其公式为：

$$SNR = \frac{P_{signal}}{P_{noise}} = 10log\frac{\sum_{n=0}^{L-1}(x[n])^2}{\sum_{n=0}^{L-1}(\hat{x}[n] - x[n])^2}$$

2.4 数字音频均衡器

均衡器是指用以校正因频率不同而引起的衰减（即传输损耗）及相位差不同的网络. 能校正衰减与频率关系的，称为“衰减均衡器”; 能校正相位差与频率关系的称为“相位均衡器”. 运用数字滤波器组成的均衡器称为数字均衡器.

§3 实验准备及函数编写

3.1 wavread() 函数

在 MATLAB R2016b 中直接调用 wavread() 函数时提示函数未定义，百度后的结果是在较新版本的 MATLAB 中删除了内置的 wavread() 函数，改用 audioread() 函数实现类似的功能.

wavread() 函数的用法是：

```
[sample,fs,bits]=wavread([filename],[N1 N2]);
```

sample 记录采样数据，fs 记录采样频率，bits 记录采样位数.[N1 N2] 选择读取文件 [filename] 的范围.

audioread() 函数的用法是：

```
[Y, FS]=audioread(FILENAME) reads an audio file specified
by the string FILE, returning the sampled data in Y and
the sample rate FS, in Hertz.

[Y, FS]=audioread(FILENAME, [START END]) returns only
samples START through END from each channel in the file.

[Y, FS]=audioread(FILENAME, DATATYPE) specifies the data
type format of Y used to represent samples read from the
file. If DATATYPE='double', Y contains double-precision
normalized samples. If DATATYPE='native', Y contains
samples in the native data type found in the file.
Interpretation of DATATYPE is case-insensitive and partial
matching is supported. If omitted, DATATYPE='double'.

[Y, FS]=audioread(FILENAME, [START END], DATATYPE);

Output Data Ranges
Y is returned as an m-by-n matrix, where m is the number of
audio samples read and n is the number of audio channels in
the file.

If you do not specify DATATYPE, or dataType is 'double',
then Y is of type double, and matrix elements are normalized
```

```
values between -1.0 and 1.0.

If DATATYPE is 'native', then Y may be one of several MATLAB
data types, depending on the file format and the
BitsPerSample of the input file:

 File Format| BitsPerSample| Data Type of Y| Data Range of Y
------------------------------------------------------------
WAVE (.wav)          8          uint8           0 <= Y <= 255
                    16          int16      -32768 <= Y <= 32767
                    24          int32       -2^32 <= Y <= 2^32-1
                    32          int32       -2^32 <= Y <= 2^32-1
                    32          single       -1.0 <= Y <= +1.0
------------------------------------------------------------
FLAC (.flac)         8          uint8           0 <= Y <= 255
                    16          int16      -32768 <= Y <= 32767
                    24          int32       -2^32 <= Y <= 2^32-1
------------------------------------------------------------
MP3 (.mp3)          N/A         single       -1.0 <= Y <= +1.0
MPEG-4(.m4a,.mp4)
OGG (.ogg)
------------------------------------------------------------

Call audioinfo to learn the BitsPerSample of the file.

Note that where Y is single or double and the BitsPerSample
is 32 or 64, values in Y might exceed +1.0 or -1.0.
```

因而可以考虑使用函数封装的方法自建一个 wavread() 函数：

```
function [x, fs, nbits] = wavread(filename)
```

```
% [x, fs, nbits] = wavread(filename)
% 模拟封装老版本的wavread函数，读取数据、采样率、采样位数

% 利用audioread读取数据和采样率
[x, fs] = audioread(filename);

% 利用audioinfo读取采样位数
info = audioinfo(filename);
nbits = info.BitsPerSample;

end
```

3.2 sound() 函数

MATLAB 中可以将向量当做音频数据形成声音并播放，用到的函数是 sound()，其用法如下：

```
sound Play vector as sound.
sound(Y,FS) sends the signal in vector Y (with sample
frequency FS) out to the speaker on platforms that
support sound. Values in Y are assumed to be in the
range -1.0 <= y <= 1.0. Values outside that range are
clipped. Stereo sounds are played, on platforms that
support it, when Y is an N-by-2 matrix.

sound(Y) plays the sound at the default sample rate
of 8192 Hz.

sound(Y,FS,BITS) plays the sound using BITS bits/sample
if possible. Most platforms support BITS=8 or 16.

```

```
15 Example:
16     load handel
17     sound(y,Fs)
18 You should hear a snippet of Handel's Hallelujah Chorus.
```

使用 clear sound 命令可以停止当前的播放.

3.3 SNR_calc() 函数

为方便计算得到量化后信号的信噪比，故编写 SNR_calc() 函数，具体代码见附 1.

```
1 >> help SNR_calc
2  SNR = SNR_calc(noise, singal)
3
4  输入为量化前后的波形，输出信噪比
```

3.4 plotTD() 函数

其主要功能是绘制并保存输入信号的时域波形，具体代码见附 1.

```
1 >> help plotTD
2     plotTD(t, sample, name)
3
4     时域波形显示函数，针对单声道、双声道信号
5     输入为原信号文件（sample，需确保为单声道或双声道信号文件，
6     否则会报错）和对应的时刻信息（t，需与信号对应，否则也会报
7     错），以及生成图像文件的名称
8     若为双声道文件，则分别用sample_left和sample_right记录左、
9     右声道，依次展示双声道、左声道、右声道波形
```

3.5 plotFD() 函数

其主要功能是绘制并保存输入信号的频域波形，具体代码见附 1.

```
>> help plotFD
  plotFD(fs, cnt_point, sample)

  频谱图显示函数
  输入为原信号文件（sample）和对应的采样率（fs）、
  采样点数（cnt_point），以及生成图像文件的名称
  依次展示该信号的频谱矢量图，幅频谱，相位谱
```

3.6 Uniform_Quantization() 函数

其主要功能是实现 8bits 均匀量化，具体代码见附 1.

```
>> help Uniform_Quantization
  [sample_Uniform, sample_8bits_Uniform_balance]
  = Uniform_Quantization(sample)

  均匀量化函数
  输入为信号文件，输出为8bits量化（四舍五入）后以及
  均衡后的信号文件
  主要功能为产生8bits量化信号（四舍五入）及均衡后信号，
  并将信噪比计算结果输出到命令窗口
  辅助实现8bits量化（向上取整）、8bits量化（向下取整）、
  8bits量化（取中间值）、nbits量化功能，进行对比
```

3.7 u_law_Quantization() 函数

其主要功能是实现 μ 律编码量化，具体代码见附 1.

```
>> help u_law_Quantization
  [sample_ulaw_encoding, sample_ulaw_expansion,
  sample_ulaw_balance] = u_law_Quantization(sample)
```

```
4
5       律编码量化函数
6     输入为信号文件，输出为 律压缩、扩张及均衡后的信号文件，
7     并将信噪比计算结果输出到命令窗口
```

3.8 A_law_Quantization() 函数

其主要功能是实现 A 律编码量化，具体代码见附 1.

```
1 >> help A_law_Quantization
2     [sample_Alaw_encoding, sample_Alaw_expansion,
3     sample_Alaw_balance] = A_law_Quantization(sample)
4
5     A律编码量化函数
6     输入为信号文件，输出为A律压缩、扩张及均衡后的信号文件，
7     并将信噪比计算结果输出到命令窗口
```

3.9 Sample_Balance() 函数

其主要功能是实现数字滤波，滤去高于 1/2 采样频率的成分，具体代码见附 1.

```
1 >> help Sample_Balance
2     sample_balance = balance(sample)
3
4     信号均衡函数
5     滤去信号中高于fs/2的频率，由数字信号恢复到模拟信号
6     具体步骤：
7     1.构建窗函数
8     2.频域滤波
```

3.10 compare() 函数

其主要功能是实现两种信号时域波形的对比，通过作差的方式放大差异，具体代码见附1.

```
>> help compare
  compare(sample1, sample2)
  对比函数，比较两种波形的差别
  实现对比方法：时域波形作差
```

3.11 average_track() 函数

其主要功能是对双声道取平均值并做均匀量化，具体代码见’拓展内容’.

```
>> help average_track
  average_track(sample)
  双声道平均函数
```

3.12 reduction() 函数

其主要功能是对音频信号乘上指数衰减因子，实现随时间衰减效果，具体代码见’拓展内容’.

```
>> help reduction
  reduction_sample = reduction(sample, t, fs)
  衰减函数
  在时域乘上指数衰减函数，以实现衰减效果
```

§4 程序主体代码

程序主体代码保存在’Digital_Audio_Processing.m’ 文件中，现附于下：

(因为排版的问题，导致大部分一行的代码分成两行显示，直接打开.m 文件更便于阅读)

```
%%%% 音频处理 %%%%

% 预处理，清空残留数据
clear;

% 读入音频文件
% 采样点数据保存在sample内，fs存储采样率信息，nbits存储
% 采样位数
% wavread函数基于audioread函数
wav = input('请输入音频文件名(如果在当前目录下请输入
[filename],如果不在当前目录下请输入绝对路径):\n','s');
[sample, fs, nbits] = wavread(wav);

% cnt_plot记录采样点数，cnt_track记录声道数，delta_t记录
% 相邻采样点间的时间间隔，t记录每个采样点对应的时刻
[cnt_point, cnt_track] = size(sample);
delta_t = 1 / fs;
t = (0:1:cnt_point - 1) / fs;



%%% 对原始音频文件的处理 %%%

% 显示音频文件的时域波形并保存至TD_origin.fig
% 编写plotTD函数作为信号时域波形输出的辅助函数
figure('name', 'time domain——origin','position',
[200,80,1500,900])
plotTD(t, sample, 'TD_origin');

% 显示音频文件的频谱图并保存至FD_origin.fig
```

```
% 编写plotFD函数作为信号频谱图输出的辅助函数
figure('name', 'frequency domain——origin','position',
[200,80,1500,900])
plotFD(fs, cnt_point, sample, 'FD_origin');



%%% 8bits均匀量化 %%%

% 编写Uniform_Quantization函数作为8bits均匀量化的辅助函数
% 均匀量化时域波形：TD_Uniform
% 均匀量化频谱图：FD_Uniform
% 均匀量化均衡时域波形：TD_Uniform_balance
% 均匀量化均衡频谱图：FD_Uniform_balance
[sample_Uniform,sample_Uniform_balance] =
Uniform_Quantization(sample);
figure('name', 'time domain——uniform','position',
[200,80,1500,900])
plotTD(t, sample_Uniform, 'TD_Uniform');
figure('name', 'frequency domain——uniform','position',
[200,80,1500,900])
plotFD(fs, cnt_point, sample_Uniform, 'FD_Uniform');

figure('name', 'time domain——uniform_balance','position',
[200,80,1500,900])
plotTD(t, sample_Uniform_balance, 'TD_Uniform_balance');
figure('name', 'frequency domain——uniform_balance',
'position',[200,80,1500,900])
plotFD(fs, cnt_point, sample_Uniform_balance,
 'FD_Uniform_balance');
```

```
61
62
63
64 %%%  律编码量化 %%%
65
66 % 编写u_law_Quantization函数作为 律编码量化的辅助函数
67 %  律压缩后时域波形 ：TD_U_Law_encoding
68 %  律压缩后频谱图 ：FD_U_Law_encoding
69 %  律扩张后时域波形 ：TD_U_Law_expansion
70 %  律扩张后频谱图 ：FD_U_Law_expansion
71 %  律均衡后时域波形 ：TD_U_Law_expansion
72 %  律均衡后频谱图 ：FD_U_Law_expansion
73 [sample_ulaw_encoding, sample_ulaw_expansion,
74 sample_ulaw_balance] = u_law_Quantization(sample);
75
76 figure('name', 'time␣domain—— law␣encoding','position',
77 [200,80,1500,900])
78 plotTD(t, sample_ulaw_encoding, 'TD_U_Law_encoding');
79 figure('name', 'frequency␣domain—— law␣encoding',
80 'position',[200,80,1500,900])
81 plotFD(fs, cnt_point, sample_ulaw_encoding,
82  'FD_U_Law_encoding');
83
84 figure('name', 'time␣domain—— law␣expansion','position',
85 [200,80,1500,900])
86 plotTD(t, sample_ulaw_expansion, 'TD_U_Law_expansion');
87 figure('name', 'frequency␣domain—— law␣expansion',
88 'position', [200,80,1500,900])
89 plotFD(fs, cnt_point, sample_ulaw_expansion,
90 'FD_U_Law_expansion');
```

```
91
92  figure('name', 'time domain—— law balance','position',
93  [200,80,1500,900])
94  plotTD(t, sample_ulaw_balance, 'TD_U_Law_balance');
95  figure('name', 'frequency domain—— law balance',
96  'position',[200,80,1500,900])
97  plotFD(fs, cnt_point, sample_ulaw_balance,
98  'FD_U_Law_balance');
99
100
101
102 %%% A律编码量化 %%%
103
104 % 编写A_law_Quantization函数作为A律编码量化的辅助函数
105 % A律压缩后时域波形 : TD_A_Law_encoding
106 % A律压缩后频谱图 : FD_A_Law_encoding
107 % A律扩张后时域波形 : TD_A_Law_expansion
108 % A律扩张后频谱图 : FD_A_Law_expansion
109 % A律扩张后时域波形 : TD_A_Law_expansion
110 % A律扩张后频谱图 : FD_A_Law_expansion
111 [sample_Alaw_encoding, sample_Alaw_expansion,
112  sample_Alaw_balance] = A_law_Quantization(sample);
113
114 figure('name', 'time domain——Alaw encoding','position',
115 [200,80,1500,900])
116 plotTD(t, sample_Alaw_encoding, 'TD_A_Law_encoding');
117 figure('name', 'frequency domain——Alaw encoding',
118 'position', [200,80,1500,900])
119 plotFD(fs, cnt_point, sample_Alaw_encoding,
120 'FD_A_Law_encoding');
```

```
121
122 figure('name', 'time domain——Alaw expansion','position',
123 [200,80,1500,900])
124 plotTD(t, sample_Alaw_expansion, 'TD_A_Law_expansion');
125 figure('name', 'frequency domain——Alaw expansion',
126 'position',[200,80,1500,900])
127 plotFD(fs, cnt_point, sample_Alaw_expansion,
128 'FD_A_Law_expansion');
129
130 figure('name', 'time domain——Alaw balance','position',
131 [200,80,1500,900])
132 plotTD(t, sample_Alaw_balance, 'TD_A_Law_balance');
133 figure('name', 'frequency domain——Alaw balance',
134 'position',[200,80,1500,900])
135 plotFD(fs, cnt_point, sample_Alaw_balance,
136 'FD_A_Law_balance');
137
138
139
140 %%% 对比 %%%
141
142 figure('name', 'time domain——comparison'
143 ,'position',[200,80,1500,900])
144 subplot(5,1,1),compare(t, sample, sample_Uniform);
145 title('origin-8bits uniform');
146 % 对比原波形和8bits均匀量化波形
147 subplot(5,1,2),compare(t, sample, sample_ulaw_expansion);
148 title('origin-ulaw');
149 % 对比原波形和 律量化波形
150 subplot(5,1,3),compare(t, sample, sample_Alaw_expansion);
```

```
151  title('origin-Alaw');
152   % 对比原波形和A律量化波形
153  subplot(5,1,4),compare(t, sample_ulaw_expansion,
154  sample_Alaw_expansion); title('ulaw-Alaw');
155 % 对比 律量化波形和A律量化波形
156  subplot(5,1,5),compare(t, sample_ulaw_expansion,
157  sample_ulaw_balance); title('ulaw-ulaw balance');
158 % 对比 律量化波形和 律量化均衡波形
159  saveas(gcf, 'comparison', 'fig');
160
161
162
163 %%% 拓展内容 %%%
164  fprintf('\n### 拓展内容 ###\n');
165  average_track(sample); % 双声道平均
166  reduction(sample, t, fs);  % 指数衰减
167
168
169
170 %%% 写入wav文件 %%%
171
172  audiowrite('sample_8bits_Uniform.wav',sample_Uniform,fs);
173  audiowrite('sample_ulaw.wav',sample_ulaw_expansion,fs);
174  audiowrite('sample_Alaw.wav',sample_Alaw_expansion,fs);
175
176 %%%%% END %%%%%
```

程序主体代码的主要思路是：首先读入音频文件信息，生成原始音频信号的时域波形和频谱图；接着依次使用 Uniform_Quantization()、u_law_Quantization()、A_law_Quantization() 函数实现均匀量化、μ 律量化、A 律量化；将生成的量化信号与原始信号进行对比，并相互比较，分析量化效果；最后将量化音频文件写入.wav 文件.

§5 音频处理结果及分析

5.1 原音频时域波形及频谱图

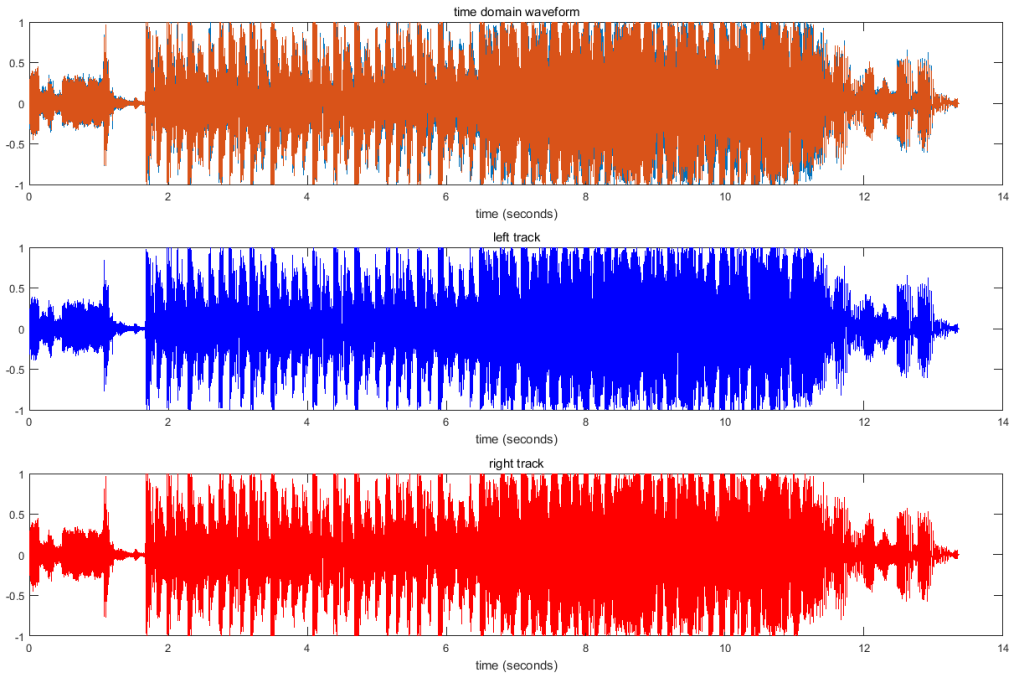

Figure 1: 原音频时域波形

由于样例音频文件为双声道音频文件，故在时域波形图中展现三条轨迹，第一条轨迹将左右声道呈现在一张图中，第二、三条轨迹分别呈现左、右声道的时域波形.

可以看到左右声道的时域波形在总体变化上呈现一致性，因而分别试听左、右声道的音频效果时没有明显差别；但在每一点的具体数值上总是存在一定的差异，应当是音频在录制时左、右声道的位置差异造成的细微区别.

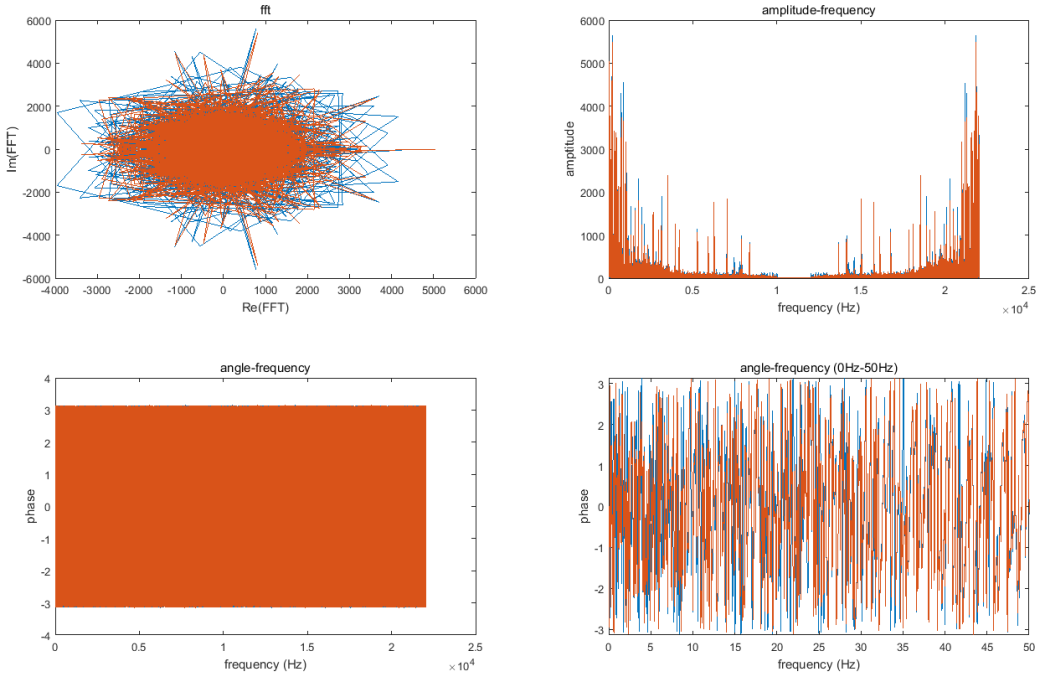

Figure 2: 原音频频谱图

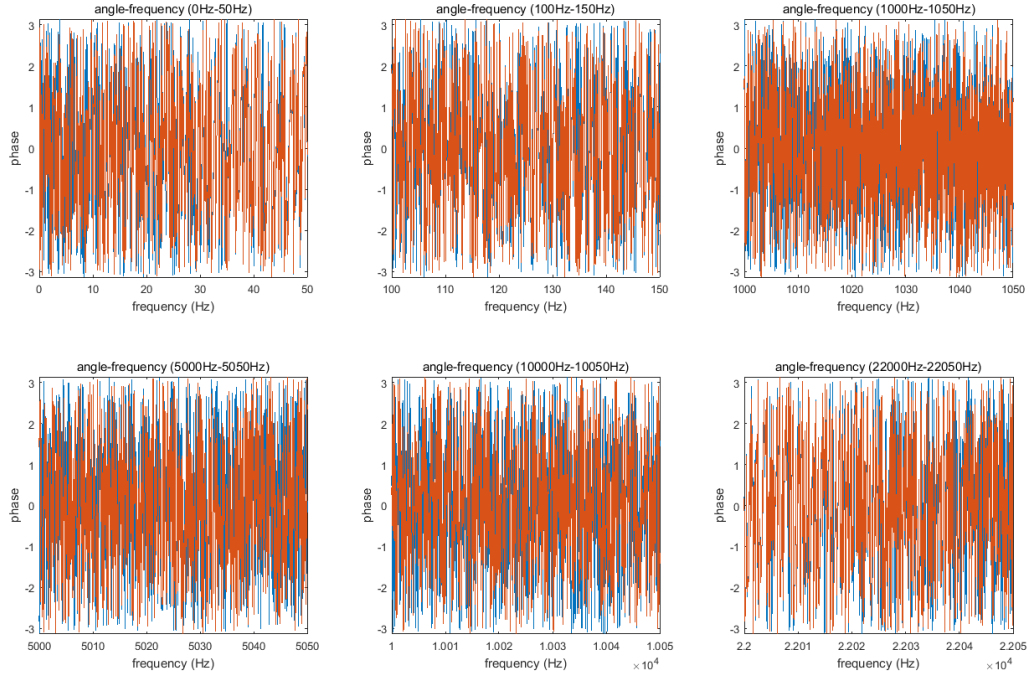

Figure 3: 原音频相位谱 (放大)

在频谱图上，依次展现了频域矢量图、幅度谱和相位谱，由于相位谱变化剧烈且取值密集，得到的图像近似为一矩形，因而第四张图放大呈现了 $0Hz-50Hz$ 内的相位变化；同时又做另一张图，依次分别呈现了 $0Hz-50Hz, 100Hz-150Hz, 1000Hz-1050Hz, 5000Hz-5050Hz, 10000Hz-10050Hz, 22000Hz-22050Hz$ 处的相位谱.

直观上可以看到，幅度谱关于 $f_s/2 = 11025Hz$ 对称分布，分析认为 f_s 是临界采样值，能够完整获取 $f_s/2$ 频率以下的波形信息；采样导致信号在频域上呈现周期性变化，因而会出现关于 $f_s/2 = 11025Hz$ 对称分布的现象.

从相位谱上观察到不同频率范围内相位变化的情况不同，但由于分析能力的限制只能从定性的角度得到直观的结果，不能完成定量分析，未发现有效的规律.

5.2 均匀量化时域波形及频谱图

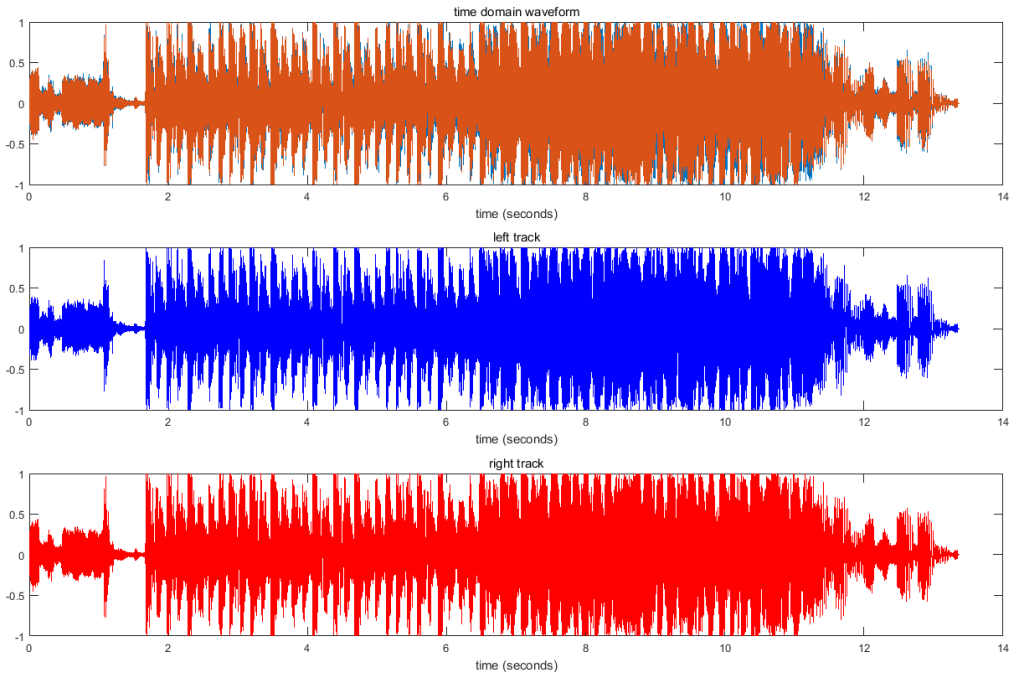

Figure 4: 均匀量化时域波形

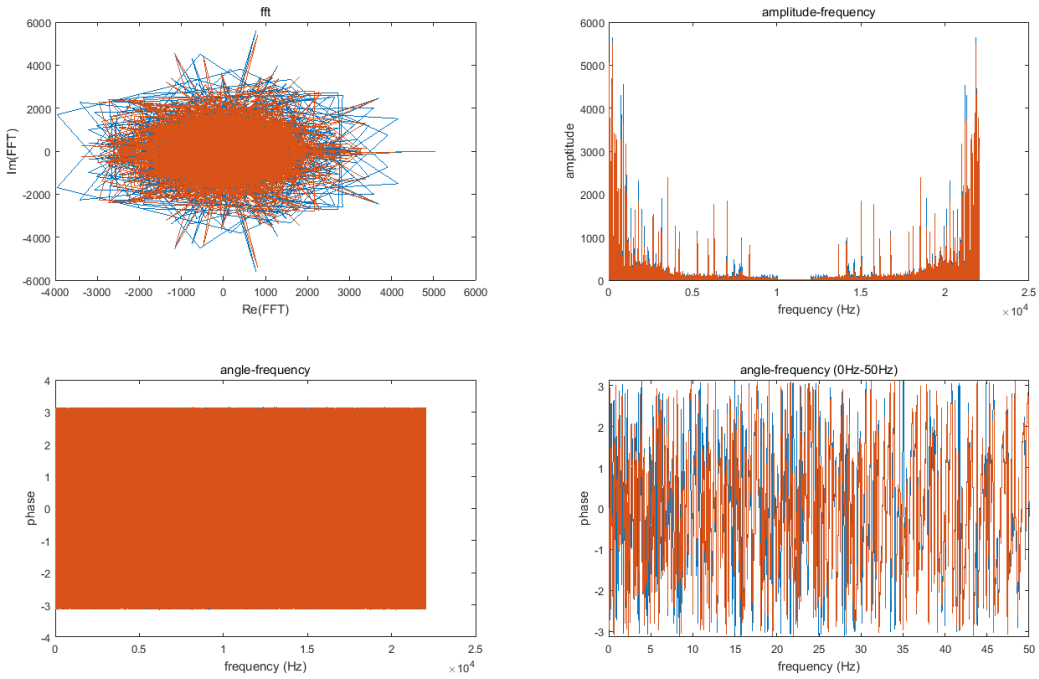

Figure 5: 原音频频谱图

与原音频时域波形和频谱图类似，得到 8bits 均匀量化之后的时域波形和频谱图.8bits 均匀量化过程采取将幅度值分为 256 等分，四舍五入取最接近的值.

此时得到的信噪比为：43.061188. 为对比采用不同近似方式的效果，故将不同近似方式的信噪比进行比较，如下表所示：

8 位均匀量化信噪比 (round, 四舍五入)	43.061188
8 位均匀量化信噪比 (ceil, 向上取整)	37.047293
8 位均匀量化信噪比 (floor, 向下取整)	37.061737
8 位均匀量化信噪比 (mid, 取中间值)	43.087116
4 位均匀量化信噪比 (round, 四舍五入)	19.135333

可以发现针对本实验的样例音频，采用四舍五入和取中间值的方式得到的信噪比要比向上或向下取整得到的信噪比更高. 从定性角度而言，取中间值和四舍五入的方式使得每一点的误差都在 0.5 个量化间隔范围内，而向上或向下取整则有可能达到 1 个量化间隔，从而使得信噪比下降，实验结果与定性分析一致. 而四舍五入和取中间值的差异很小，可以认为这是特定的音频文件特性导致的，对不同音频文件它们的大小关系可能不同，但始终差异极小.

而采用 8bits 量化的效果总是要比 4bits 量化的效果有很大的提升，因而认为每提高一个量化位数对于信噪比有较大的提升.

继续对均匀量化后的信号做均衡，即用频域上的窗函数低通滤波，使得高于 $f_s/2$ 的频率成分滤去，得到模拟信号.

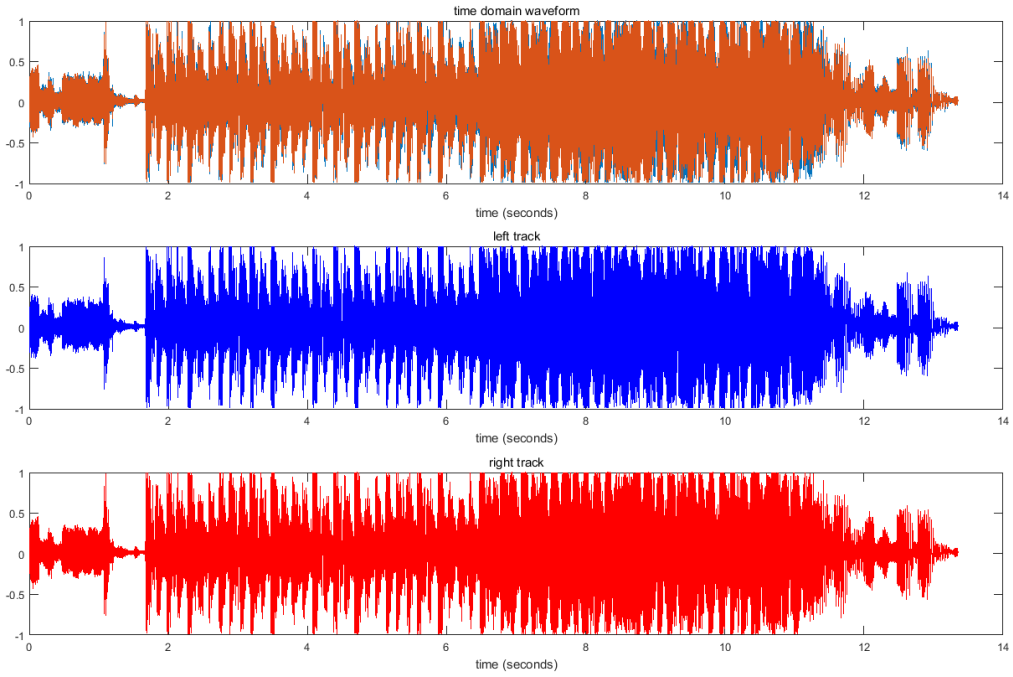

Figure 6: 均匀量化均衡时域波形

计算得 8 位均匀量化均衡后信噪比: 25.727822.

发现在做完均衡之后信号的信噪比反而下降很多，且频谱图依旧呈现出关于 $f_s/2$ 对称的现象.

对这个过程逐步分析，发现在加上窗函数滤波后，得到的信号频谱图确实只有低于 $fs/2$ 的频率成分，但在做完 ifft(快速傅里叶逆变换) 之后，得到的信号依旧是一个关于 $fs/2$ 对称的信号，也就是依旧为一个离散的信号. 这应当是快速傅里叶变换的算法针对离散时间的信号，若对连续信号采用 fft，相当于做完采样后的结果，从而得到的结果表现为离散信号的性质.

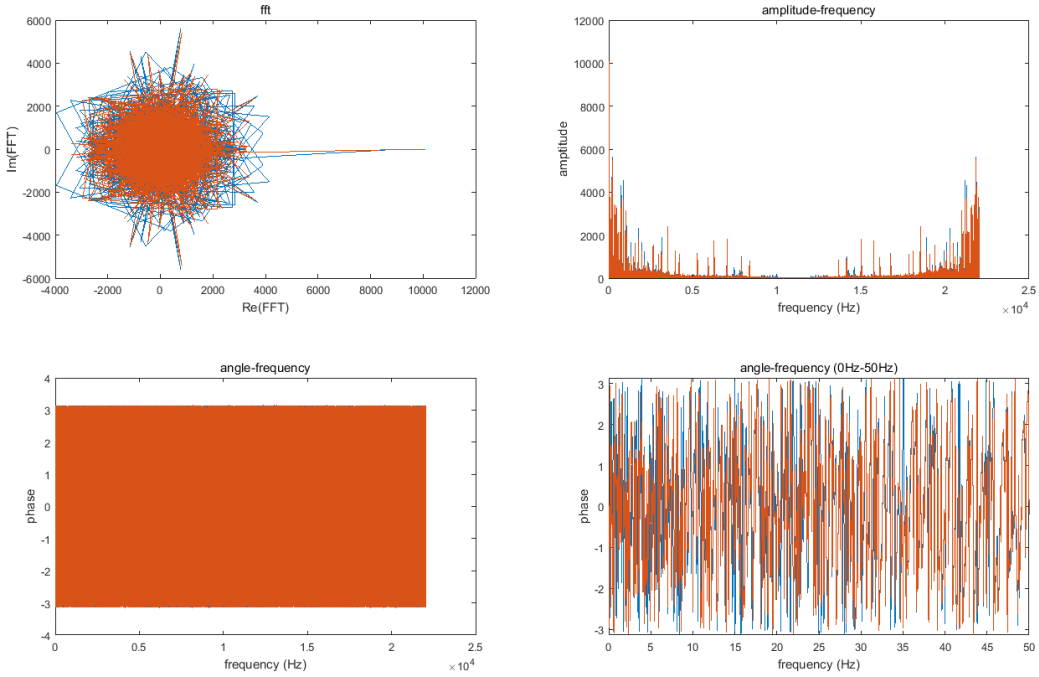

Figure 7: 均匀量化均衡频谱图

5.3 μ 律时域波形及频谱图

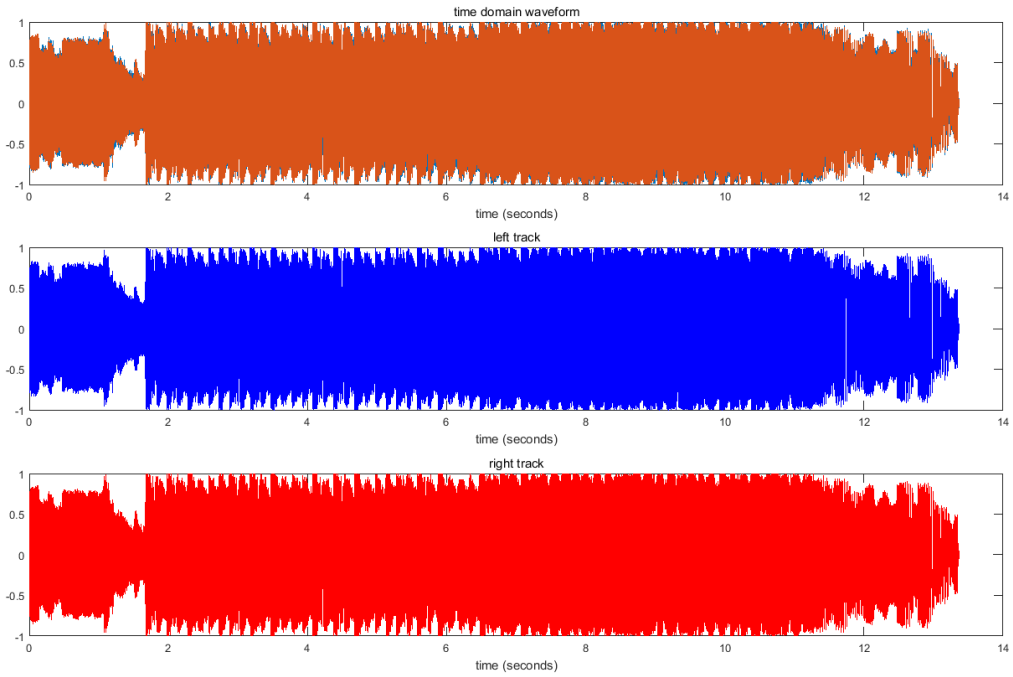

Figure 8: μ 律压缩时域波形

首先得到经过 μ 律压缩之后的信号的时域波形和频谱图. 可以看到整个信号的峰值变化幅度减小，也就是 μ 律压缩实现了将大信号压缩的功能.

继续实现 μ 律的扩张，测得此时的信噪比为：44.059978.

对于本实验的音频信号，采用 μ 律量化之后信噪比略高于均匀量化的结果. 从时域波形上定性分析，样例音频在 7s-11s 的范围内一直处于幅值较大的状态，可能这样大信号成分较多的波形会导致 μ 律量化对于大信号成分有较高精度的优势得以表现，从而有较高的信噪比.

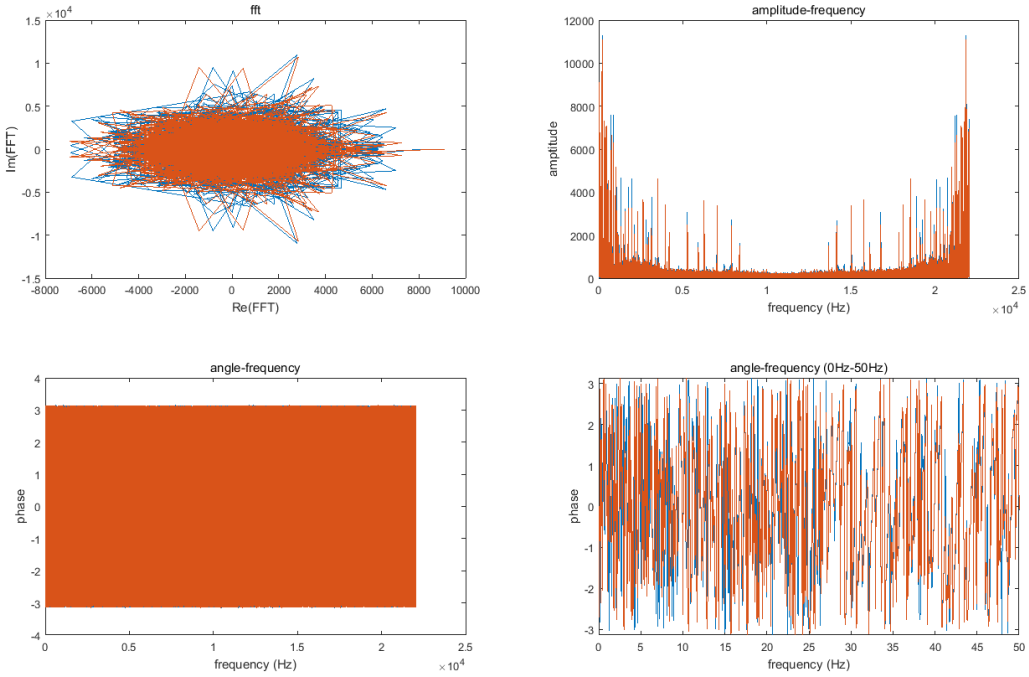

Figure 9: μ 律压缩频谱图

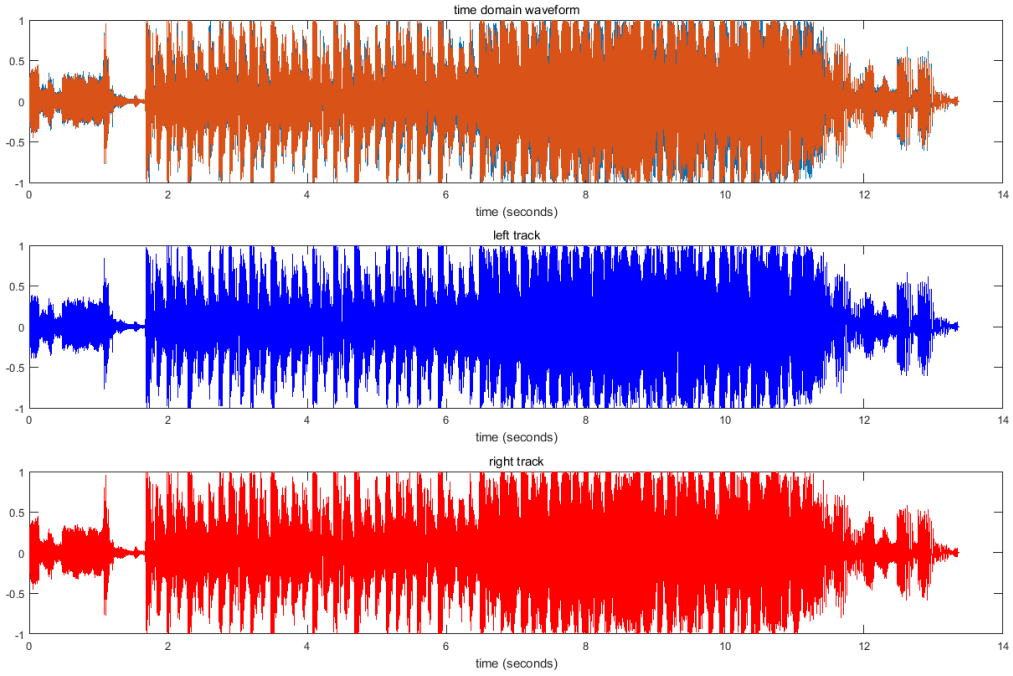

Figure 10: μ 律扩张时域波形

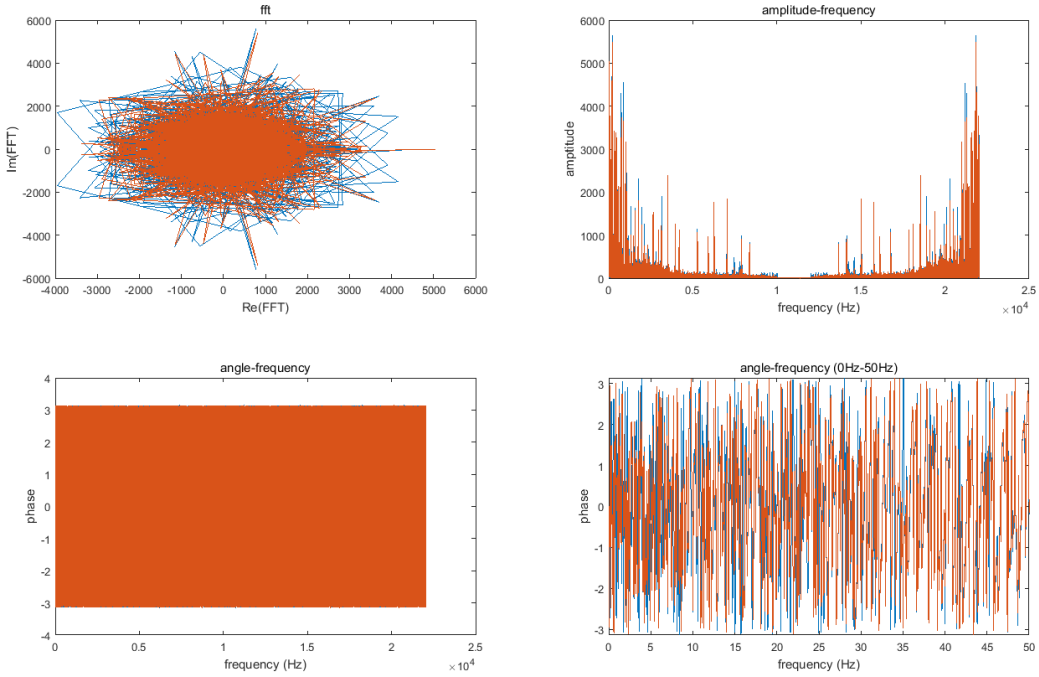

Figure 11: μ 律扩张频谱图

采用 μ 律均衡后得到的信噪比：25.743039.

和均匀量化时的结果一致，均衡都使得信噪比有较为明显的下降. 可能的原因是读取的.wav 文件本身就是采样模拟信号后得到的数字信号，因而用滤波得到的模拟信号与这样一个数字信号比较得到的信噪比会不如不经滤波的数字信号的信噪比更高.

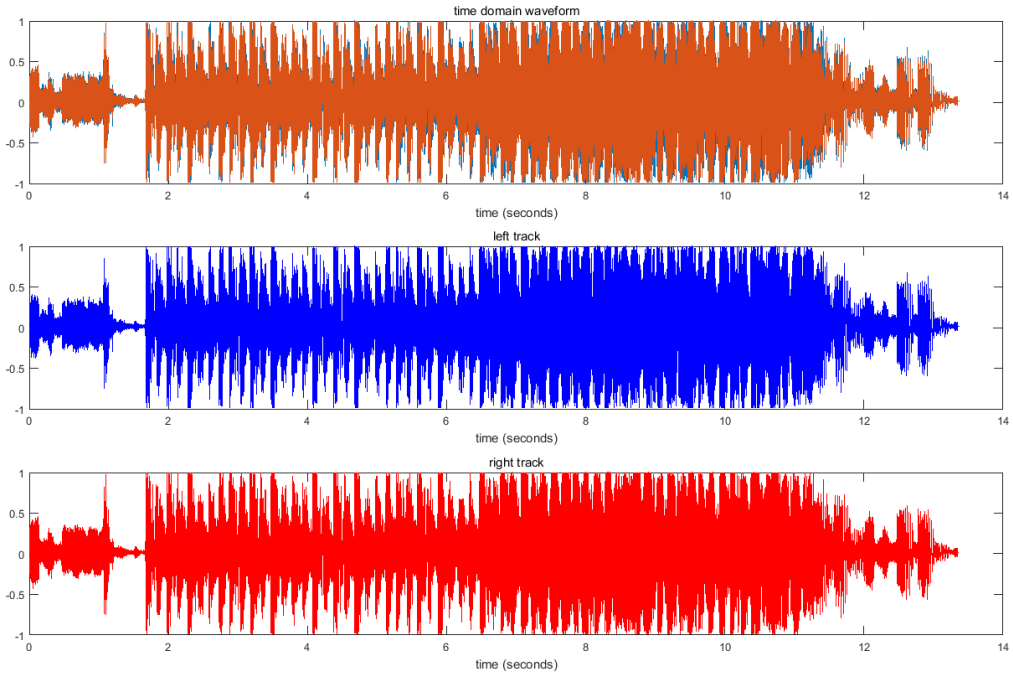

Figure 12: μ 律均衡时域波形

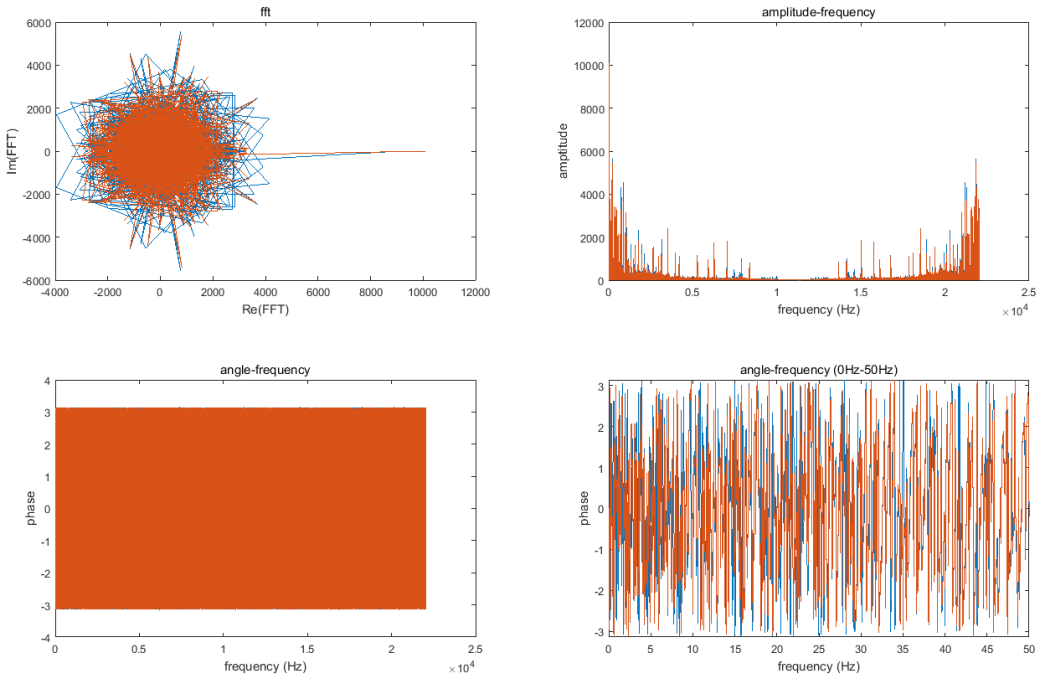

Figure 13: μ 律均衡频谱图

5.4 A 律时域波形及频谱图

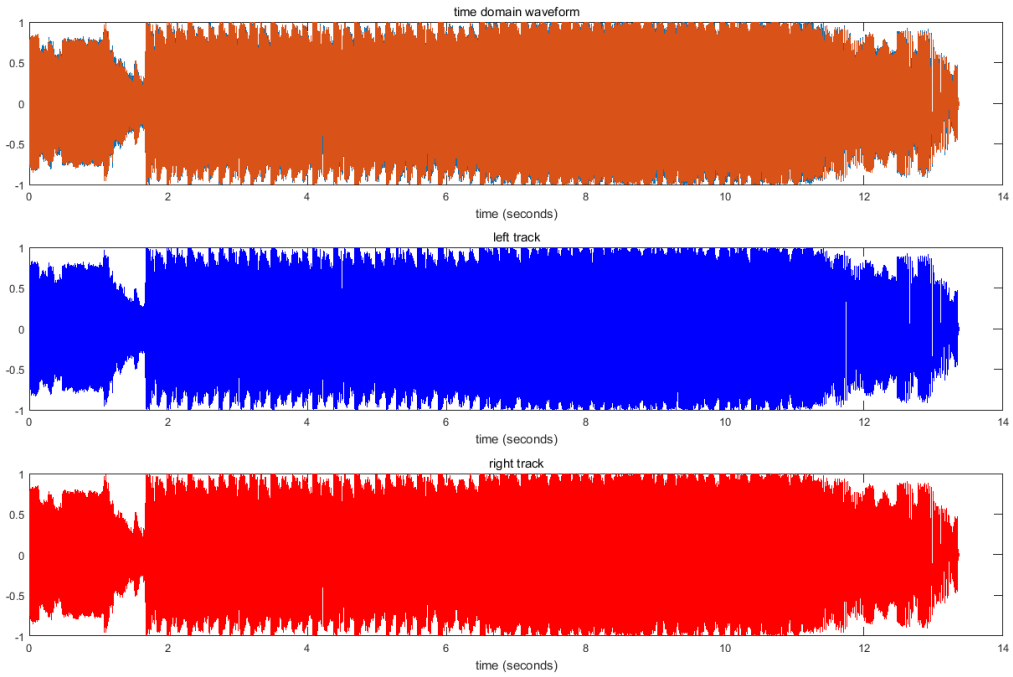

Figure 14: A 律压缩时域波形

和 μ 律情况类似，首先得到经过 A 律压缩之后的信号的时域波形和频谱图. 可以看到整个信号的峰值变化幅度减小，也就是 A 律压缩同样实现了将大信号压缩的功能.

继续实现 A 律的扩张，测得此时的信噪比为：44.251319.

对于本实验的音频信号，采用 A 律量化之后信噪比略高于均匀量化的结果，同时略高于 μ 律结果，但实际上因为相差较小 (与 μ 律结果仅相差 02dB), 并不能说明 A 律优于 μ 律.A 律信噪比高于均匀量化信噪比的解释与 μ 律类似.

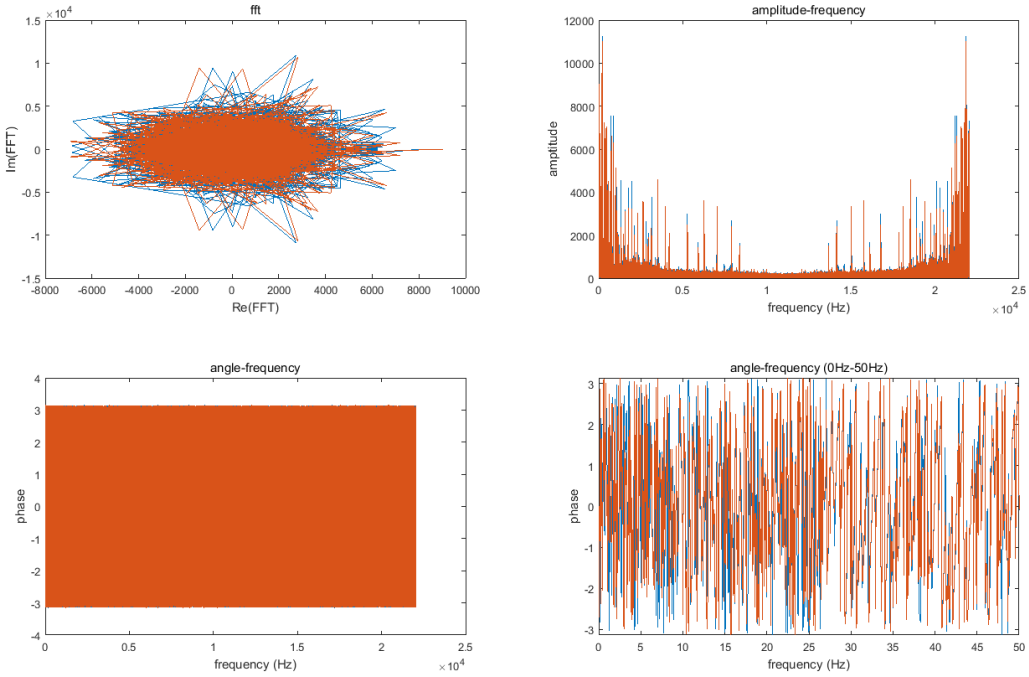

Figure 15: A 律压缩频谱图

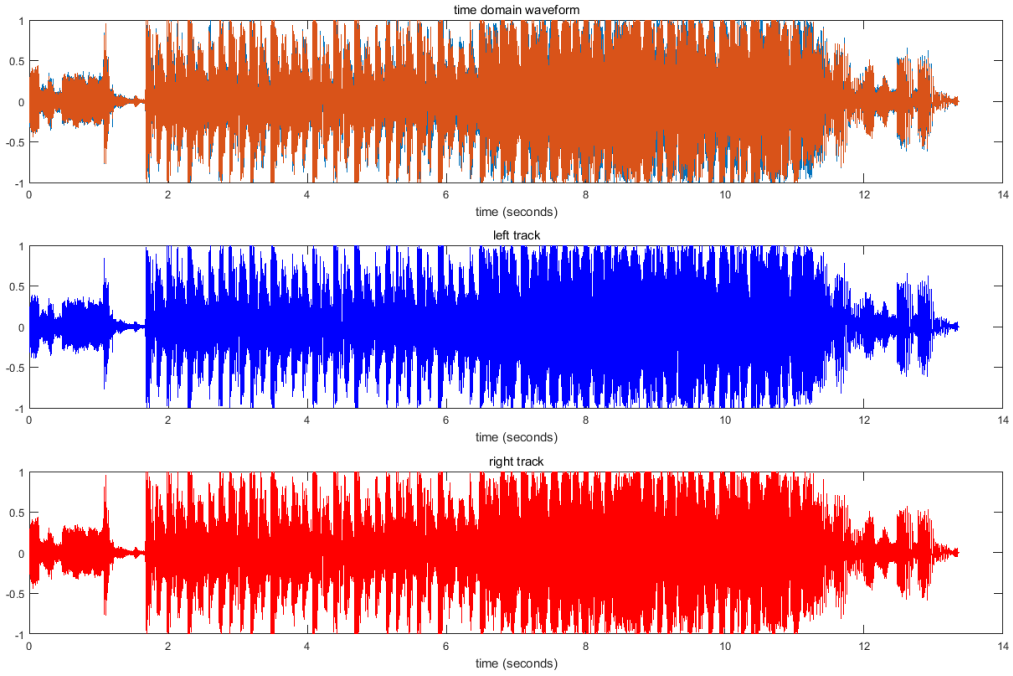

Figure 16: A 律扩张时域波形

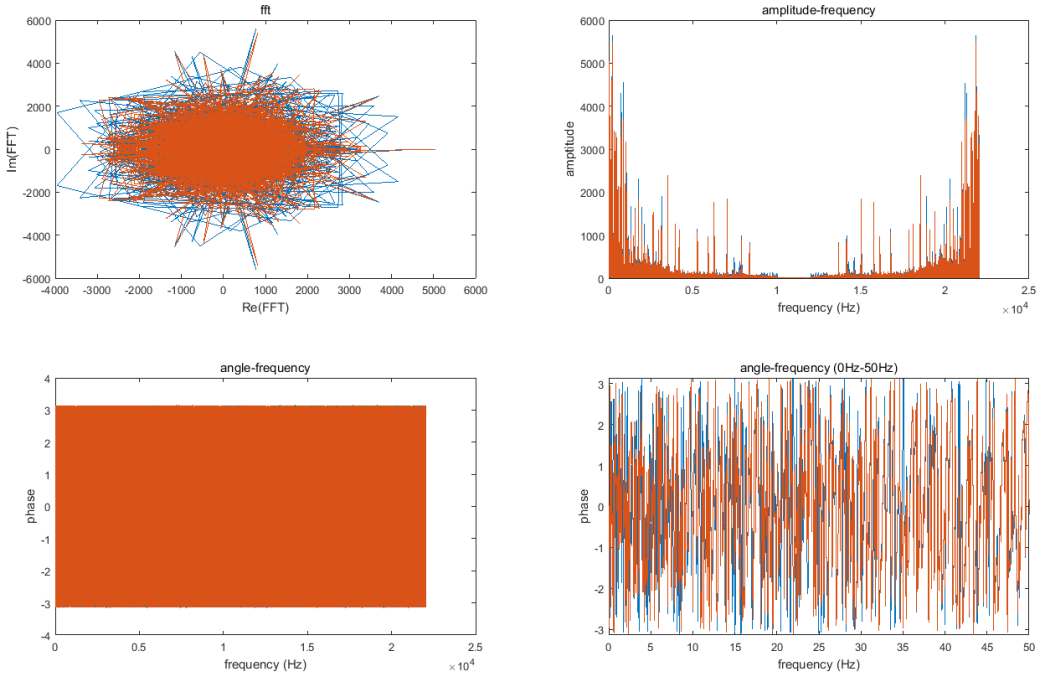

Figure 17: A 律扩张频谱图

A 律量化均衡后信噪比: 25.746108.

在数值上与 μ 律结果基本相同，且都远低于不经过均衡的结果，解释应与 μ 律解释类似.

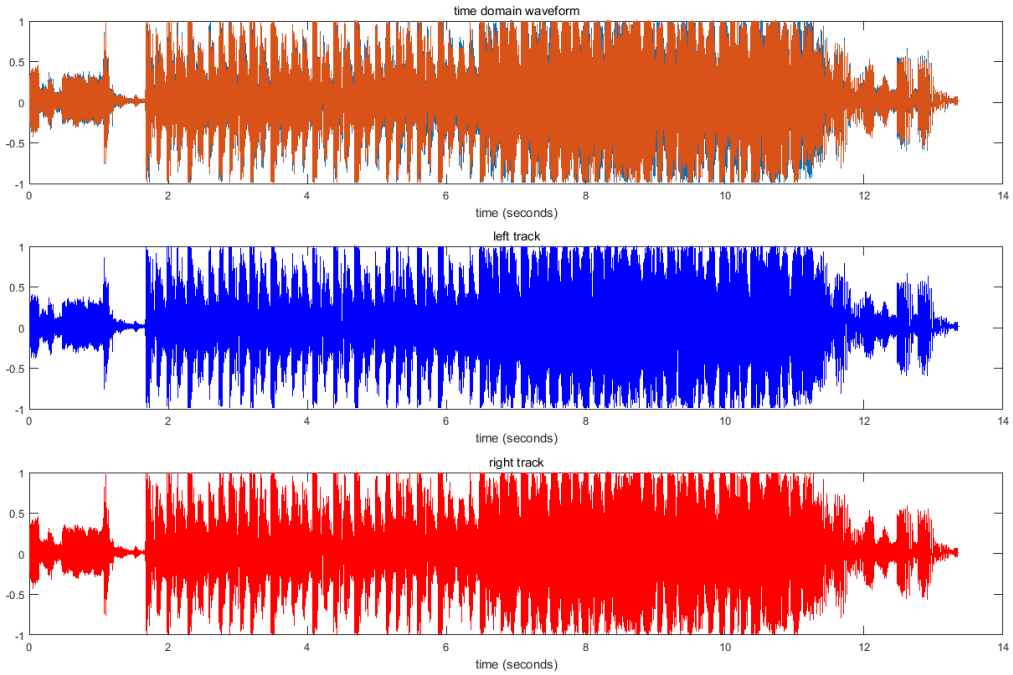

Figure 18: A 律均衡时域波形

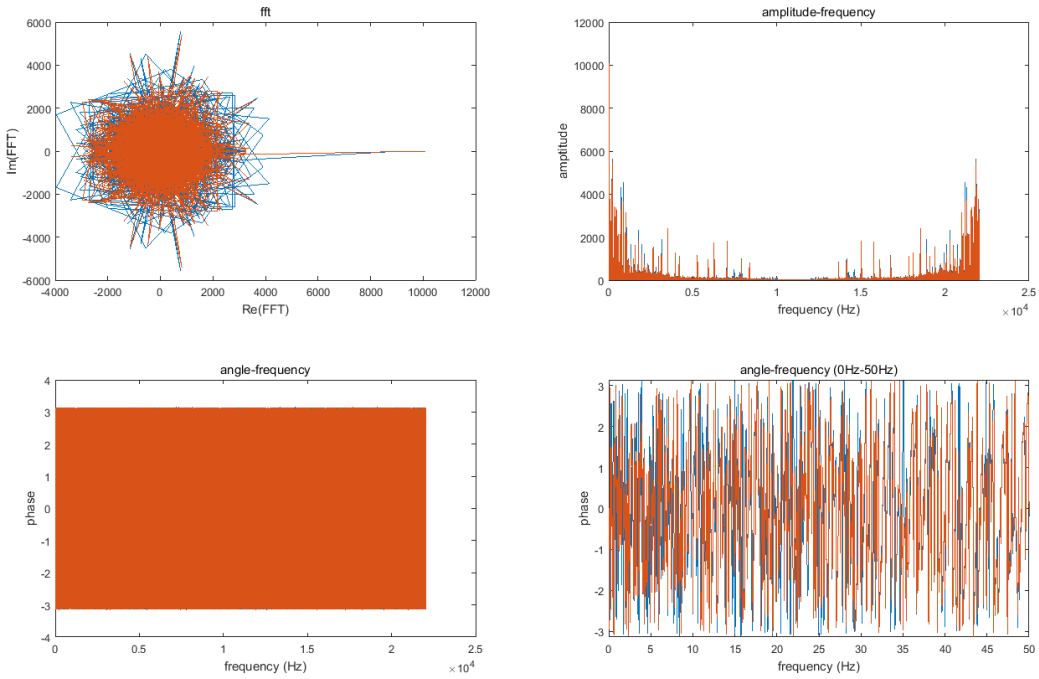

Figure 19: A 律均衡频谱图

5.5 对比

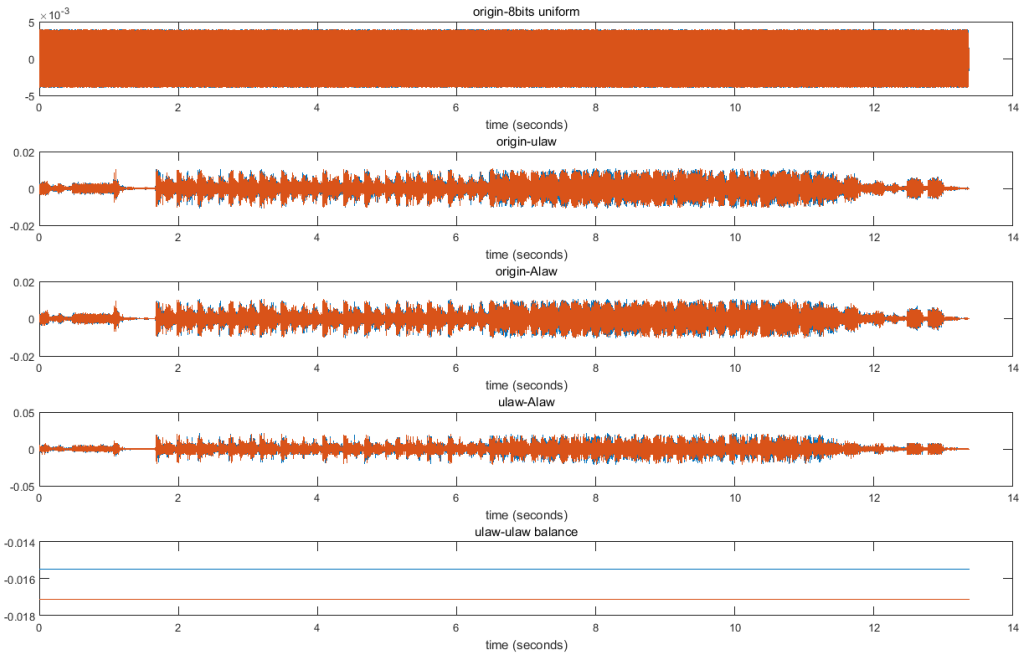

Figure 20: 时域波形对比

五条时域波形轨迹分别是：均匀量化波形与原始波形的差值、μ 律量化波形与原始波形的差值、A 律量化波形与原始波形的差值、μ 律量化波形与 A 律量化波形的差值、μ 律均衡后与原始波形的差值.

从第一条轨迹上看到，均匀量化波形与原始波形的差值均匀稳定地分布在 ± 0.0039 的范围内 (信号的最大值最小值分别为 1 和-1,0.0039 即对应峰值的 0.39%). 考虑到 8bits 量化的间隔为 $\frac{2}{2^8} = \frac{1}{128} = 0.0078125 \approx 2 * 0.0039$, 得到结果的误差与 8bits 量化的理论分析一致.

第二条曲线和第三条曲线在波形形状及变化趋势上都与原信号波形类似，分布在 ± 0.01(1%, 相对于峰值) 的范围内. 从绝对值上看，经过 μ 律和 A 律量化得到的波形与原始波形的差值似乎要大于均匀量化的结果 (0.39%, 相对于峰值)，即在绝对误差上不如均匀量化就结果. 下面做出差值相对于原始波形幅值的百分比图像 (即相对误差图像)：

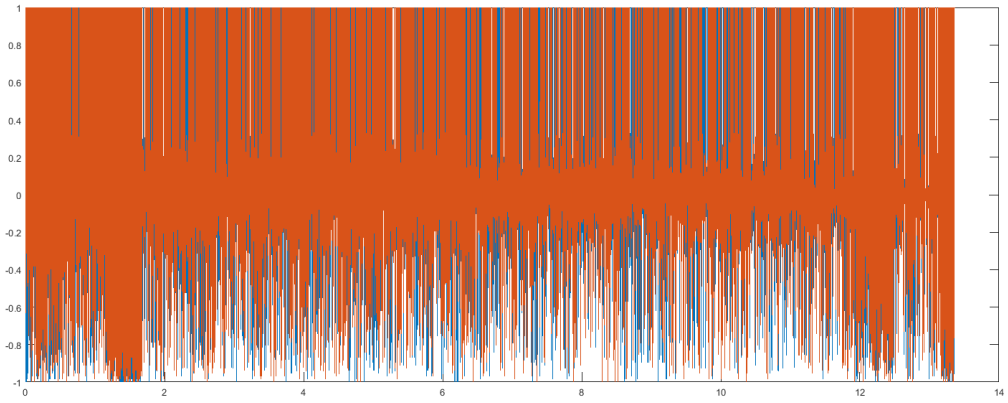

Figure 21: 均匀量化误差百分比

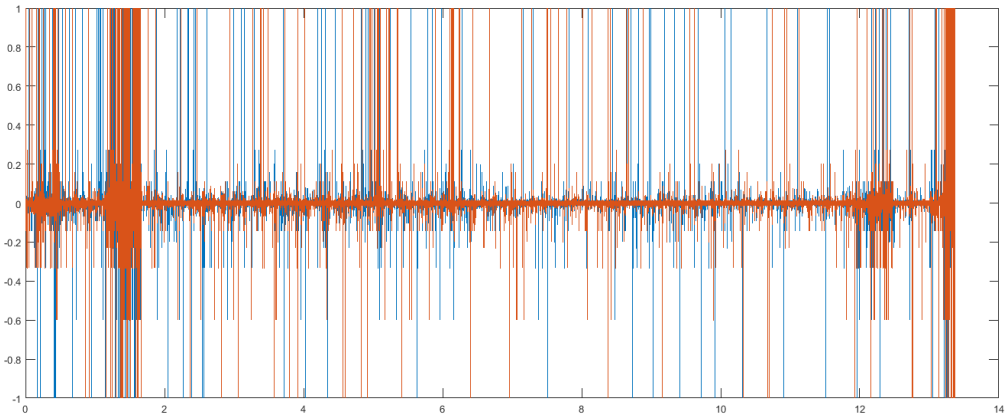

Figure 22: A 律量化误差百分比

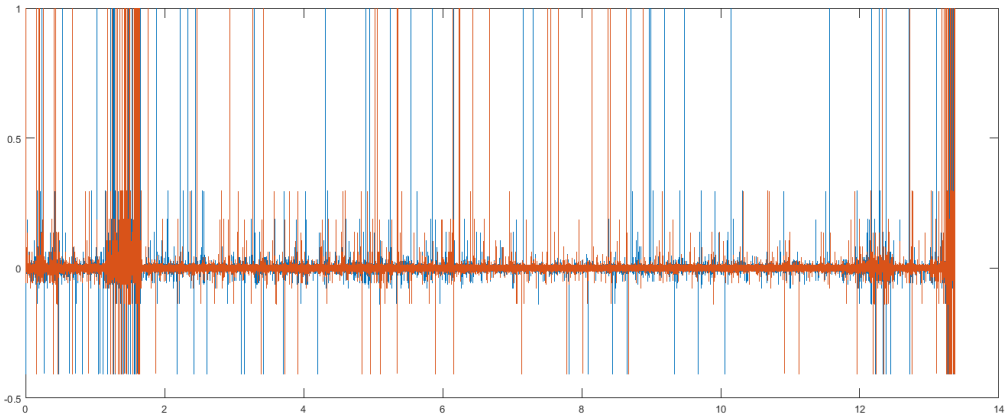

Figure 23: μ 律量化误差百分比

从图像上很清晰地看到，均匀量化得到的相对误差密集地分布在 ±20% 的范围内，大部分位于 50% 范围内，且在小信号处很容易超过 100%.

而 μ 律和 A 律的相对误差图像则“干净”很多. 可以看到它们的相对误差密集地分布在小于 5% 的范围内，大部分位于 20% 范围内，仅在信号幅值很小的时候才会超过 100%.

这样的对比很清晰地看到，虽然在绝对误差上不如均匀量化结果，信噪比也和均匀量化差异不大，但在相对误差上经过非均匀量化的信号远远优于均匀量化的结果，而相对于绝对误差，相对误差对信号传输效果的影响要重要得多. 因而可以很明显地体会到 μ 律量化和 A 律量化的优势.

同时，直观定性地观察相对误差的图，A 律量化尽管在信噪比和绝对误差上与 μ 律几乎相同，但在相对误差上，A 律更加稳定，且出现很大相对误差的概率更小，能比较明显地感受到 A 律要优于 μ 律结果.

对于第五条轨迹，也就是 μ 律均衡后与原始波形的差值，可以看到二者的差值几乎为一常数值. 也就是说，在经过本实验不成功的均衡后，得到的波形相当于在均衡前的波形上加上一定数值的偏置，因而尽管得到的音频效果差异似乎不大，但信噪比远远下降，这也解释了上面均衡后信噪比远低于均衡前的现象.

§6 拓展内容

6.1 双声道平均

在实验过程中发现，本实验采用的音频信号双声道的内容基本相同，产生误差的原因很可能是采样时采样声道的位置不同引起的. 因而考虑对双声道取平均值，分析是否能够改善信噪比.

实现此过程的代码如下：

```
function average_sample = average_track(sample)
% average_track(sample)
% 双声道平均函数

sample_left = sample(:,1);
sample_right = sample(:,2);

```

```
8  average_sample = (sample_left + sample_right) / 2;
9  average_sample = repmat(average_sample, 1, 2);
10 average_sample = round(average_sample * 128) / 128;
11 SNR_average = SNR_calc(average_sample, sample);
12 fprintf('\n双声道平均均匀量化信噪比: %f \n',SNR_average);
13
14 end
```

得到的信噪比：11.028195，远不如未做平均值时的结果. 故此方案应舍去.

6.2 减弱效果

由于已经获得音频信号的数值信息，因而考虑在时域乘上衰减的指数信号，得到随时间衰减的音频.

实现此过程的代码如下：

```
1  function reduction_sample = reduction(sample, t, fs)
2  % reduction_sample = reduction(sample, t, fs)
3  % 衰减函数
4  % 在时域乘上指数衰减函数，以实现衰减效果
5
6  production = exp(-0.5 * t);
7  production = production';
8  production = repmat(production, 1, 2);
9  reduction_sample = sample .* production;
10 audiowrite('reduction_sample.wav',reduction_sample,fs);
11 end
```

得到音频文件'reduction_sample.wav'，确实实现了衰减功能.

§7 附 0：程序使用样例

```
>> Digital_Audio_Processing
请输入音频文件名(如果在当前目录下请输入[filename],如果不在当前目录下请输入绝对路径):
music.wav

8位均匀量化信噪比: 43.061188
8位均匀量化均衡信噪比: 25.727822

请输入自定义量化位数n: 4

n位均匀量化信噪比: 19.135333

请输入μ值(正整数，本次作业中应输入255): 255

μ律量化信噪比: 44.059978
μ律量化均衡信噪比: 25.743039

A律量化信噪比: 44.251319
A律量化均衡信噪比: 25.746108

### 拓展内容 ###

双声道平均均匀量化信噪比: 11.028195
```

Figure 24: 运行结果样例

上图为运行结果样例，具体操作如下：

输入 Digital_Audio_Processing 开始运行程序；

输入需要做音频处理的文件名 (本实验中是 music.wav)；

返回均匀量化的信噪比结果；

输入自定义的均匀量化位数 n(实现和 8 位的比较，这里输入的是 4)；

返回 n 为均匀量化的信噪比；

输入 μ 律量化的参数 μ 值 (本实验取 255)；

返回 μ 律量化信噪比结果；

返回 A 律量化信噪比结果 (A 律参数固定为 87.6)；

输出拓展内容的结果;

将音频文件写入.wav 文件中.

Tips: 运行此代码会同时产生共计 19 个.fig 文件，总计大小约 1GB，请慎重运行：-D

§8 附 1：函数代码

1. wavread() 函数

```
function [x, fs, nbits] = wavread(filename)
% [x, fs, nbits] = wavread(filename)
% 模拟封装老版本的wavread函数，读取数据、采样率、
% 采样位数

% 利用audioread读取数据和采样率
[x, fs] = audioread(filename);

% 利用audioinfo读取采样位数
info = audioinfo(filename);
nbits = info.BitsPerSample;

end
```

2. SNR_calc() 函数

```
function SNR = SNR_calc(noise, signal)
%SNR = SNR_calc(noise, singal)
%输入为量化前后的波形，输出信噪比

%Ps表示信号功率
Ps = sum(signal.^2);

%Pn表示量化噪声功率
```

```
Pn = sum((noise - signal).^2);

%求得信噪比SNR = 10 * log10(Ps / Pn)
SNR = 10 * log10(Ps / Pn);

%测试时使用：fprintf('\n%d %d\n', Ps, Pn);
end
```

3. plotTD() 函数

```
function plotTD(t, sample, name)
% plotTD(t, sample, name)
% 时域波形显示函数，针对单声道、双声道信号
% 输入为原信号文件（sample，需确保为单声道或双声道信号
% 文件，否则会报错）和对应的时刻信息（t，需与信号对应，
% 否则也会报错），以及生成图像文件的名称
% 若为双声道文件，则分别用sample_left和sample_right记录
% 左、右声道，依次展示双声道、左声道、右声道波形
[point, track] = size(sample);
t_size = size(t);

% 信号与时刻信息不对应时会报错：ERROR. 输入信号和时间
% 信息不对应！
if (point ~= t_size)
    errordlg('输入信号和时间信息不对应！','error');
elseif (track == 2)
    sample_left = sample(:,1);
    sample_right = sample(:,2);
    subplot(3,1,1);plot(t, sample);
        title('time domain waveform');
        xlabel('time (seconds)');
```

```
22          ylim([-1 1]);
23      subplot(3,1,2);plot(t, sample_left,'b');
24          title('left track');
25          xlabel('time (seconds)');
26          ylim([-1 1]);
27      subplot(3,1,3);plot(t, sample_right,'r');
28          title('right track');
29          xlabel('time (seconds)');
30          ylim([-1 1]);
31  elseif (track == 1)
32      plot(t, sample);
33      title('time domain waveform');
34      xlabel('time (seconds)');
35      ylim([-1 1]);
36  else
37  % 输入文件非双声道或单声道时会报错 :ERROR. 输入信号和时间
38  % 信息不对应 !
39      errordlg('输入文件超出本程序处理范围 !','error');
40
41  end;
42
43  % 保存图像至 'TD.fig' 文件
44  saveas(gcf, name, 'fig');
45
46  end
```

4. plotFD() 函数

```
1  function plotFD(fs, cnt_point, sample, name)
2  % plotFD(fs, cnt_point, sample)
3  % 频谱图显示函数
```

```
4  % 输入为原信号文件（sample）和对应的采样率（fs）、
5  % 采样点数（cnt_point），以及生成图像文件的名称
6  % 依次展示频谱矢量图，幅频谱，相位谱
7
8  % 图1：矢量图
9  fftsample = fft(sample);
10 subplot(2,2,1); plot(fftsample);
11 title('fft');
12 xlabel('Re(FFT)');
13 ylabel('Im(FFT)');
14
15 % 图2：幅度谱
16 f = (0:1:cnt_point - 1) .* (fs / cnt_point);
17 subplot(2,2,2);plot(f, abs(fftsample));
18 title('amplitude-frequency');
19 xlabel('frequency (Hz)');
20 ylabel('amptitude');
21
22 % 图3：相位谱
23 subplot(2,2,3);plot(f, angle(fftsample));
24 title('angle-frequency');
25 xlabel('frequency (Hz)');
26 ylabel('phase');
27
28 % 图4：相位谱（放大）
29 subplot(2,2,4);plot(f, angle(fftsample));
30 axis([0, 50, -pi, pi]);
31 title('angle-frequency (0Hz-50Hz)');
32 xlabel('frequency (Hz)');
33 ylabel('phase');
```

```
34
35 % 保存图像至 'FD.fig' 文件
36 saveas(gcf, name, 'fig');
37
38 end
```

5. Uniform_Quantization() 函数

```
1  function [sample_8bits_Uniform,
2  sample_8bits_Uniform_balance] =
3  Uniform_Quantization(sample)
4  % [sample_Uniform, sample_8bits_Uniform_balance] =
5  % Uniform_Quantization(sample)
6
7  % 均匀量化函数
8  % 输入为信号文件，输出为8bits量化（四舍五入）后以及
9  % 均衡后的信号文件
10 % 主要功能为产生8bits量化信号（四舍五入）及均衡后信号，
11 % 并将信噪比计算结果输出到命令窗口
12 % 辅助实现8bits量化（向上取整）、8bits量化（向下取整）、
13 % 8bits量化（取中间值）、nbits量化功能，进行对比
14
15 % 8bits均匀量化（四舍五入）
16 % Bit8_Uniform_Quan记录波形数据
17 % SNR_Bit8记录信噪比
18 Bit8_Uniform_Quan = round(sample * 128) / 128;
19 SNR_Bit8 = SNR_calc(Bit8_Uniform_Quan, sample);
20 fprintf('\n8位均匀量化信噪比:␣%f␣\n',SNR_Bit8);
21 sample_8bits_Uniform = Bit8_Uniform_Quan;
22
23 % 8bits均匀量化（向上取整）
```

```
24 % ceil_Uniform_Quan记录波形数据
25 % SNR_ceil记录信噪比
26 ceil_Uniform_Quan = ceil(sample * 128) / 128;
27 SNR_ceil = SNR_calc(ceil_Uniform_Quan, sample);
28
29 % 8bits均匀量化（向下取整）
30 % floor_Uniform_Quan记录波形数据
31 % SNR_floor记录信噪比
32 floor_Uniform_Quan = floor(sample * 128) / 128;
33 SNR_floor = SNR_calc(floor_Uniform_Quan, sample);
34
35 % 8bits均匀量化（取向上取整和向下取整的中间值）
36 % mid_Uniform_Quan记录波形数据
37 % SNR_mid记录信噪比
38 mid_Uniform_Quan = (floor_Uniform_Quan +
39 ceil_Uniform_Quan) / 2;
40 SNR_mid = SNR_calc(mid_Uniform_Quan, sample);
41
42 %%% 8bits均匀量化均衡（滤波）%%%
43 sample_8bits_Uniform_balance =
44 Sample_Balance(sample_8bits_Uniform, length(sample));
45 SNR_Bit8_balance =
46 SNR_calc(sample_8bits_Uniform_balance, sample);
47 fprintf('\n8位均匀量化均衡信噪比：%f\n',
48 SNR_Bit8_balance);
49
50 %n_bits均匀量化
51 % Bitn_Uniform_Quan记录波形数据
52 % SNR_Bitn记录信噪比
53 syms bit_quan;
```

```
54 bit_quan = input('\n请输入自定义量化位数n：');
55 Bitn_Uniform_Quan = round(sample * (2 ^ (bit_quan - 1)))
56 / (2 ^ (bit_quan - 1));
57 SNR_Bitn = SNR_calc(Bitn_Uniform_Quan, sample);
58 fprintf('\nn位均匀量化信噪比：%f\n',SNR_Bitn);
59
60 % 将得到的信噪比数据将输出到'SNR_Uniform.txt'文件中
61 fid = fopen('SNR_Uniform.txt','wt');
62 fprintf(fid, '8位均匀量化信噪比(round):%f','SNR_Bit8);
63 fprintf(fid, '\n8位均匀量化信噪比(ceil):%f','SNR_ceil);
64 fprintf(fid, '\n8位均匀量化信噪比(floor):%f','SNR_floor);
65 fprintf(fid, '\n8位均匀量化信噪比(mid):%f','SNR_mid);
66 fprintf(fid,'\n8位均匀量化均衡信噪比：%f',
67 SNR_Bit8_balance);
68 fprintf(fid, '\nn位均匀量化信噪比：%f','SNR_Bitn);
69 fclose(fid);
70
71 end
```

6. u_law_Quantization() 函数

```
1 function [sample_ulaw_encoding, sample_ulaw_expansion,
2 sample_ulaw_balance] = u_law_Quantization(sample)
3 % [sample_ulaw_encoding, sample_ulaw_expansion,
4 % sample_ulaw_balance] = u_law_Quantization(sample)
5 % 律编码量化函数
6 % 输入为信号文件，输出为 律压缩、扩张及均衡后的信号文件，
7 % 并将信噪比计算结果输出到命令窗口
8
9 % 实现 律过程
10 % 1.输入 值（本次作业中应输入255）
```

```
% 2. 律压缩（sample_temp1）
% 3.均匀量化（sample_temp2）
% 4. 律扩张（sample_temp3）
% 5.均衡（sample_ulaw_balance）
mu = input('\n请输入 值(正整数, 本次作业中应输入255):');
mu = abs(mu);
sample_temp1 = sign(sample) .* log(1 +
(mu .* abs(sample))) / log(1 + mu);
sample_temp2 = round(sample_temp1 * 256) / 256;
sample_temp3 = sign(sample_temp2) .* ((1 + mu) .^
abs(sample_temp2) - 1) / mu;
Bit8_U_Quan = sample_temp3;

% Bit8_U_Quan记录波形数据
% SNR_Bit8_U记录信噪比
SNR_Bit8_U = SNR_calc(Bit8_U_Quan, sample);
fprintf('\n 律量化信噪比: %f \n',SNR_Bit8_U);
sample_ulaw_encoding = sample_temp2;
sample_ulaw_expansion = Bit8_U_Quan;

%%% 律均衡（滤波）%%%
sample_ulaw_balance = Sample_Balance(
sample_ulaw_expansion, length(sample));
SNR_Bit8_U_balance = SNR_calc(sample_ulaw_balance,
sample);
fprintf(' 律量化均衡信噪比: %f \n',SNR_Bit8_U_balance);

% 将得到的信噪比数据将输出到'SNR_ulaw.txt'文件中
fid = fopen('SNR_ulaw.txt','wt');
fprintf(fid,' 律量化信噪比: %f \n',SNR_Bit8_U);
```

```
41 fprintf(fid,'律量化均衡信噪比:␣%f␣\n',
42 SNR_Bit8_U_balance);
43 fclose(fid);
44
45 end
```

7. A_law_Quantization() 函数

```
1 function [sample_Alaw_encoding, sample_Alaw_expansion,
2 sample_Alaw_balance] = A_law_Quantization(sample)
3 % [sample_Alaw_encoding, sample_Alaw_expansion,
4 % sample_Alaw_balance] = A_law_Quantization(sample)
5 % A律编码量化函数
6 % 输入为信号文件，输出为A律压缩、扩张及均衡后的信号文件，
7 % 并将信噪比计算结果输出到命令窗口
8
9 % 实现A律过程
10 % 1.A律压缩（sample_temp1）
11 % 2.均匀量化（sample_temp2）
12 % 3.A律扩张（sample_temp3）
13 % 4.均衡（sample_Alaw_balance）
14
15 [m, n] = size(sample);
16 A = 87.6 * ones(m, n);
17
18 % 用flag1标记sample中绝对值小于1/A的数据，
19 % flag2标记sample中绝对值介于1/A和1之间的数据
20 flag1 = ceil(1 ./ A - abs(sample));
21 flag2 = ceil(abs(sample) - 1 ./ A);
22 sample_temp1 = sign(sample) .* (A .* flag1 .*
23 abs(sample) ./ (1 + log(A)) + flag2 .*
```

```
24  (1 + log(A .* (abs(sample) + (sample == 0))))
25  ./ (1 + log(A)));
26  sample_temp2 = round(sample_temp1 * 256) / 256;
27  sample_temp3 = sign(sample_temp2) .* (abs(sample_temp2)
28   .* flag1 .* (1 + log(A)) ./ A + flag2 .*
29   exp(abs(sample_temp2) .* (1 + log(A)) - 1) ./ A);
30  Bit8_A_Quan = sample_temp3;
31
32  % Bit8_A_Quan记录波形数据
33  % SNR_Bit8_U记录信噪比
34  SNR_Bit8_A = SNR_calc(Bit8_A_Quan, sample);
35  fprintf('\nA律量化信噪比: %f \n',SNR_Bit8_A);
36  sample_Alaw_encoding = sample_temp2;
37  sample_Alaw_expansion = Bit8_A_Quan;
38
39  %%% A律均衡（滤波）%%%
40  sample_Alaw_balance = Sample_Balance(
41  sample_Alaw_expansion, length(sample));
42  SNR_Bit8_A_balance = SNR_calc(sample_Alaw_balance,
43  sample);
44  fprintf('A律量化均衡信噪比: %f \n',SNR_Bit8_A_balance);
45
46  % 将得到的信噪比数据将输出到'SNR_Alaw.txt'文件中
47  fid = fopen('SNR_Alaw.txt','wt');
48  fprintf(fid,'A律量化信噪比: %f \n',SNR_Bit8_A);
49  fprintf(fid,'A律量化均衡信噪比: %f \n',
50  SNR_Bit8_A_balance);
51  fclose(fid);
52
53  end
```

8. Sample_Balance() 函数

```
function sample_balance = sample_balance(sample,
cnt_point)
% sample_balance = balance(sample)
% 信号均衡函数
% 滤去信号中高于fs/2的频率，由数字信号恢复到模拟信号
% 具体步骤：
% 1.构建窗函数
% 2.频域滤波

window0 = boxcar(cnt_point / 2);
[row, line] = size(sample);
window0 = repmat(window0, 1, line);
window = zeros(row, line);
window(1:cnt_point / 2, 1:line) = window0;

sample_balance = 2 * real(ifft(fft(sample) .* window));

end
```

9. compare() 函数

```
function compare(t, sample1, sample2)
% compare(sample1, sample2)
% 对比函数，比较两种波形的差别
% 实现对比方法：时域波形作差
plot(t, sample1 - sample2);
xlabel('time (seconds)');

end
```

§9 附 2：readme

实验报告及相关文件位于 project 文件夹中.

music.wav: 样例音频

实验报告请见:report.pdf

主函数程序代码请见:Digital_Audio_Processing.m

如若运行主函数代码，将会产生近 1GB 的文件，请当心 具体操作方法请见 report.pdf 的‘附 0：程序使用样例’部分

函数文件：

wavread.m：读取样例音频

SNR_calc.m：计算信噪比

plotTD.m：显示时域波形

plotFD.m：显示频谱图

Uniform_Quantization.m：均匀量化

A_law_Quantization.m：A 律量化

u_law_Quantization.m：律量化

Sample_Balance.m：均衡函数

compare.m：对比函数

average_track.m：双声道平均

reduction.m：衰减函数

函数的具体功能说明和代码请见 report.pdf

‘数据与图像’文件夹内保存有实验过程中的图像文件、音频文件及有关数据：

reduction_sample.wav：衰减后的音频

sample_8bits_Uniform.wav：8bits 均匀量化后的音频

sample_Alaw.wav：A 律量化后的音频

sample_ulaw.wav：μ 律量化后的音频

SNR_Alaw.txt：A 律量化的信噪比

SNR_ulaw.txt：μ 律量化的信噪比

SNR_Uniform.txt：8bits 均匀量化的信噪比

comparison.png：时域对比波形 0

compare1.png：时域对比波形 1

compare2.png：时域对比波形 2

compare3.png：时域对比波形 3

FD_origin.png：原始音频频谱图

zoomup.png：放大的相位谱

FD_A_Law_encoding.png：A 律压缩频谱图

FD_A_Law_expansion.png：A 律扩张频谱图

FD_A_Law_balance.png：A 律均衡频谱图

FD_U_Law_encoding.png：μ 律压缩频谱图

FD_U_Law_expansion.png：μ 律扩张频谱图

FD_U_Law_balance.png：μ 律均衡频谱图

FD_Uniform.png：8bits 均匀量化频谱图

FD_Uniform_balance.png：8bits 均匀量化均衡频谱图

TD_origin.png：原始音频时域波形

TD_A_Law_encoding.png：A 律压缩时域波形

TD_A_Law_expansion.png：A 律扩张时域波形

TD_A_Law_balance.png：A 律均衡时域波形

TD_U_Law_encoding.png：μ 律压缩时域波形

TD_U_Law_expansion.png：μ 律扩张时域波形

TD_U_Law_balance.png：μ 律均衡时域波形

TD_Uniform.png：8bits 均匀量化时域波形

TD_Uniform_balance.png：8bits 均匀量化均衡时域波形