Experiment 3

MULTIMEDIA SIGNAL COMPRESSION: SPEECH AND AUDIO

I Introduction

A key technology that enables distributing speech and audio signals without mass storage media or transmission bandwidth is compression, also known as coding. It reduces the amount of data needed to transmit and store digitally sampled audio either during analog-to-digital conversion step or after the raw file is stored digitally. Audio compression and decompression can be accomplished by various types of algorithms, which can be incorporated in software applications or programmed into special-purpose integrated-circuit chips.

Several international standards have been established for audio and video coding. These include the MPEG-1 and MPEG-2. No international or national standards have been established for compressing and decompressing the waveform speech and audio files for desktop multimedia applications. Yet there are many schemes that user can choose from for compressing the waveform files. The following sections talk about the most commonly used algorithms and various types of compression methods for audio and speech compression.

II Theories and Schemes

We have already discussed the sampling theorem in experiment 2. There it was shown that samples of an analog signal are a unique representation of the signal if the analog signal is bandlimited and if the sampling rate is at least twice the signal frequency. Since we are concerned with digital representations of speech and audio signals, we need to consider the spectral properties of speech and audio. It has been observed that for voiced sounds, the high frequencies above 4 kHz are more than 40 dB below the peak of the spectrum. On the other hand, for audio signals, the spectrum does not fall off appreciably even above 8 kHz. Thus, to accurately represent all audio sounds would require a sampling rate greater than 20 kHz. In addition, for the computer to represent a sampled signal, the possible values taken by a sample, which varies in a continuous range, must be discretized to a finite set of values. This process is called quantization.

II.1 Quantization of Sampled Signals

II.1.1 Uniform Quantization

The quantization ranges and levels may be chosen in a variety of ways depending on the intended applications of the digital representation. With uniform quantization, the dynamic range (minimum to maximum) of the signal R is divided into L equal sized intervals, each with length Δ. We call Δ the quantization step-size. The input (unquantized value) and output (quantized value) relationship in a uniform quantizer is shown in Fig. 3.1. There, x_i represents the right boundary of interval i, and $\hat{x}_i$ the quantization level of this interval. They satisfy

$$x_i - x_{i-1} = \Delta \qquad (3.1)$$

and $$\hat{x}_i - \hat{x}_{i-1} = \Delta . \qquad (3.2)$$

Any value in the *i*-th interval is mapped into the middle value in this interval, i.e.

$$Q(x) = \hat{x}_i = X_{min} + (i-1)\,\Delta + \Delta/2, \text{ if } x_{i-1} <= x < x_i \qquad (3.3)$$

In the computer, each level is represented by a binary codeword. With a total of L quantized levels, each level can be represented by $B=[log_2(L)]$ bits, as shown in Fig. 3.1.

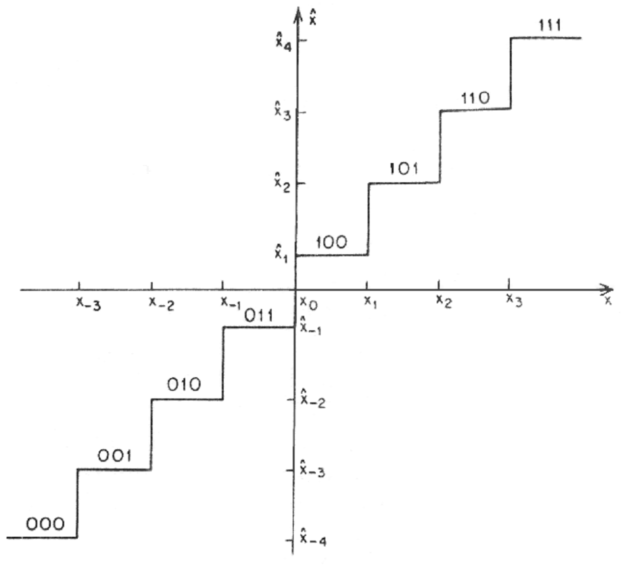


Fig. 3.1 Input-output characteristic of a 3-bit quantizer.

Given the signal range R, a uniform quantizer has only one parameter: the number of levels N or the quantization step size Δ, as the two are related by

$$\Delta = R / N . \qquad (3.4)$$

The number of levels *N* is generally chosen to be of the form 2^B so as to make the most efficient use of B-bit binary code words. If the signal has a symmetrical probability density function so that $|x(n)| \le X_{\max}$, or R= 2 $X_{\max}$, then we should set

$$2\,X_{\max} = \Delta\, 2^{B} \;\text{ or }\; \Delta = \frac{2\,X_{\max}}{2^{B}} \;. \tag{3.5}$$

In discussing the effect of quantization it is helpful to represent the quantized samples $\hat{x}(n)$ as

$$\hat{x}(n) = x(n) + e(n) \tag{3.6}$$

where *x(n)* is the unquantized sample and e(n) is the quantization error or noise. It can be seen from both Fig. 3.1 that if Δ and B are chosen as in Eq.(3.5), then

$$-\frac{\Delta}{2} \le e(n) \le \frac{\Delta}{2} \tag{3.7}$$

Define the signal-to-quantization noise ratio as

$$SNR = \frac{\sigma_x^2}{\sigma_e^2} = \frac{E[x^2(n)]}{E[e^2(n)]} = \frac{\sum_n x^2(n)}{\sum_n e^2(n)} . \tag{3.9}$$

Recall that for a signal with a uniform distribution in a range of R, the variance is $\frac{R^2}{12}$. If we assume a uniform amplitude distribution in $(-\frac{\Delta}{2}, \frac{\Delta}{2})$ for the noise, we obtain

$$\sigma_e^2 = \frac{\Delta^2}{12} = \frac{x_{\max}^2}{(3)2^{2B}} \tag{3.10}$$

Substituting Eq. (3.10) into Eq. (3.9) gives

$$SNR = \frac{(3)2^{2B}}{\left[\dfrac{X_{\max}}{\sigma_x}\right]^2} \tag{3.11}$$

or expressing the signal-to-quantizating error in dB units,

$$SNR = 10\log_{10}\left[\frac{\sigma_x^2}{\sigma_e^2}\right] = 6B + 4.77 - 20\log_{10}\left[\frac{X_{\max}}{\sigma_x}\right]. \tag{3.12}$$

If we assume that the quantizer range is such that $X_{\max} = 4\sigma_x$, then Eq. (3.12) become

$$SNR = 6B - 7.2(dB)\,. \tag{3.13}$$

This implies that every additional bit contribute to 6 dB improvement in SNR. The actual SNR value for a given B depends on the relation between $X_{\max}$ and σ_x, which depends on the probability distribution of the signal. In order to maintain a fidelity of representation with uniform quantization so that it is acceptable perceptually, it is necessary to use more bits than might be implied by the previous analysis in which we have assumed that the signal is stationary and has a symmetric distribution and that $X_{\max} = 4\sigma_x$. For example, whereas Eq. (3.13) suggests that B=7 would provide about 36 dB SNR which would most likely provide adequate quality in a communication system, it is generally accepted that about 11 bits are required to provide high quality representation of speech signals with a uniform quantizer.

II.1.2 μ-Law

The uniform quantizer is only optimal for uniformly distributed signal. For a signal that is more concentrated near small amplitude values, e.g. a Gaussian distribution with a zero mean, it is desirable to quantize more finely small amplitudes. This can be accomplished by first apply a mapping to the signal so that small values are boosted, and then apply a uniform quantization to the mapping signal. One of such mappings is

$$\begin{aligned} y(n) &= F[x(n)] \\ &= X_{\max}\frac{\log\left[1+\mu\frac{|x(n)|}{X_{\max}}\right]}{\log[1+\mu]}.sign[x(n)] \end{aligned} \tag{3.14}$$

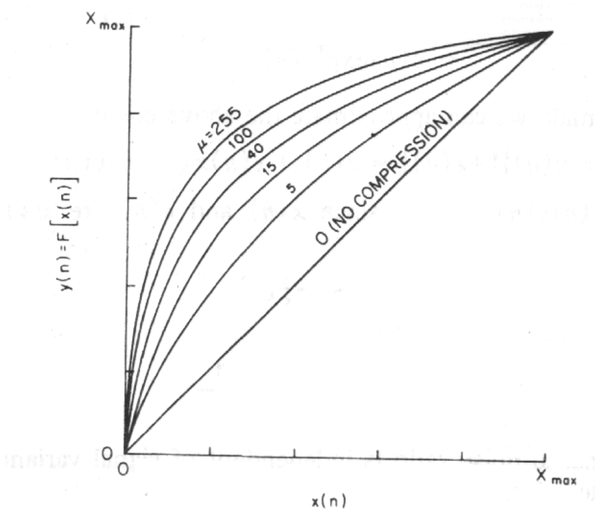


Fig 3.2 Input-output relations for a μ-law characteristics. (after Smith [2])

Fig. 3.2 shows a family of curves of *y(n)* verse *x(n)* for different values of μ. It is clear that using the function of Eq.(3.14) small input amplitudes are enhanced. Fig. 3.3 shows the distribution of quantization levels for the case μ = 40 and N=8. If μ = 0, Eq. (3.14) reduces to *y(n)* = *x(n)*; i.e., the quantization levels are uniformly spaced. However, for large μ, and for large |*x(n)*|,

$$|y(n)| \approx X_{\max} \log \left| \frac{x(n)}{X_{\max}} \right| \tag{3.15}$$

or $$|x(n)| \approx X_{\max} 10^{\frac{|y(n)|}{x_{\max}}}$$

Thus except for very low amplitudes, the quantization levels increases exponentially with the quantization index. This quantization is called μ-law quantization and is first proposed by Smith [2].

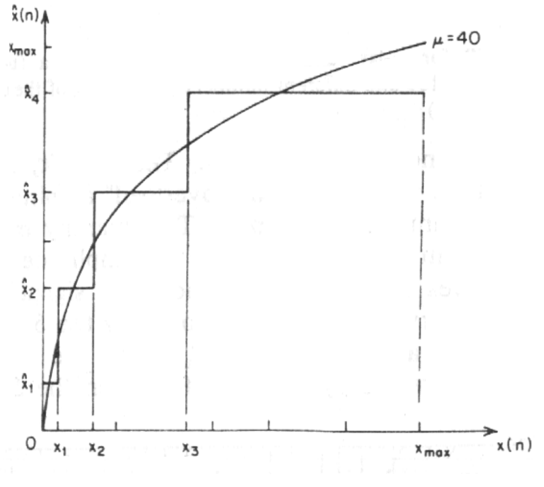


Fig 3.3 Distribution of quantization levels for a μ-law 3-bit quantizer with μ = 40. From [1].

Employing the same kind assumptions that were used to analyze the uniform quantization case, Smith [2] derived the following formula for the signal-to-quantizing noise ratio for a μ-law quantizer:

$$\begin{aligned} SNR &= 6B + 4.77 - 20\log_{10}[\ln(1+\mu)] \\ &\quad - 10\log_{10}\left[1 + \left[\frac{X_{\max}}{\mu\ \sigma_x}\right]^2 + \sqrt{2}\left[\frac{X_{\max}}{\mu\ \sigma_x}\right]\right] \end{aligned} \tag{3.16}$$

This equation, when compared to Eq. (3.13), indicates a much less severe dependence of SNR upon the quantity ($X_{\max}/\sigma_x$), which depends on the signal distribution. It can be seen that as μ increases, the SNR becomes less and less sensitive to the changes in ($X_{\max}/\sigma_x$); i.e., although the term $-20\cdot\log_{10}[\ln(1+\mu)]$ reduces the SNR, the range of ($X_{\max}/\sigma_x$) for which the SNR is constant increases with μ. Therefore, using a large μ the quantizer performance less sensitive to the variation in signal statistics.

II.2 Predictive Coding

In a typical speech waveform, adjacent samples take similar values, except at transitions between different phonemes. One way to exploit this correlation is by linear prediction coding. It first predicts a present sample $x(n)$ using a linear combination of previously reconstructed samples $\widehat{x}(n-k)$ so that

$$x_p(n) = \sum a_k \, \widehat{x}(n-k)$$

Then the error between the actual samples value and the predicted ones,

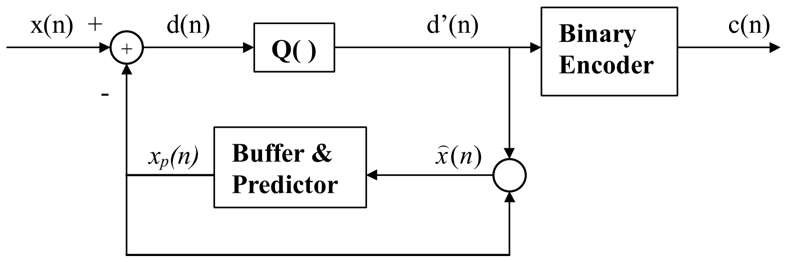
$$d(n) = x(n) - x_p(n)$$

is quantized to $\widehat{d}(n)$ and coded into a codeword *c(n).*

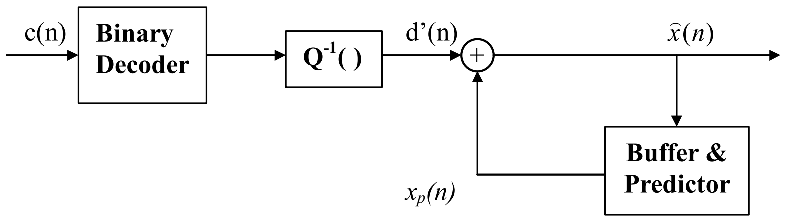
In the decoder, the same predicted value is first produced from previously decoded samples. This value is then added to the decoded quantized error value to yield the quantized value for the current samples, i.e.

$$\hat{x}(n) = x_p(n) + \hat{e}(n).$$

The encoder and decoder block diagrams of a predictive coding system are given in Fig. 3.4. A predictive coding system is more commonly known *as differential pulse coded modulation or DPCM*. The word “differential” refers to the fact that a prediction error signal is coded, and “PCM” refers to a modulation scheme where each coded bit is symbol is represented by a pulse (with amplitude 1 or 0). Directly quantizing an original sample uniformly followed by a fixed length binary encoding is called PCM.



(a)



(b)

Figure 3.4 Predictive Coding: (a) Encoder; (b) Decoder

II.2.1 Delta Modulation

A simple predictive coding system is the delta modulation (DM) system depicted in Fig. 3.5. In this case the quantizer for the prediction error has only two levels and the step size is fixed. The positive quantization level is represented by *c(n)* = 0 and the negative by *c(n)* = 1. Thus, $\hat{d}(n)$ is

$$\begin{cases} \hat{d}(n) = \Delta & or \quad c(n) = 0 & if \quad d(n) \geq 0 \\ \hat{d}(n) = -\Delta & or \quad c(n) = 1 & if \quad d(n) < 0 \end{cases} \tag{3.17}$$

A simple first order prediction is used, i.e. $x_p(n) = \hat{x}(n-1)$. It can be seen from Fig. 3.5(a) that in general, $\hat{x}(n)$ satisfies the difference equation

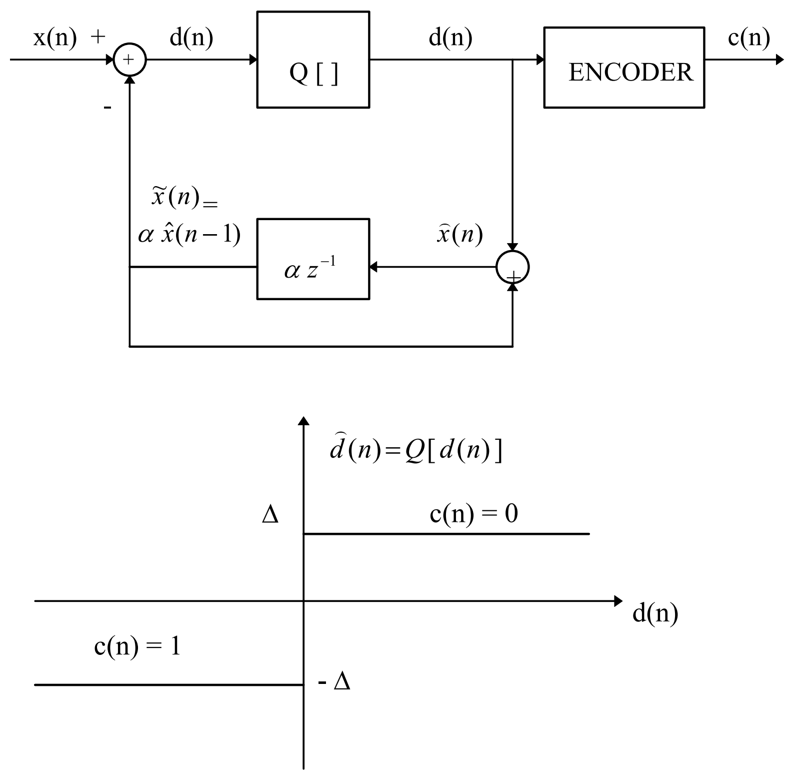
$$\hat{x}(n) = \alpha\, \hat{x}(n-1) + \hat{d}(n) \tag{3.18}$$

With $\alpha = 1$, this equation is the digital equivalent of integration, in the sense that it represents the accumulation of positive and negative increments of magnitude Δ. We also note that input to the quantizer is

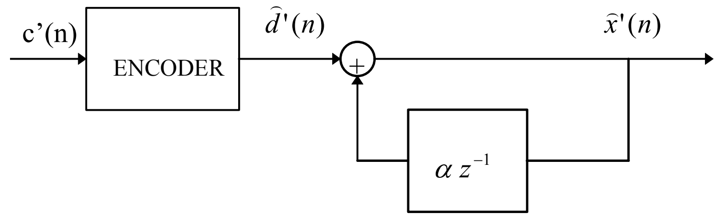
$$d(n) = x(n) - \hat{x}(n-1) = x(n) - x(n-1) - e(n-1) \qquad (3.19)$$

Thus except for the quantization error in $\hat{x}(n-1)$, *d(n)* is a first order backward difference of *x(n)*, which can be viewed as a digital approximation to the derivative of the input, the inverse of the digital integration process.

Because the error quantization is only two-level, the delta modulation has a bit rate of 1 bit/sample. If it is applied to a 16 bit/sample sequence, then it leads to a compression ratio (CR) of 16.



(a)



(b)

Fig. 3.5 Block diagram of a delta modulation system; (a) encoder; (b) decoder.

For the delta modulation to work well, the step size must be chosen properly to match the signal variation. This is a difficult task as the signal characteristics often changed from tone to tone. Fig 3.6(a) illustrates the quantization process of delta modulation with a fine step size. We can see that the step size is too small in the beginning, which causes the quantized signal lags below the actual signal magnitude. On the other hand, the step size is too large in the latter portion, which causes the quantized signal to oscillate about the actual signal. For a better performance, the step size should be adaptively adjusted, which is the subject of the next session.

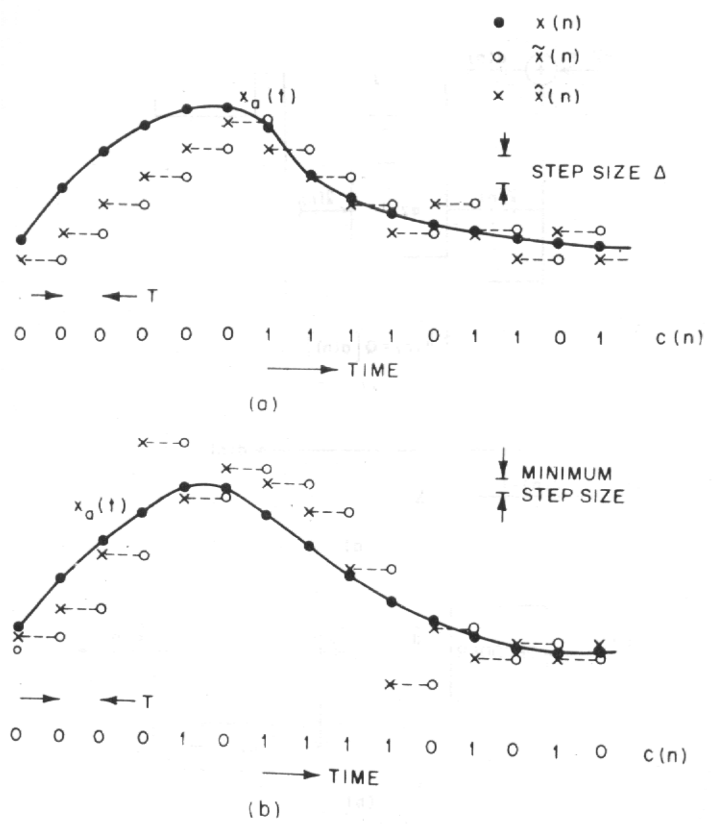


Fig. 3.6 Illustration of delta modulation; (a) using a fixed step-size; (b) using an adaptive step-size.

II.2.2 Adaptive Delta Modulation

A large variety of adaptive delta modulation (ADM) schemes have been proposed. Most of these schemes are of the feedback type, in which the step size for the two-level quantizer is adapted based on the output code words. The system that we present below was proposed by Jayant [3]. The step size in Jayant's algorithm obeys the rule

$$\Delta(n) = M\,\Delta(n-1) \tag{3.20a}$$

$$\Delta_{\min} \le \Delta(n) \le \Delta_{\max} \tag{3.20b}$$

The algorithm for choosing the step size is

$$M = P > 1 \qquad \text{if } c(n) = c(n-1)$$

$$M = Q < 1 \qquad \text{if } c(n) \neq c(n-1) \tag{3.21}$$

Fig. 3.6(b) shows how the waveform in Fig. 3.6(a) would be quantized by an adaptive delta modulator of the type described by Eqs. (3.20) and (3.21). For convenience, the parameters of the system are set at P = 2, Q = 1/2, $\alpha = 1$, and the minimum step size is shown in the figure. It can be seen that the region of large positive slope still causes a run of 0's but in this case the step size increases exponentially so as to follow the increase in the slope of the waveform. The region of granularity to the right in the figure is again signaled by an alternating sequence of 0's and 1's but in this case the step size falls rapidly to the minimum ($\Delta_{\min}$) and remains there as long as the slope is small.

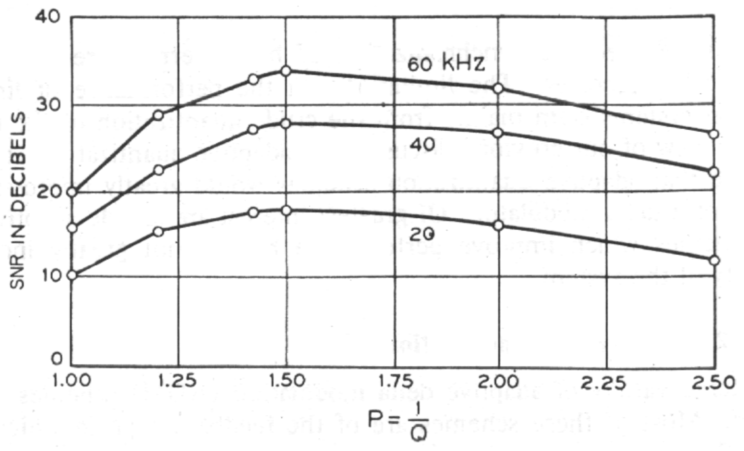


Fig. 3.7 Signal-to-noise ratios of an adaptive delta modulator as functions of P.

Figure 3.7 shows the results of a simulation for speech signals with PQ = 1 for three different sampling rates. It is evident that the maximum SNR is obtained for P = 1.5; however, the peak of all three curves is very board with SNR being within a few dB of the maximum for 1.25 < P < 2. *Notice that for the delta modulation to work well, the signal must be sampled at a much higher rate than that called for by the Nyquist sampling theorem, so that the variation between adjacent samples is small.* This phenomenon in fact reveals a fundamental trade-off between the sampling resolution and amplitude resolution. That is, *to reduce the amplitude resolution (1 bit in the delta modulation) one must increase the sampling resolution.*

In this experiment, you will be asked to play with delta modulations with a fixed and adaptive step size.

II.2.3 Higher Order DPCM

Delta modulators, as discussed in the previous section, for examples, could also be called 1-bit DPCM systems. In general, One can use more than one previous sample to predict a present sample. Also, one can use a quantizer with more than two levels. For a theoretical treatment on how to best determine the predictor coefficients and how to design an optimal quantizer. See [1]. Generally, the term DPCM is reserved for differential quantization systems in which the quantizer has more than two levels. DPCM systems with fixed predictors can provide from 4 to 11 dB improvement over direct quantization (PCM). The greatest improvement occurs in going from no prediction to first order prediction, with somewhat smaller gains resulting from increasing the predictor order up to 4 or 5, beyond which little additional gain results. For speech, a predictor of length up to 10 is used, because the speech signal can be modeled well by a higher order system. The gain in SNR implies that a DPCM system can achieve a given SNR using less bits than would be required when using the same quantizer directly on the speech waveform. Recall that when quantizing a signal directly, each additional bit leads to a gain of 6 dB. Therefore, if a DPCM system can leads to a prediction gain of 6 dB, than it will require 1 less bit than a PCM system, to achieve the same signal quality.

II.2.4 ADPCM

There are two major schemes for adaptive DPCM or ADPCM. One is DPCM with adaptive quantization, and the other is DPCM with adaptive prediction.

For DPCM with adaptive quantization, the quantizer step size is proportional to the variance of the input to the quantizer. However, since the difference signal *d(n)* will be proportional to the input, it is reasonable to control the step size from the input *x(n)* as depicted in Fig. 3.8. Several algorithms for adjusting the step size had been proposed in the past. And results indicate that such adaptation procedures can provide about 5 dB improvement in SNR over standard μ-law non-adaptive PCM. This improvement coupled with the 6 dB that can be obtained from the differential configuration with fixed prediction means that ADPCM with feed-forward adaptive prediction should achieve a SNR that is 10 -- 11 dB greater than could be obtained with a PCM system with the same number of levels.

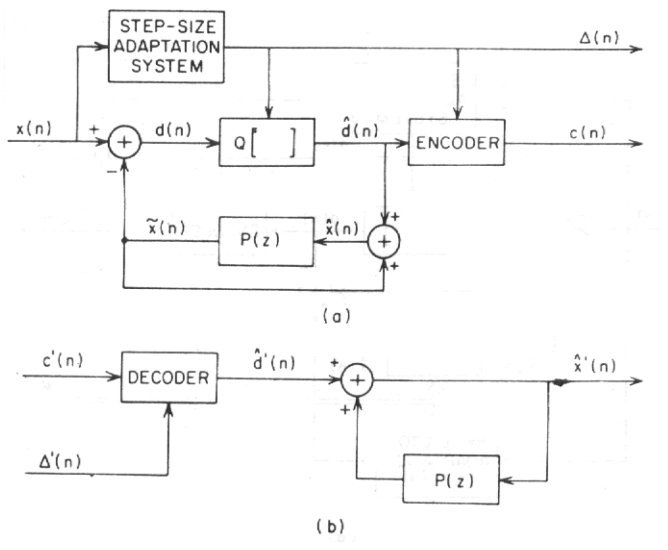


Fig. 3.8 ADPCM system with feed-forward adaptive quantization; (a) encoder, (b) decoder.

For DPCM with adaptive prediction, the predictor coefficients are assumed to be time dependent so that the predicted value is

$$\tilde{x}(n)=\sum_{k=1}^{p}\alpha_k(n)\,\hat{x}(n-k) \tag{3.23}$$

In adapting the predictor coefficients $\alpha_k(n)$ it is common to assume that the properties of the signal remain fixed over short time intervals. The predictor coefficients are therefore chosen to minimize the average squared prediction error every short time window. To learn how to derive the optimum predictor coefficients, you are encouraged to read [1].

II.3 Speech Coding Standards

There are many international standards that have been developed for coding speech signals. A partial list is given below. Except G.711 standard, essentially they all use some form of ADPCM. The coders the achieve much lower rate than conventional ADPCM is by coding a group of prediction error samples together as an excitation pattern.

1) CCITT G. 711 (A-LAW and μ-LAW)

2) CCITT G. 721 (ADPCM at 32 kbits/ sec)

3) CCITT G. 723 (CELP Based 5.3 and 6.3 kbits/sec)

4) GSM 06.10 is speech encoding used in Europe (13 kbits/sec)

5) U.S. Federal Standard :

(a) 1016 (Code excited linear prediction (CELP),4800 bits/s)

(b) 1015 (LPC-10E, 2400 bits/s).

6) MPEG Audio (refer to appendix MPEG-1 Audio)

III Experiment

1) The Matlab program given in Appendix "demo_quant.m" performs uniform quantization on a sound signal. Read through the given program to understand how it works. Run the program on a speech file recorded at 8 bits/sample and on a music file at 16 bits/sample. For each case, compare the distortion in waveform as well as sound quality obtained with different choices of the number of quantization levels, *N*. For each case (speech and music), what is the necessary *N* to obtain a good sound quality? Print the plots generated with several different choices.

2) Modify the above sample program, to replace uniform quantization by μ-law quantization. You should be able to enter the parameter μ in addition to the number of quantization levels *N*. Compare the results obtained with different μ and *N*. With a properly chosen μ, what is the minimum number of bits you have to use to retain sufficient quality for the speech and music files, respectively? How do they compare with the required bits by uniform quantization?

   Hint: you should apply μ-law to the original sample value, quantize the transformed value using a uniform quantizer, then apply inverse μ-law to the quantized value to obtain the quantized value in the original domain. To derive the inverse μ-law, you need to determine, from Eq. (3.14), how to determine *x(n)* from *y(n)*.

3) The Matlab programs given in Appendix "sindm.m" and "sinadm.m" implement DM and ADM, for a sinusoidal signal. Read through the given programs to understand how they work. Compare the results (the plots generated) obtained with different choices of Q, P, xmean, dmin, dmax. Comment on the effect when Q is too small or too large. Similarly, what is the effect when P is too small or too large? What parameter setting gives you the best result in each case? Print the plots generated with different choices.

4) Modify "sindm.m" to process sound signals. The program should be able to

   a) read an input .wav file,

   b) apply Delta Modulation to each sample,

   c) reconstruct the sample after quantization,

   d) plot the waveforms of the original and quantized signals, to allow to you see any changes in sample amplitudes. You should take a small portion of the waveform when plotting, so that you can see individual samples.

   e) save the reconstructed file as another .wav file,

f) playback the original and reconstructed .wav file, to allow you to compare their sound quality.

5) (Optional): Apply the program to speech signals sampled at 11 KHz, 22 KHz and 44 KHz, quantized to 8 bits originally. For each input file, apply DM using the above Matlab program. You must adjust the step-size to try to achieve the best possible quality in each case. Try to use the histogram of sample differences to determine appropriate step-size. Observe the sound quality and changes in amplitude in each case. At what sampling frequency, the DM compressed signal provides comparable quality as the original 8 bit signal sampled at 11 KHz ? What are the original data rates and rates after DM, for each sampling rate?

   Note that with the matlab program, although the prediction error is quantized to 1 bit/sample, the reconstructed signal is represented using double precision, and when converted to a .wav file, each sample takes 8 or 16 bits. So the size of the .wav file you created is not a correct indication of the actual compressed file size. A "real" compression program would save the prediction error using 1 bit/sample.

6) Repeat 4) and 5) using ADM. In this case, you must choose parameters P, xmean, dmin, dmax properly. Try to use the histogram of sample differences to help you determine these parameters. Compare the quality of ADM with DM, and with uniform quantization.

7) (Optional): Write a Matlab script to perform "ADM + μ–law" compression for a given input signal. Apply it to the music signal you used before. Do you need fewer bits to reach the same sound quality?

   Hint: you need to generate prediction error at each sample, apply μ–law to transform the error value, quantize the transformed value using the stepsize determined by the ADM algorithm, then convert the quantized value back by applying inverse μ–law, and finally add this reconstructed error value to the predicted value.

IV Report

Submit the matlab programs and plots you created. Explain any phenomena you observed, comment on the sound quality with different parameter settings, and answer the questions given in the experiment assignment. You must show your work to the instructor and get a signature on your printout of the programs and plots to prove that you finished the work at the lab.

V References

1) L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Sign*als, Prentice Hall 1978.

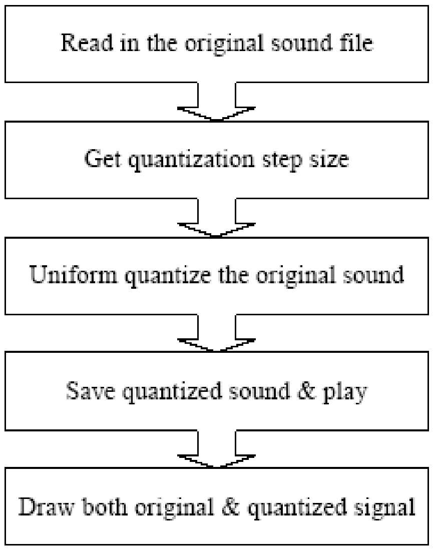
2) B. Smith, "Instantaneous Companding of Quantized Signals", *Bell System Tech. J.*, Vol. 36, No. 3, pp. 653-709, May 1957.

3) N. S. Jayant, "Adaptive Quantization With a One Word Memory", *Bell System Tech. J.*, pp. 1119-1144, September 1973.

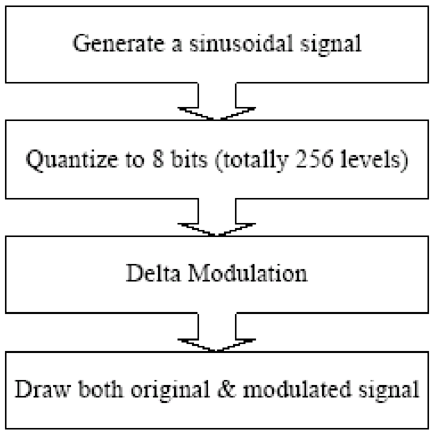
4) Guido van. Rossum (guido@cwi.nl), "FAQ: Audio File Formats", http://www.cis.ohio-state.edu .

APPENDIX Sample Matlab Programs

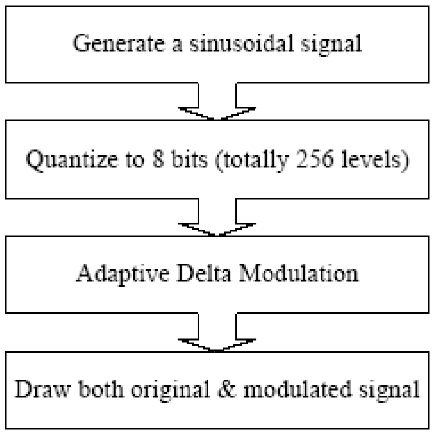
1. Flow charts for sample programs.
2. Matlab scripts for quantization, Delta Modulation (DM) and Adaptive Delta Modulation (ADM).


Read in the original sound file
Get quantization step size
Uniform quantize the original sound
Save quantized sound & play
Draw both original & quantized signal

a. Block diagram for quantization


Generate a sinusoidal signal
Quantize to 8 bits (totally 256 levels)
Delta Modulation
Draw both original & modulated signal

b. Block diagram for delta modulation


Generate a sinusoidal signal
Quantize to 8 bits (totally 256 levels)
Adaptive Delta Modulation
Draw both original & modulated signal

c. Block diagram for adaptive delta modulation

****

* MATLAB program for applying uniform quantization on a sound signal *

****

```
%quantizing a sound signal
function []=demo_quant(inname,outname, N);
if nargin < 3
   disp('Usage: sampl_quant(inname,outname, N)');
   disp('inname: input .wav file name');
   disp('outname: output .wav file name');
   disp('N: quantization level, N>1');
   return;
end;

%read in input  signal
[x,fs,N0]=wavread(inname);
xmin=min(x); xmax=max(x);
Q=(xmax-xmin)/N;
disp('N0,xmin,xmax,N,Q');
disp([N0,xmin,xmax,N,Q]);

%apply uniform quantization on each sample
xq=sign(x).*(floor((abs(x)+Q/2)/Q)*Q);

%compare sound quality
wavwrite(xq,fs,N0,outname);
sound(x,fs);
pause;
sound(xq,fs);

%plot waveforms over the entire period
t=1:length(x);
figure; plot(t,x,'r:');
hold on; plot(t,xq,'b-');
axis tight; grid on;

%plot waveform over a selected period
t=5000:5100;
figure; plot(t,x(5000:5100),'r:');
hold on; plot(t,xq(5000:5100),'b-');
axis tight; grid on;
```

****
* MATLAB program for Delta Modulation on a sinusoidal signal*
****

```
function [t,x,xx]=sindm(Q, xmean);
if nargin < 1
   disp('Usage: sindm(Q, xmean)');
   disp('Q: stepsize');
   disp('xmean: mean value of the signal');
   return;
end;


% construct a quantized sinusoid wave signal using 8-bit quantizer
% { ranged at (0,255) } for sampling time interval t=(0,1).

% given a sampling frequency.
fs=33;
t=[0:1/fs:1];
L=length(t);
f=2;
x=(sin(2*pi*f*t)+1.0)/2*255;
x=round(x); %the round operation essentially quantizes to 8 bits,
because the range
           % of x is 0-255.

% Delta Modulation
D=Q*ones(L); % fixed stepsize=30
%xmean=128;

% given the initial condition.
d(1)=x(1); %difference signal
c(1) =0; %coded signal
dd(1)=D(1); %quantized difference signal
xx(1)=xmean+dd(1); %reconstructed signal
%sindm.m
% calculate the delta modulation.
for i = 2:L,
   d(i)=x(i)-xx(i-1);
      if d(i) > 0
         c(i) = 0;
         dd(i)=D(i);
      else
         c(i) = 1;
         dd(i)=(-1)*D(i);
      end
   xx(i)=xx(i-1)+dd(i);
end

figure;
t1=[0:1/(100*fs):1];
x1=(sin(2*pi*f*t1)+1.0)/2*255;
plot(t1,x1,t,x,'*',t,xx,'x',t(2:L),xx(1:(L-1)),'o');
title('Illustration of the linear delta modulation');
legend('original signal', 'original sample', ...
      'reconstructed value','predicted value');
```

****
* MATLAB Script file for Adaptive Delta Modulation with sin as input signal *
****

```
function [t,x,xx]=sinadm(P,xmean,dmin,dmax)

if nargin < 1
   disp('Usage: sinadm(P,xmean,dmin,dmax)');
   disp('P: Adaptation Parameter, 1<=P<=3');
   disp('xmean: mean value of x');
   disp('dmin, dmax: min and max of stepsize');
   return;
end;

% given a sampling frequency.
fs=33;

% construct a quantized sinusoid wave signal using 8-bit quantizer
% { ranged at (0,255) } for sampling time interval t=(0,1).
t=[0:1/fs:1];
L=length(t);
f=2;
x=(sin(2*pi*f*t)+1.0)/2*255;
x=round(x);

% Adaptive Delta Modulation
%P=1.8;
%dmin=2;
%dmax=40;
%xmean=128;
Q=1.0/P;

% given the initial condition.
d(1)=x(1);
c(1) =0;
dd(1)=(dmin+dmax)/2;
xx(1)=xmean+dd(1);

% calculate the adaptive delta modulation.
for i = 2:L,
   d(i)=x(i)-xx(i-1);
      if d(i) > 0
         c(i) = 0;
      else
         c(i) = 1;
      end
      if c(i) == c(i-1)
         M=P;
      else
         M=Q;
      end
   dd(i)=M*dd(i-1);
   %dd(i)=round(dd(i));
      if dd(i) < dmin
         dd(i)=dmin;
      elseif dd(i) > dmax
         dd(i)=dmax;
      end
      if c(i) == 0
         xx(i)=xx(i-1)+dd(i);
      elseif c(i) == 1
         xx(i)=xx(i-1)-dd(i);
      end
end

% graph of the fixed stepsize delta modulation of a sinusoid wave
signal.
figure;
t1=[0:1/(100*fs):1];
x1=(sin(2*pi*f*t1)+1.0)/2*255;
plot(t1,x1,'-',t,x,'*',t,xx,'x',t(2:L),xx(1:(L-1)),'o');
title('Illustration of the adaptive delta modulation');
legend('original signal', 'original sample', ...
   'reconstructed value','predicted value');
```