3.2 THE BANK-OF-FILTERS FRONT-END PROCESSOR

A block diagram of the canonic structure of a complete filter-bank front-end analyzer is given in Figure 3.4. The sampled speech signal, $s(n)$, is passed through a bank of Q bandpass filters, giving the signals

$$s_i(n) = s(n) * h_i(n), \qquad 1 \le i \le Q \tag{3.1a}$$

$$= \sum_{m=0}^{M_i-1} h_i(m)s(n-m), \tag{3.1b}$$

where we have assumed that the impulse response of the i^{th} bandpass filter is $h_i(m)$ with a duration of M_i samples; hence, we use the convolution representation of the filtering operation to give an explicit expression for $s_i(n)$, the bandpass-filtered speech signal. Since the purpose of the filter-bank analyzer is to give a measurement of the energy of the speech signal in a given frequency band, each of the bandpass signals, $s_i(n)$, is passed through a nonlinearity, such as a full-wave or half-wave rectifier. The nonlinearity shifts the bandpass signal spectrum to the low-frequency band as well as creates high-frequency images. A lowpass filter is used to eliminate the high-frequency images, giving a set of signals, $u_i(n)$, $1 \le i \le Q$, which represent an estimate of the speech signal energy in each of the Q frequency bands.

To more fully understand the effects of the nonlinearity and the lowpass filter, let us assume that the output of the i^{th} bandpass filter is a pure sinusoid at frequency ω_i, i.e.

$$s_i(n) = \alpha_i \sin(\omega_i n). \tag{3.2}$$

This assumption is valid for speech in the case of steady state voiced sounds when the bandwidth of the filter is sufficiently narrow so that only a single speech harmonic is passed by the bandpass filter. If we use a full-wave rectifier as the nonlinearity, that is,

$$\begin{aligned} f(s_i(n)) &= s_i(n) \quad \text{for } s_i(n) \ge 0 \\ &= -s_i(n) \quad \text{for } s_i(n) < 0. \end{aligned} \tag{3.3}$$

Then we can represent the nonlinearity output as

$$v_i(n) = f(s_i(n)) = s_i(n) \cdot w(n), \tag{3.4}$$

where

$$w(n) = \begin{cases} +1 & \text{if } s_i(n) \ge 0 \\ -1 & \text{if } s_i(n) < 0 \end{cases} \tag{3.5}$$

as illustrated in Figure 3.5(a)–(c). Since the nonlinearity output can be viewed as a modulation in time, as shown in Eq. (3.4), then in the frequency domain we get the result that

$$V_i(e^{j\omega}) = S_i(e^{j\omega}) \circledast W(e^{j\omega}), \tag{3.6}$$

where $V_i(e^{j\omega})$, $S_i(e^{j\omega})$, and $W(e^{j\omega})$ are the Fourier transforms of the signals $v_i(n)$, $s_i(n)$ and $w(n)$, respectively, and $\circledast$ is circular convolution. The spectrum $S_i(e^{j\omega})$ is a single impulse at $\omega_0 = \omega_i$, whereas the spectrum $W(e^{j\omega})$ is a set of impulses at the odd-harmonic

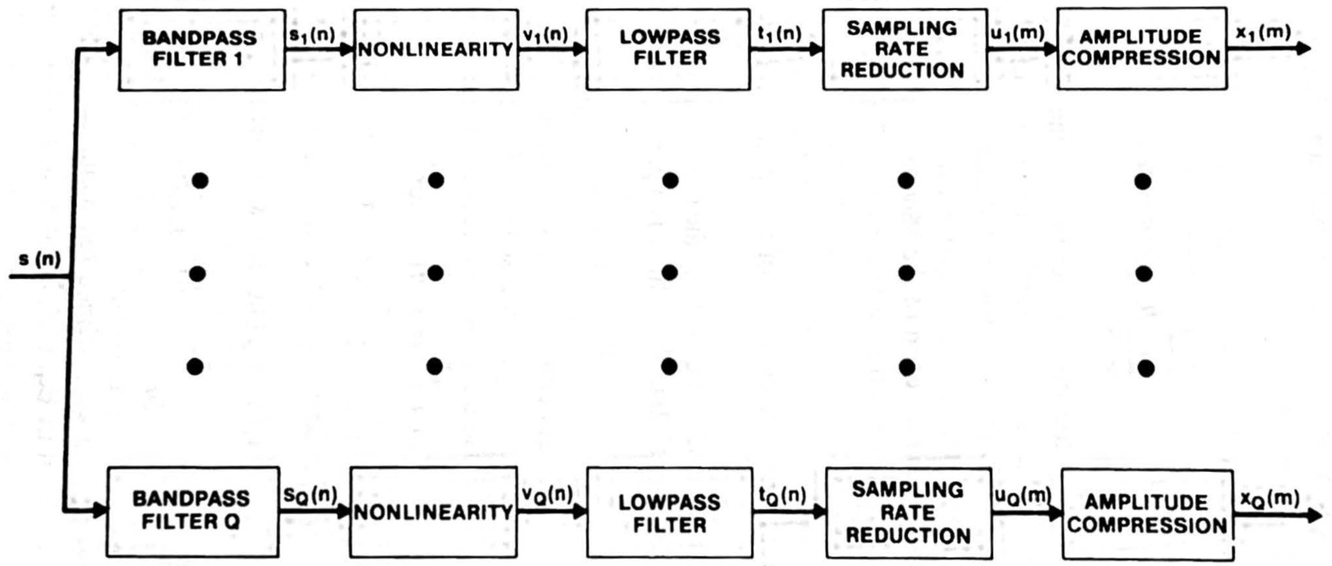

Figure 3.4 Complete bank-of-filters analysis model.

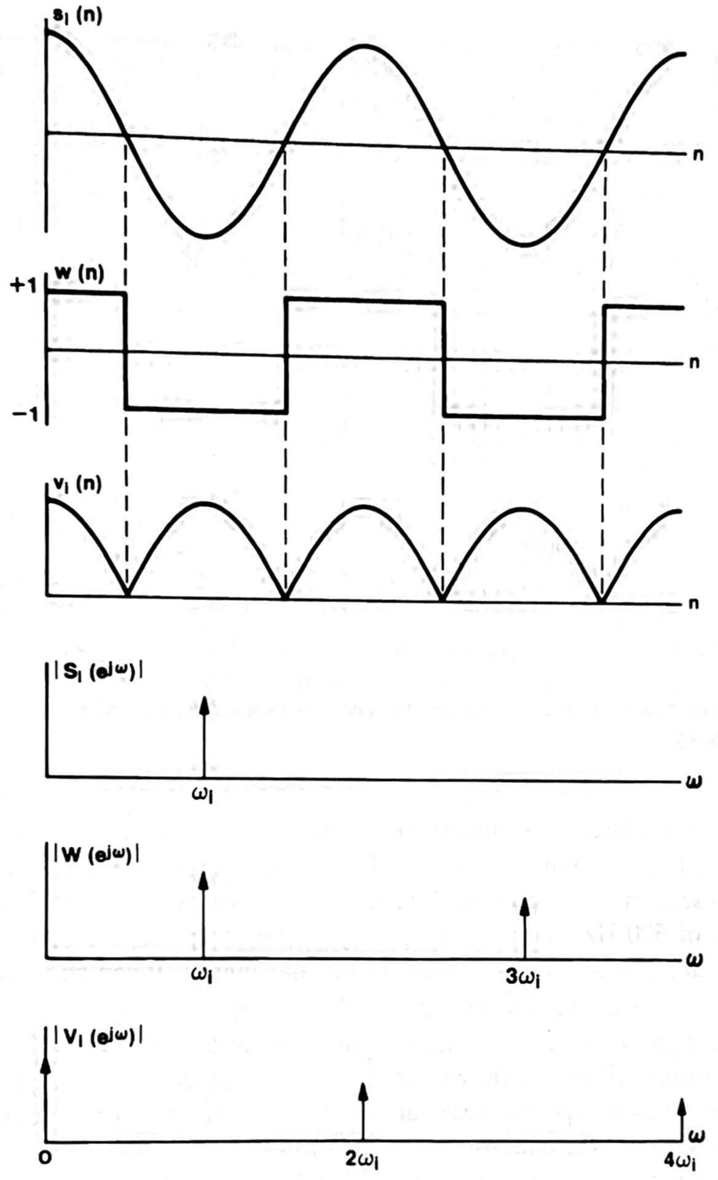

Figure 3.5 Typical waveforms and spectra for analysis of a pure sinusoid in the filter-bank model.

frequencies $\omega_q = \omega_i q$, $q = 1, 3, \ldots, q_{\max}$. Hence the spectrum of $V_i(e^{j\omega})$ is an impulse at $\omega = 0$ and a set of smaller amplitude impulses at $\omega_q = \omega_i q$, $q = 2, 4, 6, \ldots$, as shown in Figure 3.5 (d)–(f). The effect of the lowpass filter is to retain the DC component of $V_i(e^{j\omega})$ and to filter out the higher frequency components due to the nonlinearity.

The above analysis, although only strictly correct for a pure sinusoid, is a reasonably good model for voiced, quasiperiodic speech sounds so long as the bandpass filter is not so wide that it has two or more strong signal harmonics. Because of the time-varying nature of the speech signal (i.e., the quasiperiodicity), the spectrum of the lowpass signal is not a pure

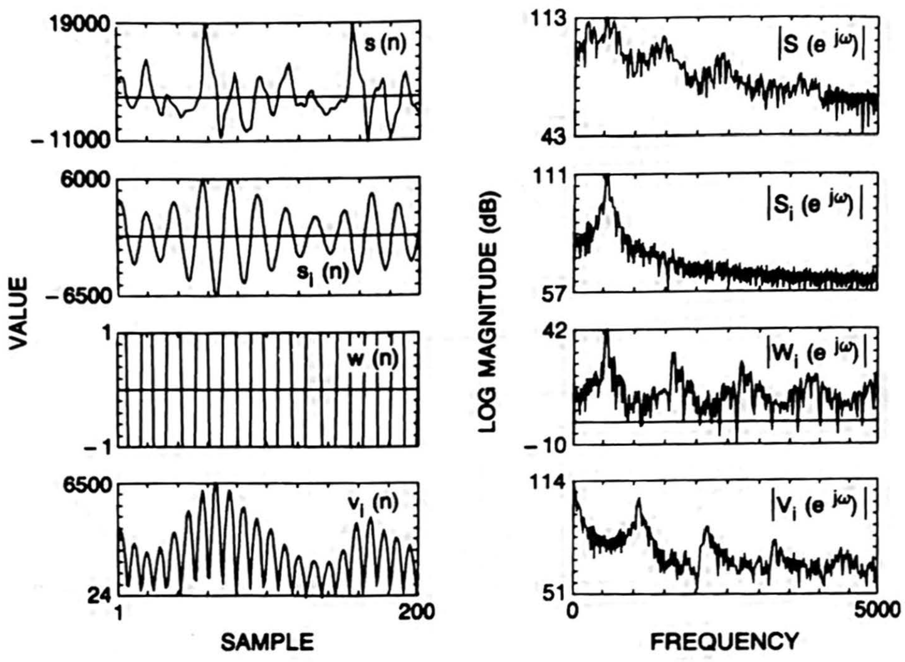

Figure 3.6 Typical waveforms and spectra of a voice speech signal in the bank-of-filters analysis model.

DC impulse, but instead the information in the signal is contained in a low-frequency band around DC. Figure 3.6 illustrates typical waveforms of $s(n)$, $s_i(n)$, $w(n)$ and $v_i(n)$ for a brief (20 msec) section of voiced speech processed by a narrow bandwidth channel with center frequency of 500 Hz (sampling frequency for this example is 10,000 Hz). Also shown are the resulting spectral magnitudes for the four signals. It can be seen that $|S_i(e^{j\omega})|$ has most of its energy around 500 Hz ($\omega = 1000\pi$), whereas $|W_i(e^{j\omega})|$ (which is quasiperiodic) approximates an odd harmonic signal with peaks at 500, 1500, 2500 Hz. The resulting signal spectrum, $|V_i(e^{j\omega})|$, shows the desired low-frequency concentration of energy as well as the undesired spectral peaks at 1000 Hz, 2000 Hz, etc. The role of the final lowpass filter is to eliminate the undesired spectral peaks.

The bandwidth of the signal, $v_i(n)$, is related to the fastest rate of motion of speech harmonics in a narrow band and is generally acknowledged to be on the order of 20–30 Hz. Hence, the final two blocks of the canonic bank-of-filters model of Figure 3.4 are a sampling rate reduction box in which the lowpass-filtered signals, $t_i(n)$, are resampled at a rate on the order of 40–60 Hz (for economy of representation), and the signal dynamic range is compressed using an amplitude compression scheme (e.g., logarithmic encoding, μ-law encoding).

Consider the design of a $Q = 16$ channel filter bank for a wideband speech signal where the highest frequency of interest is 8 kHz. Assume we use a sampling rate of $F_s = 20$ kHz on the speech data to minimize the effects of aliasing in the analog-to-digital conversion. The information (bit rate) rate of the raw speech signal is on the order of 240 kbits per second (20 k samples per second times 12 bits per sample). At the output of

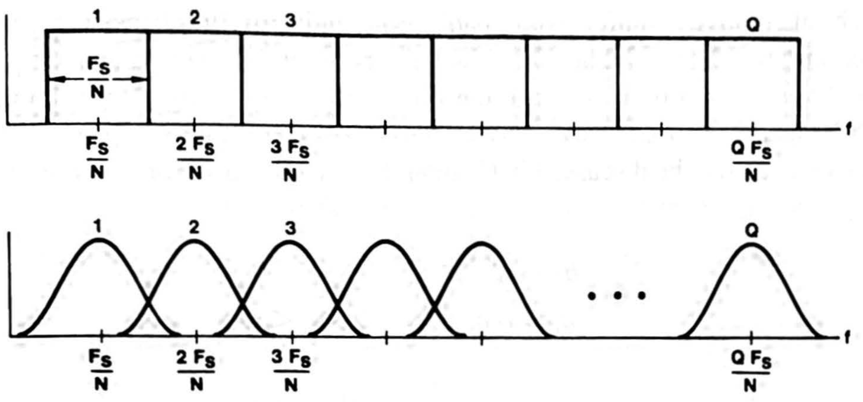

Figure 3.7 Ideal (a) and realistic (b) set of filter responses of a Q-channel filter bank covering the frequency range F_s/N to $(Q + {}^1\!/_2)F_s/N$.

the analyzer, if we use a sampling rate of 50 Hz and we use a 7 bit logarithmic amplitude compressor, we get an information rate of 16 channels times 50 samples per second per channel times 7 bits per sample, or 5600 bits per second. Thus, for this simple example we have achieved about a 40-to-1 reduction in bit rate, and hopefully such a data reduction would result in an improved representation of the significant information in the speech signal.

3.2.1 Types of Filter Bank Used for Speech Recognition

The most common type of filter bank used for speech recognition is the uniform filter bank for which the center frequency, f_i, of the i^{th} bandpass filter is defined as

$$f_i = \frac{F_s}{N} i, \qquad 1 \le i \le Q, \tag{3.7}$$

where F_s is the sampling rate of the speech signal, and N is the number of uniformly spaced filters required to span the frequency range of the speech. The actual number of filters used in the filter bank, Q, satisfies the relation

$$Q \le N/2 \tag{3.8}$$

with equality when the entire frequency range of the speech signal is used in the analysis. The bandwidth, b_i, of the i^{th} filter, generally satisfies the property

$$b_i \ge \frac{F_s}{N} \tag{3.9}$$

with equality meaning that there is no frequency overlap between adjacent filter channels, and with inequality meaning that adjacent filter channels overlap. (If $b_i < F_s/N$, then certain portions of the speech spectrum would be missing from the analysis and the resulting speech spectrum would not be considered very meaningful.) Figure 3.7a shows a set of Q ideal, non-overlapping, bandpass filters covering the range from $F_s/N({}^1\!/_2)$ to $(F_s/N)(Q + {}^1\!/_2)$. Similarly Figure 3.7b shows a more realistic set of Q overlapping filters covering approximately the same range.

The alternative to uniform filter banks is nonuniform filter banks designed according to some criterion for how the individual filters should be spaced in frequency. One commonly used criterion is to space the filters uniformly along a logarithmic frequency scale. (A logarithmic frequency scale is often justified from a human auditory perception point of view, as will be discussed in Chapter 4.) Thus for a set of Q bandpass filters with center frequencies, f_i, and bandwidths, b_i, $1 \le i \le Q$, we set

$$b_1 = C \tag{3.10a}$$

$$b_i = \alpha b_{i-1}, \qquad 2 \le i \le Q \tag{3.10b}$$

$$f_i = f_1 + \sum_{j=1}^{i-1} b_j + \frac{(b_i - b_1)}{2}, \tag{3.11}$$

where C and f_1 are the arbitrary bandwidth and center frequency of the first filter, and α is the logarithmic growth factor.

The most commonly used values of α are $\alpha = 2$, which gives an octave band spacing of adjacent filters, and $\alpha = 4/3$ which gives a $1/3$ octave filter spacing. Consider the design of a four band, octave-spaced, non-overlapping filter bank covering the frequency band from 200 to 3200 Hz (with a sampling rate of 6.67 kHz). Figure 3.8a shows the ideal filters for this filter bank. Values for f_1 and C of 300 Hz and 200 Hz are used, giving the following filter specifications:

Filter 1:	$f_1 = 300$ Hz,	$b_1 = 200$ Hz
Filter 2:	$f_2 = 600$ Hz,	$b_2 = 400$ Hz
Filter 3:	$f_3 = 1200$ Hz,	$b_3 = 800$ Hz
Filter 4:	$f_4 = 2400$ Hz,	$b_4 = 1600$ Hz

An example of a 12-band, 1/3-octave, ideal filter-bank specifications, covering the band from about 200 to 3200 Hz, is given in Figure 3.8b. For this example, $C = 50$ Hz, and $f_1 \simeq 225$ Hz.

An alternative criterion for designing a nonuniform filter bank is to use the critical band scale directly. The spacing of filters along the critical band is based on perceptual studies and is intended to choose bands that give equal contribution to speech articulation. The general shape of the critical band scale is given in Figure 3.9. The scale is close to linear for frequencies below about 1000 Hz (i.e., the bandwidth is essentially constant as a function f), and is close to logarithmic for frequencies above 1000 Hz (i.e., the bandwidth is essentially exponential as a function of f). Several variants on the critical band scale have been used, including the mel scale and the bark scale. The differences between these variants are small and are, for the most part, insignificant with regard to design of filter banks for speech-recognition purposes. For example, Figure 3.8c shows a 7-band critical-band filter-bank specification.

Other criteria for designing nonuniform filter banks have been proposed in the literature. For the most part, the uniform and nonuniform designs based on critical band scales have been the most widely used and studied filter-bank methods.

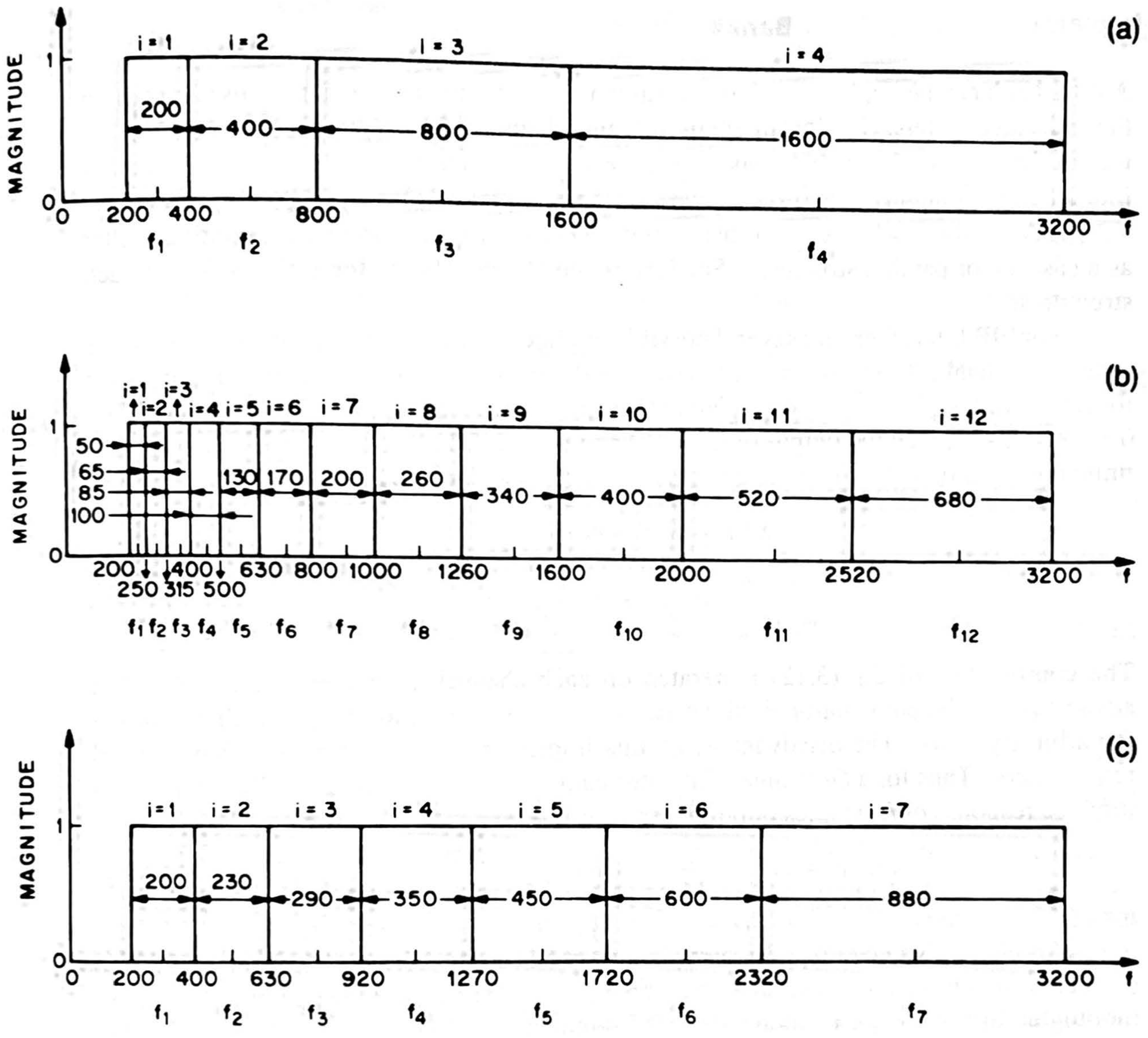
(a)
MAGNITUDE
i = 1
i = 2
i = 3
i = 4
200
400
800
1600
0 200 400 800 1600 3200 f
f1 f2 f3 f4
(b)
MAGNITUDE
i=1 i=2 i=3 i=4 i=5 i=6 i=7 i=8 i=9 i=10 i=11 i=12
50
65
85
100
130 170 200 260 340 400 520 680
200 250 315 400 500 630 800 1000 1260 1600 2000 2520 3200 f
f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12
(c)
MAGNITUDE
i = 1 i = 2 i = 3 i = 4 i = 5 i = 6 i = 7
200 230 290 350 450 600 880
0 200 400 630 920 1270 1720 2320 3200 f
f1 f2 f3 f4 f5 f6 f7

Figure 3.8 Ideal specifications of a 4-channel octave band-filter bank (a), a 12-channel third-octave band filter bank (b), and a 7-channel critical band scale filter bank (c) covering the telephone bandwidth range (200–3200 Hz).

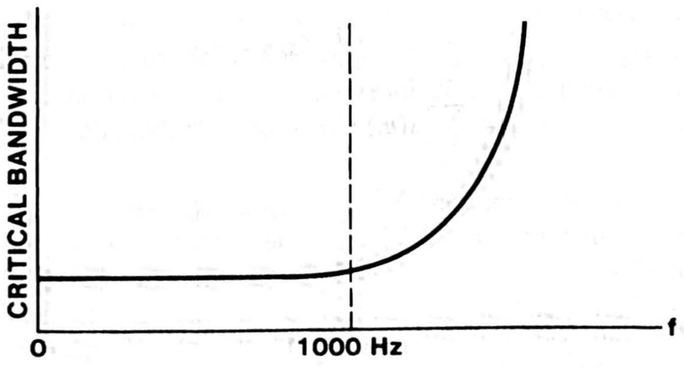

Figure 3.9 The variation of bandwidth with frequency for the perceptually based critical band scale.

3.2.2 Implementations of Filter Banks

A filter bank can be implemented in several ways, depending on the method used to design the individual filters. Design methods for digital filters fall into two broad classes: (1) infinite impulse response (IIR) and (2) finite impulse response (FIR) methods. For IIR filters (also commonly called recursive filters in the literature), the most straightforward, and generally the most efficient implementation is to realize each individual bandpass filter as a cascade or parallel structure. (See Reference [1], pp. 40–46, for a discussion of such structures.)

For FIR filters there are several possible methods of implementing the bandpass filters in the filter bank. The most straightforward and the simplest implementation is the direct form structure. In this case, if we denote the impulse response for the i^{th} channel as $h_i(n)$, $0 \le n \le L-1$, then the output of the i^{th} channel, $x_i(n)$, can be expressed as the discrete, finite convolution of the input signal, $s(n)$, with the impulse response, $h_i(n)$, i.e.

$$x_i(n) = s(n) * h_i(n) \tag{3.12a}$$

$$= \sum_{m=0}^{L-1} h_i(m)s(n-m). \tag{3.12b}$$

The computation of Eq. (3.12) is iterated on each channel i, for $i = 1, 2, \ldots, Q$. The advantages of the convolutional, direct form structure are its simplicity and that it works for arbitrary $h_i(n)$. The disadvantage of this implementation is the high computational requirement. Thus for a Q-channel FIR filter bank, where each bandpass FIR filter has an impulse response of L samples duration, we require

$$C_{\text{DFFIR}} = LQ \quad \cdot, + \qquad \text{(multiplication, addition)} \tag{3.13}$$

for a complete evaluation of $x_i(n)$, $i = 1, 2, \ldots, Q$, at a single value of n.

An alternative, less-expensive implementation can be derived for the case in which each bandpass filter impulse response can be represented as a fixed lowpass window, $w(n)$, modulated by the complex exponential, $e^{j\omega_i n}$—that is,

$$h_i(n) = w(n)e^{j\omega_i n}. \tag{3.14}$$

In this case Eq. (2.12b) becomes

$$x_i(n) = \sum_m w(m)e^{j\omega_i m}s(n-m)$$

$$= \sum_m s(m)\, w(n-m)e^{j\omega_i(n-m)}$$

$$= e^{j\omega_i n}\sum_m s(m)w(n-m)e^{-j\omega_i m} \tag{3.15a}$$

$$= e^{j\omega_i n}S_n(e^{j\omega_i}), \tag{3.15b}$$

where $S_n(e^{j\omega_i})$ is the short-time Fourier transform of $s(n)$ at frequency $\omega_i = 2\pi f_i$. The importance of Eq. (3.15) is that efficient procedures often exist for evaluating the short-

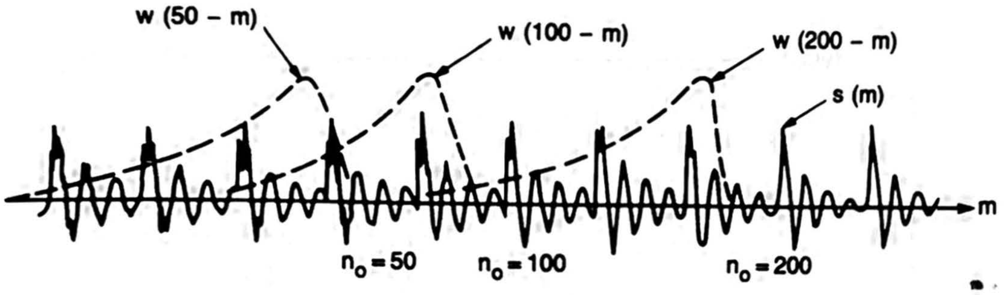

Figure 3.10 The signals $s(m)$ and $w(n - m)$ used in evaluation of the short-time Fourier transform.

time Fourier transform using FFT methods. We will discuss such procedures shortly; first, however, we briefly review the interpretations of the short-time Fourier transform (see Ref. [2] for a more complete discussion of this fascinating branch of signal processing).

3.2.2.1 Frequency Domain Interpretation of the Short-Time Fourier Transform

The short-time Fourier transform of the sequence $s(m)$ is defined as

$$S_n(e^{j\omega_i}) = \sum_m s(m)w(n-m)e^{-j\omega_i m}. \tag{3.16}$$

If we take the point of view that we are evaluating $S_n(e^{j\omega_i})$ for a fixed $n = n_0$, then we can interpret Eq. (3.16) as

$$S_{n_0}(e^{j\omega_i}) = \text{FT}[s(m)w(n_0 - m)]\big|_{\omega=\omega_i} \tag{3.17}$$

where FT[·] denotes the Fourier Transform. Thus $S_{n_0}(e^{j\omega_i})$ is the conventional Fourier transform of the windowed signal, $s(m)\ w(n_0 - m)$, evaluated at the frequency $\omega = \omega_i$. Figure 3.10 illustrates the signals $s(m)$ and $w(n - m)$, at times $n = n_0 = 50, 100$, and 200 to show which parts of $s(m)$ are used in the computation of the short-time Fourier transform. Since $w(m)$ is an FIR filter (i.e., of finite size), if we denote that size by L, then using the conventional Fourier transform interpretation of $S_n(e^{j\omega_i})$, we can state the following:

1. If L is large, relative to the signal periodicity (pitch), then $S_n(e^{j\omega_i})$ gives good frequency resolution. That is, we can resolve individual pitch harmonics but only roughly see the overall spectral envelope of the section of speech within the window.
2. If L is small relative to the signal periodicity, then $S_n(e^{j\omega_i})$ gives poor frequency resolution (i.e., no pitch harmonics are resolved), but a good estimate of the gross spectral shape is obtained.

To illustrate these points, Figures 3.11–3.14 show examples of windowed signals, $s(m)w(n - m)$, (part a of each figure) and the resulting log magnitude short time spectra, $20 \log_{10} |S_n(e^{j\omega})|$ (part b of each figure). Figure 3.11 shows results for an $L = 500$-point Hamming window applied to a section of voiced speech. The periodicity of the signal is clearly seen in the windowed time waveform, as well as in the short-time spectrum in which the fundamental frequency and its harmonics show up as narrow peaks at equally spaced

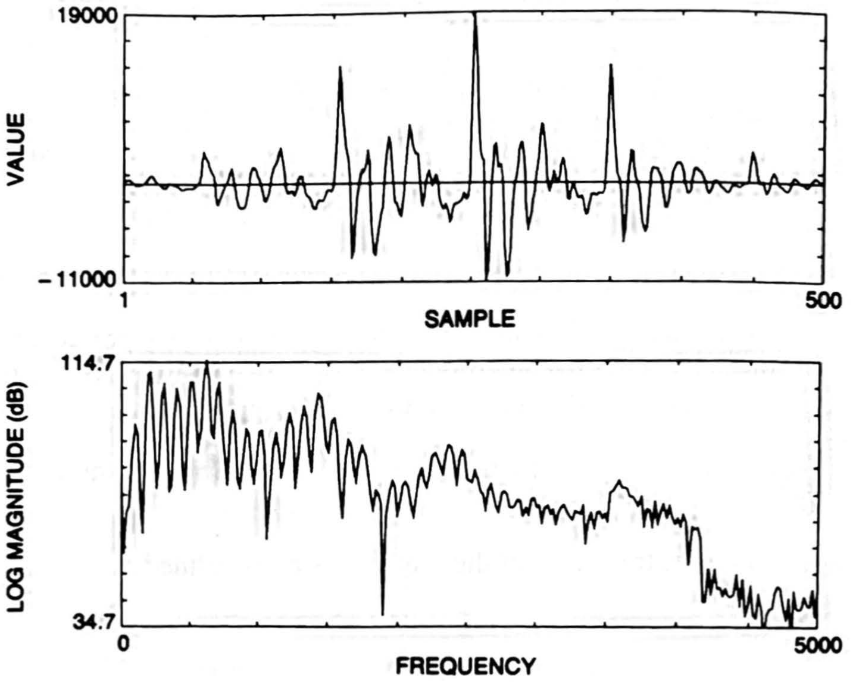

Figure 3.11 Short-time Fourier transform using a long (500 points or 50 msec) Hamming window on a section of voiced speech.

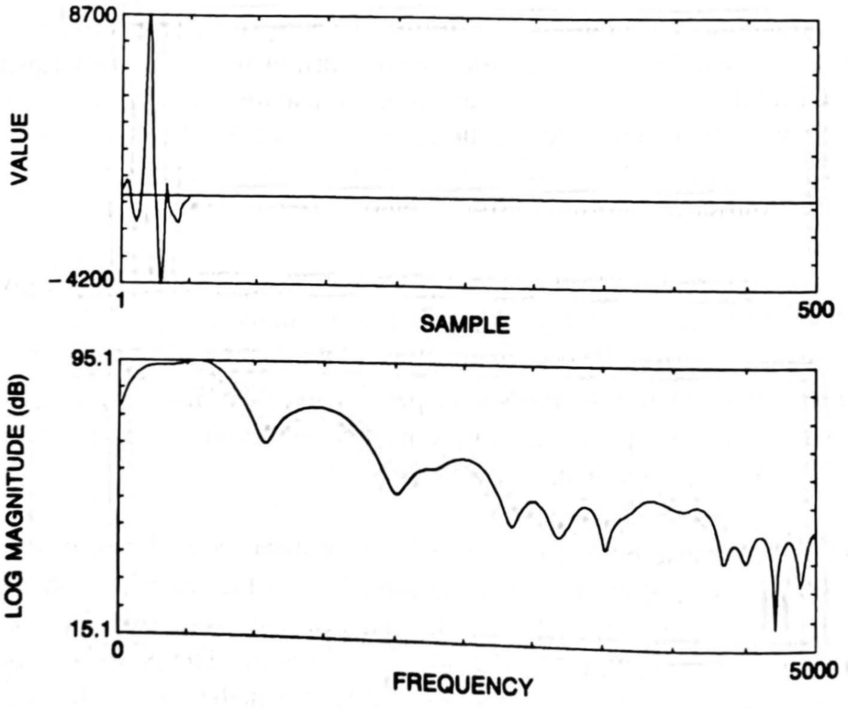

Figure 3.12 Short-time Fourier transform using a short (50 points or 5 msec) Hamming window on a section of voiced speech.

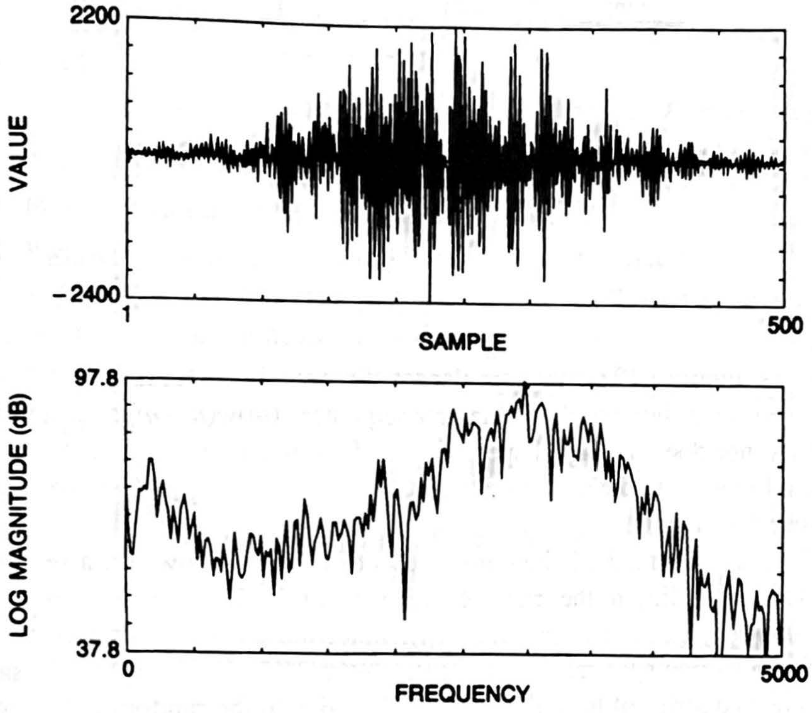

Figure 3.13 Short-time Fourier transform using a long (500 points or 50 msec) Hamming window on a section of unvoiced speech.

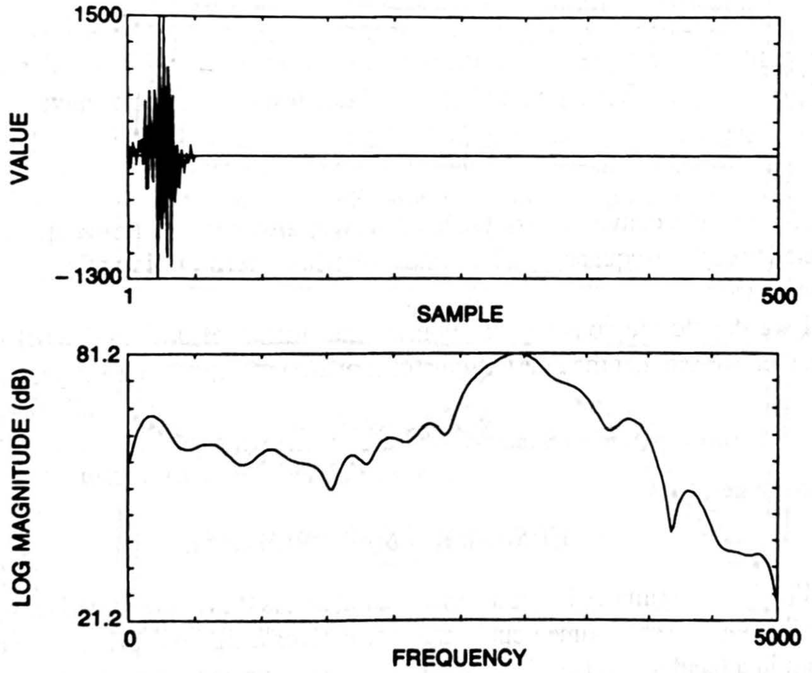

Figure 3.14 Short-time Fourier transform using a short (50 points or 5 msec) Hamming window on a section of unvoiced speech.

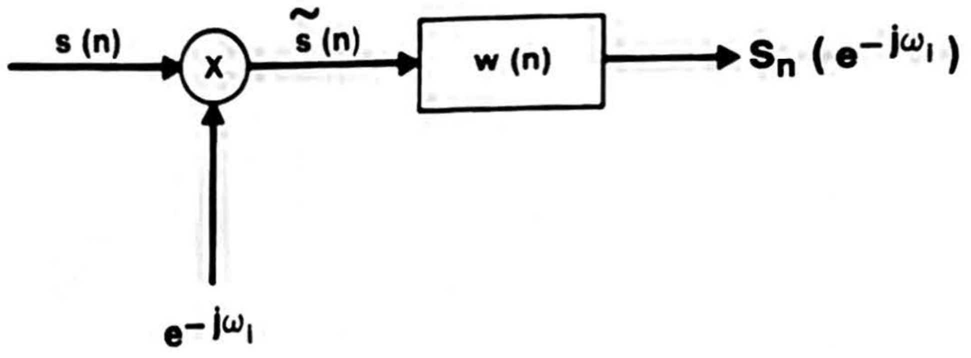

Figure 3.15 Linear filter interpretation of the short-time Fourier transform.

frequencies. Figure 3.12 shows a similar set of comparisons for an $L = 50$-point Hamming window. For such short windows, the time sequence $s(m)w(n-m)$ does not show the signal periodicity, nor does the signal spectrum. In fact, what we see in the short-time Fourier transform log magnitude is a few rather broad peaks in frequency corresponding roughly to the speech formants.

Figures 3.13 and 3.14 show the effects of using windows on a section of unvoiced speech (corresponding to the fricative /sh/) for an $L = 500$ sample window (Figure 3.13) and $L = 50$ sample window (Figure 3.14). Since there is no periodicity in the signal, the resulting short-time spectral magnitude of Figure 3.13, for the $L = 500$ sample window shows a ragged series of local peaks and valleys due to the random nature of the unvoiced speech. Using the shorter window smoothes out the random fluctuations in the short-time spectral magnitude and again shows the broad spectral envelope very well.

3.2.2.2 Linear Filtering Interpretation of the Short-Time Fourier Transform

The linear filtering interpretation of the short-time Fourier transform is derived by considering $S_n(e^{j\omega_i})$, of Eq. (3.16), for fixed values of ω_i, in which case we have

$$S_n(e^{j\omega_i}) = s(n)e^{-j\omega_i n} \circledast w(n). \tag{3.18}$$

That is, $S_n(e^{j\omega_i})$ is a convolution of the lowpass window, $w(n)$, with the speech signal, $s(n)$, modulated to center frequency ω_i. This linear filtering interpretation of $S_n(e^{j\omega_i})$ is illustrated in Figure 3.15.

If we denote the conventional Fourier transforms of $s(n)$ and $w(n)$ by $S(e^{j\omega})$ and $W(e^{j\omega})$, then we see that the Fourier transform of $\tilde{s}(n)$ of Figure 3.15 is just

$$\tilde{S}(e^{j\omega}) = S(e^{j(\omega+\omega_i)}) \tag{3.19}$$

and thus we get

$$\mathrm{FT}(S_n(e^{j\omega_i})) = S(e^{j(\omega+\omega_i)})W(e^{j\omega}). \tag{3.20}$$

Since $W(e^{j\omega})$ approximates 1 over a narrow band, and is 0 everywhere else, we see that, for fixed values, ω_i, the short-time Fourier transform gives a signal representative of the signal spectrum in a band around ω_i. Thus the short-time Fourier transform, $S_n(e^{j\omega_i})$, represents the signal spectral analysis at frequency ω_i by a filter whose bandwidth is that of $W(e^{j\omega})$.

3.2.2.3 Review Exercises

Exercise 3.1

A speech signal is sampled at a rate of 20,000 samples per second ($F_s = 20$ kHz). A 20-msec window is used for short-time spectral analysis, and the window is moved by 10 msec in consecutive analysis frames. Assume that a radix-2 FFT is used to compute DFTs.

1. How many speech samples are used in each segment?
2. What is the frame rate of the short-time spectral analysis?
3. What size DFT and FFT are required to guarantee that no time-aliasing will occur?
4. What is the resulting frequency resolution (spacing in Hz) between adjacent spectral samples?

Solution 3.1

1. Twenty msec of speech at the rate of 20,000 samples per second gives

$$20 \times 10^{-3} \text{ sec} \times 20{,}000 \text{ samples/sec} = 400 \text{ samples}.$$

Each section of speech is 400 samples in duration.

2. Since the shift between consecutive speech frames is 10 msec (i.e., 200 samples at a 20,000 samples/sec rate), the frame rate is

$$\text{frame rate} = \frac{1}{\text{frame shift}} = \frac{1}{10 \times 10^{-3} \text{ sec}} = 100/\text{sec}.$$

That is, 100 spectral analyses are performed per second of speech.

3. To avoid time aliasing in using the DFT to evaluate the short-time Fourier transform, we require the DFT size to be at least as large as the frame size of the analysis frame. Hence, from part 1, we require at least a 400-point DFT. Since we are using a radix 2 FFT, we require, in theory, a 512-point FFT (the smallest power of 2 greater than 400) to compute the DFT without time aliasing. (We would use the 400 speech samples as the first 400 points of the 512-point array; we pad 112 zero-valued samples to the end of the array to fill in and give a 512-point array.) Since the speech signal is real (as opposed to complex), we can use an FFT size of 256 by appropriate signal preprocessing and postprocessing with a complex FFT algorithm.

4. The frequency resolution of the analysis is defined as

$$\text{frequency resolution} = \frac{\text{sampling rate}}{\text{DFT size}} = \frac{20{,}000 \text{ Hz}}{512} \cong 39 \text{ Hz}.$$

Exercise 3.2

If the sequences $s(n)$ and $w(n)$ have normal (long-time) Fourier transforms $S(e^{j\omega})$ and $W(e^{j\omega})$, then show that the short-time Fourier transform

$$S_n(e^{j\omega}) = \sum_{m=-\infty}^{\infty} s(m)w(n-m)e^{-j\omega m}$$

can be expressed in the form

$$S_n(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\theta})e^{j\theta n}S(e^{j(\omega+\theta)})d\theta.$$

That is, $S_n(e^{j\omega})$ is a smoothed (by the window spectrum) spectral estimate of $S(e^{j\omega})$ at frequency ω.

Solution 3.2

The long-time Fourier transforms of $s(n)$ and $w(n)$ can be expressed as

$$S(e^{j\omega}) = \sum_{n=-\infty}^{\infty} s(n)e^{-j\omega n}$$

$$W(e^{j\omega}) = \sum_{n=-\infty}^{\infty} w(n)e^{-j\omega n}.$$

The window sequence, $w(n)$, can be recovered from its long-time Fourier transform via the integration

$$w(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\omega})e^{j\omega n}d\omega.$$

Hence, the short-time Fourier transform

$$S_n(e^{j\omega}) = \sum_{m=-\infty}^{\infty} s(m)w(n-m)e^{-j\omega m}$$

can be put in the form (by substituting for $w(n-m)$):

$$\begin{aligned} S_n(e^{j\omega}) &= \sum_{m=-\infty}^{\infty} s(m)\left[\frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\theta})e^{j\theta(n-m)}d\theta\right]e^{-j\omega m} \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\theta})e^{j\theta n}\left[\sum_{m=-\infty}^{\infty} s(m)e^{-j(\omega+\theta)m}\right]d\theta \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\theta})e^{j\theta n}S(e^{j(\omega+\theta)})d\theta. \end{aligned}$$

Exercise 3.3

If we define the short-time spectrum of a signal in terms of its short-time Fourier transform as

$$X_n(e^{j\omega}) = \left|S_n(e^{j\omega})\right|^2$$

and we define the short-time autocorrelation of the signal as

$$R_n(k) = \sum_{m=-\infty}^{\infty} w(n-m)s(m)w(n-k-m)\,s(m+k)$$

then show that for

$$S_n(e^{j\omega}) = \sum_{m=-\infty}^{\infty} s(m)w(n-m)e^{-j\omega m}$$

$R_n(k)$ and $X_n(e^{j\omega})$ are related as a normal (long-time) Fourier transform pair. In other words, show that $X_n(e^{j\omega})$ is the (long-time) Fourier transform of $R_n(k)$, and vice versa.

Solution 3.3

Given the definition of $S_n(e^{j\omega})$ we have

$$\begin{aligned} X_n(e^{j\omega}) &= \left|S_n(e^{j\omega})\right|^2 = [S_n(e^{j\omega})][S_n(e^{j\omega})]^* \\ &= \left[\sum_{m=-\infty}^{\infty} s(m)w(n-m)e^{-j\omega m}\right]\left[\sum_{r=-\infty}^{\infty} s(r)w(n-r)e^{j\omega r}\right] \\ &= \sum_{r=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} w(n-m)s(m)w(n-r)s(r)e^{-j\omega(m-r)} \end{aligned}$$

Let $r = k + m$, then:

$$\begin{aligned} X_n(e^{j\omega}) &= \sum_{k=-\infty}^{\infty}\left[\sum_{m=-\infty}^{\infty} s(m)w(n-m)s(m+k)\, w(n-k-m)\right]e^{j\omega k} \\ &= \sum_{k=-\infty}^{\infty} R_n(k)e^{j\omega k} = \sum_{k=-\infty}^{\infty} R_n(k)e^{-j\omega k} \end{aligned}$$

(since $R_n(k) = R_n(-k)$); therefore

$$X_n(e^{j\omega}) = \left|S_n(e^{j\omega})\right|^2 \overset{\text{FT}}{\longleftrightarrow} R_n(k).$$

3.2.2.4 FFT Implementation of Uniform Filter Bank Based on the Short-Time Fourier Transform

We now return to the question of how to efficiently implement the computation of the set of filter-bank outputs (Eq. (3.15)) for the uniform filter bank. If we assume, reasonably, that we are interested in a uniform frequency spacing—that is, if

$$f_i = i(F_s/N), \qquad i = 0, 1, \ldots, N-1 \tag{3.21}$$

then Eq. (3.15a) can be written as

$$x_i(n) = e^{j(\frac{2\pi}{N})in}\sum_m s(m)w(n-m)e^{-j(\frac{2\pi}{N})im}. \tag{3.22}$$

Now consider breaking up the summation over m, into a double summation of r and k, in which

$$m = Nr + k, \quad 0 \le k \le N-1, \quad -\infty < r < \infty. \tag{3.23}$$

In other words, we break up the computation over m into pieces of size N. If we let

$$s_n(m) = s(m)w(n-m), \tag{3.24}$$

then Eq. (3.22) can be written as

$$x_i(n) = e^{j(\frac{2\pi}{N})in}\sum_r\left[\sum_{k=0}^{N-1} s_n(Nr+k)\right]e^{-j(\frac{2\pi}{N})i(Nr+k)}. \tag{3.25}$$

Since $e^{-j2\pi ir} = 1$, for all i, r, then

$$x_i(n) = e^{j(\frac{2\pi}{N})in} \sum_{k=0}^{N-1} \left[\sum_r s_n(Nr + k) \right] e^{-j(\frac{2\pi}{N})ik}. \tag{3.26}$$

If we define

$$u_n(k) = \sum_r s_n(Nr + k), \qquad 0 \le k \le N - 1 \tag{3.27}$$

we wind up with

$$x_i(n) = e^{j(\frac{2\pi}{N})in} \left[\sum_{k=0}^{N-1} u_n(k) e^{-j(\frac{2\pi}{N})ik} \right] \tag{3.28}$$

which is the desired result; that is, $x_i(n)$ is a modulated N-point DFT of the sequence $u_n(k)$.

Thus the basic steps in the computation of a uniform filter bank via FFT methods are as follows:

1. Form the windowed signal $s_n(m) = s(m)\, w(n - m)$, $m = n - L + 1, \ldots, n$, where $w(n)$ is a causal, finite window of duration L samples. Figure 3.16a illustrates this step.
2. Form $u_n(k) = \sum_r s_n(Nr + k)$, $0 \le k \le N - 1$. That is, break the signal $s_n(m)$ into pieces of size N samples and add up the pieces (alias them back unto itself) to give a signal of size N samples. Figures 3.16b and c illustrate this step for the case in which $L \gg N$.
3. Take the N-point DFT of $u_n(k)$.
4. Modulate the DFT by the sequence $e^{j(\frac{2\pi}{N})in}$.

The modulation step 4 can be avoided by circularly shifting the sequence, $u_n(k)$, by $n \oplus N$ samples (where $\oplus$ is the modulo operation), to give $u_n((k - n))_N$, $0 \le k \le N - 1$, prior to the DFT computation.

The computation to implement the uniform filter bank via Eq. (3.28) is essentially

$$C_{\text{FBFFT}} \cong 2N \log N \cdot, +. \tag{3.29}$$

Consider now the ratio, R, between the computation for the direct form implementation of a uniform filter bank (Eq. (3.13)), and the FFT implementation (Eq. (3.29)), such that

$$R = \frac{C_{\text{DFFIR}}}{C_{\text{FBFFT}}} = \frac{LQ}{2N \log N}. \tag{3.30}$$

If we assume $N = 32$ (i.e., a 16-channel filter bank), with $L = 128$ (i.e., 12.8 msec impulse response filter at a 10-kHz sampling rate), and $Q = 16$ channels, we get

$$R = \frac{128 \cdot 16}{2 \cdot 32 \cdot 5} = 6.4.$$

The FFT implementation is about 6.4 times more efficient than the direct form structure.

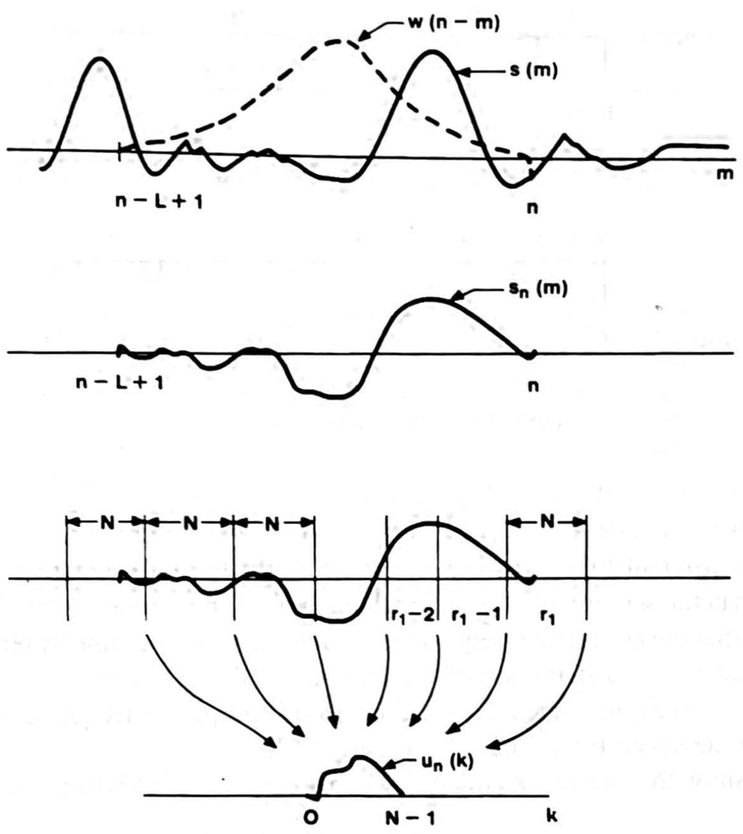

Figure 3.16 FFT implementation of a uniform filter bank.

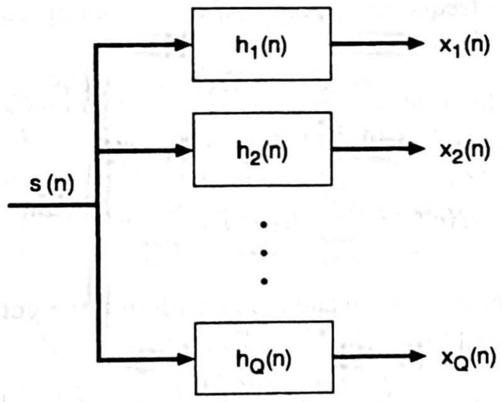

Figure 3.17 Direct form implementation of an arbitrary nonuniform filter bank.

3.2.2.5 Nonuniform FIR Filter Bank Implementations

The most general form of a nonuniform FIR filter bank is shown in Figure 3.17, where the k^{th} bandpass filter impulse response, $h_k(n)$, represents a filter with center frequency ω_k, and bandwidth $\Delta\omega_k$. The set of Q bandpass filters is intended to cover the frequency range of

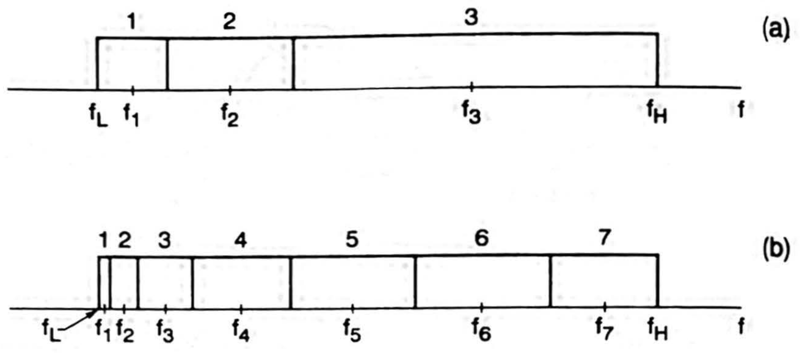

Figure 3.18 Two arbitrary nonuniform filter-bank ideal filter specifications consisting of either 3 bands (part a) or 7 bands (part b).

interest for the intended speech-processing application.

In its most general form, each bandpass filter is implemented via a direct convolution; that is, no efficient FFT structure can be used. In the case where each bandpass filter is designed via the windowing design method (Ref. [1]), using the same lowpass window, we can show that the composite frequency response of the Q-channel filter bank is independent of the number and distribution of the individual filters. Thus a filter bank with the three filters shown in Figure 3.18a has the exact same composite frequency response as the filter bank with the seven filters shown in Figure 3.18b.

To show this we denote the impulse response of the k^{th} bandpass filter as

$$h_k(n) = w(n)\tilde{h}_k(n), \tag{3.31}$$

where $w(n)$ is the FIR window, and $\tilde{h}_k(n)$ is the impulse response of the ideal bandpass filter being designed. The frequency response of the k^{th} bandpass filter, $H_k(e^{j\omega})$, can be written as

$$H_k(e^{j\omega}) = W(e^{j\omega}) \circledast \tilde{H}_k(e^{j\omega}). \tag{3.32}$$

Thus the frequency response of the composite filter bank, $H(e^{j\omega})$, can be written as

$$H(e^{j\omega}) = \sum_{k=1}^{Q} H_k(e^{j\omega}) = \sum_{k=1}^{Q} W(e^{j\omega}) \circledast \tilde{H}_k(e^{j\omega}). \tag{3.33}$$

By interchanging the summation and the convolution we get

$$H(e^{j\omega}) = W(e^{j\omega}) \circledast \sum_{k=1}^{Q} \tilde{H}_k(e^{j\omega}). \tag{3.34}$$

By realizing that the summation of Eq. (3.34) is the summation of ideal frequency responses, we see that it is independent of the number and distribution of the individual filters. Thus we can write the summation as

$$\hat{H}(e^{j\omega}) = \sum_{k=1}^{Q} \tilde{H}_k(e^{j\omega}) = \begin{cases} 1, & \omega_{\min} \le \omega \le \omega_{\max} \\ 0, & \text{otherwise} \end{cases}, \tag{3.35}$$

where $\omega_{\min}$ is the lowest frequency in the filter bank, and $\omega_{\max}$ is the highest frequency. Then Eq. (3.34) can be expressed as

$$H(e^{j\omega}) = W(e^{j\omega}) \circledast \hat{H}(e^{j\omega}) \tag{3.36}$$

independent of the number of ideal filters, Q, and their distribution in frequency, which is the desired result.

3.2.2.6 FFT-Based Nonuniform Filter Banks

One possible way to exploit the FFT structure for implementing uniform filter banks discussed earlier is to design a large uniform filter bank (e.g., $N = 128$ or 256 channels) and then create the nonuniformity by combining two or more uniform channels. This technique of combining channels is readily shown to be equivalent to applying a modified analysis window to the sequence prior to the FFT. To see this, consider taking an N-point DFT of the sequence $x(n)$ (derived from the speech signal, $s(n)$, by windowing by $w(n)$). Thus we get

$$X_k = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}nk}, \qquad 0 \le k \le N-1 \tag{3.37}$$

as the set of DFT values. If we consider adding DFT outputs X_k and X_{k+1}, we get

$$X_k + X_{k+1} = \sum_{n=0}^{N-1} x(n) \left(e^{-j\frac{2\pi}{N}nk} + e^{-j\frac{2\pi}{N}n(k+1)} \right) \tag{3.38}$$

which can be written as

$$X'_k = X_k + X_{k+1} = \sum_{n=0}^{N-1} \left[x(n) 2e^{-j\frac{\pi n}{N}} \cos\left(\frac{\pi n}{N}\right) \right] e^{-j\frac{2\pi}{N}nk} \tag{3.39}$$

i.e. the equivalent k^{th} channel value, X'_k, could have been obtained by weighting the sequence, $x(n)$, in time, by the complex sequence $2e^{-j\frac{\pi n}{N}} \cos\left(\frac{\pi n}{N}\right)$. If more than two channels are combined, then a different equivalent weighting sequence results. Thus FFT channel combining is essentially a "quick and dirty" method of designing broader bandpass filters and is a simple and effective way of realizing certain types of nonuniform filter bank analysis structures.

3.2.2.7 Tree Structure Realizations of Nonuniform Filter Banks

A third method used to implement certain types of nonuniform filter banks is the tree structure in which the speech signal is filtered in stages, and the sampling rate is successively reduced at each stage for efficiency of implementation. An example of such a realization is given in Figure 3.19a for the 4-band, octave-spaced filter bank shown (ideally) in Figure 3.19b. The original speech signal, $s(n)$, is filtered initially into two bands, a low band and a high band, using quadrature mirror filters (QMFs)—i.e., filters whose frequency responses are complementary. The high band, which covers half the spectrum, is reduced in sampling rate by a factor of 2, and represents the highest octave band ($\pi/2 \le \omega \le \pi$) of

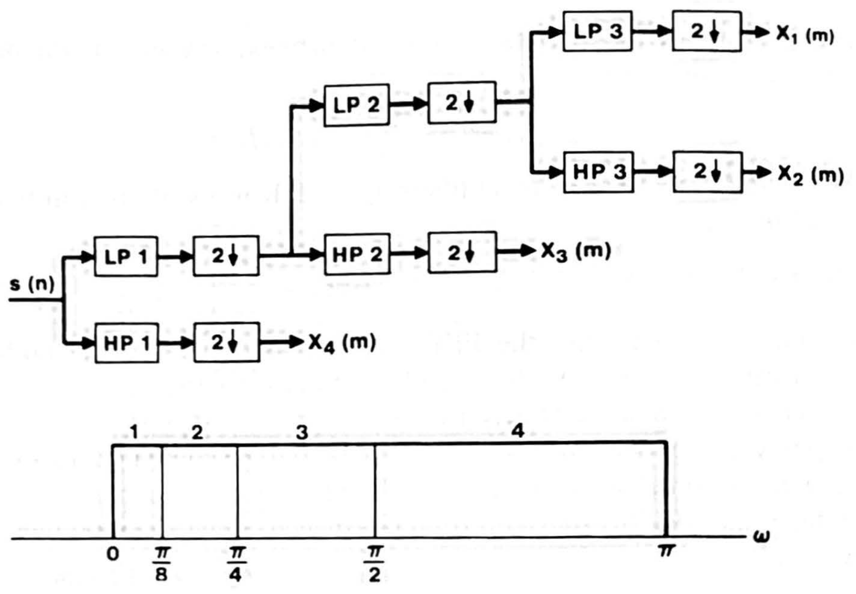

Figure 3.19 Tree structure implementation of a 4-band, octave-spaced, filter bank.

the filter bank. The low band is similarly reduced in sampling rate by a factor of 2, and is fed into a second filtering stage in which the signal is again split into two equal bands using QMF filters. Again the high band of stage 2 is decimated by a factor of 2 and is used as the next-highest filter bank output; the low band is also decimated by a factor of 2 and fed into a third stage of QMF filters. These third-stage outputs, in this case after decimation by a factor of 2, are used as the two lowest filter bands.

QMF filter bank structures are quite efficient and have been used for a number of speech-processing applications [3]. Their efficiency for arbitrary nonuniform filter bank structures is not as good as for the octave band designs of Figure 3.19.

3.2.3 Summary of Considerations for Speech-Recognition Filter Banks

In the previous sections we discussed several methods of implementing filter banks for speech recognition. We have not gone into great detail here because our goal was to make the reader familiar with the issues involved in filter-bank design and implementation, not to make the reader an expert in signal processing. The interested reader is urged to pursue this fascinating area further by studying the material in the References at the end of this chapter. In this section we summarize the considerations that go into choosing the number and types of filters used in the structures discussed earlier in this section.

The first consideration for any filter bank is the type of digital filter used. The choices are IIR (recursive) and FIR (nonrecursive) designs. The IIR designs have the advantage of being implementable in simple, efficient structures. The big disadvantage of IIR filters is that their phase response is nonlinear; hence, to minimize this disadvantage a trade-off is usually made between the ideal magnitude characteristics that can readily be realized, and the highly nonideal phase characteristics. On the other hand, FIR filters can achieve linear phase without compromising the ability to approximate ideal magnitude characteristics; however, they are usually computationally expensive in implementation. For speech-recognition applications, we have shown how an FFT structure can often be applied to alleviate considerably the computational inefficiency of FIR filter banks; hence, most practical digital filter bank structures use FIR filters (usually in an FFT realization).

Once the type of filter has been decided, the next consideration is the number of filters to be used in the filter bank. For uniform filter banks, the number of filters, Q, cannot be too small or else the ability of the filter bank to resolve the speech spectrum is greatly impaired. Thus values of Q less than about 8 are generally avoided. Similarly, the value of Q cannot be too large (unless there is considerable filter overlap), because the filter bandwidths would eventually be too narrow for some talkers (e.g., high-pitch females or children), and there would be a high probability that certain bands would have extremely low speech energy (i.e., no prominent harmonic would fall within the band). Thus, practical systems tend to have values of $Q \leq 32$. Although uniformly spaced filter banks have been widely used for recognition, many practical systems have used nonuniform spacing in an effort to reduce overall computation and to characterize the speech spectrum in a manner considered more consistent with human perception.

A final consideration for practical filter-bank analyzers is the choice of nonlinearity and lowpass filter used at the output of each channel. Typically the nonlinearity has been a full wave rectifier (FWR), a half wave rectifier (HWR), or a center clipper. The resultant spectrum is only weakly sensitive to the nonlinearity. The lowpass filter used in practice varies from a simple integrator to a fairly good quality IIR lowpass filter (typically a Bessel filter).

3.2.4 Practical Examples of Speech-Recognition Filter Banks

Figures 3.20–3.25 [4] show examples of a wide range of speech-recognition filter banks, including both uniform and nonuniform designs. Figure 3.20 is for a 15-channel uniform filter bank in which the basic lowpass filter was designed using the windowing technique with a 101-point Kaiser window. Part a of the figure shows the impulse response of the lowpass filter (i.e., an ideal lowpass filter response multiplied by a Kaiser window). Part b of the figure shows the responses of the individual filters in the filter bank (note there is no overlap between adjacent filters), and part c shows the composite frequency response of the overall filter bank. The sidelobe peak ripple of each individual filter is down about 60 dB, and the composite frequency response is essentially ideally flat over the entire frequency range of interest (approximate 100–3000 Hz).

By contrast, Figure 3.21 is for a 15-channel uniform filter bank in which the basic lowpass filter was a Kaiser window (instead of the Kaiser windowed version of the ideal lowpass filter). From parts b and c of this figure, it can be seen that the individual bandpass filters are narrower in bandwidth than those of Figure 3.20; furthermore, the composite

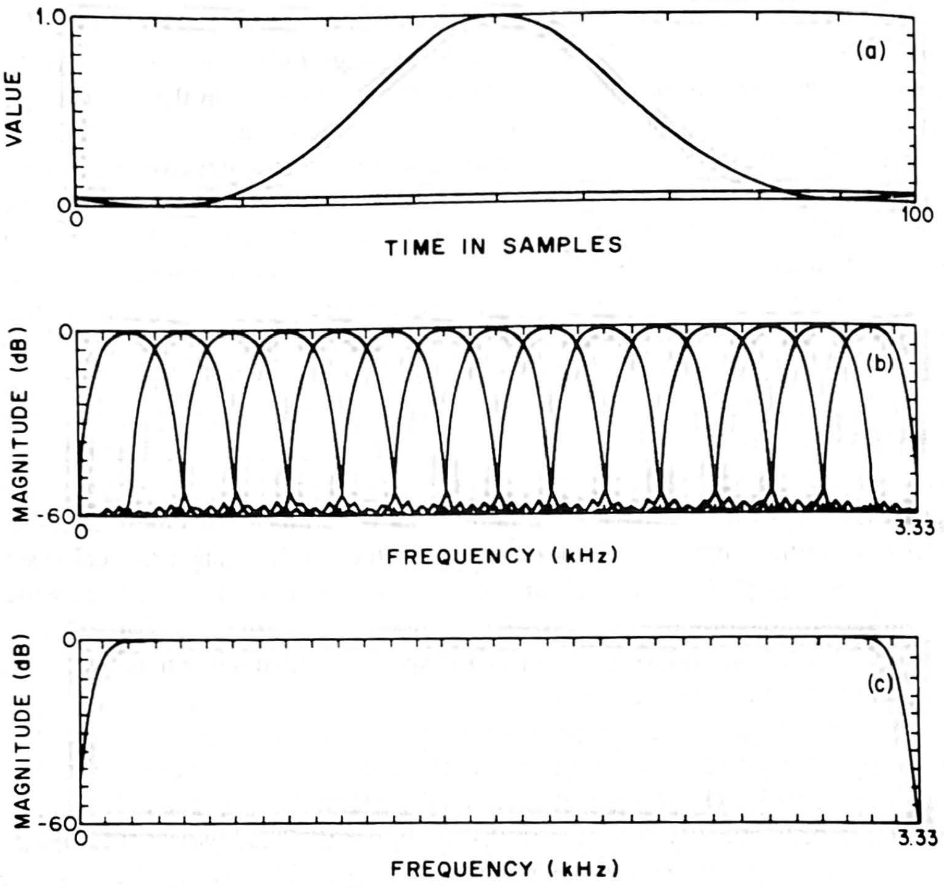

Figure 3.20 Window sequence, $w(n)$, (part a), the individual filter response (part b), and the composite response (part c) of a $Q = 15$ channel, uniform filter bank, designed using a 101-point Kaiser window smoothed lowpass window (after Dautrich et al. [4]).

filter-bank response shows 18 dB gaps at the boundaries between each filter. Clearly, this filter bank would be unacceptable for speech-recognition applications.

Figures 3.22 and 3.23 show individual filter frequency responses, and the composite frequency response, for a 4-channel, octave-band filter bank, and a 12-channel, 1/3 octave filter bank, frequency, respectively. Each of these nonuniform filter banks was designed to cover the frequency band from 200 to 3200 Hz and used linear-phase FIR filters (101 points for the octave band design, and 201 points for the 1/3 octave band design) for each individual channel. The peak sidelobe ripple was about −40 dB for both filter banks.

Figure 3.24 shows a similar set of responses for a 7-channel critical band filter bank in which each individual filter encompassed two critical bands. Again we used 101-point, linear phase, FIR filters with a peak sidelobe of −54 dB to realize each individual bandpass filter. Finally, Figure 3.25 shows the responses of a 13-channel, critical band filter bank in which the individual channels were highly overlapping. The individual bandpass filter responses are rather poor (e.g., the ratios of center frequency to bandwidth of each filter was about 8). However, this poor frequency resolution characteristic was balanced somewhat by the excellent time resolution of the filters.

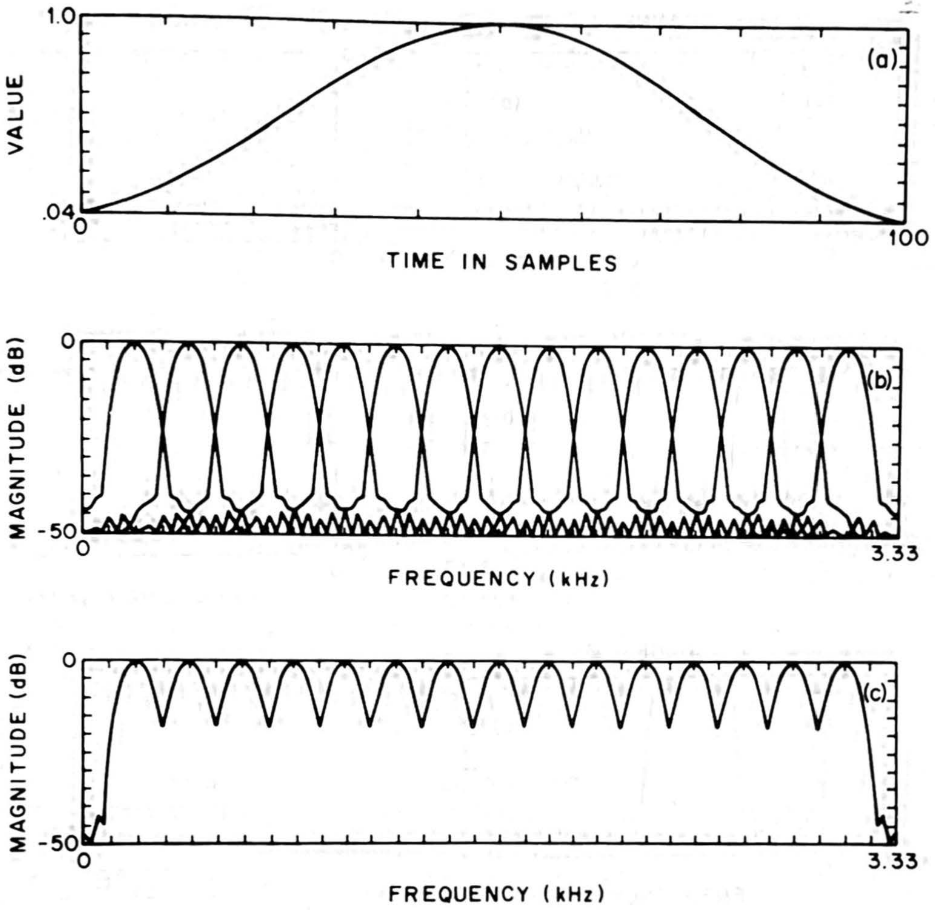

Figure 3.21 Window sequence, $w(n)$, (part a), the individual filter responses (part b), and the composite response (part c) of a $Q = 15$ channel, uniform filter bank, designed using a 101-point Kaiser window directly as the lowpass window (after Dautrich et al. [4]).

3.2.5 Generalizations of Filter-Bank Analyzer

Although we have been concerned primarily with designing and implementing individual channels of a filter-bank analyzer, there is a generalized structure that must be considered as part of the canonic filter-bank analysis method. This generalized structure is shown in Figure 3.26. The generalization includes a signal preprocessor that "conditions" the speech signal, $s(n)$, to a new form, $\hat{s}(n)$, which is "more suitable" for filter-bank analysis, and a postprocessor that operates on the filter-bank output vectors, $x(m)$, to give the processed vectors $\hat{x}(m)$ that are "more suitable" for recognition. Although a wide range of signal-processing operations could go into the preprocessor and postprocessor boxes, perhaps the most reasonable ones include the following.

Preprocessor Operations

- signal preemphasis (to equalize the inherent spectral tilt in speech)

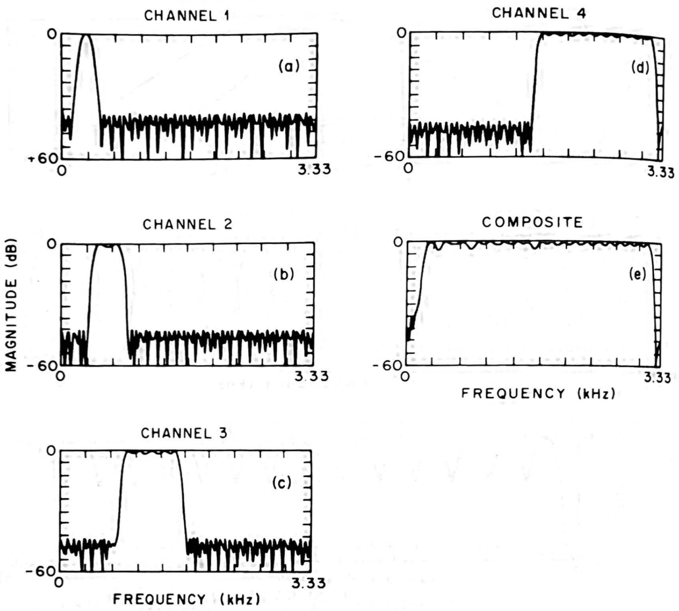

Figure 3.22 Individual channel responses (parts a to d) and composite filter response (part c) of a $Q = 4$ channel, octave band design, using 101-point FIR filters in each band (after Dautrich et al. [4]).

- noise elimination
- signal enhancement (to make the formant peaks more prominent)

Postprocessor Operations

- temporal smoothing of sequential filter-bank output vectors
- frequency smoothing of individual filter-bank output vectors
- normalization of each filter-bank output vector
- thresholding and/or quantization of the filter-bank output vectors
- principal components analysis of the filter-bank output vector.

The purpose of the preprocessor is to make the speech signal as clean as possible so far as the filter bank analyzer is concerned; hence, noise is eliminated, long-time spectral trends are removed, and the signal is spectrally flattened to give the best immunity to measurement imperfections. Similarly, the purpose of the postprocessor is to clean up the

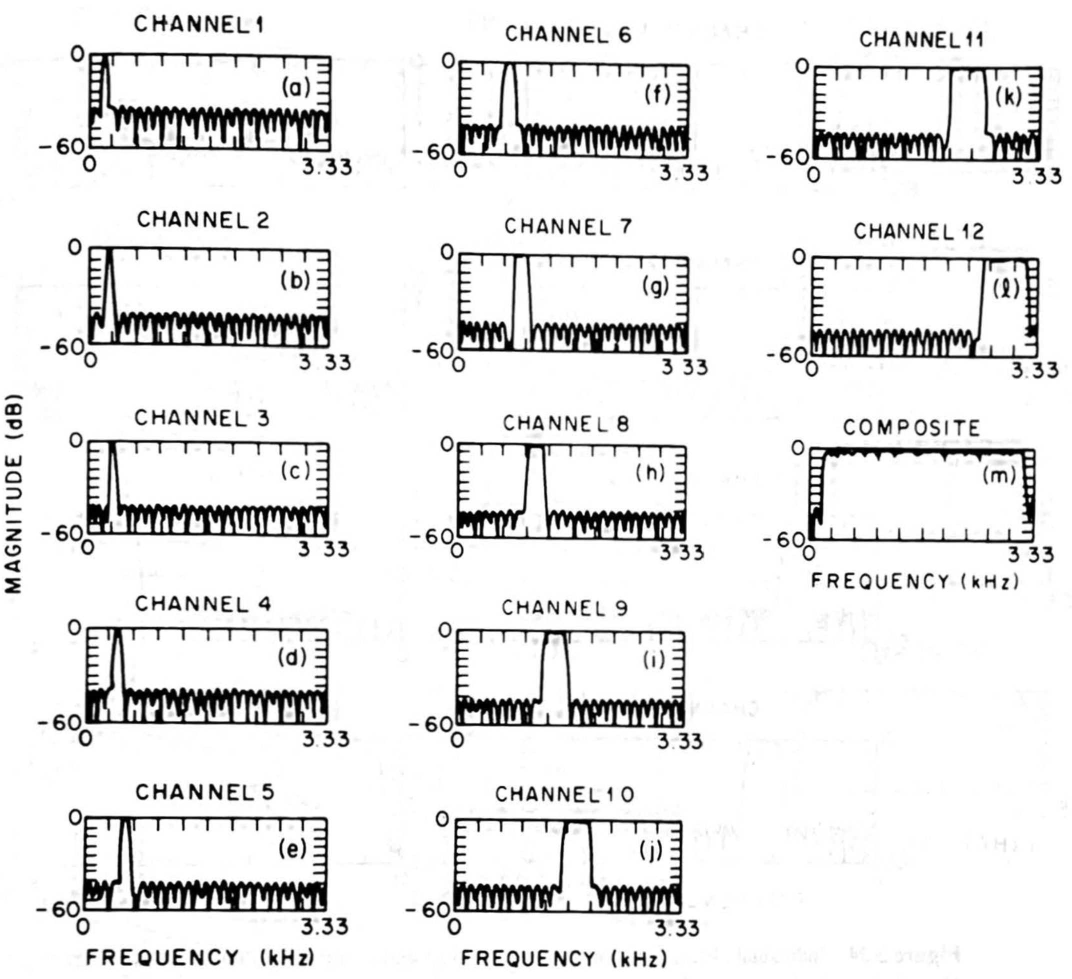

Figure 3.23 Individual channel responses and composite filter response of a $Q = 12$ channel, 1/3 octave band design, using 201-point FIR filters in each band (after Dautrich et al. [4]).

sequence of feature vectors from the filter-bank analyzer so as to best represent the spectral information in the speech signal and thereby to maximize the chances of successful speech recognition [4,5].

3.3 LINEAR PREDICTIVE CODING MODEL FOR SPEECH RECOGNITION

The theory of linear predictive coding (LPC), as applied to speech, has been well understood for many years (see for example Ref. [6]). In this section we describe the basics of how LPC has been applied in speech-recognition systems. The mathematical details and derivations will be omitted here; the interested reader is referred to the references.

Before describing a general LPC front-end processor for speech recognition, it is worthwhile to review the reasons why LPC has been so widely used. These include the following:

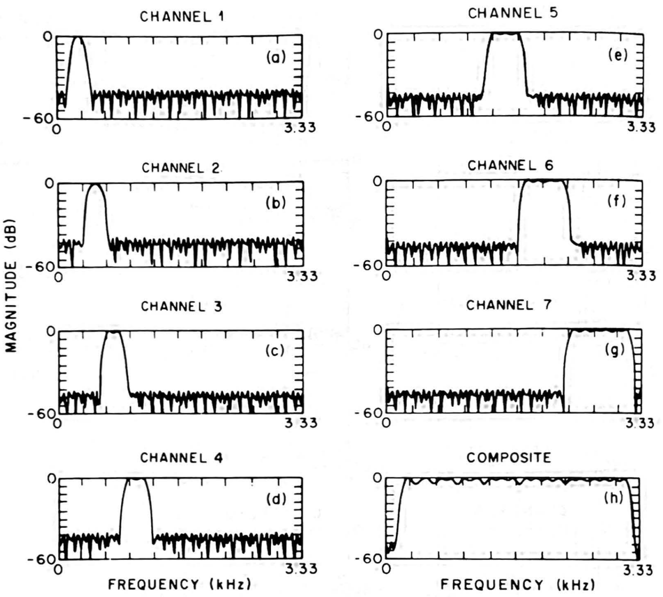

Figure 3.24 Individual channel responses (parts a to g) and composite filter response (part h) of a $Q = 7$ channel critical band filter bank design (after Dautrich et al. [4]).

1. LPC provides a good model of the speech signal. This is especially true for the quasi steady state voiced regions of speech in which the all-pole model of LPC provides a good approximation to the vocal tract spectral envelope. During unvoiced and transient regions of speech, the LPC model is less effective than for voiced regions, but it still provides an acceptably useful model for speech-recognition purposes.
2. The way in which LPC is applied to the analysis of speech signals leads to a reasonable source-vocal tract separation. As a result, a parsimonious representation of the vocal tract characteristics (which we know are directly related to the speech sound being produced) becomes possible.
3. LPC is an analytically tractable model. The method of LPC is mathematically precise and is simple and straightforward to implement in either software or hardware. The computation involved in LPC processing is considerably less than that required for an all-digital implementation of the bank-of-filters model described in Section 3.2.
4. The LPC model works well in recognition applications. Experience has shown that

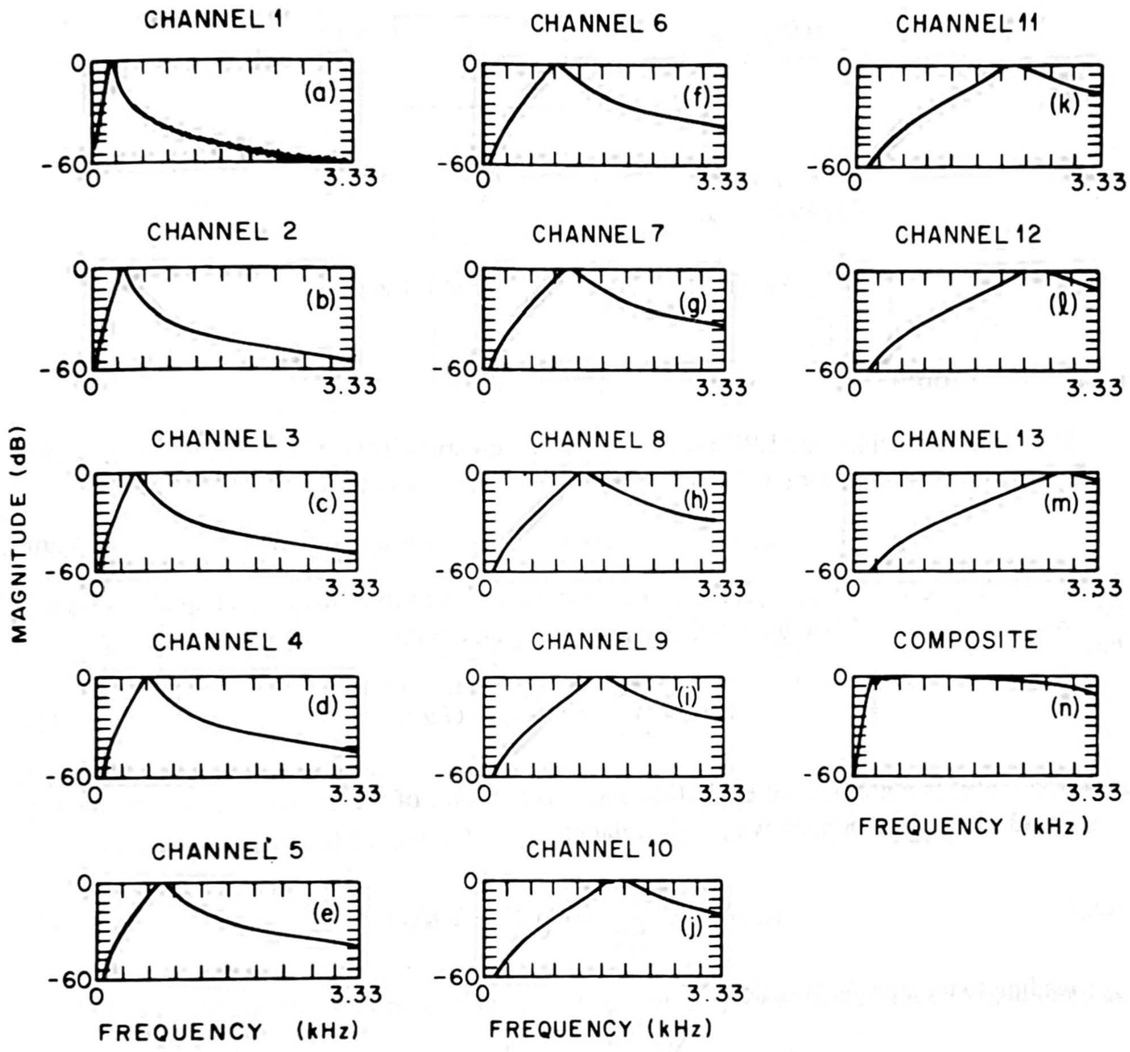

Figure 3.25 Individual channel responses and composite filter response of a $Q = 13$ channel, critical band spacing filter bank, using highly overlapping filters in frequency (after Dautrich et al. [4]).

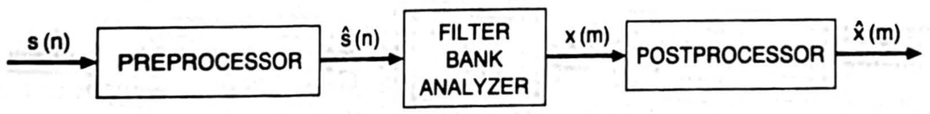

Figure 3.26 Generalization of filter-bank analysis model.

the performance of speech recognizers, based on LPC front ends, is comparable to or better than that of recognizers based on filter-bank front ends (see References [4,5,7]).

Based on the above considerations, LPC front-end processing has been used in a large number of recognizers. In particular, most of the systems to be described in this book are based on this model.

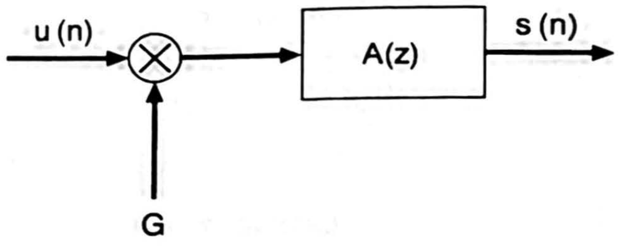

Figure 3.27 Linear prediction model of speech.

3.3.1 The LPC Model

The basic idea behind the LPC model is that a given speech sample at time n, $s(n)$, can be approximated as a linear combination of the past p speech samples, such that

$$s(n) \approx a_1 s(n-1) + a_2 s(n-2) + \cdots + a_p s(n-p), \tag{3.40}$$

where the coefficients $a_1, a_2, \ldots, a_p$ are assumed constant over the speech analysis frame. We convert Eq. (3.40) to an equality by including an excitation term, $G\,u(n)$, giving:

$$s(n) = \sum_{i=1}^{p} a_i s(n-i) + G\,u(n), \tag{3.41}$$

where $u(n)$ is a normalized excitation and G is the gain of the excitation. By expressing Eq. (3.41) in the z-domain we get the relation

$$S(z) = \sum_{i=1}^{p} a_i z^{-i} S(z) + G\,U(z) \tag{3.42}$$

leading to the transfer function

$$H(z) = \frac{S(z)}{G\,U(z)} = \frac{1}{1 - \sum_{i=1}^{p} a_i z^{-i}} = \frac{1}{A(z)}. \tag{3.43}$$

The interpretation of Eq. (3.43) is given in Figure 3.27, which shows the normalized excitation source, $u(n)$, being scaled by the gain, G, and acting as input to the all-pole system, $H(z) = \frac{1}{A(z)}$, to produce the speech signal, $s(n)$. Based on our knowledge that the actual excitation function for speech is essentially either a quasiperiodic pulse train (for voiced speech sounds) or a random noise source (for unvoiced sounds), the appropriate synthesis model for speech, corresponding to the LPC analysis, is as shown in Figure 3.28. Here the normalized excitation source is chosen by a switch whose position is controlled by the voiced/unvoiced character of the speech, which chooses either a quasiperiodic train of pulses as the excitation for voiced sounds, or a random noise sequence for unvoiced sounds. The appropriate gain, G, of the source is estimated from the speech signal, and the scaled source is used as input to a digital filter ($H(z)$), which is controlled by the vocal tract

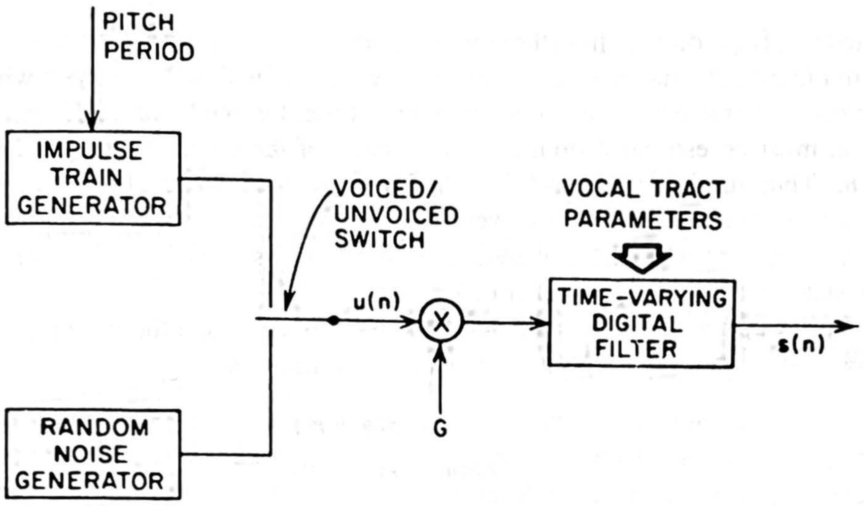

Figure 3.28 Speech synthesis model based on LPC model.

parameters characteristic of the speech being produced. Thus the parameters of this model are voiced/unvoiced classification, pitch period for voiced sounds, the gain parameter, and the coefficients of the digital filter, $\{a_k\}$. These parameters all vary slowly with time.

3.3.2 LPC Analysis Equations

Based on the model of Figure 3.27, the exact relation between $s(n)$ and $u(n)$ is

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + G\,u(n). \tag{3.44}$$

We consider the linear combination of past speech samples as the estimate $\tilde{s}(n)$, defined as

$$\tilde{s}(n) = \sum_{k=1}^{p} a_k s(n-k). \tag{3.45}$$

We now form the prediction error, $e(n)$, defined as

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \tag{3.46}$$

with error transfer function

$$A(z) = \frac{E(z)}{S(z)} = 1 - \sum_{k=1}^{p} a_k z^{-k}. \tag{3.47}$$

Clearly, when $s(n)$ is actually generated by a linear system of the type shown in Figure 3.27, then the prediction error, $e(n)$, will equal $G\,u(n)$, the scaled excitation.

The basic problem of linear prediction analysis is to determine the set of predictor coefficients, $\{a_k\}$, directly from the speech signal so that the spectral properties of the digital filter of Figure 3.28 match those of the speech waveform within the analysis window. Since the spectral characteristics of speech vary over time, the predictor coefficients at a given time, n, must be estimated from a *short* segment of the speech signal occurring around time n. Thus the basic approach is to find a set of predictor coefficients that minimize the mean-squared prediction error over a short segment of the speech waveform. (Usually this type of short time spectral analysis is performed on successive frames of speech, with frame spacing on the order of 10 msec.)

To set up the equations that must be solved to determine the predictor coefficients, we define short-term speech and error segments at time n as

$$s_n(m) = s(n+m) \tag{3.48a}$$

$$e_n(m) = e(n+m) \tag{3.48b}$$

and we seek to minimize the mean squared error signal at time n

$$E_n = \sum_m e_n^2(m) \tag{3.49}$$

which, using the definition of $e_n(m)$ in terms of $s_n(m)$, can be written as

$$E_n = \sum_m \left[s_n(m) - \sum_{k=1}^{p} a_k s_n(m-k) \right]^2 . \tag{3.50}$$

To solve Eq. (3.50), for the predictor coefficients, we differentiate E_n with respect to each a_k and set the result to zero,

$$\frac{\partial E_n}{\partial a_k} = 0, \qquad k = 1, 2, \ldots, p \tag{3.51}$$

giving

$$\sum_m s_n(m-i)s_n(m) = \sum_{k=1}^{p} \hat{a}_k \sum_m s_n(m-i)s_n(m-k). \tag{3.52}$$

By recognizing that terms of the form $\sum s_n(m-i)\ s_n(m-k)$ are terms of the short-term covariance of $s_n(m)$, i.e.,

$$\phi_n(i,k) = \sum_m s_n(m-i)s_n(m-k) \tag{3.53}$$

we can express Eq. (3.52) in the compact notation

$$\boxed{\phi_n(i,0) = \sum_{k=1}^{p} \hat{a}_k \phi_n(i,k)} \tag{3.54}$$

which describes a set of p equations in p unknowns. It is readily shown that the minimum mean-squared error, $\hat{E}_n$, can be expressed as

$$\hat{E}_n = \sum_m s_n^2(m) - \sum_{k=1}^{p} \hat{a}_k \sum_m s_n(m) s_n(m-k) \tag{3.55}$$

$$= \phi_n(0,0) - \sum_{k=1}^{p} \hat{a}_k \phi_n(0,k). \tag{3.56}$$

Thus the minimum mean-squared error consists of a fixed term ($\phi_n(0,0)$) and terms that depend on the predictor coefficients.

To solve Eq. (3.54) for the optimum predictor coefficients (the $\hat{a}_k$s) we have to compute $\phi_n(i,k)$ for $1 \le i \le p$ and $0 \le k \le p$, and then solve the resulting set of p simultaneous equations. In practice, the method of solving the equations (as well as the method of computing the ϕs) is a strong function of the range of m used in defining both the section of speech for analysis and the region over which the mean-squared error is computed. We now discuss two standard methods of defining this range for speech.

3.3.3 The Autocorrelation Method

A fairly simple and straightforward way of defining the limits on m in the summations is to assume that the speech segment, $s_n(m)$, is identically zero outside the interval $0 \le m \le N-1$. This is equivalent to assuming that the speech signal, $s(m+n)$, is multiplied by a finite length window, $w(m)$, which is identically zero outside the range $0 \le m \le N-1$. Thus the speech sample for minimization can be expressed as

$$s_n(m) = \begin{cases} s(m+n) \cdot w(m), & 0 \le m \le N-1 \\ 0, & \text{otherwise.} \end{cases} \tag{3.57}$$

The effect of weighting of the speech by a window is illustrated in Figures 3.29–3.31. In each of these figures, the upper panel shows the running speech waveform, $s(m)$, the middle panel shows the weighted section of speech (using a Hamming window for $w(m)$), and the bottom panel shows the resulting error signal, $e_n(m)$, based on optimum selection of the predictor parameters.

Based on Eq. (3.57), for $m < 0$, the error signal $e_n(m)$ is exactly zero since $s_n(m) = 0$ for all $m < 0$ and therefore there is no prediction error. Furthermore, for $m > N-1+p$ there is again no prediction error because $s_n(m) = 0$ for all $m > N-1$. However, in the region of $m = 0$ (i.e., from $m = 0$ to $m = p-1$) the windowed speech signal $s_n(m)$ is being predicted from previous samples, some of which are arbitrarily zero. Hence the potential for relatively large prediction errors exists in this region and can actually be seen to exist in the bottom panel of Figure 3.29. Furthermore, in the region of $m = N-1$ (i.e., from $m = N-1$ to $m = N-1+p$) the potential of large prediction errors again exists because the zero-valued (weighted) speech signal is being predicted from at least some nonzero previous speech samples. In the bottom panel of Figure 3.30 we see this effect at the end of the prediction error waveform. These two effects are especially prominent for voiced speech when the beginning of a pitch period occurs at or very close to the $m = 0$ or $m = N-1$ points of the sample. For unvoiced speech, these problems are essentially eliminated because no part of the waveform is position sensitive. Hence we see

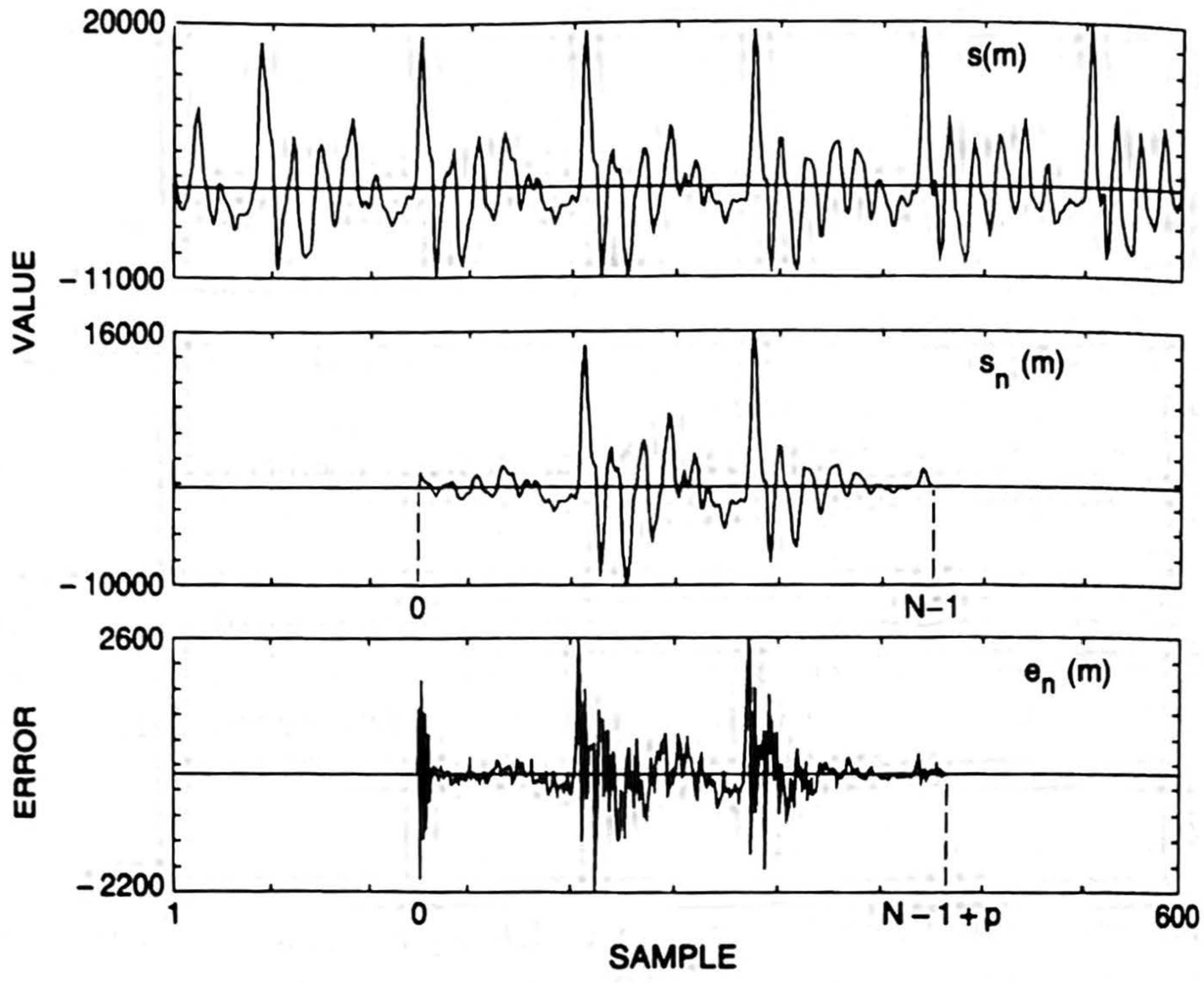

Figure 3.29 Illustration of speech sample, weighted speech section, and prediction error for voiced speech where the prediction error is large at the beginning of the section.

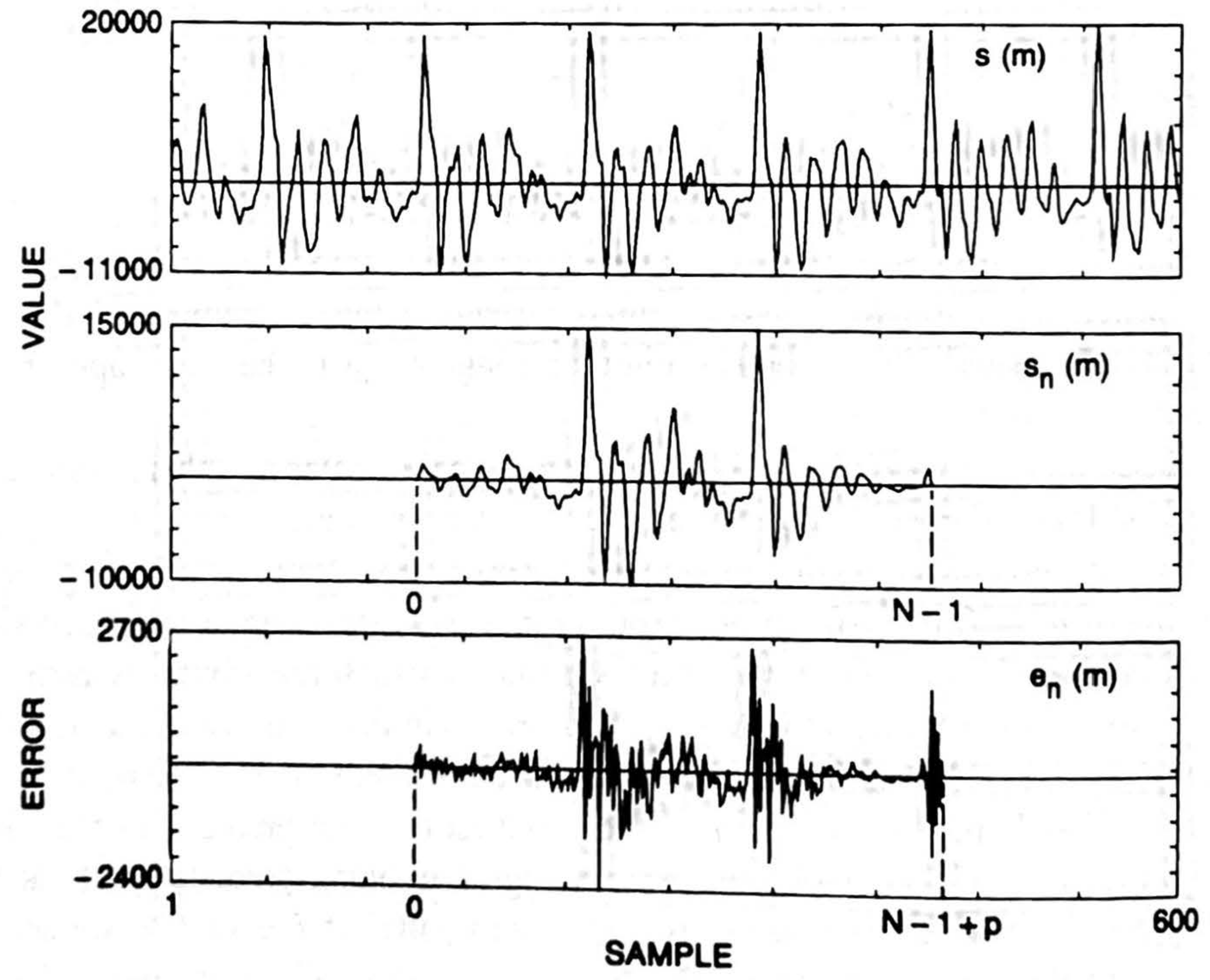

Figure 3.30 Illustration of speech sample, weighted speech section, and prediction error for voiced speech where the prediction error is large at the end of the section.

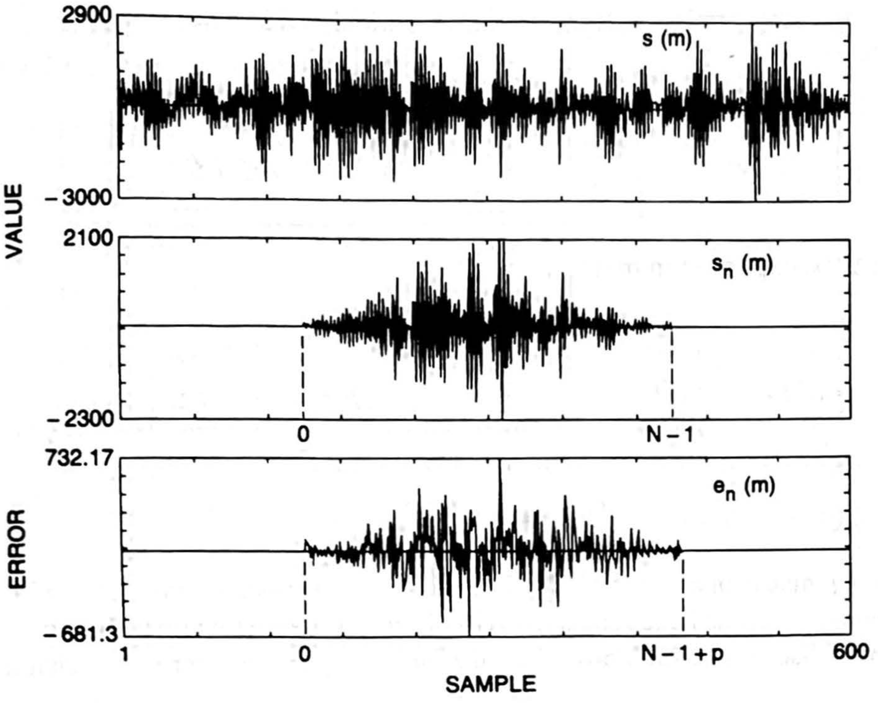

Figure 3.31 Illustration of speech sample, weighted speech section, and prediction error for unvoiced speech where there are almost no artifacts at the boundaries of the section.

neither effect occurring in the bottom panel of Figure 3.3.1. The purpose of the window of Eq. (3.57) is to taper the signal near $m = 0$ and near $m = N - 1$ so as to minimize the errors at section boundaries.

Based on using the weighted signal of Eq. (3.57) the mean-squared error becomes

$$E_n = \sum_{m=0}^{N-1+p} e_n^2(m) \tag{3.58}$$

and $\phi_n(i, k)$ can be expressed as

$$\phi_n(i, k) = \sum_{m=0}^{N-1+p} s_n(m - i)s_n(m - k), \qquad \begin{matrix} 1 \le i \le p \\ 0 \le k \le p \end{matrix} \tag{3.59}$$

or

$$\phi_n(i, k) = \sum_{m=0}^{N-1-(i-k)} s_n(m)s_n(m + i - k), \qquad \begin{matrix} 1 \le i \le p \\ 0 \le k \le p \end{matrix}. \tag{3.60}$$

Since Eq. (3.60) is only a function of $i - k$ (rather than the two independent variables i and k), the covariance function, $\phi_n(i, k)$, reduces to the simple autocorrelation function, i.e.,

$$\phi_n(i, k) = r_n(i - k) = \sum_{m=0}^{N-1-(i-k)} s_n(m)s_n(m + i - k). \tag{3.61}$$

Since the autocorrelation function is symmetric, i.e. $r_n(-k) = r_n(k)$, the LPC equations can be expressed as

$$\boxed{\sum_{k=1}^{p} r_n(|i-k|)\hat{a}_k = r_n(i), \qquad 1 \le i \le p} \tag{3.62}$$

and can be expressed in matrix form as

$$\begin{bmatrix} r_n(0) & r_n(1) & r_n(2) & \cdots & r_n(p-1) \\ r_n(1) & r_n(0) & r_n(1) & \cdots & r_n(p-2) \\ r_n(2) & r_n(1) & r_n(0) & \cdots & r_n(p-3) \\ \vdots & \vdots & \vdots & & \vdots \\ r_n(p-1) & r_n(p-2) & r_n(p-3) & \cdots & r_n(0) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \\ \vdots \\ \hat{a}_p \end{bmatrix} = \begin{bmatrix} r_n(1) \\ r_n(2) \\ r_n(3) \\ \vdots \\ r_n(p) \end{bmatrix}. \tag{3.63}$$

The $p \times p$ matrix of autocorrelation values is a Toeplitz matrix (symmetric with all diagonal elements equal) and hence can be solved efficiently through several well-known procedures. (We will discuss one such procedure, the Durbin algorithm, later in this chapter.)

3.3.4 The Covariance Method

An alternative to using a weighting function or window for defining $s_n(m)$ is to fix the interval over which the mean-squared error is computed to the range $0 \le m \le N-1$ and to use the unweighted speech directly—that is,

$$E_n = \sum_{m=0}^{N-1} e_n^2(m) \tag{3.64}$$

with $\phi_n(i,k)$ defined as

$$\phi_n(i,k) = \sum_{m=0}^{N-1} s_n(m-i)s_n(m-k), \qquad \begin{matrix} 1 \le i \le p \\ 0 \le k \le p \end{matrix} \tag{3.65}$$

or, by a change of variables,

$$\phi_n(i,k) = \sum_{m=-i}^{N-i-1} s_n(m)s_n(m+i-k), \qquad \begin{matrix} 1 \le i \le p \\ 0 \le k \le p \end{matrix}. \tag{3.66}$$

If we consider when $i = p$ we see that the computation of Eq. (3.66) involves speech samples $s_n(m)$ defined from $m = -p$ up to $m = N-1-p$ and, when $k = 0$, $s_n(m+i-k)$ involves samples from 0 to $N-1$. Hence the range of speech required for the complete computation is from $s_n(-p)$ to $s_n(N-1)$—that is, the samples $s_n(-p), s_n(-p+1), \ldots, s_n(-1)$, *outside* the error minimization interval, are required.

Using the extended speech interval to define the covariance values, $\phi_n(i,k)$, the matrix form of the LPC analysis equations becomes

$$\begin{bmatrix} \phi_n(1,1) & \phi_n(1,2) & \phi_n(1,3) & \cdots & \phi_n(1,p) \\ \phi_n(2,1) & \phi_n(2,2) & \phi_n(2,3) & \cdots & \phi_n(2,p) \\ \phi_n(3,1) & \phi_n(3,2) & \phi_n(3,3) & \cdots & \phi_n(3,p) \\ \vdots & \vdots & \vdots & & \vdots \\ \phi_n(p,1) & \phi_n(p,2) & \phi_n(p,3) & \cdots & \phi_n(p,p) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \\ \vdots \\ \hat{a}_p \end{bmatrix} = \begin{bmatrix} \phi_n(1,0) \\ \phi_n(2,0) \\ \phi_n(3,0) \\ \vdots \\ \phi_n(p,0) \end{bmatrix}. \qquad (3.67)$$

The resulting covariance matrix is symmetric (since $\phi_n(i,k) = \phi_n(k,i)$) but not Toeplitz, and can be solved efficiently by a set of techniques called the Cholesky decomposition method [6]. Since the full covariance form of the LPC analysis equations is generally *not* used for speech-recognition systems, we will not discuss this method further but instead will concentrate on the autocorrelation method of LPC analysis for the remainder of this chapter.

3.3.5 Review Exercise

Exercise 3.4
Given an LPC system of the form

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}}$$

how would you evaluate $H(e^{j\omega})$ using FFT techniques?

Solution 3.4
Define the LPC polynomial as

$$A(z) = \frac{G}{H(z)} = 1 - \sum_{k=1}^{p} a_k z^{-k}.$$

This finite polynomial in z has a time domain response, $f(n)$, which is an FIR sequence of the form

$$f(n) = \begin{cases} 1, & n = 0 \\ -a_n, & 1 \le n \le p \\ 0, & \text{otherwise} \end{cases}.$$

Hence we can evaluate $A(e^{j\omega})$, using FFTs, by supplementing $f(n)$ with sufficient zero-valued samples to form an N-point sequence (e.g., $N = 256$, or $N = 512$), and taking the DFT of that sequence giving $A(e^{j\frac{2\pi}{N}k})$, $0 \le k \le N-1$, i.e. $A(e^{j\omega})|_{\omega=\frac{2\pi k}{N}}$. We can then evaluate $H(e^{j\omega})$ for $\omega = \frac{2\pi k}{N}$, $k = 0, 1, \ldots, N-1$ as $G/A(e^{j\omega})|_{\omega=\frac{2\pi k}{N}}$.

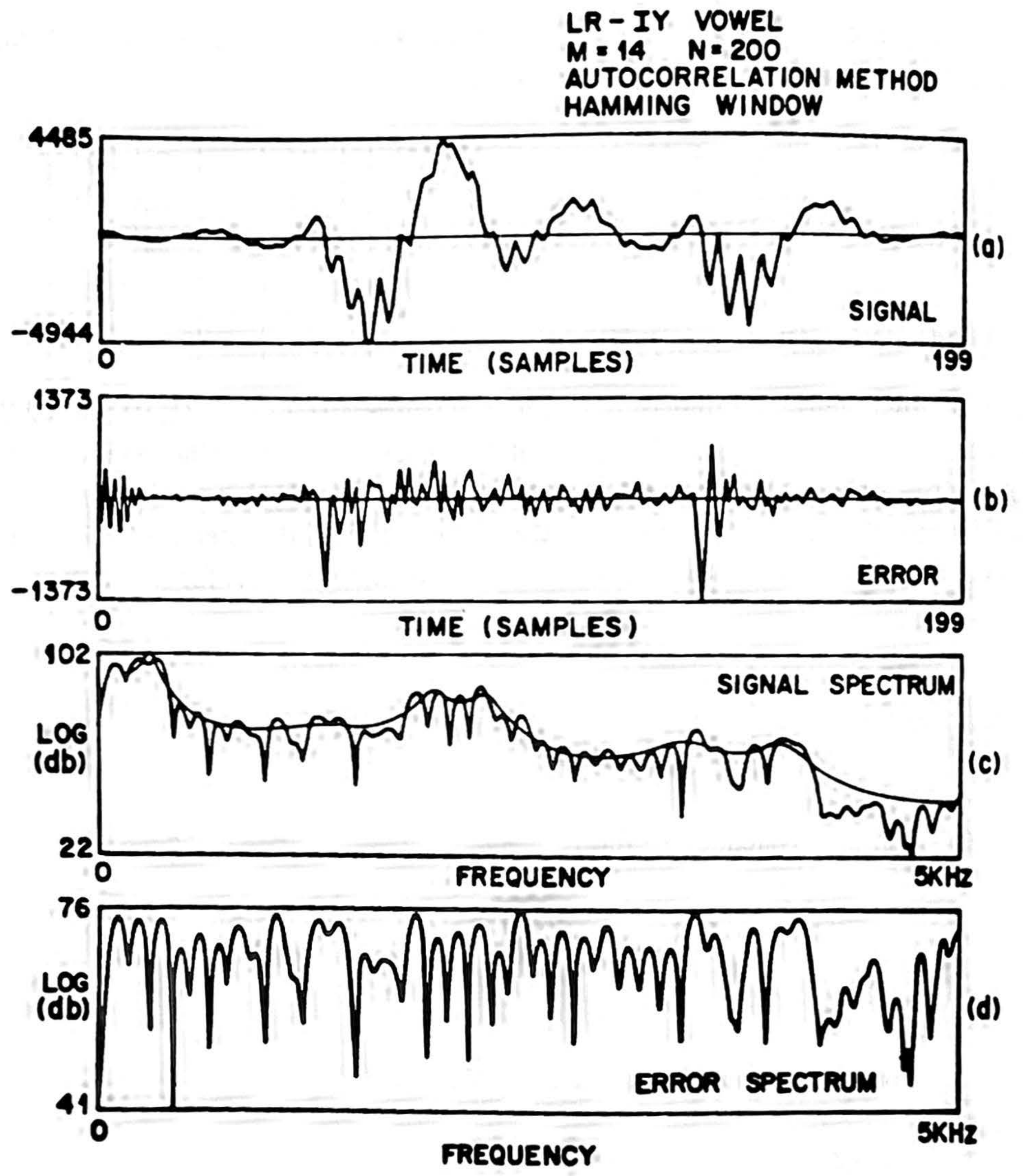

Figure 3.32 Typical signals and spectra for LPC autocorrelation method for a segment of speech spoken by a male speaker (after Rabiner et al. [8]).

3.3.6 Examples of LPC Analysis

To illustrate some of the properties of the signals involved in LPC analysis, Figures 3.32 and 3.33 show series of waveform and spectral plots of the windowed speech signal (part a), the prediction error signal (part b), the signal log spectrum (FFT-based) fitted by an LPC log spectrum (as defined from Exercises 3.4, part c), and the log spectrum of the prediction error signal (part d). The results in Figure 3.32 are for the IY vowel spoken by a male speaker; those of Figure 3.33 are for the AH vowel spoken by a female speaker. For both examples the speech sample size was 20 msec (200 samples at a 10-kHz rate) and the analysis was performed using a $p = 14^{\text{th}}$ order LPC analysis. For the male speaker, about two periods of signal were used in the analysis frame. The error signal is a factor of almost 4 smaller in magnitude than the speech signal and has a much flatter spectral trend than

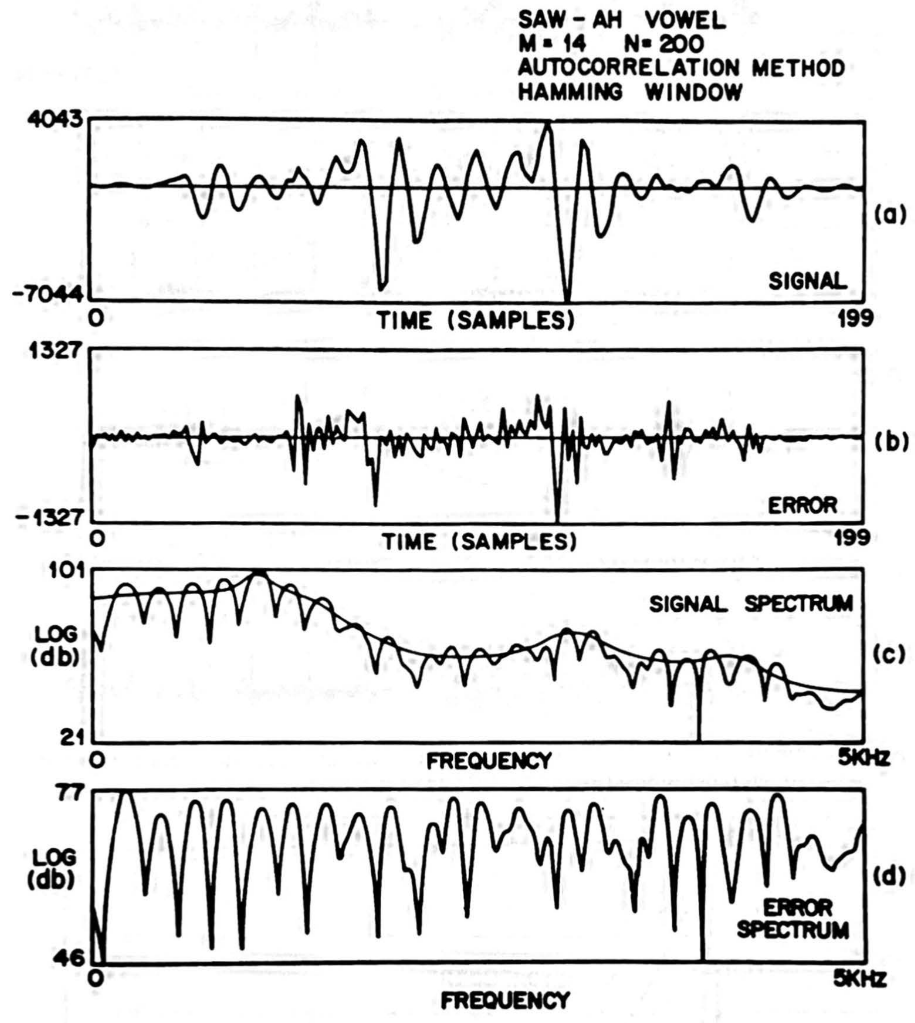

Figure 3.33 Typical signals and spectra for LPC autocorrelation method for a segment of speech spoken by a female speaker (after Rabiner et al. [8]).

the speech signal. This is the important "whitening" characteristics of the LPC analysis whereby the error signal spectrum is approximately a flat spectrum signal representing the source characteristics rather than those of the vocal tract. Similar behavior is seen in the plots of the vowel from the female talker. Finally, it can be seen that fairly close matches exist between the peaks of the FFT-based signal spectrum and the LPC spectrum.

Figures 3.34–3.36 illustrate some additional properties of the LPC analysis method. Figure 3.34 shows a series of sections of the waveforms (differentiated for preemphasis) for several vowels, and the corresponding prediction error signals. (The prediction error signals have been scaled up in value so as to make their amplitudes comparable to those of the signal; hence, gains of about 4 to 1 were used.) The high-frequency nature of the prediction error signal is seen in all these examples. What can also be seen is that, for many cases, the prediction error signal exhibits sharp pulses at intervals corresponding to the pitch periods of these vowels. This characteristic behavior has been used as the basis

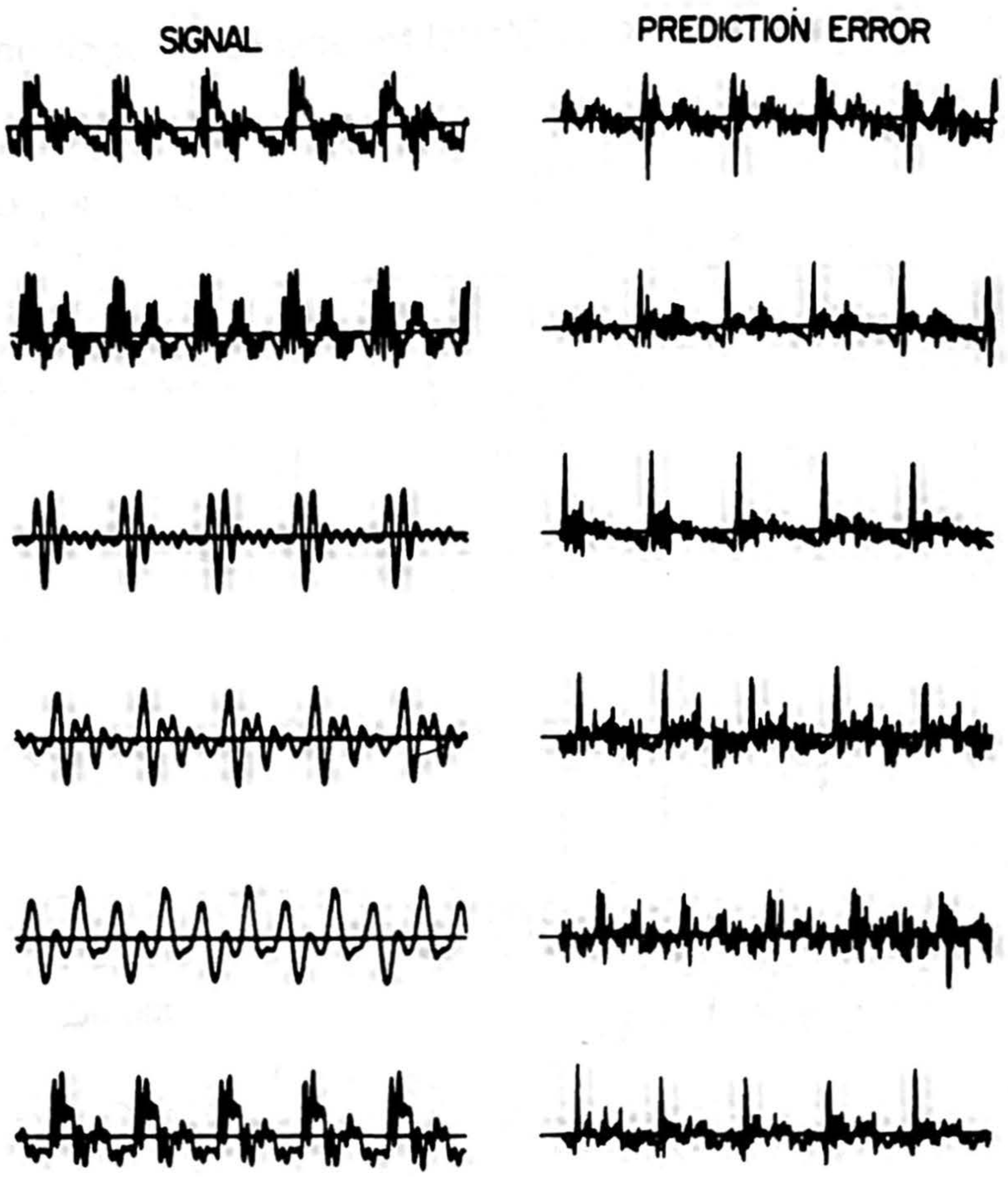

Figure 3.34 Examples of signal (differentiated) and prediction error for several vowels (after Strube [9]).

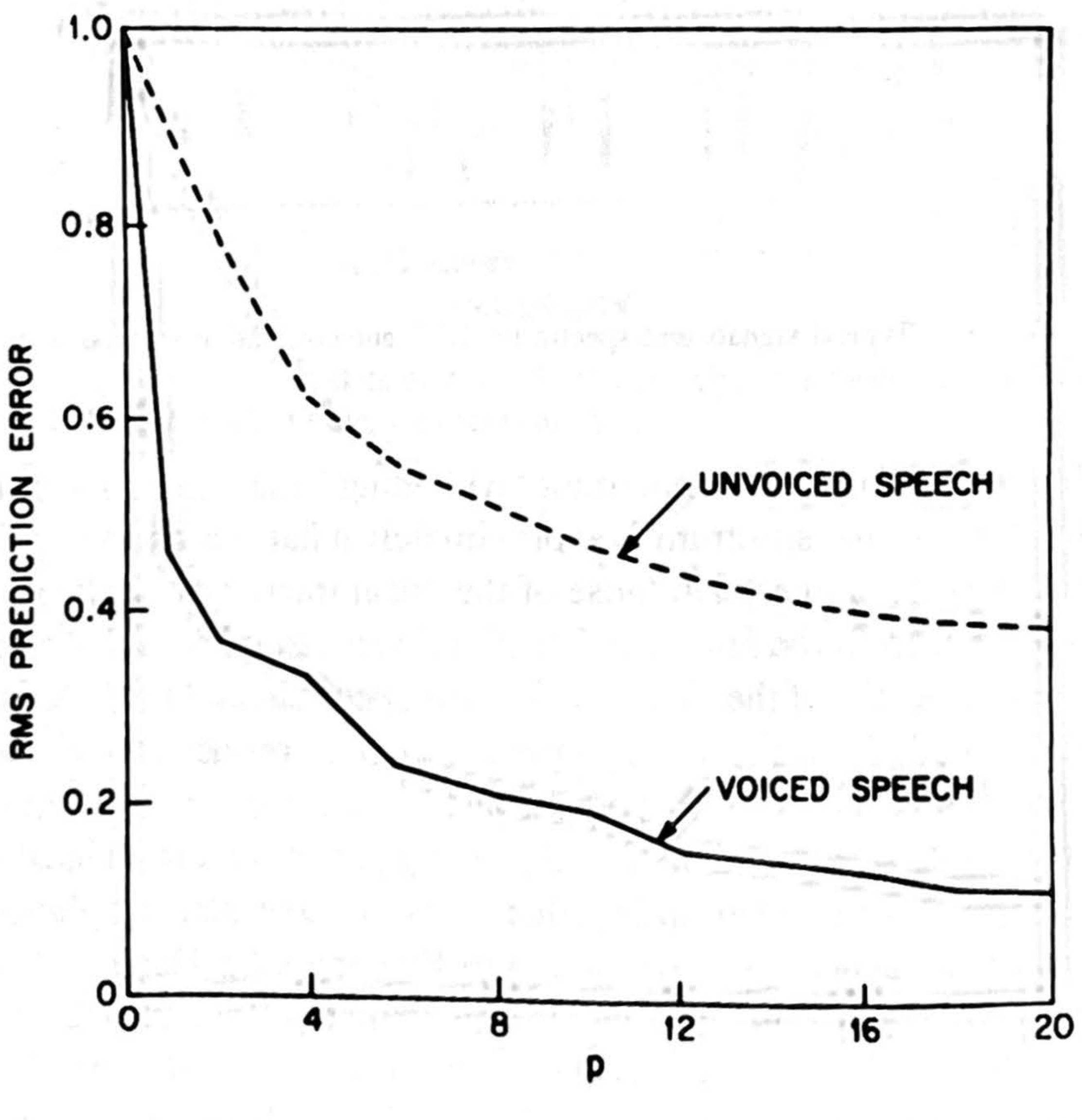

Figure 3.35 Variation of the RMS prediction error with the number of predictor coefficients, p (after Atal and Hanauer [10]).

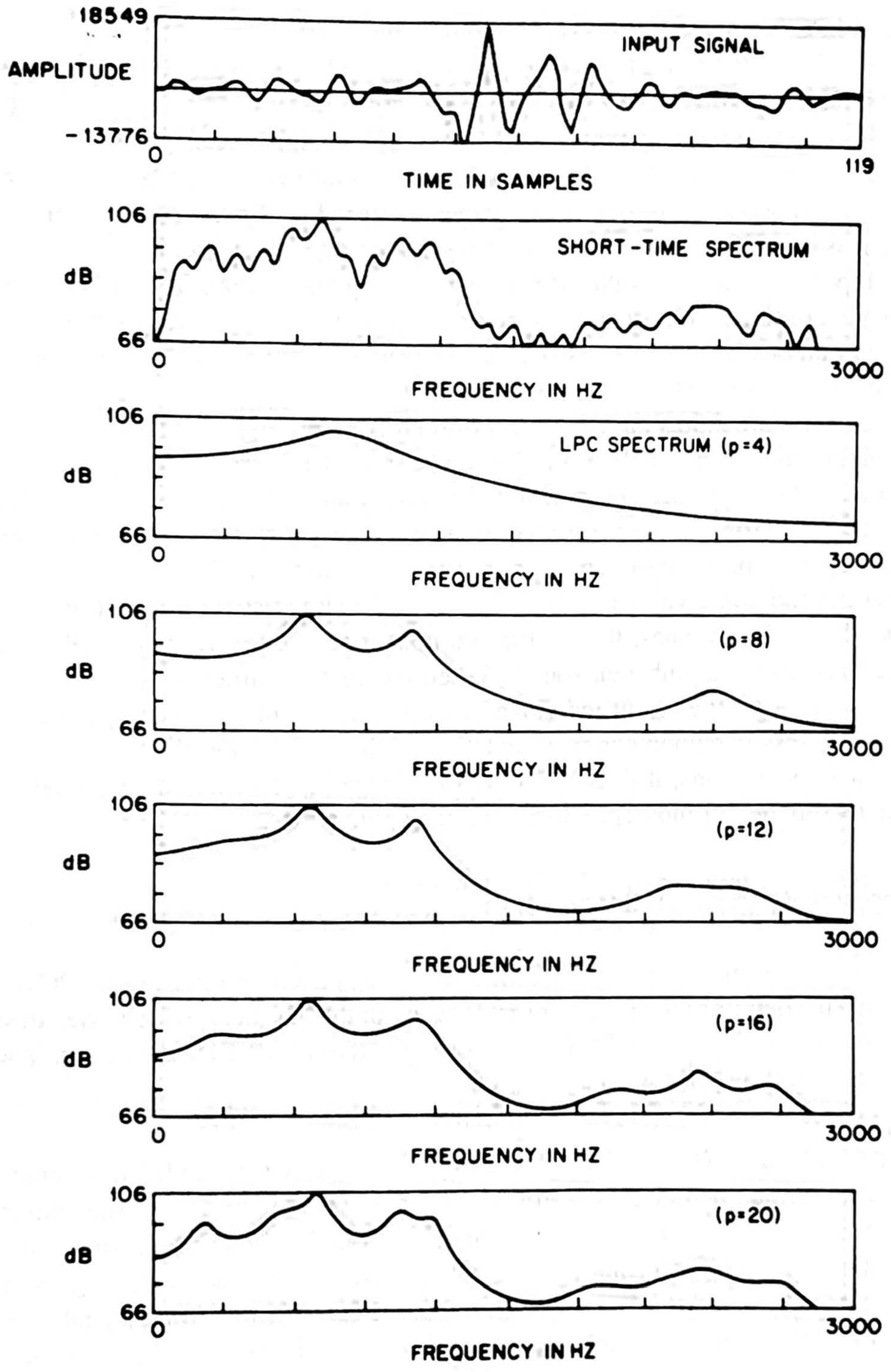

Figure 3.36 Spectra for a vowel sound for several values of predictor order, p.

for several LPC-based pitch period estimation methods.

Figure 3.35 shows the effect of LPC prediction order, p, on the RMS prediction error, E_n, for both sections of voiced speech (solid curve) and unvoiced speech (dashed curve). The prediction error in the curves is normalized by the signal energy such that at $p = 0$ (i.e., no prediction) $E_n = R_n(0)$. A sharp decrease in normalized prediction error occurs for small values of p (e.g., 1–4); however, beyond this value of p the normalized prediction error decreases much more slowly. It is also seen that the normalized prediction error for unvoiced speech, for a given value of p, is significantly higher than for voiced speech. The interpretation of this result is that unvoiced speech is less linearly predictable than voiced speech, a result one would anticipate based on our understanding of the speech-production mechanisms.

Finally, Figure 3.36 shows the effect of prediction order, p, on the all-pole spectrum and its ability to match details in the FFT spectrum of the speech segment. Shown in this figure are the input speech segment, the Fourier transform of that segment, and linear predictive spectra for values of p from 4 to 20. It is clear that as p increases, more of the detailed properties of the signal spectrum are preserved in the LPC spectrum. It is equally clear that beyond some value of p, the details of the signal spectrum that are preserved are generally irrelevant ones; that is, they do not reflect the relevant spectral resonances or antiresonances of the inherent sound. When the analysis order, p, becomes large, the LPC spectrum often tries to fit individual pitch harmonics of the speech signal, thereby resulting in a less parsimonious representation of the sound. On the basis of extensive experimental evaluations, it is generally acknowledged that values of p on the order of 8–10 are reasonable for most speech-recognition applications.

3.3.7 LPC Processor for Speech Recognition

At this point, rather than spending more time discussing general properties of LPC methods, we describe the details of the LPC front-end processor that has been widely used in speech-recognition systems. Figure 3.37 shows a block diagram of the LPC processor. The basic steps in the processing include the following:

1. **Preemphasis**—The digitized speech signal, $s(n)$, is put through a low-order digital system (typically a first-order FIR filter), to spectrally flatten the signal and to make it less susceptible to finite precision effects later in the signal processing. The digital system used in the preemphasizer is either fixed or slowly adaptive (e.g., to average transmission conditions, noise backgrounds, or even to average signal spectrum). Perhaps the most widely used preemphasis network is the fixed first-order system:

$$H(z) = 1 - \tilde{a}z^{-1}, \qquad 0.9 \leq a \leq 1.0. \tag{3.68}$$

In this case, the output of the preemphasis network, $\tilde{s}(n)$, is related to the input to the network, $s(n)$, by the difference equation

$$\tilde{s}(n) = s(n) - \tilde{a}s(n-1). \tag{3.69}$$

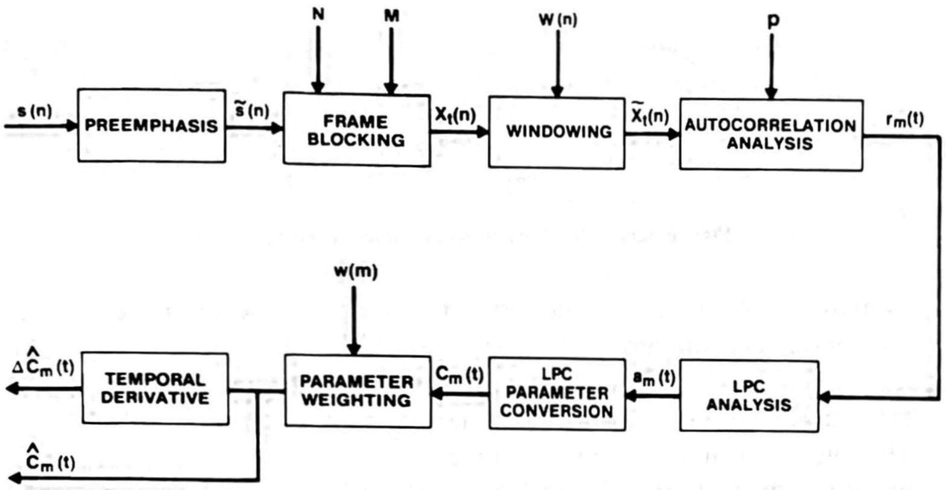

Figure 3.37 Block diagram of LPC processor for speech recognition.

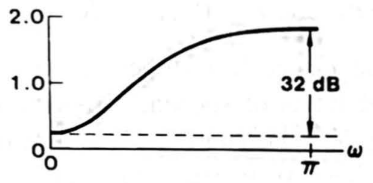

Figure 3.38 Magnitude spectrum of LPC preemphasis network for $\tilde{a} = 0.95$.

The most common value for $\tilde{a}$ is around 0.95. (For fixed-point implementations a value of $\tilde{a} = 15/16 = 0.9375$ is often used.) A simple example of a first-order *adaptive* preemphasizer is the transfer function

$$H(z) = 1 - \tilde{a}_n z^{-1}, \tag{3.70}$$

where $\tilde{a}_n$ changes with time (n) according to the chosen adaptation criterion. One possibility is to choose $\tilde{a}_n = r_n(1)/r_n(0)$. Figure 3.38 shows the magnitude characteristics of $H(e^{j\omega})$ for the value $\tilde{a} = 0.95$. It can be seen that at $\omega = \pi$ (half the sampling rate) there is a 32 dB boost in the magnitude over that at $\omega = 0$.

2. **Frame Blocking**—In this step the preemphasized speech signal, $\tilde{s}(n)$, is blocked into frames of N samples, with adjacent frames being separated by M samples. Figure 3.39 illustrates the blocking into frames for the case in which $M = (1/3)N$. The first illustrated frame consists of the first N speech samples. The second frame begins M samples after the first frame, and overlaps it by $N - M$ samples. Similarly, the third frame begins $2M$ samples after the first frame (or M samples after the second frame) and overlaps it by $N - 2M$ samples. This process continues until all the speech is accounted for within one or more frames. It is easy to see that if $M \leq N$, then adjacent frames overlap (as in Figure 3.39), and the resulting LPC spectral estimates will be correlated from frame to frame; if $M \ll N$, then LPC spectral estimates from frame to frame will be quite smooth. On the other hand, if $M > N$, there will be no

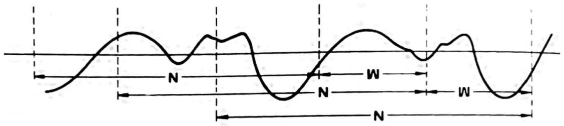

Figure 3.39 Blocking of speech into overlapping frames.

overlap between adjacent frames; in fact, some of the speech signal will be totally lost (i.e., never appear in any analysis frame), and the correlation between the resulting LPC spectral estimates of adjacent frames will contain a noisy component whose magnitude increases as M increases (i.e., as more speech is omitted from analysis). This situation is intolerable in any practical LPC analysis for speech recognition. If we denote the ℓ^{th} frame of speech by $x_\ell(n)$, and there are L frames within the entire speech signal, then

$$x_\ell(n) = \tilde{s}(M\ell + n), \qquad n = 0, 1, \ldots, N-1, \quad \ell = 0, 1, \ldots, L-1. \tag{3.71}$$

That is, the first frame of speech, $x_0(n)$, encompasses speech samples $\tilde{s}(0), \tilde{s}(1), \ldots, \tilde{s}(N-1)$, the second frame of speech, $x_1(n)$, encompasses speech samples $\tilde{s}(M), \tilde{s}(M+1), \ldots, \tilde{s}(M+N-1)$, and the L^{th} frame of speech, $x_{L-1}(n)$, encompasses speech samples $\tilde{s}(M(L-1)), \tilde{s}(M(L-1)+1), \ldots, \tilde{s}(M(L-1)+N-1)$. Typical values for N and M are 300 and 100 when the sampling rate of the speech is 6.67 kHz. These correspond to 45-msec frames, separated by 15 msec, or a 66.7-Hz frame rate.

3. **Windowing**—The next step in the processing is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. The concept here is identical to the one discussed with regard to the frequency domain interpretation of the short-time spectrum in Section 3.2—namely, to use the window to taper the signal to zero at the beginning and end of each frame. If we define the window as $w(n)$, $0 \le n \le N-1$, then the result of windowing is the signal

$$\tilde{x}_\ell(n) = x_\ell(n)w(n), \qquad 0 \le n \le N-1. \tag{3.72}$$

A "typical" window used for the autocorrelation method of LPC (the method most widely used for recognition systems) is the Hamming window, which has the form

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \le n \le N-1. \tag{3.73}$$

4. **Autocorrelation Analysis**—Each frame of windowed signal is next autocorrelated to give

$$r_\ell(m) = \sum_{n=0}^{N-1-m} \tilde{x}_\ell(n)\tilde{x}_\ell(n+m), \qquad m = 0, 1, \ldots, p, \tag{3.74}$$

where the highest autocorrelation value, p, is the order of the LPC analysis. Typically, values of p from 8 to 16 have been used, with $p = 8$ being the value used for most systems to be described in this book. A side benefit of the autocorrelation analysis is that the zeroth autocorrelation, $R_\ell(0)$, is the energy of the ℓ^{th} frame. The frame energy is an important parameter for speech-detection systems and will be discussed further in the next chapter.

5. **LPC Analysis**—The next processing step is the LPC analysis, which converts each frame of $p + 1$ autocorrelations into an "LPC parameter set," in which the set might be the LPC coefficients, the reflection (or PARCOR) coefficients, the log area ratio coefficients, the cepstral coefficients, or any desired transformation of the above sets. The formal method for converting from autocorrelation coefficients to an LPC parameter set (for the LPC autocorrelation method) is known as Durbin's method and can formally be given as the following algorithm (for convenience, we will omit the subscript ℓ on $r_\ell(m)$):

$$E^{(0)} = r(0) \tag{3.75}$$

$$k_i = \left\{ r(i) - \sum_{j=1}^{L-1} \alpha_j^{(i-1)} r(|i-j|) \right\} \Big/ E^{(i-1)}, \qquad 1 \le i \le p \tag{3.76}$$

$$\alpha_i^{(i)} = k_i \tag{3.77}$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \tag{3.78}$$

$$E^{(i)} = (1 - k_i^2) E^{(i-1)}, \tag{3.79}$$

where the summation in Eq. (3.76) is omitted for $i = 1$. The set of equations (3.75–3.79) are solved recursively for $i = 1, 2, \ldots, p$, and the final solution is given as

$$a_m = \text{LPC coefficients} = \alpha_m^{(p)}, \qquad 1 \le m \le p \tag{3.80}$$

$$k_m = \text{PARCOR coefficients} \tag{3.81}$$

$$g_m = \text{log area ratio coefficients} = \log\left(\frac{1-k_m}{1+k_m}\right). \tag{3.82}$$

6. **LPC Parameter Conversion to Cepstral Coefficients**—A very important LPC parameter set, which can be derived directly from the LPC coefficient set, is the LPC cepstral coefficients, $c(m)$. The recursion used is

$$c_0 = ln\sigma^2 \tag{3.83a}$$

$$c_m = a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k}, \qquad 1 \le m \le p \tag{3.83b}$$

$$c_m = \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k}, \qquad m > p, \tag{3.83c}$$

where σ^2 is the gain term in the LPC model. The cepstral coefficients, which are the coefficients of the Fourier transform representation of the log magnitude spectrum, have been shown to be a more robust, reliable feature set for speech recognition than the LPC coefficients, the PARCOR coefficients, or the log area ratio coefficients. Generally, a cepstral representation with $Q > p$ coefficients is used, where $Q \simeq \left(\frac{3}{2}\right) p$.

7. **Parameter Weighting**—Because of the sensitivity of the low-order cepstral coefficients to overall spectral slope and the sensitivity of the high-order cepstral coefficients to noise (and other forms of noiselike variability), it has become a standard technique to weight the cepstral coefficients by a tapered window so as to minimize these sensitivities. A formal way of justifying the use of a cepstral window is to consider the Fourier representation of the log magnitude spectrum and the differentiated (in frequency) log magnitude spectrum, such that

$$\log \left|S(e^{j\omega})\right| = \sum_{m=-\infty}^{\infty} c_m e^{-j\omega m} \tag{3.84}$$

$$\frac{\partial}{\partial\omega}\left[\log \left|S(e^{j\omega})\right|\right] = \sum_{m=-\infty}^{\infty} (-jm) c_m e^{-j\omega m}. \tag{3.85}$$

The differential log magnitude spectrum has the property that any fixed spectral slope in the log magnitude spectrum becomes a constant; furthermore, any prominent spectral peak in the log magnitude spectrum (e.g., the formants) is well preserved as a peak in the differentiated log magnitude spectrum. Hence, by considering the multiplication by $(-jm)$ in the representation of the differentiated log magnitude spectrum as a form of weighting, we get

$$\frac{\partial}{\partial\omega}\left[\log \left|S(e^{j\omega})\right|\right] = \sum_{m=-\infty}^{\infty} \hat{c}_m e^{-j\omega m}, \tag{3.86}$$

where

$$\hat{c}_m = c_m(-jm). \tag{3.87}$$

To achieve the robustness for large values of m (i.e., low weight near $m = Q$) and to truncate the infinite computation of Eq. (3.86), we must consider a more general weighting of the form

$$\hat{c}_m = w_m c_m, \qquad 1 \le m \le Q, \tag{3.88}$$

where an appropriate weighting is the bandpass lifter (filter in the cepstral domain)

$$w_m = \left[1 + \frac{Q}{2}\sin\left(\frac{\pi m}{Q}\right)\right], \qquad 1 \le m \le Q. \tag{3.89}$$

This weighting function truncates the computation and de-emphasizes c_m around $m = 1$ and around $m = Q$.

8. **Temporal Cepstral Derivative**—The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given analysis frame [11]. An improved representation can be obtained by extending the analysis to include information about the temporal cepstral derivative (both first and second derivatives have been investigated and found to improve the performance of speech-recognition systems). To introduce temporal order into the

cepstral representation, we denote the m^{th} cepstral coefficient at time t by $c_m(t)$. Of course, in practice, the sampling time t refers to the analysis frame rather than an arbitrary time instance. The way in which the cepstral time derivative is approximated is as follows: The time derivative of the log magnitude spectrum has a Fourier series representation of the form

$$\frac{\partial}{\partial t}\left[\log\left|S(e^{j\omega},t)\right|\right] = \sum_{m=-\infty}^{\infty} \frac{\partial c_m(t)}{\partial t} e^{-j\omega m}. \tag{3.90}$$

Hence, the temporal cepstral derivative must be determined in an appropriate manner. It is well known that since $c_m(t)$ is a discrete time representation (where t is the frame index), simply using a first- or second-order difference is inappropriate to approximate the derivative (it is very noisy). Hence, a reasonable compromise is to approximate $\partial c_m(t)/\partial t$ by an orthogonal polynomial fit (a least-squares estimate of the derivative) over a finite length window; that is,

$$\frac{\partial c_m(t)}{\partial t} = \Delta c_m(t) \approx \mu \sum_{k=-K}^{K} k c_m(t+k), \tag{3.91}$$

where μ is an appropriate normalization constant and $(2K+1)$ is the number of frames over which the computation is performed. Typically, a value of $K = 3$ has been found appropriate for computation of the first-order temporal derivative. Based on the computations described above, for each frame t, the result of the LPC analysis is a vector of Q weighted cepstral coefficients and an appended vector of Q cepstral time derivatives; that is,

$$\mathbf{o}_t' = (\hat{c}_1(t), \hat{c}_2(t), \ldots, \hat{c}_Q(t), \Delta c_1(t), \Delta c_2(t), \ldots, \Delta c_Q(t)), \tag{3.92}$$

where $\mathbf{o}_t$ is a vector with $2Q$ components and $'$ denotes matrix transpose. Similarly, if second-order temporal derivatives are computed (giving $\Delta^2 c_m(t)$), these are appended to $\mathbf{o}_t$ giving a vector with $3Q$ components (see Section 4.6 for more details).

3.3.8 Review Exercises

Exercise 3.5

To illustrate LPC analysis via the autocorrelation method, consider a predictor of order $p = 2$. Assume an autocorrelation vector with components $R = (r(0), r(1), r(2))$. Use the Durbin method, described in the previous section, to solve for the LPC coefficients a_1 and a_2 in terms of the Rs. Check your answer by solving the matrix equation

$$\begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \end{bmatrix}$$

using simple matrix algebra.

Solution 3.5

Using the Durbin method, we get the following steps:

$$E^{(0)} = r(0)$$

$$\begin{aligned} k_1 &= r(1)/r(0) \\ \alpha_1^{(1)} &= r(1)/r(0) \\ E^{(1)} &= (r^2(0) - r^2(1))/r(0) \end{aligned}$$

$$\begin{aligned} k_2 &= (r(2)r(0) - r^2(1))/(r^2(0) - r^2(1)) \\ \alpha_2^{(2)} &= (r(2)r(0) - r^2(1))/(r^2(0) - r^2(1)) \\ \alpha_1^{(2)} &= (r(1)r(0) - r(1)r(2))/(r^2(0) - r^2(1)) \end{aligned}$$

$$\begin{aligned} a_1 &= \alpha_1^{(2)} \\ a_2 &= \alpha_2^{(2)} \end{aligned}$$

Using matrix algebra we get

$$\begin{aligned} a_1 r(0) + a_2 r(1) &= r(1) \\ a_1 r(1) + a_2 r(0) &= r(2) \end{aligned}$$

Solving directly for a_1 and a_2 we get

$$\begin{aligned} a_1 &= (r(1)r(0) - r(1)r(2))/(r^2(0) - r^2(1)) \\ a_2 &= (r(2)r(0) - r^2(1))/(r^2(0) - r^2(1)) \end{aligned}$$

which is the same result as obtained via the Durbin method.

Exercise 3.6

Consider two (windowed) speech sequences $x(n)$ and $\hat{x}(n)$ both defined for $0 \le n \le N-1$. (Outside this region both sequences are defined to be 0.) We perform an LPC analysis (using the autocorrelation method) on each frame. Thus, from the autocorrelation sequences

$$r(k) = \sum_{n=0}^{N-1-k} x(n)x(n+k), \qquad 0 \le k \le p$$

$$\hat{r}(k) = \sum_{n=0}^{N-1-k} \hat{x}(n)\hat{x}(n+k), \qquad 0 \le k \le p$$

we solve for the predictor parameter $\mathbf{a}' = (a_0, a_1, \ldots, a_p)$ and $\hat{\mathbf{a}}' = (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_p)$ ($a_0 = \hat{a}_0 = -1$) where $'$ denotes matrix transpose.

1. Show that the prediction error (residual), defined as

$$E^{(p)} = \sum_{n=0}^{N-1+p} e^2(n) = \sum_{n=0}^{N-1-p} \left[-\sum_{i=0}^{p} a_i x(n-i) \right]^2$$

can be written in the form

$$E^{(p)} = \mathbf{a}'\mathbf{R}_x\mathbf{a},$$

where $\mathbf{R}_x$ is a $(p+1)$ by $(p+1)$ matrix. Determine $\mathbf{R}_x$.

2. Consider passing the sequence $\hat{x}(n)$ through the inverse LPC system with LPC coefficients **a**, to give the error signal $\tilde{e}(n)$, defined as

$$\tilde{e}(n) = -\sum_{i=0}^{p} a_i \hat{x}(n-i).$$

Show that the mean-squared error, $\tilde{E}^{(p)}$, defined by

$$\tilde{E}^{(p)} = \sum_{n=0}^{N-1+p} [\tilde{e}(n)]^2$$

can be written in the form

$$\tilde{E}^{(p)} = \mathbf{a}'\mathbf{R}_{\hat{x}}\mathbf{a},$$

where $\mathbf{R}_{\hat{x}}$ is a $(p+1)$ by $(p+1)$ matrix. Determine $\mathbf{R}_{\hat{x}}$.

3. If we form the ratio

$$D = \frac{\tilde{E}^{(p)}}{E^{(p)}}$$

what can be said about the range of D?

(This exercise gives an initial appreciation of the concept of distortion measures. Chapter 4 discusses this topic in great detail.)

Solution 3.6

1. Since

$$e(n) = -\sum_{i=0}^{p} a_i x(n-i)$$

$$E^{(p)} = \sum_{n=0}^{N-1+p} e^2(n) + \sum_{n=0}^{N-1+p}\left[-\sum_{i=0}^{p} a_i x(n-i)\right]\left[-\sum_{j=0}^{p} a_j x(n-j)\right]$$

$$= \sum_{i=0}^{p} a_i \sum_{j=0}^{p} a_j \sum_{n=0}^{N-1+p} x(n-i)x(n-j).$$

But

$$\sum_{n=0}^{N-1+p} x(n-i)x(n-j) = \sum_{n=0}^{N-1+p} x(n)x(n-j+i) = r(|i-j|).$$

Thus

$$E^{(p)} = \sum_{i=0}^{p} a_i \sum_{j=0}^{p} a_j r(|i-j|) = \mathbf{a}'\mathbf{R}_x\mathbf{a},$$

where

$$\mathbf{R}_x = \begin{bmatrix} r(0) & r(1) & \cdots & r(p) \\ r(1) & r(0) & \cdots & r(p-1) \\ \vdots & \vdots & & \vdots \\ r(p) & r(p-1) & \cdots & r(0) \end{bmatrix}$$

2. $\tilde{e}(n) = -\sum_{i=0}^{p} a_i \hat{x}(n-i)$
   Repeating the derivation of part 1, we get

$$\tilde{E}^{(p)} = \sum_{i=0}^{p} a_i \sum_{j=0}^{p} a_j \hat{r}(|i-j|) = \mathbf{a}'\mathbf{R}_{\hat{x}}\mathbf{a},$$

   where

$$\mathbf{R}_{\hat{x}} = \begin{bmatrix} \hat{r}(0) & \hat{r}(1) & \cdots & \hat{r}(p) \\ \hat{r}(1) & \hat{r}(0) & \cdots & \hat{r}(p-1) \\ \vdots & \vdots & & \vdots \\ \hat{r}(p) & \hat{r}(p-1) & \cdots & \hat{r}(0) \end{bmatrix}$$

3. $D = \dfrac{\tilde{E}^{(p)}}{E^{(p)}} = \dfrac{\mathbf{a}'\mathbf{R}_x\mathbf{a}}{\mathbf{a}'\mathbf{R}_{\hat{x}}\mathbf{a}}$ Since D is a ratio of prediction residuals, and since $E^{(p)}$ is the *minimum* prediction residual for LPC system $\mathbf{a}$, then $\tilde{E}^{(p)}$ must be greater than (or equal to) $E^{(p)}$. Therefore

$$D \geq 1.0$$

Exercise 3.7

A proposed measure of spectral distance between two frames of speech represented by LPC coefficient sets $\mathbf{a}$ and $\hat{\mathbf{a}}$, and augmented autocorrelation matrices $\mathbf{R}_x$ and $\mathbf{R}_{\hat{x}}$ (see Exercise 3.6) is:

$$D(\mathbf{a}, \hat{\mathbf{a}}) = \frac{\mathbf{a}'\mathbf{R}_{\hat{x}}\mathbf{a}}{\hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}}$$

1. Show that the distance function $D(\mathbf{a}, \hat{\mathbf{a}})$ can be written in the computationally efficient form

$$D(\mathbf{a}, \hat{\mathbf{a}}) = \left[\frac{(r_a(0)\hat{r}(0) + 2\sum_{i=1}^{p} r_a(i)\hat{r}(i))}{\hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}} \right],$$

   where $r_a(i)$ is the autocorrelation of the $\mathbf{a}$ array, i.e.,

$$r_a(i) = \sum_{j=0}^{p-i} a_j a_{j+i}, \qquad 0 \leq i \leq p.$$

2. Assume that the quantities (i.e., vectors, matrices, scalars) $\mathbf{a}, \hat{\mathbf{a}}, \mathbf{R}_x, \mathbf{R}_{\hat{x}}, \hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}, \mathbf{R}_{\hat{x}}$ and $\mathbf{R}_a$ are precomputed; that is, they are available at the time the distance calculation is required. Contrast the computation required to evaluate $D(\mathbf{a}, \hat{\mathbf{a}})$ using both expressions for D given in this exercise.

Solution 3.7

We have that

$$D(\mathbf{a}, \hat{\mathbf{a}}) = \frac{\mathbf{a}'\mathbf{R}_{\hat{x}}, \mathbf{a}}{\hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}} = \frac{\tilde{E}^{(p)}}{\hat{E}^{(p)}}.$$

From Exercise 3.6 we get

$$\tilde{E}^{(p)} = \sum_{i=0}^{p} a_i \sum_{j=0}^{p} a_j \hat{r}(|j-i|).$$

Letting $k = j - i$ ($j = k + i$) we get

$$\tilde{E}^{(p)} = \sum_{i=0}^{p} a_i \sum_{k=-i}^{p-i} a_{k+i}\hat{r}(|k|).$$

By rearranging the summations on i and k and by recognizing that $a_\ell = 0$, $\ell < 0$ and $a_\ell = 0$, $\ell > p$, we can complete the square by summing on k from $-p$ (the smallest value of k) to $+p$ (the largest value of k), giving

$$\tilde{E}^{(p)} = \sum_{k=-p}^{p} \left[\sum_{i=0}^{p-|k|} a_i a_{k+i} \right] \hat{r}(|k|).$$

The inner summation is defined as $r_a(k)$, hence

$$\tilde{E}^{(p)} = \sum_{k=-p}^{p} r_a(k)\hat{r}(|k|).$$

Since $r_a(k) = r_a(-k)$ and $\hat{r}(k) = \hat{r}(-k)$ we can write $\tilde{E}^{(p)}$ as

$$\tilde{E}^{(p)} = r_a(0)\hat{r}(0) + 2\sum_{k=1}^{p} r_a(k)\hat{r}(k).$$

2 Since all the individual quantities are precomputed, to evaluate D as a ratio of residuals; that is,

$$D = \frac{\mathbf{a}'\mathbf{R}_{\hat{x}}\mathbf{a}}{\hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}}$$

requires

a. $(p+1) \times (p+2)$ multiplies and adds to multiply $\mathbf{a}'$ by $\mathbf{R}_{\hat{x}}$ and then multiply the result by $\mathbf{a}$.

b. 1 divide to give D since $\hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}$ is a precomputed scalar.

For the alternative method of evaluating D, as discussed in part 1 of this exercise, we require:

a. $(p+1)$ multiplies and adds to give the product $\hat{r}(k)r_a(k)$ for $1 \le k \le p$ and to give $\hat{r}(0)r_a(0)$.

b. 1 divide to give D since $\hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}$ is a precomputed scalar.

Thus, neglecting the divide, the alternative computation of D requires a factor of $(p+2)$ less computation and therefore is significantly more efficient than direct computation of the ratio of prediction residuals.

3.3.9 Typical LPC Analysis Parameters

The computation of the LPC analysis system of Figure 3.37 is specified by a number of variable parameters, including

N number of samples in the analysis frame

M number of samples shift between frames

p LPC analysis order

Q dimension of LPC derived cepstral vector

K number of frames over which cepstral time derivatives are computed.

Although each of these parameters can be varied over a wide range of values, the following table gives typical values for analysis systems at three different sampling rates (6.67 kHz, 8 kHz, 10 kHz).

Typical Values of LPC Analysis Parameters for Speech-Recognition Systems

parameter	$F_s = 6.67$ kHz		$F_s = 8$ kHz		$F_s = 10$ kHz	
N	300	(45 msec)	240	(30 msec)	300	(30 msec)
M	100	(15 msec)	80	(10 msec)	100	(10 msec)
p	8		10		10	
Q	12		12		12	
K	3		3		3	

3.4 VECTOR QUANTIZATION

The results of either the filter-bank analysis or the LPC analysis are a series of vectors characteristic of the time-varying spectral characteristics of the speech signal. For convenience, we denote the spectral vectors as $\mathbf{v}_\ell$, $\ell = 1, 2, \ldots, L$, where typically each vector is a p-dimensional vector. If we compare the information rate of the vector representation to that of the raw (uncoded) speech waveform, we see that the spectral analysis has significantly reduced the required information rate. Consider, for example, 10-kHz sampled speech with 16-bit speech amplitudes. A raw signal information rate of 160,000 bps is required to store the speech samples in uncompressed format. For the spectral analysis, consider vectors of dimension $p = 10$ using 100 spectral vectors per second. If we again represent each spectral component to 16-bit precision, the required storage is about $100 \times 10 \times 16$ bps, or 16,000 bps—about a 10-to-1 reduction over the uncompressed signal. Such compressions in storage rate are impressive. Based on the concept of ultimately needing only a single spectral representation for each basic speech unit, it may be possible to further reduce the raw spectral representation of speech to those drawn from a small, finite number of "unique" spectral vectors, each corresponding to one of the basic speech units (i.e., the phonemes). This ideal representation is, of course, impractical, because there is so much variability in the spectral properties of each of the basic speech units. However, the concept of building a codebook of "distinct" analysis vectors, albeit with significantly more code words than the basic set of phonemes, remains an attractive idea and is the basis behind a set of techniques commonly called vector quantization (VQ) methods. Based on this line of reasoning, assume that we require a codebook with about 1024 unique spectral vectors (i.e., about 25 variants for each of the 40 basic speech units). Then to represent an arbitrary spectral vector all we need is a 10-bit number—the index of the codebook vector that best matches the input vector. Assuming a rate of 100 spectral vectors per second, we see that a total bit rate of about 1000 bps is required to represent the spectral vectors of a speech signal. This rate is about $1/16^{\text{th}}$ the rate required by the continuous spectral vectors. Hence the VQ representation is potentially an extremely efficient representation of the spectral information in the speech signal. This is one of the main reasons for the interest in VQ methods.

Before discussing the concepts involved in designing and implementing a practical VQ system, we first discuss the advantages and disadvantages of this type of representation. The key advantages of the VQ representation are

- reduced storage for spectral analysis information. We have already shown that the VQ representation is potentially very efficient. This efficiency can be exploited in a number of ways in practical VQ-based speech-recognition systems.
- reduced computation for determining similarity of spectral analysis vectors. In speech recognition a major component of the computation is the determination of spectral similarity between a pair of vectors. Based on the VQ representation, this spectral similarity computation is often reduced to a table lookup of similarities between pairs of codebook vectors.
- discrete representation of speech sounds. By associating a phonetic label (or possibly a set of phonetic labels or a phonetic class) with each codebook vector, the process of choosing a best codebook vector to represent a given spectral vector becomes equivalent to assigning a phonetic label to each spectral frame of speech. A range of recognition systems exist that exploit these labels so as to recognize speech in an efficient manner.

The disadvantages of the use of a VQ codebook to represent speech spectral vectors are

- an inherent spectral distortion in representing the actual analysis vector. Since there is only a finite number of codebook vectors, the process of choosing the "best" representation of a given spectral vector inherently is equivalent to quantizing the vector and leads, by definition, to a certain level of quantization error. As the size of the codebook increases, the size of the quantization error decreases. However, with any finite codebook there will always be some nonzero level of quantization error.
- the storage required for codebook vectors is often nontrivial. The larger we make the codebook (so as to reduce quantization error), the more storage is required for the codebook entries. For codebook sizes of 1000 or larger, the storage is often nontrivial. Hence an inherent trade-off among quantization error, processing for choosing the codebook vector, and storage of codebook vectors exists, and practical designs balance each of these three factors.

3.4.1 Elements of a Vector Quantization Implementation

To build a VQ codebook and implement a VQ analysis procedure, we need the following:

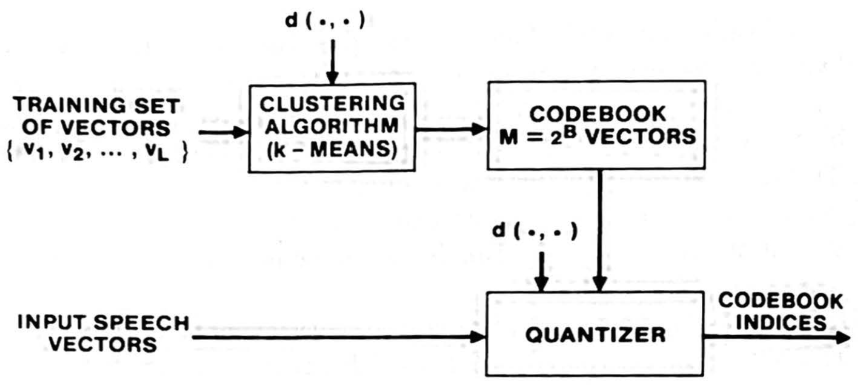

Figure 3.40 Block diagram of the basic VQ training and classification structure.

1. a large set of spectral analysis vectors, $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_L$, which form a training set. The training set is used to create the "optimal" set of codebook vectors for representing the spectral variability observed in the training set. If we denote the size of the VQ codebook as $M = 2^B$ vectors (we call this a B-bit codebook), then we require $L \gg M$ so as to be able to find the best set of M codebook vectors in a robust manner. In practice, it has been found that L should be at least $10M$ in order to train a VQ codebook that works reasonably well.
2. a measure of similarity, or distance, between a pair of spectral analysis vectors so as to be able to cluster the training set vectors as well as to associate or classify arbitrary spectral vectors into unique codebook entries. We denote the spectral distance, $d(\mathbf{v}_i, \mathbf{v}_j)$, between two vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ as d_{ij}. We defer a discussion of spectral distance measures to Chapter 4.
3. a centroid computation procedure. On the basis of the partitioning that classifies the L training set vectors into M clusters we choose the M codebook vectors as the centroid of each of the M clusters.
4. a classification procedure for arbitrary speech spectral analysis vectors that chooses the codebook vector closest to the input vector and uses the codebook index as the resulting spectral representation. This is often referred to as the nearest-neighbor labeling or optimal encoding procedure. The classification procedure is essentially a quantizer that accepts, as input, a speech spectral vector and provides, as output, the codebook index of the codebook vector that best matches the input.

Figure 3.40 shows a block diagram of the basic VQ training and classification structure. In the following sections we discuss each element of the VQ structure in more detail.

3.4.2 The VQ Training Set

To properly train the VQ codebook, the training set vectors should span the anticipated range of the following:

- talkers, including ranges in age, accent, gender, speaking rate, levels, and other variables.
- speaking conditions, such as quiet room, automobile, and noisy workstation.
- transducers and transmission systems, including wideband microphones, telephone handsets (with both carbon and electret microphones), direct transmission, telephone channel, wideband channel, and other devices.
- speech units including specific-recognition vocabularies (e.g., digits) and conversational speech.

The more narrowly focused the training set (i.e., limited talker populations, quiet room speaking, carbon button telephone over a standard telephone channel, vocabulary of digits) the smaller the quantization error in representing the spectral information with a fixed-size codebook. However, for applicability to a wide range of problems, the training set should be as broad, in each of the above dimensions, as possible.

3.4.3 The Similarity or Distance Measure

The spectral distance measure for comparing spectral vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ is of the form

$$d(\mathbf{v}_i, \mathbf{v}_j) = d_{ij} \begin{cases} = 0 & \text{if } \mathbf{v}_i = \mathbf{v}_j \\ > 0 & \text{otherwise} \end{cases} . \qquad (3.93)$$

As we will see in Chapter 4, the distance measure commonly used for comparing filter-bank vectors is an L_1, L_2, or covariance weighted spectral difference, whereas for LPC vectors (and related feature sets such as LPC derived cepstral vectors), measures such as the likelihood and cepstral distance measures are generally used.

3.4.4 Clustering the Training Vectors

The way in which a set of L training vectors can be clustered into a set of M codebook vectors is the following (this procedure is known as the generalized Lloyd algorithm or the K-means clustering algorithm):

1. Initialization: Arbitrarily choose M vectors (initially out of the training set of L vectors) as the initial set of code words in the codebook.
2. Nearest-Neighbor Search: For each training vector, find the code word in the current codebook that is closest (in terms of spectral distance), and assign that vector to the corresponding cell (associated with the closest code word).
3. Centroid Update: Update the code word in each cell using the centroid of the training vectors assigned to that cell.
4. Iteration: Repeat steps 2 and 3 until the average distance falls below a preset threshold.

Figure 3.41 illustrates the result of designing a VQ codebook by showing the partitioning of a (2-dimensional) spectral vector space into distinct regions, each of which is

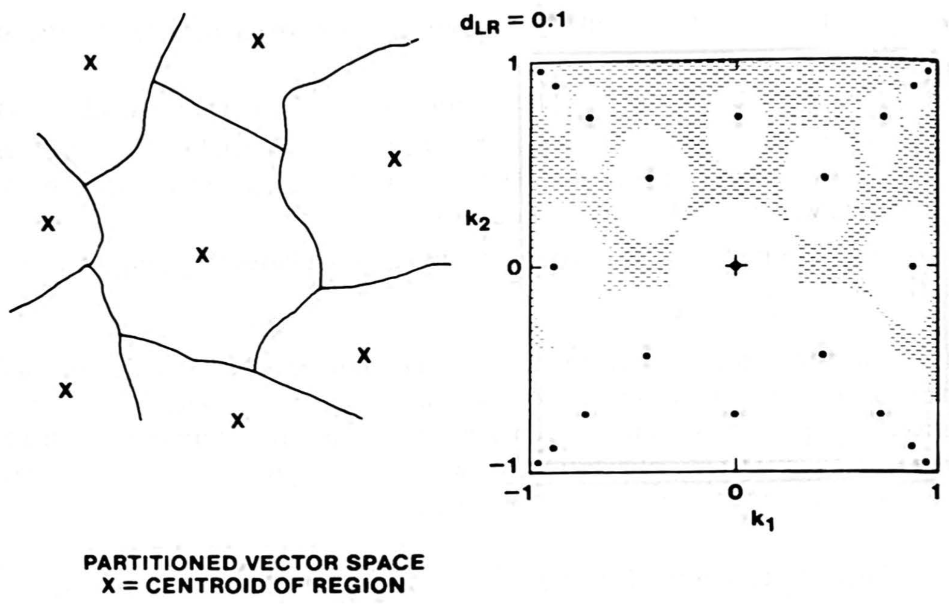

Figure 3.41 Partitioning of a vector space into VQ cells with each cell represented by a centroid vector.

represented by a centroid vector. The shape of each partitioned cell is highly dependent on the spectral distortion measure and the statistics of the vectors in the training set. (For example, if a Euclidean distance is used, the cell boundaries are hyperplanes.)

Although the above iterative procedure works well, it has been shown that it is advantageous to design an M-vector codebook in stages—i.e., by first designing a 1-vector codebook, then using a splitting technique on the code words to initialize the search for a 2-vector codebook, and continuing the splitting process until the desired M-vector codebook is obtained. This procedure is called the binary split algorithm and is formally implemented by the following procedure:

1. Design a 1-vector codebook; this is the centroid of the entire set of training vectors (hence, no iteration is required here).
2. Double the size of the codebook by splitting each current codebook $\mathbf{y}_n$ according to the rule
$$\begin{aligned} \mathbf{y}_n^+ &= \mathbf{y}_n(1+\epsilon) \\ \mathbf{y}_n^- &= \mathbf{y}_n(1-\epsilon), \end{aligned} \tag{3.94}$$
where n varies from 1 to the current size of the codebook, and ϵ is a splitting parameter (typically ϵ is chosen in the range $0.01 \le \epsilon \le 0.05$).
3. Use the K-means iterative algorithm (as discussed above) to get the best set of centroids for the split codebook (i.e., the codebook of twice the size).
4. Iterate steps 2 and 3 until a codebook of size M is designed.

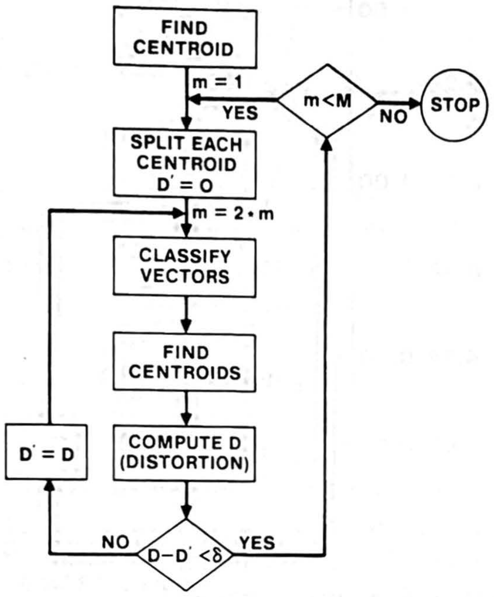

Figure 3.42 Flow diagram of binary split codebook generation algorithm.

Figure 3.42 shows, in a flow diagram, the detailed steps of the binary split VQ codebook generation technique. The box labeled "Classify Vectors" is the nearest-neighbor search procedure, and the box labeled "Find Centroids" is the centroid update procedure of the K-means algorithm. The box labeled "Compute D (Distortion)" sums the distances of all training vectors in the nearest-neighbor search so as to determine whether the procedure has converged (i.e., $D = D'$ of the previous iteration).

To illustrate the effect of codebook size (i.e., number of codebook vectors) on average training set distortion, Figure 3.43 [12] shows experimentally measured values of distortion (in terms of the likelihood ratio measure and the equivalent dB values; see Chapter 4 for more details) versus codebook size (as measured in bits per frame, B) for vectors of both voiced and unvoiced speech. It can be seen that very significant reductions in distortion are achieved in going from a codebook size of 1 bit (2 vectors) to about 7 bits (128 vectors) for both voiced and unvoiced speech. Beyond this point, reductions in distortion are much smaller.

One initial motivation for considering the use of a VQ codebook was the assumption that, in the limit, the codebook should ideally have about 40 vectors—i.e., one vector per speech sound. However, since the codebook vectors represent short time spectral measurements, there is inherently a certain degree of variability in specific codebook entries. Figure 3.44 shows a comparison of codebook vector locations in the $F_1 - F_2$ plane for a 32-vector codebook, along with the vowel ellipses discussed in Chapter 2. (The 32 codewords were generated from a training set of conversational speech spoken by a set of male talkers. The training set included both speech and background signals.) It can be seen that the correspondence between codebook vector location and vowel location is weak. Furthermore, there appears to be a tendency to cluster around the neutral vowel /ɝ/.

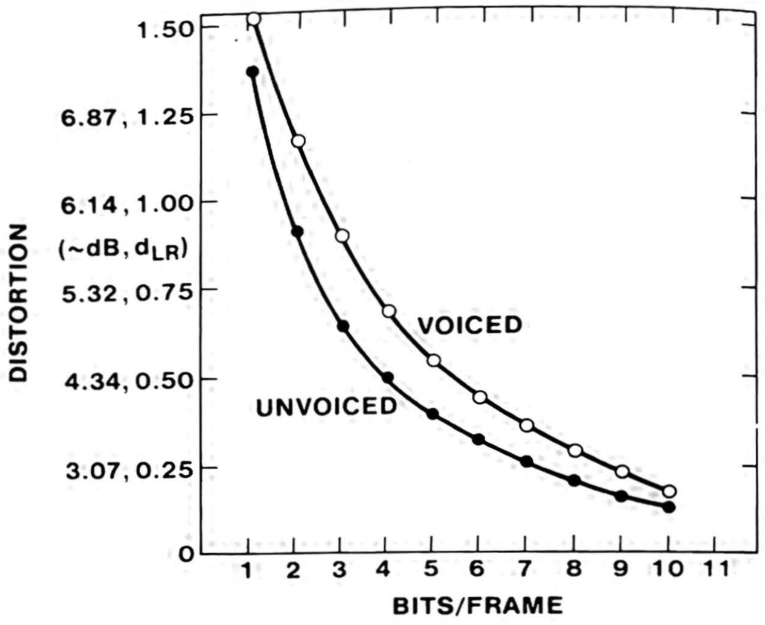

Figure 3.43 Codebook distortion versus codebook size (measured in bits per frame) for both voiced and unvoiced speech (after Juang et al. [12]).

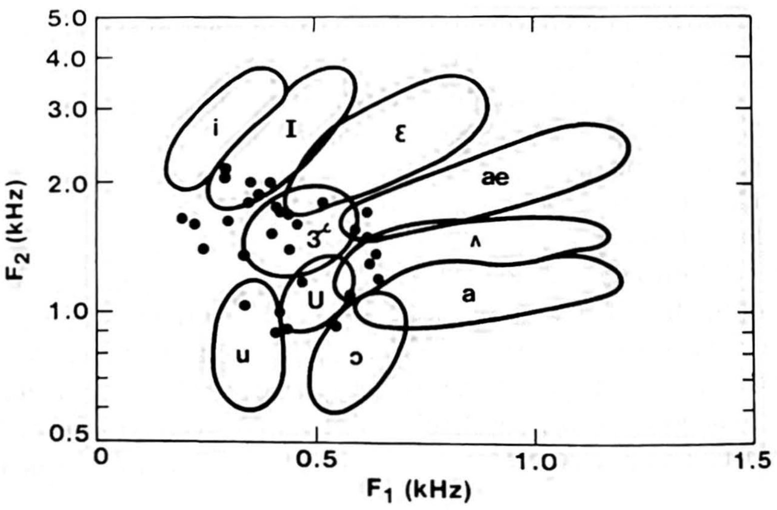

Figure 3.44 Codebook vector locations in the $F_1 - F_2$ plane (for a 32-vector codebook) superimposed on the vowel ellipses (after Juang et al. [12]).

This can be attributed, in part, to both the distortion measure and to the manner in which spectral centroids are computed.

3.4.5 Vector Classification Procedure

The classification procedure for arbitrary spectral vectors is basically a full search through the codebook to find the "best" match. Thus if we denote the codebook vectors of an