INTRODUCTION TO SPEECH PROCESSING

The processing of speech signals is one of the most fruitful areas of application of digital signal processing techniques. Basic speech processing problems such as speech enhancement, speech synthesis, digital speech coding, and speech recognition all present interesting opportunities for application of digital signal processing algorithms. In such applications it is very important to understand the properties of the speech signal in order to make intelligent use of DSP techniques. This set of projects is intended as an introduction to speech processing with the goal of illustrating the properties of speech and the application of DSP techniques in a short-time analysis framework. Background for these projects can be found in [3] to [6].

PROJECT 1: SPEECH SEGMENTATION

Segmentation and phonetic labeling of speech waveforms is a fundamental problem in speech processing. In general, this is very difficult to do automatically, and even computer-aided segmentation by human beings requires a great deal of skill and knowledge on the part of the analyst. Nevertheless, it is very instructive to attempt to identify the parts of the speech waveform that correspond to the different phonemes of the utterance.

Hints

While MATLAB is far from an ideal tool for this purpose, plotting functions such as `plot( )`, `subplot( )`, and `stem( )` can be used for looking at short segments of speech waveforms. You will find that in some versions of MATLAB the basic plotting function `plot( )` has a vector-length limitation of 4094. For this reason the M-file `striplot( )` is available in Appendix A for plotting long vectors in a multiline format. An example of a speech waveform plotted using `striplot( )` is shown in Fig. 10.2. This plot shows the waveform of an utterance of the sentence *Oak is strong and also gives shade.* The waveform was sampled with a sampling rate of 8000 samples/s.

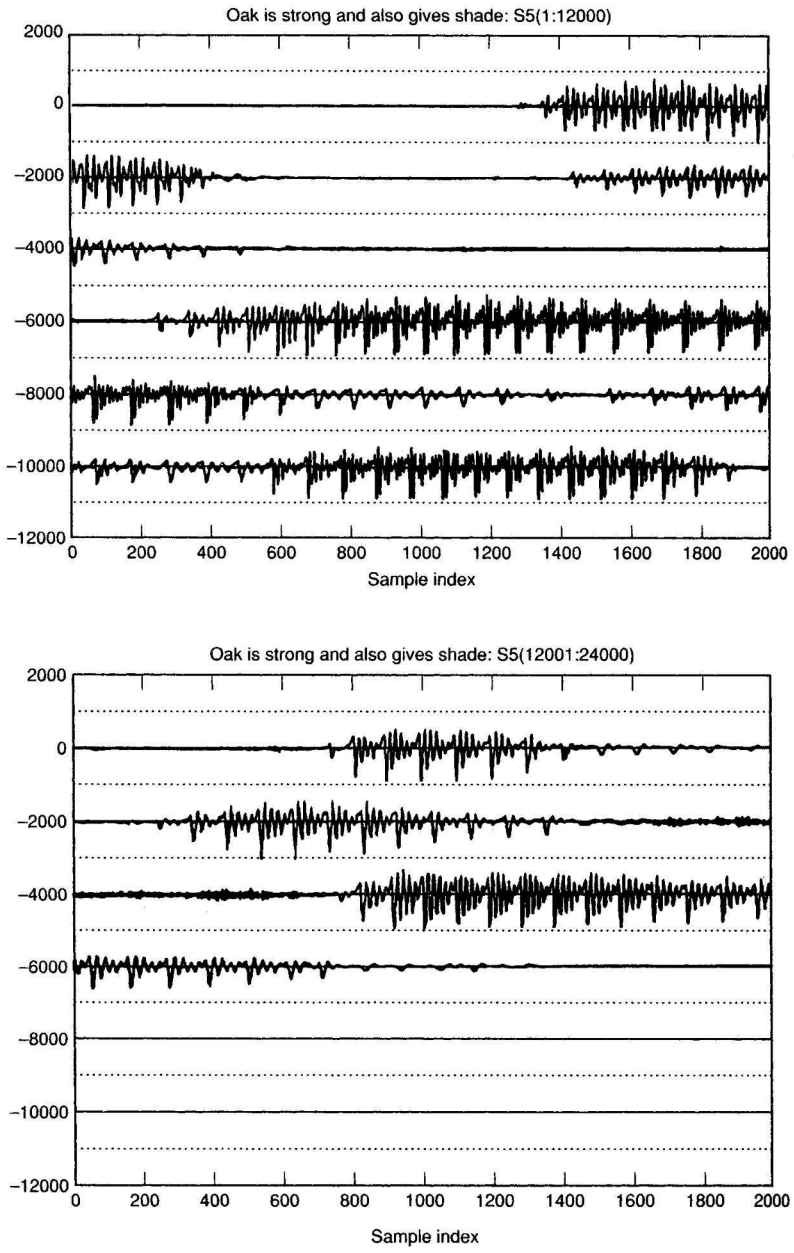

Figure 10.2
Waveform of an utterance of the sentence "Oak is strong and also gives shade."

Speech waveforms are represented in MATLAB by vectors of samples usually taken at a sampling rate of at least 8000 samples/s. This sometimes presents a problem due to the

inherent variable-length limitation of some versions of MATLAB. With MATLAB 4.0 and 5.0, there is no problem in representing speech signals of several seconds' duration. For these cases several speech waveforms are provided in the distribution package. These are named `S1.MAT-S6.MAT`. Each of these files represents a complete utterance of length 24,576 samples at a sampling rate of 8000 samples/s. These files can be loaded into MATLAB with the command `load s5`, for example. For other versions of MATLAB, these sentences have been segmented into separate files of length 1000 samples using the M-file `chopfile` from Appendix A. Thus the waveform contained in file `S5.MAT` is also contained in the files `S5_1.MAT-s5_25.MAT`. Using these files, smaller sections of the waveform can be held in MATLAB's variable space for processing. Although this is obviously a bit awkward, it is necessary when only a limited version of MATLAB is available.

EXERCISE 1.1

Phonetic Representation of Text

First write out a phonetic representation of the sentence *Oak is strong and also gives shade* using the "ARPABET" system of phonetic symbols defined in Table 10.3.

TABLE 10.3
ARPABET Phonetic Symbol System

ARPABET	*Example*	*ARPABET*	*Example*	*ARPABET*	*Example*
IY	beat	AY	buy	F	fat
IH	bit	OY	boy	TH	thing
EY	bait	Y	you	S	sat
EH	bet	W	wit	SH	shut
AE	bat	R	rent	V	vat
AA	Bob	L	let	DH	that
AH	but	M	met	Z	zoo
AO	bought	N	net	ZH	azure
OW	boat	NX	sing	CH	church
UH	book	P	pet	JH	judge
UW	boot	T	ten	WH	which
AX	about	K	kit	EL	battle
IX	roses	B	bet	EM	bottom
ER	bird	D	debt	EN	button
AXR	butter	H	get	DX	batter
AW	down	HH	hat	Q	(glottal stop)

EXERCISE 1.2

Phonetic Labeling Using Waveform Plots

Use the plots of Fig. 10.2 together with the plotting features of MATLAB to examine the waveform in the file `S5.SP`, and make your best decisions on where each phoneme of the utterance begins and ends. Be alert for phonemes that are missing or barely realized in the waveform. There may be a period of "silence" or "noise" at the beginning and end of the file. Be sure to mark the beginning and end of these intervals, too, and label the interval with the corresponding ARPABET symbol. Make a table showing the phonemes and the starting and ending samples for each.

EXERCISE 1.3

Listening (*Optional*)

If you have D/A capability available on your machine, it is instructive to listen to the signal with some of the phonemes removed. Use the basic capabilities of MATLAB to construct two vectors corresponding to the following

Utterance 1: `Oa_ i_ __o__ a__ a__o _i___ __a__`

Utterance 2: `__k _s str_ng _nd _ls_ g_ves sh_de`

That is, utterance 1 has only the sounds corresponding to the vowel letters in the sentence "Oak is strong and also gives shade," and utterance 2 has only the sounds corresponding to the consonant letters. (*Hint: If you have utterance 1 and the original utterance, how can you easily compute utterance 2?*) Write files in a format appropriate for available D to A facilities, and listen to them.

From the text representations of utterances 1 and 2, it appears that it would be easy to decode the sentence from the consonants but much more difficult to obtain it from only the vowels. After listening to the two utterances, which seems to be "most intelligible?"

PROJECT 2: PREEMPHASIS OF SPEECH

Speech signals have a spectrum that falls off at high frequencies. In some applications it is desirable that this high-frequency falloff be compensated by "preemphasis." A simple and widely used method of preemphasis is linear filtering by a "first difference" filter of the form

$$y[n] = x[n] - \alpha x[n-1] \tag{2-1}$$

where $x[n]$ is the input speech signal and $y[n]$ is the output "preemphasized speech" and α is an adjustable parameter.

EXERCISE 2.1

Preliminary Analysis

Determine analytical expressions for the impulse response, system function (z-transform of the impulse response), and the frequency response of the linear time-invariant system represented by (2-1). Use `freqz( )` to plot the frequency response of the preemphasis system for $\alpha =$ 0.5, 0.9, and 0.98. Plot all three functions together, and label the frequency axis appropriately for an 8-kHz sampling rate. How should α be chosen so that the high frequencies will be "boosted"?

EXERCISE 2.2

MATLAB Implementation

Use the MATLAB functions `filter( )` and `conv( )` to implement the preemphasis filter for $\alpha = 0.98$. What is the difference in the outputs for the two methods?

If you have not been able to read all of the waveform into MATLAB in one piece, what would you have to do at the edges of the subpieces to implement the preemphasis filter across the entire waveform?

EXERCISE 2.3

Plotting the Preemphasized Signal

Use `striplot( )` to plot the waveform of the preemphasized speech signal for $\alpha = 0.98$. If you are able to use long vectors, make plots comparable to Fig. 10.2. If your vector length

is limited, plot as much as you can for comparison to Fig. 10.2. How does the preemphasized speech waveform differ from the original? What characteristics are unchanged by preemphasis?

Optional: Write an M-file to plot one segment of the input signal followed by the corresponding segment of the output signal, followed by the next segment of the input, and so on. You should be able to do this by creating a new vector from the input and output, and then calling the `striplot( )` M-file.

EXERCISE 2.4

Listening (*Optional*)

If you have D to A capability, create a file in appropriate format containing the original speech followed by a half-second of silence followed by the preemphasized speech. Listen to this waveform and describe the qualitative difference between the two versions of the same utterance.

PROJECT 3: SHORT-TIME FOURIER ANALYSIS

The short-time Fourier transform (STFT) is defined as

$$X_n(e^{j\lambda}) = \sum_{m=-\infty}^{\infty} w[n-m]x[m]e^{-j\lambda m} \tag{3-1}$$

$$= e^{-j\lambda n}\sum_{m=-\infty}^{\infty} w[-m]x[n+m]e^{-j\lambda m} = e^{-j\lambda n}\tilde{X}_n(e^{j\lambda}) \tag{3-2}$$

where $-\infty < n < \infty$ and $0 \le \lambda < 2\pi$ (or any other interval of length 2π). The concept of the time-varying spectrum underlies many of the most useful discrete-time processing algorithms for speech signals.

We can evaluate the STFT at a discrete set of frequencies $\lambda_k = 2\pi k/N$ and at a fixed time n through the use of the DFT (and FFT). If we assume that the window is such that $w[-m] = 0$ for $m < 0$ and $m > L-1$, a simple manipulation of (3-2) gives

$$X_n[k] = X_n(e^{j(2\pi/N)k}) = \sum_{m=n}^{n+L-1} w[n-m]x[m]e^{-j(2\pi/N)km} \tag{3-3}$$

$$= e^{-j(2\pi/N)kn}\sum_{m=0}^{L-1} w[-m]x[n+m]e^{-j(2\pi/N)km} = e^{-j(2\pi/N)kn}\tilde{X}_n[k] \tag{3-4}$$

where if $\tilde{w}[m] = w[-m]$,

$$\tilde{X}_n[k] = \sum_{m=0}^{L-1} \tilde{w}[m]x[n+m]e^{-j(2\pi/N)km} \qquad k = 0, 1, \ldots, N-1 \tag{3-5}$$

Note that $X_n[k]$ and $\tilde{X}_n[k]$ differ only by the exponential phase factor $e^{-j(2\pi/N)kn}$ and therefore $|X_n[k]| = |\tilde{X}_n[k]|$. Equation (3-5) simply states that $\tilde{X}_n[k]$ can be computed by the following steps:

a. Select L samples of the signal at time n; $\{x[n], x[n+1], \ldots, x[n+L-1]\}$. (For symmetric windows, it may be convenient to assume that n is at the center of the window interval.)

b. Multiply the samples of the speech segment by the window samples forming the sequence $\{\tilde{w}[m]x[n+m]\}$, $m = 0, 1, \ldots, L-1$.

c. Compute the N-point DFT of the "windowed speech segment" (padding with zeros if $N > L$).

d. Multiply by $e^{-j(2\pi/N)kn}$ (this can be omitted if only the magnitude of the STFT is to be computed).

e. Steps (a)–(d) are repeated for each value of n.

EXERCISE 3.1

Effect of Window Length

The length of the window is a key parameter of the the STFT. If the window is short compared to features in the time waveform, the STFT will track changes in these features. If the window is relatively long, changes with time will be blurred, but the STFT will have good resolution in the frequency (k) dimension. The following is an M-file from Appendix A for demonstrating the effect of window length on the DFT of a segment of speech.

```
function        speccomp(x,ncenter,win,nfft,pltinc)
%       speccomp(x,ncenter,win,nfft,pltinc)
%               x=input signal
%               ncenter=sample number that windows are centered on
%               win=vector of windows lengths to use;
%                   should use odd lengths e.g., [401,201,101]
%               nfft=fft size
%               pltinc=offset of plots (in dB)
%
%       Plots spectra with different window lengths all centered
%       at the same place.
%
if( (ncenter - fix(max(win)/2) < 1) | (ncenter + fix(max(win)/2) > length(x)) )
        disp('Window too long for position in input segment')
        return
end
nwins=length(win);
X=zeros(nfft,nwins);
con=1;
coninc=10^(pltinc/20);
for k=1:nwins
        n1=ncenter-fix(win(k)/2);
        n2=ncenter+fix(win(k)/2);
        X(:,k)=con*fft(x(n1:n2).*hamming(win(k)),nfft);
        con=con/coninc;
end
f=(0:nfft/2)*(8000/nfft);
X=sqrt(-1)*20*log10(abs(X(1:nfft/2+1,:)))+(ones(nwins,1)*f).';
plot(X)
xlabel('frequency in Hz'),ylabel('log magnitude in dB')
title( 'Short-Time Spectra with Different Window Lengths')
```

This M-file computes the DFT of windowed segments of the input signal. All of the windowed segments are centered on the same sample of the signal. All window lengths should be odd to maintain symmetry around this point of the waveform. Figure 10.3 shows an example output from this program.

Study the MATLAB M-file above to determine how it works and what it does. Note the use of the complex data feature of `plot( )` to make it convenient to plot multiple spectra on the same graph.

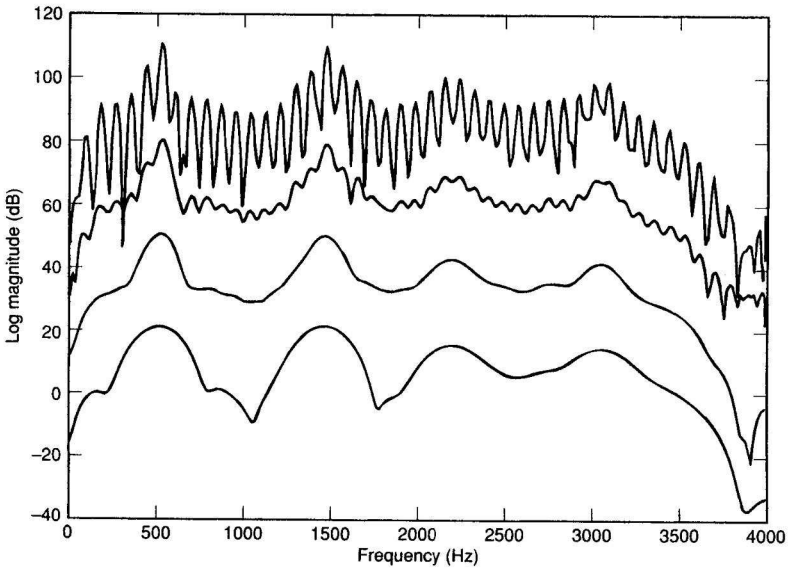

Figure 10.3
Comparison of spectral slices for windows of length 401, 201, 101, and 51 samples.

Run this program on the speech signal `S5.MAT` selecting as the center point the three cases `ncenter = 3750, 16100, 17200`. Use your results in Project 1 to determine the phonemes that occur at these three times. Use several different window lengths: for example, `[401,201,101,51]` and `nfft = 512`. What is the effect of shortening the window? Repeat for the preemphasized speech of Project 2 and compare to the results for the original speech waveform. Are the results as predicted in Exercise 2.1? Use the plots to estimate the formant frequencies of the two voiced segments of speech. Try other segments of the waveform if you have time.

EXERCISE 3.2

Effect of Window Position

Now write an M-file to compute and plot the STFT as a function of k for several equally spaced values of n. Your M-file should have the following calling sequence:

```
function        stspect(x,nstart,ninc,nwin,nfft,nsect,pltinc)
%       stspect(x,nstart,ninc,nwin,nfft,nsect,pltinc)
%               x=input signal
%               nstart=sample number that first window is centered on
%               ninc=offset between windowed segments
%               nwin=window length (should be odd)
%               nfft=fft size
%               nsect=number of sections to plot
%               pltinc=offset of spectra in plot (in dB)
%
%       Plots sequence of spectra spaced by ninc and starting with
%       window centered at nstart.
```

Your program should create a plot like that of Exercise 3.1 with frequency on the horizontal axis, but this time each spectrum corresponds to a different time rather than a different window

length. You may wish to use the M-file of Exercise 3.1 as the basis for your program. Only a few simple modifications should be necessary. Figure 10.4 shows an example of how your output should look.

Test your program for the three cases `nstart = 3750, 16100, 17200` as in Exercise 3.1. Use values of `nsect = 10, ninc = 200, nwin = 401,` and `nfft = 512`. Can you see how the formant frequencies vary with time for the voiced segments?

Also try your program on the preemphasized speech and note again the effect of the preemphasis filter.

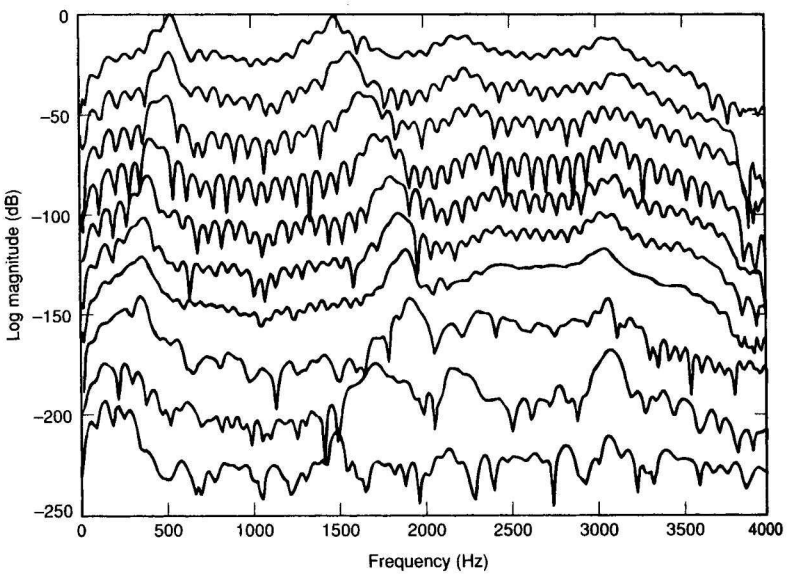

Figure 10.4
Short-time spectrum: 201-point window, 200 samples between segments.

SPEECH MODELING

The basis for most digital speech processing algorithms is a discrete-time system model for the production of samples of the speech waveform. Many useful models have been used as the basis for speech synthesis, speech coding, and speech recognition algorithms. The purpose of this set of projects is examine some of the details of the model depicted in Fig. 10.5.

PROJECT 1: GLOTTAL PULSE MODELS

The model of Fig. 10.5 is the basis for thinking about the speech waveform, and in some cases such a system is used explicitly as an speech synthesizer. In speech production, the excitation for voiced speech is a result of the quasi-periodic opening and closing of the opening between the vocal cords (the glottis). This is modeled in Fig. 10.5 by a combination of the impulse train generator and the glottal pulse model filter. The shape of the pulse affects the magnitude and phase of the spectrum of the synthetic speech output of the model. In this project we study the part labeled "Glottal Pulse Model $G(z)$" in Fig. 10.5.

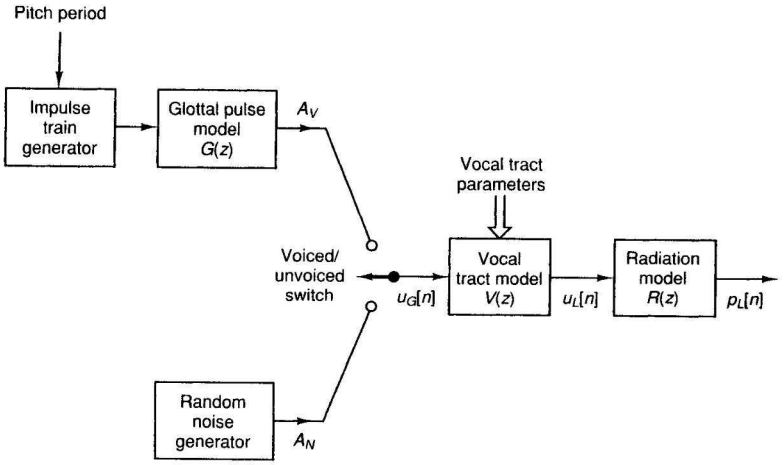

Figure 10.5
Discrete-time system model for speech production.

EXERCISE 1.1

Exponential Model

A simple model that we will call the *exponential model* is represented by

$$G(z) = \frac{-ae\ln(a)\,z^{-1}}{(1-az^{-1})^2} \tag{1-1}$$

where $e = 2.71828\ldots$ is the natural log base. Determine an analytical expression for $g[n]$, the inverse z-transform of $G(z)$. [The numerator of (1-1) is chosen so that $g[n]$ has maximum value of approximately 1.) Write an M-file to generate `Npts` samples of the corresponding glottal pulse waveform $g[n]$ and compute the frequency response of the glottal pulse model. The calling sequence for this function should be

```
[gE,GE,W]=glottalE(a,Npts,Nfreq)
```

where `gE` is the exponential glottal waveform vector of length `Npts`, and `GE` is the frequency response of the exponential glottal model at the `Nfreq` frequencies `W` between 0 and π radians. You will use this function later.

EXERCISE 1.2

Rosenberg Model

Rosenberg [9] used inverse filtering to extract the glottal waveform from speech. Based on his experimental results, he devised a model for use in speech synthesis, which is given by the equation

$$g_R[n] = \begin{cases} \frac{1}{2}[1-\cos(\pi n/N_1)] & 0 \le n \le N_1 \\ \cos[\pi(n-N_1)/(2N_2)] & N_1 \le n \le N_1+N_2 \\ 0 & \text{otherwise} \end{cases} \tag{1-2}$$

This model incorporates most of the important features of the time waveform of glottal waves estimated by inverse filtering and by high-speed motion pictures [3, 9].

Write an M-file to compute all N_1+N_2+1 samples of a Rosenberg glottal pulse with parameters N_1 and N_2 and to compute the frequency response of the Rosenberg glottal pulse model. The calling sequence for this function should be

```
[gR,GR,W]=glottalR(N1,N2,Nfreq)
```

where `gR` is the Rosenberg glottal waveform vector of length `N1+N2+1`, and `GR` is the frequency response of the glottal model at the `Nfreq` frequencies `W` between 0 and π radians.

EXERCISE 1.3

Comparison of Glottal Pulse Models

In this exercise you will compare three glottal pulse models.

a. First, use the M-files from Exercises 1.1 and 1.2 to compute `Npts=51` samples of the exponential glottal pulse `gE` for `a=0.91` and compute the Rosenberg pulse `gR` for the parameters `N1=40` and `N2=10`.

b. Also compute a new pulse `gRflip` by time-reversing `gR` using the MATLAB function `fliplr( )` for row vectors or `flipud( )` for column vectors. This has the effect of creating a new causal pulse of the form

$$g_{\text{Rflip}}[n] = g_R[-(n - N_1 - N_2)] \tag{1-3}$$

Determine the analytical relationship between $G_{\text{Rflip}}(e^{j\omega})$, the Fourier transform of $g_{\text{Rflip}}[n]$, and $G_R(e^{j\omega})$, the Fourier transform of $g_R[n]$.

c. Now plot all three of these 51-point vectors on the same graph using `plot( )`. Also plot the frequency response magnitude in dB for all three pulses on the same graph. Experiment with the parameters of the models to see how the time-domain wave shapes affect the frequency response.

d. Write an M-file to plot Rosenberg pulses for the three cases $N_2 = 10, 15, 25$ with $N_1 + N_2 = 50$ all on the same graph. Similarly, plot the Fourier transforms of these pulses together on another graph. What effect does the parameter N_2 have on the Fourier transform?

e. The exponential model has a zero at $z = 0$ and a double pole at $z = a$. For the parameters `N1=40` and `N2=10`, use the MATLAB function `roots( )` to find the zeros of the z-transform of the Rosenberg model and also the zeros of the flipped Rosenberg model. Plot them using the M-file `zplane( )`. Note that the Rosenberg model has all its zeros outside the unit circle (except one at $z = 0$). Such a system is called a *maximum-phase* system. The flipped Rosenberg model, however, should be found to have all its zeros inside the unit circle, and thus it is a *minimum-phase* system. Show that, in general, if a signal is maximum-phase, then flipping it as in (1-3) produces a minimum-phase signal, and vice versa.

PROJECT 2: LOSSLESS TUBE VOCAL TRACT MODELS

One approach to modeling sound transmission in the vocal tract is through the use of concatenated lossless acoustic tubes as depicted in Fig. 10.6.

Using the acoustic theory of speech production [3, 4, 10], it can be shown that the lossless assumption and the regular structure lead to simple wave equations and simple boundary conditions at the tube junctions, so that a solution for the transmission properties of the model is relatively straightforward and can be interpreted as in Fig. 10.7a, where $\tau = \Delta x/c$ is the one-way propagation delay of the sections. For sampled signals with

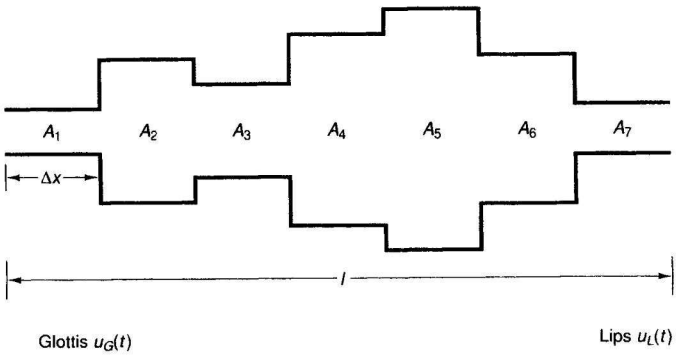

Figure 10.6
Concatenation of ($N = 7$) lossless acoustic tubes of equal length as a model of sound transmission in the vocal tract.

sampling period $T = 2\tau$, the structure of Fig. 10.7a (or equivalently Fig. 10.6) implies a corresponding discrete-time lattice filter [4] as shown in Fig. 10.7b or c.

Lossless tube models are useful for gaining insight into the acoustic theory of speech production, and they are also useful for implementing speech synthesis systems. It is shown in [4] that if $r_G = 1$, the discrete-time vocal tract model consisting of a concatenation of N lossless tubes of equal length has system function

$$V(z) = \frac{\prod_{k=1}^{N}(1 + r_k)z^{-N/2}}{D(z)} \tag{2-1}$$

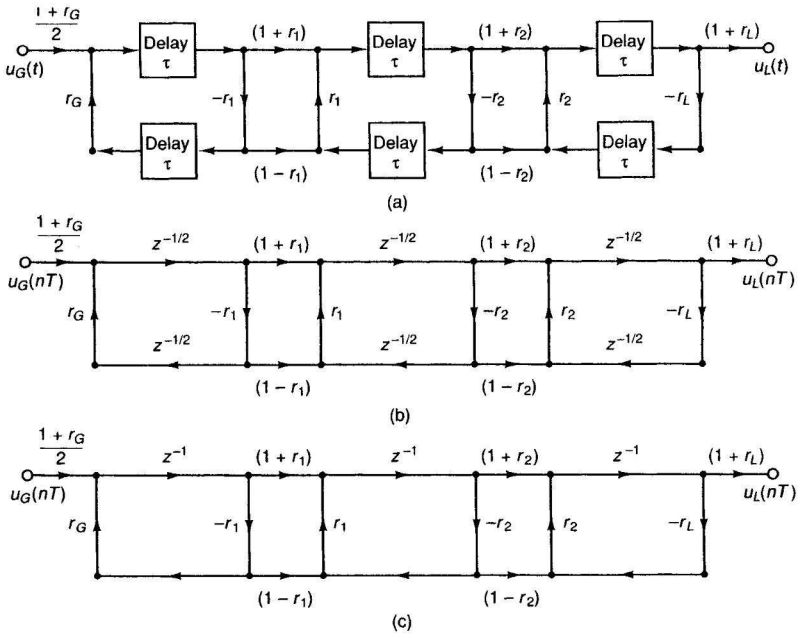

Figure 10.7
(a) Signal flow graph for lossless tube model ($N = 3$) of the vocal tract; (b) equivalent discrete-time system; (c) equivalent discrete-time system using only whole-sample delays in ladder part.

The denominator polynomial $D(z)$ in (2-1) satisfies the polynomial recursion [4]

$$\begin{aligned} D_0(z) &= 1 \\ D_k(z) &= D_{k-1}(z) + r_k z^{-k} D_{k-1}(z^{-1}) \qquad k = 1, 2, \ldots, N \\ D(z) &= D_N(z) \end{aligned} \tag{2-2}$$

where the r_k's in (2-2) are the reflection coefficients at the tube junctions,

$$r_k = \frac{A_{k+1} - A_k}{A_{k+1} + A_k} \tag{2-3}$$

In deriving the recursion in (2-2), it was assumed that there were no losses at the glottal end ($r_G = 1$) and that all the losses are introduced at the lip end through the reflection coefficient

$$r_N = r_L = \frac{A_{N+1} - A_N}{A_{N+1} + A_N} \tag{2-4}$$

where A_{N+1} is the area of an impedance-matched (no reflections at its end) tube that can be chosen to introduce a loss in the system [4].

Suppose that we have a set of areas for a lossless tube model, and we wish to obtain the system function for the system so that we can use the MATLAB `filter( )` function to implement the model; that is, we want to obtain the system function of (2-1) in the form

$$V(z) = \frac{G}{D(z)} = \frac{G}{1 - \sum_{k=1}^{N} \alpha_k z^{-k}} \tag{2-5}$$

[Note that in (2-5) we have dropped the delay of $N/2$ samples, which is inconsequential for use in synthesis.] The following MATLAB M-file called `AtoV.m` implements (2-2) and (2-3); that is, it takes an array of tube areas and a reflection coefficient at the lip end and finds the parameters of (2-5) along with the reflection coefficients.

As test data for this project, the area functions shown in Table 10.4 were obtained by interpolating and resampling area function data for Russian vowels as given by Fant [10].

TABLE 10.4
Vocal Tract Area Data for Two Russian Vowels.

Section	1	2	3	4	5	6	7	8	9	10
vowel AA	1.6	2.6	0.65	1.6	2.6	4	6.5	8	7	5
vowel IY	2.6	8	10.5	10.5	8	4	0.65	0.65	1.3	3.2

```
function        [r,D,G]=AtoV(A,rN)
%       function to find reflection coefficients
%       and system function for
%       lossless tube models.
%               [r,D,G]=AtoV(A,rN)
%               rN = reflection coefficient at lips (abs value < 1)
%               A = array of areas
%               D = array of denominator coefficients
%               G = numerator of system function
%               r = corresponding reflection coefficients
%       assumes no losses at the glottis end (rG=1).
[M,N] = size(A);
if(M~=1)  A = A';  end      %-- make row vector
```

```
N = length(A);
r = [];
for m=1:N-1
     r = [r (A(m+1)-A(m))/(A(m+1)+A(m))];
end
r = [r rN];
D = [1];
G = 1;
for m=1:N
     G = G*(1+r(m));
     D = [D 0] + r(m).*[0 fliplr(D)];
end
```

EXERCISE 2.1

Frequency Response and Pole–Zero Plot

a. Use the M-file `AtoV( )` to obtain the denominator $D(z)$ of the vocal tract system function, and make plots of the frequency response for each area function for `rN=0.71` and also for the totally lossless case $rN = 1$. Plot the two frequency responses for a given vowel on the same plot.

b. Factor the polynomials $D(z)$ and plot the poles in the z-plane using `zplane( )`. Plot the roots of the lossy case as o's and the roots of the lossless case as x's. (See `help zplane` from Appendix A.) Where do the roots lie for the lossless case? How do the roots of $D(z)$ shift as `rN` decreases away from unity? Convert the angles of the roots to analog frequencies corresponding to a sampling rate of $1/T = 10,000$ samples/s, and compare to the formant frequencies expected for these vowels [3, 4, 10]. For this sampling rate, what is the effective length of the vocal tract, in centimeters?

EXERCISE 2.2

Finding the Model from the System Function

The inverse problem arises when we want to obtain the areas and reflection coefficients for a lossless tube model given the system function in the form of (2-5). We know that the denominator of the system function, $D(z)$, satisfies (2-2). In this part we use (2-2) to develop an algorithm for finding the reflection coefficients and the areas of a lossless tube model having a given system function denominator.

a. Show that r_N is equal to the coefficient of z^{-N} in the denominator of $V(z)$ (i.e., $r_N = -\alpha_N$).

b. Use (2-2) to show that

$$D_{k-1}(z) = \frac{D_k(z) - r_k z^{-k} D_k(z^{-1})}{1 - r_k^2} \qquad k = N, N-1, \ldots, 2$$

c. How would you use the results of parts (a) and (b) to find r_{N-1} from $D_N(z) = D(z)$?

d. Using the results of parts (a), (b), and (c), state an algorithm for finding all of the reflection coefficients r_k, $k = 1, 2, \ldots, N$ and all of the tube areas A_k, $k = 1, 2, \ldots, N$. Are the A_k's unique? Write a MATLAB function to implement your algorithm for converting from $D(z)$ to reflection coefficients and areas. This M-file should adhere to the following definition:

```
function        [r,A]=VtoA(D,A1)
%       function to find reflection coefficients
%       and tube areas for lossless tube models.
%        [r,A]=VtoA(D,A1)
%               A1 = arbitrary area of first section
```

```
%              D = array of denominator coefficients
%              A = array of areas for lossless tube model
%              r = corresponding reflection coefficients
%       assumes no losses at the glottis end (rG=1).
```

[This new M-file can be similar in structure to `AtoV( )`.] For the vowel /a/, the denominator of the 10th-order model should be (to four-digit accuracy)

$$D(z) = 1 - 0.0460z^{-1} - 0.6232z^{-2} + 0.3814z^{-3} + 0.2443z^{-4} + 0.1973z^{-5} + 0.2873z^{-6} + 0.3655z^{-7} - 0.4806z^{-8} - 0.1153z^{-9} + 0.7100z^{-10}$$

Use your MATLAB program to find the corresponding reflection coefficients and tube areas and compare to the data for the vowel /a/ in Table 10.4. If your program is working, there may still be small differences between its output and the data of Table 10.4. Why?

PROJECT 3: VOWEL SYNTHESIS

For voiced speech, the speech model of Fig. 10.5 can be simplified to the system of Fig. 10.8. The excitation signal $e[n]$ is a quasi-periodic impulse train and the glottal pulse model could be either the exponential or the Rosenberg pulse. The vocal tract model could be a lattice filter of the form of Fig. 10.7c, or it could be an equivalent direct-form difference equation as implemented by MATLAB.

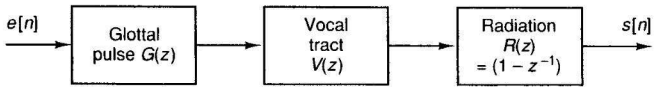

Figure 10.8 Simplified model for synthesizing voiced speech.

Hints

In this project we use the M-files written in Projects 1 and 2, together with the `filter( )` and `conv( )` functions to implement parts of the system of Fig. 10.8 and thereby synthesize periodic vowel sounds. A periodic pulse train can be synthesized by using the M-file `zerofill( )` from Appendix A, together with the MATLAB function `ones( )`.

EXERCISE 3.1

Periodic Vowel Synthesis

Assume a sampling rate of 10000 samples/s. Create a periodic impulse train vector `e` of length 1000 samples, with period corresponding to a fundamental frequency of 100 Hz. Then use combinations of `filter( )` and `conv( )` to implement the system of Fig. 10.8.

Use the excitation `e` and radiation system $R(z) = (1 - z^{-1})$ to synthesize speech for both area functions given above and for all three glottal pulses studied in Project 2. Use `subplot( )` and `plot( )` to make a plot comparing 1000 samples of the synthetic speech outputs for the exponential glottal pulse and the Rosenberg minimum-phase pulse. Make another plot comparing the outputs for the two Rosenberg pulses.

EXERCISE 3.2

Frequency Response of Vowel Synthesizer

Plot the frequency response (log magnitude in dB) of the overall system with system function $H(z) = G(z)V(z)R(z)$ for the case of the Rosenberg glottal pulse, $R(z) = (1 - z^{-1})$, and vocal tract response for the vowel /a/. Save your result for use in Exercise 3.3.

EXERCISE 3.3

Short-Time Fourier Transform of Synthetic Vowel

Compute the DFT of a Hamming-windowed segment (401 points) of the synthetic vowel and plot the log magnitude on the same graph as the frequency response of the synthesizer.

EXERCISE 3.4

Noise Excitation (Whispered Speech)

In producing whispered speech, the vocal tract is excited by turbulent airflow produced at the glottis. This can be modeled by exciting only the cascaded vocal tract and radiation filters with random noise. Using the function `randn( )`, excite the cascaded vocal tract/radiation filters for the vowel AA with a zero-mean Gaussian noise input. Plot the waveform and repeat Exercises 3.2 and 3.3 for the "whispered" vowel.

EXERCISE 3.5

Listening to the Output (Optional)

If D to A facilities are available on your computer, create files of synthetic voiced and whispered vowels of length corresponding to 0.5 s duration in the proper binary format, and play them out through the D to A system. For a 16-bit D to A converter you should scale the samples appropriately and use `round( )` to convert them to integers (of magnitude $\leq$ 32767) before writing the file. Does the synthetic speech sound like the desired vowels?

SPEECH QUANTIZATION

OVERVIEW

Sampling and quantization (or A-to-D conversion) of speech waveforms is important in digital speech processing because it is the first step in any digital speech processing system, and because one of the basic problems of speech processing is digital coding of the speech signal for digital transmission and/or storage. Sampling and quantization of signals is generally implemented by a system of the form of Fig. 10.9. In a hardware realization, the sample-and-hold circuit samples the input continuous-time signal and holds the value constant during the sampling period T. This gives a constant signal at the input of the A-to-D converter, whose purpose is to decide which of its quantization levels is closest to the input sample value. Every T seconds, the A-to-D converter emits a digital code corresponding to that level. Normally, the digital code is assigned according to a convenient binary number system such as two's-complement so that the binary numbers can be taken as numerical representations of the sample values.

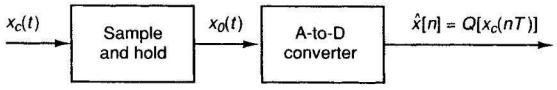

Figure 10.9
Representation of hardware for sampling and quantization of speech signals.

An equivalent representation of sampling and quantization is depicted in Fig. 10.10. This representation is convenient because it separates the sampling and quantization into two independent operations. The operation of the ideal sampler is well understood. The sampling theorem states that a bandlimited signal can be reconstructed precisely from samples taken at the rate of twice the highest frequency in the spectrum of the signal. In these projects it will be assumed that the speech signal has been low-pass filtered and

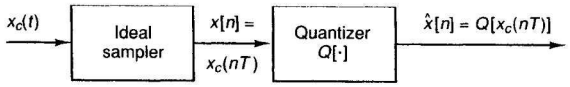

Figure 10.10 Representation of sampling and quantization that facilitates analysis and simulation.

sampled at a high enough sampling rate to avoid significant aliasing distortion. Therefore, it will be possible to focus solely on quantization of speech signal waveforms. Appropriate background reading for the projects can be found in [4] and [7].

PROJECT 1: SPEECH PROPERTIES

In this project you will use MATLAB tools to examine a particular speech waveform and verify some fundamental statistical properties of speech signals that are important for quantization.

Hints

The speech files `S1.MAT` – `S6.MAT` are available in Appendix A. The files were sampled with sampling rate 8000 samples per second and originally quantized to 12 bits. Subsequently, the samples were multiplied by 16 to raise the amplitude levels to just under 32767 (i.e., the maximum value for a 16-bit integer). Thus, these files are 12-bit samples pretending to be 16-bit samples. This will generally not be a problem in this project.

EXERCISE 1.1

Speech Waveform Plotting

First, load the file `S5.MAT` and create a vector of length 8000 samples, starting at sample 1200. *Divide the sample values by 32768 so that all samples have value less than 1.* Plot all 8000 samples with 2000 samples/line using the plotting function `striplot( )`.[2]

EXERCISE 1.2

Statistical Analysis

Compute the minimum, maximum, average, and mean-squared value of the 8000 samples from the file `S5.MAT`. Use the MATLAB function `hist( )` to plot a histogram of the 8000 samples. Experiment with the number and location of the histogram bins to obtain a useful plot. The histogram should show that the small samples are more probable than large samples in the speech waveform. Is this consistent with what you see in the waveform plot? See [4] and [7] for discussions of continuous probability density function models for the distribution of speech amplitudes.

EXERCISE 1.3

Spectral Analysis

Use the MATLAB M-file `spectrum( )` or the M-file `welch( )` from Appendix A to compute an estimate of the long-time average power spectrum of speech using the 8000 samples from file `S5.MAT`. Plot the spectrum in dB units, labeling the frequency axis appropriately. Save this spectrum estimate for use in Exercise 2.3 of Project 2.

PROJECT 2: UNIFORM QUANTIZATION

Figure 10.11 shows the input–output relation for a 3-bit uniform quantizer in which the input samples are *rounded* to the nearest quantization level and the output saturates for samples

[2] If you are using the Student Version 3.5 of MATLAB, you will be limited to variables of about 1000 samples. If such signal lengths are used, you should expect greater statistical variability in your results in some of the exercises.

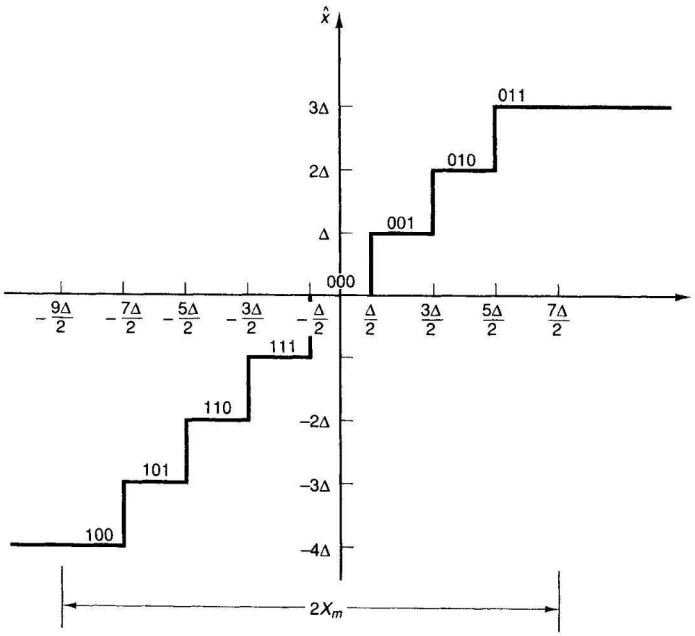

Figure 10.11
Input–output characteristic for a 3-bit uniform quantizer.

outside the range $-X_m - \Delta/2 \leq x < X_m - \Delta/2$. In discussing the effects of quantization it is useful to define the quantization error as

$$e[n] = \hat{x}[n] - x[n] \tag{2-1}$$

This definition leads to the additive noise model for quantization that is depicted in Fig. 10.12. If the signal sample $x[n]$ remains in the nonsaturating range of the quantizer, it is clear that the quantization error samples satisfy

$$-\Delta/2 \leq e[n] < \Delta/2 \tag{2-2}$$

Furthermore, speech is a complicated signal that fluctuates rapidly among the quantization levels, and if Δ is small enough, the amplitude of the signal is likely to traverse many quantization steps in one sample time. Under these conditions, it is found that the quantization error sequence is well described by the following model:

1. The error sequence $e[n]$ is uncorrelated with the unquantized sequence $x[n]$.
2. The error sequence has the properties of white noise: that is, it has a flat power spectrum, and the error samples are uncorrelated with one another.
3. The probability distribution of the error samples is uniform over the range of quantization error amplitudes.

These assumptions are tested in this project.

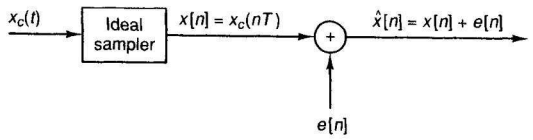

Figure 10.12
Additive noise model for sampling and quantization.

EXERCISE 2.1

Uniform Quantizer M-file

In this project you will use a uniform quantizer M-file `fxquant( )` from Appendix A to perform several quantization experiments. The comments from this M-file are as follows:

```
function X = fxquant( s, bit, rmode, lmode )
%X = fxquant( S, BIT, RMODE, LMODE )  simulated fixed-point arithmetic
%       fxquant  returns the input signal S reduced to a word-length
%       of BIT bits and limited to the range [-1,1). The type of
%       word-length reduction and limitation may be chosen with
%       RMODE:  'round'         rounding to nearest level
%               'trunc'         2's complement truncation
%               'magn'          magnitude truncation
%       LMODE:  'sat'           saturation limiter
%               'overfl'        2's complement overflow
%               'triangle'      triangle limiter
%               'none'          no limiter
```

As is clear from above, this M-file can implement a number of different quantizer functions. An important point is that the range of the quantizer is $[-1, 1)$. This is why the samples from the file `S5.MAT` were divided by 32,768.

To plot the input–output characteristics of this quantizer, type the following MATLAB statements:

```
x=-2:.001:2;
plot(x,fxquant(x,3,'round','sat'))
```

This displays the quantizer function for a 3-bit rounding quantizer with saturation. What is Δ for this quantizer, and over what range of x does the quantization error satisfy (2-2)? Now consider the statement

```
plot(x,fxquant(x,3,'round','sat')-x)
```

What is plotted in this case?

Change the parameters of the quantizer and repeat the plots to help understand the different ways that quantization can be implemented.

EXERCISE 2.2

Quantization Experiments

Use `fxquant( )` to quantize the 8000 input speech samples from the file `S5.MAT`. Using rounding and saturation, compute the quantization error sequences for 10-, 8-, and 4-bit quantization. Use the program `striplot( )` to plot these error sequences. What are the important differences among them? Do they look like they fit the white noise model? Make histograms of the quantization noise samples. Do they seem to fit the uniform amplitude distribution model?

EXERCISE 2.3

Spectral Analysis of Quantization Noise

Use `spectrum()` or `welch( )` to compute the power spectrum of the quantization noise sequences for 10, 8, and 4 bits. Plot these spectra on the same plot as the power spectrum of the speech samples. [*Remember*: The power spectrum in dB is $10\log_{10}(P)$.] Do the noise spectra support the white noise assumption? What is the approximate difference in dB between the noise spectra for 10- and 8-bit quantization? (See beginning of Project 4.)

EXERCISE 2.4

Quantization by Truncation

Set the parameter RMODE of `fxquant( )` to `'trunc'` and repeat Exercises 2.2 and 2.3. What is the main difference between the results for rounding and those for truncation?

PROJECT 3: μ-LAW COMPANDING

One of the problems with uniform quantization is that the maximum size of the quantization errors is the same no matter how big or small the samples are. For a coarse quantizer, low-level fricatives and other sounds may disappear completely because their amplitude is below the minimum stepsize. μ-Law compression/expansion is a way to obtain quantization errors that are effectively proportional to the size of the sample.

Hints

A convenient way of describing μ-law quantization is depicted in Fig. 10.13. In this representation, a μ-law compressor precedes a uniform quantizer. The combination (inside the dashed box) is a μ-law quantizer.

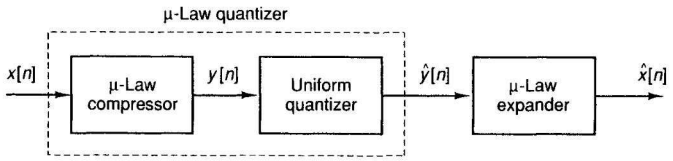

Figure 10.13
Representation of μ-law quantization.

The μ-law compressor is defined by the equation

$$y[n] = X_{\max} \frac{\log\left[1 + \mu \frac{|x[n]|}{X_{\max}}\right]}{\log(1+\mu)} \cdot \text{sign}(x[n])$$

The μ-law compressor is discussed in detail in [4, 7].

The following M-file implements the μ-law compressor on a signal vector whose maximum value is assumed to be $X_{\max} = 1$:

```
function        y=mulaw(x,mu)
%               function for mu-law compression
%       y=mulaw(x,mu)
%               x=input signal vector, column vector with max value 1
%               mu=compression parameter (mu=255 used for telephony)
sign=ones(length(x),1);
sign(find(x<0))=-sign(find(x<0));
y=(1/log(1+mu))*log(1*mu*abs(x)).*sign;
```

Note the use of the `find( )` function to locate the negative samples.

EXERCISE 3.1

μ-Law Compressor

This exercise is concerned with the μ-law compressor and its inverse.

a. Create a linearly increasing input vector `[0:0.0005:1]` and use it with the function `mulaw( )` to plot the μ-law characteristic for $\mu = 100$, 255, and 500 all on the same plot. $\mu = 255$ is a standard value used in telephony.

b. Using the segment of speech from file `S5.MAT` and a value of $\mu = 255$, plot the output waveform $y[n]$ of the μ-law compressor. Observe how the low-amplitude samples are increased in magnitude. Plot a histogram of the output samples and compare it to the histogram of the original samples.

c. To implement the system of Fig. 10.13, you must write an M-file for the inverse of the μ-law compressor. This M-file should have the following calling sequence and parameters:

```
function        x=mulawinv(y,mu)
%       function for inverse mulaw
%       x=mulawinv(y,mu)
%               y=input column vector Xmax=1
%               mu=mulaw compression parameter
%               x=expanded output vector
```

Use the technique used in `mulaw( )` to set the signs of the samples. Test the inverse system by applying it directly to the output of `mulaw()` without quantization.

EXERCISE 3.2

μ-Law Quantization

The MATLAB statement

```
yh=fxquant(mulaw(x,255),6,'round','sat');
```

implements a 6-bit μ-law quantizer. That is, it is the compressed samples that would be represented by 6 bits. When the samples are used in a signal processing computation or when a continuous-time signal is reconstructed, the samples must be expanded. Hence, the quantization errors will also be expanded, so that to determine the quantization error, it is necessary to compare the output of the inverse system to the original samples. That is, the quantization error would be `e=mulawinv(yh,255)-x;`. With this in mind, repeat all the exercises of Project 2 for the system of Fig. 10.13.

PROJECT 4: SIGNAL-TO-NOISE-RATIOS

A convenient way of comparing quantizers is to compute the ratio of signal power to quantization noise power. For experiments in MATLAB, a convenient definition of SNR is

$$\text{SNR} = 10 \log \left(\frac{\sum_{n=0}^{L-1} (x[n])^2}{\sum_{n=0}^{L-1} (\hat{x}[n] - x[n])^2} \right) \tag{4-1}$$

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

Hints

Under the assumptions of the noise model given in Project 2, it can be shown that the signal-to-noise ratio for a uniform quantizer with 2^{B+1} levels (B bits plus sign) has the form [4, 7]

$$\text{SNR} = 6B + 10.8 - 20 \log_{10} \left(\frac{X_m}{\sigma_x} \right) \tag{4-2}$$

where X_m is the clipping level of the quantizer (in our case $X_m = 1$) and σ_x is the rms value of the input signal amplitude. Thus (4-2) shows that the signal-to-noise ratio increases 6 dB per bit added to the quantizer word length. Furthermore, (4-2) shows that if the signal level is decreased by a factor of 2, the signal-to-noise ratio decreases by 6 dB.

EXERCISE 4.1

Signal-to-Noise Computation

Write an M-File to compute the signal-to-noise ratio as defined in (4-1). Its calling sequence and parameters should be

```
function        [s_n_r,e]=snr(xh,x);
%       function for computing signal-to-noise ratio
%       [s_n_r,e]=snr(xh,x)
%               xh=quantized signal
%               x=unquantized signal
%               e=quantization error signal (optional)
%               s_n_r=snr in dB
```

Use your SNR function to compute the SNRs for uniform quantization with 8 and 9 bits. Do the results differ by the expected amount?

EXERCISE 4.2

Comparison of Uniform and μ-Law Quantization

An important consideration in quantizing speech is that signal levels can vary with speakers and with transmission/recording conditions. This can result in significant variations of signal-to-noise ratio for a fixed quantizer. The following M-file from Appendix A compares uniform and μ-law quantization for a fixed quantizer with inputs of decreasing amplitude (by factors of 2). Using the M-files that were written in Projects 2 and 3 and the M-file `qplot( )`, make a plot for 10 bits with $\mu = 255$ over a range of 10 factors of 2. Explain the shape of the two curves. The program `qplot( )` plots the signal-to-noise ratios of a uniform and a μ-law quantizer for the same number of bits. Modify the program so that quantizers with several different numbers of bits can be compared on the same plot. Use the modified M-file to create a plot for 10, 8, 6, and 4 bits with $\mu = 255$ over a range of 10 factors of 2.

```
function        qplot(s,nbits,mu,ncases)
%       function for plotting dependence of signal-to-noise
%       ratio on decreasing signal level
%       qplot(s,nbits,mu,ncases)
%               s=input test signal
%               nbits=number of bits in quantizer
%               mu=mu-law compression parameter
%               ncases=number of cases to plot
%
P=zeros(ncases,2);
x=s;
for i=1:ncases
sh=fxquant(x,nbits,'round','sat');
P(i,1)=(i-1)+sqrt(-1)*snr(sh,x);
y=mulaw(x,mu);
yh=fxquant(y,nbits,'round','sat');
```

```
xh=mulawinv(yh,mu);,2)=(i-1)+sqrt(-1)*snr(xh,x);
x=x/2;
end
plot(P)
title(['SNR for ',num2str(nbits),'-bit Uniform and ',num2str(mu),...
'-Law Quantizers'])
xlabel('power of 2 divisor');ylabel('SNR in dB')
```

Note how the complex plotting feature of `plot( )` *is used as a convenience in plotting multiple graphs on the same axes.*

Your plots should show that the μ-law quantizer maintains a constant signal-to-noise ratio over an input amplitude range of about 64:1. How many bits are required for a uniform quantizer to maintain at least the same signal-to-noise ratio as that of a 6-bit μ-law quantizer over the same range?

PROJECT 5: LISTENING TO QUANTIZED SPEECH (*optional*)

If your computer has D-to-A capability, it is instructive to listen to the quantized speech. Use MATLAB to create a binary file for your D-to-A system in the form

```
(quantized speech) (0.5 s silence)
(original speech)  (0.5 s silence)
(quantized speech)
```

Remember that the quantizer M-file `fxquant( )` *requires a maximum value of 1. You should multiply the samples by the appropriate constant (probably 32,768) and convert to integer before writing the file.* Listen to this file. Can you hear the quantization noise?

Another interesting experiment is to listen to the quantization noise. Form a file in the following format:

```
(quantized speech) (0.5 s silence)
(original speech) (0.5 s silence)
(quantization noise)
```

In this case the quantization noise should be scaled up more than the speech signal itself in order to hear the noise at the same level as the speech. Does the quantization noise sound like "white noise"? Does the noise have any of the characteristics of the speech signal?