Lab10 Digital Coding of Speech Signals-part2

张旭东 12011923

1. Introduction

Speech coding is to encode the analog speech signal and convert analog speech signal to digital signal, which can reduce the error rate of transmission and carry out digital transmission. The basic methods of speech coding can be divided into waveform coding and parameters coding. Waveform coding is to do sampling, quantization, coding on analog speech signal in time domain and convert them to digital voice signal. While parameters coding is to find out the characteristic parameters of the speech and encode the characteristic parameters based on the pronunciation mechanism of human language.

The purpose of this experiment is to test and understand the process of μ-law quantization, compare uniform and μ-law quantizers on their SNRs and test and understand the process of adaptive quantization.

2. Lab result and analysis

2.1 Problem1

The goal of this MATLAB exercise is to experiment with the process of μ-law compression of speech.

$$y[n] = \frac{ln[1+\mu|x[n]|]}{ln[1+\mu]} sign[x[n]] \tag{1}$$

The function of μ-law compression has been given, which is `y=mulaw(x,mu)`, where `x` is the input signal, `mu` is the compression parameter and `y` is the μ-law compressed signal.

The function is used to plot the μ-law characteristic for $\mu = 1, 20, 50, 100, 255, 500$ with a linearly increasing input vector(-1:0.001:1). The result is shown in `Fig 1`.

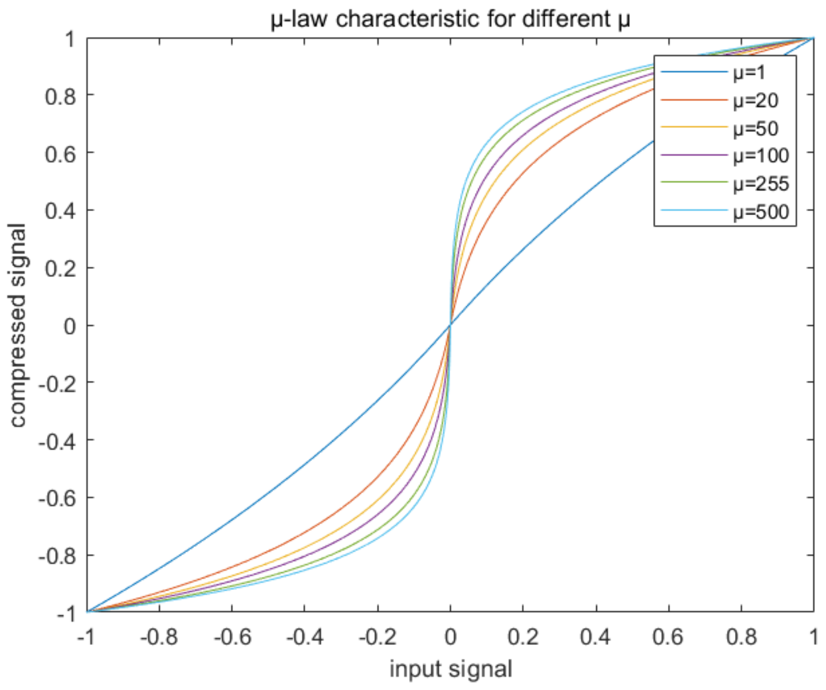


Fig.1 μ-law characteristic for different μ

From the above figure, it is obvious that with μ increasing, the difference between compressed signal whose input is negative signal and that whose input is positive signal become larger and larger. When the input is positive, larger μ maps small value to larger value. And when the input is negative, larger μ maps large value to smaller value.

After that, the segment of speech, from sample 1300 to sample 18800, from the file `s5.wav` is used to test the function under μ=255. The output waveform $y[n]$ of the μ-law compressor and the waveform of original speech segment is shown in `Fig 2`.

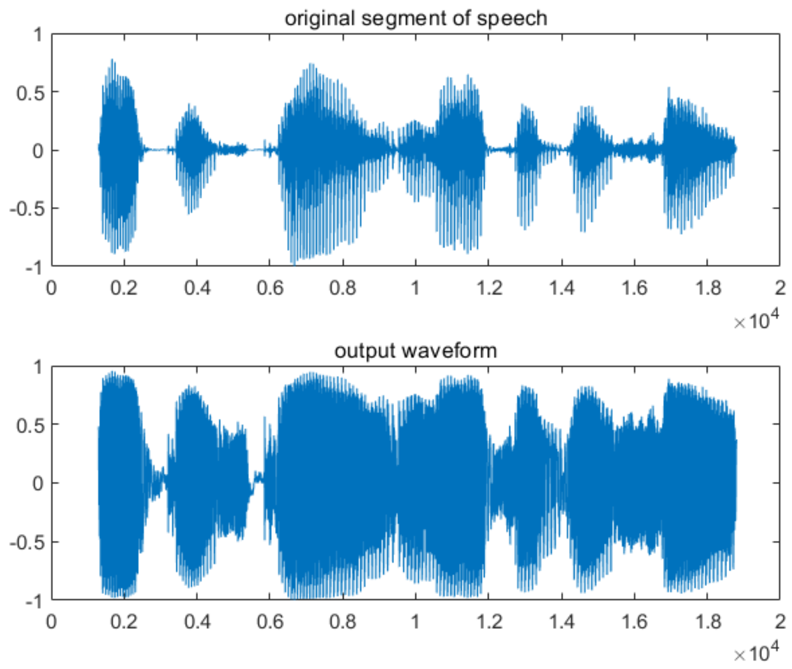


Fig.2 original waveform and output waveform under μ=255

From `Fig 2`, it can be observed that the amplitude of samples with small amplitude has been increased. Besides, the degree of increase in places with larger amplitude is not as large as that in places with small amplitude. The reason behind this can be found from `Fig 1`. When the input is positive, larger μ maps small value to larger value. For a sample whose amplitude is larger, the amplitude of output not grow as significantly as smaller samples.

The histogram of the output samples is shown in `Fig 3`.

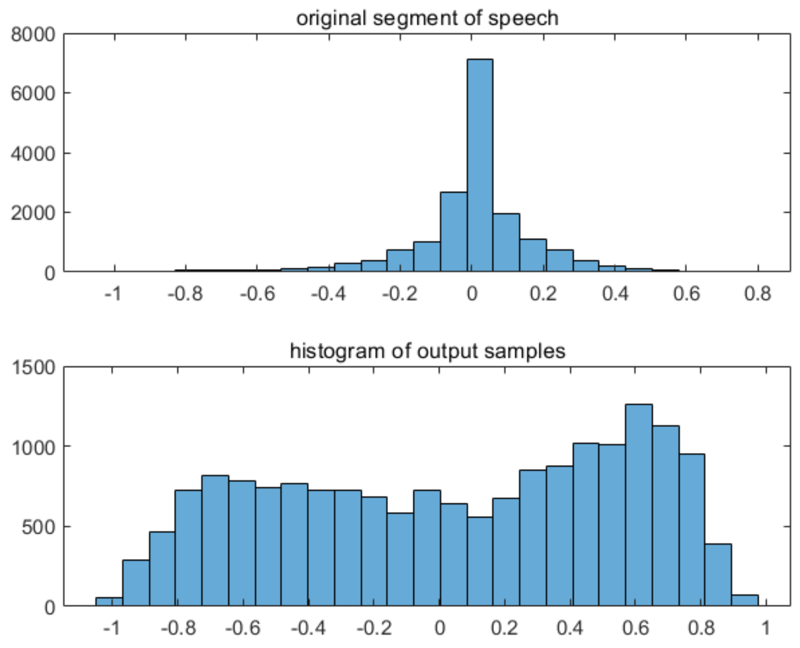


Fig.3 histogram of the output samples

From `Fig 3`, a obvious change can be observed. That is the number of samples in each interval is uniform after being processed by μ-law compression.

The block diagram of a full μ-law quantization method is shown in `Fig 4`.

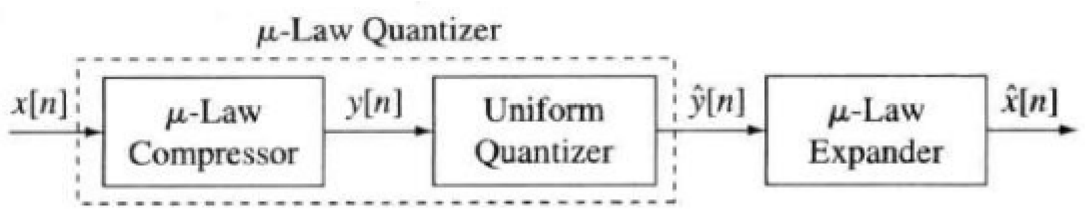


Fig.4 a full μ-law quantization method

The function for the inverse of μ-law compression has been given, which is `x=mulawinv(y,mu)`, where `y` is input column vector, `mu` is mulaw compression parameters and `x` is expanded output vector. A linearly increasing input vector(-1:0.001:1) is used to test the function. The result is shown in `Fig 5`.

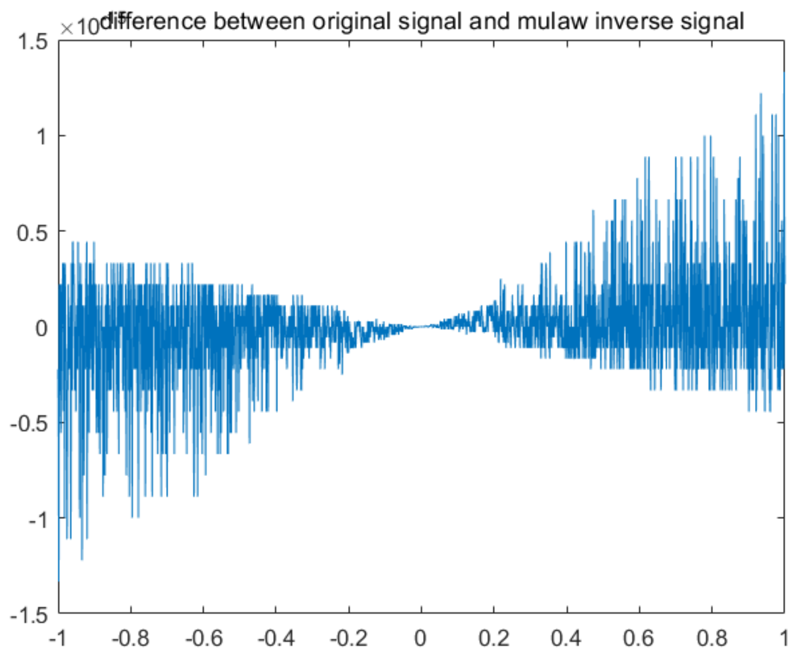


Fig.5 difference between original signal and μ-law inverse

`Fig 5` shows that the difference between the original signal and signals of μ-law inverse is very small, whose order is 10^{-15}. That means the error can be ignored and the function `x=mulawinv(y,mu)` can recover original signal perfectly.

The MATLAB statement `yh=fxquant(mulaw(x,255),6,'round','sat')` implement a 6-bit μ-law quantizer. Thus the μ-law compressed samples are coded by a uniform quantizer with 6 bits. When the coded samples are used in a signal processing computation, or when a continuous-time signal needs to be reconstructed, the uniformly coded samples of the μ-law quantizer must be expanded. Hence, the quantization error is likewise expanded, so that to determine the quantization error, it is necessary to compare the output of inverse system to the original samples. Thus the quantization error is `e=mulawinv(yh,255)-x`. Uniform quantizers with different bits is used and computed and the first 8000 samples of the resulting quantization error is plotted. Histogram of the quantization error amplitudes and power spectrum of the resulting quantization errors is used to analyze the problem.

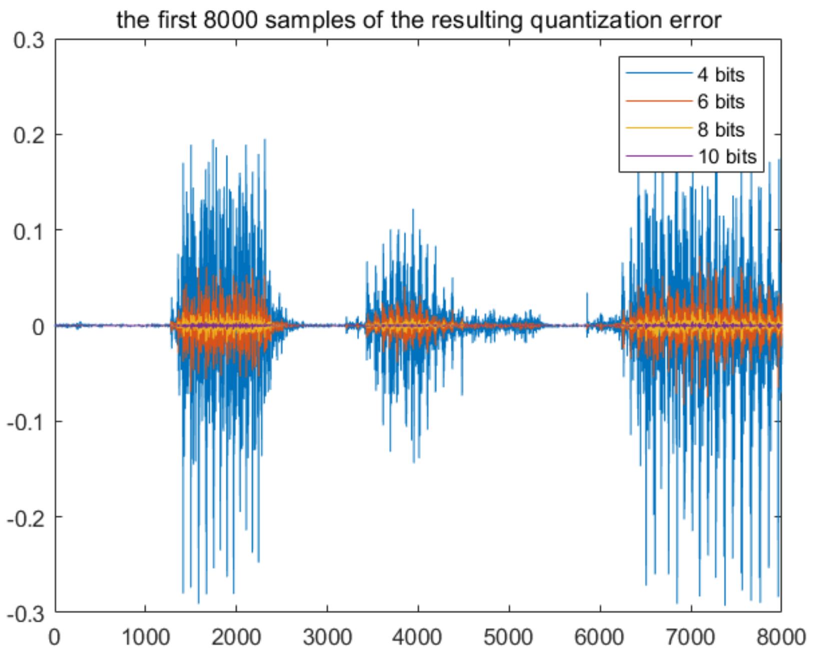


Fig.6 the first samples of the resulting quantization error

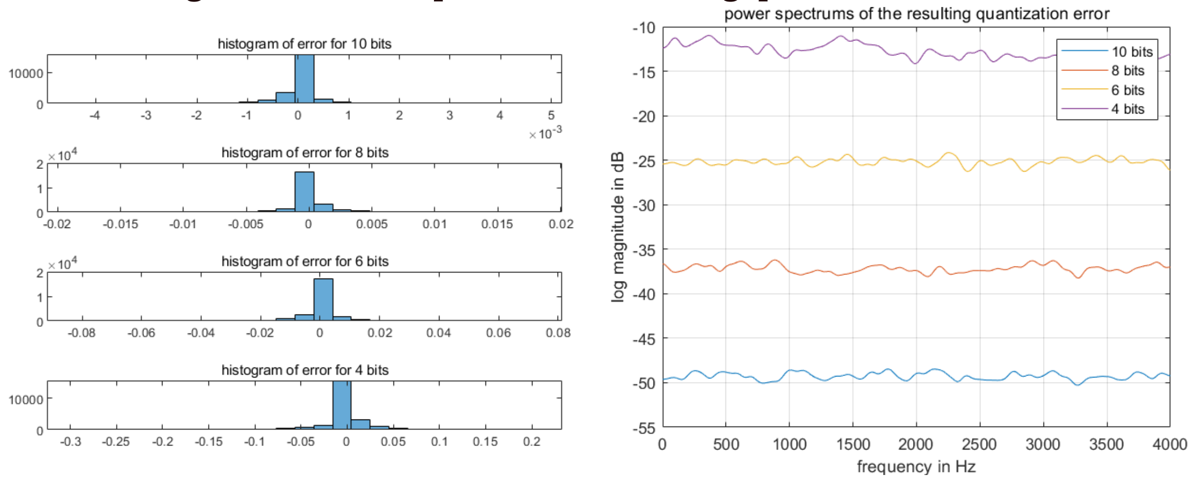


Fig.7 histogram and power spectrum

From the above figures, what can known is that using μ uniform quantizer, the error decrease with the bits used increasing. Besides, the center of histogram is more and more closer to 0 with the bits used increasing. That means the number of error samples whose amplitude is 0 is increasing. What's more, the energy of error decreases with the number of bits used increasing. In a conclusion, the more bits used, the performance of μ uniform quantizer.

The whole code is as below:

```
%a
x=-1:0.001:1;
y_1=mulaw(x,1);
y_20=mulaw(x,20);
y_50=mulaw(x,50);
y_100=mulaw(x,100);
y_255=mulaw(x,255);
y_500=mulaw(x,500);

figure;
plot(x,y_1);hold on
plot(x,y_20);hold on
plot(x,y_50);hold on
plot(x,y_100);hold on
plot(x,y_255);hold on
plot(x,y_500);hold on
legend('μ=1','μ=20','μ=50','μ=100','μ=255','μ=500');
title('μ-law characteristic for different μ');
xlabel('input signal');
ylabel('compressed signal');

%b
[s,fs]=audioread('s5.wav');
segment=s(1300:18800);
y_segment=mulaw(segment,255);
figure;
subplot(2,1,1);
plot(1300:1:18800,segment);
title('original segment of speech');
subplot(2,1,2);
```

```matlab
plot(1300:1:18800,y_segment);
title('output waveform');

figure;
subplot(2,1,2);
y_hist=histogram(y_segment,25);
title('histogram of output samples');
subplot(2,1,1);
original_hist=histogram(segment,25);
title('original segment of speech');

%c
v=mulawinv(mulaw(x,255),255);
figure;
plot(-1:0.001:1,x-v');
title('difference between original signal and mulaw inverse signal');

%d

yh_10=fxquant(mulaw(s,255),10,'round','sat');
yh_8=fxquant(mulaw(s,255),8,'round','sat');
yh_6=fxquant(mulaw(s,255),6,'round','sat');
yh_4=fxquant(mulaw(s,255),4,'round','sat');

error_10=mulawinv(yh_10,255)-s;
error_8=mulawinv(yh_8,255)-s;
error_6=mulawinv(yh_6,255)-s;
error_4=mulawinv(yh_4,255)-s;

figure;
plot(error_4(1:8000));hold on;
plot(error_6(1:8000));hold on;
plot(error_8(1:8000));hold on;
plot(error_10(1:8000));hold on;
legend('4 bits','6 bits','8 bits','10 bits');
title('the first 8000 samples of the resulting quantization error');
```

```
figure
subplot(4,1,1);
error10_hist=histogram(error_10,25);
title('histogram of error for 10 bits');
subplot(4,1,2);
error8_hist=histogram(error_8,25);
title('histogram of error for 8 bits');
subplot(4,1,3);
error6_hist=histogram(error_6,25);
title('histogram of error for 6 bits');
subplot(4,1,4);
error4_hist=histogram(error_4,25);
title('histogram of error for 4 bits');

figure
pspect(error_10,fs,1024,128);hold on;
pspect(error_8,fs,1024,128);hold on;
pspect(error_6,fs,1024,128);hold on;
pspect(error_4,fs,1024,128);hold on;
legend('10 bits','8 bits','6 bits','4 bits');
title('power spectrums of the resulting quantization error');
```

2.2 Problem2

The goal of this MATLAB exercise is to compare uniform and μ-law quantizers on the basis of their SNRs.

A convenient definition of a waveform (with duration L samples) SNR is

$$SNR = 10log(\frac{\sum_{n=0}^{L-1}(x[n])^2}{\sum_{n=0}^{L-1}(\widehat{x}[n] - x[n])^2}) \tag{2}$$

Note that the division by L, as required for averaging, cancels in the numerator and denominator.

It was shown in this chapter that the SNR for a uniform B-bit quantizer was of the form

$$SNR = 6B + 4.77 - 20log_{10}(\frac{X_{max}}{\sigma_x}) \tag{3}$$

where X_{max} is the clipping level of the quantizer and σ_x is the rms value of the input signal amplitude. SNR increases 6 dB per bit added to the quantizer word length. Further the SNR decreases by 6 dB if the signal level is decreased by a factor of 2.

An m-file is written to compute the SNR giving the unquantized and quantized versions of the signal. The code of function `MySNR` is as below:

```
function [s_n_r,e]=MySNR(xh,x)
          e=xh-x;
          s_n_r=10*log10(sum(x.^2)/sum(e.^2));
end
```

The SNRs for uniform quantization with 8 and 9 bits is computed using the function to test the correctness. What's more, according to `formula 3`, another method to calculate the SNR is used to compare the results of two methods.

```
38.2914                                36.6343

44.2102                                42.6343
```

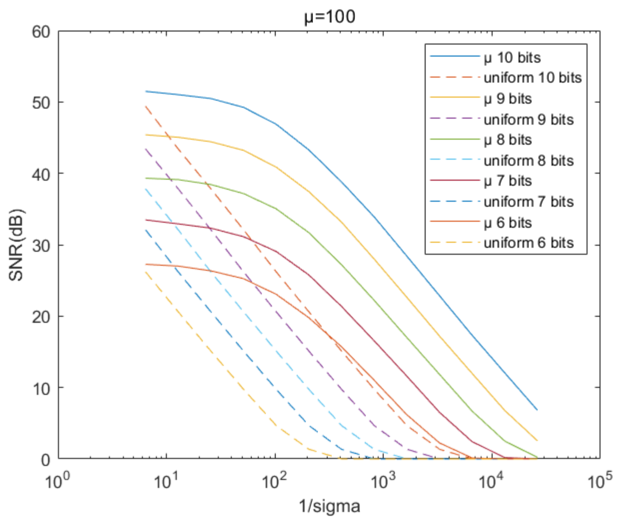
Fig.8 result of MySNR(left) and formula 3

From the result, the SNR from the function written is 38.2914dB for 8 bits and 44.2102dB for 9 bits. The SNR calculated from `formula 3` is 36.6343dB for 8 bits and 42.6343dB for 9 bits. The result is in line with expectation because the SNR for 9 bits is 6dB greater than that for 8 bits.

An important consideration in quantizing speech is that signal levels can vary with speakers and with transmission/recording conditions. A function `SNRplot` is written to plot the measured SNR for uniform and μ-law quantization as a function of $\frac{1}{\sigma_x}$. The code of function `SNRplot` is as below:

```
function [sig]=SNRplot(s,u,bits)
    factor=2.^(0:-1:-12)';
    sig=factor*s';
    sig=sig';
    SNR_uniform=zeros(1,13);
    SNR_u=zeros(1,13);
    deviation=std(sig);
    for i=1:13
        signal_i=sig(:,i);
        signal_q=fxquant(mulaw(signal_i,u),bits,'round','sat');
        signal_expand=mulawinv(signal_q,u);
        SNR_uniform(i)=MySNR(fxquant(signal_i,bits,'round','sat'),signal_i);
        SNR_u(i)=MySNR(signal_expand,signal_i);
    end
    semilogx(1./deviation,SNR_u);hold on;
    semilogx(1./deviation,SNR_uniform,'--');
    xlabel('1/sigma');
    ylabel('SNR(dB)');
end
```

Then measure the SNR for all 13 cases($2\hat{}(0:-1:-12)$) and repeat the calculation for $10, 9, 8, 7$ and 6 bit uniform quantizers and for $\mu = 100, 255, 500$ over the same range. The result is shown in `Fig 9`.



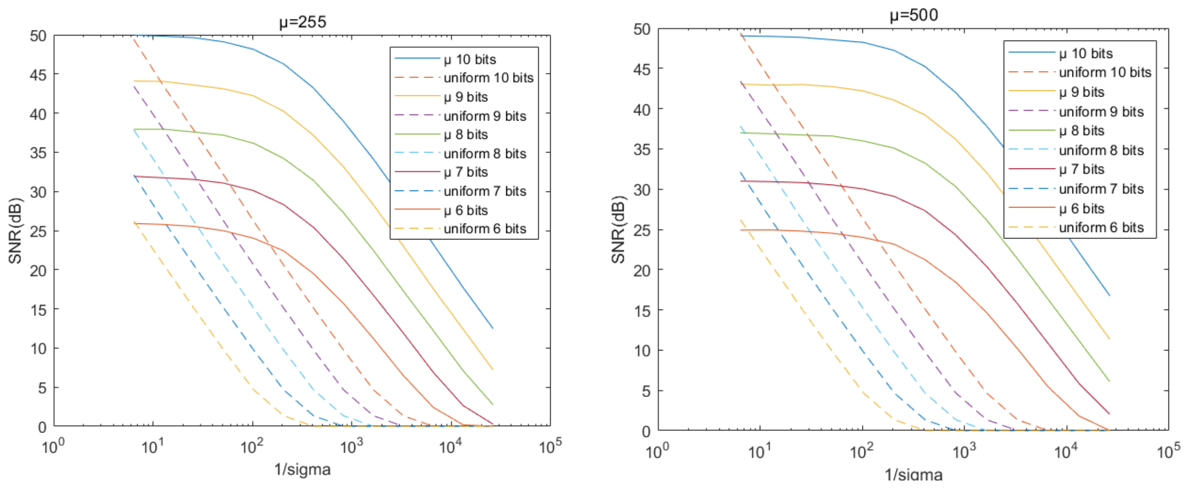


Fig.9 result of each case

From the result, it is easy to find that μ-law has a 6dB difference for each bit value. Compared to uniform way, the dependence on X_{max}/σ_x for the result of μ-law is relatively small. Besides, when the number of bits is the same, the SNR measured by μ-law is larger than that measured by uniform quantization.

When μ is 100, 10 bits are required for a uniform quantizer to maintain at least the same SNR as does a 6-bit, μ-law quantizer over the same range. When μ is 255, about 11 bits are required for a uniform quantizer to maintain at least the same SNR as does a 6-bit, μ-law quantizer over the same range. When μ is 255, about 12 bits are required for a uniform quantizer to maintain at least the same SNR as does a 6-bit, μ-law quantizer over the same range.

The whole code is as below

```
[s,fs]=audioread('s5.wav');
%a
x=s(1300:18800);
xh_8=fxquant(x,8,'round','sat');
xh_9=fxquant(x,9,'round','sat');

[snr_8,e_8]=MySNR(xh_8,x);
[snr_9,e_9]=MySNR(xh_9,x);
disp(snr_8);
disp(snr_9);

th_snr_8=6*8+4.77-20*log10(1/sqrt(var(s)));
th_snr_9=6*9+4.77-20*log10(1/sqrt(var(s)));
```

```
disp(th_snr_8);
disp(th_snr_9);

%b
u_100=100;
u_255=255;
u_500=500;
figure
for i=10:-1:6
    SNRplot(s,u_100,i);
end
title('μ=100');
legend('μ 10 bits','uniform 10 bits','μ 9 bits','uniform 9 bits','μ 8
bits','uniform 8 bits','μ 7 bits','uniform 7 bits','μ 6 bits','uniform 6
bits');
figure
for i=10:-1:6
    SNRplot(s,u_255,i);
end
title('μ=255');
legend('μ 10 bits','uniform 10 bits','μ 9 bits','uniform 9 bits','μ 8
bits','uniform 8 bits','μ 7 bits','uniform 7 bits','μ 6 bits','uniform 6
bits');
figure
for i=10:-1:6
    SNRplot(s,u_500,i);
end
title('μ=500');
legend('μ 10 bits','uniform 10 bits','μ 9 bits','uniform 9 bits','μ 8
bits','uniform 8 bits','μ 7 bits','uniform 7 bits','μ 6 bits','uniform 6
bits');
```

2.3 Problem3

The goal of this MATLAB exercise is to demonstrate the process of adaptive quantization using either an IIR filter or an FIR filter. There are two ways of adapting the quantizer, namely gain adaptation or step size adaptation. Both adaptive methods require the estimation of the signal standard deviation, $\sigma[n]$. Assuming that the signal has a mean of zero, the signal variance, $\sigma^2[n]$, can be computed using an IIR filter of the form:

$$H(z) = \frac{(1-\alpha)^{z-1}}{(1-\alpha z^{-1})} \tag{4}$$

as:

$$\sigma^2[n] = \alpha\sigma^2[n-1] + (1-\alpha)x^2[n-1] \tag{5}$$

or, using a rectangular window FIR filter of length M samples, as:

$$\sigma^2[n] = \frac{1}{M}\sum_{m=n-M+1}^{n} x^2[m] \tag{6}$$

An IIR filter with values of α of 0.9 and 0.99 is used to calculate the standard deviation, $\sigma[n]$, using the speech file `s.wav`. The speech samples, $x[n]$, and the superimposed standard deviation samples, $\sigma[n]$ and the gain equalized speech samples, $\frac{x[n]}{\sigma[n]}$ for the sample rang of $2700 \le n \le 6700$ are plotted. The result is shown in `Fig 10`.

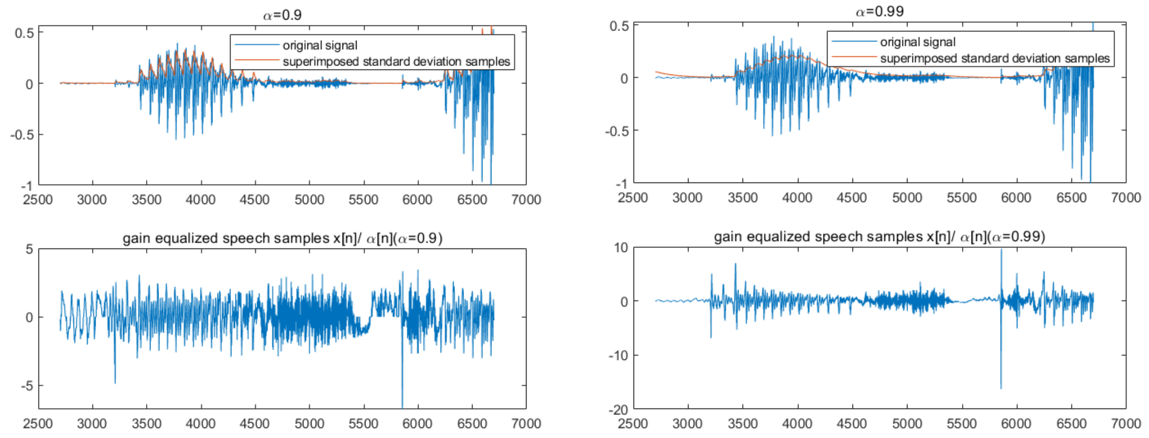


Fig.10 superimposed standard deviation samples and gain equalized speech samples with an IIR filter

Both of waveforms of superimposed standard deviation samples with $\alpha = 0.9$ and $\alpha = 0.99$ look like the envelope of the original signal and superimposed standard deviation with $\alpha = 0.99$ is smoother than that of $\alpha = 0.9$. However, superimposed standard deviation with $\alpha = 0.9$ is better than that of $\alpha = 0.99$ in terms of rate of adapting to the changing speech level. What's more, its ability to equalize the local dynamic range of the speech signal is better.

Then, repeat the above exercise using an FIR filter (rectangular window) of duration $M = 10$ and $M = 100$ samples. The speech samples, $x[n]$, and the superimposed standard deviation samples, $\sigma[n]$ and the gain equalized speech samples, $\frac{x[n]}{\sigma[n]}$ for the sample rang of $2700 \le n \le 6700$ are plotted. The result is shown in `Fig 11`.

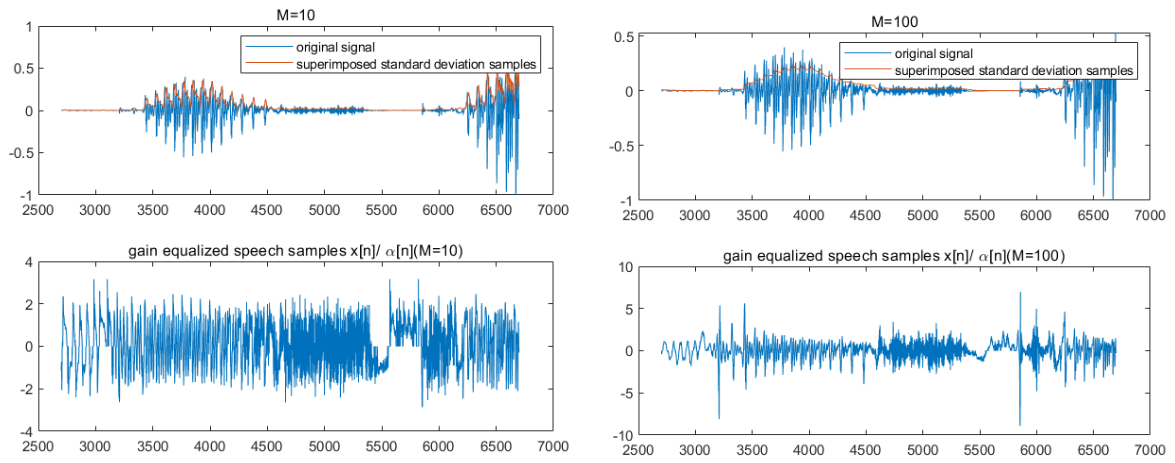


Fig.11 superimposed standard deviation samples and gain equalized speech samples with an FIR filter

From `Fig 11`, it is easy to find that with the length of window increasing, the curve of superimposed standard deviation samples become more and more smooth. However, this means more details is lost. Compared to the result using the IIR filter, the amplitude of gain equalized speech samples using the FIR filter is smaller. The reason behind this is that the number of samples used in method using the IIR filter is less than that used in method using the FIR filter.

The whole code is as below:

```
[s,fs]=audioread('s5.wav');
alpha_1=0.9;
alpha_2=0.99;
va1=zeros(1,length(s));
va2=zeros(1,length(s));
%IIR
```

```matlab
for i=2:length(s)
    va1(i)=alpha_1*va1(i-1)+(1-alpha_1)*(s(i-1)^2);
    va2(i)=alpha_2*va2(i-1)+(1-alpha_2)*(s(i-1)^2);
end

va1_std=sqrt(va1);
va2_std=sqrt(va2);

figure
subplot(2,1,1);
plot(2700:1:6700,s(2700:6700));hold on
plot(2700:1:6700,va1_std(2700:6700));
title('\alpha=0.9');
legend('original signal','superimposed standard deviation samples');
subplot(2,1,2);
plot(2700:1:6700,(s(2700:6700))./(va1_std(2700:6700)'));
title('gain equalized speech samples x[n]/ \alpha[n](\alpha=0.9)');

figure
subplot(2,1,1);
plot(2700:1:6700,s(2700:6700));hold on
plot(2700:1:6700,va2_std(2700:6700));
title('\alpha=0.99');
legend('original signal','superimposed standard deviation samples');
subplot(2,1,2);
plot(2700:1:6700,(s(2700:6700))./(va2_std(2700:6700)'));
title('gain equalized speech samples x[n]/ \alpha[n](\alpha=0.99)');

%FIR
va1_FIR=zeros(1,length(s));% M=10
va2_FIR=zeros(1,length(s));% M=100
M1=10;
M2=100;
for i=M1:length(s)
    va1_FIR(i)=1/M1*sum(s(i-M1+1:i).^2);
end
for i=M2:length(s)
```

```
    va2_FIR(i)=1/M2*sum(s(i-M2+1:i).^2);
end
va1_FIR_std=sqrt(va1_FIR);
va2_FIR_std=sqrt(va2_FIR);

figure
subplot(2,1,1);
plot(2700:1:6700,s(2700:6700));hold on
plot(2700:1:6700,va1_FIR_std(2700:6700));
title('M=10');
legend('original signal','superimposed standard deviation samples');
subplot(2,1,2);
plot(2700:1:6700,(s(2700:6700))./(va1_FIR_std(2700:6700)'));
title('gain equalized speech samples x[n]/ \alpha[n](M=10)');

figure
subplot(2,1,1);
plot(2700:1:6700,s(2700:6700));hold on
plot(2700:1:6700,va2_FIR_std(2700:6700));
title('M=100');
legend('original signal','superimposed standard deviation samples');
subplot(2,1,2);
plot(2700:1:6700,(s(2700:6700))./(va2_FIR_std(2700:6700)'));
title('gain equalized speech samples x[n]/ \alpha[n](M=100)');
```

3. Conclusion

The main harvest in this experiment is to understand the process of μ-law quantization and the process of adaptive quantization, compare uniform and μ-law quantizers on their SNRs. For μ-law quantization, larger μ maps small value to larger value when the input is positive. Besides, when the number of bits is the same, the SNR measured by μ-law is larger than that measured by uniform quantization. What's more, for adaptive quantization, the amplitude of gain equalized speech samples using the FIR filter is smaller than that of using the IIR filter. Using the IIR filter, the performance is better with smaller α in terms of rate of adapting to the changing speech level and the ability to equalize the local dynamic range of the speech signal is better. Using the FIR filter, the performance is better with shorter duration in terms of rate of adapting to the changing speech level and the ability to equalize the local dynamic range of the speech signal is better.