

---

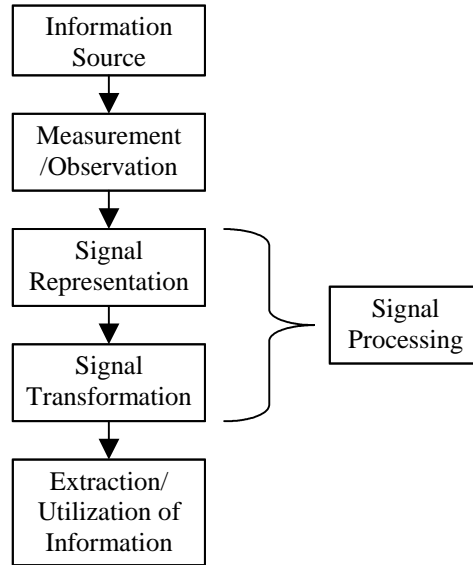
# CHAPTER 5

---

## Digital Signal Processing

One of the most popular ways of characterizing speech is in terms of a *signal* or acoustic waveform. Shown in Figure 5.1 is a representation of the speech signal that ensures that the information content can be easily extracted by human listeners or computers. This is why digital signal processing plays a fundamental role for spoken language processing. We describe here the fundamentals of digital signal processing: digital signals and systems, frequency-domain transforms for both continuous and discrete frequencies, digital filters, the relationship between analog and digital signals, filterbanks, and stochastic processes. In this chapter we set the mathematical foundations of frequency analysis that allow us to develop specific techniques for speech signals in Chapter 6.

The main theme of this chapter is the development of frequency-domain methods computed through the Fourier transform. When we boost the bass knob in our amplifier we are increasing the gain at low frequencies, and when we boost the treble knob we are increasing the gain at high frequencies. Representation of speech signals in the frequency domain is especially useful because the frequency structure of a phoneme is generally unique.

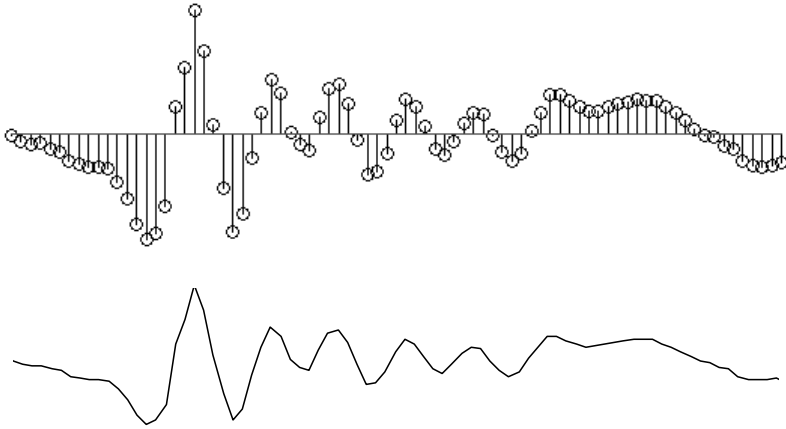


**Figure 5.1** Signal processing is both a representation and a transformation that allows a useful information extraction from a source. The representation and transformation are based on a model of the signal, often parametric, that is convenient for subsequent processing.

## 5.1. DIGITAL SIGNALS AND SYSTEMS

To process speech signals, it is convenient to represent them mathematically as functions of a continuous variable  $t$ , which represents time. Let us define an *analog signal*  $x_a(t)$  as a function varying continuously in time. If we sample the signal  $x$  with a sampling period  $T$  (i.e.,  $t = nT$ ), we can define a discrete-time signal as  $x[n] = x_a(nT)$ , also known as *digital signal*<sup>1</sup>. In this book we use parentheses to describe an analog signal and brackets for digital signals. Furthermore we can define the sampling frequency  $F_s$  as  $F_s = 1/T$ , the inverse of the sampling period  $T$ . For example, for a sampling rate  $F_s = 8\text{kHz}$ , its corresponding sampling period is 125 microseconds. In Section 5.5 it is shown that, under some circumstances, the analog signal  $x_a(t)$  can be recovered exactly from the digital signal  $x[n]$ . Figure 5.2 shows an analog signal and its corresponding digital signal. In subsequent figures, for convenience, we will sometimes plot digital signals as continuous functions.

<sup>1</sup> Actually the term digital signal is defined as a discrete-time signal whose values are represented by integers within a range, whereas a general discrete-time signal would be represented by real numbers. Since the term digital signal is much more commonly used, we will use that term, except when the distinction between them is necessary.



**Figure 5.2** Analog signal and its corresponding digital signal.

The term *Digital Signal Processing* (DSP) refers to methods for manipulating the sequence of numbers  $x[n]$  in a digital computer. The acronym DSP is also used to refer to a *Digital Signal Processor*, i.e., a microprocessor specialized to perform DSP operations.

We start with sinusoidal signals and show they are the fundamental signals for linear systems. We then introduce the concept of convolution and linear time-invariant systems. Other digital signals and nonlinear systems are also introduced.

### 5.1.1. Sinusoidal Signals

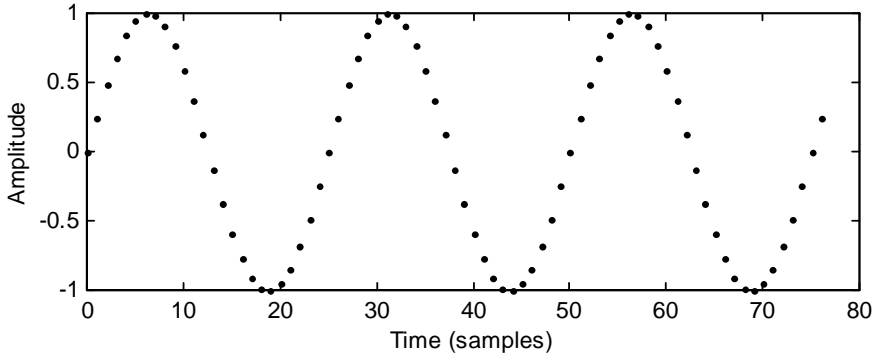
One of the most important signals is the sine wave or *sinusoid*

$$x_0[n] = A_0 \cos(\omega_0 n + \phi_0) \quad (5.1)$$

where  $A_0$  is the sinusoid's amplitude,  $\omega_0$  the angular frequency and  $\phi_0$  the phase. The angle in the trigonometric functions is expressed in radians, so that the angular frequency  $\omega_0$  is related to the normalized linear frequency  $f_0$  by the relation  $\omega_0 = 2\pi f_0$ , and  $0 \leq f_0 \leq 1$ . This signal is *periodic*<sup>2</sup> with period  $T_0 = 1/f_0$ . In Figure 5.3 we can see an example of a sinusoid with frequency  $f_0 = 0.04$ , or a period of  $T_0 = 25$  samples.

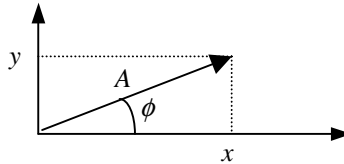
Sinusoids are important because speech signals can be decomposed as sums of sinusoids. When we boost the bass knob in our amplifier we are increasing the gain for sinusoids of low frequencies, and when we boost the treble knob we are increasing the gain for sinusoids of high frequencies.

<sup>2</sup> A signal  $x[n]$  is periodic with period  $N$  if and only if  $x[n] = x[n+N]$ , which requires  $\omega_0 = 2\pi/N$ . This means that the digital signal in Eq. (5.1) is not periodic for all values of  $\omega_0$ , even though its continuous signal counterpart  $x(t) = A_0 \cos(\omega_0 t + \phi_0)$  is periodic for all values of  $\omega_0$  (see Section 5.5).



**Figure 5.3** A digital sinusoid with a period of 25 samples.

What is the sum of two sinusoids  $x_0[n]$  and  $x_1[n]$  of the same frequency  $\omega_0$  but different amplitudes  $A_0$  and  $A_1$ , and phases  $\phi_0$  and  $\phi_1$ ? The answer is another sinusoid of the same frequency but a different amplitude  $A$  and phase  $\phi$ . While this can be computed through trigonometric identities, it is somewhat tedious and not very intuitive. For this reason we introduce another representation based on complex numbers, which proves to be very useful when we study digital filters.



**Figure 5.4** Complex number representation in Cartesian form  $z = x + jy$  and polar form  $z = Ae^{j\phi}$ . Thus  $x = A\cos\phi$  and  $y = A\sin\phi$ .

A complex number  $x$  can be expressed as  $z = x + jy$ , where  $j = \sqrt{-1}$ ,  $x$  is the real part and  $y$  is the imaginary part, with both  $x$  and  $y$  being real numbers. Using Euler's relation, given a real number  $\phi$ , we have

$$e^{j\phi} = \cos\phi + j\sin\phi \quad (5.2)$$

so that the complex number  $z$  can also be expressed in polar form as  $z = Ae^{j\phi}$ , where  $A$  is the amplitude and  $\phi$  is the phase. Both representations can be seen in Figure 5.4, where the real part is shown in the abscissa ( $x$ -axis) and the imaginary part in the ordinate ( $y$ -axis).

Using complex numbers, the sinusoid in Eq. (5.1) can be expressed as the real part of the corresponding complex exponential

$$x_0[n] = A_0 \cos(\omega_0 n + \phi_0) = \text{Re}\{A_0 e^{j(\omega_0 n + \phi_0)}\} \quad (5.3)$$

and thus the sum of two complex exponential signals equals

$$A_0 e^{j(\omega_0 n + \phi_0)} + A_1 e^{j(\omega_0 n + \phi_1)} = e^{j\omega_0 n} (A_0 e^{j\phi_0} + A_1 e^{j\phi_1}) = e^{j\omega_0 n} A e^{j\phi} = A e^{j(\omega_0 n + \phi)} \quad (5.4)$$

Taking the real part in both sides results in

$$A_0 \cos(\omega_0 n + \phi_0) + A_1 \cos(\omega_0 n + \phi_1) = A \cos(\omega_0 n + \phi) \quad (5.5)$$

or in other words, the sum of two sinusoids of the same frequency is another sinusoid of the same frequency.

To compute  $A$  and  $\phi$ , dividing Eq. (5.4) by  $e^{j\omega_0 n}$  leads to a relationship between the amplitude  $A$  and phase  $\phi$ :

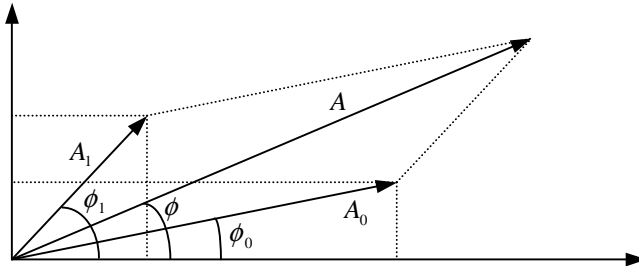
$$A_0 e^{j\phi_0} + A_1 e^{j\phi_1} = A e^{j\phi} \quad (5.6)$$

Equating real and imaginary parts in Eq. (5.6) and dividing them we obtain:

$$\tan \phi = \frac{A_0 \sin \phi_0 + A_1 \sin \phi_1}{A_0 \cos \phi_0 + A_1 \cos \phi_1} \quad (5.7)$$

and adding the squared of real and imaginary parts and using trigonometric identities<sup>3</sup>

$$A^2 = A_0^2 + A_1^2 + 2A_0 A_1 \cos(\phi_0 - \phi_1) \quad (5.8)$$



**Figure 5.5** Geometric representation of the sum of two sinusoids of the same frequency. It follows the complex number representation in Cartesian form of Figure 5.4.

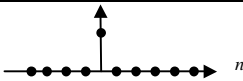
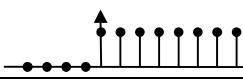
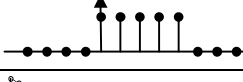
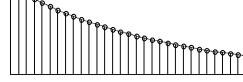
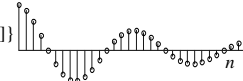
This complex representation of Figure 5.5 lets us analyze and visualize the amplitudes and phases of sinusoids of the same frequency as vectors. The sum of  $N$  sinusoids of the same frequency is another sinusoid of the same frequency that can be obtained by adding the real and imaginary parts of all complex vectors. In Section 5.1.3.3 we show that the output of a linear time-invariant system to a sinusoid is another sinusoid of the same frequency.

<sup>3</sup>  $\sin^2 \phi + \cos^2 \phi = 1$  and  $\cos(a - b) = \cos a \cos b + \sin a \sin b$

### 5.1.2. Other Digital Signals

In the field of digital signal processing there are other signals that repeatedly arise and that are shown in Table 5.1.

**Table 5.1** Some useful digital signals: the Kronecker delta, unit step, rectangular signal, real exponential ( $a < 1$ ) and real part of a complex exponential ( $r < 1$ ).

<i>Kronecker delta, or unit impulse</i>	$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$	
<i>Unit step</i>	$u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$	
<i>Rectangular signal</i>	$\text{rect}_N[n] = \begin{cases} 1 & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$	
<i>Real exponential</i>	$x[n] = a^n u[n]$	
<i>Complex exponential</i>	$x[n] = a^n u[n] = r^n e^{jn\omega_0} u[n]$ $= r^n (\cos n\omega_0 + j \sin n\omega_0) u[n]$	

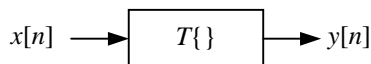
If  $r = 1$  and  $\omega_0 \neq 0$  we have a complex sinusoid as shown in Section 5.1.1. If  $\omega_0 = 0$  we have a real exponential signal, and if  $r < 1$  and  $\omega_0 \neq 0$  we have an exponentially decaying oscillatory sequence, also known as a damped sinusoid.

### 5.1.3. Digital Systems

A digital system is a system that, given an input signal  $x[n]$ , generates an output signal  $y[n]$ :

$$y[n] = T\{x[n]\} \quad (5.9)$$

whose input/output relationship can be seen in Figure 5.6.



**Figure 5.6** Block diagram of a digital system whose input is digital signal  $x[n]$ , and whose output is digital signal  $y[n]$ .

In general, a digital system  $T$  is defined to be linear *iff* (if and only if)

$$T\{a_1 x_1[n] + a_2 x_2[n]\} = a_1 T\{x_1[n]\} + a_2 T\{x_2[n]\} \quad (5.10)$$

for any values of  $a_1$ ,  $a_2$  and any signals  $x_1[n]$  and  $x_2[n]$ .

Here, we study systems according to whether or not they are linear and/or time invariant.

### 5.1.3.1. Linear Time-Invariant Systems

A system is *time-invariant* if given Eq. (5.9), then

$$y[n - n_0] = T\{x[n - n_0]\} \tag{5.11}$$

Linear digital systems of a special type, the so-called *linear time-invariant (LTI)*<sup>4</sup>, are described by

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n - k] = x[n] * h[n] \tag{5.12}$$

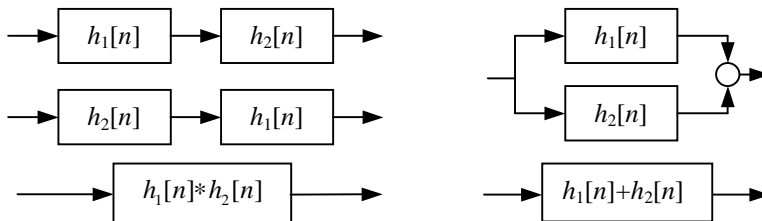
where  $*$  is defined as the *convolution* operator. It is left to the reader to show that the linear system in Eq. (5.12) indeed satisfies Eq. (5.11).

LTI systems are completely characterized by the signal  $h[n]$ , which is known as the system's *impulse response* because it is the output of the system when the input is an impulse  $x[n] = \delta[n]$ . Most of the systems described in this book are LTI systems.

**Table 5.2** Properties of the convolution operator.

Commutative	$x[n] * h[n] = h[n] * x[n]$
Associative	$x[n] * (h_1[n] * h_2[n]) = (x[n] * h_1[n]) * h_2[n] = x[n] * h_1[n] * h_2[n]$
Distributive	$x[n] * (h_1[n] + h_2[n]) = x[n] * h_1[n] + x[n] * h_2[n]$

The convolution operator is commutative, associative and distributive as shown in Table 5.2 and Figure 5.7.



**Figure 5.7** The block diagrams on the left, representing the commutative property, are equivalent. The block diagrams on the right, representing the distributive property, are also equivalent.

<sup>4</sup> Actually the term linear time-invariant (LTI) systems is typically reserved for continuous or analog systems, and linear shift-invariant system is used for discrete-time signals, but we will use LTI for discrete-time signals too since it is widely used in this context.

### 5.1.3.2. Linear Time-Varying Systems

An interesting type of digital systems is that whose output is a linear combination of the input signal at different times:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]g[n, n-k] \quad (5.13)$$

The digital system in Eq. (5.13) is linear, since it satisfies Eq. (5.10). The Linear Time-Invariant systems of Section 5.1.3.1 are a special case of Eq. (5.13) when  $g[n, n-k] = h[n-k]$ . The systems in Eq. (5.13) are called *linear time-varying* (LTV) systems, because the weighting coefficients can vary with time.

A useful example of such system is the so-called *amplitude modulator*

$$y[n] = x[n]\cos \omega_0 n \quad (5.14)$$

used in AM transmissions. As we show in Chapter 6, speech signals are the output of LTV systems. Since these systems are difficult to analyze, we often approximate them with linear time-invariant systems.

**Table 5.3** Examples of nonlinear systems for speech processing. All of them are memoryless except for the median smoother.

Nonlinear System	Equation
Median Smoother of order $(2N+1)$	$y[n] = \text{median}\{x[n-N], \dots, x[n], \dots, x[n+N]\}$
Full-Wave Rectifier	$y[n] =  x[n] $
Half-Wave Rectifier	$y[n] = \begin{cases} x[n] & x[n] \geq 0 \\ 0 & x[n] < 0 \end{cases}$
Frequency Modulator	$y[n] = A \cos(\omega_0 + \Delta\omega x[n])n$
Hard-Limiter	$y[n] = \begin{cases} A & x[n] \geq A \\ x[n] &  x[n]  < A \\ -A & x[n] \leq -A \end{cases}$
Uniform Quantizer ( $L$ -bit) with $2N = 2^L$ intervals of width $\Delta$	$y[n] = \begin{cases} (N-1/2)\Delta & x[n] \geq (N-1)\Delta \\ (m+1/2)\Delta & m\Delta \leq x[n] < (m+1)\Delta & 0 \leq m < N-1 \\ (-m+1/2)\Delta & -m\Delta \leq x[n] < -(m-1)\Delta & 0 < m < N-1 \\ (-N+1/2)\Delta & x[n] < -(N-1)\Delta \end{cases}$



### 5.1.3.3. Nonlinear Systems

Many *nonlinear* systems do not satisfy Eq. (5.10). Table 5.3 includes a list of typical nonlinear systems used in speech processing. All these nonlinear systems are memoryless, because the output at time  $n$  depends only on the input at time  $n$ , except for the *median smoother* of order  $(2N + 1)$  whose output depends also on the previous and the following  $N$  samples.

## 5.2. CONTINUOUS-FREQUENCY TRANSFORMS

A very useful transform for LTI systems is the Fourier transform, because it uses complex exponentials as its basis functions, and its generalization: the  $z$ -transform. In this section we cover both transforms, which are continuous functions of frequency, and their properties.

### 5.2.1. The Fourier Transform

It is instructive to see what the output of a LTI system with impulse response  $h[n]$  is when the input is a complex exponential. Substituting  $x[n] = e^{j\omega_0 n}$  in Eq. (5.12) and using the commutative property of the convolution we obtain

$$y[n] = \sum_{k=-\infty}^{\infty} h[k] e^{j\omega_0(n-k)} = e^{j\omega_0 n} \sum_{k=-\infty}^{\infty} h[k] e^{-j\omega_0 k} = e^{j\omega_0 n} H(e^{j\omega_0}) \quad (5.15)$$

which is another complex exponential of the same frequency and amplitude multiplied by the complex quantity  $H(e^{j\omega_0})$  given by

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h[n] e^{-j\omega n} \quad (5.16)$$

Since the output of a LTI system to a complex exponential is another complex exponential, it is said that complex exponentials are *eigensignals* of LTI systems, with the complex quantity  $H(e^{j\omega_0})$  being their *eigenvalue*.

The quantity  $H(e^{j\omega})$  is defined as the *discrete-time Fourier transform* of  $h[n]$ . It is clear from Eq. (5.16) that  $H(e^{j\omega})$  is a periodic function of  $\omega$  with period  $2\pi$ , and therefore we need to keep only one period to fully describe it, typically  $-\pi < \omega < \pi$  (Figure 5.8).

$H(e^{j\omega})$  is a complex function of  $\omega$  which can be expressed in terms of the real and imaginary parts:

$$H(e^{j\omega}) = H_r(e^{j\omega}) + jH_i(e^{j\omega}) \quad (5.17)$$

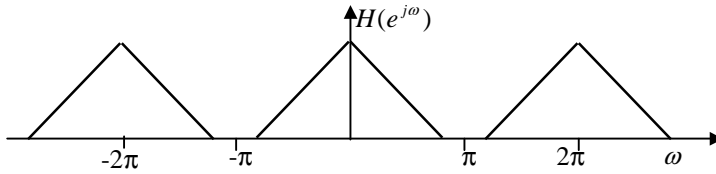
or in terms of the magnitude and phase as

$$H(e^{j\omega}) = |H(e^{j\omega})| e^{j \arg\{H(e^{j\omega})\}} \quad (5.18)$$

Thus if the input to the LTI system is a sinusoid as in Eq. (5.1), the output will be

$$y_0[n] = A_0 |H(e^{j\omega_0})| \cos(\omega_0 n + \phi_0 + \arg\{H(e^{j\omega_0})\}) \quad (5.19)$$

according to Eq. (5.15). Therefore if  $|H(e^{j\omega_0})| > 1$ , the LTI system will amplify that frequency, and likewise it will attenuate, or *filter* it, if  $|H(e^{j\omega_0})| < 1$ . That is one reason why these systems are also called filters. The Fourier transform  $H(e^{j\omega})$  of a filter  $h[n]$  is called the system's *frequency response* or *transfer function*.



**Figure 5.8**  $H(e^{j\omega})$  is a periodic function of  $\omega$ .

The angular frequency  $\omega$  is related to the normalized linear frequency  $f$  by the simple relation  $\omega = 2\pi f$ . We show in Section 5.5 that linear frequency  $f_l$  and normalized frequency  $f$  are related by  $f_l = fF_s$ , where  $F_s$  is the sampling frequency.

The inverse *discrete-time Fourier transform* is defined as

$$h[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega}) e^{j\omega n} d\omega \quad (5.20)$$

The Fourier transform is invertible, and Eq. (5.16) and (5.20) are transform pairs:

$$\begin{aligned} h[n] &= \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega}) e^{j\omega n} d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left( \sum_{m=-\infty}^{\infty} h[m] e^{-j\omega m} \right) e^{j\omega n} d\omega \\ &= \sum_{m=-\infty}^{\infty} h[m] \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega(n-m)} d\omega = \sum_{m=-\infty}^{\infty} h[m] \delta[n-m] = h[n] \end{aligned} \quad (5.21)$$

since

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega(n-m)} d\omega = \delta[n-m] \quad (5.22)$$

A sufficient condition for the existence of the Fourier transform is

$$\sum_{n=-\infty}^{\infty} |h[n]| < \infty \quad (5.23)$$

Although we have computed the Fourier transform of the impulse response of a filter  $h[n]$ , Eq. (5.16) and (5.20) can be applied to any signal  $x[n]$ .

## 5.2.2. Z-Transform

The  $z$ -transform is a generalization of the Fourier transform. The  $z$ -transform of a digital signal  $h[n]$  is defined as

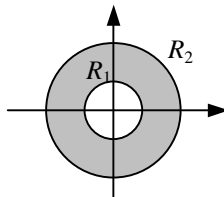
$$H(z) = \sum_{n=-\infty}^{\infty} h[n]z^{-n} \quad (5.24)$$

where  $z$  is a complex variable. Indeed, the Fourier transform of  $h[n]$  equals its  $z$ -transform evaluated at  $z = e^{j\omega}$ . While the Fourier and  $z$ -transforms are often used interchangeably, we normally use the Fourier transform to plot the filter's frequency response, and the  $z$ -transform to analyze more general filter characteristics, given its polynomial functional form. We can also use the  $z$ -transform for unstable filters, which do not have Fourier transforms.

Since Eq. (5.24) is an infinite sum, it is not guaranteed to exist. A sufficient condition for convergence is:

$$\sum_{n=-\infty}^{\infty} |h[n]| |z|^{-n} < \infty \quad (5.25)$$

which is true only for a *region of convergence* (ROC) in the complex  $z$ -plane  $R_1 < |z| < R_2$  as indicated in Figure 5.9.



**Figure 5.9** Region of convergence of the  $z$ -transform in the complex plane.

For a signal  $h[n]$  to have a Fourier transform, its  $z$ -transform  $H(z)$  has to include the unit circle,  $|z| = 1$ , in its convergence region. Therefore, a sufficient condition for the existence of the Fourier transform is given in Eq. (5.23) by applying Eq. (5.25) to the unit circle.

An LTI system is defined to be *causal* if its impulse response is a causal signal, *i.e.*  $h[n] = 0$  for  $n < 0$ . Similarly, a LTI system is *anti-causal* if  $h[n] = 0$  for  $n > 0$ . While all

physical systems are causal, noncausal systems are still useful since causal systems could be decomposed into causal and anti-causal systems.

A system is defined to be *stable* if for every bounded input it produces a bounded output. A necessary and sufficient condition for an LTI system to be stable is

$$\sum_{n=-\infty}^{\infty} |h[n]| < \infty \quad (5.26)$$

which means, according to Eq. (5.23), that  $h[n]$  has a Fourier transform, and therefore that its  $z$ -transform includes the unit circle in its region of convergence.

Just as in the case of Fourier transforms, we can use the  $z$ -transform for any signal, not just for a filter's impulse response.

The *inverse  $z$ -transform* is defined as

$$h[n] = \frac{1}{2\pi j} \oint H(z) z^{n-1} dz \quad (5.27)$$

where the integral is performed along a closed contour that is within the *region of convergence*. Eqs. (5.24) and (5.27) plus knowledge of the region of convergence form a transform pair: *i.e.* one can be exactly determined if the other is known. If the integral is performed along the unit circle (*i.e.*, doing the substitution  $z = e^{j\omega}$ ) we obtain Eq. (5.20), the inverse Fourier transform.

### 5.2.3. Z-Transforms of Elementary Functions

In this section we compute the  $z$ -transforms of the signals defined in Table 5.1. The  $z$ -transforms of such signals are summarized in Table 5.4. In particular we compute the  $z$ -transforms of left-sided and right-sided complex exponentials, which are essential to compute the inverse  $z$ -transform of rational polynomials. As we see in Chapter 6, speech signals are often modeled as having  $z$ -transforms that are rational polynomials.

**Table 5.4**  $Z$ -transforms of some useful signals together with their region of convergence.

Signal	$Z$ -Transform	Region of Convergence
$h_1[n] = \delta[n - N]$	$H_1(z) = z^{-N}$	$z \neq 0$
$h_2[n] = u[n] - u[n - N]$	$H_2(z) = \frac{1 - z^{-N}}{1 - z^{-1}}$	$z \neq 0$
$h_3[n] = a^n u[n]$	$H_3(z) = \frac{1}{1 - az^{-1}}$	$ a  <  z $
$h_4[n] = -a^n u[-n - 1]$	$H_4(z) = \frac{1}{1 - az^{-1}}$	$ z  <  a $

### 5.2.3.1. Right-Sided Complex Exponentials

A right-sided complex exponential sequence

$$h_3[n] = a^n u[n] \quad (5.28)$$

has a  $z$ -transform given by

$$H_3(z) = \sum_{n=0}^{\infty} a^n z^{-n} = \lim_{N \rightarrow \infty} \frac{1 - (az^{-1})^N}{1 - az^{-1}} = \frac{1}{1 - az^{-1}} \quad \text{for } |a| < |z| \quad (5.29)$$

by using the sum of the terms of a geometric sequence and making  $N \rightarrow \infty$ . This region of convergence ( $|a| < |z|$ ) is typical of causal signals (those that are zero for  $n < 0$ ).

When a  $z$ -transform is expressed as the ratio of two polynomials, the roots of the numerator are called *zeros*, and the roots of the denominator are called *poles*. Zeros are the values of  $z$  for which the  $z$ -transform equals 0, and poles are the values of  $z$  for which the  $z$ -transform equals infinity.

$H_3(z)$  has a pole at  $z = a$ , because its value goes to infinity at  $z = a$ . According to Eq. (5.26),  $h_3[n]$  is a stable signal if and only if  $|a| < 1$ , or in other words, if its pole is inside the unit circle. In general, a causal and stable system has all its poles inside the unit circle. As a corollary, a system which has poles outside the unit circle is either noncausal or unstable or both. This is a very important fact, which we exploit throughout the book.

### 5.2.3.2. Left-Sided Complex Exponentials

A left-sided complex exponential sequence

$$h_4[n] = -a^n u[-n-1] \quad (5.30)$$

has a  $z$ -transform given by

$$\begin{aligned} H_4(z) &= -\sum_{n=-\infty}^{-1} a^n z^{-n} = -\sum_{n=1}^{\infty} a^{-n} z^n = 1 - \sum_{n=0}^{\infty} a^{-n} z^n \\ &= 1 - \frac{1}{1 - a^{-1}z} = \frac{-a^{-1}z}{1 - a^{-1}z} = \frac{1}{1 - az^{-1}} \end{aligned} \quad \text{for } |z| < |a| \quad (5.31)$$

This region of convergence ( $|z| < |a|$ ) is typical of noncausal signals (those that are nonzero for  $n < 0$ ). Observe that  $H_3(z)$  and  $H_4(z)$  are functionally identical and only differ in the region of convergence. In general, the region of convergence of a signal that is nonzero for  $-\infty < n < \infty$  is  $R_1 < |z| < R_2$ .

### 5.2.3.3. Inverse Z-Transform of Rational Functions

Integrals in the complex plane such as Eq. (5.27) are not easy to do, but fortunately they are not necessary for the special case of  $H(z)$  being a rational polynomial transform. In this case, partial fraction expansion can be used to decompose the signal into a linear combination of signals like  $h_1[n]$ ,  $h_3[n]$  and  $h_4[n]$  as in Table 5.4.

For example,

$$H_5(z) = \frac{2 + 8z^{-1}}{2 - 5z^{-1} - 3z^{-2}} \quad (5.32)$$

has as roots of its denominator  $z = 3, -1/2$ . Therefore it can be decomposed as

$$H_5(z) = \frac{A}{1 - 3z^{-1}} + \frac{B}{1 + (1/2)z^{-1}} = \frac{(2A + 2B) + (A - 6B)z^{-1}}{2 - 5z^{-1} - 3z^{-2}} \quad (5.33)$$

so that  $A$  and  $B$  are the solution of the following set of linear equations:

$$\begin{aligned} 2A + 2B &= 2 \\ A - 6B &= 8 \end{aligned} \quad (5.34)$$

whose solution is  $A = 2$  and  $B = -1$ , and thus Eq. (5.33) is expressed as

$$H_5(z) = 2 \left( \frac{1}{1 - 3z^{-1}} \right) - \left( \frac{1}{1 + (1/2)z^{-1}} \right) \quad (5.35)$$

However, we cannot compute the inverse  $z$ -transform unless we know the region of convergence. If, for example, we are told that the region of convergence includes the unit circle (necessary for the system to be stable), then the inverse transform of

$$H_4(z) = \frac{1}{1 - 3z^{-1}} \quad (5.36)$$

must have a region of convergence of  $|z| < 3$  according to Table 5.4, and thus be a left-sided complex exponential:

$$h_4[n] = -3^n u[-n - 1] \quad (5.37)$$

and the transform of

$$H_3(z) = \frac{1}{1 + (1/2)z^{-1}} \quad (5.38)$$

must have a region of convergence of  $1/2 < |z|$  according to Table 5.4, and thus be a right-sided complex exponential:

$$h_3[n] = (-1/2)^n u[n] \quad (5.39)$$

so that

$$h_5[n] = -2 \cdot 3^n u[-n-1] - (-1/2)^n u[n] \quad (5.40)$$

While we only showed an example here, the method used generalizes to rational transfer functions with more poles and zeros.

## 5.2.4. Properties of the Z and Fourier Transform

In this section we include a number of properties that are used throughout the book and that can be derived from the definition of Fourier and  $z$ -transforms. Of special interest are the convolution property and Parseval's theorem, which are described below.

### 5.2.4.1. The Convolution Property

The  $z$ -transform of  $y[n]$ , convolution of  $x[n]$  and  $h[n]$ , can be expressed as a function of their  $z$ -transforms:

$$\begin{aligned} Y(z) &= \sum_{n=-\infty}^{\infty} y[n]z^{-n} = \sum_{n=-\infty}^{\infty} \left( \sum_{k=-\infty}^{\infty} x[k]h[n-k] \right) z^{-n} \\ &= \sum_{k=-\infty}^{\infty} x[k] \left( \sum_{n=-\infty}^{\infty} h[n-k]z^{-n} \right) = \sum_{k=-\infty}^{\infty} x[k] \left( \sum_{n=-\infty}^{\infty} h[n]z^{-(n+k)} \right) \\ &= \sum_{k=-\infty}^{\infty} x[k]z^{-k} H(z) = X(z)H(z) \end{aligned} \quad (5.41)$$

which is the fundamental property of LTI systems: "The  $z$ -transform of the convolution of two signals is the product of their  $z$ -transforms." This is also known as the convolution property. The ROC of  $Y(z)$  is now the intersection of the ROCs of  $X(z)$  and  $H(z)$  and cannot be empty for  $Y(z)$  to exist.

Likewise, we can obtain a similar expression for the Fourier transforms:

$$Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega}) \quad (5.42)$$

A dual version of the convolution property can be proven for the product of digital signals:

$$x[n]y[n] \leftrightarrow \frac{1}{2\pi} X(e^{j\omega}) * Y(e^{j\omega}) \quad (5.43)$$

whose transform is the continuous convolution of the transforms with a scale factor. The convolution of functions of continuous variables is defined as

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \quad (5.44)$$

Note how this differs from the discrete convolution of Eq. (5.12).

### 5.2.4.2. Power Spectrum and Parseval's Theorem

Let's define the *autocorrelation* of signal  $x[n]$  as

$$R_{xx}[n] = \sum_{m=-\infty}^{\infty} x[m+n]x^*[m] = \sum_{l=-\infty}^{\infty} x[l]x^*[-(n-l)] = x[n] * x^*[-n] \quad (5.45)$$

where the superscript asterisk (\*) means complex conjugate<sup>5</sup> and should not be confused with the convolution operator.

Using the fundamental property of LTI systems in Eq. (5.42) and the symmetry properties in Table 5.5, we can express its Fourier transform  $S_{xx}(\omega)$  as

$$S_{xx}(\omega) = X(\omega)X^*(\omega) = |X(\omega)|^2 \quad (5.46)$$

which is the *power spectrum*. The Fourier transform of the autocorrelation is the power spectrum:

$$R_{xx}[n] \leftrightarrow S_{xx}(\omega) \quad (5.47)$$

or alternatively

$$R_{xx}[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{xx}(\omega)e^{j\omega n} d\omega \quad (5.48)$$

If we set  $n = 0$  in Eq. (5.48) and use Eq. (5.45) and (5.46), we obtain

$$\sum_{n=-\infty}^{\infty} |x[n]|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(\omega)|^2 d\omega \quad (5.49)$$

which is called *Parseval's theorem* and says that we can compute the signal's energy in the time domain or in the frequency domain.

In Table 5.5 we list, in addition to the convolution property and Parseval's theorem, a number of properties that can be derived from the definition of Fourier and  $z$ -transforms.

## 5.3. DISCRETE-FREQUENCY TRANSFORMS

Here we describe transforms, including the DFT, DCT and FFT, that take our discrete-time signal into a discrete frequency representation. Discrete-frequency transforms are the natural

<sup>5</sup> If  $z = x + jy = Ae^{j\phi}$ , its complex conjugate is defined as  $z^* = x - jy = Ae^{-j\phi}$



transform for periodic signals, though we show in Section 5.7 and Chapter 6 how they are also useful for aperiodic signals such as speech.

**Table 5.5** Properties of the Fourier and  $z$ -transforms.

Property	Signal	Fourier Transform	$z$ -Transform
Linearity	$ax_1[n] + bx_2[n]$	$aX_1(e^{j\omega}) + bX_2(e^{j\omega})$	$aX_1(z) + bX_2(z)$
Symmetry	$x[-n]$	$X(e^{-j\omega})$	$X(z^{-1})$
	$x^*[n]$	$X^*(e^{-j\omega})$	$X^*(z^*)$
	$x^*[-n]$	$X^*(e^{j\omega})$	$X^*(1/z^*)$
	$x[n]$ real	$X(e^{j\omega})$ is Hermitian $X(e^{-j\omega}) = X^*(e^{j\omega})$ $ X(e^{j\omega}) $ is even <sup>6</sup> $\text{Re}\{X(e^{j\omega})\}$ is even $\text{arg}\{X(e^{j\omega})\}$ is odd <sup>7</sup> $\text{Im}\{X(e^{j\omega})\}$ is odd	$X(z^*) = X^*(z)$
	Even $\{x[n]\}$	$\text{Re}\{X(e^{j\omega})\}$	
	Odd $\{x[n]\}$	$j \text{Im}\{X(e^{j\omega})\}$	
Time-shifting	$x[n - n_0]$	$X(e^{j\omega})e^{-j\omega n_0}$	$X(z)z^{-n_0}$
Modulation	$x[n]e^{j\omega_0 n}$	$X(e^{j(\omega - \omega_0)})$	$X(e^{-j\omega_0} z)$
	$x[n]z_0^n$		$X(z/z_0)$
Convolution	$x[n] * h[n]$	$X(e^{j\omega})H(e^{j\omega})$	$X(z)H(z)$
	$x[n]y[n]$	$\frac{1}{2\pi} X(e^{j\omega}) * Y(e^{j\omega})$	
Parseval's Theorem	$R_{xx}[n] = \sum_{m=-\infty}^{\infty} x[m+n]x^*[m]$	$S_{xx}(\omega) =  X(\omega) ^2$	$X(z)X^*(1/z^*)$

A discrete transform of a signal  $x[n]$  is another signal defined as

$$X[k] = \mathcal{T}\{x[n]\} \tag{5.50}$$

Linear transforms are special transforms that decompose the input signal  $x[n]$  into a linear combination of other signals:

<sup>6</sup> A function  $f(x)$  is called even if and only if  $f(x) = f(-x)$ .

<sup>7</sup> A function  $f(x)$  is called odd if and only if  $f(x) = -f(-x)$ .

$$x[n] = \sum_{k=-\infty}^{\infty} X[k] \varphi_k[n] \quad (5.51)$$

where  $\varphi_k[n]$  is a set of *orthonormal* functions

$$\langle \varphi_k[n], \varphi_l[n] \rangle = \delta[k - l] \quad (5.52)$$

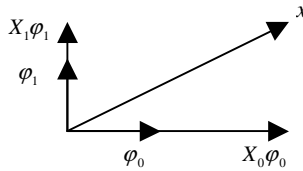
with the inner product defined as

$$\langle \varphi_k[n], \varphi_l[n] \rangle = \sum_{n=-\infty}^{\infty} \varphi_k[n] \varphi_l^*[n] \quad (5.53)$$

With this definition, the coefficients  $X[k]$  are the projection of  $x[n]$  onto  $\varphi_k[n]$ :

$$X[k] = \langle x[n], \varphi_k[n] \rangle \quad (5.54)$$

as illustrated in Figure 5.10.



**Figure 5.10** Orthonormal expansion of a signal  $x[n]$  in a two-dimensional space.

### 5.3.1. The Discrete Fourier Transform (DFT)

If a  $x_N[n]$  signal is periodic with period  $N$  then

$$x_N[n] = x_N[n + N] \quad (5.55)$$

and the signal is uniquely represented by  $N$  consecutive samples. Unfortunately, since Eq. (5.23) is not met, we cannot guarantee the existence of its Fourier transform. The *Discrete Fourier Transform* (DFT) of a periodic signal  $x_N[n]$  is defined as

$$X_N[k] = \sum_{n=0}^{N-1} x_N[n] e^{-j2\pi nk/N} \quad 0 \leq k < N \quad (5.56)$$

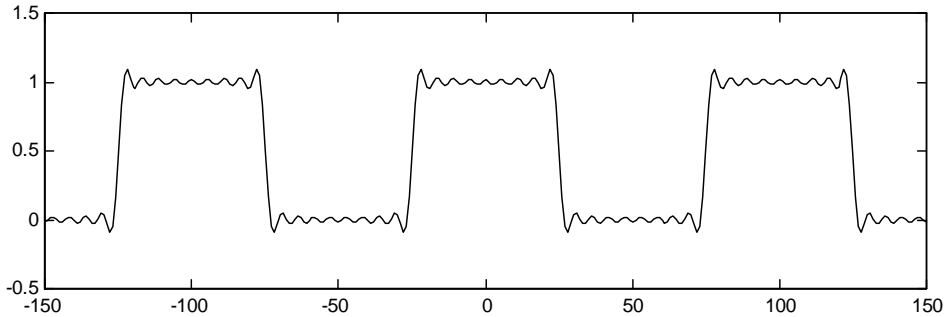
$$x_N[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_N[k] e^{j2\pi nk/N} \quad 0 \leq n < N \quad (5.57)$$

which are transform pairs. Equation (5.57) is also referred as a *Fourier series* expansion.

In Figure 5.11 we see the approximation of a periodic square signal with period  $N = 100$  as a sum of 19 *harmonic* sinusoids, *i.e.*, we used only the first 19  $X_N[k]$  coefficients in Eq. (5.57).

$$\tilde{x}_N[n] = \frac{1}{N} \sum_{k=-18}^{18} X_N[k] e^{j2\pi nk/N} = \frac{X_N[0]}{N} + \frac{2}{N} \sum_{k=1}^{18} X_N[k] \cos(2\pi nk/N) \quad (5.58)$$

Had we used 100 harmonic sinusoids, the periodic signal would have been reproduced *exactly*. Nonetheless, retaining a smaller number of sinusoids can provide a decent approximation for a periodic signal.



**Figure 5.11** Decomposition of a periodic square signal with period 100 samples as a sum of 19 harmonic sinusoids with frequencies  $\omega_k = 2\pi k/100$ .

## 5.3.2. Fourier Transforms of Periodic Signals

Using the DFT, we now discuss how to compute the Fourier transforms of a complex exponential, an impulse train, and a general periodic signal, since they are signals often used in DSP. We also present a relationship between the continuous-frequency Fourier transform and the discrete Fourier transform.

### 5.3.2.1. The Complex Exponential

One of the simplest periodic functions is the complex exponential  $x[n] = e^{j\omega_0 n}$ . Since it has infinite energy, we cannot compute its Fourier transform in its strict sense. Since such signals are so useful, we devise an alternate formulation.

First, let us define the function

$$d_{\Delta}(\omega) = \begin{cases} 1/\Delta & 0 \leq \omega < \Delta \\ 0 & \text{otherwise} \end{cases} \quad (5.59)$$

which has the following property

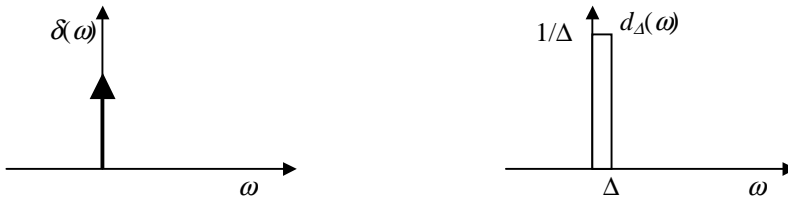
$$\int_{-\infty}^{\infty} d_{\Delta}(\omega) d\omega = 1 \quad (5.60)$$

for all values of  $\Delta > 0$ .

It is useful to define the continuous delta function  $\delta(\omega)$ , also known as the *Dirac delta*, as

$$\delta(\omega) = \lim_{\Delta \rightarrow 0} d_{\Delta}(\omega) \quad (5.61)$$

which is a *singular function* and can be seen in Figure 5.12. The Dirac delta is a function of a continuous variable and should not be confused with the Kronecker delta, which is a function of a discrete variable.



**Figure 5.12** Representation of the  $\delta(\omega)$  function and its approximation  $d_{\Delta}(\omega)$ .

Using Eqs. (5.59) and (5.61) we can then see that

$$\int_{-\infty}^{\infty} X(\omega) \delta(\omega) d\omega = \lim_{\Delta \rightarrow 0} \int_{-\infty}^{\infty} X(\omega) d_{\Delta}(\omega) d\omega = X(0) \quad (5.62)$$

and similarly

$$\int_{-\infty}^{\infty} X(\omega) \delta(\omega - \omega_0) d\omega = X(\omega_0) \quad (5.63)$$

so that

$$X(\omega) \delta(\omega - \omega_0) = X(\omega_0) \delta(\omega - \omega_0) \quad (5.64)$$

because the integrals on both sides are identical.

Using Eq. (5.63), we see that the convolution of  $X(\omega)$  and  $\delta(\omega - \omega_0)$  is

$$X(\omega) * \delta(\omega - \omega_0) = \int_{-\infty}^{\infty} X(u) \delta(\omega - \omega_0 - u) du = X(\omega - \omega_0) \quad (5.65)$$

For the case of a complex exponential, inserting  $X(\omega) = e^{j\omega n}$  into Eq. (5.63) results in

$$\int_{-\infty}^{\infty} \delta(\omega - \omega_0) e^{j\omega n} d\omega = e^{j\omega_0 n} \quad (5.66)$$

By comparing Eq. (5.66) with (5.20) we can then obtain

$$e^{j\omega_0 n} \leftrightarrow 2\pi\delta(\omega - \omega_0) \tag{5.67}$$

so that the Fourier transform of a complex exponential is an impulse concentrated at frequency  $\omega_0$ .

### 5.3.2.2. The Impulse Train

Since the impulse train

$$p_N[n] = \sum_{k=-\infty}^{\infty} \delta[n - kN] \tag{5.68}$$

is periodic with period  $N$ , it can be expanded in Fourier series according to (5.56) as

$$P_N[k] = 1 \tag{5.69}$$

so that using the inverse Fourier series Eq. (5.57),  $p_N[n]$  can alternatively be expressed as

$$p_N[n] = \frac{1}{N} \sum_{k=0}^{N-1} e^{j2\pi kn/N} \tag{5.70}$$

which is an alternate expression to Eq. (5.68) as a sum of complex exponentials. Taking the Fourier transform of Eq. (5.70) and using Eq. (5.67) we obtain

$$P_N(e^{j\omega}) = \frac{2\pi}{N} \sum_{k=0}^{N-1} \delta(\omega - 2\pi k/N) \tag{5.71}$$

which is another impulse train in the frequency domain (See Figure 5.13). The impulse train in the time domain is given in terms of the Kronecker delta, and the impulse train in the frequency domain is given in terms of the Dirac delta.

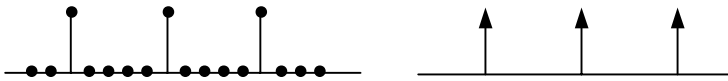


Figure 5.13 An impulse train signal and its Fourier transform, which is also an impulse train.

### 5.3.2.3. General Periodic Signals

We now compute the Fourier transform of a general periodic signal using the results of Section 5.3.2.2 and show that, in addition to being periodic, the transform is also discrete. Given a periodic signal  $x_N[n]$  with period  $N$ , we define another signal  $x[n]$ :

$$x[n] = \begin{cases} x_N[n] & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases} \tag{5.72}$$

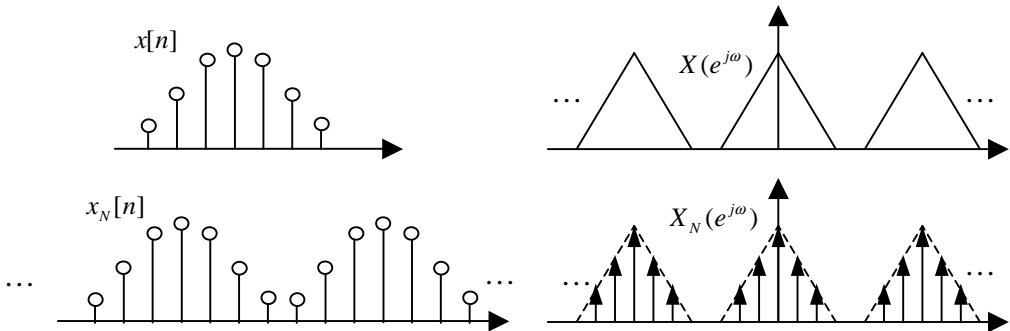
so that

$$x_N[n] = \sum_{k=-\infty}^{\infty} x[n-kN] = x[n] * \sum_{k=-\infty}^{\infty} \delta[n-kN] = x[n] * p_N[n] \quad (5.73)$$

which is the convolution of  $x[n]$  with an impulse train  $p_N[n]$  as in Eq. (5.68). Since  $x[n]$  is of finite length, it has a Fourier transform  $X(e^{j\omega})$ . Using the convolution property  $X_N(e^{j\omega}) = X(e^{j\omega})P_N(e^{j\omega})$ , where  $P_N(e^{j\omega})$  is the Fourier transform of  $p_N[n]$  as given by Eq. (5.71), we obtain another impulse train:

$$X_N(e^{j\omega}) = \frac{2\pi}{N} \sum_{k=-\infty}^{\infty} X(e^{j2\pi k/N}) \delta(\omega - 2\pi k/N) \quad (5.74)$$

Therefore the Fourier transform  $X_N(e^{j\omega})$  of a periodic signal  $x_N[n]$  can be expressed in terms of samples  $\omega_k = 2\pi k/N$ , spaced  $2\pi/N$  apart, of the Fourier transform  $X(e^{j\omega})$  of  $x[n]$ , one period of the signal  $x_N[n]$ . The relationships between  $x[n]$ ,  $x_N[n]$ ,  $X(e^{j\omega})$  and  $X_N(e^{j\omega})$  are shown in Figure 5.14.



**Figure 5.14** Relationships between finite and periodic signals and their Fourier transforms. On one hand,  $x[n]$  is a length  $N$  discrete signal whose transform  $X(e^{j\omega})$  is continuous and periodic with period  $2\pi$ . On the other hand,  $x_N[n]$  is a periodic signal with period  $N$  whose transform  $X_N(e^{j\omega})$  is discrete and periodic.

### 5.3.3. The Fast Fourier Transform (FFT)

There is a family of fast algorithms to compute the DFT, which are called Fast Fourier Transforms (FFT). Direct computation of the DFT from Eq. (5.56) requires  $N^2$  operations, assuming that the trigonometric functions have been pre-computed. The FFT algorithm only requires on the order of  $N \log_2 N$  operations, so it is widely used for speech processing.

### 5.3.3.1. Radix-2 FFT

Let's express the discrete Fourier transform of  $x[n]$

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N} = \sum_{n=0}^{N-1} x[n] W_N^{nk} \quad 0 \leq k < N \quad (5.75)$$

where we have defined for convenience

$$W_N = e^{-j2\pi/N} \quad (5.76)$$

Equation (5.75) requires  $N^2$  complex multiplies and adds. Now, let's suppose  $N$  is even, and let  $f[n] = x[2n]$  represent the even-indexed samples of  $x[n]$ , and  $g[n] = x[2n+1]$  the odd-indexed samples. We can express Eq. (5.75) as

$$X[k] = \sum_{n=0}^{N/2-1} f[n] W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} g[n] W_{N/2}^{nk} = F[k] + W_N^k G[k] \quad (5.77)$$

where  $F[k]$  and  $G[k]$  are the  $N/2$  point DFTs of  $f[n]$  and  $g[n]$ , respectively. Since both  $F[k]$  and  $G[k]$  are defined for  $0 \leq k < N/2$ , we need to also evaluate them for  $N/2 \leq k < N$ , which is straightforward, since

$$F[k + N/2] = F[k] \quad (5.78)$$

$$G[k + N/2] = G[k] \quad (5.79)$$

If  $N/2$  is also even, then both  $f[n]$  and  $g[n]$  can be decomposed into sequences of even and odd indexed samples and therefore its DFT can be computed using the same process. Furthermore, if  $N$  is an integer power of 2, this process can be iterated and it can be shown that the number of multiplies and adds is  $N \log_2 N$ , which is a significant saving from  $N^2$ . This is the *decimation-in-time* algorithm and can be seen in Figure 5.15. A dual algorithm called *decimation-in-frequency* can be derived by decomposing the signal into its first  $N/2$  and its last  $N/2$  samples.

### 5.3.3.2. Other FFT Algorithms

Although the radix-2 FFT is the best known algorithm, there are other variants that are faster and are more often used in practice. Among those are the radix-4, radix-8, split-radix and prime-factor algorithm.

The same process used in the derivation of the radix-2 decimation-in-time algorithm applies if we decompose the sequences into four sequences:  $f_1[n] = x[4n]$ ,  $f_2[n] = x[4n+1]$ ,  $f_3[n] = x[4n+2]$ , and  $f_4[n] = x[4n+3]$ . This is the radix-4 algorithm,

which can be applied when  $N$  is a power of 4, and is generally faster than an equivalent radix-2 algorithm.

Similarly there are radix-8 and radix-16 algorithms for  $N$  being powers of 8 and 16 respectively, which use fewer multiplies and adds. But because of possible additional control logic, it is not obvious that they will be faster, and every algorithm needs to be optimized for a given processor.

There are values of  $N$ , such as  $N = 128$ , for which we cannot use radix-4, radix-8 nor radix-16, so we have to use the less efficient radix-2. A combination of radix-2 and radix-4, called *split-radix* [5], has been shown to have fewer multiplies than both radix-2 and radix-4, and can be applied to  $N$  being a power of 2.

Finally, another possible decomposition is  $N = p_1 p_2 \cdots p_L$  with  $p_i$  being prime numbers. This leads to the *prime-factor algorithm* [2]. While this family of algorithms offers a similar number of operations as the algorithms above, it offers more flexibility in the choice of  $N$ .

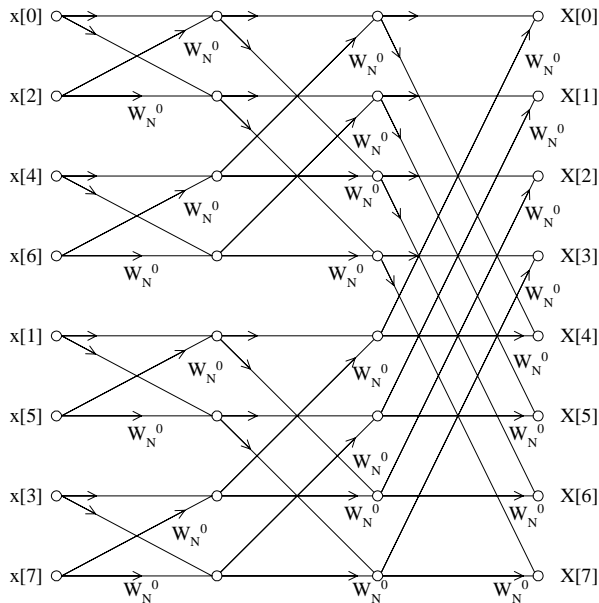


Figure 5.15 Decimation in time radix-2 algorithm for an 8-point FFT.

### 5.3.3.3. FFT Subroutines

Typically, FFT subroutines are computed *in-place* to save memory and have the form

```
fft (float *xr, float *xi, int n)
```

where  $xr$  and  $xi$  are the real and imaginary parts respectively of the input sequence, before calling the subroutine, and the real and imaginary parts of the output transform, after return-



ing from it. C code that implements a decimation-in-time radix-2 FFT of Figure 5.15 is shown in Figure 5.16.

```

void fft2 (float *x, float *y, int n, int m)
{
    int n1, n2, i, j, k, l;
    float xt, yt, c, s;
    double e, a;

    /* Loop through all m stages */
    n2 = n;
    for (k = 0; k < m; k++) {
        n1 = n2;
        n2 = n2 / 2;
        e = PI2 / n1;
        for (j = 0; j < n2; j++) {
            /* Compute Twiddle factors */
            a = j * e;
            c = (float) cos (a);
            s = (float) sin (a);

            /* Do the butterflies */
            for (i = j; i < n; i += n1) {
                l = i + n2;
                xt = x[i] - x[l];
                x[i] = x[i] + x[l];
                yt = y[i] - y[l];
                y[i] = y[i] + y[l];
                x[l] = c * xt + s * yt;
                y[l] = c * yt - s * xt;
            }
        }

        /* Bit reversal: descrambling */
        j = 0;
        for (i = 0; i < n - 1; i++) {
            if (i < j) {
                xt = x[j];
                x[j] = x[i];
                x[i] = xt;
                yt = y[j];
                y[j] = y[i];
                y[i] = yt;
            }
            k = n / 2;
            while (k <= j) {
                j -= k;
                k /= 2;
            }
            j += k;
        }
    }
}

```

**Figure 5.16** C source for a decimation-in-time radix-2 FFT. Before calling the subroutine,  $x$  and  $y$  contain the real and imaginary parts of the input signal respectively. After returning from the subroutine,  $x$  and  $y$  contain the real and imaginary parts of the Fourier transform of the input signal.  $n$  is the length of the FFT and is related to  $m$  by  $n = 2^m$ .

The first part of the subroutine in Figure 5.16 is doing the so-called *butterflies*, which use the trigonometric factors, also called *twiddle factors*. Normally, those twiddle factors are

pre-computed and stored in a table. The second part of the subroutine deals with the fact that the output samples are not linearly ordered (see Figure 5.15), in fact the indexing has the bits reversed, which is why we need to do *bit reversal*, also called *descrambling*.

To compute the inverse FFT an additional routine is not necessary; it can be computed with the subroutine above. To see that, we expand the DFT in Eq. (5.56) into its real and imaginary parts:

$$X_R[k] + jX_I[k] = \sum_{n=0}^{N-1} (x_R[n] + jx_I[n]) e^{-j2\pi nk/N} \quad (5.80)$$

take complex conjugate and multiply by  $j$  to obtain

$$X_I[k] + jX_R[k] = \sum_{n=0}^{N-1} (x_I[n] + jx_R[n]) e^{j2\pi nk/N} \quad (5.81)$$

which has the same functional form as the expanded inverse DFT of Eq. (5.57)

$$x_R[k] + jx_I[k] = \frac{1}{N} \sum_{n=0}^{N-1} (X_R[n] + jX_I[n]) e^{j2\pi nk/N} \quad (5.82)$$

so that the inverse FFT can be computed by calling `fft (xi, xr, n)` other than the  $(1/N)$  factor.

Often the input signal  $x[n]$  is real, so that we know from the symmetry properties of Table 5.5 that its Fourier transform is Hermitian. This symmetry can be used to compute the length- $N$  FFT more efficiently with a length  $(N/2)$  FFT. One way of doing so is to define  $f[n] = x[2n]$  to represent the even-indexed samples of  $x[n]$ , and  $g[n] = x[2n+1]$  the odd-indexed samples. We can then define a length  $(N/2)$  complex signal  $h[n]$  as

$$h[n] = f[n] + jg[n] = x[2n] + jx[2n+1] \quad (5.83)$$

whose DFT is

$$H[k] = F[k] + jG[k] = H_R[k] + jH_I[k] \quad (5.84)$$

Since  $f[n]$  and  $g[n]$  are real, their transforms are Hermitian and thus

$$H^*[-k] = F^*[-k] - jG^*[-k] = F[k] - jG[k] \quad (5.85)$$

Using Eqs. (5.84) and (5.85), we can obtain  $F[k]$  and  $G[k]$  as a function of  $H_R[k]$  and  $H_I[k]$ :

$$F[k] = \frac{H[k] + H^*[-k]}{2} = \left( \frac{H_R[k] + H_R[-k]}{2} \right) + j \left( \frac{H_I[k] - H_I[-k]}{2} \right) \quad (5.86)$$

$$G[k] = \frac{H[k] - H^*[-k]}{2j} = \left( \frac{H_I[k] + H_I[-k]}{2} \right) - j \left( \frac{H_R[k] - H_R[-k]}{2} \right) \quad (5.87)$$

As shown in Eq. (5.77),  $X[k]$  can be obtained as a function of  $F[k]$  and  $G[k]$

$$X[k] = F[k] + G[k]W_N^{-k} \quad (5.88)$$

so that the DFT of the real sequence  $x[n]$  is obtained through Eqs. (5.83), (5.86), (5.87) and (5.88). The computational complexity is a length  $(N/2)$  complex FFT plus  $N$  real multiplies and  $3N$  real adds.

### 5.3.4. Circular Convolution

The convolution of two periodic signals is not defined according to Eq. (5.12). Given two periodic signals  $x_1[n]$  and  $x_2[n]$  with period  $N$ , we define their *circular convolution* as

$$y[n] = x_1[n] \otimes x_2[n] = \sum_{m=0}^{N-1} x_1[m]x_2[n-m] = \sum_{m=\langle N \rangle} x_1[m]x_2[n-m] \quad (5.89)$$

where  $m = \langle N \rangle$  in Eq. (5.89) means that the sum lasts only one period. In fact, the sum could be over any  $N$  consecutive samples, not just the first  $N$ . Moreover,  $y[n]$  is also periodic with period  $N$ . Furthermore, it is left to the reader to show that

$$Y[k] = X_1[k]X_2[k] \quad (5.90)$$

*i.e.*, the DFT of  $y[n]$  is the product of the DFTs of  $x_1[n]$  and  $x_2[n]$ .

An important application of the above result is the computation of a regular convolution using a circular convolution. Let  $x_1[n]$  and  $x_2[n]$  be two signals such that  $x_1[n] = 0$  outside  $0 \leq n < N_1$ , and  $x_2[n] = 0$  outside  $0 \leq n < N_2$ . We know that their regular convolution  $y[n] = x_1[n] * x_2[n]$  is zero outside  $0 \leq n < N_1 + N_2 - 1$ . If we choose an integer  $N$  such that  $N \geq N_1 + N_2 - 1$ , we can define two periodic signals  $\tilde{x}_1[n]$  and  $\tilde{x}_2[n]$  with period  $N$  such that

$$\tilde{x}_1[n] = \begin{cases} x_1[n] & 0 \leq n < N_1 \\ 0 & N_1 \leq n < N \end{cases} \quad (5.91)$$

$$\tilde{x}_2[n] = \begin{cases} x_2[n] & 0 \leq n < N_2 \\ 0 & N_2 \leq n < N \end{cases} \quad (5.92)$$

where  $x_1[n]$  and  $x_2[n]$  have been *zero padded*. It can be shown that the circular convolution  $\tilde{y}[n] = \tilde{x}_1[n] \otimes \tilde{x}_2[n]$  is identical to  $y[n]$  for  $0 \leq n < N$ , which means that  $y[n]$  can be ob-

tained as the inverse DFT of  $\tilde{Y}[k] = \tilde{X}_1[k]\tilde{X}_2[k]$ . This method of computing the regular convolution of two signals is more efficient than the direct calculation when  $N$  is large. While the crossover point will depend on the particular implementations of the FFT and convolution, as well as the processor, in practice this has been found beneficial for  $N \geq 1024$ .

### 5.3.5. The Discrete Cosine Transform (DCT)

The Discrete Cosine Transform (DCT) is a widely used for speech processing. It has several definitions. The DCT-II  $C[k]$  of a real signal  $x[n]$  is defined by:

$$C[k] = \sum_{n=0}^{N-1} x[n] \cos(\pi k(n+1/2)/N) \quad \text{for } 0 \leq k < N \quad (5.93)$$

with its inverse given by

$$x[n] = \frac{1}{N} \left\{ C[0] + 2 \sum_{k=1}^{N-1} C[k] \cos(\pi k(n+1/2)/N) \right\} \quad \text{for } 0 \leq n < N \quad (5.94)$$

The DCT-II can be derived from the DFT by assuming  $x[n]$  is a real periodic sequence with period  $2N$  and with an even symmetry  $x[n] = x[2N-1-n]$ . It is left to the reader to show, that  $X[k]$  and  $C[k]$  are related by

$$X[k] = 2e^{j\pi k/2N} C[k] \quad \text{for } 0 \leq k < N \quad (5.95)$$

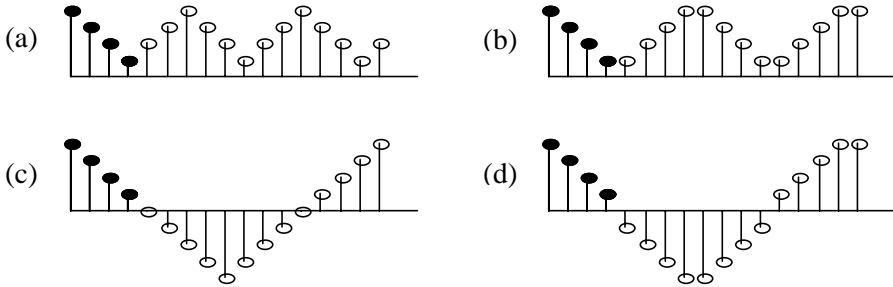
$$X[2N-k] = 2e^{-j\pi k/2N} C[k] \quad \text{for } 0 \leq k < N \quad (5.96)$$

It is left to the reader to prove Eq. (5.94) is indeed the inverse transform using Eqs. (5.57), (5.95), and (5.96). Other versions of the DCT-II have been defined that differ on the normalization constants but are otherwise the same.

There are eight different ways to extend an  $N$ -point sequence and make it both periodic and even, such that can be uniquely recovered. The DCT-II is just one of the ways, with three others being shown in Figure 5.17.

The DCT-II is the most often used discrete cosine transform because of its *energy compaction*, which results in its coefficients being more concentrated at low indices than the DFT. This property allows us to approximate the signal with fewer coefficients [10].

From Eq. (5.95) and (5.96) we see that the DCT-II of a real sequence can be computed with a length- $2N$  FFT of a real and even sequence, which in turn can be computed with a length  $(N/2)$  complex FFT and some additional computations. Other fast algorithms have been derived to compute the DCT directly [15], using the principles described in Section 5.3.3.1. Two-dimensional transforms can also be used for image processing.



**Figure 5.17** Four ways to extend a four-point sequence  $x[n]$  to make it both periodic and have even symmetry. The figures in (a), (b), (c) and (d) correspond to the DCT-I, DCT-II, DCT-III and DCT-IV respectively.

## 5.4. DIGITAL FILTERS AND WINDOWS

We describe here the fundamentals of digital filter design and study *finite-impulse response* (FIR) and *infinite-impulse response* (IIR) filters, which are special types of linear time-invariant digital filters. We establish the time-frequency duality and study the ideal low-pass filter (frequency limited) and its dual window functions (time limited). These transforms are applied to stochastic processes.

### 5.4.1. The Ideal Low-Pass Filter

It is useful to find an impulse response  $h[n]$  whose Fourier transform is

$$H(e^{j\omega}) = \begin{cases} 1 & |\omega| < \omega_0 \\ 0 & \omega_0 < |\omega| < \pi \end{cases} \quad (5.97)$$

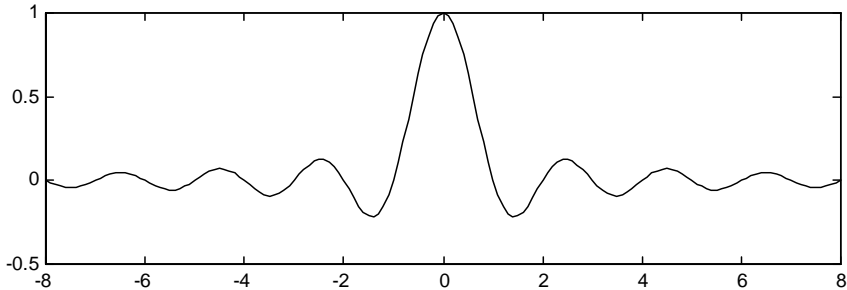
which is the ideal *low-pass* filter because it lets all frequencies below  $\omega_0$  pass through unaffected and completely blocks frequencies above  $\omega_0$ . Using the definition of Fourier transform, we obtain

$$h[n] = \frac{1}{2\pi} \int_{-\omega_0}^{\omega_0} e^{j\omega n} d\omega = \frac{(e^{j\omega_0 n} - e^{-j\omega_0 n})}{2\pi j n} = \frac{\sin \omega_0 n}{\pi n} = \left(\frac{\omega_0}{\pi}\right) \text{sinc}(\omega_0 n) \quad (5.98)$$

where we have defined the so-called sinc function as

$$\text{sinc}(x) = \frac{\sin \pi x}{\pi x} \quad (5.99)$$

which is a real and even function of  $x$  and is plotted in Figure 5.18. Note that the sinc function is 0 when  $x$  is a nonzero integer.



**Figure 5.18** A sinc function, which is the impulse response of the ideal low-pass filter with a scale factor.

Thus, an ideal low-pass filter is noncausal since it has an impulse response with an infinite number of nonzero coefficients.

## 5.4.2. Window Functions

Window functions are signals that are concentrated in time, often of limited duration. While window functions such as *triangular*, *Kaiser*, *Barlett*, and *prolate spheroidal* occasionally appear in digital speech processing systems, the *rectangular*, *Hanning*, and *Hamming* are the most widely used. Window functions are also concentrated in low frequencies. These window functions are useful in digital filter design and all throughout Chapter 6.

### 5.4.2.1. The Rectangular Window

The *rectangular* window is defined as

$$h_{\pi}[n] = u[n] - u[n - N] \quad (5.100)$$

and we refer to it often in this book. Its  $z$ -transform is given by

$$H_{\pi}(z) = \sum_{n=0}^{N-1} z^{-n} \quad (5.101)$$

which results in a polynomial of order  $(N - 1)$ . Multiplying both sides of Eq. (5.101) by  $z^{-1}$ , we obtain

$$z^{-1}H_{\pi}(z) = \sum_{n=1}^N z^{-n} = H_{\pi}(z) - 1 + z^{-N} \quad (5.102)$$

and therefore the sum of the terms of a geometric series can also be expressed as

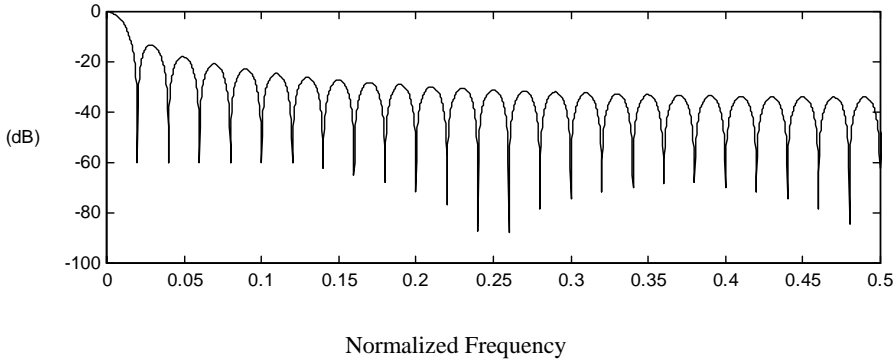
$$H_{\pi}(z) = \frac{1 - z^{-N}}{1 - z^{-1}} \quad (5.103)$$

Although  $z = 1$  appears to be a pole in Eq. (5.103), it actually isn't because it is canceled by a zero at  $z = 1$ . Since  $h_{\pi}[n]$  has finite length, Eq. (5.25) must be satisfied for  $z \neq 0$ , so the region of convergence is everywhere but at  $z = 0$ . Moreover, all finite-length sequences have a region of convergence that is the complete  $z$ -plane except for possibly  $z = 0$ .

The Fourier transform of the rectangular window is, using Eq. (5.103):

$$\begin{aligned} H_{\pi}(e^{j\omega}) &= \frac{1 - e^{-j\omega N}}{1 - e^{-j\omega}} = \frac{(e^{j\omega N/2} - e^{-j\omega N/2})e^{-j\omega N/2}}{(e^{j\omega/2} - e^{-j\omega/2})e^{-j\omega/2}} \\ &= \frac{\sin \omega N / 2}{\sin \omega / 2} e^{-j\omega(N-1)/2} = A(\omega)e^{-j\omega(N-1)/2} \end{aligned} \quad (5.104)$$

where  $A(\omega)$  is real and even. The function  $A(\omega)$ , plotted in Figure 5.19 in dB,<sup>8</sup> is 0 for  $\omega_k = 2\pi k / N$  with  $k \neq \{0, \pm N, \pm 2N, \dots\}$ , and is the discrete-time equivalent of the sinc function.



**Figure 5.19** Frequency response (magnitude in dB) of the rectangular window with  $N = 50$ , which is a digital sinc function.

### 5.4.2.2. The Generalized Hamming Window

The *generalized Hamming window* is defined as

<sup>8</sup> An energy value  $E$  is expressed in decibels (dB) as  $\bar{E} = 10 \log_{10} E$ . If the energy value is  $2E$ , it is therefore 3dB higher. Logarithmic measurements like dB are useful because they correlate well with how the human auditory system perceives volume.

$$h_h[n] = \begin{cases} (1-\alpha) - \alpha \cos(2\pi n/N) & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases} \quad (5.105)$$

and can be expressed in terms of the rectangular window in Eq. (5.100) as

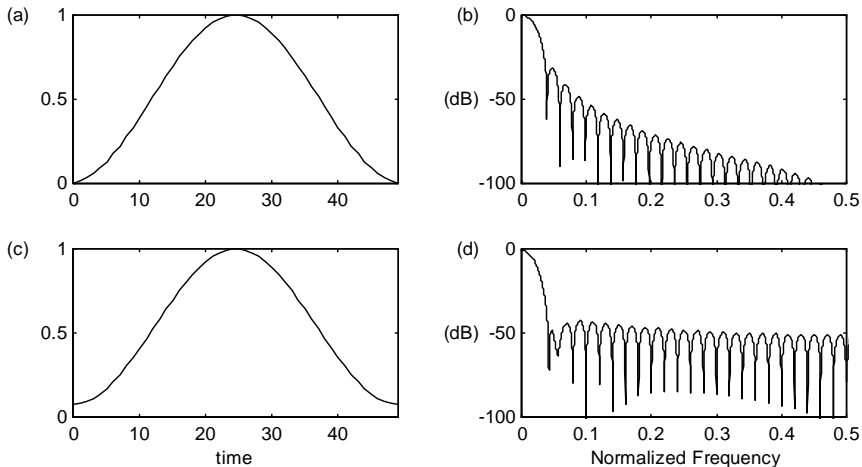
$$h_h[n] = (1-\alpha)h_\pi[n] - \alpha h_\pi[n] \cos(2\pi n/N) \quad (5.106)$$

whose transform is

$$H_h(e^{j\omega}) = (1-\alpha)H_\pi(e^{j\omega}) - (\alpha/2)H_\pi(e^{j(\omega-2\pi/N)}) - (\alpha/2)H_\pi(e^{j(\omega+2\pi/N)}) \quad (5.107)$$

after using the modulation property in Table 5.5. When  $\alpha = 0.5$  the window is known as the *Hanning* window, whereas for  $\alpha = 0.46$  it is the *Hamming* window. Hanning and Hamming windows and their magnitude frequency responses are plotted in Figure 5.20.

The main lobe of both Hamming and Hanning is twice as wide as that of the rectangular window, but the attenuation is much greater than that of the rectangular window. The secondary lobe of the Hanning window is 31 dB below the main lobe, whereas for the Hamming window it is 44 dB below. On the other hand, the attenuation of the Hanning window decays with frequency quite rapidly, which is not the case for the Hamming window, whose attenuation stays approximately constant for all frequencies.



**Figure 5.20** (a) Hanning window and (b) the magnitude of its frequency response in dB; (c) Hamming window and (d) the magnitude of its frequency response in dB for  $N = 50$ .

### 5.4.3. FIR Filters

From a practical point of view, it is useful to consider LTI filters whose impulse responses have a limited number of nonzero coefficients:



$$h[n] = \begin{cases} b_n & 0 \leq n \leq M \\ 0 & \text{otherwise} \end{cases} \quad (5.108)$$

These types of LTI filters are called *finite-impulse response* (FIR) filters. The input/output relationship in this case is

$$y[n] = \sum_{r=0}^M b_r x[n-r] \quad (5.109)$$

The  $z$ -transform of  $x[n-r]$  is

$$\sum_{n=-\infty}^{\infty} x[n-r]z^{-n} = \sum_{n=-\infty}^{\infty} x[n]z^{-(n+r)} = z^{-r}X(z) \quad (5.110)$$

Therefore, given that the  $z$ -transform is linear,  $H(z)$  is

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{r=0}^M b_r z^{-r} = Az^{-L} \prod_{r=1}^{M-L} (1 - c_r z^{-1}) \quad (5.111)$$

whose region of convergence is the whole  $z$ -plane except for possibly  $z=0$ . Since  $\sum_{r=0}^M |b_r|$  is finite, FIR systems are always stable, which makes them very attractive. Several special types of FIR filters will be analyzed below: linear-phase, first-order and low-pass FIR filters.

### 5.4.3.1. Linear-Phase FIR Filters

Linear-phase filters are important because, other than a delay, the phase of the signal is unchanged. Only the magnitude is affected. Therefore, the temporal properties of the input signal are preserved. In this section we show that linear-phase FIR filters can be built if the filter exhibits symmetry.

Let's explore the particular case of  $h[n]$  real,  $M=2L$ , an even number, and  $h[n] = h[M-n]$  (called a *Type-I* filter). In this case

$$\begin{aligned} H(e^{j\omega}) &= \sum_{n=0}^M h[n]e^{-j\omega n} = h[L]e^{-j\omega L} + \sum_{n=0}^{L-1} (h[n]e^{-j\omega n} + h[M-n]e^{-j\omega(2L-n)}) \\ &= h[L]e^{-j\omega L} + \sum_{n=0}^{L-1} h[n](e^{-j\omega(n-L)} + e^{j\omega(n-L)})e^{-j\omega L} \\ &= \left( h[L] + \sum_{n=1}^L 2h[n+L]\cos(\omega n) \right) e^{-j\omega L} = A(\omega)e^{-j\omega L} \end{aligned} \quad (5.112)$$

where  $A(\omega)$  is a real and even function of  $\omega$ , since the cosine is an even function, and  $A(\omega)$  is a linear combination of cosines. Furthermore, we see that the phase

$\arg\{H(e^{j\omega})\} = L\omega$ , which is a linear function of  $\omega$ , and therefore  $h[n]$  is called a *linear-phase* system. It can be shown that if  $h[n] = -h[M - n]$ , we also get a linear phase system but  $A(\omega)$  this time is a pure imaginary and odd function (Type III filter). It is left to the reader to show that in the case of  $M$  being odd the system is still linear phase (Types II and IV filters). Moreover,  $h[n]$  doesn't have to be real and:

$$h[n] = \pm h^*[M - n] \quad (5.113)$$

is a sufficient condition for  $h[n]$  to be linear phase.

### 5.4.3.2. First-Order FIR Filters

A special case of FIR filters is the first-order filter:

$$y[n] = x[n] + \alpha x[n - 1] \quad (5.114)$$

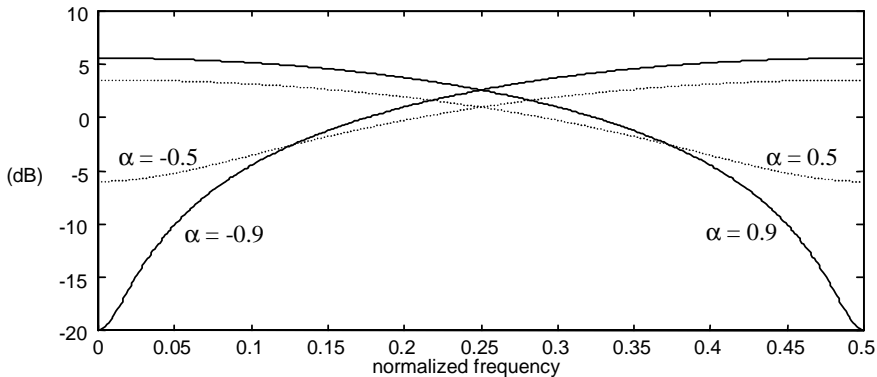
for real values of  $\alpha$ , which, unless  $\alpha = 1$ , is not linear phase. Its  $z$ -transform is

$$H(z) = 1 + \alpha z^{-1} \quad (5.115)$$

It is of interest to analyze the magnitude and phase of its frequency response

$$\begin{aligned} |H(e^{j\omega})|^2 &= |1 + \alpha(\cos \omega - j \sin \omega)|^2 \\ &= (1 + \alpha \cos \omega)^2 + (\alpha \sin \omega)^2 = 1 + \alpha^2 + 2\alpha \cos \omega \end{aligned} \quad (5.116)$$

$$\theta(e^{j\omega}) = -\arctan\left(\frac{\alpha \sin \omega}{1 + \alpha \cos \omega}\right) \quad (5.117)$$



**Figure 5.21** Frequency response of the first order FIR filter for various values of  $\alpha$ .

It is customary to display the magnitude response in decibels (dB):

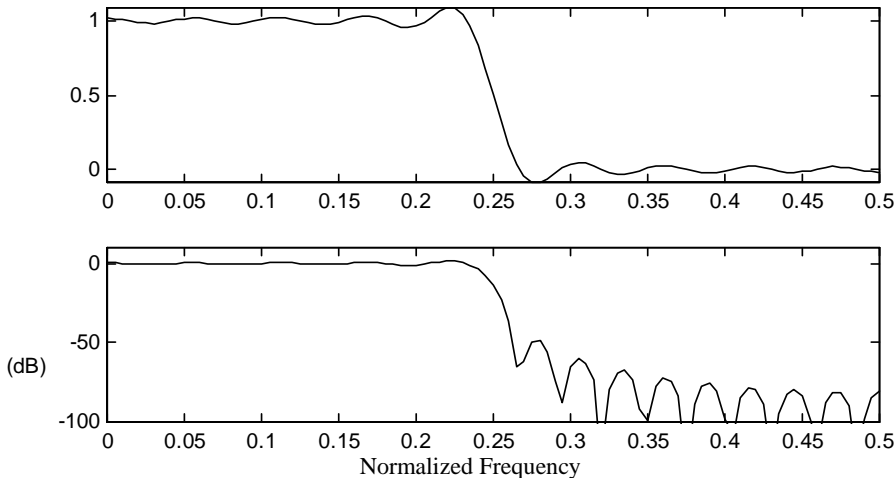
$$10 \log |H(e^{j\omega})|^2 = 10 \log [(1 + \alpha)^2 + 2\alpha \cos \omega] \quad (5.118)$$

as shown in Figure 5.21 for various values of  $\alpha$ .

We see that for  $\alpha > 0$  we have a *low-pass* filter whereas for  $\alpha < 0$  it is a *high-pass* filter, also called a *pre-emphasis* filter, since it emphasizes the high frequencies. In general, filters that boost the high frequencies and attenuate the low frequencies are called *high-pass* filters, and filters that emphasize the low frequencies and de-emphasize the high frequencies are called *low-pass* filters. The parameter  $\alpha$  controls the slope of the curve.

### 5.4.3.3. Window Design FIR Lowpass Filters

The ideal lowpass filter lets all frequencies below  $\omega_0$  go through and eliminates all energy from frequencies above that range. As we described in Section 5.4.1, the ideal lowpass filter has an infinite impulse response, which poses difficulties for implementation in a practical system, as it requires an infinite number of multiplies and adds.

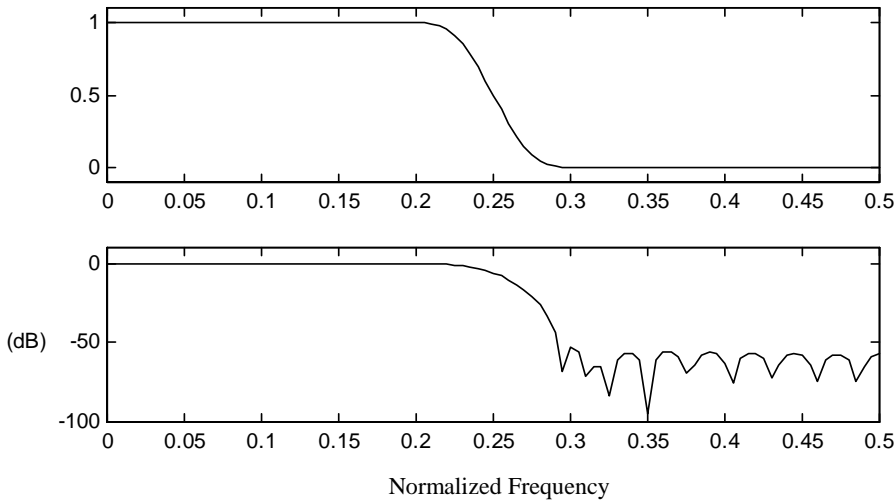


**Figure 5.22** Magnitude frequency response of the truncated sinc signal ( $N=200$ ) for  $\omega_0 = \pi/4$ . It is an approximation to the ideal low-pass filter, though we see that overshoots are present near the transition. The first graph is linear magnitude and the second is in dB.

Since we know that the sinc function decays over time, it is reasonable to assume that a truncated sinc function that keeps a large enough number of samples  $N$  could be a good approximation to the ideal low-pass filter. Figure 5.22 shows the magnitude of the frequency response of such a truncated sinc function for different values of  $N$ . While the approximation gets better for larger  $N$ , the overshoot near  $\omega_0$  doesn't go away and it facts stays at about

9% of the discontinuity even for large  $N$ . This is known as the *Gibbs phenomenon*, since Yale professor Josiah Gibbs first noticed it in 1899.

In computing the truncated sinc function, we have implicitly multiplied the ideal low-pass filter, the sinc function, by a rectangular window. In the so-called *window design* filter design method, the filter coefficients are obtained by multiplying the ideal sinc function by a tapering window function, such as the Hamming window. The resulting frequency response is the convolution of the ideal lowpass filter function with the transform of the window (shown in Figure 5.23), and it does not exhibit the overshoots seen above, at the expense of a slower transition.



**Figure 5.23** Magnitude frequency response of a low-pass filter obtained with the window design method and a Hamming window ( $N = 200$ ). The first graph is linear magnitude and the second is in dB.

#### 5.4.3.4. Parks McClellan Algorithm

While the window design method is simple, it is hard to predict what the final response will be. Other methods have been proposed whose coefficients are obtained to satisfy some constraints. If our constraints are a maximum ripple of  $\delta_p$  in the *passband* ( $0 \leq \omega < \omega_p$ ), and a minimum attenuation of  $\delta_s$  in the *stopband* ( $\omega_s \leq \omega < \pi$ ), the optimal solution is given by the Parks McClellan algorithm [14].

The transformation

$$x = \cos \omega \tag{5.119}$$

maps the interval  $0 \leq \omega \leq \pi$  into  $-1 \leq x \leq 1$ . We note that

$$\cos(n\omega) = T_n(\cos \omega) \tag{5.120}$$

where  $T_n(x)$  is the  $n^{\text{th}}$ -order Chebyshev polynomial. The first two Chebyshev polynomials are given by  $T_0(x) = 1$  and  $T_1(x) = x$ . If we add the following trigonometric identities

$$\begin{aligned} \cos(n+1)\omega &= \cos n\omega \cos \omega - \sin n\omega \sin \omega \\ \cos(n-1)\omega &= \cos n\omega \cos \omega + \sin n\omega \sin \omega \end{aligned} \tag{5.121}$$

and use Eqs. (5.119) and (5.120), we obtain the following recursion formula:

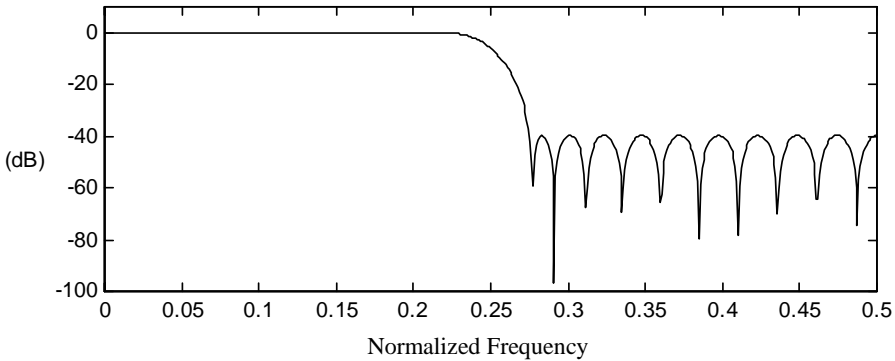
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad \text{for } n > 1 \tag{5.122}$$

Using Eq. (5.120), the magnitude response of a linear phase Type-I filter in Eq. (5.112) can be expressed as an  $L^{\text{th}}$ -order polynomial in  $\cos \omega$ :

$$A(\omega) = \sum_{k=0}^L a_k (\cos \omega)^k \tag{5.123}$$

which, using Eq. (5.119) results in a polynomial

$$P(x) = \sum_{k=0}^L a_k x^k \tag{5.124}$$



**Figure 5.24** Magnitude frequency response of a length-19 lowpass filter designed with the Parks McClellan algorithm.

Given that a desired response is  $D(x) = D(\cos \omega)$ , we define the weighted squared error as

$$E(x) = E(\cos \omega) = W(\cos \omega)[D(\cos \omega) - P(\cos \omega)] = W(x)[D(x) - P(x)] \tag{5.125}$$

where  $W(\cos \omega)$  is the weighting in  $\omega$ . A necessary and sufficient condition for this weighted squared error to be minimized is to have  $P(x)$  alternate between minima and maxima. For the case of a low-pass filter,

$$D(\cos \omega) = \begin{cases} 1 & \cos \omega_p \leq \cos \omega \leq 1 \\ 0 & -1 \leq \cos \omega \leq \cos \omega_s \end{cases} \quad (5.126)$$

and the weight in the stopband is several times larger than that in the passband.

These constraints and the response of a filter designed with such a method are shown in Figure 5.24. We can thus obtain a similar transfer function with fewer coefficients using this method.

#### 5.4.4. IIR Filters

Other useful filters are a function of past values of the input and also the output

$$y[n] = \sum_{k=1}^N a_k y[n-k] + \sum_{r=0}^M b_r x[n-r] \quad (5.127)$$

whose  $z$ -transform is given by

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{r=0}^M b_r z^{-r}}{1 - \sum_{k=1}^N a_k z^{-k}} \quad (5.128)$$

which in turn can be expressed as a function of the roots of the numerator  $c_r$  (called *zeros*), and denominator  $d_k$  (called *poles*) as

$$H(z) = \frac{Az^{-L} \prod_{r=1}^{M-L} (1 - c_r z^{-1})}{\prod_{k=1}^N (1 - d_k z^{-1})} \quad (5.129)$$

It is not obvious what the impulse response of such a system is by looking at either Eq. (5.128) or Eq. (5.129). To do that, we can compute the inverse  $z$ -transform of Eq. (5.129). If  $M < N$  in Eq. (5.129),  $H(z)$  can be expanded into partial fractions (see Section 5.2.3.3) as

$$H(z) = \sum_{k=1}^N \frac{A_k}{1 - d_k z^{-1}} \quad (5.130)$$

and if  $M \geq N$

$$H(z) = \sum_{k=1}^N \frac{A_k}{1 - d_k z^{-1}} + \sum_{k=0}^{M-N} B_k z^{-k} \quad (5.131)$$

which we can now compute, since we know that the inverse  $z$ -transform of  $H_k(z) = A_k/(1-d_k z^{-1})$  is

$$h_k[n] = \begin{cases} A_k d_k^n u[n] & |d_k| < 1 \\ -A_k d_k^n u[-n-1] & |d_k| > 1 \end{cases} \quad (5.132)$$

so that the convergence region includes the unit circle and therefore  $h_k[n]$  is stable. Therefore, a necessary and sufficient condition for  $H(z)$  to be stable *and* causal simultaneously is that all its poles be inside the unit circle: *i.e.*,  $|d_k| < 1$  for all  $k$ , so that its impulse response is given by

$$h[n] = B_n + \sum_{k=1}^N A_k d_k^n u[n] \quad (5.133)$$

which has an infinite impulse response, and hence its name.

Since IIR systems may have poles outside the unit circle, they are not guaranteed to be stable and causal like their FIR counterparts. This makes IIR filter design more difficult, since only stable and causal filters can be implemented in practice. Moreover, unlike FIR filters, IIR filters do not have linear phase. Despite these difficulties, IIR filters are popular because they are more efficient than FIR filters in realizing steeper roll-offs with fewer coefficients. In addition, as shown in Chapter 6, they represent many physical systems.

#### 5.4.4.1. First-Order IIR Filters

An important type of IIR filter is the first-order filter of the form

$$y[n] = Ax[n] + \alpha y[n-1] \quad (5.134)$$

for  $\alpha$  real. Its transfer function is given by

$$H(z) = \frac{A}{1-\alpha z^{-1}} \quad (5.135)$$

This system has one pole and no zeros. As we saw in our discussion of  $z$ -transforms in Section 5.2.3, a necessary condition for this system to be both stable and causal is that  $|\alpha| < 1$ . Since for the low-pass filter case  $0 < \alpha < 1$ , it is convenient to define  $\alpha = e^{-b}$  where  $b > 0$ . In addition, the corresponding impulse response is infinite:

$$h[n] = \alpha^n u[n] \quad (5.136)$$

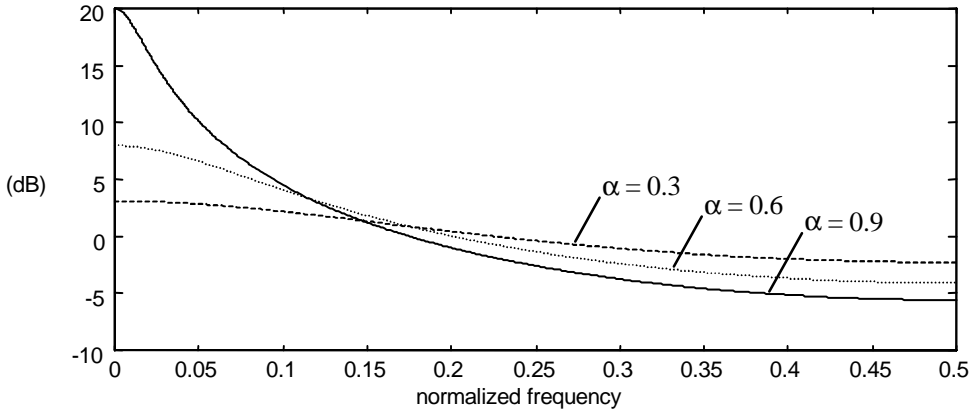
whose Fourier transform is

$$H(e^{j\omega}) = \frac{A}{1-\alpha e^{-j\omega}} = \frac{A}{1-e^{-b-j\omega}} \quad (5.137)$$

and magnitude square is given by

$$|H(e^{j\omega})|^2 = \frac{|A|^2}{1 + \alpha^2 - 2\alpha \cos \omega} \quad (5.138)$$

which is shown in Figure 5.25 for  $\alpha > 0$ , which corresponds to a low-pass filter.



**Figure 5.25** Magnitude frequency response of the first-order IIR filter.

The bandwidth of a low-pass filter is defined as the point where its magnitude square is half of its maximum value. Using the first-order Taylor approximation of the exponential function, the following approximation can be used when  $b \rightarrow 0$ :

$$|H(e^{j0})|^2 = \frac{A^2}{|1 - e^{-b}|^2} \approx \frac{A^2}{b^2} \quad (5.139)$$

If the bandwidth  $\omega_b$  is also small, we can similarly approximate

$$|H(e^{j\omega_b})|^2 = \frac{A^2}{|1 - e^{-b - j\omega_b}|^2} \approx \frac{A^2}{|b + j\omega_b|^2} = \frac{A^2}{(b^2 + \omega_b^2)} \quad (5.140)$$

so that for  $\omega_b = b$  we have  $|H(e^{jb})|^2 \approx 0.5 |H(e^{j0})|^2$ . In other words, the bandwidth of this filter equals  $b$ , for small values of  $b$ . The relative error in this approximation<sup>9</sup> is smaller than 2% for  $b < 0.5$ , which corresponds to  $0.6 < \alpha < 1$ . The relationship with the unnormalized bandwidth  $B$  is

$$\alpha = e^{-2\pi B / F_s} \quad (5.141)$$

<sup>9</sup> The exact value is  $\omega_b = \arccos[2 - \cosh b]$ , where  $\cosh b = (e^b + e^{-b})/2$  is the hyperbolic cosine.



For  $\alpha < 0$  it behaves as a high-pass filter, and a similar discussion can be carried out.

### 5.4.4.2. Second-Order IIR Filters

An important type of IIR filters is the set of second-order filters of the form

$$y[n] = Ax[n] + a_1 y[n-1] + a_2 y[n-2] \quad (5.142)$$

whose transfer function is given by

$$H(z) = \frac{A}{1 - a_1 z^{-1} - a_2 z^{-2}} \quad (5.143)$$

This system has two poles and no zeros. A special case is when the coefficients  $A$ ,  $a_1$  and  $a_2$  are real. In this case the two poles are given by

$$z = \frac{a_1 \pm \sqrt{a_1^2 + 4a_2}}{2} \quad (5.144)$$

which for the case of  $a_1^2 + 4a_2 > 0$  yields two real roots, and is a degenerate case of two first-order systems. The more interesting case is when  $a_1^2 + 4a_2 < 0$ . In this case we see that the two roots are complex conjugates of each other, which can be expressed in their magnitude and phase notation as

$$z = e^{-\sigma \pm j\omega_0} \quad (5.145)$$

As we mentioned before,  $\sigma > 0$  is a necessary and sufficient condition for the poles to be inside the unit circle and thus for the system to be stable. With those values, the  $z$ -transform is given by

$$H(z) = \frac{A}{(1 - e^{-\sigma + j\omega_0} z^{-1})(1 - e^{-\sigma - j\omega_0} z^{-1})} = \frac{A}{1 - 2e^{-\sigma} \cos(\omega_0) z^{-1} + e^{-2\sigma} z^{-2}} \quad (5.146)$$

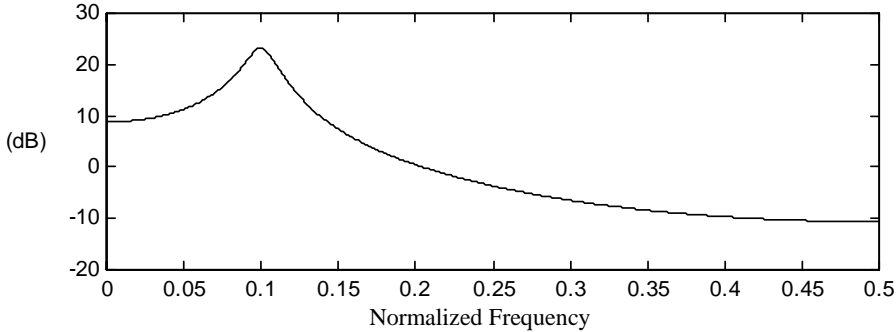
In Figure 5.26 we show the magnitude of its Fourier transform for a value of  $\sigma$  and  $\omega_0$ . We see that the response is centered around  $\omega_0$  and is more concentrated for smaller values of  $\sigma$ . This is a type of *bandpass* filter, since it favors frequencies in a band around  $\omega_0$ . It is left to the reader as an exercise to show that the bandwidth<sup>10</sup> is approximately  $2\sigma$ . The smaller the ratio  $\sigma/\omega_0$ , the sharper the resonance. The filter coefficients can be expressed as a function of the unnormalized bandwidth  $B$  and resonant frequency  $F$  and the sampling frequency  $F_s$  (all expressed in Hz) as

$$a_1 = 2e^{-\pi B/F_s} \cos(2\pi F/F_s) \quad (5.147)$$

<sup>10</sup> The bandwidth of a bandpass filter is the region between half maximum magnitude squared values.

$$a_2 = -e^{-2\pi B/F_s} \quad (5.148)$$

These types of systems are also known as second-order resonators and will be of great use for speech synthesis (Chapter 16), particularly for formant synthesis.



**Figure 5.26** Frequency response of the second-order IIR filter for center frequency of  $F = 0.1F_s$  and bandwidth  $B = 0.01F_s$ .

## 5.5. DIGITAL PROCESSING OF ANALOG SIGNALS

To use the digital signal processing methods, it is necessary to convert the speech signal  $x(t)$ , which is analog, to a digital signal  $x[n]$ , which is formed by periodically sampling the analog signal  $x(t)$  at intervals equally spaced  $T$  seconds apart:

$$x[n] = x(nT) \quad (5.149)$$

where  $T$  is defined as the sampling period, and its inverse  $F_s = 1/T$  as the sampling frequency. In the speech applications considered in this book,  $F_s$  can range from 8000 Hz for telephone applications to 44,100 Hz for high-fidelity audio applications. This section explains the sampling theorem, which essentially says that the analog signal  $x(t)$  can be uniquely recovered given its digital signal  $x[n]$  if the analog signal  $x(t)$  has no energy for frequencies above the *Nyquist* frequency  $F_s/2$ .

We not only prove the sampling theorem, but also provide great insight into the analog-digital conversion, which is used in Chapter 7.

### 5.5.1. Fourier Transform of Analog Signals

The Fourier transform of an analog signal  $x(t)$  is defined as

$$X(\Omega) = \int_{-\infty}^{\infty} x(t)e^{-j\Omega t} dt \quad (5.150)$$

with its inverse transform being

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\Omega)e^{j\Omega t} d\Omega \quad (5.151)$$

They are transform pairs. You can prove similar relations for the Fourier transform of analog signals as for their digital signals counterpart.

### 5.5.2. The Sampling Theorem

Let's define  $x_p(t)$

$$x_p(t) = x(t)p(t) \quad (5.152)$$

as a sampled version of  $x(t)$ , where

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \quad (5.153)$$

where  $\delta(t)$  is the Dirac delta defined in Section 5.3.2.1. Therefore,  $x_p(t)$  can also be expressed as

$$x_p(t) = \sum_{n=-\infty}^{\infty} x(t)\delta(t - nT) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT) = \sum_{n=-\infty}^{\infty} x[n]\delta(t - nT) \quad (5.154)$$

after using Eq. (5.149). In other words,  $x_p(t)$  can be uniquely specified given the digital signal  $x[n]$ .

Using the modulation property of Fourier transforms of analog signals, we obtain

$$X_p(\Omega) = \frac{1}{2\pi} X(\Omega) * P(\Omega) \quad (5.155)$$

Following a derivation similar to that in Section 5.3.2.2, one can show that the transform of the impulse train  $p(t)$  is given by

$$P(\Omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(\Omega - k\Omega_s) \quad (5.156)$$

where  $\Omega_s = 2\pi F_s$  and  $F_s = 1/T$ , so that

$$X_p(\Omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X(\Omega - k\Omega_s) \quad (5.157)$$

From Figure 5.27 it can be seen that if

$$X(\Omega) = 0 \text{ for } |\Omega| > \Omega_s / 2 \quad (5.158)$$

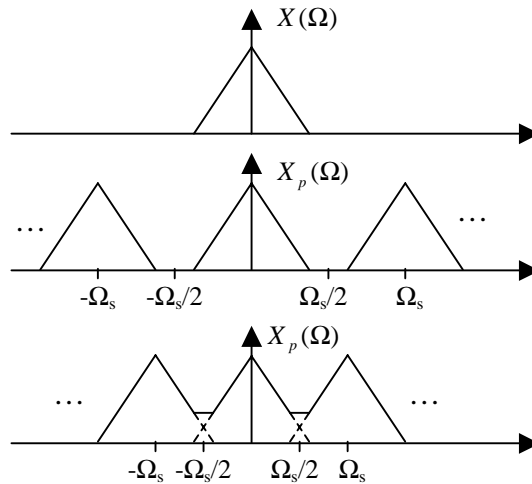
then  $X(\Omega)$  can be completely recovered from  $X_p(\Omega)$  as follows

$$X(\Omega) = R_{\Omega_s}(\Omega)X_p(\Omega) \quad (5.159)$$

where

$$R_{\Omega_s}(\Omega) = \begin{cases} 1 & |\Omega| < \Omega_s / 2 \\ 0 & \text{otherwise} \end{cases} \quad (5.160)$$

is an ideal lowpass filter. We can also see that if Eq. (5.158) is not met, then *aliasing* will take place and  $X(\Omega)$  can no longer be recovered from  $X_p(\Omega)$ . Since, in general, we cannot be certain that Eq. (5.158) is true, the analog signal is low-pass filtered with an ideal filter given by Eq. (5.160), which is called anti-aliasing filter, prior to sampling. Limiting the bandwidth of our analog signal is the price we have to pay to be able to manipulate it digitally.



**Figure 5.27**  $X(\Omega)$ ,  $X_p(\Omega)$  for the case of no aliasing and aliasing.

The inverse Fourier transform of Eq. (5.160), computed through Eq. (5.151), is a sinc function

$$r_T(t) = \text{sinc}(t/T) = \frac{\sin(\pi t/T)}{\pi t/T} \quad (5.161)$$

so that using the convolution property in Eq. (5.159) we obtain

$$x(t) = r_T(t) * x_p(t) = r_T(t) * \sum_{k=-\infty}^{\infty} x[k] \delta(t - kT) = \sum_{k=-\infty}^{\infty} x[k] r_T(t - kT) \quad (5.162)$$

The *sampling theorem* states that we can recover the continuous time signal  $x(t)$  just from its samples  $x[n]$  using Eqs. (5.161) and (5.162). The angular frequency  $\Omega_s = 2\pi F_s$  is expressed in terms of the sampling frequency  $F_s$ .  $T = 1/F_s$  is the sampling period, and  $F_s/2$  the *Nyquist* frequency. Equation (5.162) is referred to as *bandlimited interpolation* because  $x(t)$  is reconstructed by interpolating  $x[n]$  with sinc functions that are bandlimited.

Now let's see the relationship between  $X_p(\Omega)$  and  $X(e^{j\omega})$ , the Fourier transform of the discrete sequence  $x[n]$ . From Eq. (5.154) we have

$$X_p(\Omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\Omega n T} \quad (5.163)$$

so that the continuous transform  $X_p(\Omega)$  equals the discrete Fourier transform  $X(e^{j\omega})$  at  $\omega = \Omega T$ .

### 5.5.3. Analog-to-Digital Conversion

The process of converting an analog signal  $x(t)$  into a digital signal  $x[n]$  is called *Analog-to-Digital conversion*, or A/D for short, and the device that does it called an *Analog-to-Digital Converter*. In Section 5.5.2 we saw that an ideal low-pass anti-aliasing filter was required on the analog signal, which of course is not realizable in practice so that an approximation has to be used. In practice, sharp analog filters can be implemented on the same chip using switched capacitor filters, which have attenuations above 60 dB in the stop band so that aliasing tends not to be an important issue for speech signals. The passband is not exactly flat, but this again does not have much significance for speech signals (for other signals, such as those used in modems, this issue needs to be studied more carefully).

Although such sharp analog filters are possible, they can be expensive and difficult to implement. One common solution involves the use of a simple analog low-pass filter with a large attenuation at  $MF_s/2$ , a multiple of the required cutoff frequency. Then *oversampling* is done at the new rate  $MF_s$ , followed by a sharper digital filter with a cut-off frequency of  $F_s/2$  and downsampling (see Section 5.6). This is equivalent to having used a sharp analog filter, with the advantage of a lower-cost implementation. This method also allows variable sampling rates with minimal increase in cost and complexity. This topic is discussed in more detail in Chapter 7 in the context of sigma-delta modulators.

In addition, the pulses in Eq. (5.59) cannot be zero length in practice, and therefore the sampling theorem does not hold. However, current hardware allows the pulses to be small enough that the analog signal can be approximately recovered. The signal level is then maintained during  $T$  seconds, while the conversion to digital is being carried out.

A real A/D converter cannot provide real numbers for  $x[n]$ , but rather a set of integers typically represented with 16 bits, which gives a range between  $-32,768$  and  $32,767$ . Such conversion is achieved by comparing the analog signal to a number of different signal levels. This means that *quantization noise* has been added to the digital signal. This is typically not a big problem for speech signals if using 16 bits or more since, as is shown in Chapter 7, other noises will mask the quantization noise anyway. Typically, quantization noise becomes an issue only if 12 or fewer bits are used. A more detailed study of the effects of quantization is presented in Chapter 7.

Finally, A/D subsystems are not exactly linear, which adds another source of distortion. This nonlinearity can be caused by, among things, jitter and drift in the pulses and unevenly spaced comparators. For popular A/D subsystems, such as *sigma-delta* A/D, an offset is typically added to  $x[n]$ , which in practice is not very important, because speech signals do not contain information at  $f = 0$ , and thus can be safely ignored.

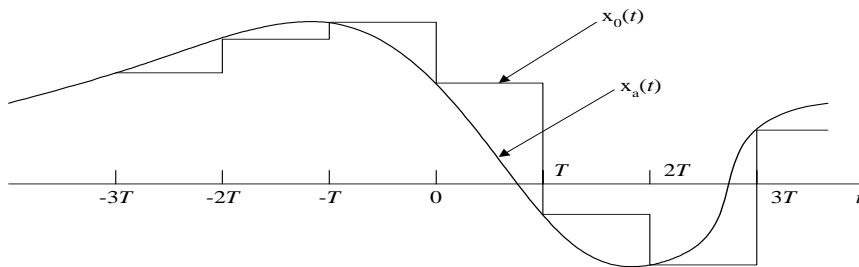
#### 5.5.4. Digital-to-Analog Conversion

The process of converting the digital signal  $x[n]$  back into an analog  $x(t)$  is called *digital-to-analog conversion*, or D/A for short. The ideal band-limited interpolation requires ideal sinc functions as shown in Eq. (5.162), which are not realizable. To convert the digital signal to analog, a zero-order hold filter

$$h_0(t) = \begin{cases} 1 & 0 < t < T \\ 0 & \text{otherwise} \end{cases} \quad (5.164)$$

is often used, which produces an analog signal as shown in Figure 5.28. The output of such a filter is given by

$$x_o(t) = h_0(t) * \sum_{n=-\infty}^{\infty} x[n]\delta(t-nT) = \sum_{n=-\infty}^{\infty} x[n]h_0(t-nT) \quad (5.165)$$



**Figure 5.28** Output of a zero-order hold filter.

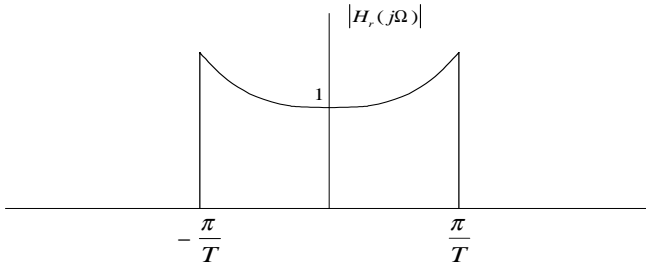
The Fourier transform of the zero-hold filter in Eq. (5.164) is, using Eq. (5.150),

$$H_0(\Omega) = \frac{2 \sin(\Omega T / 2)}{\Omega} e^{-j\Omega T / 2} \quad (5.166)$$

and, since we need an ideal lowpass filter to achieve the band-limited interpolation of Eq. (5.162), the signal  $x_0(t)$  has to be filtered with a reconstruction filter with transfer function

$$H_r(\Omega) = \begin{cases} \frac{\Omega T / 2}{\sin(\Omega T / 2)} e^{j\Omega T / 2} & |\Omega| < \pi / T \\ 0 & |\Omega| > \pi / T \end{cases} \quad (5.167)$$

In practice, the phase compensation is ignored, as it amounts to a delay of  $T/2$  seconds. Its magnitude response can be seen in Figure 5.29. In practice, such an analog filter is not realizable and an approximation is made. Since the zero-order hold filter is already low-pass, the reconstruction filter doesn't need to be that sharp.



**Figure 5.29** Magnitude frequency response of the reconstruction filter used in digital-to-analog converters after a zero-hold filter.

In the above discussion we note that practical A/D and D/A systems introduce distortions, which causes us to wonder whether it is a good idea to go through this process just to manipulate digital signals. It turns out that for most speech processing algorithms described in Chapter 6, the advantages of operating with digital signals outweigh the disadvantage of the distortions described above. Moreover, commercial A/D and D/A systems are such that the errors and distortions can be arbitrarily small. The fact that music in digital format (as in compact discs) has won out over analog format (cassettes) shows that this is indeed the case. Nonetheless, it is important to be aware of the above limitations when designing a system.

## 5.6. MULTIRATE SIGNAL PROCESSING

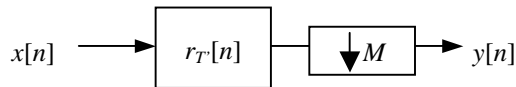
The term *Multirate Signal Processing* refers to processing of signals sampled at different rates. A particularly important problem is that of sampling-rate conversion. It is often the case that we have a digital signal  $x[n]$  sampled at a sampling rate  $F_s$ , and we want to obtain an equivalent signal  $y[n]$  but at a different sampling rate  $F'_s$ . This often occurs in A/D systems that oversample in order to use smaller quantizers, such as a delta or sigma delta-

quantizer (see Chapter 7), and a simpler analog filter, and then have to downsample the signal. Other examples include mixing signals of different sampling rates and downsampling to reduce computation (many signal processing algorithms have a computational complexity proportional to the sampling rate or its square).

A simple solution is to convert the digital signal  $x[n]$  into an analog signal  $x(t)$  with a D/A system running at  $F_s$  and then convert it back to digital with an A/D system running at  $F'_s$ . An interesting problem is whether this could be done in the digital domain directly, and the techniques to do so belong to the general class of multi-rate processing.

### 5.6.1. Decimation

If we want to reduce the sampling rate by a factor of  $M$ , *i.e.*,  $T' = MT$ , we take every  $M$  samples. In order to avoid aliasing, we need to lowpass filter the signal to bandlimit it to frequencies  $1/T'$ . This is shown in Figure 5.30, where the arrow pointing down indicates the decimation.



**Figure 5.30** Block diagram of the decimation process.

Since the output is not desired at all instants  $n$ , but only every  $M$  samples, the computation can be reduced by a factor of  $M$  over the case where lowpass filtering is done first and *decimation* later. To do this we express the analog signal  $x_i(t)$  at the output of the lowpass filter as

$$x_i(t) = \sum_{k=-\infty}^{\infty} x[k]r_{T'}(t - kT) \quad (5.168)$$

and then look at the value  $t' = nT'$ . The decimated signal  $y[n]$  is then given by

$$y[n] = x_i(nT') = \sum_{k=-\infty}^{\infty} x[k]r_{T'}(nT' - kT) = \sum_{k=-\infty}^{\infty} x[k]\text{sinc}\left(\frac{Mn - k}{M}\right) \quad (5.169)$$

which can be expressed as

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[Mn - k] \quad (5.170)$$

where

$$h[n] = \text{sinc}(n/M) \quad (5.171)$$



In practice, the ideal lowpass filter  $h[n]$  is approximated by an FIR filter with a cutoff frequency of  $1/(2M)$ .

### 5.6.2. Interpolation

If we want to increase the sampling rate by a factor of  $N$ , so that  $T' = T/N$ , we do not have any aliasing and no further filtering is necessary. In fact we already know one out of every  $N$  output samples

$$y[Nn] = x[n] \tag{5.172}$$

and we just need to compute the  $(N - 1)$  samples in-between. Since we know that  $x[n]$  is a bandlimited signal, we can use the sampling theorem in Eq. (5.162) to reconstruct the analog signal as

$$x_t(t) = \sum_{k=-\infty}^{\infty} x[k]r_T(t - kT) \tag{5.173}$$

and thus the interpolated signal  $y[n]$  as

$$y[n] = x(nT') = \sum_{k=-\infty}^{\infty} x[k]r_T(nT' - kT) = \sum_{k=-\infty}^{\infty} x[k]\text{sinc}\left(\frac{n - kN}{N}\right) \tag{5.174}$$

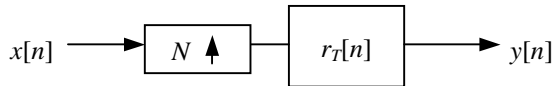
Now let's define

$$x'[k'] = \begin{cases} x[Nk] & k' = Nk \\ 0 & \text{otherwise} \end{cases} \tag{5.175}$$

which, inserted into Eq. (5.174), gives

$$y[n] = \sum_{k'=-\infty}^{\infty} x'[k']\text{sinc}((n - k')/N) \tag{5.176}$$

This can be seen in Figure 5.31, where the block with the arrow pointing up implements Eq. (5.175).



**Figure 5.31** Block diagram of the interpolation process.

Equation (5.174) can be expressed as

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n - kN] \tag{5.177}$$

where we have defined

$$h[n] = \text{sinc}(n/N) \quad (5.178)$$

Again, in practice, the ideal low-pass filter  $h[n]$  is approximated by an FIR filter with a cutoff frequency of  $1/(2N)$ .

### 5.6.3. Resampling

To resample the signal so that  $T' = TM/N$ , or  $F'_s = F_s(N/M)$ , we can first upsample the signal by  $N$  and then downsample it by  $M$ . However, there is a more efficient way. Proceeding similarly to decimation and interpolation, one can show the output is given by

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[nM - kN] \quad (5.179)$$

where

$$h[n] = \text{sinc}\left(\frac{n}{\max(N, M)}\right) \quad (5.180)$$

for the ideal case. In practice,  $h[n]$  is an FIR filter with a cutoff frequency of  $1/(2\max(N, M))$ . We can see that Eq. (5.179) is a superset of Eqs. (5.170) and (5.177).

## 5.7. FILTERBANKS

A filterbank is a collection of filters that span the whole frequency spectrum. In this section we describe the fundamentals of filterbanks, which are used in speech and audio coding, echo cancellation, and other applications. We first start with a filterbank with two equal bands, then explain multi-resolution filterbanks, and present the FFT as a filterbank. Finally we introduce the concept of lapped transforms and wavelets.

### 5.7.1. Two-Band Conjugate Quadrature Filters

A two-band filterbank is shown in Figure 5.32, where the filters  $f_0[n]$  and  $g_0[n]$  are low-pass filters, and the filters  $f_1[n]$  and  $g_1[n]$  are high-pass filters, as shown in Figure 5.33. Since the output of  $f_0[n]$  has a bandwidth half of that of  $x[n]$ , we can sample it at half the rate of  $x[n]$ . We do that by decimation (throwing out every other sample), as shown in Figure 5.32. The output of such a filter plus decimation is  $x_0[m]$ . Similar results can be shown for  $f_1[n]$  and  $x_1[n]$ .

For reconstruction, we upsample  $x_0[m]$ , by inserting a 0 between every sample. Then we low-pass filter it with filter  $g_0[n]$  to complete the interpolation, as we saw in Section 5.6. A similar process can be done with the high pass filters  $f_1[n]$  and  $g_1[n]$ . Adding the two bands produces  $\tilde{x}[n]$ , which is identical to  $x[n]$  if the filters are ideal.

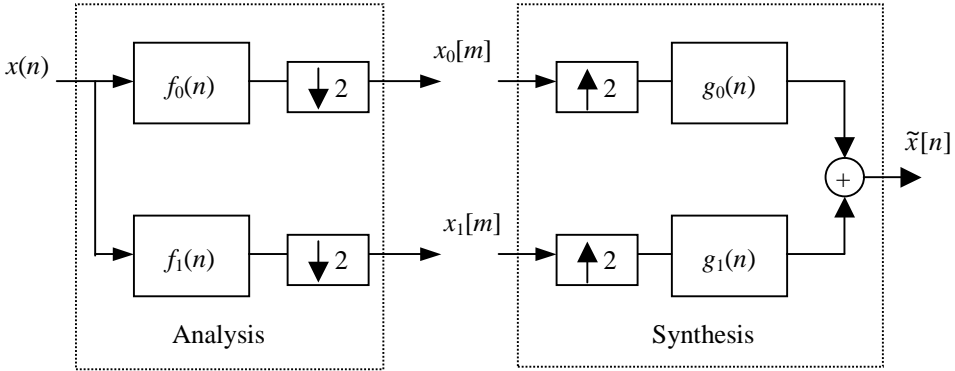


Figure 5.32 Two-band filterbank.

In practice, however, ideal filters such as those in Figure 5.33 are not achievable, so we would like to know if it is possible to build a filterbank that has perfect reconstruction with FIR filters. The answer is affirmative, and in this section we describe conjugate quadrature filters, which are the basis for the solutions.

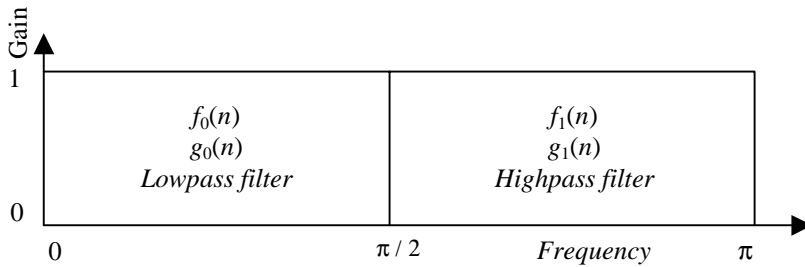


Figure 5.33 Ideal frequency responses of analysis and synthesis filters for the two-band filterbank.

To investigate this, let's analyze the cascade of a downsampler and an upsampler (Figure 5.34). The output  $y[n]$  is a signal whose odd samples are zero and whose even samples are the same as those of the input signal  $x[n]$ .

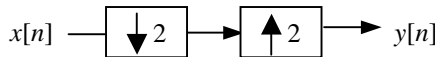


Figure 5.34 Cascade of a downsampler and an upsampler.

The  $z$ -transform of the output is given by

$$\begin{aligned} Y(z) &= \sum_{\substack{n=-\infty \\ \text{even}}}^{\infty} x[n]z^{-n} = \frac{1}{2} \sum_{n=-\infty}^{\infty} x[n]z^{-n} + \frac{1}{2} \sum_{n=-\infty}^{\infty} (-1)^n x[n]z^{-n} \\ &= \frac{X(z) + X(-z)}{2} \end{aligned} \quad (5.181)$$

Using Eq. (5.181) and the system in Figure 5.32, we can express the  $z$ -transform of the output in Figure 5.32 as

$$\begin{aligned} \tilde{X}(z) &= \left( \frac{F_0(z)G_0(z) + F_1(z)G_1(z)}{2} \right) X(z) \\ &+ \left( \frac{F_0(-z)G_0(z) + F_1(-z)G_1(z)}{2} \right) X(-z) \\ &= \left( \frac{F_0(z)X(z) + F_0(-z)X(-z)}{2} \right) G_0(z) + \left( \frac{F_1(z)X(z) + F_1(-z)X(-z)}{2} \right) G_1(z) \end{aligned} \quad (5.182)$$

which for perfect reconstruction requires the output to be a delayed version of the input, and thus

$$\begin{aligned} F_0(z)G_0(z) + F_1(z)G_1(z) &= 2z^{-(L-1)} \\ F_0(-z)G_0(z) + F_1(-z)G_1(z) &= 0 \end{aligned} \quad (5.183)$$

These conditions are met if we select the so-called *Conjugate Quadrature Filters* (CQF) [17], which are FIR filters that specify  $f_1[n]$ ,  $g_0[n]$ , and  $g_1[n]$  as a function of  $f_0[n]$ :

$$\begin{aligned} f_1[n] &= (-1)^n f_0[L-1-n] \\ g_0[n] &= f_0[L-1-n] \\ g_1[n] &= f_1[L-1-n] \end{aligned} \quad (5.184)$$

where  $f_0[n]$  is an FIR filter of even length  $L$ . The  $z$ -transforms of Eq. (5.184) are

$$\begin{aligned} F_1(z) &= z^{-(L-1)} F_0(-z^{-1}) \\ G_0(z) &= z^{-(L-1)} F_0(z^{-1}) \\ G_1(z) &= F_0(-z) \end{aligned} \quad (5.185)$$

so that the second equation in Eq. (5.183) is met if  $L$  is even. In order to analyze the first equation in Eq. (5.183), let's define  $P(z)$  as

$$\begin{aligned}
 P(z) &= F_0(z)F_0(z^{-1}) \\
 p[n] &= \sum_m f_0[m]f_0[m+n]
 \end{aligned}
 \tag{5.186}$$

then insert Eq. (5.185) into (5.183), use Eq. (5.186), and obtain the following condition:

$$P(z) + P(-z) = 2 \tag{5.187}$$

Taking the inverse  $z$ -transform of Eq. (5.186) and using Eq. (5.181), we obtain

$$p[n] = \begin{cases} 1 & n = 0 \\ 0 & n = 2k \end{cases} \tag{5.188}$$

so that all even samples of the autocorrelation of  $f_0[n]$  are zero, except for  $n = 0$ . Since  $f_0[n]$  is a half-band low-pass filter,  $p[n]$  is also a half-band low-pass filter. The ideal half-band filter  $h[n]$

$$h[n] = \frac{\sin(\pi n / 2)}{\pi n} \tag{5.189}$$

satisfies Eq. (5.188), as does any half-band zero-phase filter (a linear phase filter with no delay). Therefore, the steps to build CQF are

1. Design a  $(2L - 1)$  tap<sup>11</sup> half-band linear-phase low-pass filter  $p[n]$  with any available technique, for an even value of  $L$ . For example, one could use the Parks McClellan algorithm, constraining the passband and stopband cutoff frequencies so that  $\omega_p = \pi - \omega_s$  and using an error weighting that is the same for the passband and stopband. This results in a half-band linear-phase filter with equal ripple  $\delta$  in both bands. Another possibility is to multiply the ideal half-band filter in Eq. (5.189) by a window with low-pass characteristics.
2. Add a value  $\delta$  to  $p[0]$  so that we can guarantee that  $P(e^{j\omega}) \geq 0$  for all  $\omega$  and thus is a legitimate power spectral density.
3. Spectrally factor  $P(z) = F_0(z)F_0(z^{-1})$  by computing its roots.
4. Compute  $f_1[n]$ ,  $g_0[n]$  and  $g_1[n]$  from Eq. (5.184).

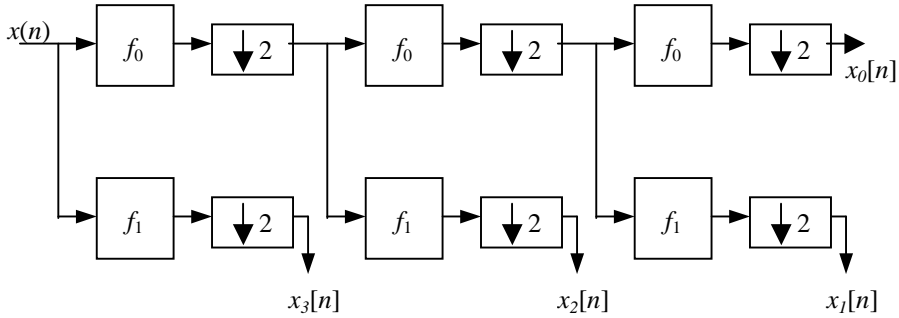
## 5.7.2. Multiresolution Filterbanks

While the above filterbank has equal bandwidth for both filters, it may be desirable to have varying bandwidths, since it has been proven to work better in speech recognition systems. In this section we show how to use the two-band conjugate quadrature filters described in the previous section to design a filterbank with more than two bands. In fact, multi-

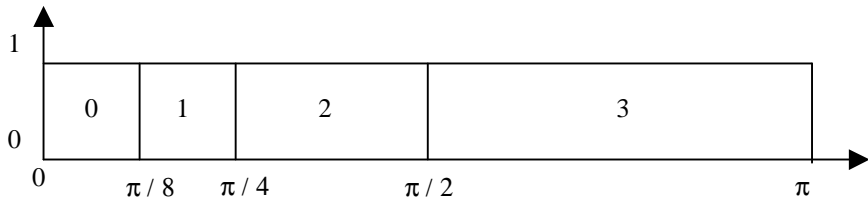
---

<sup>11</sup> A filter with  $N$  taps is a filter of length  $N$ .

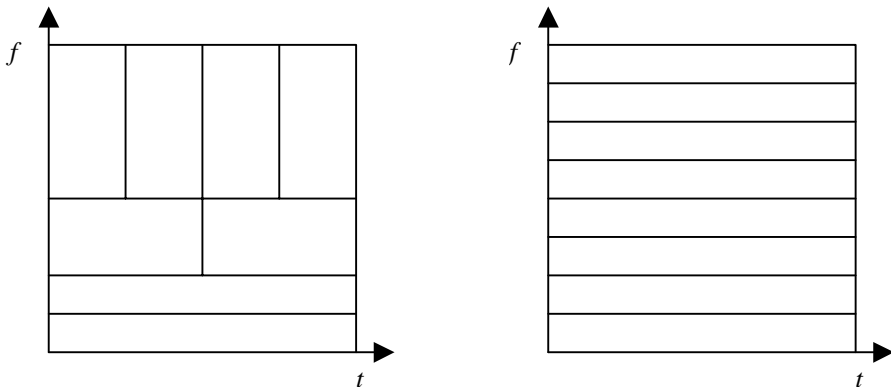
resolution analysis such as that of Figure 5.35, are possible with bands of different bandwidths (see Figure 5.36).



**Figure 5.35** Analysis part of a multi-resolution filterbank designed with conjugate quadrature filters. Only  $f_0[n]$  needs to be specified.



**Figure 5.36** Ideal frequency responses of the multi-resolution filterbank of Figure 5.35. Note that  $x_0[n]$  and  $x_1[n]$  occupy 1/8 of the total bandwidth.



**Figure 5.37** Two different time-frequency tilings: the non-uniform filterbank and that obtain through a short-time Fourier transform. Notice that the area of each tile is constant.

One interesting result is that the product of time resolution and frequency resolution is constant (all the tiles in Figure 5.37 have the same area), since filters with smaller bandwidths do not need to be sampled as often. Instead of using Fourier basis for decomposition, multi-resolution filterbanks allow more flexibility in the tiling of the time-frequency plane.

### 5.7.3. The FFT as a Filterbank

It turns out that we can use the Fourier transform to construct a filterbank. To do that, we decompose the input signal  $x[n]$  as a sum of *short-time* signals  $x_m[n]$

$$x[n] = \sum_{m=-\infty}^{\infty} x_m[n] \quad (5.190)$$

where  $x_m[n]$  is obtained as

$$x_m[n] = x[n]w_m[n] \quad (5.191)$$

the product of  $x[n]$  by a *window* function  $w_m[n]$  of length  $N$ . From Eqs. (5.190) and (5.191) we see that the window function has to satisfy

$$\sum_{m=-\infty}^{\infty} w_m[n] = 1 \quad \forall n \quad (5.192)$$

If the short-term signals  $x_m[n]$  are spaced  $M$  samples apart, we define the window  $w_m[n]$  as:

$$w_m[n] = w[n - Mm] \quad (5.193)$$

where  $w[n] = 0$  for  $n < 0$  and  $n > N$ . The windows  $w_m[n]$  overlap in time while satisfying Eq. (5.192).

Since  $x_m[n]$  has  $N$  nonzero values, we can evaluate its length- $N$  DFT as

$$\begin{aligned} X_m[k] &= \sum_{l=0}^{N-1} x_m[Mm+l]e^{-j\omega_k l} \\ &= \sum_{l=0}^{N-1} x[Mm+l]w[l]e^{-j\omega_k l} = \sum_{l=0}^{N-1} x[Mm+l]f_k[-l] \end{aligned} \quad (5.194)$$

where  $\omega_k = 2\pi k / N$  and the analysis filters  $f_k[l]$  are given by

$$f_k[l] = w[-l]e^{j\omega_k l} \quad (5.195)$$

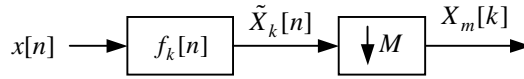
If we define  $\tilde{X}_k[n]$  as

$$\tilde{X}_k[n] = x[n] * f_k[n] = \sum_{r=-\infty}^{\infty} x[n-r]f_k[r] = \sum_{l=0}^{N-1} x[n+l]f_k[-l] \quad (5.196)$$

then Eqs. (5.194) and (5.196) are related by

$$X_m[k] = \tilde{X}_k[mM] \quad (5.197)$$

This manipulation is shown in Figure 5.38, so that the DFT output  $X_m[k]$  is  $\tilde{X}_k[n]$  decimated by  $M$ .



**Figure 5.38** Fourier analysis used to build a linear filter.

The short-time signal  $x_m[n]$  can be recovered through the inverse DFT of  $X_m[k]$  as

$$x_m[mM + l] = h[l] \sum_{k=0}^{N-1} X_m[k] e^{j\omega_k l} \quad (5.198)$$

where  $h[n]$  has been defined as

$$h[n] = \begin{cases} 1/N & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases} \quad (5.199)$$

so that Eq. (5.198) is valid for all values of  $l$ , and not just  $0 \leq l < N$ .

Making the change of variables  $mM + l = n$  in Eq. (5.198) and inserting it into Eq. (5.190) results in

$$\begin{aligned} x[n] &= \sum_{m=-\infty}^{\infty} h[n-mM] \sum_{k=0}^{N-1} X_m[k] e^{j\omega_k (n-mM)} \\ &= \sum_{k=0}^{N-1} \sum_{m=-\infty}^{\infty} X_m[k] g_k[n-mM] \end{aligned} \quad (5.200)$$

where the synthesis filters  $g_k[n]$  are defined as

$$g_k[n] = h[n] e^{j\omega_k n} \quad (5.201)$$

Now, let's define the upsampled version of  $X_m[k]$  as

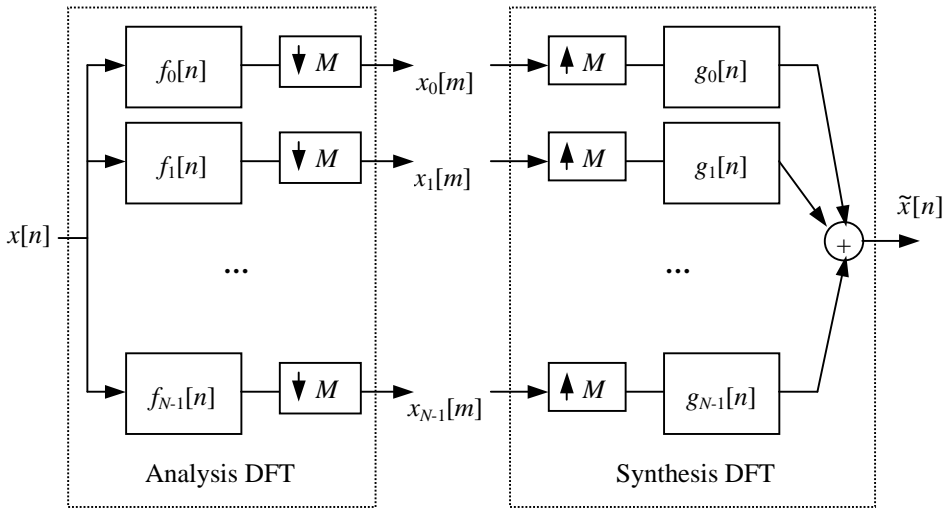
$$\hat{X}_k[l] = \begin{cases} X_m[k] & l = mM \\ 0 & \text{otherwise} \end{cases} \quad (5.202)$$



which, inserted into Eq. (5.200), yields

$$x[n] = \sum_{k=0}^{N-1} \sum_{l=-\infty}^{\infty} \hat{X}_k[l] g_k[n-l] = \sum_{k=0}^{N-1} \hat{X}_k[n] * g_k[n] \quad (5.203)$$

Thus, the signal can be reconstructed. The block diagram of the analysis/resynthesis filterbank implemented by the DFT can be seen in Figure 5.39, where  $x_k[m] = X_m[k]$  and  $\tilde{x}[n] = x[n]$ .



**Figure 5.39** A filterbank with  $N$  analysis and synthesis filters.

For perfect reconstruction we need  $N \geq M$ . If  $w[n]$  is a rectangular window of length  $N$ , the frame rate has to be  $M = N$ . We can also use overlapping windows with  $N = 2M$  (50% overlap), such as Hamming or Hanning windows, and still get perfect reconstruction. The use of such overlapping windows increases the data rate by a factor of 2, but the analysis filters have much less spectral leakage because of the higher attenuation of the Hamming/Hanning window outside the main lobe.

#### 5.7.4. Modulated Lapped Transforms

The filterbank of Figure 5.39 is useful because, as we see in Chapter 7, it is better to quantize the spectral coefficients than the waveform directly. If the DFT coefficients are quantized, there will be some discontinuities at frame boundaries. To solve this problem we can distribute the window  $w[n]$  between the analysis and synthesis filters so that

$$w[n] = w_a[n] w_s[n] \quad (5.204)$$

so that the analysis filters are given by

$$f_k[n] = w_a[-n]e^{j\omega_k n} \quad (5.205)$$

and the synthesis filters by

$$g_k[n] = w_s[n]e^{-j\omega_k n} \quad (5.206)$$

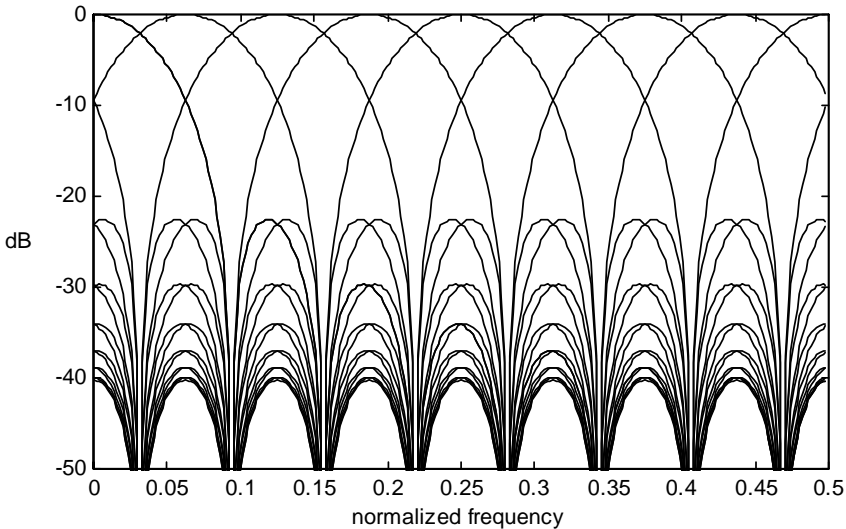
This way, if there is a quantization error, the use of a tapering synthesis window will substantially decrease the border effect. A common choice is  $w_a[n] = w_s[n]$ , which for the case of  $w[n]$  being a Hanning window divided by  $N$ , results in

$$w_a[n] = w_s[n] = \frac{1}{\sqrt{N}} \sin\left(\frac{\pi n}{N}\right) \quad \text{for } 0 \leq n < N \quad (5.207)$$

so that the analysis and synthesis filters are the reversed versions of each other:

$$f_k[-n] = g_k[n] = \frac{\sin(\pi n / N)}{\sqrt{N}} e^{j2\pi nk / N} \Pi_N[n] = h_k^N[n] \quad (5.208)$$

whose frequency response can be seen in Figure 5.40.

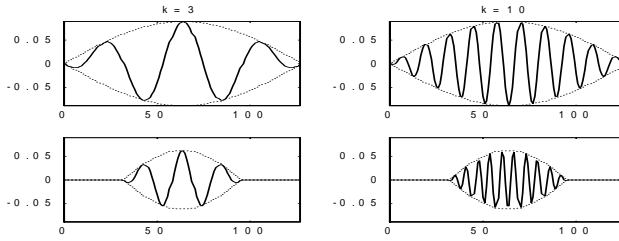


**Figure 5.40** Frequency response of the Lapped Orthogonal Transform filterbank.

The functions  $h_k^N[n]$  in Eq. (5.208) are sine modulated complex exponentials, which have the property

$$h_k^{N/2}[n] = 2^{-1/2} h_k^N[2n] \quad (5.209)$$

which is a property typical of functions called *wavelets*, i.e., they can be obtained from each other by stretching by 2 and scaling them appropriately. Such wavelets can be seen in Figure 5.41.



**Figure 5.41** Iterations of the wavelet  $h_k^N[n]$  for several values of  $k$  and  $N$ .

If instead of modulating a complex exponential we use a cosine sequence, we obtain the *Modulated Lapped Transform* (MLT) [7], also known as the *Modified Discrete Cosine Transform* (MDCT):

$$p_{kn} = f_k[2M - 1 - n] = g_k[n] = h[n] \sqrt{\frac{2}{M}} \cos \left[ \left( k + \frac{1}{2} \right) \left( n + \frac{M + 1}{2} \right) \frac{\pi}{M} \right] \quad (5.210)$$

for  $k = 0, 1, \dots, M - 1$  and  $n = 0, 1, \dots, 2M - 1$ . There are  $M$  filters with  $2M$  taps each, and  $h[n]$  is a symmetric window  $h[n] = h[2M - 1 - n]$  that satisfies

$$h^2[n] + h^2[n + M] = 1 \quad (5.211)$$

where the most common choice for  $h[n]$  is

$$h[n] = \sin \left[ \left( n + \frac{1}{2} \right) \frac{\pi}{2M} \right] \quad (5.212)$$

A fast algorithm can be used to compute these filters based on the DCT, which is called the *Lapped Orthogonal Transform* (LOT).

## 5.8. STOCHASTIC PROCESSES

While in this chapter we have been dealing with *deterministic signals*, we also need to deal with *noise*, such as the static present in a poorly tuned AM station. To analyze noise signals we need to introduce the concept of stochastic processes, also known as random processes. A *discrete-time* stochastic process  $\mathbf{x}[n]$ , also denoted by  $\mathbf{x}_n$ , is a sequence of random variables for each time instant  $n$ . *Continuous-time* stochastic processes  $\mathbf{x}(t)$ , random variables for each value of  $t$ , will not be the focus of this book, though their treatment is similar to that of discrete-time processes. We use bold for random variables and regular text for deterministic signals.

Here, we cover the statistics of stochastic processes, defining stationary and ergodic processes and the output of linear systems to such processes.

---

### Example

We can define a random process  $\mathbf{x}[n]$  as

$$\mathbf{x}[n] = \cos[\omega n + \phi] \quad (5.213)$$

where  $\phi$  is real random variable with a uniform pdf in the interval  $(-\pi, \pi)$ . Several realizations of this random process are displayed in Figure 5.42.

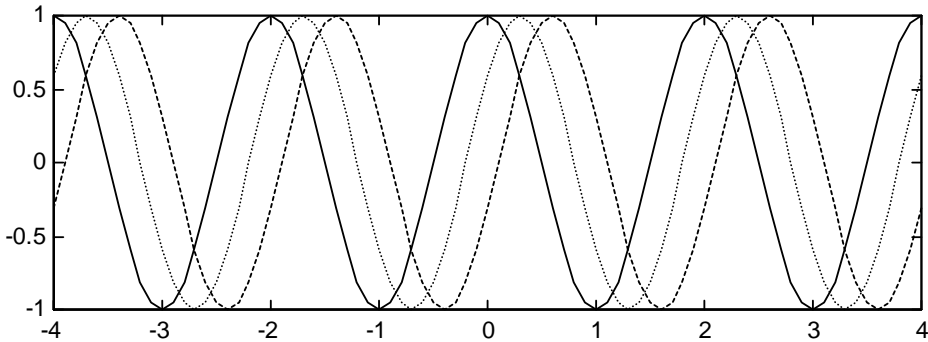


Figure 5.42 Several realizations of a sinusoidal random process with a random phase.

---

## 5.8.1. Statistics of Stochastic Processes

In this section we introduce several statistics of stochastic processes such as distribution, density function, mean and autocorrelation. We also define several types of processes depending on these statistics.

For a specific  $n$ ,  $\mathbf{x}[n]$  is a random variable with *distribution*

$$F(x, n) = P\{\mathbf{x}[n] \leq x\} \quad (5.214)$$

Its first derivative with respect to  $x$  is the first-order density function, or simply the *probability density function* (pdf)

$$f(x, n) = \frac{dF(x, n)}{dx} \quad (5.215)$$

The second-order distribution of the process  $\mathbf{x}[n]$  is the joint distribution

$$F(x_1, x_2; n_1, n_2) = P\{\mathbf{x}[n_1] \leq x_1, \mathbf{x}[n_2] \leq x_2\} \quad (5.216)$$

of the random variables  $\mathbf{x}[n_1]$  and  $\mathbf{x}[n_2]$ . The corresponding density equals

$$f(x_1, x_2; n_1, n_2) = \frac{\partial^2 F(x_1, x_2; n_1, n_2)}{\partial x_1 \partial x_2} \quad (5.217)$$

A complex random process  $\mathbf{x}[n] = \mathbf{x}_r[n] + j\mathbf{x}_i[n]$  is specified in terms of the joint statistics of the real processes  $\mathbf{x}_r[n]$  and  $\mathbf{x}_i[n]$ .

The *mean*  $\mu[n]$  of  $\mathbf{x}[n]$ , also called first-order moment, is defined as the expected value of the random variable  $\mathbf{x}[n]$  for each value of  $n$ :

$$\mu_x[n] = E\{\mathbf{x}[n]\} = \int_{-\infty}^{\infty} \mathbf{x}[n] f(\mathbf{x}, n) d\mathbf{x} \quad (5.218)$$

The autocorrelation of complex random process  $\mathbf{x}[n]$ , also called second-order moment, is defined as

$$R_{xx}[n_1, n_2] = E\{\mathbf{x}[n_1] \mathbf{x}^*[n_2]\} = R_{xx}^*[n_2, n_1] \quad (5.219)$$

which is a statistical average, unlike the autocorrelation of a deterministic signal defined in Eq. (5.45), which was an average over time.

### Example

Let's look at the following sinusoidal random process

$$\mathbf{x}[n] = \mathbf{r} \cos[\omega n + \phi] \quad (5.220)$$

where  $\mathbf{r}$  and  $\phi$  are independent and  $\phi$  is uniform in the interval  $(-\pi, \pi)$ . This process is *zero-mean* because

$$\mu_x[n] = E\{\mathbf{r} \cos[\omega n + \phi]\} = E\{\mathbf{r}\} E\{\cos[\omega n + \phi]\} = 0 \quad (5.221)$$

since  $\mathbf{r}$  and  $\phi$  are independent and

$$E\{\cos[\omega n + \phi]\} = \int_{-\pi}^{\pi} \cos[\omega n + \phi] \frac{1}{2\pi} d\phi = 0 \quad (5.222)$$

Its autocorrelation is given by

$$\begin{aligned} R_{xx}[n_1, n_2] &= E\{\mathbf{r}^2\} \int_{-\pi}^{\pi} \cos[\omega n_1 + \phi] \cos[\omega n_2 + \phi] \frac{1}{2\pi} d\phi \\ &= \frac{1}{2} E\{\mathbf{r}^2\} \int_{-\pi}^{\pi} \{\cos[\omega(n_1 + n_2) + \phi] + \cos[\omega(n_2 - n_1)]\} \frac{1}{2\pi} d\phi \\ &= \frac{1}{2} E\{\mathbf{r}^2\} \cos[\omega(n_2 - n_1)] \end{aligned} \quad (5.223)$$

which only depends on the time difference  $n_2 - n_1$ .

An important property of a stochastic process is that its autocorrelation  $R_{xx}[n_1, n_2]$  is a *positive-definite* function, i.e., for any  $a_i, a_j$

$$\sum_i \sum_j a_i a_j^* R_{xx}[n_i, n_j] \geq 0 \quad (5.224)$$

which is a consequence of the identity

$$0 \leq E \left\{ \left| \sum_i a_i \mathbf{x}[n_i] \right|^2 \right\} = \sum_i \sum_j a_i a_j^* E \{ \mathbf{x}[n_i] \mathbf{x}^*[n_j] \} \quad (5.225)$$

Similarly, the autocovariance of a complex random process is defined as

$$C_{xx}[n_1, n_2] = E \{ (\mathbf{x}[n_1] - \mu_x[n_1]) (\mathbf{x}[n_2] - \mu_x[n_2])^* \} = R_{xx}[n_1, n_2] - \mu_x[n_1] \mu_x^*[n_2] \quad (5.226)$$

The correlation coefficient of process  $\mathbf{x}[n]$  is defined as

$$r_{xx}[n_1, n_2] = \frac{C_{xx}[n_1, n_2]}{\sqrt{C_{xx}[n_1, n_1] C_{xx}[n_2, n_2]}} \quad (5.227)$$

An important property of the correlation coefficient is that it is bounded by 1:

$$|r_{xx}[n_1, n_2]| \leq 1 \quad (5.228)$$

which is the *Cauchy-Schwarz* inequality. To prove it, we note that for any real number  $a$

$$\begin{aligned} 0 &\leq E \left\{ \left| a(\mathbf{x}[n_1] - \mu[n_1]) + (\mathbf{x}[n_2] - \mu[n_2]) \right|^2 \right\} \\ &= a^2 C_{xx}[n_1, n_1] + 2a C_{xx}[n_1, n_2] + C_{xx}[n_2, n_2] \end{aligned} \quad (5.229)$$

Since the quadratic function in Eq. (5.229) is positive for all  $a$ , its roots have to be complex, and thus its discriminant has to be negative:

$$C_{xx}^2[n_1, n_2] - C_{xx}[n_1, n_1] C_{xx}[n_2, n_2] \leq 0 \quad (5.230)$$

from which Eq. (5.228) is derived.

The cross-correlation of two stochastic processes  $\mathbf{x}[n]$  and  $\mathbf{y}[n]$  is defined as

$$R_{xy}[n_1, n_2] = E \{ \mathbf{x}[n_1] \mathbf{y}^*[n_2] \} = R_{yx}^*[n_2, n_1] \quad (5.231)$$

where we have explicitly indicated with subindices the random process. Similarly, their cross-covariance is

$$C_{xy}[n_1, n_2] = R_{xy}[n_1, n_2] - \mu_x[n_1]\mu_y^*[n_2] \quad (5.232)$$

Two processes  $\mathbf{x}[n]$  and  $\mathbf{y}[n]$  are called *orthogonal* iff

$$R_{xy}[n_1, n_2] = 0 \quad \text{for every } n_1 \text{ and } n_2 \quad (5.233)$$

They are called *uncorrelated* iff

$$C_{xy}[n_1, n_2] = 0 \quad \text{for every } n_1 \text{ and } n_2 \quad (5.234)$$

*Independent processes.* If two processes  $\mathbf{x}[n]$  and  $\mathbf{y}[n]$  are such that the random variables  $\mathbf{x}[n_1], \mathbf{x}[n_2], \dots, \mathbf{x}[n_m]$ , and  $\mathbf{y}[n'_1], \mathbf{y}[n'_2], \dots, \mathbf{y}[n'_m]$  are mutually independent, then these processes are called independent. If two processes are independent, then they are also uncorrelated, though the converse is not generally true.

*Gaussian processes.* A process  $\mathbf{x}[n]$  is called Gaussian if the random variables  $\mathbf{x}[n_1], \mathbf{x}[n_2], \dots, \mathbf{x}[n_m]$  are jointly Gaussian for any  $m$  and  $n_1, n_2, \dots, n_m$ . If two processes are Gaussian and also uncorrelated, then they are also statistically independent.

## 5.8.2. Stationary Processes

Stationary processes are those whose statistical properties do not change over time. While truly stationary processes do not exist in speech signals, they are a reasonable approximation and have the advantage of allowing us to use the Fourier transforms defined in Section 5.1.3.3. In this section we define stationarity and analyze some of its properties.

A stochastic process is called *strict-sense stationary* (SSS) if its statistical properties are invariant to a shift of the origin: *i.e.*, both processes  $\mathbf{x}[n]$  and  $\mathbf{x}[n+l]$  have the same statistics for any  $l$ . Likewise, two processes  $\mathbf{x}[n]$  and  $\mathbf{y}[n]$  are called *jointly strict-sense stationary* if their joint statistics are the same as those of  $\mathbf{x}[n+l]$  and  $\mathbf{y}[n+l]$  for any  $l$ .

From the definition, it follows that the  $m^{\text{th}}$ -order density of an SSS process must be such that

$$f(x_1, \dots, x_m; n_1, \dots, n_m) = f(x_1, \dots, x_m; n_1 + l, \dots, n_m + l) \quad (5.235)$$

for any  $l$ . Thus the first-order density satisfies  $f(x, n) = f(x, n+l)$  for any  $l$ , which means that it is independent of  $n$ :

$$f(x, n) = f(x) \quad (5.236)$$

or, in other words, the density function is constant with time.

Similarly,  $f(x_1, x_2; n_1 + l, n_2 + l)$  is independent of  $l$ , which leads to the conclusion

$$f(x_1, x_2; n_1, n_2) = f(x_1, x_2; m) \quad m = n_1 - n_2 \quad (5.237)$$

or, in other words, the joint density of  $\mathbf{x}[n]$  and  $\mathbf{x}[n+m]$  is not a function of  $n$ , only of  $m$ , the time difference between the two samples.

Let's compute the first two moments of a SSS process:

$$E\{x[n]\} = \int x[n]f(x[n]) = \int xf(x) = \mu \quad (5.238)$$

$$E\{x[n+m]x^*[n]\} = \int x[n+m]x^*[n]f(x[n+m], x[n]) = R_{xx}[m] \quad (5.239)$$

or, in other words, its mean is not a function of time and its autocorrelation depends only on  $m$ .

A stochastic process  $\mathbf{x}[n]$  that obeys Eq. (5.238) and (5.239) is called *wide-sense stationary* (WSS). From this definition, a SSS process is also a WSS process but the converse is not true in general. Gaussian processes are an important exception, and it can be proved that a WSS Gaussian process is also SSS.

For example, the random process of Eq. (5.213) is WSS, because it has zero mean and its autocorrelation function, as given by Eq. (5.223), is only a function of  $m = n_1 - n_2$ . By setting  $m = 0$  in Eq. (5.239) we see that the average power of a WSS stationary process

$$E\{|x[n]|^2\} = R[0] \quad (5.240)$$

is independent of  $n$ .

The autocorrelation of a WSS process is a conjugate-symmetric function, also referred to as a *Hermitian* function:

$$R[-m] = E\{x[n-m]x^*[n]\} = E\{x[n]x^*[n+m]\} = R^*[m] \quad (5.241)$$

so that if  $x[n]$  is real,  $R[m]$  is even.

From Eqs. (5.219), (5.238), and (5.239) we can compute the autocovariance as

$$C[m] = R[m] - |\mu|^2 \quad (5.242)$$

and its correlation coefficient as

$$r[m] = C[m] / C[0] \quad (5.243)$$

Two processes  $\mathbf{x}[n]$  and  $\mathbf{y}[n]$  are called jointly WSS if both are WSS and their cross-correlation depends only on  $m = n_1 - n_2$ :

$$R_{xy}[m] = E\{x[n+m]y^*[n]\} \quad (5.244)$$

$$C_{xy}[m] = R_{xy}[m] - \mu_x \mu_y^* \quad (5.245)$$



### 5.8.2.1. Ergodic Processes

A critical problem in the theory of stochastic processes is the estimation of their various statistics, such as the mean and autocorrelation given that often only one realization of the random process is available. The first approximation would be to replace the expectation in Eq. (5.218) with its *ensemble average*:

$$\mu[n] \cong \frac{1}{M} \sum_{i=0}^{M-1} x_i[n] \quad (5.246)$$

where  $x_i[n]$  are different samples of the random process.

As an example, let  $\mathbf{x}[n]$  be the frequency-modulated (FM) random process received by a FM radio receiver:

$$\mathbf{x}[n] = a[n] + \mathbf{v}[n] \quad (5.247)$$

which contains some additive noise  $\mathbf{v}[n]$ . The realization  $x_i[n]$  received by receiver  $i$  will be different from the realization  $x_j[n]$  for receiver  $j$ . We know that each signal has a certain level of noise, so one would hope that by averaging them, we could get the mean of the process for a sufficiently large number of radio receivers.

In many cases, however, only one sample of the process is available. According to Eq. (5.246) this would mean that the sample signal equals the mean, which does not seem very robust. We could also compute the signal's time average, but this may not tell us much about the random process in general. However, for a special type of random processes called ergodic, their ensemble averages equal appropriate time averages.

A process  $\mathbf{x}[n]$  with constant mean

$$E\{\mathbf{x}[n]\} = \mu \quad (5.248)$$

is called *mean-ergodic* if, with probability 1, the ensemble average equals the time average when  $N$  approaches infinity:

$$\lim_{N \rightarrow \infty} \mu_N = \mu \quad (5.249)$$

where  $\mu_N$  is the time average

$$\mu_N = \frac{1}{N} \sum_{n=-N/2}^{N/2-1} \mathbf{x}[n] \quad (5.250)$$

which, combined with Eq. (5.248), indicates that  $\mu_N$  is a random variable with mean  $\mu$ . Taking expectations in Eq. (5.250) and using Eq. (5.248), it is clear that

$$E\{\mu_N\} = \mu \quad (5.251)$$

so that proving Eq. (5.249) is equivalent to proving

$$\lim_{N \rightarrow \infty} \sigma_N^2 = 0 \quad (5.252)$$

with  $\sigma_N^2$  being the variance of  $\mu_N$ . It can be shown [12] that a process  $\mathbf{x}[n]$  is mean ergodic iff

$$\lim_{N \rightarrow \infty} \frac{1}{N^2} \sum_{n=-N/2}^{N/2-1} \sum_{m=-N/2}^{N/2-1} C_{xx}[n, m] = 0 \quad (5.253)$$

It can also be shown [12] that a sufficient condition for a WSS process to be mean ergodic is to satisfy

$$\lim_{m \rightarrow \infty} C_{xx}[m] = 0 \quad (5.254)$$

which means that if the random variables  $\mathbf{x}[n]$  and  $\mathbf{x}[n+m]$  are uncorrelated for large  $m$ , then process  $\mathbf{x}[n]$  is mean ergodic. This is true for many regular processes.

A similar condition can be proven for a WSS process to be covariance ergodic. In most cases in this book we assume ergodicity, first because of convenience for mathematical tractability, and second because it is a good approximation to assume that samples that are far apart are uncorrelated. *Ergodicity allows us to compute means and covariances of random processes by their time averages.*

### 5.8.3. LTI Systems with Stochastic Inputs

If the WSS random process  $\mathbf{x}[n]$  is the input to an LTI system with impulse response  $h[n]$ , the output

$$\mathbf{y}[n] = \sum_{m=-\infty}^{\infty} h[m]\mathbf{x}[n-m] = \sum_{m=-\infty}^{\infty} h[n-m]\mathbf{x}[m] \quad (5.255)$$

is another WSS random process. To prove this we need to show that the mean is not a function of  $n$ :

$$\mu_y[n] = E\{\mathbf{y}[n]\} = \sum_{m=-\infty}^{\infty} h[m]E\{\mathbf{x}[n-m]\} = \mu_x \sum_{m=-\infty}^{\infty} h[m] \quad (5.256)$$

The cross-correlation between input and output is given by

$$\begin{aligned} R_{xy}[m] &= E\{\mathbf{x}[n+m]\mathbf{y}^*[n]\} = \sum_{l=-\infty}^{\infty} h^*[l]E\{\mathbf{x}[n+m]\mathbf{x}^*[n-l]\} \\ &= \sum_{l=-\infty}^{\infty} h^*[l]R_{xx}[m+l] = \sum_{l=-\infty}^{\infty} h^*[-l]R_{xx}[m-l] = h^*[-m] * R_{xx}[m] \end{aligned} \quad (5.257)$$

and the autocorrelation of the output

$$\begin{aligned}
R_{yy}[m] &= E\{\mathbf{y}[n+m]\mathbf{y}^*[n]\} = \sum_{l=-\infty}^{\infty} h[l]E\{\mathbf{x}[n+m-l]\mathbf{y}^*[n]\} \\
&= \sum_{l=-\infty}^{\infty} h[l]R_{xy}[m-l] = h[m] * R_{xy}[m] = h[m] * h^*[-m] * R_{xx}[m]
\end{aligned} \tag{5.258}$$

is only a function of  $m$ .

### 5.8.4. Power Spectral Density

The Fourier transform of a WSS random process  $\mathbf{x}[n]$  is a stochastic process in the variable  $\omega$

$$\mathbf{X}(\omega) = \sum_{n=-\infty}^{\infty} \mathbf{x}[n]e^{-j\omega n} \tag{5.259}$$

whose autocorrelation is given by

$$\begin{aligned}
E\{\mathbf{X}(\omega+u)\mathbf{X}^*(\omega)\} &= E\left\{\sum_{l=-\infty}^{\infty} \mathbf{x}[l]e^{-j(\omega+u)l} \sum_{m=-\infty}^{\infty} \mathbf{x}^*[m]e^{j\omega m}\right\} = \\
&= \sum_{n=-\infty}^{\infty} e^{-j(\omega+u)n} \sum_{m=-\infty}^{\infty} E\{\mathbf{x}[m+n]\mathbf{x}^*[m]\}e^{-jum}
\end{aligned} \tag{5.260}$$

where we made a change of variables  $l = n + m$  and changed the order of expectation and summation. Now, if  $\mathbf{x}[n]$  is WSS

$$R_{xx}[n] = E\{\mathbf{x}[m+n]\mathbf{x}^*[m]\} \tag{5.261}$$

and if we set  $u = 0$  in Eq. (5.260) together with Eq. (5.261), then we obtain

$$S_{xx}(\omega) = E\{|\mathbf{X}(\omega)|^2\} = \sum_{n=-\infty}^{\infty} R_{xx}[n]e^{-j\omega n} \tag{5.262}$$

$S_{xx}(\omega)$  is called the *power spectral density* of the WSS random process  $\mathbf{x}[n]$ , and it is the Fourier transform of its autocorrelation function  $R_{xx}[n]$ , with the inversion formula being

$$R_{xx}[n] = \frac{1}{2\pi} \int_{-\infty}^{\infty} S_{xx}(\omega)e^{j\omega n} d\omega \tag{5.263}$$

Note that Eqs. (5.48) and (5.263) are identical, though in one case we compute the autocorrelation of a signal as a time average, and the other is the autocorrelation of a random process as an ensemble average. For an ergodic process both are the same.

Just as we take Fourier transforms of deterministic signals, we can also compute the power spectral density of a random process as long as it is wide-sense stationary, which is why these wide-sense stationary processes are so useful.

If the random process  $\mathbf{x}[n]$  is real then  $R_{xx}[n]$  is real and even and, using properties in Table 5.5,  $S_{xx}(\omega)$  is also real and even.

Parseval's theorem for random processes also applies here:

$$E\left\{|\mathbf{x}[n]|^2\right\} = R_{xx}[0] = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{xx}(\omega) d\omega \quad (5.264)$$

so that we can compute the signal's energy from the area under  $S_{xx}(\omega)$ . Let's get a physical interpretation of  $S_{xx}(\omega)$ . In order to do that we can similarly derive the cross-power spectrum  $S_{xy}(\omega)$  of two WSS random processes  $\mathbf{x}[n]$  and  $\mathbf{y}[n]$  as the Fourier transform of their cross-correlation:

$$S_{xy}(\omega) = \sum_{n=-\infty}^{\infty} R_{xy}[n] e^{-j\omega n} \quad (5.265)$$

which allows us, taking Fourier transforms in Eq. (5.257), to obtain the cross-power spectrum between input and output to a linear system as

$$S_{xy}(\omega) = S_{xx}(\omega) H^*(\omega) \quad (5.266)$$

Now, taking the Fourier transform of Eq. (5.258), the power spectrum of the output is thus given by

$$S_{yy}(\omega) = S_{xy}(\omega) H(\omega) = S_{xx}(\omega) |H(\omega)|^2 \quad (5.267)$$

Finally, suppose we filter  $\mathbf{x}[n]$  through the ideal bandpass filter

$$H_b(\omega) = \begin{cases} \sqrt{\pi/c} & \omega_0 - c < \omega < \omega_0 + c \\ 0 & \text{otherwise} \end{cases} \quad (5.268)$$

The energy of the output process is

$$0 \leq E\left\{|\mathbf{y}[n]|^2\right\} = R_{yy}[0] = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{yy}(\omega) d\omega = \frac{1}{2c} \int_{\omega_0-c}^{\omega_0+c} S_{xx}(\omega) d\omega \quad (5.269)$$

so that taking the limit when  $c \rightarrow 0$  results in

$$0 \leq \lim_{c \rightarrow 0} \frac{1}{2c} \int_{\omega_0-c}^{\omega_0+c} S_{xx}(\omega) d\omega = S_{xx}(\omega_0) \quad (5.270)$$

which is the *Wiener-Khinchin* theorem and says that the power spectrum of a WSS process  $\mathbf{x}[n]$ , real or complex, is always positive for any  $\omega$ . Equation (5.269) also explains the

name power spectral density, because  $S_{xx}(\omega)$  represents the density of power at any given frequency  $\omega$ .

### 5.8.5. Noise

A process  $\mathbf{x}[n]$  is *white noise* if, and only if, its samples are uncorrelated:

$$C_{xx}[n_1, n_2] = C[n_1] \delta[n_1 - n_2] \quad (5.271)$$

and is zero-mean  $\mu_x[n] = 0$ .

If in addition  $\mathbf{x}[n]$  is WSS, then

$$C_{xx}[n] = R_{xx}[n] = q \delta[n] \quad (5.272)$$

which has a flat power spectral density

$$S_{xx}(\omega) = q \quad \text{for all } \omega \quad (5.273)$$

The thermal noise phenomenon in metallic resistors can be accurately modeled as white Gaussian noise. White noise doesn't have to be Gaussian (white Poisson impulse noise is one of many other possibilities).

*Colored noise* is defined as a zero-mean WSS process whose samples are correlated with autocorrelation  $R_{xx}[n]$ . Colored noise can be generated by passing white noise through a filter  $h[n]$  such that  $S_{xx}(\omega) = |H(\omega)|^2$ . A type of colored noise that is very frequently encountered in speech signals is the so-called *pink noise*, whose power spectral density decays with  $\omega$ . A more in-depth discussion of noise and its effect on speech signals is included in Chapter 10.

## 5.9. HISTORICAL PERSPECTIVE AND FURTHER READING

It is impossible to cover the field of Digital Signal Processing in just one chapter. The book by Oppenheim and Shafer [10] is one of the most widely used as a comprehensive treatment. For a more in-depth coverage of digital filter design, you can read the book by Parks and Burrus [13]. A detailed study of the FFT is provided by Burrus and Parks [2]. The theory of signal processing for analog signals can be found in Oppenheim and Willsky [11]. The theory of random signals can be found in Papoulis [12]. Multirate processing is well studied in Crochiere and Rabiner [4]. Razavi [16] covers analog-digital conversion. Software programs, such as MATLAB [1], contain a large number of packaged subroutines. Malvar [7] has extensive coverage of filterbanks and lapped transforms.

The field of Digital Signal Processing has a long history. The greatest advances in the field started in the 17<sup>th</sup> century. In 1666, English mathematician and physicist Sir *Isaac Newton* (1642-1727) invented differential and integral calculus, which was independently

discovered in 1675 by German mathematician *Gottfried Wilhelm Leibniz* (1646-1716). They both developed discrete mathematics and numerical methods to solve such equations when closed-form solutions were not available. In the 18<sup>th</sup> century, these techniques were further extended. Swiss brothers *Johann* (1667-1748) and *Jakob Bernoulli* (1654-1705) invented the calculus of variations and polar coordinates. French mathematician *Joseph Louis Lagrange* (1736-1813) developed algorithms for numerical integration and interpolation of continuous functions. The famous Swiss mathematician *Leonhard Euler* (1707-1783) developed the theory of complex numbers and number theory so useful in the DSP field, in addition to the first full analytical treatment of algebra, the theory of equations, trigonometry and analytical geometry. In 1748, Euler examined the motion of a vibrating string and discovered that sinusoids are eigenfunctions for linear systems. Swiss scientist *Daniel Bernoulli* (1700-1782), son of Johann Bernoulli, also conjectured in 1753 that all physical motions of a string could be represented by linear combinations of normal modes. However, both Euler and Bernoulli, and later Lagrange, discarded the use of trigonometric series because *it was impossible to represent signals with corners*. The 19<sup>th</sup> century brought us the theory of harmonic analysis. One of those who contributed most to the field of Digital Signal Processing is *Jean Baptiste Joseph Fourier* (1768-1830), a French mathematician who in 1822 published *The Analytical Theory of Heat*, where he derived a mathematical formulation for the phenomenon of heat conduction. In this treatise, he also developed the concept of Fourier series and harmonic analysis and the Fourier transform. One of Fourier's disciples, the French mathematician *Simeon-Denis Poisson* (1781-1840), studied the convergence of Fourier series together with countryman *Augustin Louis Cauchy* (1789-1857). Nonetheless, it was German *Peter Dirichlet* (1805-1859) who gave the first set of conditions sufficient to guarantee the convergence of a Fourier series. French mathematician *Pierre Simon Laplace* (1749-1827) invented the Laplace transform, a transform for continuous-time signals over the whole complex plane. French mathematician *Marc-Antoine Parseval* (1755-1836) derived the theorem that carries his name. German *Leopold Kronecker* (1823-1891) did work with discrete delta functions. French mathematician *Charles Hermite* (1822-1901) discovered complex conjugate matrices. American *Josiah Willard Gibbs* (1839-1903) studied the phenomenon of Fourier approximations to periodic square waveforms.

Until the early 1950s, all signal processing was analog, including the *long-playing* (LP) record first released in 1948. Pulse Code Modulation (PCM) had been invented by *Paul M. Rainey* in 1926 and independently by *Alan H. Reeves* in 1937, but it wasn't until 1948 when *Oliver, Pierce, and Shannon* [9] laid the groundwork for PCM (see Chapter 7 for details). Bell Labs engineers developed a PCM system in 1955, the so-called T-1 carrier system, which was put into service in 1962 as the world's first common-carrier digital communications system and is still used today. The year 1948 also saw the invention of the transistor at Bell Labs and a small prototype computer at Manchester University and marked the birth of modern Digital Signal Processing. In 1958, *Jack Kilby* of Texas Instruments invented the integrated circuit and in 1970, researchers at Lincoln Laboratories developed the first real-time DSP computer, which performed signal processing tasks about 100 times faster than general-purpose computers of the time. In 1978, Texas Instruments introduced *Speak & Spell*<sup>TM</sup>, a toy that included an integrated circuit especially designed for speech synthesis. Intel Corporation introduced in 1971 the 4-bit Intel 4004, the first general-purpose

microprocessor chip, and in 1972 they introduced the 8-bit 8008. In 1982 Texas Instruments introduced the TMS32010, the first commercially viable single-chip Digital Signal Processor (DSP), a microprocessor specially designed for fast signal processing operations. At a cost of about \$100, the TMS32010 was a 16-bit fixed-point chip with a hardware multiplier built-in that executed 5 million instructions per second (MIPS). Gordon Moore, Intel's founder, came up with the law that carries his name stating that computing power doubles every 18 months, allowing ever faster processors. By the end of the 20<sup>th</sup> century, DSP chips could perform floating-point operations at a rate over 1000MIPS and had a cost below \$5, so that today they are found in many devices from automobiles to cellular phones.

While hardware improvements significantly enabled the development of the field, digital algorithms were also needed. The 1960s saw the discovery of many of the concepts described in this chapter. In 1965, *James W. Cooley* and *John W. Tukey* [3] discovered the FFT, although it was later found [6] that German mathematician *Carl Friedrich Gauss* (1777-1855) had already invented it over a century earlier. The FFT sped up calculations by orders of magnitude, which opened up many possible algorithms for the slow computers of the time. *James F. Kaiser*, *Bernard Gold*, and *Charles Rader* published key papers on digital filtering. *John Stockham* and *Howard Helms* independently discovered fast convolution by doing convolution with FFTs.

An association that has had a large impact on the development of modern Digital Signal Processing is the Institute of Electrical and Electronic Engineers (IEEE), which has over 350,000 members in 150 nations and is the world's largest technical organization. It was founded in 1884 as the American Institute of Electrical Engineers (AIEE). IEEE's other parent organization, the Institute of Radio Engineers (IRE), was founded in 1912, and the two merged in 1963. The IEEE Signal Processing Society is a society within the IEEE devoted to Signal Processing. Originally founded on 1948 as the Institute of Radio Engineers Professional Group on Audio, it was later renamed the IEEE Group on Audio (1964), the IEEE Audio and Electroacoustics group (1965), the IEEE group on Acoustics Speech and Signal Processing (1974), the Acoustic, Speech and Signal Processing Society (1976), and finally IEEE Signal Processing Society (1989). In 1976 the society initiated its practice of holding an annual conference, the International Conference on Acoustic, Speech and Signal Processing (ICASSP), which has been held every year since, and whose proceedings constitute an invaluable reference. Frederik Nebeker [8] provides a history of the society's first 50 years rich in insights from the pioneers.

## REFERENCES

- [1] Burrus, C.S., *et al.*, *Computer-Based Exercises for Signal Processing Using Matlab*, 1994, Upper Saddle River, NJ, Prentice Hall.
- [2] Burrus, C.S. and T.W. Parks, *DFT/FFT and Convolution Algorithms: Theory and Implementation*, 1985, New York, John Wiley.
- [3] Cooley, J.W. and J.W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, 1965, **19**(Apr.), pp. 297-301.
- [4] Crochiere, R.E. and L.R. Rabiner, *Multirate Digital Signal Processing*, 1983, Upper Saddle River, NJ, Prentice-Hall.

- [5] Duhamel, P. and H. Hollman, "Split Radix FFT Algorithm," *Electronic Letters*, 1984, **20**(January), pp. 14-16.
- [6] Heideman, M.T., D.H. Johnson, and C.S. Burrus, "Gauss and the History of the Fast Fourier Transform," *IEEE ASSP Magazine*, 1984, **1**(Oct), pp. pp 14-21.
- [7] Malvar, H., *Signal Processing with Lapped Transforms*, 1992, Artech House.
- [8] Nebeker, F., *Fifty Years of Signal Processing: The IEEE Signal Processing Society and its Technologies*, 1998, IEEE.
- [9] Oliver, B.M., J.R. Pierce, and C. Shannon, "The Philosophy of PCM," *Proc. Institute of Radio Engineers*, 1948, **36**, pp. pp 1324-1331.
- [10] Oppenheim, A.V. and R.W. Schaffer, *Discrete-Time Signal Processing*, 1999, Prentice-Hall, Upper Saddle River, NJ.
- [11] Oppenheim, A.V. and A.S. Willsky, *Signals and Systems*, 1997, Upper Saddle River, NJ, Prentice-Hall.
- [12] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, 3<sup>rd</sup> ed, 1991, New York, McGraw-Hill.
- [13] Parks, T.W. and C.S. Burrus, *Digital Filter Design*, 1987, New York, John Wiley.
- [14] Parks, T.W. and J.H. McClellan, "A Program for the Design of Linear Phase Finite Impulse Response Filters," *IEEE Trans. on Audio Electroacoustics*, 1972, **AU-20**(Aug), pp. 195-199.
- [15] Rao, K.R. and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages and Applications*, 1990, San Diego, CA, Academic Press.
- [16] Razavi, B., *Principles of Data Conversion System Design*, 1995, IEEE Press.
- [17] Smith, M.J.T. and T.P. Barnwell, "A Procedure for Designing Exact Reconstruction Filter Banks for Tree Structured Subband Coders," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1984, San Diego, Calif pp. 27.1.1-27.1.4.



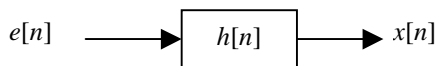
---

# CHAPTER 6

---

## Speech Signal Representations

This chapter presents several representations for speech signals useful in speech coding, synthesis and recognition. The central theme is the decomposition of the speech signal as a source passed through a linear time-varying filter. This filter can be derived from models of speech production based on the theory of acoustics where the source represents the air flow at the vocal cords, and the filter represents the resonances of the vocal tract which change over time. Such a source-filter model is illustrated in Figure 6.1. We describe methods to compute both the source or *excitation*  $e[n]$  and the filter  $h[n]$  from the speech signal  $x[n]$ .



**Figure 6.1** Basic source-filter model for speech signals.

To estimate the filter we present methods inspired by speech production models (such as linear predictive coding and cepstral analysis) as well as speech perception models (such

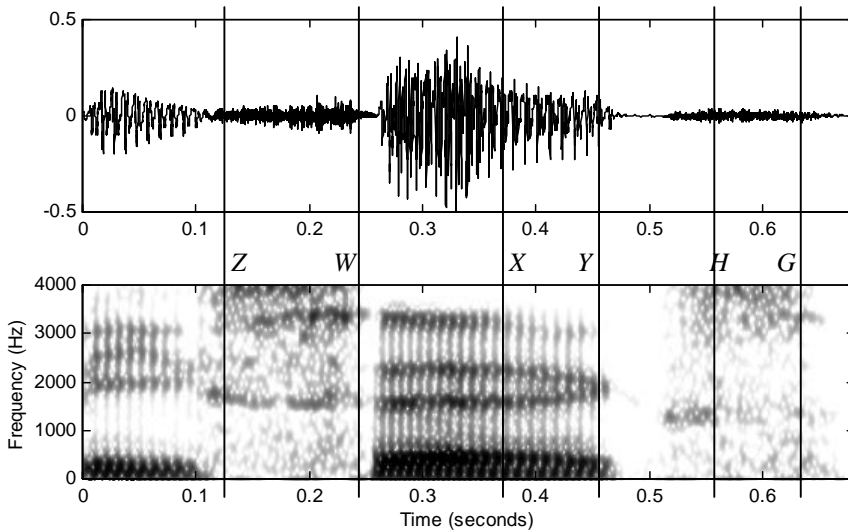
as mel-frequency cepstrum). Once the filter has been estimated, the source can be obtained by passing the speech signal through the inverse filter. Separation between source and filter is one of the most difficult challenges in speech processing.

It turns out that phoneme classification (either by human or by machines) is mostly dependent on the characteristics of the filter. Traditionally, speech recognizers estimate the filter characteristics and ignore the source. Many speech synthesis techniques use a source-filter model because it allows flexibility in altering the pitch and the filter. Many speech coders also use this model because it allows a low bit rate.

We first introduce the spectrogram as a representation of the speech signal that highlights several of its properties and describe the short-time Fourier analysis, which is the basic tool to build the spectrograms of Chapter 2. We then introduce several techniques used to separate source and filter: LPC and cepstral analysis, perceptually motivated models, formant tracking, and pitch tracking.

## 6.1. SHORT-TIME FOURIER ANALYSIS

In Chapter 2, we demonstrated how useful *spectrograms* are to analyze phonemes and their transitions. A spectrogram of a time signal is a special two-dimensional representation that displays time in its horizontal axis and frequency in its vertical axis. A gray scale is typically used to indicate the energy at each point  $(t, f)$  with white representing low energy and black high energy. In this section we cover short-time Fourier analysis, the basic tool with which to compute them.



**Figure 6.2** (a) Waveform with (b) its corresponding wideband spectrogram. Darker areas mean higher energy for that time and frequency. Note the vertical lines spaced by pitch peri-

ods.

The idea behind a spectrogram, such as that in Figure 6.2, is to compute a Fourier transform every 5 milliseconds or so, displaying the energy at each time/frequency point. Since some regions of speech signals shorter than, say, 100 milliseconds often appear to be periodic, we use the techniques discussed in Chapter 5. However, the signal is no longer periodic when longer segments are analyzed, and therefore the exact definition of Fourier transform cannot be used. Moreover, that definition requires knowledge of the signal for infinite time. For both reasons, a new set of techniques called *short-time analysis*, are proposed. These techniques decompose the speech signal into a series of short segments, referred to as *analysis frames*, and analyze each one independently.

In Figure 6.2 (a), note the assumption that the signal can be approximated as periodic within  $X$  and  $Y$  is reasonable. In regions  $(Z, W)$  and  $(H, G)$ , the signal is not periodic and looks like *random noise*. The signal in  $(Z, W)$  appears to have different noisy characteristics than those of segment  $(H, G)$ . The use of an analysis frame implies that the region is short enough for the behavior (periodicity or noise-like appearance) of the signal to be approximately constant. If the region where speech seems periodic is too long, the pitch period is not constant and not all the periods in the region are similar. In essence, the speech region has to be short enough so that the signal is *stationary* in that region: *i.e.*, the signal characteristics (whether periodicity or noise-like appearance) are uniform in that region. A more formal definition of stationarity is given in Chapter 5.

Similarly to the filterbanks described in Chapter 5, given a speech signal  $x[n]$ , we define the short-time signal  $x_m[n]$  of frame  $m$  as

$$x_m[n] = x[n]w_m[n] \quad (6.1)$$

the product of  $x[n]$  by a *window* function  $w_m[n]$ , which is zero everywhere except in a small region.

While the window function can have different *values* for different frames  $m$ , a popular choice is to keep it constant for all frames:

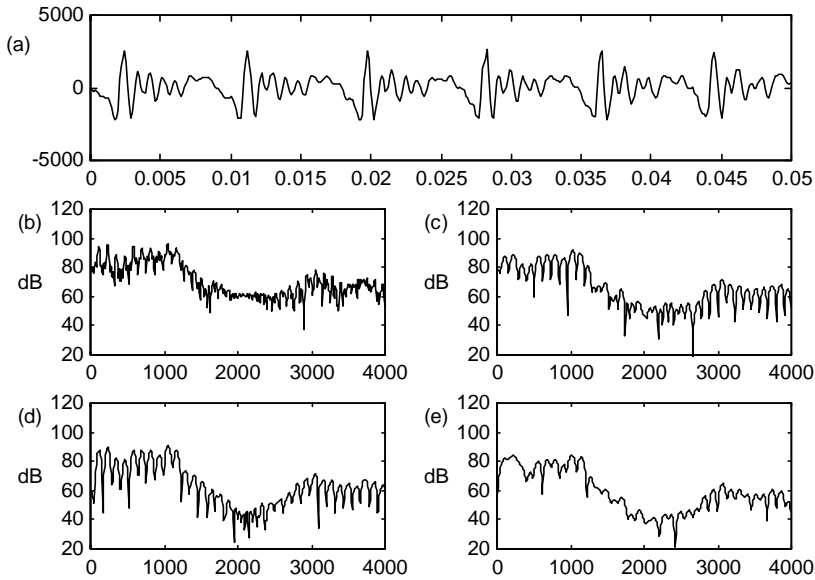
$$w_m[n] = w[m-n] \quad (6.2)$$

where  $w[n] = 0$  for  $|n| > N/2$ . In practice, the window length is on the order of 20 to 30 ms.

With the above framework, the short-time Fourier representation for frame  $m$  is defined as

$$X_m(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x_m[n]e^{-j\omega n} = \sum_{n=-\infty}^{\infty} w[m-n]x[n]e^{-j\omega n} \quad (6.3)$$

with all the properties of Fourier transforms studied in Chapter 5.



**Figure 6.3** Short-time spectrum of male voiced speech (vowel /ah/ with local pitch of 110Hz): (a) time signal, spectra obtained with (b) 30ms rectangular window and (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15ms Hamming window. The window lobes are not visible in (e), since the window is shorter than 2 times the pitch period. Note the spectral leakage present in (b).

In Figure 6.3 we show the short-time spectrum of voiced speech. Note that there are a number of peaks in the spectrum. To interpret this, assume the properties of  $x_m[n]$  persist outside the window, and that, therefore, the signal is periodic with period  $M$  in the true sense. In this case, we know (see Chapter 5) that its spectrum is a sum of impulses

$$X_m(e^{j\omega}) = \sum_{k=-\infty}^{\infty} X_m[k] \delta(\omega - 2\pi k / M) \quad (6.4)$$

Given that the Fourier transform of  $w[n]$  is

$$W(e^{j\omega}) = \sum_{n=-\infty}^{\infty} w[n] e^{-j\omega n} \quad (6.5)$$

so that the transform of  $w[m-n]$  is  $W(e^{-j\omega})e^{-j\omega m}$ . Therefore, using the convolution property, the transform of  $x[n]w[m-n]$  for fixed  $m$  is the convolution in the frequency domain

$$X_m(e^{j\omega}) = \sum_{k=-\infty}^{\infty} X_m[k] W(e^{j(\omega-2\pi k/N)}) e^{j(\omega-2\pi k/N)m} \quad (6.6)$$

which is a sum of weighted  $W(e^{j\omega})$ , shifted on every harmonic, the narrow peaks seen in Figure 6.3 (b) with a rectangular window. The short-time spectrum of a periodic signal exhibits peaks (equally spaced  $2\pi/M$  apart) representing the harmonics of the signal. We estimate  $X_m[k]$  from the short-time spectrum  $X_m(e^{j\omega})$ , and we see the importance of the length and choice of window.

Equation (6.6) indicates that one cannot recover  $X_m[k]$  by simply retrieving  $X_m(e^{j\omega})$ , although the approximation can be reasonable if there is a small value of  $\lambda$  such that

$$W(e^{j\omega}) \approx 0 \text{ for } |\omega - \omega_k| > \lambda \quad (6.7)$$

which is the case outside the main lobe of the window's frequency response.

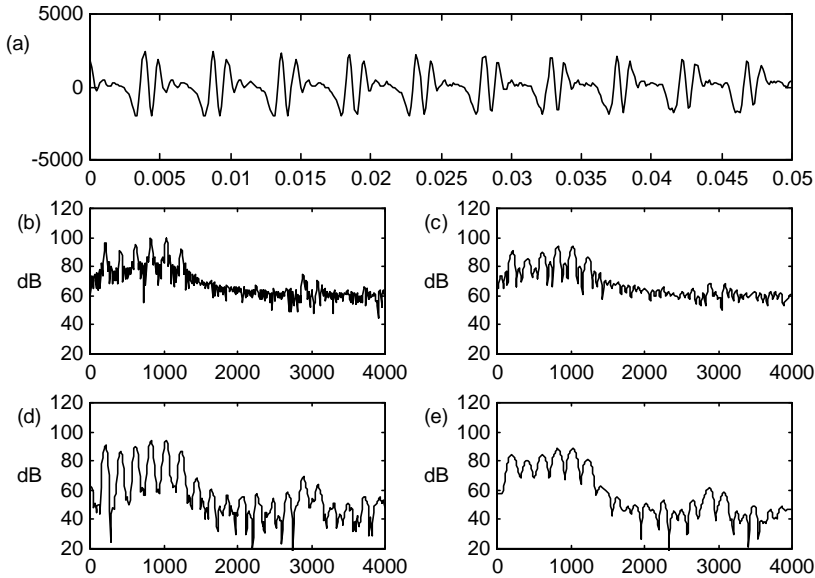
Recall from Section 5.4.2.1 that, for a rectangular window of length  $N$ ,  $\lambda = 2\pi/N$ . Therefore, Eq. (6.7) is satisfied if  $N \geq M$ , *i.e.*, the rectangular window contains at least one pitch period. The width of the main lobe of the window's frequency response is inversely proportional to the length of the window. The pitch period in Figure 6.3 is  $M = 71$  at a sampling rate of 8 kHz. A shorter window is used in Figure 6.3 (c), which results in wider analysis lobes, though still visible.

Also recall from Section 5.4.2.2 that for a Hamming window of length  $N$ ,  $\lambda = 4\pi/N$ : twice as wide as that of the rectangular window, which entails  $N \geq 2M$ . Thus, for Eq. (6.7) to be met, a Hamming window must contain at least two pitch periods. The lobes are visible in Figure 6.3 (d) since  $N = 240$ , but they are not visible in Figure 6.3 (e) since  $N = 120$ , and  $N < 2M$ .

In practice, one cannot know what the pitch period is ahead of time, which often means you need to prepare for the lowest pitch period. A low-pitched voice with a  $F_0 = 50\text{Hz}$  requires a rectangular window of at least 20 ms and a Hamming window of at least 40 ms for the condition in Eq. (6.7) to be met. If speech is non-stationary within 40ms, taking such a long window implies obtaining an average spectrum during that segment instead of several distinct spectra. For this reason, the rectangular window provides better *time resolution* than the Hamming window. Figure 6.4 shows analysis of female speech for which shorter windows are feasible.

But the frequency response of the window is not completely zero outside its main lobe, so one needs to see the effects of this incorrect assumption. From Section 5.4.2.1 note that the second lobe of a rectangular window is only approximately 17 dB below the main lobe. Therefore, for the  $k^{\text{th}}$  harmonic the value of  $X_m(e^{j2\pi k/M})$  contains not  $X_m[k]$ , but also a weighted sum of  $X_m[l]$ . This phenomenon is called *spectral leakage* because the amplitude of one harmonic leaks over the rest and masks its value. If the signal's spectrum is white, spectral leakage does not cause a major problem, since the effect of the second lobe

on a harmonic is only  $10\log_{10}(1+10^{-17/10}) = 0.08\text{dB}$ . On the other hand, if the signal's spectrum decays more quickly in frequency than the decay of the window, the spectral leakage results in inaccurate estimates.



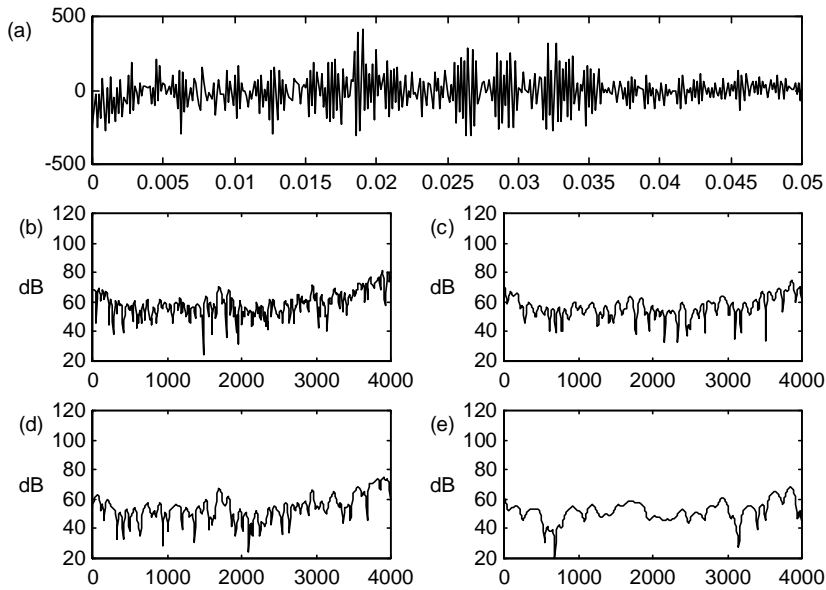
**Figure 6.4** Short-time spectrum of female voiced speech (vowel /aa/ with local pitch of 200Hz): (a) time signal, spectra obtained with (b) 30 ms rectangular window and (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15 ms Hamming window. In all cases the window lobes are visible, since the window is longer than 2 times the pitch period. Note the spectral leakage present in (b) and (c).

From Section 5.4.2.2, observe that the second lobe of a Hamming window is approximately 43 dB, which means that the spectral leakage effect is much less pronounced. Other windows, such as Hanning, or triangular windows, also offer less spectral leakage than the rectangular window. This important fact is the reason why, despite their better time resolution, rectangular windows are rarely used for speech analysis. In practice, window lengths are on the order of 20 to 30 ms. This choice is a compromise between the stationarity assumption and the frequency resolution.

In practice, the Fourier transform in Eq. (6.3) is obtained through an FFT. If the window has length  $N$ , the FFT has to have a length greater than or equal to  $N$ . Since FFT algorithms often have lengths that are powers of 2 ( $L = 2^R$ ), the windowed signal with length  $N$  is augmented with  $(L - N)$  zeros either before, after, or both. This process is called *zero-padding*. A larger value of  $L$  provides a finer description of the discrete Fourier transform; but it does not increase the analysis frequency resolution: this is the sole mission of the window length  $N$ .

In Figure 6.3, observe the broad peaks, resonances or formants, which represent the filter characteristics. For voiced sounds there is typically more energy at low frequencies than at high frequencies, also called *roll-off*. It is impossible to determine exactly the filter characteristics, because we know only samples at the harmonics, and we have no knowledge of the values in between. In fact, the resonances are less obvious in Figure 6.4 because the harmonics sample the spectral envelope less densely. For high-pitched female speakers and children, it is even more difficult to locate the formant resonances from the short-time spectrum.

Figure 6.5 shows the short-time analysis of unvoiced speech, for which no regularity is observed.



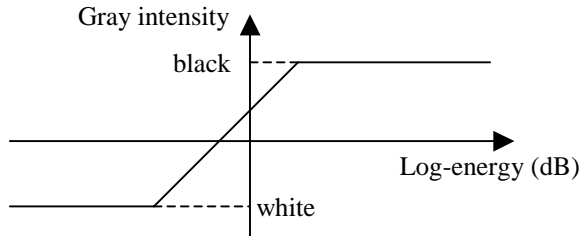
**Figure 6.5** Short-time spectrum of unvoiced speech. (a) time signal, (b) 30 ms rectangular window (c) 15 ms rectangular window, (d) 30 ms Hamming window (e) 15 ms Hamming window.

### 6.1.1. Spectrograms

Since the spectrogram displays just the energy and not the phase of the short-term Fourier transform, we compute the energy as

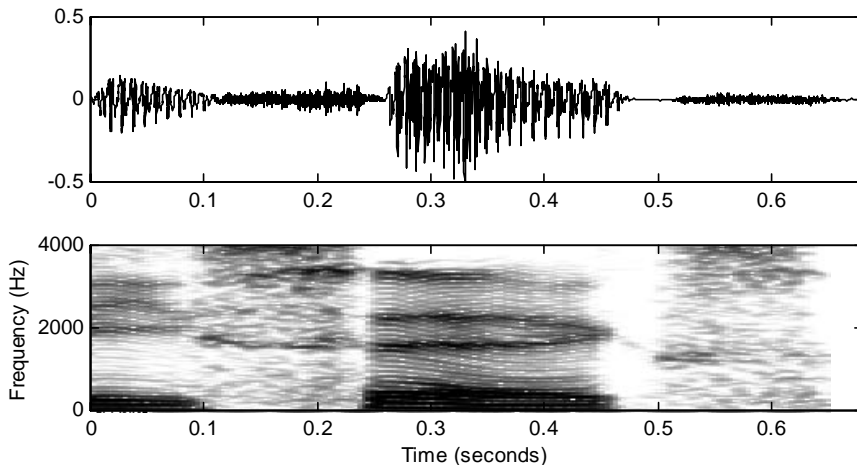
$$\log |X[k]|^2 = \log (X_r^2[k] + X_i^2[k]) \quad (6.8)$$

with this value converted to a gray scale according to Figure 6.6. Pixels whose values have not been computed are interpolated. The slope controls the contrast of the spectrogram, while the saturation points for white and black control the dynamic range.



**Figure 6.6** Conversion between log-energy values (in the  $x$ -axis) and gray scale (in the  $y$ -axis). Larger log-energies correspond to a darker gray color. There is a linear region for which more log-energy corresponds to darker gray, but there is saturation at both ends. Typically there is 40 to 60 dB between the pure white and the pure black.

There are two main types of spectrograms: *narrow-band* and *wide-band*. Wide-band spectrograms use relatively short windows ( $< 10$  ms) and thus have good time resolution at the expense of lower frequency resolution, since the corresponding filters have wide bandwidths ( $> 200$  Hz) and the harmonics cannot be seen. Note the vertical stripes in Figure 6.2, due to the fact that some windows are centered at the high part of a pitch pulse, and others in between have lower energy. Spectrograms can aid in determining formant frequencies and fundamental frequency, as well as voiced and unvoiced regions.



**Figure 6.7** Waveform (a) with its corresponding narrowband spectrogram (b). Darker areas mean higher energy for that time and frequency. The harmonics can be seen as horizontal lines spaced by fundamental frequency. The corresponding wideband spectrogram can be seen in Figure 6.2.



Narrow-band spectrograms use relatively long windows ( $> 20$  ms), which lead to filters with narrow bandwidth ( $< 100$  Hz). On the other hand, time resolution is lower than for wide-band spectrograms (see Figure 6.7). Note that the harmonics can be clearly seen, because some of the filters capture the energy of the signal's harmonics, and filters in between have little energy.

Some implementation details also need to be taken into account. Since speech signals are real, the Fourier transform is Hermitian, and its power spectrum is also even. Thus, it is only necessary to display values for  $0 \leq k \leq N/2$  for  $N$  even. In addition, while the traditional spectrogram uses a gray scale, a color scale can also be used, or even a 3-D representation. In addition, to make the spectrograms easier to read, sometimes the signal is first pre-emphasized (typically with a first-order difference FIR filter) to boost the high frequencies to counter the roll-off of natural speech.

By inspecting both narrow-band and wide-band spectrograms, we can learn the filter's magnitude response and whether the source is voiced or not. Nonetheless it is very difficult to separate source and filter due to nonstationarity of the speech signal, spectral leakage, and the fact that only the filter's magnitude response can be known at the signal's harmonics.

### 6.1.2. Pitch-Synchronous Analysis

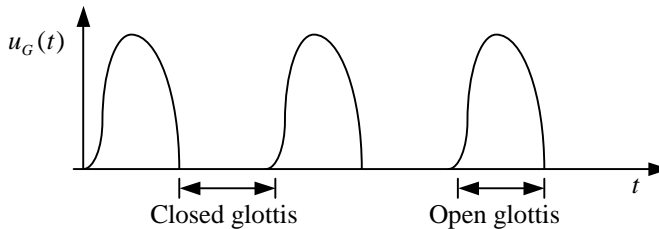
In the previous discussion, we assumed that the window length is fixed, and we saw the tradeoffs between a window that contained several pitch periods (narrow-band spectrograms) and a window that contained less than a pitch period (wide-band spectrograms). One possibility is to use a rectangular window whose length is exactly one pitch period; this is called *pitch-synchronous* analysis. To reduce spectral leakage a tapering window, such as Hamming or Hanning, can be used, with the window covering exactly two pitch periods. This latter option provides a very good compromise between time and frequency resolution. In this representation, no stripes can be seen in either time or frequency. The difficulty in computing pitch synchronous analysis is that, of course, we need to know the local pitch period, which, as we see in Section 6.7, is not an easy task.

## 6.2. ACOUSTICAL MODEL OF SPEECH PRODUCTION

Speech is a sound wave created by vibration that is propagated in the air. Acoustic theory analyzes the laws of physics that govern the propagation of sound in the vocal tract. Such a theory should consider three-dimensional wave propagation, the variation of the vocal tract shape with time, losses due to heat conduction and viscous friction at the vocal tract walls, softness of the tract walls, radiation of sound at the lips, nasal coupling and excitation of sound. While a detailed model that considers all of the above is not yet available, some models provide a good approximation in practice, as well as a good understanding of the physics involved.

### 6.2.1. Glottal Excitation

As discussed in Chapter 2, the vocal cords constrict the path from the lungs to the vocal tract. This is illustrated in Figure 6.8. As lung pressure is increased, air flows out of the lungs and through the opening between the vocal cords (*glottis*). At one point the vocal cords are together, thereby blocking the airflow, which builds up *pressure* behind them. Eventually the pressure reaches a level sufficient to force the vocal cords to open and thus allow air to flow through the glottis. Then, the pressure in the glottis falls and, if the tension in the vocal cords is properly adjusted, the reduced pressure allows the cords to come together, and the cycle is repeated. This condition of sustained oscillation occurs for voiced sounds. The *closed-phase* of the oscillation takes place when the glottis is closed and the *volume velocity* is zero. The *open-phase* is characterized by a non-zero volume velocity, in which the lungs and the vocal tract are coupled.



**Figure 6.8** Glottal excitation: volume velocity is zero during the closed-phase, during which the vocal cords are closed.

Rosenberg's glottal model [39] defines the shape of the glottal volume velocity with the *open quotient*, or duty cycle, as the ratio of pulse duration to pitch period, and the *speed quotient* as the ratio of the rising to falling pulse durations.

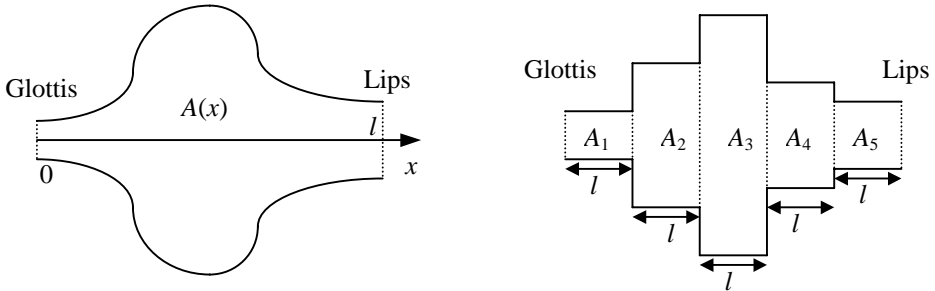
### 6.2.2. Lossless Tube Concatenation

A widely used model for speech production is based on the assumption that the vocal tract can be represented as a concatenation of lossless tubes, as shown in Figure 6.9. The constant cross-sectional areas  $\{A_k\}$  of the tubes approximate the area function  $A(x)$  of the vocal tract. If a large number of tubes of short length are used, we reasonably expect the frequency response of the concatenated tubes to be close to those of a tube with continuously varying area function.

For frequencies corresponding to wavelengths that are long compared to the dimensions of the vocal tract, it is reasonable to assume plane wave propagation along the axis of the tubes. If in addition we assume that there are no losses due to viscosity or thermal conduction, and that the area  $A$  does not change over time, the sound waves in the tube satisfy the following pair of differential equations:

$$\begin{aligned}
 -\frac{\partial p(x,t)}{\partial x} &= \frac{\rho}{A} \frac{\partial u(x,t)}{\partial t} \\
 \frac{\partial u(x,t)}{\partial x} &= \frac{A}{\rho c^2} \frac{\partial p(x,t)}{\partial t}
 \end{aligned}
 \tag{6.9}$$

where  $p(x,t)$  is the sound pressure in the tube at position  $x$  and time  $t$ ,  $u(x,t)$  is the volume velocity flow in the tube at position  $x$  and time  $t$ ,  $\rho$  is the density of air in the tube,  $c$  is the velocity of sound and  $A$  is the cross-sectional area of the tube.

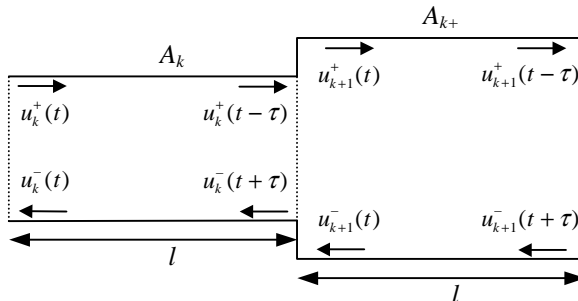


**Figure 6.9** Approximation of a tube with continuously varying area  $A(x)$  as a concatenation of 5 lossless acoustic tubes.

Since Eqs. (6.9) are linear, the pressure and volume velocity in tube  $k^{th}$  are related by

$$\begin{aligned}
 u_k(x,t) &= u_k^+(t-x/c) - u_k^-(t+x/c) \\
 p_k(x,t) &= \frac{\rho c}{A_k} [u_k^+(t-x/c) + u_k^-(t+x/c)]
 \end{aligned}
 \tag{6.10}$$

where  $u_k^+(t-x/c)$  and  $u_k^-(t-x/c)$  are the traveling waves in the positive and negative directions respectively and  $x$  is the distance measured from the left-hand end of tube  $k^{th}$ :  $0 \leq x \leq l$ . The reader can prove that this is indeed the solution by substituting Eq. (6.10) into (6.9).



**Figure 6.10** Junction between two lossless tubes.

When there is a junction between two tubes, as in Figure 6.10, part of the wave is reflected at the junction, as measured by  $r_k$ , the reflection coefficient

$$r_k = \frac{A_{k+1} - A_k}{A_{k+1} + A_k} \quad (6.11)$$

so that the larger the difference between the areas the more energy is reflected. The proof [9] is beyond the scope of this book. Since  $A_k$  and  $A_{k+1}$  are positive, it is easy to show that  $r_k$  satisfies the condition

$$-1 \leq r_k \leq 1 \quad (6.12)$$

A relationship between the  $z$ -transforms of the volume velocity at the glottis  $u_G[n]$  and the lips  $u_L[n]$  for a concatenation of  $N$  lossless tubes can be derived [9] using a discrete-time version of Eq. (6.10) and taking into account boundary conditions for every junction:

$$V(z) = \frac{U_L(z)}{U_G(z)} = \frac{0.5z^{-N/2} (1+r_G) \prod_{k=1}^N (1+r_k)}{\begin{bmatrix} 1 & -r_G \end{bmatrix} \left( \prod_{k=1}^N \begin{bmatrix} 1 & -r_k \\ -r_k z^{-1} & z^{-1} \end{bmatrix} \right) \begin{bmatrix} 1 \\ 0 \end{bmatrix}} \quad (6.13)$$

where  $r_G$  is the reflection coefficient at the glottis and  $r_N = r_L$  is the reflection coefficient at the lips. Equation (6.11) is still valid for the glottis and lips, where  $A_0 = \rho c / Z_G$  is the equivalent area at the glottis and  $A_{N+1} = \rho c / Z_L$  the equivalent area at the lips.  $Z_G$  and  $Z_L$  are the equivalent impedances at the glottis and lips, respectively. Such impedances relate the volume velocity and pressure, for the lips the expression is

$$U_L(z) = P_L(z) / Z_L \quad (6.14)$$

In general, the concatenation of  $N$  lossless tubes results in an  $N$ -pole system as shown in Eq. (6.13). For a concatenation of  $N$  tubes, there are at most  $N/2$  complex conjugate poles, or resonances or formants. These resonances occur when a given frequency gets *trapped* in the vocal tract because it is reflected back at the lips and then again back at the glottis.

Since each tube has length  $l$  and there are  $N$  of them, the total length is  $L = lN$ . The propagation delay in each tube  $\tau = l/c$ , and the sampling period is  $T = 2\tau$ , the round trip in a tube. We can find a relationship between the number of tubes  $N$  and the sampling frequency  $F_s = 1/T$ :

$$N = \frac{2LF_s}{c} \quad (6.15)$$

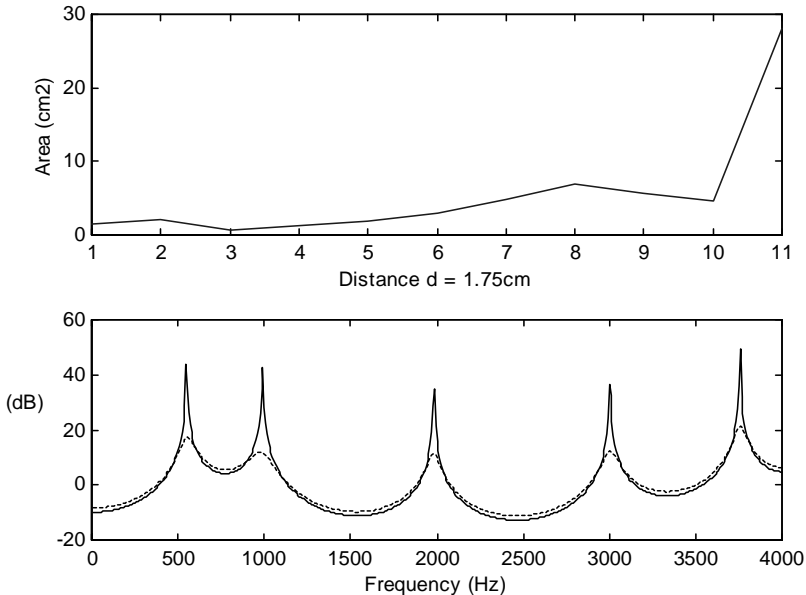
For example, for  $F_s = 8000$  kHz,  $c = 34000$  cm/s, and  $L = 17$  cm, the average length of a male adult vocal tract, we obtain  $N = 8$ , or alternatively 4 formants. Experimentally, the vocal tract transfer function has been observed to have approximately 1 formant per kilohertz. Shorter vocal tract lengths (females or children) have fewer resonances per kilohertz and vice versa.

The pressure at the lips has been found to approximate the derivative of volume velocity, particularly at low frequencies. Thus,  $Z_L(z)$  can be approximated by

$$Z_L(z) \approx R_0(1 - z^{-1}) \quad (6.16)$$

which is 0 for low frequencies and reaches  $R_0$  asymptotically. This dependency upon frequency results in a reflection coefficient that is also a function of frequency. For low frequencies,  $r_L = 1$ , and no loss occurs. At higher frequencies, loss by radiation translates into widening of formant bandwidths.

Similarly, the glottal impedance is also a function of frequency in practice. At high frequencies,  $Z_G$  is large and  $r_G \approx 1$  so that all the energy is transmitted. For low frequencies,  $r_G < 1$ , whose main effect is an increase of bandwidth for the lower formants.



**Figure 6.11** Area function and frequency response for vowel /a/ and its approximation as a concatenation of 10 lossless tubes. A reflection coefficient at the load of  $k = 0.72$  (dotted line) is displayed. For comparison, the case of  $k = 1.0$  (solid line) is also shown.

Moreover, energy is lost as a result of vibration of the tube walls, which is more pronounced at low frequencies. Energy is also lost, to a lesser extent, as a result of viscous friction between the air and the walls of the tube, particularly at frequencies above 3kHz. The yielding walls tend to raise the resonance frequencies while the viscous and thermal losses tend to lower them. The net effect in the transfer function is a broadening of the resonances' bandwidths.

Despite thermal losses, yielding walls in the vocal tract, and the fact that both  $r_L$  and  $r_G$  are functions of frequency, the all-pole model of Eq. (6.13) for  $V(z)$  has been found to be a good approximation in practice [13]. In Figure 6.11 we show the measured area function of a vowel and its corresponding frequency response obtained using the approximation as a concatenation of 10 lossless tubes with a constant  $r_L$ . The measured formants and corresponding bandwidths match quite well with this model despite all the approximations made. Thus, this concatenation of lossless tubes model represents reasonably well the acoustics inside the vocal tract. Inspired by the above results, we describe in Section 6.3 "Linear Predictive Coding," an all-pole model for speech.

In the production of the nasal consonants, the velum is lowered to trap the nasal tract to the pharynx, whereas a complete closure is formed in the oral tract (/m/ at the lips, /n/ just back of the teeth and /ng/ just forward of the velum itself. This configuration is shown in Figure 6.12, which shows two branches, one of them completely closed. For nasals, the radiation occurs primarily at the nostrils. The set of resonances is determined by the shape and length of the three tubes. At certain frequencies, the wave reflected in the closure cancels the wave at the pharynx, preventing energy from appearing at nostrils. The result is that for nasal sounds, the vocal tract transfer function  $V(z)$  has anti-resonances (zeros) in addition to resonances. It has also been observed that nasal resonances have broader bandwidths than non-nasal voiced sounds, due to the greater viscous friction and thermal loss because of the large surface area of the nasal cavity.

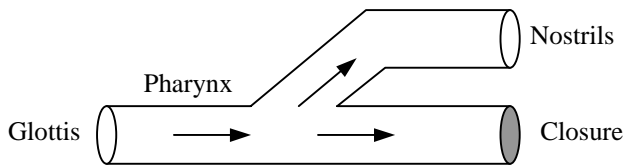


Figure 6.12 Coupling of the nasal cavity with the oral cavity.

### 6.2.3. Source-Filter Models of Speech Production

As shown in Chapter 10, speech signals are captured by microphones that respond to changes in air pressure. Thus, it is of interest to compute the pressure at the lips  $P_L(z)$ , which can be obtained as

$$P_L(z) = U_L(z)Z_L(z) = U_G(z)V(z)Z_L(z) \quad (6.17)$$

For voiced sounds we can model  $u_G[n]$  as an impulse train convolved with  $g[n]$ , the glottal pulse (see Figure 6.13). Since  $g[n]$  is of finite length, its  $z$ -transform is an all-zero system.

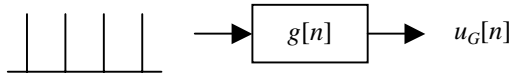


Figure 6.13 Model of the glottal excitation for voiced sounds.

The complete model for both voiced and unvoiced sounds is shown in Figure 6.14. We have modeled  $u_G[n]$  in unvoiced sounds as random noise.

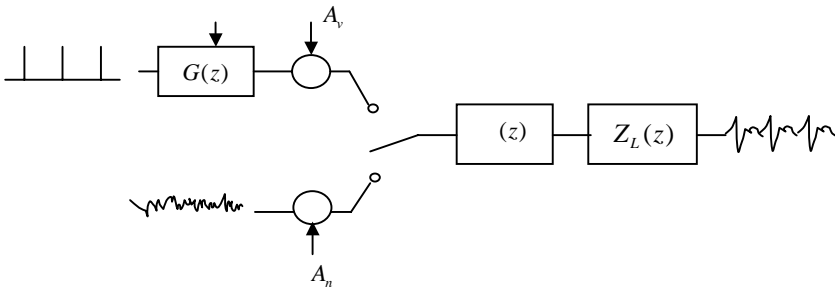


Figure 6.14 General discrete-time model of speech production. The excitation can be either an impulse train with period  $T$  and amplitude  $A_v$  driving a filter  $G(z)$  or random noise with amplitude  $A_n$ .

We can simplify the model in Figure 6.14 by grouping  $G(z)$ ,  $V(z)$ , and  $Z_L(z)$  into  $H(z)$  for voiced sounds, and  $V(z)$  and  $Z_L(z)$  into  $H(z)$  for unvoiced sounds. The simplified model is shown in Figure 6.15, where we make explicit the fact that the filter changes over time.

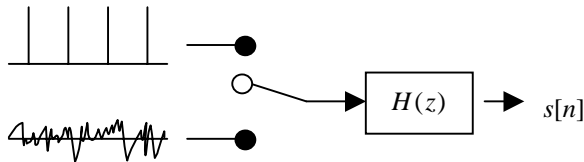


Figure 6.15 Source-filter model for voiced and unvoiced speech.

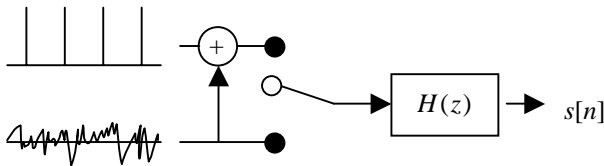
This model is a decent approximation, but fails on voiced fricatives, since those sounds contain both a periodic component and an aspirated component. In this case, a *mixed excitation* model can be applied, using for voiced sounds a sum of both an impulse train and colored noise (Figure 6.16).

The model in Figure 6.15 is appealing because the source is white (has a flat spectrum) and all the *coloring* is in the filter. Other source-filter decompositions attempt to model the source as the signal at the glottis, in which the source is definitely not white. Since  $G(z)$ ,  $Z_L(z)$  contain zeros, and  $V(z)$  can also contain zeros for nasals,  $H(z)$  is no

longer all-pole. However, recall from in Chapter 5, we state that the  $z$ -transform of  $x[n] = a^n u[n]$  is

$$X(z) = \sum_{n=0}^{\infty} a^n z^{-n} = \frac{1}{1 - az^{-1}} \quad \text{for } |a| < |z| \quad (6.18)$$

so that by inverting Eq. (6.18) we see that a zero can be expressed with infinite poles. This is the reason why all-pole models are still reasonable approximations as long as a large enough number of poles is used. Fant [12] showed that on the average the speech spectrum contains one pole per kHz. Setting the number of poles  $p$  to  $F_s + 2$ , where  $F_s$  is the sampling frequency expressed in kHz, has been found to work well in practice.



**Figure 6.16** A mixed excitation source-filter model of speech.

### 6.3. LINEAR PREDICTIVE CODING

A very powerful method for speech analysis is based on *linear predictive coding* (LPC) [4, 7, 19, 24, 27], also known as LPC analysis or *auto-regressive* (AR) modeling. This method is widely used because it is fast and simple, yet an effective way of estimating the main parameters of speech signals.

As shown in Section 6.2, an all-pole filter with a sufficient number of poles is a good approximation for speech signals. Thus, we could model the filter  $H(z)$  in Figure 6.15 as

$$H(z) = \frac{X(z)}{E(z)} = \frac{1}{1 - \sum_{k=1}^p a_k z^{-k}} = \frac{1}{A(z)} \quad (6.19)$$

where  $p$  is the order of the LPC analysis. The *inverse filter*  $A(z)$  is defined as

$$A(z) = 1 - \sum_{k=1}^p a_k z^{-k} \quad (6.20)$$

Taking inverse  $z$ -transforms in Eq. (6.19) results in

$$x[n] = \sum_{k=1}^p a_k x[n-k] + e[n] \quad (6.21)$$



Linear predictive coding gets its name from the fact that it predicts the current sample as a linear combination of its past  $p$  samples:

$$\tilde{x}[n] = \sum_{k=1}^p a_k x[n-k] \quad (6.22)$$

The prediction error when using this approximation is

$$e[n] = x[n] - \tilde{x}[n] = x[n] - \sum_{k=1}^p a_k x[n-k] \quad (6.23)$$

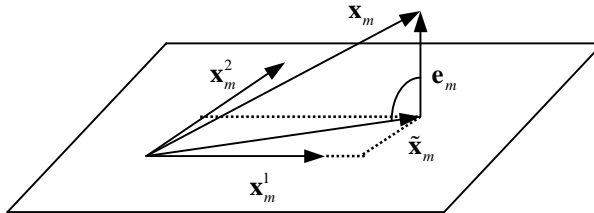
### 6.3.1. The Orthogonality Principle

To estimate the predictor coefficients from a set of speech samples, we use the short-term analysis technique. Let's define  $x_m[n]$  as a segment of speech selected in the vicinity of sample  $m$ :

$$x_m[n] = x[m+n] \quad (6.24)$$

We define the short-term prediction error for that segment as

$$E_m = \sum_n e_m^2[n] = \sum_n (x_m[n] - \tilde{x}_m[n])^2 = \sum_n \left( x_m[n] - \sum_{j=1}^p a_j x_m[n-j] \right)^2 \quad (6.25)$$



**Figure 6.17** The orthogonality principle. The prediction error is orthogonal to the past samples.

In the absence of knowledge about the probability distribution of  $a_i$ , a reasonable estimation criterion is minimum mean squared error, introduced in Chapter 4. Thus, given a signal  $x_m[n]$ , we estimate its corresponding LPC coefficients as those that minimize the total prediction error  $E_m$ . Taking the derivative of Eq. (6.25) with respect to  $a_i$  and equating to 0, we obtain:

$$\langle \mathbf{e}_m, \mathbf{x}_m^i \rangle = \sum_n e_m[n] x_m[n-i] = 0 \quad 1 \leq i \leq p \quad (6.26)$$

where we have defined  $\mathbf{e}_m$  and  $\mathbf{x}_m^i$  as vectors of samples, and their inner product has to be 0. This condition, known as *orthogonality principle*, says that the predictor coefficients that minimize the prediction error are such that the error must be orthogonal to the past vectors, and is seen in Figure 6.17.

Equation (6.26) can be expressed as a set of  $p$  linear equations

$$\sum_n x_m[n-i] x_m[n] = \sum_{j=1}^p a_j \sum_n x_m[n-i] x_m[n-j] \quad i = 1, 2, \dots, p \quad (6.27)$$

For convenience, we can define the correlation coefficients as

$$\phi_m[i, j] = \sum_n x_m[n-i] x_m[n-j] \quad (6.28)$$

so that Eqs. (6.27) and (6.28) can be combined to obtain the so-called *Yule-Walker equations*:

$$\sum_{j=1}^p a_j \phi_m[i, j] = \phi_m[i, 0] \quad i = 1, 2, \dots, p \quad (6.29)$$

Solution of the set of  $p$  linear equations results in the  $p$  LPC coefficients that minimize the prediction error. With  $a_i$  satisfying Eq. (6.29), the total prediction error in Eq. (6.25) takes on the following value:

$$E_m = \sum_n x_m^2[n] - \sum_{j=1}^p a_j \sum_n x_m[n] x_m[n-j] = \phi[0, 0] - \sum_{j=1}^p a_j \phi[0, j] \quad (6.30)$$

It is convenient to define a normalized prediction error  $u[n]$  with unity energy

$$\sum_n u_m^2[n] = 1 \quad (6.31)$$

and a gain  $G$ , such that

$$e_m[n] = G u_m[n] \quad (6.32)$$

The gain  $G$  can be computed from the short-term prediction error

$$E_m = \sum_n e_m^2[n] = G^2 \sum_n u_m^2[n] = G^2 \quad (6.33)$$

## 6.3.2. Solution of the LPC Equations

The solution of the Yule-Walker equations in Eq. (6.29) can be achieved with any standard matrix inversion package. Because of the special form of the matrix here, some efficient solutions are possible, as described below. Also, each solution offers a different insight so we present three different algorithms: the covariance method, the autocorrelation method, and the lattice method.

### 6.3.2.1. Covariance Method

The covariance method [4] is derived by defining directly the interval over which the summation in Eq. (6.28) takes place:

$$E_m = \sum_{n=0}^{N-1} e_m^2[n] \quad (6.34)$$

so that  $\phi_m[i, j]$  in Eq. (6.28) becomes

$$\phi_m[i, j] = \sum_{n=0}^{N-1} x_m[n-i]x_m[n-j] = \sum_{n=-i}^{N-1-j} x_m[n]x_m[n+i-j] = \phi_m[j, i] \quad (6.35)$$

and Eq. (6.29) becomes

$$\begin{pmatrix} \phi_m[1,1] & \phi_m[1,2] & \phi_m[1,3] & \cdots & \phi_m[1,p] \\ \phi_m[2,1] & \phi_m[2,2] & \phi_m[2,3] & \cdots & \phi_m[2,p] \\ \phi_m[3,1] & \phi_m[3,2] & \phi_m[3,3] & \cdots & \phi_m[3,p] \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \phi_m[p,1] & \phi_m[p,2] & \phi_m[p,3] & \cdots & \phi_m[p,p] \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \cdots \\ a_p \end{pmatrix} = \begin{pmatrix} \phi_m[1,0] \\ \phi_m[2,0] \\ \phi_m[3,0] \\ \cdots \\ \phi_m[p,0] \end{pmatrix} \quad (6.36)$$

which can be expressed as the following matrix equation

$$\Phi \mathbf{a} = \boldsymbol{\psi} \quad (6.37)$$

where the matrix  $\Phi$  in Eq. (6.37) is symmetric and *positive definite*, for which efficient methods are available, such as the Cholesky decomposition. For this method, also called the squared root method, the matrix  $\Phi$  is expressed as

$$\Phi = \mathbf{V} \mathbf{D} \mathbf{V}^t \quad (6.38)$$

where  $\mathbf{V}$  is a lower triangular matrix (whose main diagonal elements are 1's), and  $\mathbf{D}$  is a diagonal matrix. So each element of  $\Phi$  can be expressed as

$$\phi[i, j] = \sum_{k=1}^j V_{ik} d_k V_{jk} \quad 1 \leq j < i \quad (6.39)$$

or alternatively

$$V_{ij}d_j = \phi[i, j] - \sum_{k=1}^{j-1} V_{ik}d_k V_{jk} \quad 1 \leq j < i \quad (6.40)$$

and for the diagonal elements

$$\phi[i, i] = \sum_{k=1}^i V_{ik}d_k V_{ik} \quad (6.41)$$

or alternatively

$$d_i = \phi[i, i] - \sum_{k=1}^{i-1} V_{ik}^2 d_k, \quad i \geq 2 \quad (6.42)$$

with

$$d_1 = \phi[1, 1] \quad (6.43)$$

The Cholesky decomposition starts with Eq. (6.43) then alternates between Eqs. (6.40) and (6.42). Once the matrices  $\mathbf{V}$  and  $\mathbf{D}$  have been determined, the LPC coefficients are solved in a two-step process. The combination of Eqs. (6.37) and (6.38) can be expressed as

$$\mathbf{V}\mathbf{Y} = \boldsymbol{\psi} \quad (6.44)$$

with

$$\mathbf{Y} = \mathbf{D}\mathbf{V}'\mathbf{a} \quad (6.45)$$

or alternatively

$$\mathbf{V}'\mathbf{a} = \mathbf{D}^{-1}\mathbf{Y} \quad (6.46)$$

Therefore, given matrix  $\mathbf{V}$  and Eq. (6.44),  $\mathbf{Y}$  can be solved recursively as

$$Y_i = \psi_i - \sum_{j=1}^{i-1} V_{ij}Y_j, \quad 2 \leq i \leq p \quad (6.47)$$

with the initial condition

$$Y_1 = \psi_1 \quad (6.48)$$

Having determined  $\mathbf{Y}$ , Eq. (6.46) can be solved recursively in a similar way

$$a_i = Y_i / d_i - \sum_{j=i+1}^p V_{ji}a_j, \quad 1 \leq i < p \quad (6.49)$$

with the initial condition

$$a_p = Y_p / d_p \quad (6.50)$$

where the index  $i$  in Eq. (6.49) proceeds backwards.

The term covariance analysis is somewhat of a misnomer, since we know from Chapter 5 that the covariance of a signal is the correlation of that signal with its mean removed. It was so called because the matrix in Eq. (6.36) has the properties of a covariance matrix, though this algorithm is more like a cross-correlation.

### 6.3.2.2. Autocorrelation Method

The summation in Eq. (6.28) had no specific range. In the autocorrelation method [24, 27], we assume that  $x_m[n]$  is 0 outside the interval  $0 \leq n < N$  :

$$x_m[n] = x[m+n]w[n] \quad (6.51)$$

with  $w[n]$  being a window (such as a Hamming window) which is 0 outside the interval  $0 \leq n < N$ . With this assumption, the corresponding prediction error  $e_m[n]$  is non-zero over the interval  $0 \leq n < N + p$ , and, therefore, the total prediction error takes on the value

$$E_m = \sum_{n=0}^{N+p-1} e_m^2[n] \quad (6.52)$$

With this range, Eq. (6.28) can be expressed as

$$\phi_m[i, j] = \sum_{n=0}^{N+p-1} x_m[n-i]x_m[n-j] = \sum_{n=0}^{N-1-(i-j)} x_m[n]x_m[n+i-j] \quad (6.53)$$

or alternatively

$$\phi_m[i, j] = R_m[i-j] \quad (6.54)$$

with  $R_m[k]$  being the autocorrelation sequence of  $x_m[n]$ :

$$R_m[k] = \sum_{n=0}^{N-1-k} x_m[n]x_m[n+k] \quad (6.55)$$

Combining Eqs. (6.54) and (6.29), we obtain

$$\sum_{j=1}^p a_j R_m[|i-j|] = R_m[i] \quad (6.56)$$

which corresponds to the following matrix equation

$$\begin{pmatrix} R_m[0] & R_m[1] & R_m[2] & \cdots & R_m[p-1] \\ R_m[1] & R_m[0] & R_m[1] & \cdots & R_m[p-2] \\ R_m[2] & R_m[1] & R_m[0] & \cdots & R_m[p-3] \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ R_m[p-1] & R_m[p-2] & R_m[p-3] & \cdots & R_m[0] \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \cdots \\ a_p \end{pmatrix} = \begin{pmatrix} R_m[1] \\ R_m[2] \\ R_m[3] \\ \cdots \\ R_m[p] \end{pmatrix} \quad (6.57)$$

The matrix in Eq. (6.57) is symmetric and all the elements in its diagonals are identical. Such matrices are called *Toeplitz*. Durbin's recursion exploits this fact resulting in a very efficient algorithm (for convenience, we omit the subscript  $m$  of the autocorrelation function), whose proof is beyond the scope of this book:

### 1. Initialization

$$E^0 = R[0] \quad (6.58)$$

2. Iteration. For  $i = 1, \dots, p$  do the following recursion:

$$k_i = \left( R[i] - \sum_{j=1}^{i-1} a_j^{i-1} R[i-j] \right) / E^{i-1} \quad (6.59)$$

$$a_i^i = k_i \quad (6.60)$$

$$a_j^i = a_j^{i-1} - k_i a_{i-j}^{i-1}, \quad 1 \leq j < i \quad (6.61)$$

$$E^i = (1 - k_i^2) E^{i-1} \quad (6.62)$$

3. Final solution:

$$a_j = a_j^p \quad 1 \leq j \leq p \quad (6.63)$$

where the coefficients  $k_i$ , called *reflection coefficients*, are bounded between  $-1$  and  $1$  (see Section 6.3.2.3). In the process of computing the predictor coefficients of order  $p$ , the recursion finds the solution of the predictor coefficients for all orders less than  $p$ .

Replacing  $R[j]$  by the normalized autocorrelation coefficients  $r[j]$ , defined as

$$r[j] = R[j] / R[0] \quad (6.64)$$

results in identical LPC coefficients, and the recursion is more robust to problems with arithmetic precision. Likewise, the normalized prediction error at iteration  $i$  is defined by dividing Eq. (6.30) by  $R[0]$ , which, using Eq. (6.54), results in

$$V^i = \frac{E^i}{R[0]} = 1 - \sum_{j=1}^i a_j r[j] \quad (6.65)$$

The normalized prediction error is, using Eqs. (6.62) and (6.65),

$$V^p = \prod_{i=1}^p (1 - k_i^2) \quad (6.66)$$

### 6.3.2.3. Lattice Formulation

In this section we derive the lattice formulation [7, 19], an equivalent algorithm to the Levinson Durbin recursion, which has some precision benefits. It is advantageous to define the *forward prediction error* obtained at stage  $i$  of the Levinson Durbin procedure as

$$e^i[n] = x[n] - \sum_{k=1}^i a_k^i x[n-k] \quad (6.67)$$

whose  $z$ -transform is given by

$$E^i(z) = A^i(z)X(z) \quad (6.68)$$

with  $A^i(z)$  being defined by

$$A^i(z) = 1 - \sum_{k=1}^i a_k^i z^{-k} \quad (6.69)$$

which, combined with Eq. (6.61), results in the following recursion:

$$A^i(z) = A^{i-1}(z) - k_i z^{-i} A^{i-1}(z^{-1}) \quad (6.70)$$

Similarly, we can define the so-called *backward prediction error* as

$$b^i[n] = x[n-i] - \sum_{k=1}^i a_k^i x[n+k-i] \quad (6.71)$$

whose  $z$ -transform is

$$B^i(z) = z^{-i} A^i(z^{-1})X(z) \quad (6.72)$$

Now combining Eqs. (6.68), (6.70), and (6.72), we obtain

$$E^i(z) = A^{i-1}(z)X(z) - k_i z^{-i} A^{i-1}(z^{-1})X(z) = E^{i-1}(z) - k_i B^{i-1}(z) \quad (6.73)$$

whose inverse  $z$ -transform is given by

$$e^i[n] = e^{i-1}[n] - k_i b^{i-1}[n-1] \quad (6.74)$$

Also, substituting Eqs. (6.70) into (6.72) and using Eq. (6.68), we obtain

$$B^i(z) = z^{-1} B^{i-1}(z) - k_i E^{i-1}(z) \quad (6.75)$$

whose inverse  $z$ -transform is given by

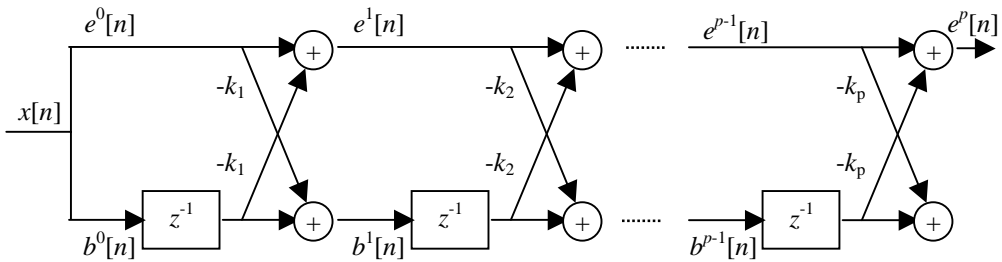
$$b^i[n] = b^{i-1}[n-1] - k_i e^{i-1}[n] \quad (6.76)$$

Equations (6.74) and (6.76) define the forward and backward prediction error sequences for an  $i^{\text{th}}$ -order predictor in terms of the corresponding forward and backward prediction errors of an  $(i-1)^{\text{th}}$ -order predictor. We initialize the recursive algorithm by noting that the  $0^{\text{th}}$ -order predictor is equivalent to using no predictor at all; thus

$$e^0[n] = b^0[n] = x[n] \quad (6.77)$$

and the final prediction error is  $e[n] = e^p[n]$ .

A block diagram of the lattice method is given in Figure 6.18, which resembles a lattice, whence its name.



**Figure 6.18** Block diagram of the lattice filter.

While the computation of the  $k_i$  coefficients can be done through the Levinson Durbin recursion of Eqs. (6.59) through (6.62), it can be shown that an equivalent calculation can be found as a function of the forward and backward prediction errors. To do so we minimize the sum of the forward prediction errors

$$E^i = \sum_{n=0}^{N-1} (e^i[n])^2 \quad (6.78)$$

by substituting Eq. (6.74) in (6.78), taking the derivative with respect to  $k_i$ , and equating to 0:

$$k_i = \frac{\sum_{n=0}^{N-1} e^{i-1}[n] b^{i-1}[n-1]}{\sum_{n=0}^{N-1} (b^{i-1}[n-1])^2} \quad (6.79)$$

Using Eqs. (6.67) and (6.71), it can be shown that



$$\sum_{n=0}^{N-1} (e^{i-1}[n])^2 = \sum_{n=0}^{N-1} (b^{i-1}[n-1])^2 \tag{6.80}$$

since minimization of both yields identical Yule-Walker equations. Thus Eq. (6.79) can be alternatively expressed as

$$k_i = \frac{\sum_{n=0}^{N-1} e^{i-1}[n]b^{i-1}[n-1]}{\sqrt{\sum_{n=0}^{N-1} (e^{i-1}[n])^2 \sum_{n=0}^{N-1} (b^{i-1}[n-1])^2}} = \frac{\langle \mathbf{e}^{i-1}, \mathbf{b}^{i-1} \rangle}{|\mathbf{e}^{i-1}| |\mathbf{b}^{i-1}|} \tag{6.81}$$

where we have defined the vectors  $\mathbf{e}^i = (e^i[0] \cdots e^i[N-1])$  and  $\mathbf{b}^i = (b^i[0] \cdots b^i[N-1])$ . The inner product of two vectors  $\mathbf{x}$  and  $\mathbf{y}$  is defined as

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{n=0}^{N-1} x[n]y[n] \tag{6.82}$$

and its norm as

$$|\mathbf{x}|^2 = \langle \mathbf{x}, \mathbf{x} \rangle = \sum_{n=0}^{N-1} x^2[n] \tag{6.83}$$

Equation (6.81) has the form of a normalized cross-correlation function, and, therefore, the reason the reflection coefficients are also called *partial correlation coefficients* (PARCOR). As with any normalized cross-correlation function, the  $k_i$  coefficients are bounded by

$$-1 \leq k_i \leq 1 \tag{6.84}$$

This is a necessary and sufficient condition for all the roots of the polynomial  $A(z)$  to be inside the unit circle, therefore guaranteeing a stable filter. This condition can be checked to avoid numerical imprecision by stopping the recursion if the condition is not met. The inverse lattice filter can be seen in Figure 6.19, which resembles the lossless tube model. This is why the  $k_i$  are also called *reflection coefficients*.

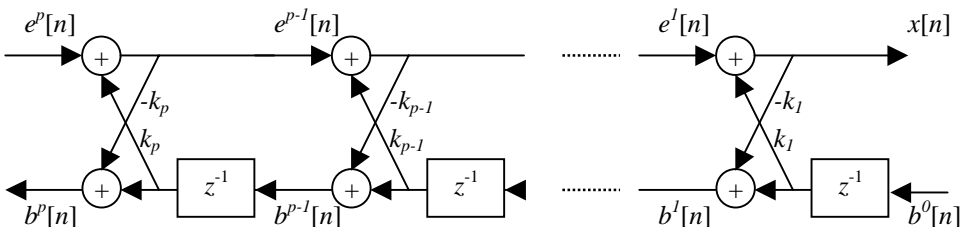


Figure 6.19 Inverse lattice filter used to generate the speech signal, given its residual.

Lattice filters are often used in fixed-point implementation, because lack of precision doesn't result in unstable filters. Any error that may take place – for example due to quantization – is generally not sufficient to cause  $k_i$  to fall outside the range in Eq. (6.84). If, owing to round-off error, the reflection coefficient falls outside the range, the lattice filter can be ended at the previous step.

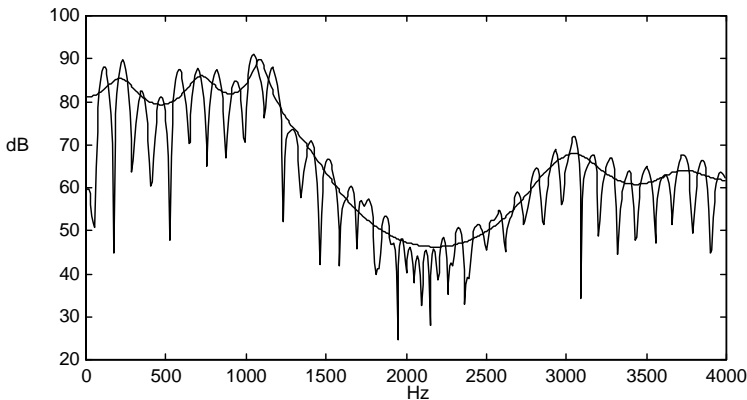
More importantly, linearly varying coefficients can be implemented in this fashion. While, typically, the reflection coefficients are constant during the analysis frame, we can implement a linear interpolation of the reflection coefficients to obtain the error signal. If the coefficients of both frames are in the range in Eq. (6.84), the linearly interpolated reflection coefficients also have that property, and thus the filter is stable. This is a property that the predictor coefficients don't have.

### 6.3.3. Spectral Analysis via LPC

Let's now analyze the frequency-domain behavior of the LPC analysis by evaluating

$$H(e^{j\omega}) = \frac{G}{1 - \sum_{k=1}^p a_k e^{-j\omega k}} = \frac{G}{A(e^{j\omega})} \quad (6.85)$$

which is an *all-pole* or IIR filter. If we plot  $H(e^{j\omega})$ , we expect to see peaks at the roots of the denominator. Figure 6.20 shows the 14-order LPC spectrum of the vowel of Figure 6.3 (d).



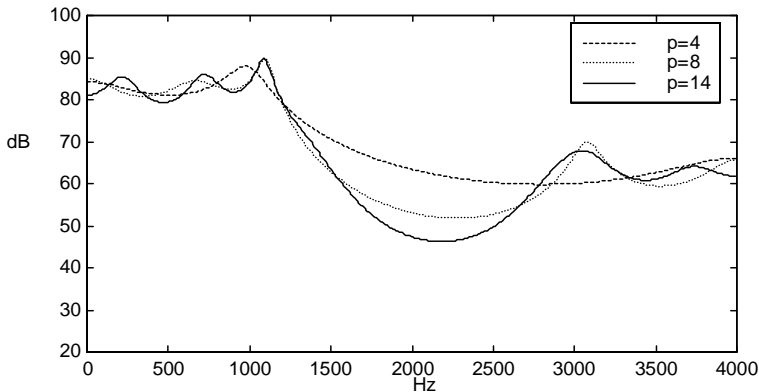
**Figure 6.20** LPC spectrum of the /ah/ phoneme in the word *lifes* of Figure 6.3. Used here are a 30-ms Hamming window and the autocorrelation method with  $p = 14$ . The short-time spectrum is also shown.

For the autocorrelation method, the squared error of Eq. (6.52) can be expressed, using Eq. (6.85) and Parseval's theorem, as

$$E_m = \frac{G^2}{2\pi} \int_{-\pi}^{\pi} \frac{|X_m(e^{j\omega})|^2}{|H(e^{j\omega})|^2} d\omega \quad (6.86)$$

Since the integrand in Eq. (6.86) is positive, minimizing  $E_m$  is equivalent to minimizing the ratio of the energy spectrum of the speech segment  $|X_m(e^{j\omega})|^2$  to the magnitude squared of the frequency response of the linear system  $|H(e^{j\omega})|^2$ . The LPC spectrum matches more closely the peaks than the valleys (see Figure 6.20), because the regions where  $|X_m(e^{j\omega})| > |H(e^{j\omega})|$  contribute more to the error than those where  $|H(e^{j\omega})| > |X_m(e^{j\omega})|$ .

Even nasals, which have zeros in addition to poles, can be represented with an infinite number of poles. In practice, if  $p$  is large enough we can approximate the signal spectrum with arbitrarily small error. Figure 6.21 shows different fits for different values of  $p$ . The higher  $p$ , the more details of the spectrum are preserved.



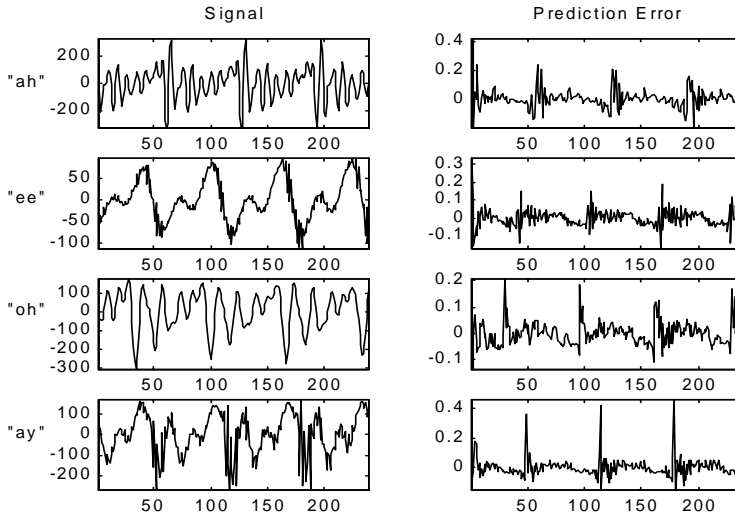
**Figure 6.21** LPC spectra of Figure 6.20 for various values of the predictor order  $p$ .

The prediction order is not known for arbitrary speech, so we need to set it to balance spectral detail with estimation errors.

### 6.3.4. The Prediction Error

So far, we have concentrated on the filter component of the source-filter model. Using Eq. (6.23), we can compute the prediction error signal, also called the *excitation*, or *residual* signal. For unvoiced speech synthetically generated by white noise following an LPC filter we expect the residual to be approximately white noise. In practice, this approximation is quite good, and replacement of the residual by white noise followed by the LPC filter typically results in no audible difference. For voiced speech synthetically generated by an impulse train following an LPC filter, we expect the residual to approximate an impulse train. In practice, this is not the case, because the all-pole assumption is not altogether valid; thus, the residual, although it contains spikes, is far from an impulse train. Replacing the residual by an impulse train, followed by the LPC filter, results in speech that sounds somewhat ro-

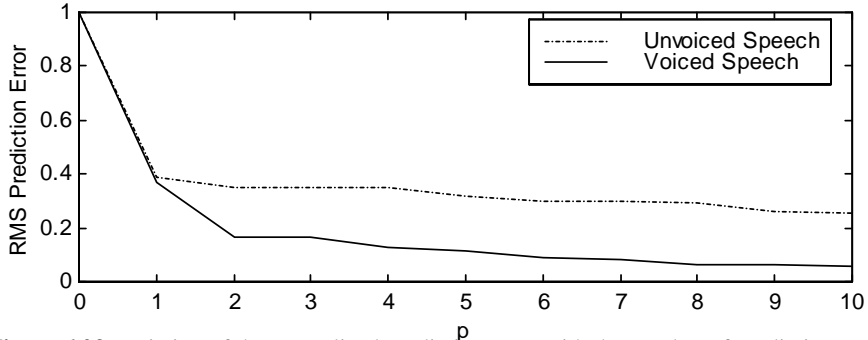
botic, partly because real speech is not perfectly periodic (it has a random component as well), and because the zeroes are not modeled with the LPC filter. Residual signals computed from inverse LPC filters for several vowels are shown in Figure 6.22.



**Figure 6.22** LPC prediction error signals for several vowels.

How do we choose  $p$ ? This is an important design question. Larger values of  $p$  lead to lower prediction errors (see Figure 6.23). Unvoiced speech has higher error than voiced speech, because the LPC model is more accurate for voiced speech. In general, the normalized error rapidly decreases, and then converges to a value of around 12 - 14 for 8 kHz speech. If we use a large value of  $p$ , we are fitting the individual harmonics; thus the LPC filter is modeling the source, and the separation between source and filter is not going to be so good. The more coefficients we have to estimate, the larger the variance of their estimates, since the number of available samples is the same. A rule of thumb is to use 1 complex pole per kHz plus 2 - 4 poles to model the radiation and glottal effects.

For unvoiced speech, both the autocorrelation and the covariance methods provide similar results. For voiced speech, however, the covariance method can provide better estimates if the analysis window is shorter than the local pitch period and the window only includes samples from the closed phase (when the vocal tract is closed at the glottis and speech signal is due mainly to free resonances). This is called *pitch synchronous* analysis and results in lower prediction error, because the true excitation is close to zero during the whole analysis window. During the open phase, the trachea, the vocal folds, and the vocal tract are acoustically coupled, and this coupling will change the free resonances. Additionally, the prediction error is higher for both the autocorrelation and the covariance methods if samples from the open phase are included in the analysis window, because the prediction during those instants is poor.



**Figure 6.23** Variation of the normalized prediction error with the number of prediction coefficients  $p$  for the voiced segment of Figure 6.3 and the unvoiced speech of Figure 6.5. The auto-correlation method was used with a 30 ms Hamming window, and a sampling rate of 8 kHz.

### 6.3.5. Equivalent Representations

There are a number of alternate useful representations of the predictor coefficients. The most important are the line spectrum pairs, reflection coefficients, log-area ratios, and the roots of the predictor polynomial.

#### 6.3.5.1. Line Spectral Frequencies

Line Spectral Frequencies (LSF) [18] provide an equivalent representation of the predictor coefficients that is very popular in speech coding. It is derived from computing the roots of the polynomials  $P(z)$  and  $Q(z)$  defined as

$$P(z) = A(z) + z^{-(p+1)} A(z^{-1}) \quad (6.87)$$

$$Q(z) = A(z) - z^{-(p+1)} A(z^{-1}) \quad (6.88)$$

To gain insight on these roots, look at a second-order predictor filter with a pair of complex roots:

$$A(z) = 1 - a_1 z^{-1} - a_2 z^{-2} = 1 - 2\rho_0 \cos(2\pi f_0) z^{-1} + \rho_0^2 z^{-2} \quad (6.89)$$

where  $0 < \rho_0 < 1$  and  $0 < f_0 < 0.5$ . Inserting Eq. (6.89) into (6.87) and (6.88) results in

$$P(z) = 1 - (a_1 + a_2)z^{-1} - (a_1 + a_2)z^{-2} + z^{-3} \quad (6.90)$$

$$Q(z) = 1 - (a_1 - a_2)z^{-1} + (a_1 - a_2)z^{-2} - z^{-3}$$

From Eq. (6.90) we see that  $z = -1$  is a root of  $P(z)$  and  $z = 1$  a root of  $Q(z)$ , which can be divided out and results in

$$\begin{aligned} P(z) &= (1+z^{-1})(1-2\beta_1 z^{-1}+z^{-2}) \\ Q(z) &= (1-z^{-1})(1-2\beta_2 z^{-1}+z^{-2}) \end{aligned} \quad (6.91)$$

where  $\beta_1$  and  $\beta_2$  are given by

$$\begin{aligned} \beta_1 &= \frac{a_1 + a_2 + 1}{2} = \rho_0 \cos(2\pi f_0) + \frac{1 - \rho_0^2}{2} \\ \beta_2 &= \frac{a_1 - a_2 - 1}{2} = \rho_0 \cos(2\pi f_0) - \frac{1 - \rho_0^2}{2} \end{aligned} \quad (6.92)$$

It can be shown that  $|\beta_1| < 1$  and  $|\beta_2| < 1$  for all possible values of  $f_0$  and  $\rho_0$ . With this property, the roots of  $P(z)$  and  $Q(z)$  in Eq. (6.91) are complex and given by  $\beta_1 \pm j\sqrt{1-\beta_1^2}$  and  $\beta_2 \pm j\sqrt{1-\beta_2^2}$ , respectively. Because they lie in the unit circle, they can be uniquely represented by their angles

$$\begin{aligned} \cos(2\pi f_1) &= \rho_0 \cos(2\pi f_0) + \frac{1 - \rho_0^2}{2} \\ \cos(2\pi f_2) &= \rho_0 \cos(2\pi f_0) - \frac{1 - \rho_0^2}{2} \end{aligned} \quad (6.93)$$

where  $f_1$  and  $f_2$  are the *line spectral frequencies* of  $A(z)$ . Since  $|\rho_0| < 1$ ,  $\cos(2\pi f_2) < \cos(2\pi f_0)$ , and thus  $f_2 > f_0$ . It's also the case that  $\cos(2\pi f_1) > \cos(2\pi f_0)$  and thus  $f_1 < f_0$ . Furthermore, as  $\rho_0 \rightarrow 1$ , we see from Eq. (6.93) that  $f_1 \rightarrow f_0$  and  $f_2 \rightarrow f_0$ . We conclude that, given a pole at  $f_0$ , the two line spectral frequencies bracket it, *i.e.*,  $f_1 < f_0 < f_2$ , and that they are closer together as the pole of the second-order resonator gets closer to the unit circle.

We have proven that for a second-order predictor, the roots of  $P(z)$  and  $Q(z)$  lie in the unit circle, that  $\pm 1$  are roots, and that, once sorted, the roots of  $P(z)$  and  $Q(z)$  alternate. Although we do not prove it here, it can be shown that these conclusions hold for other predictor orders, and, therefore, the  $p$  predictor coefficients can be transformed into  $p$  line spectral frequencies. We also know that  $z=1$  is always a root of  $Q(z)$ , whereas  $z=-1$  is a root of  $P(z)$  for even  $p$  and a root of  $Q(z)$  for odd  $p$ .

To compute the LSF for  $p > 2$ , we replace  $z = \cos(\omega)$  and compute the roots of  $P(\omega)$  and  $Q(\omega)$  by any available root finding method. A popular technique, given that there are  $p$  roots which are real in  $\omega$  and bounded between 0 and 0.5, is to bracket them by observing changes in sign of both functions in a dense grid. To compute the predictor coefficients from the LSF coefficients we can factor  $P(z)$  and  $Q(z)$  as a product of second-order filters as in Eq. (6.91), and then  $A(z) = (P(z) + Q(z))/2$ .

In practice, LSF are useful because of *sensitivity* (a quantization of one coefficient generally results in a spectral change only around that frequency) and *efficiency* (LSF result

in low spectral distortion). This doesn't occur with other representations. As long as the LSF coefficients are ordered, the resulting LPC filter is stable, though the proof is beyond the scope of this book. LSF coefficients are used extensively in Chapter 7.

### 6.3.5.2. Reflection Coefficients

For the autocorrelation method, the predictor coefficients may be obtained from the reflection coefficients by the following recursion:

$$\begin{aligned} a_i^i &= k_i & i &= 1, \dots, p \\ a_j^i &= a_j^{i-1} - k_i a_{i-j}^{i-1} & 1 \leq j < i \end{aligned} \quad (6.94)$$

where  $a_i = a_i^p$ . Similarly, the reflection coefficients may be obtained from the prediction coefficients using a backward recursion of the form

$$\begin{aligned} k_i &= a_i^i & i &= p, \dots, 1 \\ a_j^{i-1} &= \frac{a_j^i + a_i^i a_{i-j}^i}{1 - k_i^2} & 1 \leq j < i \end{aligned} \quad (6.95)$$

where we initialize  $a_i^p = a_i$ .

Reflection coefficients are useful when implementing LPC filters whose values are interpolated over time, because, unlike the predictor coefficients, they are guaranteed to be stable at all times as long as the anchors satisfy Eq. (6.84).

### 6.3.5.3. Log-Area Ratios

The *log-area ratio* coefficients are defined as

$$g_i = \ln \left( \frac{1 - k_i}{1 + k_i} \right) \quad (6.96)$$

with the inverse being given by

$$k_i = \frac{1 - e^{g_i}}{1 + e^{g_i}} \quad (6.97)$$

The log-area ratio coefficients are equal to the natural logarithm of the ratio of the areas of adjacent sections of a lossless tube equivalent of the vocal tract having the same transfer function. Since for stable predictor filters  $-1 < k_i < 1$ , we have from Eq. (6.96) that  $-\infty < g_i < \infty$ . For speech signals, it is not uncommon to have some reflection coefficients close to 1, and quantization of those values can cause a large change in the predictor's transfer function. On the other hand, the log-area ratio coefficients have relatively flat spectral

sensitivity (i.e., a small change in their values causes a small change in the transfer function) and thus are useful in coding.

#### 6.3.5.4. Roots of Polynomial

An alternative to the predictor coefficients results from computing the complex roots of the predictor polynomial:

$$A(z) = 1 - \sum_{k=1}^p a_k z^{-k} = \prod_{k=1}^p (1 - z_k z^{-1}) \quad (6.98)$$

These roots can be represented as

$$z_k = e^{(-\pi b_k + j2\pi f_k)/F_s} \quad (6.99)$$

where  $b_k$ ,  $f_k$ , and  $F_s$  represent the bandwidth, center frequency, and sampling frequency, respectively. Since  $a_k$  are real, all complex roots occur in conjugate pairs so that if  $(b_k, f_k)$  is a root, so is  $(b_k, -f_k)$ . The bandwidths  $b_k$  are always positive, because the roots are inside the unit circle ( $|z_k| < 1$ ) for a stable predictor. Real roots  $z_k = e^{-\pi b_k / F_s}$  can also occur. While algorithms exist to compute the complex roots of a polynomial, in practice there are sometimes numerical difficulties in doing so.

If the roots are available, it is straightforward to compute the predictor coefficients by using Eq. (6.98). Since the roots of the predictor polynomial represent resonance frequencies and bandwidths, they are used in formant synthesizers of Chapter 16.

## 6.4. CEPSTRAL PROCESSING

A *homomorphic* transformation  $\hat{x}[n] = D(x[n])$  is a transformation that converts a convolution

$$x[n] = e[n] * h[n] \quad (6.100)$$

into a sum

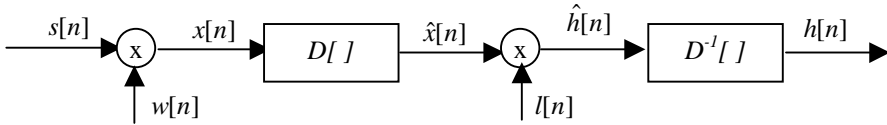
$$\hat{x}[n] = \hat{e}[n] + \hat{h}[n] \quad (6.101)$$

In this section we introduce the *cepstrum* as one homomorphic transformation [32] that allows us to separate the source from the filter. We show that we can find a value  $N$  such that the cepstrum of the filter  $\hat{h}[n] \approx 0$  for  $n \geq N$ , and that the cepstrum of the excitation  $\hat{e}[n] \approx 0$  for  $n < N$ . With this assumption, we can approximately recover both  $e[n]$  and  $h[n]$  from  $\hat{x}[n]$  by homomorphic filtering. In Figure 6.24, we show how to recover  $h[n]$  with a homomorphic filter:



$$l[n] = \begin{cases} 1 & |n| < N \\ 0 & |n| \geq N \end{cases} \quad (6.102)$$

where  $D$  is the cepstrum operator.



**Figure 6.24** Homomorphic filtering to recover the filter's response from a periodic signal. We have used the homomorphic filter of Eq. (6.102).

The excitation signal can similarly be recovered with a homomorphic filter given by

$$l[n] = \begin{cases} 1 & |n| \geq N \\ 0 & |n| < N \end{cases} \quad (6.103)$$

### 6.4.1. The Real and Complex Cepstrum

The *real cepstrum* of a digital signal  $x[n]$  is defined as

$$c[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln |X(e^{j\omega})| e^{j\omega n} d\omega \quad (6.104)$$

and the *complex cepstrum* of  $x[n]$  is defined as

$$\hat{x}[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln X(e^{j\omega}) e^{j\omega n} d\omega \quad (6.105)$$

where the complex logarithm is used:

$$\hat{X}(e^{j\omega}) = \ln X(e^{j\omega}) = \ln |X(e^{j\omega})| + j\theta(\omega) \quad (6.106)$$

and the phase  $\theta(\omega)$  is given by

$$\theta(\omega) = \arg [X(e^{j\omega})] \quad (6.107)$$

You can see from Eqs. (6.104) and (6.105) that both the real and the complex cepstrum satisfy Eq. (6.101) and thus they are homomorphic transformations.

If the signal  $x[n]$  is real, both the real cepstrum  $c[n]$  and the complex cepstrum  $\hat{x}[n]$  are also real signals. Therefore the term complex cepstrum doesn't mean that it is a complex signal but rather that the complex logarithm is taken.

It can easily be shown that  $c[n]$  is the even part of  $\hat{x}[n]$ :

$$c[n] = \frac{\hat{x}[n] + \hat{x}[-n]}{2} \quad (6.108)$$

From here on, when we refer to cepstrum without qualifiers, we are referring to the real cepstrum, since it is the most widely used in speech technology.

The cepstrum was invented by Bogert et al. [6], and its term was coined by reversing the first syllable of the word spectrum, given that it is obtained by taking the inverse Fourier transform of the log-spectrum. Similarly, they defined the term *quefrency* to represent the independent variable  $n$  in  $c[n]$ . The quefrency has dimension of time.

## 6.4.2. Cepstrum of Pole-Zero Filters

A very general type of filters are those with rational transfer functions

$$H(z) = \frac{Az^r \prod_{k=1}^{M_i} (1 - a_k z^{-1}) \prod_{k=1}^{M_o} (1 - u_k z)}{\prod_{k=1}^{N_i} (1 - b_k z^{-1}) \prod_{k=1}^{N_o} (1 - v_k z)} \quad (6.109)$$

with the magnitudes of  $a_k$ ,  $b_k$ ,  $u_k$  and  $v_k$  all less than 1. Therefore,  $(1 - a_k z^{-1})$  and  $(1 - b_k z^{-1})$  represent the zeros and poles inside the unit circle, whereas  $(1 - u_k z)$  and  $(1 - v_k z)$  represent the zeros and poles outside the unit circle, and  $z^r$  is a shift from the time origin. Thus, the complex logarithm is

$$\begin{aligned} \hat{H}(z) = & \ln[A] + \ln[z^r] + \sum_{k=1}^{M_i} \ln(1 - a_k z^{-1}) \\ & - \sum_{k=1}^{N_i} \ln(1 - b_k z^{-1}) + \sum_{k=1}^{M_o} \ln(1 - u_k z) - \sum_{k=1}^{N_o} \ln(1 - v_k z) \end{aligned} \quad (6.110)$$

where the term  $\log[z^r]$  contributes to the imaginary part of the complex cepstrum only with a term  $j\omega r$ . Since it just carries information about the time origin, it's typically ignored. We use the Taylor series expansion

$$\ln(1 - x) = - \sum_{n=1}^{\infty} \frac{x^n}{n} \quad (6.111)$$

in Eq. (6.110) and take inverse  $z$ -transforms to obtain

$$\hat{h}[n] = \begin{cases} \log[A] & n = 0 \\ \sum_{k=1}^{N_i} \frac{b_k^n}{n} - \sum_{k=1}^{M_i} \frac{a_k^n}{n} & n > 0 \\ \sum_{k=1}^{M_o} \frac{u_k^n}{n} - \sum_{k=1}^{N_o} \frac{v_k^n}{n} & n < 0 \end{cases} \quad (6.112)$$

If the filter's impulse response doesn't have zeros or poles outside the unit circle, the so-called *minimum phase* signals, then  $\hat{h}[n] = 0$  for  $n < 0$ . *Maximum phase* signals are those with  $\hat{h}[n] = 0$  for  $n > 0$ . If a signal is minimum phase, its complex cepstrum can be uniquely determined from its real cepstrum:

$$\hat{h}[n] = \begin{cases} 0 & n < 0 \\ c[n] & n = 0 \\ 2c[n] & n > 0 \end{cases} \quad (6.113)$$

It is easy to see from Eq. (6.112) that both the real and complex cepstrum are decaying sequences, which is the reason why, typically, a finite number of coefficients are sufficient to approximate it, and, therefore, people refer to the truncated cepstrum signal as a *cepstrum vector*.

### 6.4.2.1. LPC-Cepstrum

The case when the rational transfer function in Eq. (6.109) has been obtained with an LPC analysis is particularly interesting, since LPC analysis is such a widely used method. While Eq. (6.112) applies here, too, it is useful to find a recursion which doesn't require us to compute the roots of the predictor polynomial. Given the LPC filter

$$H(z) = \frac{G}{1 - \sum_{k=1}^p a_k z^{-k}} \quad (6.114)$$

we take the logarithm

$$\hat{H}(z) = \ln G - \ln \left( 1 - \sum_{l=1}^p a_l z^{-l} \right) = \sum_{k=-\infty}^{\infty} \hat{h}[k] z^{-k} \quad (6.115)$$

and the derivative of both sides with respect to  $z$

$$\frac{-\sum_{n=1}^p na_n z^{-n-1}}{1 - \sum_{l=1}^p a_l z^{-l}} = -\sum_{k=-\infty}^{\infty} k\hat{h}[k]z^{-k-1} \quad (6.116)$$

Multiplying both sides by  $-z\left(1 - \sum_{l=1}^p a_l z^{-l}\right)$ , we obtain

$$\sum_{n=1}^p na_n z^{-n} = \sum_{n=-\infty}^{\infty} n\hat{h}[n]z^{-n} - \sum_{l=1}^p \sum_{k=-\infty}^{\infty} k\hat{h}[k]a_l z^{-k-l} \quad (6.117)$$

which, after replacing  $l = n - k$ , and equating terms in  $z^{-1}$ , results in

$$\begin{aligned} na_n &= n\hat{h}[n] - \sum_{k=1}^{n-1} k\hat{h}[k]a_{n-k} & 0 < n \leq p \\ 0 &= n\hat{h}[n] - \sum_{k=n-p}^{n-1} k\hat{h}[k]a_{n-k} & n > p \end{aligned} \quad (6.118)$$

so that the complex cepstrum can be obtained from the LPC coefficients by the following recursion:

$$\hat{h}[n] = \begin{cases} 0 & n < 0 \\ \ln G & n = 0 \\ a_n + \sum_{k=1}^{n-1} \left(\frac{k}{n}\right) \hat{h}[k]a_{n-k} & 0 < n \leq p \\ \sum_{k=n-p}^{n-1} \left(\frac{k}{n}\right) \hat{h}[k]a_{n-k} & n > p \end{cases} \quad (6.119)$$

where the value for  $n = 0$  can be obtained from Eqs. (6.115) and (6.111). We note that, while there are a finite number of LPC coefficients, the number of cepstrum coefficients is infinite. Speech recognition researchers have shown empirically that a finite number is sufficient: 12 - 20 depending on the sampling rate and whether or not frequency warping is done. In Chapter 8 we discuss the use of the cepstrum in speech recognition.

This recursion should not be used in the reverse mode to compute the LPC coefficients from *any* set of cepstrum coefficients, because the recursion in Eq. (6.119) assumes an all-pole model with all poles inside the unit circle, and that might not be the case for an arbitrary cepstrum sequence, so that the recursion might yield a set of unstable LPC coefficients. In some experiments it has been shown that quantized LPC-cepstrum can yield unstable LPC coefficients over 5% of the time.

### 6.4.3. Cepstrum of Periodic Signals

It is important to see what the cepstrum of periodic signals looks like. To do so, let's consider the following signal:

$$x[n] = \sum_{k=0}^{M-1} \alpha_k \delta[n - kN] \quad (6.120)$$

which can be viewed as an impulse train of period  $N$  multiplied by an analysis window, so that only  $M$  impulses remain. Its  $z$ -transform is

$$X(z) = \sum_{k=0}^{M-1} \alpha_k z^{-kN} \quad (6.121)$$

which is a polynomial in  $z^{-N}$  rather than  $z^{-1}$ . Therefore,  $X(z)$  can be expressed as a product of factors of the form  $(1 - a_k z^{-Nk})$  and  $(1 - u_k z^{Nk})$ . Following the derivation in Section 6.4.2, it is clear that its complex cepstrum is nonzero only at integer multiples of  $N$ :

$$\hat{x}[n] = \sum_{k=-\infty}^{\infty} \beta_k \delta[n - kN] \quad (6.122)$$

A particularly interesting case is when  $\alpha_k = \alpha^k$  with  $0 < \alpha < 1$ , so that Eq. (6.121) can be expressed as

$$X(z) = 1 + \alpha z^{-N} + \dots + (\alpha z^{-N})^{M-1} = \frac{1 - (\alpha z^{-N})^M}{1 - \alpha z^{-N}} \quad (6.123)$$

so that taking the logarithm of Eq. (6.123) and expanding it in Taylor series using Eq. (6.111) results in

$$\hat{X}(z) = \ln X(z) = \sum_{r=1}^{\infty} \frac{\alpha^r}{r} z^{-rN} - \sum_{l=1}^{\infty} \frac{\alpha^{lM}}{l} z^{-lMN} = \sum_{n=1}^{\infty} \hat{x}[n] z^{-n} \quad (6.124)$$

which lets us compute the complex cepstrum as

$$\hat{x}[n] = \sum_{r=1}^{\infty} \frac{\alpha^r}{r} \delta[n - rN] - \sum_{l=1}^{\infty} \frac{\alpha^{lM}}{l} \delta[n - lMN] \quad (6.125)$$

An infinite impulse train can be obtained by making  $\alpha \rightarrow 1$  and  $M \rightarrow \infty$  in Eq. (6.125):

$$\hat{x}[n] = \sum_{r=1}^{\infty} \frac{\delta[n - rN]}{r} \quad (6.126)$$

We see from Eq. (6.126) that the cepstrum of an impulse train goes to 0 as  $n$  increases. This justifies our assumption of homomorphic filtering.

### 6.4.4. Cepstrum of Speech Signals

We can compute the cepstrum of a speech segment by windowing the signal with a window of length  $N$ . In practice, the cepstrum is not computed through Eq. (6.112), since root-finding algorithms are slow and offer numerical imprecision for the large values of  $N$  used. Instead, we can compute the cepstrum directly through its definition of Eq. (6.105), using the DFT as follows:

$$X_a[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N}, \quad 0 \leq k < N \quad (6.127)$$

$$\hat{X}_a[k] = \ln X_a[k], \quad 0 \leq k < N \quad (6.128)$$

$$\hat{x}_a[n] = \frac{1}{N} \sum_{k=0}^{N-1} \hat{X}_a[k] e^{-j2\pi nk/N}, \quad 0 \leq n < N \quad (6.129)$$

The subscript  $a$  means that the new complex cepstrum  $\hat{x}_a[n]$  is an aliased version of  $\hat{x}[n]$  given by

$$\hat{x}_a[n] = \sum_{r=-\infty}^{\infty} \hat{x}[n+rN] \quad (6.130)$$

which can be derived by using the sampling theorem of Chapter 5, by reversing the concepts of time and frequency.

This aliasing introduces errors in the estimation that can be reduced by choosing a large value for  $N$ .

Computation of the complex cepstrum requires computing the complex logarithm and, in turn, the phase. However, given the principal value of the phase  $\theta_p[k]$ , there are infinite possible values for  $\theta[k]$ :

$$\theta[k] = \theta_p[k] + 2\pi n_k \quad (6.131)$$

From Chapter 5 we know that if  $x[n]$  is real,  $\arg[X(e^{j\omega})]$  is an odd function and also continuous. Thus we can do *phase unwrapping* by choosing  $n_k$  to guarantee that  $\theta[k]$  is a smooth function, i.e., by forcing the difference between adjacent values to be small:

$$|\theta[k] - \theta[k-1]| < \pi \quad (6.132)$$

A linear phase term  $r$  as in Eq. (6.110), would contribute to the phase difference in Eq. (6.132) with  $2\pi r/N$ , which may result in errors in the phase unwrapping if  $\theta[k]$  is changing sufficiently rapidly. In addition, there could be large changes in the phase difference if  $X_a[k]$  is noisy. To guarantee that we can track small phase differences, a value of  $N$  several

times larger than the window size is required: *i.e.*, the input signal has to be zero-padded prior to the FFT computation. Finally, the delay  $r$  in Eq. (6.109), can be obtained by forcing the phase to be an odd function, so that:

$$\theta[N/2] = \pi r \quad (6.133)$$

For unvoiced speech, the unwrapped phase is random, and therefore only the real cepstrum has meaning. In practical situations, even voiced speech has some frequencies at which noise dominates (typically very low and high frequencies), which results in phase  $\theta[k]$  that changes drastically from frame to frame. Because of this, the complex cepstrum in Eq. (6.105) is rarely used for real speech signals. Instead, the real cepstrum is used much more often:

$$C_a[k] = \ln |X_a[k]|, \quad 0 \leq k < N \quad (6.134)$$

$$c_a[n] = \frac{1}{N} \sum_{k=0}^{N-1} C_a[k] e^{-j2\pi nk/N}, \quad 0 \leq n < N \quad (6.135)$$

Similarly, it can be shown that for the new real cepstrum  $c_a[n]$  is an aliased version of  $c[n]$  given by

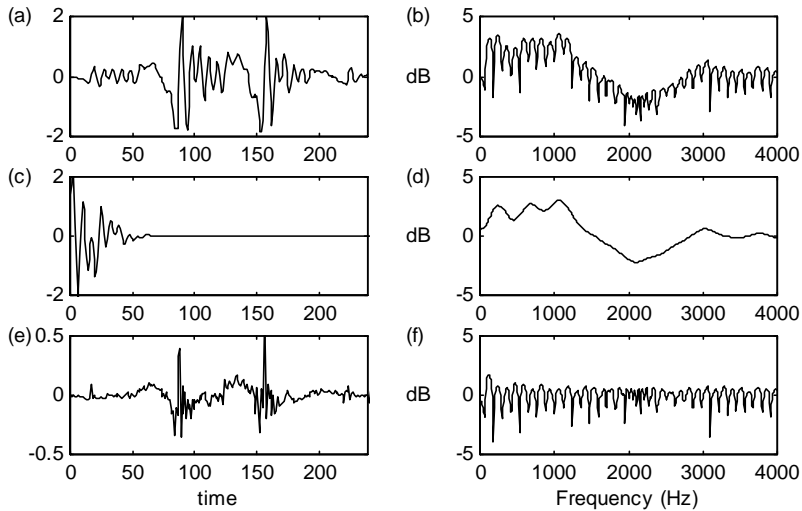
$$c_a[n] = \sum_{r=-\infty}^{\infty} c[n+rN] \quad (6.136)$$

which again has aliasing that can be reduced by choosing a large value for  $N$ .

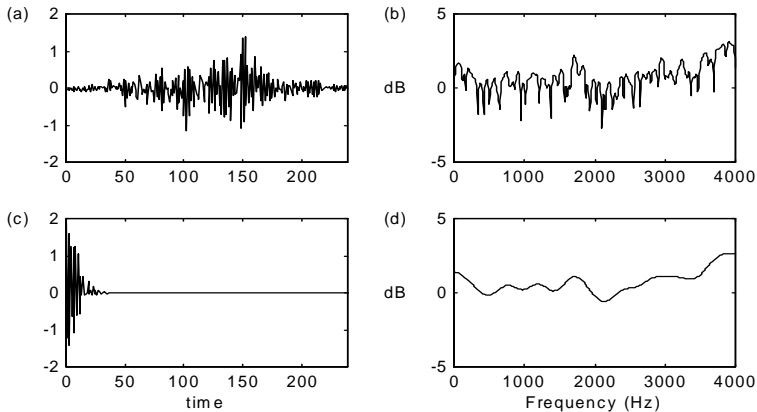
### 6.4.5. Source-Filter Separation via the Cepstrum

We have seen that, if the filter is a rational transfer function, and the source is an impulse train, the homomorphic filtering of Figure 6.24 can approximately separate them. Because of problems in estimating the phase in speech signals (see Section 6.4.4), we generally compute the real cepstrum using Eqs. (6.127), (6.134) and (6.135), and then compute the complex cepstrum under the assumption of a minimum phase signal according to Eq. (6.113). The result of separating source and filter using this cepstral deconvolution is shown in Figure 6.25 for voiced speech and Figure 6.26 for unvoiced speech.

The real cepstrum of white noise  $x[n]$  with an expected magnitude spectrum  $|X(e^{j\omega})| = 1$  is 0. If colored noise is present, the cepstrum of the observed colored noise  $\hat{y}[n]$  is identical to the cepstrum of the coloring filter  $\hat{h}[n]$ , except for a gain factor. The above is correct if we take an infinite number of noise samples, but in practice, this cannot be done and a limited number have to be used, so that this is only an approximation, though it is often used in speech processing algorithms.



**Figure 6.25** Separation of source and filter using homomorphic filtering for voiced speech with the scheme of Figure 6.24 with  $N = 20$  in the homomorphic filter of Eq. (6.102) with the real cepstrum: (a) windowed signal, (b) log-spectrum, (c) filter's impulse response, (d) smoothed log-spectrum, (e) windowed excitation signal, (f) log-spectrum of high-part of cepstrum. Note that the windowed excitation is not a windowed impulse train because of the minimum phase assumption.



**Figure 6.26** Separation of source and filter using homomorphic filtering for unvoiced speech with the scheme of Figure 6.24 with  $N = 20$  in the homomorphic filter of Eq. (6.102) with the real cepstrum: (a) windowed signal, (b) log-spectrum, (c) filter's impulse response, (d) smoothed log-spectrum.



## 6.5. PERCEPTUALLY-MOTIVATED REPRESENTATIONS

In this section we describe some aspects of human perception, and methods motivated by the behavior of the human auditory system: Mel-Frequency Cepstrum Coefficients (MFCC) and Perceptual Linear Prediction (PLP). These methods have been successfully used in speech recognition. First we present several nonlinear frequency scales that have been used in such representations.

### 6.5.1. The Bilinear Transform

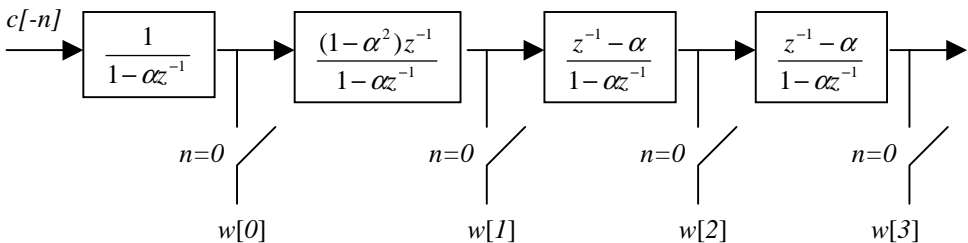
The transformation

$$s = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}} \tag{6.137}$$

for  $0 < \alpha < 1$  belongs to the class of *bilinear* transforms. It is a mapping in the complex plane that maps the unit circle onto itself. The frequency transformation is obtained by making the substitution  $z = e^{j\omega}$  and  $s = e^{j\Omega}$ :

$$\Omega = \omega + 2 \arctan \left[ \frac{\alpha \sin(\omega)}{1 - \alpha \cos(\omega)} \right] \tag{6.138}$$

This transformation is very similar to the Bark and mel scale for an appropriate choice of the parameter  $\alpha$  (see Chapter 2). Oppenheim [31] showed that the advantage of this transformation is that it can be used to transform a time sequence in the linear frequency into another time sequence in the warped frequency, as shown in Figure 6.27. This bilinear transform has been successfully applied to cepstral and autocorrelation coefficients.



**Figure 6.27** Implementation of the frequency-warped cepstral coefficients as a function of the linear-frequency cepstrum coefficients. Both sets of coefficients are causal. The input is the time-reversed cepstrum sequence, and the output can be obtained by sampling the outputs of the filters at time  $n = 0$ . The filters used for  $w[m]$   $m > 2$  are the same. Note that, for a finite-length cepstrum, an infinite-length warped cepstrum results.

For a finite number of cepstral coefficients the bilinear transform in Figure 6.27 results in an infinite number of warped cepstral coefficients. Since truncation is usually done in practice, the bilinear transform is equivalent to a matrix multiplication, where the matrix is a function of the warping parameter  $\alpha$ . Shikano [43] showed these warped cepstral coefficients were beneficial for speech recognition.

## 6.5.2. Mel-Frequency Cepstrum

The *Mel-Frequency Cepstrum Coefficients* (MFCC) is a representation defined as the real cepstrum of a windowed short-time signal derived from the FFT of that signal. The difference from the real cepstrum is that a nonlinear frequency scale is used, which approximates the behavior of the auditory system. Davis and Mermelstein [8] showed the MFCC representation to be beneficial for speech recognition.

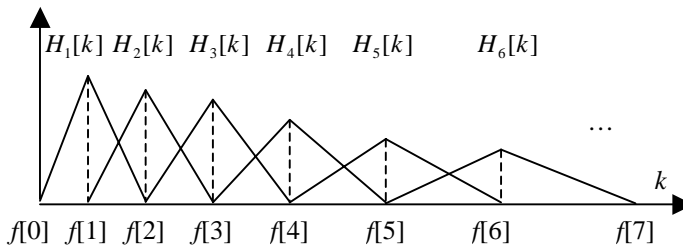
Given the DFT of the input signal

$$X_a[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N}, \quad 0 \leq k < N \quad (6.139)$$

we define a filterbank with  $M$  filters ( $m = 1, 2, \dots, M$ ), where filter  $m$  is triangular filter given by:

$$H_m[k] = \begin{cases} 0 & k < f[m-1] \\ \frac{2(k - f[m-1])}{(f[m+1] - f[m-1])(f[m] - f[m-1])} & f[m-1] \leq k \leq f[m] \\ \frac{2(f[m+1] - k)}{(f[m+1] - f[m-1])(f[m+1] - f[m])} & f[m] \leq k \leq f[m+1] \\ 0 & k > f[m+1] \end{cases} \quad (6.140)$$

Such filters compute the average spectrum around each center frequency with increasing bandwidths, and they are displayed in Figure 6.28.



**Figure 6.28** Triangular filters used in the computation of the mel-cepstrum using Eq. (6.140).

Alternatively, the filters can be chosen as

$$H_m^i[k] = \begin{cases} 0 & k < f[m-1] \\ \frac{(k - f[m-1])}{(f[m] - f[m-1])} & f[m-1] \leq k \leq f[m] \\ \frac{(f[m+1] - k)}{(f[m+1] - f[m])} & f[m] \leq k \leq f[m+1] \\ 0 & k > f[m+1] \end{cases} \quad (6.141)$$

which satisfies  $\sum_{m=0}^{M-1} H_m^i[k] = 1$ . The mel-cepstrum computed with  $H_m[k]$  or  $H_m^i[k]$  will differ by a constant vector for all inputs, so the choice becomes unimportant when used in a speech recognition system that has trained with the same filters.

Let's define  $f_l$  and  $f_h$  to be the lowest and highest frequencies of the filterbank in Hz,  $F_s$  the sampling frequency in Hz,  $M$  the number of filters, and  $N$  the size of the FFT. The boundary points  $f[m]$  are uniformly spaced in the mel-scale:

$$f[m] = \left( \frac{N}{F_s} \right) B^{-1} \left( B(f_l) + m \frac{B(f_h) - B(f_l)}{M+1} \right) \quad (6.142)$$

where the mel-scale  $B$  is given by Eq. (2.6), and  $B^{-1}$  is its inverse

$$B^{-1}(b) = 700(\exp(b/1125) - 1) \quad (6.143)$$

We then compute the log-energy at the output of each filter as

$$S[m] = \ln \left[ \sum_{k=0}^{N-1} |X_a[k]|^2 H_m[k] \right], \quad 0 \leq m < M \quad (6.144)$$

The mel frequency cepstrum is then the discrete cosine transform of the  $M$  filter outputs:

$$c[n] = \sum_{m=0}^{M-1} S[m] \cos(\pi n(m+1/2)/M) \quad 0 \leq n < M \quad (6.145)$$

where  $M$  varies for different implementations from 24 to 40. For speech recognition, typically only the first 13 cepstrum coefficients are used. It is important to note that the MFCC representation is no longer a homomorphic transformation. It would be if the order of summation and logarithms in Eq. (6.144) were reversed:

$$S[m] = \sum_{k=0}^{N-1} \ln \left( |X_a[k]|^2 H_m[k] \right) \quad 0 \leq m < M \quad (6.146)$$

In practice, however, the MFCC representation is approximately homomorphic for filters that have a smooth transfer function. The advantage of the MFCC representation using

(6.144) instead of (6.146) is that the filter energies are more robust to noise and spectral estimation errors. This algorithm has been used extensively as a feature vector for speech recognition systems.

While the definition of cepstrum in Section 6.4.1 uses an inverse DFT, since  $S[m]$  is even, a DCT-II can be used instead (see Chapter 5).

### 6.5.3. Perceptual Linear Prediction (PLP)

*Perceptual Linear Prediction* (PLP) [16] uses the standard Durbin recursion of Section 6.3.2.2 to compute LPC coefficients, and typically the LPC coefficients are transformed to LPC-cepstrum using the recursion in Section 6.4.2.1. But unlike standard linear prediction, the autocorrelation coefficients are not computed in the time domain through Eq. (6.55).

The autocorrelation  $R_x[n]$  is the inverse Fourier transform of the power spectrum  $|X(\omega)|^2$  of the signal. We cannot compute the continuous-frequency Fourier transform easily, but we can take an FFT to compute  $X[k]$ , so that the autocorrelation can be obtained as the inverse Fourier transform of  $|X[k]|^2$ . Since the discrete Fourier transform is not performing linear convolution but circular convolution, we need to make sure that the FFT size is larger than twice the window length (see Section 5.3.4) for this to hold. This alternate way of computing autocorrelation coefficients, entailing two FFTs and  $N$  multiplies and adds, should yield identical results. Since normally only a small number  $p$  of autocorrelation coefficients are needed, this is generally not a cost-effective way to do it, unless the first FFT has to be computed for other reasons.

Perceptual linear prediction uses the above method, but replaces  $|X[k]|^2$  by a perceptually motivated power spectrum. The most important aspect is the non-linear frequency scaling, which can be achieved through a set of filterbanks similar to those described in Section 6.5.2, so that this critical-band power spectrum can be sampled in approximately 1-bark intervals. Another difference is that, instead of taking the logarithm on the filterbank energy outputs, a different non-linearity compression is used, often the cubic root. It is reported [16] that the use of this different non-linearity is beneficial for speech recognizers in noisy conditions.

## 6.6. FORMANT FREQUENCIES

Formant frequencies are the resonances in the vocal tract and, as we saw in Chapter 2, they convey the differences between different sounds. Expert spectrogram readers are able to recognize speech by looking at a spectrogram, particularly at the formants. It has been argued that they are very useful features for speech recognition, but they haven't been widely used because of the difficulty in estimating them.

One way of obtaining formant candidates at a frame level is to compute the roots of a  $p^{\text{th}}$ -order LPC polynomial [3, 26]. There are standard algorithms to compute the complex

roots of a polynomial with real coefficients [36], though convergence is not guaranteed. Each complex root  $z_i$  can be represented as

$$z_i = \exp(-\pi b_i + j2\pi f_i) \quad (6.147)$$

where  $f_i$  and  $b_i$  are the formant frequency and bandwidth, respectively, of the  $i^{\text{th}}$  root. Real roots are discarded and complex roots are sorted by increasing  $f$ , discarding negative values. The remaining pairs  $(f_i, b_i)$  are the formant candidates. Traditional formant trackers discard roots whose bandwidths are higher than a threshold [46], say 200 Hz.

*Closed-phase* analysis of voiced speech [5] uses only the regions for which the glottis is closed and thus there is no excitation. When the glottis is open, there is a coupling of the vocal tract with the lungs and the resonance bandwidths are somewhat larger. Determination of the closed-phase regions directly from the speech signal is difficult, so often an *electroglottograph* (EGG) signal is used [23]. EGG signals, obtained by placing electrodes at the speaker's throat, are very accurate in determining the times when the glottis is closed. Using samples in the closed-phase covariance analysis can yield accurate results [46]. For female speech, the closed-phase is short, and sometimes non-existent, so such analysis can be a challenge. EGG signals are useful also for pitch tracking and are described in more detail in Chapter 16.

Another common method consists of finding the peaks on a smoothed spectrum, such as that obtained through an LPC analysis [26, 40]. The advantage of this method is that you can always compute the peaks and it is more computationally efficient than extracting the complex roots of a polynomial. On the other hand, this procedure generally doesn't estimate the formant's bandwidth. The first three formants are typically estimated this way for formant synthesis (see Chapter 16), since they are the ones that allow sound classification, whereas the higher formants are more speaker dependent.

Sometimes, the signal goes through some *conditioning*, which includes sampling rate conversion to remove frequencies outside the range we are interested in. For example, if we are interested only in the first three formants, we can safely downsample the input signal to 8 kHz, since we know all three formants should be below 4 kHz. This downsampling reduces computation and the chances of the algorithm to find formant values outside the expected range (otherwise peaks or roots could be chosen above 4 kHz which we know do not correspond to any of the first three formants). Pre-emphasis filtering is also often used to whiten the signal.

Because of the thresholds imposed above, it is possible that the formants are not continuous. For example, when the vocal tract's spectral envelope is changing rapidly, bandwidths obtained through the above methods are overestimates of the true bandwidths, and they may exceed the threshold and thus be rejected. It is also possible for the peak-picking algorithm to classify a harmonic as a formant during some regions where it is much stronger than the other harmonics. Due to the thresholds used, a given frame could have no formants, only one formant (either first, second, or third), two, three, or more. Formant alignment from one frame to another has often been done using heuristics to prevent such discontinuities.

### 6.6.1. Statistical Formant Tracking

It is desirable to have an approach that does not use any thresholds on formant candidates and uses a probabilistic model to do the tracking instead of heuristics [1]. The formant candidates can be obtained from roots of the LPC polynomial, peaks in the smoothed spectrum or even from a dense sample of possible points. If the first  $n$  formants are desired, and we have  $(p/2)$  formant candidates, a maximum of  $r$   $n$ -tuples are considered, where  $r$  is given by

$$r = \binom{p/2}{n} \quad (6.148)$$

A Viterbi search (see Chapter 8) is then carried out to find the most likely path of formant  $n$ -tuples given a model with some a priori knowledge of formants. The prior distribution for formant targets is used to determine which formant candidate to use of all possible choices for the given phoneme (*i.e.*, we know that F1 for an AE should be around 800 Hz). Formant continuity is imposed through the prior distribution of the formant slopes. This algorithm produces  $n$  formants for every frame, including silence.

Since we are interested in obtaining the first three formants ( $n=3$ ) and F3 is known to be lower than 4 kHz, it is advantageous to downsample the signal to 8 kHz in order to avoid obtaining formant candidates above 4 kHz and to let us use a lower-order analysis which offers fewer numerical problems when computing the roots. With  $p = 14$ , it results in a maximum of  $r = 35$  triplets for the case of no real roots.

Let  $\mathbf{X}$  be a sequence of  $T$  feature vectors  $\mathbf{x}_t$  of dimension  $n$ :

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T)' \quad (6.149)$$

where the prime denotes transpose.

We estimate the formants with the knowledge of what sound occurs at that particular time, for example by using a speech recognizer that segments the waveform into different phonemes (see Chapter 9) or states  $q_t$  within a phoneme. In this case we assume that the output distribution of each state  $i$  is modeled by one Gaussian density function with a mean  $\mu_i$  and covariance matrix  $\Sigma_i$ . We can define up to  $N$  states, with  $\lambda$  being the set of all means and covariance matrices for all:

$$\lambda = (\mu_1, \Sigma_1, \mu_2, \Sigma_2, \dots, \mu_N, \Sigma_N) \quad (6.150)$$

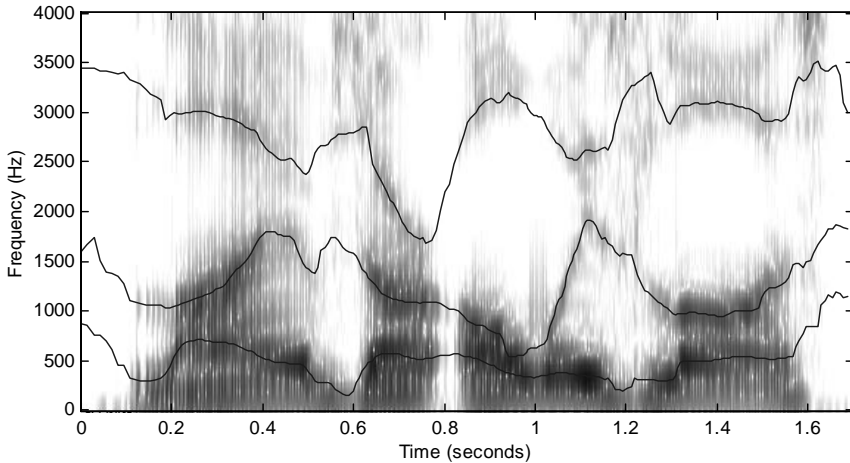
Therefore, the log-likelihood for  $\mathbf{X}$  is given by

$$\ln p(\mathbf{X} | \hat{\mathbf{q}}, \lambda) = -\frac{TM}{2} \ln(2\pi) - \frac{1}{2} \sum_{t=1}^T \ln |\Sigma_{q_t}| - \frac{1}{2} \sum_{t=1}^T (\mathbf{x}_t - \mu_{q_t})' \Sigma_{q_t}^{-1} (\mathbf{x}_t - \mu_{q_t}) \quad (6.151)$$

Maximizing  $\mathbf{X}$  in Eq. (6.151) leads to the trivial solution  $\hat{\mathbf{X}} = (\mu_{q_1}, \mu_{q_2}, \dots, \mu_{q_T})'$ , a piecewise function whose value is that of the best  $n$ -tuple candidate. This function has discontinuities at state boundaries and thus is not likely to represent well the physical phenomena of speech.

This problem arises because the slopes at state boundaries do not match the slopes of natural speech. To avoid these discontinuities, we would like to match not only the target formants at each state, but also the formant slopes at each state. To do that, we augment the feature vector  $\mathbf{x}_t$  at frame  $t$  with the delta vector  $\mathbf{x}_t - \mathbf{x}_{t-1}$ . Thus, we increase the parameter space of  $\lambda$  with the corresponding means  $\delta_i$  and covariance matrices  $\Gamma_i$  of these delta parameters, and assume statistical independence among them. The corresponding new log-likelihood has the form

$$\begin{aligned} \ln p(\mathbf{X} | \hat{\mathbf{q}}, \lambda) = & K - \frac{1}{2} \sum_{t=1}^T \ln |\Sigma_{q_t}| - \frac{1}{2} \sum_{t=2}^T \ln |\Gamma_{q_t}| \\ & - \frac{1}{2} \sum_{t=1}^T (\mathbf{x}_t - \mu_{q_t})' \Sigma_{q_t}^{-1} (\mathbf{x}_t - \mu_{q_t}) - \frac{1}{2} \sum_{t=2}^T (\mathbf{x}_t - \mathbf{x}_{t-1} - \delta_{q_t})' \Gamma_{q_t}^{-1} (\mathbf{x}_t - \mathbf{x}_{t-1} - \delta_{q_t}) \end{aligned} \quad (6.152)$$



**Figure 6.29** Spectrogram and three smoothed formants.

Maximization of Eq. (6.152) with respect to  $\mathbf{x}_t$  requires solving several sets of linear equations. If  $\Gamma_i$  and  $\Sigma_i$  are diagonal covariance matrices, it results in a set of linear equations for each of the  $M$  dimensions

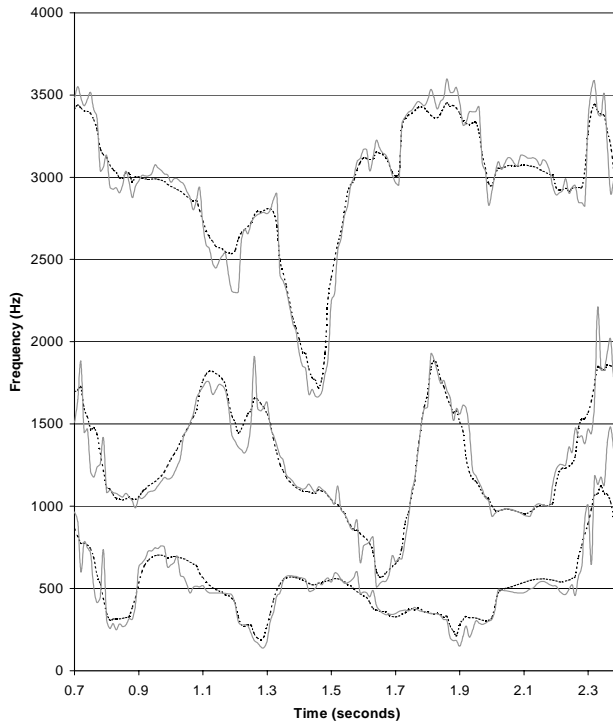
$$\mathbf{B}\mathbf{X} = \mathbf{c} \quad (6.153)$$

where  $\mathbf{B}$  is a tridiagonal matrix (all values are zero except for those in the main diagonal and its two adjacent diagonals), which leads to a very efficient solution [36]. For example, the values of  $\mathbf{B}$  and  $\mathbf{c}$  for  $T = 3$  are given by

$$\mathbf{B} = \begin{pmatrix} \frac{1}{\sigma_{q_1}^2} + \frac{1}{\gamma_{q_2}^2} & -\frac{1}{\gamma_{q_2}^2} & 0 \\ -\frac{1}{\gamma_{q_2}^2} & \frac{1}{\sigma_{q_2}^2} + \frac{1}{\gamma_{q_2}^2} + \frac{1}{\gamma_{q_3}^2} & -\frac{1}{\gamma_{q_3}^2} \\ 0 & -\frac{1}{\gamma_{q_3}^2} & \frac{1}{\sigma_{q_3}^2} + \frac{1}{\gamma_{q_3}^2} \end{pmatrix} \quad (6.154)$$

$$\mathbf{c} = \left( \frac{\mu_{q_1}}{\sigma_{q_1}^2} - \frac{\delta_{q_2}}{\gamma_{q_2}^2}, \frac{\mu_{q_2}}{\sigma_{q_2}^2} + \frac{\delta_{q_2}}{\gamma_{q_2}^2} - \frac{\delta_{q_3}}{\gamma_{q_3}^2}, \frac{\mu_{q_3}}{\sigma_{q_3}^2} + \frac{\delta_{q_3}}{\gamma_{q_3}^2} \right)' \quad (6.155)$$

where just one dimension is represented, and the process is repeated for all dimensions with a computational complexity of  $O(TM)$ .



**Figure 6.30** Raw formants (ragged gray line) and smoothed formants (dashed line).

The maximum likelihood sequence  $\hat{\mathbf{x}}_i$  is close to the targets  $\mu_i$  while keeping the slopes close to  $\delta_i$  for a given state  $i$ , thus estimating a continuous function. Because of the delta coefficients, the solution depends on all the parameters of all states and not just the current state. This procedure can be performed for the formants as well as the bandwidths.



The parameters  $\mu_i$ ,  $\Sigma_i$ ,  $\delta_i$ , and  $\Gamma_i$  can be re-estimated using the EM algorithm described in Chapter 8. In [1] it is reported that two or three iterations are sufficient for speaker-dependent data.

The formant track obtained through this method can be rough, and it may be desired to smooth it. Smoothing without knowledge about the speech signal would result in either blurring the sharp transitions that occur in natural speech, or maintaining ragged formant tracks where the underlying physical phenomena vary slowly with time. Ideally we would like a larger adjustment to the raw formant when the error in the estimate is large relative to the variance of the corresponding state within a phoneme. This can be done by modeling the formant measurement error as a Gaussian distribution. Figure 6.29 shows an utterance from a male speaker with the smoothed formant tracks, and Figure 6.30 compares the raw and smoothed formants. When no real formant is visible from the spectrogram, the algorithm tends to assign a large bandwidth (not shown in the figure).

## 6.7. THE ROLE OF PITCH

Pitch determination is very important for many speech processing algorithms. The concatenative speech synthesis methods of Chapter 16 require pitch tracking on the desired speech segments if prosody modification is to be done. Chinese speech recognition systems use pitch tracking for tone recognition, which is important in disambiguating the myriad of homophones. Pitch is also crucial for prosodic variation in text-to-speech systems (see Chapter 15) and spoken language systems (see Chapter 17). While in the previous sections we have dealt with features representing the filter, pitch represents the source of the model illustrated in Figure 6.1.

Pitch determination algorithms also use short-term analysis techniques, which means that for every frame  $\mathbf{x}_m$  we get a score  $f(T | \mathbf{x}_m)$  that is a function of the candidate pitch periods  $T$ . These algorithms determine the optimal pitch by maximizing

$$T_m = \arg \max_T f(T | \mathbf{x}_m) \quad (6.156)$$

We describe several such functions computed through the autocorrelation method and the normalized cross-correlation method, as well as the signal conditioning that is often performed. Other approaches based on cepstrum [28] have also been used successfully. a good summary of techniques used for pitch tracking is provided by [17, 45].

Pitch determination using Eq. (6.156) is error prone, and a smoothing stage is often done. This smoothing, described in Section 6.7.4, takes into consideration that the pitch does not change quickly over time.

### 6.7.1. Autocorrelation Method

A commonly used method to estimate pitch is based on detecting the highest value of the autocorrelation function in the region of interest. This region must exclude  $m = 0$ , as that is

the absolute maximum of the autocorrelation function [37]. As discussed in Chapter 5, the statistical autocorrelation of a sinusoidal random process

$$\mathbf{x}[n] = \cos(\omega_0 n + \varphi) \quad (6.157)$$

is given by

$$R[m] = E\{\mathbf{x}^*[n]\mathbf{x}[n+m]\} = \frac{1}{2} \cos(\omega_0 m) \quad (6.158)$$

which has maxima for  $m = lT_0$ , the pitch period and its harmonics, so that we can find the pitch period by computing the highest value of the autocorrelation. Similarly, it can be shown that any WSS periodic process  $\mathbf{x}[n]$  with period  $T_0$  also has an autocorrelation  $R[m]$  which exhibits its maxima at  $m = lT_0$ .

In practice, we need to obtain an estimate  $\hat{R}[m]$  from knowledge of only  $N$  samples. If we use a window  $w[n]$  of length  $N$  on  $\mathbf{x}[n]$  and assume it to be real, the empirical autocorrelation function is given by

$$\hat{R}[m] = \frac{1}{N} \sum_{n=0}^{N-1-|m|} w[n]\mathbf{x}[n]w[n+|m|]\mathbf{x}[n+|m|] \quad (6.159)$$

whose expected value can be shown to be

$$E\{\hat{R}[m]\} = R[m](w[m] * w[-m]) \quad (6.160)$$

where

$$w[m] * w[-m] = \sum_{n=0}^{N-|m|-1} w[n]w[n+|m|] \quad (6.161)$$

which, for the case of a rectangular window of length  $N$ , is given by

$$w[m] * w[-m] = \begin{cases} 1 - \frac{|m|}{N} & |m| < N \\ 0 & |m| \geq N \end{cases} \quad (6.162)$$

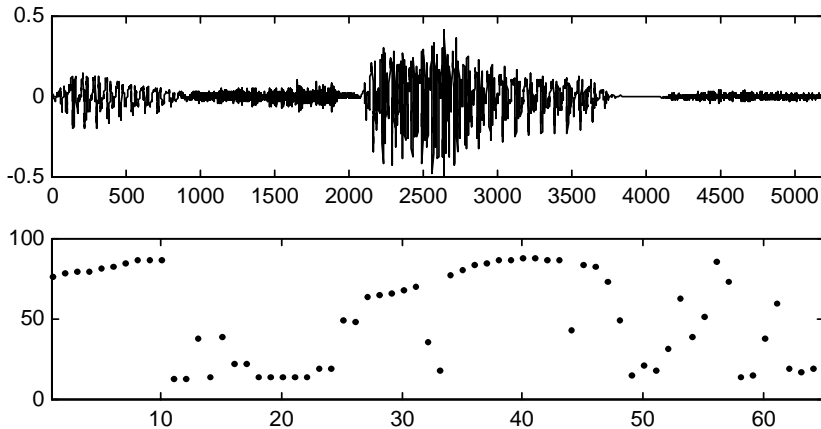
which means that  $\hat{R}[m]$  is a biased estimator of  $R[m]$ . So, if we compute the peaks based on Eq. (6.159), the estimate of the pitch will also be biased. Although the variance of the estimate is difficult to compute, it is easy to see that as  $m$  approaches  $N$ , fewer and fewer samples of  $x[n]$  are involved in the calculation, and thus the variance of the estimate is expected to increase. If we multiply Eq. (6.159) by  $N/(N-m)$ , the estimate will be unbiased but the variance will be larger.

Using the empirical autocorrelation in Eq. (6.159) for the random process in Eq. (6.157) results in an expected value of

$$E\{\hat{R}[m]\} = \left(1 - \frac{|m|}{N}\right) \frac{\cos(\omega_0 m)}{2}, \quad |m| < N \quad (6.163)$$

whose maximum coincides with the pitch period for  $m > m_0$ .

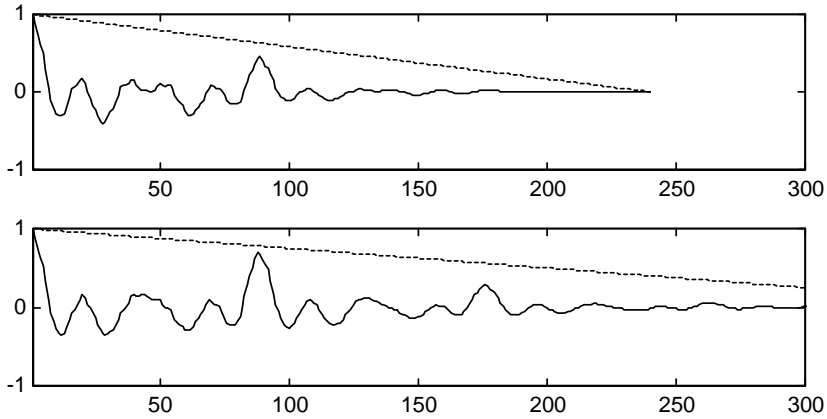
Since pitch periods can be as low as 40 Hz (for a very low-pitched male voice) or as high as 600 Hz (for a very high-pitched female or child's voice), the search for the maximum is conducted within a region. This F0 detection algorithm is illustrated in Figure 6.31 where the lag with highest autocorrelation is plotted for every frame. In order to see periodicity present in the autocorrelation, we need to use a window that contains at least two pitch periods, which, if we want to detect a 40Hz pitch, implies 50ms (see Figure 6.32). For window lengths so long, the assumption of stationarity starts to fail, because a pitch period at the beginning of the window can be significantly different than at the end of the window. One possible solution to this problem is to estimate the autocorrelation function with different window lengths for different lags  $m$ .



**Figure 6.31** Waveform and unsmoothed pitch track with the autocorrelation method. A frame shift of 10 ms, a Hamming window of 30 ms, and a sampling rate of 8kHz were used. Notice that two frames in the voiced region have an incorrect pitch. The pitch values in the unvoiced regions are essentially random.

The candidate pitch periods in Eq. (6.156) can be simply  $T_m = m$ ; i.e., the pitch period is any integer number of samples. For low values of  $T_m$ , the frequency resolution is lower than for high values. To maintain a relatively constant frequency resolution, we do not have to search all the pitch periods for large  $T_m$ . Alternatively, if the sampling frequency is not high, we may need to use fractional pitch periods (often done in the speech coding algorithms of Chapter 7)

The autocorrelation function can be efficiently computed by taking a signal, windowing it, and taking an FFT and then the square of the magnitude.



**Figure 6.32** Autocorrelation function for frame 40 in Figure 6.31. The maximum occurs at 89 samples. A sampling frequency of 8 kHz and window shift of 10ms are used. The top figure is using a window length of 30 ms, whereas the bottom one is using 50 ms. Notice the quasi-periodicity in the autocorrelation function.

## 6.7.2. Normalized Cross-Correlation Method

A method that is free from these border problems and has been gaining in popularity is based on the *normalized cross-correlation* [2]

$$\alpha_t(T) = \cos(\theta) = \frac{\langle \mathbf{x}_t, \mathbf{x}_{t-T} \rangle}{|\mathbf{x}_t| |\mathbf{x}_{t-T}|} \quad (6.164)$$

where  $\mathbf{x}_t = \{x[t - N/2], x[t - N/2 + 1], \dots, x[t + N/2 - 1]\}$  is a vector of  $N$  samples centered at time  $t$ , and  $\langle \mathbf{x}_t, \mathbf{x}_{t-T} \rangle$  is the inner product between the two vectors defined as

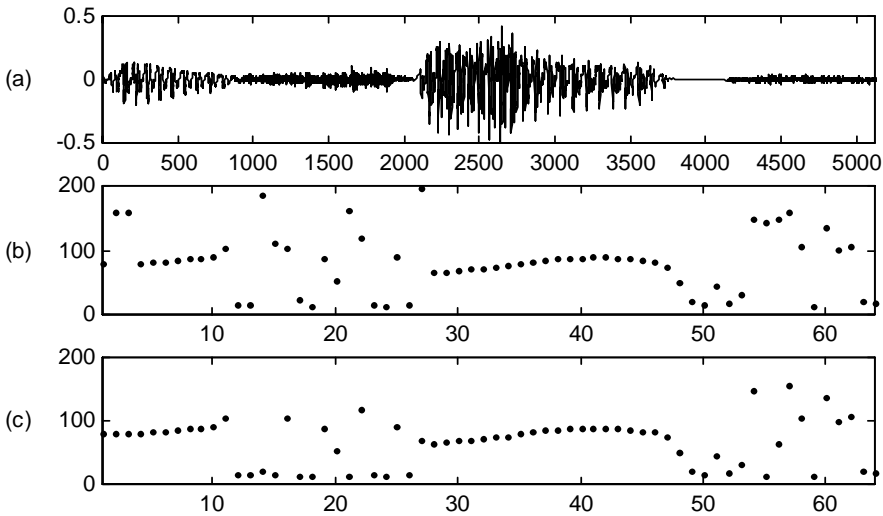
$$\langle \mathbf{x}_n, \mathbf{y}_l \rangle = \sum_{m=-N/2}^{N/2-1} x[n+m]y[l+m] \quad (6.165)$$

so that, using Eq. (6.165), the normalized cross-correlation can be expressed as

$$\alpha_t(T) = \frac{\sum_{n=-N/2}^{N/2-1} x[t+n]x[t+n-T]}{\sqrt{\sum_{n=-N/2}^{N/2-1} x^2[t+n] \sum_{m=-N/2}^{N/2-1} x^2[t+m+T]}} \quad (6.166)$$

where we see that the numerator in Eq. (6.166) is very similar to the autocorrelation in Section 6.7.1, but where  $N$  terms are used in the addition for all values of  $T$ .

The maximum of the normalized cross-correlation method is shown in Figure 6.33 (b). Unlike the autocorrelation method, the estimate of the normalized cross-correlation is not biased by the term  $(1 - m/N)$ . For perfectly periodic signals, this results in identical values of the normalized cross-correlation function for  $kT$ . This can result in pitch halving, where  $2T$  can be chosen as the pitch period, which happens in Figure 6.33 (b) at the beginning of the utterance. Using a decaying bias  $(1 - m/M)$  with  $M \gg N$ , can be useful in reducing pitch halving, as we see in Figure 6.33 (c).



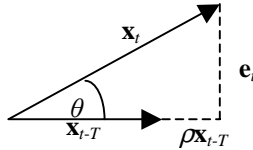
**Figure 6.33** (a) Waveform and (b) (c) unsmoothed pitch tracks with the normalized cross-correlation method. A frame shift of 10 ms, window length of 10 ms, and sampling rate of 8 kHz were used. (b) is the standard normalized cross-correlation method, whereas (c) has a decaying term. If we compare it to the autocorrelation method of Figure 6.31, the middle voiced region is correctly identified in both (b) and (c), but two frames at the beginning of (b) that have pitch halving are eliminated with the decaying term. Again, the pitch values in the unvoiced regions are essentially random.

Because the number of samples involved in the calculation is constant, this estimate is unbiased and has lower variance than that of the autocorrelation. Unlike the autocorrelation method, the window length could be lower than the pitch period, so that the assumption of stationarity is more accurate and it has more time resolution. While pitch trackers based on the normalized cross-correlation typically perform better than those based on the autocorrelation, they also require more computation, since all the autocorrelation lags can be efficiently computed through 2 FFTs and  $N$  multiplies and adds (see Section 5.3.4).

Let's gain some insight about the normalized cross-correlation. If  $x[n]$  is periodic with period  $T$ , then we can predict it from a vector  $T$  samples in the past as:

$$\mathbf{x}_t = \rho \mathbf{x}_{t-T} + \mathbf{e}_t \quad (6.167)$$

where  $\rho$  is the prediction gain. The normalized cross-correlation measures the angle between the two vectors, as can be seen in Figure 6.34, and since it is a cosine, it has the property that  $-1 \leq \alpha_n(P) \leq 1$ .



**Figure 6.34** The prediction of  $\mathbf{x}_t$  with  $\mathbf{x}_{t-T}$  results in an error  $\mathbf{e}_t$ .

If we choose the value of the prediction gain  $\rho$  so as to minimize the prediction error

$$|\mathbf{e}_t|^2 = |\mathbf{x}_t|^2 - |\mathbf{x}_t|^2 \cos^2(\theta) = |\mathbf{x}_t|^2 - |\mathbf{x}_t|^2 \alpha_t^2(T) \quad (6.168)$$

and assume  $\mathbf{e}_t$  is a zero-mean Gaussian random vector with a standard deviation  $\sigma_{|\mathbf{x}_t|}$ , then

$$\ln f(\mathbf{x}_t | T) = K + \frac{\alpha_t^2(T)}{2\sigma^2} \quad (6.169)$$

so that the maximum likelihood estimate corresponds to finding the value  $T$  with highest normalized cross-correlation. Using Eq. (6.166), it is possible that  $\alpha_t(T) < 0$ . In this case, there is negative correlation between  $\mathbf{x}_t$  and  $\mathbf{x}_{t-T}$ , and it is unlikely that  $T$  is a good choice for pitch. Thus, we need to force  $\rho > 0$ , so that Eq. (6.169) is converted into

$$\ln f(\mathbf{x}_t | T) = K + \frac{(\max(0, \alpha_t(T)))^2}{2\sigma^2} \quad (6.170)$$

The normalized cross-correlation of Eq. (6.164) predicts the current frame with a frame that occurs  $T$  samples before. Voiced speech may exhibit low correlation with a previous frame at a spectral discontinuity, such as those appearing at stops. To account for this, an enhancement can be done to consider not only the *backward* normalized cross-correlation, but also the *forward* normalized cross-correlation, by looking at a frame that occurs  $T$  samples ahead of the current frame, and taking the highest of both.

$$\ln f(\mathbf{x}_t | T) = K + \frac{(\max(0, \alpha_t(T), \alpha_t(-T)))^2}{2\sigma^2} \quad (6.171)$$

### 6.7.3. Signal Conditioning

Noise in the signal tends to make pitch estimation less accurate. To reduce this effect, signal conditioning or pre-processing has been proposed prior to pitch estimation [44]. Typically this involves bandpass filtering to remove frequencies above 1 or 2 kHz, and below 100 Hz or so. High frequencies do not have much voicing information and have significant noise energy, whereas low frequencies can have 50/60 Hz interference from power lines or non-linearities from some A/D subsystems that can also mislead a pitch estimation algorithm.

In addition to the noise in the very low frequencies and aspiration at high bands, the stationarity assumption is not so valid at high frequencies. Even a slowly changing pitch, say, nominal 100 Hz increasing 5 Hz in 10 ms, results in a fast-changing harmonic: the 30<sup>th</sup> harmonic at 3000 Hz changes 150 Hz in 10 ms. The corresponding short-time spectrum no longer shows peaks at those frequencies.

Because of this, it is advantageous to filter out such frequencies prior to the computation of the autocorrelation or normalized cross-correlation. If an FFT is used to compute the autocorrelation, this filter is easily done by setting to 0 the undesired frequency bins.

### 6.7.4. Pitch Tracking

Pitch tracking using the above methods typically fails in several cases:

- *Sub-harmonic errors.* If a signal is periodic with period  $T$ , it is also periodic with period  $2T$ ,  $3T$ , etc. Thus, we expect the scores to be also high for the multiples of  $T$ , which can mislead the algorithm. Because the signal is never perfectly stationary, those multiples, or sub-harmonics, tend to have slightly lower scores than the fundamental. If the pitch is identified as  $2T$ , pitch halving is said to occur.
- *Harmonic errors.* If harmonic  $M$  dominates the signal's total energy, the score at pitch period  $T/M$  will be large. This can happen if the harmonic falls in a formant frequency that boosts its amplitude considerably compared to that of the other harmonics. If the pitch is identified as  $T/2$ , pitch doubling is said to occur.
- *Noisy conditions.* When the SNR is low, pitch estimates are quite unreliable for most methods.
- *Vocal fry.* While pitch is generally continuous, for some speakers it can suddenly change and even halve, particularly at the end of an unstressed voiced region. The pitch here is really not well defined and imposing smoothness constraints can hurt the system.
- *F0 jumps up or down by an octave occasionally.*
- *Breathy voiced speech* is difficult to distinguish from periodic background noise.
- *Narrow-band filtering* of unvoiced excitations by certain vocal tract configurations can lead to signals that appear periodic.

For these reasons, pitch trackers do not determine the pitch value at frame  $m$  based exclusively on the signal at that frame. For a frame where there are several pitch candidates with similar scores, the fact that pitch does not change abruptly with time is beneficial in disambiguation, because possibly the following frame has a clearer pitch candidate, which can help.

To integrate the normalized cross-correlation into a probabilistic framework, you can combine tracking with the use of a priori information [10]. Let's define  $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{M-1}\}$  as a sequence of input vectors for  $M$  consecutive frames centered at equally spaced time instants, say 10 ms. Furthermore, if we assume that the  $\mathbf{x}_i$  are independent of each other, the joint distribution takes on the form:

$$f(\mathbf{X} | \mathbf{T}) = \prod_{i=0}^{M-1} f(\mathbf{x}_i | T_i) \quad (6.172)$$

where  $\mathbf{T} = \{T_0, T_1, \dots, T_{M-1}\}$  is the pitch track for the input. The *maximum a posteriori* (MAP) estimate of the pitch track is:

$$\mathbf{T}_{MAP} = \max_{\mathbf{T}} f(\mathbf{T} | \mathbf{X}) = \max_{\mathbf{T}} \frac{f(\mathbf{T})f(\mathbf{X} | \mathbf{T})}{f(\mathbf{X})} = \max_{\mathbf{T}} f(\mathbf{T})f(\mathbf{X} | \mathbf{T}) \quad (6.173)$$

according to Bayes' rule, with the term  $f(\mathbf{X} | \mathbf{T})$  being given by Eq. (6.172) and  $f(\mathbf{x}_i | T_i)$  by Eq. (6.169), for example.

The function  $f(\mathbf{T})$  constitutes the *a priori* statistics for the pitch and can help disambiguate the pitch, by avoiding pitch doubling or halving given knowledge of the speaker's average pitch, and by avoiding rapid transitions given a model of how pitch changes over time. One possible approximation is given by assuming that the a priori probability of the pitch period at frame  $i$  depends only on the pitch period for the previous frame:

$$f(\mathbf{T}) = f(T_0, T_1, \dots, T_{M-1}) = f(T_{M-1} | T_{M-2})f(T_{M-2} | T_{M-3}) \cdots f(T_1 | T_0)f(T_0) \quad (6.174)$$

One possible choice for  $f(T_i | T_{i-1})$  is to decompose it into a component that depends on  $T_i$  and another that depends on the difference  $(T_i - T_{i-1})$ . If we approximate both as Gaussian densities, we obtain

$$\ln f(T_i | T_{i-1}) = K' - \frac{(T_i - \mu)^2}{2\beta^2} - \frac{(T_i - T_{i-1} - \delta)^2}{2\gamma^2} \quad (6.175)$$

so that when Eqs. (6.170) and (6.175) are combined, the log-probability of transitioning to  $T_i$  at time  $t$  from pitch  $T_j$  at time  $t - 1$  is given by

$$S_t(T_i, T_j) = \frac{(\max(0, \alpha_t(T_i)))^2}{2\sigma^2} - \frac{(T_i - \mu)^2}{2\beta^2} - \frac{(T_i - T_j - \delta)^2}{2\gamma^2} \quad (6.176)$$



so that the log-likelihood in Eq. (6.173) can be expressed as

$$\ln f(\mathbf{T})f(\mathbf{X}|\mathbf{T}) = (\max(0, \alpha_0(T_0)))^2 + \max_{i_t} \sum_{t=1}^{M-1} S_t(T_{i_t}, T_{i_{t-1}}) \quad (6.177)$$

which can be maximized through dynamic programming. For a region where pitch is not supposed to change,  $\delta = 0$ , the term  $(T_i - T_j)^2$  in Eq. (6.176) acts as a penalty that keeps the pitch track from jumping around. A mixture of Gaussians can be used instead to model different rates of pitch change, as in the case of Mandarin Chinese with four tones characterized by different slopes. The term  $(T_i - \mu)^2$  attempts to get the pitch close to its expected value to avoid pitch doubling or halving, with the average  $\mu$  being different for male and female speakers. Pruning can be done during the search without loss of accuracy (see Chapter 12).

Pitch trackers also have to determine whether a region of speech is voiced or unvoiced. A good approach is to build a statistical classifier with techniques described in Chapter 8 based on energy and the normalized cross-correlation described above. Such classifiers, *i.e.*, an HMM, penalize jumps between voiced and unvoiced frames to avoid voiced regions having isolated unvoiced frame inside and vice versa. A threshold can be used on the a posteriori probability to distinguish voiced from unvoiced frames.

## 6.8. HISTORICAL PERSPECTIVE AND FUTURE READING

In 1978, Lawrence R. Rabiner and Ronald W. Schafer [38] wrote a book summarizing the work to date on digital processing of speech, which remains a good source for the reader interested in further reading in the field. The book by Deller, Hansen and Proakis [9] includes more recent work and is also an excellent reference. O'Shaughnessy [33] also has a thorough description of the subject. Malvar [25] covers filterbanks and lapped transforms extensively.

The extensive wartime interest in sound spectrography led Koenig and his colleagues at Bell Laboratories [22] in 1946 to the invaluable development of a tool that has been used for speech analysis since then: the spectrogram. Potter et al. [35] showed the usefulness of the analog spectrogram in analyzing speech. The spectrogram facilitated research in the field and led Peterson and Barney [34] to publish in 1952 a detailed study of formant values of different vowels. The development of computers and the FFT led Oppenheim, in 1970 [30], to develop digital spectrograms, which imitated the analog counterparts.

The MIT Acoustics Lab started work in speech in 1948 with Leo R. Beranek, who in 1954 published the seminal book *Acoustics*, where he studied sound propagation in tubes. In 1950, Kenneth N. Stevens joined the lab and started work on speech perception. Gunnar Fant visited the lab at that time and as a result started a strong speech production effort at KTH in Sweden.

The 1960s marked the birth of digital speech processing. Two books, Gunnar Fant's *Acoustical Theory of Speech Production* [13] in 1960 and James Flanagan's *Speech Analysis: Synthesis and Perception* [14] in 1965, had a great impact and sparked interest in the

field. The advent of the digital computer prompted Kelly and Gertsman to create in 1961 the first digital speech synthesizer [21]. Short-time Fourier analysis, cepstrum, LPC analysis, pitch and formant tracking, and digital filterbanks were the fruit of that decade.

Short-time frequency analysis was first proposed for analog signals by Fano [11] in 1950 and later by Schroeder and Atal [42].

The mathematical foundation behind linear predictive coding dates to the autoregressive models of George Udny Yule (1927) and Gilbert Walker (1931), which led to the well-known Yule-Walker equations. These equations resulted in a Toeplitz matrix, named after Otto Toeplitz (1881 - 1940) who studied it extensively. N. Levinson suggested in 1947 an efficient algorithm to invert such a matrix, which J. Durbin refined in 1960 and is now known as the Levinson-Durbin recursion. The well-known LPC analysis consisted of the application of the above results to speech signals, as developed by Bishnu Atal [4], J. Burg [7], Fumitada Itakura and S. Saito [19] in 1968, Markel [27] and John Makhoul [24] in 1973.

The cepstrum was first proposed in 1964 by Bogert, Healy and John Tukey [6] and further studied by Alan V. Oppenheim [29] in 1965. The popular mel-frequency cepstrum was proposed by Davis and Mermelstein [8] in 1980, combining the advantages of cepstrum with knowledge of the non-linear perception of frequency by the human auditory system that had been studied by E. Zwicker [47] in 1961.

The study of digital filterbanks was first proposed by Schafer and Rabiner in 1971 for IIR and in 1975 for FIR filters.

Formant tracking was first investigated by Ken Stevens and James Flanagan in the late 1950s, with the foundations for most modern techniques being developed by Schafer and Rabiner [40], Itakura [20], and Markel [26]. Pitch tracking through digital processing was first studied by B. Gold [15] in 1962 and then improved by A. M. Noll [28], M. Schroeder [41], and M. Sondhi [44] in the late 1960s.

## REFERENCES

- [1] Acero, A., "Formant Analysis and Synthesis using Hidden Markov Models," *Eurospeech*, 1999, Budapest pp. 1047-1050.
- [2] Atal, B.S., *Automatic Speaker Recognition Based on Pitch Contours*, PhD Thesis 1968, Polytechnic Institute of Brooklyn, .
- [3] Atal, B.S. and L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," *Journal of the Acoustical Society of America*, 1971, **50**, pp. 637-655.
- [4] Atal, B.S. and M.R. Schroeder, "Predictive Coding of Speech Signals," *Report of the 6th Int. Congress on Acoustics*, 1968, Tokyo, Japan.
- [5] Berouti, M.G., D.G. Childers, and A. Paige, "Glottal Area versus Glottal Volume Velocity," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1977, Hartford, Conn pp. 33-36.
- [6] Bogert, B., M. Healy, and J. Tukey, "The Quefrequency Alanalysis of Time Series for Echoes," *Proc. Symp. on Time Series Analysis*, 1963, New York, J. Wiley pp. 209-243.

- [7] Burg, J., "Maximum Entropy Spectral Analysis," *Proc. of the 37th Meeting of the Society of Exploration Geophysicists*, 1967.
- [8] Davis, S. and P. Mermelstein, "Comparison of Parametric Representations for Monosyllable Word Recognition in Continuously Spoken Sentences," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1980, **28**(4), pp. 357-366.
- [9] Deller, J.R., J.H.L. Hansen, and J.G. Proakis, *Discrete-Time Processing of Speech Signals*, 2000, IEEE Press.
- [10] Droppo, J. and A. Acero, "Maximum a Posteriori Pitch Tracking," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia pp. 943-946.
- [11] Fano, R.M., "Short-time Autocorrelation Functions and Power Spectra," *Journal of the Acoustical Society of America*, 1950, **22**(Sep), pp. 546-550.
- [12] Fant, G., "On the Predictability of Formant Levels and Spectrum Envelopes from Formant Frequencies" in *For Roman Jakobson*, M. Halle, Editor 1956, The Hague, NL, pp. 109-120, Mouton & Co.
- [13] Fant, G., *Acoustic Theory of Speech Production*, 1970, The Hague, NL, Mouton.
- [14] Flanagan, J., *Speech Analysis Synthesis and Perception*, 1972, New York, Springer-Verlag.
- [15] Gold, B., "Computer Program for Pitch Extraction," *Journal of the Acoustical Society of America*, 1962, **34**(7), pp. 916-921.
- [16] Hermansky, H., "Perceptual Linear Predictive (PLP) Analysis of Speech," *Journal of the Acoustical Society of America*, 1990, **87**(4), pp. 1738-1752.
- [17] Hess, W., *Pitch Determination of Speech Signals*, 1983, New York, Springer-Verlag.
- [18] Itakura, F., "Line Spectrum Representation of Linear Predictive Coefficients," *Journal of the Acoustical Society of America*, 1975, **57**(4), pp. 535.
- [19] Itakura, F. and S. Saito, "Analysis Synthesis Telephony Based on the Maximum Likelihood Method," *Proc. 6th Int. Congress on Acoustics*, 1968, Tokyo, Japan.
- [20] Itakura, F. and S. Saito, "A Statistical Method for Estimation of Speech Spectral Density and Formant Frequencies," *Elec. and Comm. in Japan*, 1970, **53-A**(1), pp. 36-43.
- [21] Kelly, J.L. and L.J. Gerstman, "An Artificial Talker Driven From Phonetic Input," *Journal of Acoustical Society of America*, 1961, **33**, pp. 835.
- [22] Koenig, R., H.K. Dunn, and L.Y. Lacy, "The Sound Spectrograph," *Journal of the Acoustical Society of America*, 1946, **18**, pp. 19-49.
- [23] Krishnamurthy, A.K. and D.G. Childers, "Two Channel Speech Analysis," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1986, **34**, pp. 730-743.
- [24] Makhoul, J., "Spectral Analysis of Speech by Linear Prediction," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1973, **21**(3), pp. 140-148.
- [25] Malvar, H., *Signal Processing with Lapped Transforms*, 1992, Artech House.
- [26] Markel, J.D., "Digital Inverse Filtering - A New Tool for Formant Trajectory Estimation," *IEEE Trans. on Audio and Electroacoustics*, 1972, **AU-20**(June), pp. 129-137.

- [27] Markel, J.D. and A.H. Gray, "On Autocorrelation Equations as Applied to Speech Analysis," *IEEE Trans. on Audio and Electroacoustics*, 1973, **AU-21**(April), pp. 69-79.
- [28] Noll, A.M., "Cepstrum Pitch Determination," *Journal of the Acoustical Society of America*, 1967, **41**, pp. 293-309.
- [29] Oppenheim, A.V., *Superposition in a Class of Nonlinear Systems*, 1965, Research Lab. Of Electronics, MIT, Cambridge, Massachusetts.
- [30] Oppenheim, A.V., "Speech Spectrograms Using the Fast Fourier Transform," *IEEE Spectrum*, 1970, **7**(Aug), pp. 57-62.
- [31] Oppenheim, A.V. and D.H. Johnson, "Discrete Representation of Signals," *The Proc. of the IEEE*, 1972, **60**(June), pp. 681-691.
- [32] Oppenheim, A.V., R.W. Schafer, and T.G. Stockham, "Nonlinear Filtering of Multiplied and Convolved Signals," *Proc. of the IEEE*, 1968, **56**, pp. 1264-1291.
- [33] O'Shaughnessy, D., *Speech Communication -- Human and Machine*, 1987, Addison-Wesley.
- [34] Peterson, G.E. and H.L. Barney, "Control Methods Used in a Study of the Vowels," *Journal of the Acoustical Society of America*, 1952, **24**(2), pp. 175-184.
- [35] Potter, R.K., G.A. Kopp, and H.C. Green, *Visible Speech*, 1947, New York, D. Van Nostrand Co. Republished by Dover Publications, Inc. 1966.
- [36] Press, W.H., *et al.*, *Numerical Recipes in C*, 1988, New York, NY, Cambridge University Press.
- [37] Rabiner, L.R., "On the Use of Autocorrelation Analysis for Pitch Detection," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1977, **25**, pp. 24-33.
- [38] Rabiner, L.R. and R.W. Schafer, *Digital Processing of Speech Signals*, 1978, Englewood Cliffs, NJ, Prentice-Hall.
- [39] Rosenberg, A.E., "Effect of Glottal Pulse Shape on the Quality of Natural Vowels," *Journal of the Acoustical Society of America*, 1971, **49**, pp. 583-590.
- [40] Schafer, R.W. and L.R. Rabiner, "System for Automatic Formant Analysis of Voiced Speech," *Journal of the Acoustical Society of America*, 1970, **47**, pp. 634-678.
- [41] Schroeder, M., "Period Histogram and Product Spectrum: New Methods for Fundamental Frequency Measurement," *Journal of the Acoustical Society of America*, 1968, **43**(4), pp. 829-834.
- [42] Schroeder, M.R. and B.S. Atal, "Generalized Short-Time Power Spectra and Autocorrelation," *Journal of the Acoustical Society of America*, 1962, **34**(Nov), pp. 1679-1683.
- [43] Shikano, K., K.-F. Lee, and R. Reddy, "Speaker Adaptation through Vector Quantization," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1986, Tokyo, Japan pp. 2643-2646.
- [44] Sondhi, M.M., "New Methods for Pitch Extraction," *IEEE Trans. on Audio and Electroacoustics*, 1968, **16**(June), pp. 262-268.
- [45] Talkin, D., "A Robust Algorithm for Pitch Tracking" in *Speech Coding and Synthesis*, W.B. Kleijn and K.K. Paliwal, eds. 1995, Amsterdam, pp. 485-518, Elsevier.

- [46] Yegnanarayana, B. and R.N.J. Veldhuis, "Extraction of Vocal-Tract System Characteristics from Speech Signals," *IEEE Trans. on Speech and Audio Processing*, 1998, **6**(July), pp. 313-327.
- [47] Zwicker, E., "Subdivision of the Audible Frequency Range into Critical Bands," *Journal of the Acoustical Society of America*, 1961, **33**(Feb), pp. 248.

---

# CHAPTER 7

---

## Speech Coding

*T*ransmission of speech using data networks requires the speech signal to be digitally encoded. Voice over IP has become very popular because of the Internet, where bandwidth limitations make it necessary to compress the speech signal. Digital storage of audio signals, which can result in higher quality and smaller size than the analog counterpart, is commonplace in compact discs, digital video discs, and MP3 files. Many spoken language systems also use coded speech for efficient communication. For these reasons we devote a chapter to speech and audio coding techniques.

Rather than exhaustively cover all the existing speech and audio coding algorithms we uncover their underlying technology and enumerate some of the most popular standards. The coding technology discussed in this chapter has a strong link to both speech recognition and speech synthesis. For example, the speech synthesis algorithms described in Chapter 16 use many techniques described here.

## 7.1. SPEECH CODERS ATTRIBUTES

How do we compare different speech or audio coders? We can refer to a number of factors, such as signal bandwidth, bit rate, quality of reconstructed speech, noise robustness, computational complexity, delay, channel-error sensitivity and standards.

Speech signals can be bandlimited to 10 kHz without significantly affecting the hearer's perception. The telephone network limits the bandwidth of speech signals to between 300 and 3400 Hz, which gives *telephone speech* a lower quality. Telephone speech is typically sampled at 8 kHz. The term *wideband speech* is used for a bandwidth of 50–7000 Hz and a sampling rate of 16 kHz. Finally, *audio coding* is used in dealing with high-fidelity audio signals, in which case the signal is sampled at 44.1 kHz.

Reduction in bit rate is the primary purpose of speech coding. The previous bit stream can be compressed to a lower rate by removing redundancy in the signal, resulting in savings in storage and transmission bandwidth. If only redundancy is removed, the original signal can be recovered exactly (*lossless* compression). In *lossy* compression, the signal cannot be recovered exactly, though hopefully it will sound similar to the original.

Depending on system and design constraints, fixed-rate or variable-rate speech coders can be used. Variable-rate coders are used for non-real time applications, such as voice storage (silence can be coded with fewer bits than fricatives, which in turn use fewer bits than vowels), or for packet voice transmissions, such as CDMA cellular for better channel utilization. Transmission of coded speech through a noisy channel may require devoting more bits to channel coding and fewer to source coding. For most real-time communication systems, a maximum bit rate is specified.

The quality of the reconstructed speech signal is a fundamental attribute of a speech coder. Bit rate and quality are intimately related: the lower the bit rate, the lower the quality. While the bit rate is inherently a number, it is difficult to quantify the quality. The most widely used measure of quality is the *Mean Opinion Score* (MOS) [25], which is the result of averaging opinion scores for a set of between 20 and 60 untrained subjects. Each listener characterizes each set of utterances with a score on a scale from 1 (unacceptable quality) to 5 (excellent quality), as shown in Table 7.1. An MOS of 4.0 or higher defines *good* or *toll* quality, where the reconstructed speech signal is generally indistinguishable from the original signal. An MOS between 3.5 and 4.0 defines *communication* quality, which is sufficient for telephone communications. We show in Section 7.2.1 that if each sample is quantized with 16 bits, the resulting signal has *toll* quality (essentially indistinguishable from the unquantized signal). See Chapter 16 for more details on perceptual quality measurements.

**Table 7.1** Mean Opinion Score (MOS) is a numeric value computed as an average for a number of subjects, where each number maps to the above subjective quality.

Excellent	Good	Fair	Poor	Bad
5	4	3	2	1

Another measure of quality is the *signal-to-noise ratio* (SNR), defined as the ratio between the signal's energy and the noise's energy in terms of dB:

$$SNR = \frac{\sigma_x^2}{\sigma_e^2} = \frac{E\{x^2[n]\}}{E\{e^2[n]\}} \quad (7.1)$$

The MOS rating of a codec on noise-free speech is often higher than its MOS rating for noisy speech. This is generally caused by specific assumptions in the speech coder that tend to be violated when a significant amount of noise is present in the signal. This phenomenon is more accentuated for lower-bit-rate coders that need to make more assumptions.

The computational complexity and memory requirements of a speech coder determine the cost and power consumption of the hardware on which it is implemented. In most cases, real-time operation is required at least for the decoder. Speech coders can be implemented in *inexpensive Digital Signal Processors* (DSP) that form part of many consumer devices, such as answering machines and DVD players, for which storage tends to be relatively more expensive than processing power. DSPs are also used in cellular phones because bit rates are limited.

All speech coders have some delay, which, if excessive, can affect the dynamics of a two-way communication. For instance, delays over 150 ms can be unacceptable for highly interactive conversations. Coder delay is the sum of different types of delay. The first is the *algorithmic delay* arising because speech coders usually operate on a block of samples, called a *frame*, which needs to be accumulated before processing can begin. Often the speech coder requires some additional *look-ahead* beyond the frame to be encoded. The *computational delay* is the time that the speech coder takes to process the frame. For real-time operation, the computational delay has to be smaller than the algorithmic delay. A block of bits is generally assembled by the encoder prior to transmission, possibly to add error-correction properties to the bit stream, which cause *multiplexing delay*. Finally, there is the *transmission delay*, due to the time it takes for the frame to traverse the channel. The decoder will incur a *decoder delay* to reconstruct the signal. In practice, the total delay of many speech coders is at least three frames.

If the coded speech needs to be transmitted over a channel, we need to consider possible channel errors, and our speech decoder should be insensitive to at least some of them. There are two types of errors: random errors and burst errors, and they could be handled differently. One possibility to increase the robustness against such errors is to use channel coding techniques, such as those proposed in Chapter 3. Joint source and channel coding allows us to find the right combination of bits to devote to speech coding with the right amount devoted to channel coding, adjusting this ratio adaptively depending on the channel. Since channel coding will only reduce the number of errors, and not eliminate them, graceful degradation of speech quality under channel errors is typically a design factor for speech coders. When the channel is the Internet, complete frames may be missing because they have not arrived in time. Therefore, we need techniques that degrade gracefully with missing frames.



## 7.2. SCALAR WAVEFORM CODERS

In this section we describe several waveform coding techniques, such as linear PCM,  $\mu$ -law, and A-law PCM, APCM, DPCM, DM, and ADPCM, that quantize each sample using scalar quantization. These techniques attempt to approximate the waveform, and, if a large enough bit rate is available, will get arbitrarily close to it.

### 7.2.1. Linear Pulse Code Modulation (PCM)

Analog-to-digital converters perform both sampling and quantization simultaneously. To better understand how this process affects the signal it's better to study them separately. We analyzed the effects of sampling in Chapter 5, so now we analyze the effects of quantization, which encodes each sample with a fixed number of bits. With  $B$  bits, it is possible to represent  $2^B$  separate quantization levels. The output of the quantizer  $\hat{x}[n]$  is given by

$$\hat{x}[n] = Q\{x[n]\} \quad (7.2)$$

Linear *Pulse Code Modulation* (PCM) is based on the assumption that the input discrete signal  $x[n]$  is bounded

$$|x[n]| \leq X_{\max} \quad (7.3)$$

and that we use *uniform quantization* with quantization step size  $\Delta$  which is constant for all levels  $x_i$

$$x_i - x_{i-1} = \Delta \quad (7.4)$$

The input/output characteristics are shown by Figure 7.1 for the case of a 3-bit uniform quantizer. The so-called *mid-riser* quantizer has the same number of positive and negative levels, whereas the *mid-tread* quantizer has one more negative than positive levels. The code  $c[n]$  is expressed in two's complement representation, which for Figure 7.1 varies between  $-4$  and  $+3$ . For the mid-riser quantizer the output  $\hat{x}[n]$  can be obtained from the code  $c[n]$  through

$$\hat{x}[n] = \text{sign}(c[n]) \frac{\Delta}{2} + c[n]\Delta \quad (7.5)$$

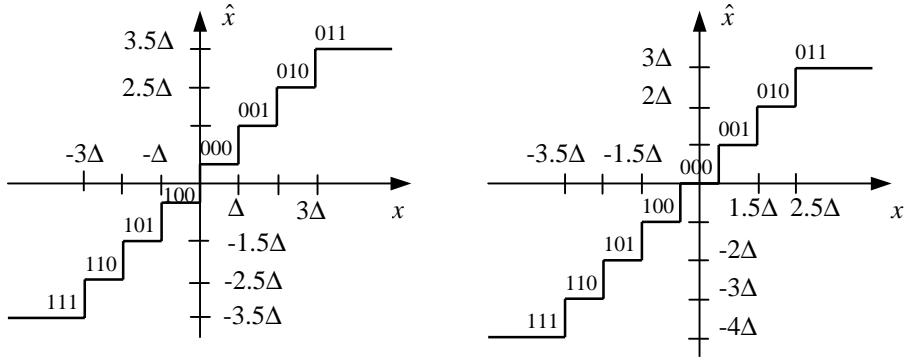
and for the mid-tread quantizer

$$\hat{x}[n] = c[n]\Delta \quad (7.6)$$

which is often used in computer systems that use two's complement representation.

There are two independent parameters for a uniform quantizer: the number of levels  $N = 2^B$ , and the step size  $\Delta$ . Assuming Eq. (7.3), we have the relationship

$$2X_{\max} = \Delta 2^B \quad (7.7)$$



**Figure 7.1** Three-bit uniform quantization characteristics: (a) mid-riser, (b) mid-tread.

In quantization, it is useful to express the relationship between the unquantized sample  $x[n]$  and the quantized sample  $\hat{x}[n]$  as

$$\hat{x}[n] = x[n] + e[n] \quad (7.8)$$

with  $e[n]$  being the quantization noise. If we choose  $\Delta$  and  $B$  to satisfy Eq. (7.7), then

$$-\frac{\Delta}{2} \leq e[n] \leq \frac{\Delta}{2} \quad (7.9)$$

While there is obviously a deterministic relationship between  $e[n]$  and  $x[n]$ , it is convenient to assume a probabilistic model for the quantization noise:

1.  $e[n]$  is white:  $E\{e[n]e[n+m]\} = \sigma_e^2 \delta[m]$
2.  $e[n]$  and  $x[n]$  are uncorrelated:  $E\{x[n]e[n+m]\} = 0$
3.  $e[n]$  is uniformly distributed in the interval  $(-\Delta/2, \Delta/2)$

These assumptions are unrealistic for some signals, except in the case of speech signals, which rapidly fluctuate between different quantization levels. The assumptions are reasonable if the step size  $\Delta$  is a small enough, or alternatively the number of levels is large enough (say more than  $2^6$ ).

The variance of such uniform distribution (see Chapter 3) is

$$\sigma_e^2 = \frac{\Delta^2}{12} = \frac{X_{\max}^2}{3 \times 2^{2B}} \quad (7.10)$$

after using Eq. (7.7). The SNR is given by

$$\text{SNR}(dB) = 10 \log_{10} \left( \frac{\sigma_x^2}{\sigma_e^2} \right) = (20 \log_{10} 2)B + 10 \log_{10} 3 - 20 \log_{10} \left( \frac{X_{\max}}{\sigma_x} \right) \quad (7.11)$$

which implies that each bit contributes to 6 dB of SNR, since  $20 \log_{10} 2 \cong 6$ .

Speech samples can be approximately described as following a *Laplacian* distribution [40]

$$p(x) = \frac{1}{\sqrt{2}\sigma_x} e^{-\frac{\sqrt{2}|x|}{\sigma_x}} \quad (7.12)$$

and the probability of  $x$  falling outside the range  $(-4\sigma_x^2, 4\sigma_x^2)$  is 0.35%. Thus, using  $X_{\max} = 4\sigma_x$ ,  $B = 7$  bits in Eq. (7.11) results in an SNR of 35 dB, which would be acceptable in a communications system. Unfortunately, signal energy can vary over 40 dB, due to variability from speaker to speaker as well as variability in transmission channels. Thus, in practice, it is generally accepted that 11 bits are needed to achieve an SNR of 35dB while keeping the clipping to a minimum.

Digital audio stored in computers (Windows WAV, Apple AIF, Sun AU, and SND formats among others) use 16-bit linear PCM as their main format. The *Compact Disc-Digital Audio* (CD-DA or simply CD) also uses 16-bit linear PCM. Invented in the late 1960s by James T. Russell, it was launched commercially in 1982 and has become one of the most successful examples of consumer electronics technology: there were about 700 million audio CD players in 1997. A CD can store up to 74 minutes of music, so the total amount of digital data that must be stored on a CD is  $44,100 \text{ samples}/(\text{channel} \cdot \text{second}) * 2 \text{ bytes/sample} * 2 \text{ channels} * 60 \text{ seconds/minute} * 74 \text{ minutes} = 783,216,000 \text{ bytes}$ . This 747 MB are stored in a disk only 12 centimeters in diameter and 1.2 mm thick. CD-ROMs can record only 650 MB of computer data because they use the remaining bits for error correction.

## 7.2.2. $\mu$ -law and A-law PCM

Human perception is affected by SNR, because adding noise to a signal is not as noticeable if the signal energy is large enough. Ideally, we want SNR to be constant for all quantization levels, which requires the step size to be proportional to the signal value. This can be done by using a logarithmic *comparer*<sup>1</sup>

$$y[n] = \ln|x[n]| \quad (7.13)$$

followed by a uniform quantizer on  $y[n]$  so that

$$\hat{y}[n] = y[n] + \varepsilon[n] \quad (7.14)$$

and, thus,

$$\hat{x}[n] = \exp\{\hat{y}[n]\} \text{sign}\{x[n]\} = x[n] \exp\{\varepsilon[n]\} \quad (7.15)$$

after using Eq. (7.13) and (7.14). If  $\varepsilon[n]$  is small, then Eq. (7.15) can be expressed as

$$\hat{x}[n] \cong x[n](1 + \varepsilon[n]) = x[n] + x[n]\varepsilon[n] \quad (7.16)$$

<sup>1</sup> A comparer is a nonlinear function that compands one part of the  $x$ -axis.

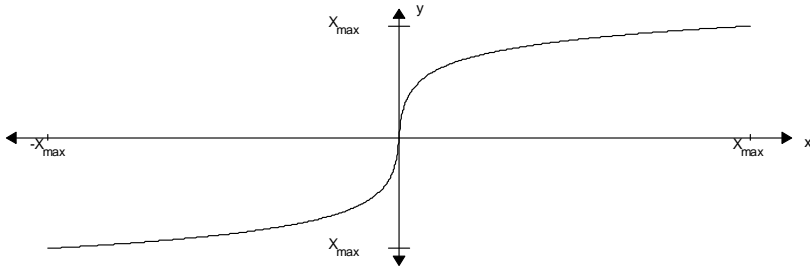
and, thus, the  $SNR = 1/\sigma_\varepsilon^2$  is constant for all levels. This type of quantization is not practical, because an infinite number of quantization steps would be required. An approximation is the so-called  $\mu$ -law [51]:

$$y[n] = X_{\max} \frac{\log \left[ 1 + \mu \frac{|x[n]|}{X_{\max}} \right]}{\log[1 + \mu]} \text{sign}\{x[n]\} \quad (7.17)$$

which is approximately logarithmic for large values of  $x[n]$  and approximately linear for small values of  $x[n]$ . A related compander called A-law is also used

$$y[n] = X_{\max} \frac{1 + \log \left[ \frac{A|x[n]|}{X_{\max}} \right]}{1 + \log A} \text{sign}\{x[n]\} \quad (7.18)$$

which has greater resolution than  $\mu$ -law for small sample values, but a range equivalent to 12 bits. In practice, they both offer similar quality. The  $\mu$ -law curve can be seen in Figure 7.2.



**Figure 7.2** Nonlinearity used in the  $\mu$ -law compression.

In 1972 the ITU-T<sup>2</sup> recommendation G.711 standardized telephone speech coding at 64 kbps for digital transmission of speech through telephone networks. It uses 8 bits per sample and an 8-kHz sampling rate with either  $\mu$ -law or A-law. In North America and Japan,  $\mu$ -law with  $\mu = 255$  is used, whereas, in the rest of the world, A-law with  $A = 87.56$  is used. Both compression characteristics are very similar and result in an approximate SNR of 35 dB. Without the logarithmic compressor, a uniform quantizer requires approximately 12 bits per sample to achieve the same level of quality. All the speech coders for telephone speech described in this chapter use G.711 as a baseline reference, whose quality is considered *toll*,

<sup>2</sup> The International Telecommunication Union (ITU) is a part of the United Nations Economic, Scientific and Cultural Organization (UNESCO). ITU-T is the organization within ITU responsible for setting global telecommunication standards. Within ITU-T, Study Group 15 (SG15) is responsible for formulating speech coding standards. Prior to 1993, telecommunication standards were set by the *Comité Consultatif International Téléphonique et Télégraphique* (CCITT), which was reorganized into the ITU-T that year.

and an MOS of about 4.0. G.711 is used by most digital central office switches, so that when you make a telephone call using your plain old telephone service (POTS), your call is encoded with G.711. G.711 has an MOS of about 4.3.

### 7.2.3. Adaptive PCM

When quantizing speech signals we confront a dilemma. On the one hand, we want the quantization step size to be large enough to accommodate the maximum peak-to-peak range of the signal and avoid clipping. On the other hand, we need to make the step size small to minimize the quantization noise. One possible solution is to adapt the step size to the level of the input signal.

The basic idea behind *Adaptive PCM* (APCM) is to let the step size  $\Delta[n]$  be proportional to the standard deviation of the signal  $\sigma[n]$ :

$$\Delta[n] = \Delta_0 \sigma[n] \quad (7.19)$$

An equivalent method is to use a fixed quantizer but have a time-varying gain  $G[n]$ , which is inversely proportional to the signal's standard deviation

$$G[n] = G_0 / \sigma[n] \quad (7.20)$$

Estimation of the signal's variance, or short-time energy, is typically done by low-pass filtering  $x^2[n]$ . With a first-order IIR filter, the variance  $\sigma^2[n]$  is computed as

$$\sigma^2[n] = \alpha \sigma^2[n-1] + (1-\alpha)x^2[n-1] \quad (7.21)$$

with  $\alpha$  controlling the time constant of the filter  $T = -1/(F_s \ln \alpha)$ ,  $F_s$  the sampling rate, and  $0 < \alpha < 1$ . In practice,  $\alpha$  is chosen so that the time constant ranges between 1 ms ( $\alpha = 0.88$  at 8 kHz) and 10 ms ( $\alpha = 0.987$  at 8 kHz).

Alternatively,  $\sigma^2[n]$  can be estimated from the past  $M$  samples:

$$\sigma^2[n] = \frac{1}{M} \sum_{m=n-M}^{n-1} x^2[m] \quad (7.22)$$

In practice, it is advantageous to set limits on the range of values of  $\Delta[n]$  and  $G[n]$ :

$$\Delta_{\min} \leq \Delta[n] \leq \Delta_{\max} \quad (7.23)$$

$$G_{\min} \leq G[n] \leq G_{\max} \quad (7.24)$$

with the ratios  $\Delta_{\max} / \Delta_{\min}$  and  $G_{\max} / G_{\min}$  determining the dynamic range of the system. If our objective is to obtain a relatively constant SNR over a range of 40 dB, these ratios can be 100.

*Feedforward adaptation* schemes require us to transmit, in addition to the quantized signal, either the step size  $\Delta[n]$  or the gain  $G[n]$ . Because these values evolve slowly with time, they can be sampled and quantized at a low rate. The overall rate will be the sum of the

bit rate required to transmit the quantized signal plus the bit rate required to transmit either the gain or the step size.

Another class of adaptive quantizers use *feedback adaptation* to avoid having to send information about the step size or gain. In this case, the step size and gain are estimated from the quantizer output, so that they can be recreated at the decoder without any extra information. The corresponding short-time energy can then be estimated through a first-order IIR filter as in Eq. (7.21) or a rectangular window as in Eq. (7.22), but replacing  $x^2[n]$  by  $\hat{x}^2[n]$ .

Another option is to adapt the step size

$$\Delta[n] = P\Delta[n-1] \tag{7.25}$$

where  $P > 1$  if the previous codeword corresponds to the largest positive or negative quantizer level, and  $P < 1$  if the previous codeword corresponds to the smallest positive or negative quantizer level. A similar process can be done for the gain.

APCM exhibits an improvement between 4–8 dB over  $\mu$ -law PCM for the same bit rate.

### 7.2.4. Differential Quantization

Speech coding is about finding redundancy in the signal and removing it. We know that there is considerable correlation between adjacent samples, because on the average the signal doesn't change rapidly from sample to sample. A simple way of capturing this is to quantize the difference  $d[n]$  between the current sample  $x[n]$  and its predicted value  $\tilde{x}[n]$

$$d[n] = x[n] - \tilde{x}[n] \tag{7.26}$$

with its quantized value represented as

$$\hat{d}[n] = Q\{d[n]\} = d[n] + e[n] \tag{7.27}$$

where  $e[n]$  is the quantization error. Then, the quantized signal is the sum of the predicted signal  $\tilde{x}[n]$  and the quantized difference  $\hat{d}[n]$

$$\hat{x}[n] = \tilde{x}[n] + \hat{d}[n] = x[n] + e[n] \tag{7.28}$$

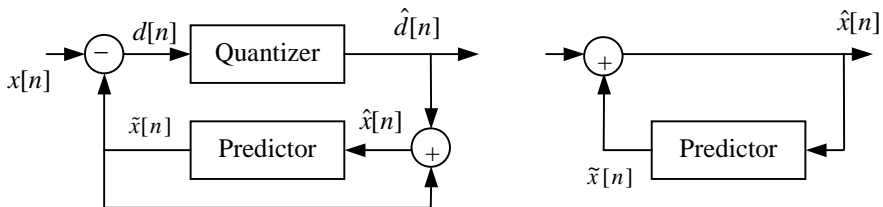


Figure 7.3 Block diagram of a DPCM encoder and decoder with feedback prediction.

If the prediction is good, Eq. (7.28) tells us that the quantization error will be small. Statistically, we need the variance of  $e[n]$  to be lower than that of  $x[n]$  for differential coding to provide any gain. Systems of this type are generically called *Differential Pulse Code Modulation* (DPCM) [11] and can be seen in Figure 7.3.

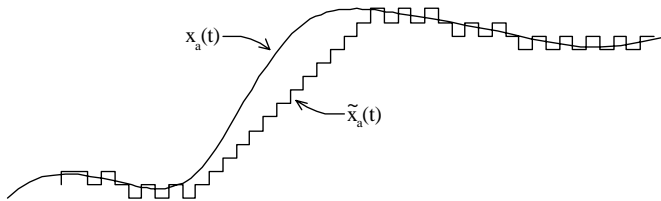
*Delta Modulation* (DM) [47] is a 1-bit DPCM, which predicts the current sample to be the same as the past sample:

$$\tilde{x}[n] = x[n-1] \quad (7.29)$$

so that we transmit whether the current sample is above or below the previous sample.

$$d[n] = \begin{cases} \Delta & x[n] > x[n-1] \\ -\Delta & x[n] \leq x[n-1] \end{cases} \quad (7.30)$$

with  $\Delta$  being the step size. If  $\Delta$  is too small, the reconstructed signal will not increase as fast as the original signal, a condition known as *slope overload distortion*. When the slope is small, the step size  $\Delta$  also determines the peak error; this is known as *granular noise*. Both quantization errors can be seen in Figure 7.4. The choice of  $\Delta$  that minimizes the mean squared error will be a tradeoff between slope overload and granular noise.

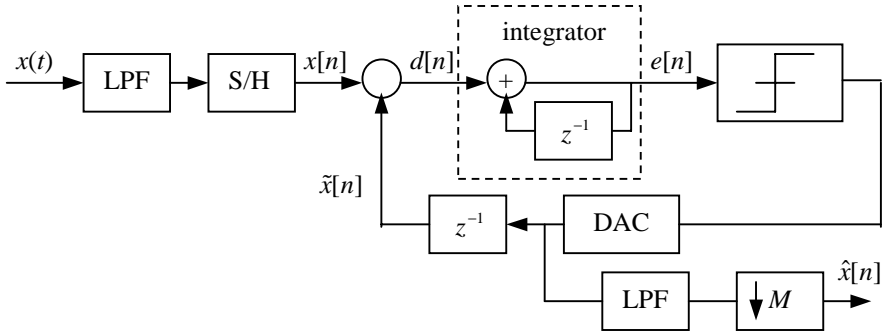


**Figure 7.4** An example of slope overload distortion and granular noise in a DM encoder.

If the signal is oversampled by a factor  $N$ , and the step size is reduced by the same amount (i.e.,  $\Delta/N$ ), the slope overload will be the same, but the granular noise will decrease by a factor  $N$ . While the coder is indeed very simple, sampling rates of over 200 kbps are needed for SNRs comparable to PCM, so DM is rarely used as a speech coder.

However, delta modulation is useful in the design of analog-digital converters, in a variant called sigma-delta modulation [44] shown in Figure 7.5. First the signal is lowpass filtered with a simple analog filter, and then it is oversampled. Whenever the predicted signal  $\tilde{x}[n]$  is below the original signal  $x[n]$ , the difference  $d[n]$  is positive. This difference  $d[n]$  is averaged over time with a digital integrator whose output is  $e[n]$ . If this situation persists, the accumulated error  $e[n]$  will exceed a positive value  $A$ , which causes a 1 to be encoded into the stream  $q[n]$ . A digital-analog converter is used in the loop which increments by one the value of the predicted signal  $\tilde{x}[n]$ . The system acts in the opposite way if the predicted signal  $\tilde{x}[n]$  is above the original signal  $x[n]$  for an extended period of time. Since the signal is oversampled, it changes very slowly from one sample to the next, and this quantization

can be accurate. The advantages of this technique as an analog-digital converter are that inexpensive analog filters can be used and only a simple 1-bit A/D is needed. The signal can next be low-passed filtered with a more accurate digital filter and then downsampled.



**Figure 7.5** A sigma-delta modulator used in an oversampling analog-digital converter.

*Adaptive Delta Modulation (ADM)* combines ideas from adaptive quantization and delta modulation with the so-called *Continuously Variable Slope Delta Modulation (CVSDM)* [22] having a step size that increases

$$\Delta[n] = \begin{cases} \alpha\Delta[n-1] + k_1 & \text{if } e[n], e[n-1] \text{ and } e[n-2] \text{ have same sign} \\ \alpha\Delta[n-1] + k_2 & \text{otherwise} \end{cases} \quad (7.31)$$

with  $0 < \alpha < 1$  and  $0 < k_2 \ll k_1$ . The step size increases if the last three errors have the same sign and decreases otherwise.

Improved DPCM is achieved through linear prediction in which  $\tilde{x}[n]$  is a linear combination of past quantized values  $\hat{x}[n]$

$$\tilde{x}[n] = \sum_{k=1}^p a_k \hat{x}[n-k] \quad (7.32)$$

DPCM systems with fixed prediction coefficients can provide from 4 to 11 dB improvement over direct linear PCM, for prediction orders up to  $p = 4$ , at the expense of increased computational complexity. Larger improvements can be obtained by adapting the prediction coefficients. The coefficients can be transmitted in a feedforward fashion or not transmitted if the feedback scheme is selected.

ADPCM [6] combines differential quantization with adaptive step-size quantization. ITU-T recommendation G.726 uses ADPCM at bit rates of 40, 32, 24, and 16 kbps, with 5, 4, 3, and 2 bits per sample, respectively. It employs an adaptive feedback quantizer and an adaptive feedback pole-zero predictor. Speech at bit rates of 40 and 32 kbps offer toll quality, while the other rates don't. G.727 is called embedded ADPCM because the 2-bit quantizer is embedded into the 3-bit quantizer, which is embedded into the 4-bit quantizer, and into the 5-bit quantizer. This makes it possible for the same codec to use a lower bit rate, with a graceful degradation in quality, if channel capacity is temporarily limited. Earlier



standards G.721 [7, 13] (created in 1984) and G.723 have been subsumed by G.726 and G.727. G.727 has a MOS of 4.1 for 32 kbps and is used in submarine cables. The Windows WAV format also supports a variant of ADPCM. These standards are shown in Table 7.2.

**Table 7.2** Common scalar waveform standards used.

Standard	Bit Rate (kbits/sec)	MOS	Algorithm	Sampling Rate (kHz)
Stereo CD Audio	1411	5.0	16-bit linear PCM	44.1
WAV, AIFF, SND	Variable	-	16/8-bit linear PCM	8, 11.025, 16, 22.05, 44.1, 48
G.711	64	4.3	$\mu$ -law/A-law PCM	8
G.727	40, 32, 24, 16	4.2 (32k)	ADPCM	8
G.722	64, 56, 48		Subband ADPCM	16

Wideband speech (50–7000 Hz) increases intelligibility of fricatives and overall perceived quality. In addition, it provides more subject presence and adds a feeling of transparent communication. ITU-T Recommendation G.722 encodes wideband speech with bit rates of 48, 56, and 64 kbps. Speech is divided into two subbands with QMF filters (see Chapter 5). The upper band is encoded using a 16-kbps ADPCM similar to the G.727 standard. The lower band is encoded using a 48-kbps ADPCM with the 4- and 5-bit quantizers embedded in the 6-bit quantizer. The quality of this system scores almost 1 MOS higher than that of telephone speech.

## 7.3. SCALAR FREQUENCY DOMAIN CODERS

Frequency domain is advantageous because:

1. The samples of a speech signal have a great deal of correlation among them, whereas frequency domain components are approximately uncorrelated and
2. The perceptual effects of masking described in Chapter 2 can be more easily implemented in the frequency domain. These effects are more pronounced for high-bandwidth signals, so frequency-domain coding has been mostly used for CD-quality signals and not for 8-kHz speech signals.

### 7.3.1. Benefits of Masking

As discussed in Chapter 2, masking is a phenomenon by which human listeners cannot perceive a sound if it is below a certain level. The consequence is that we don't need to encode

such sound. We now illustrate how this masked threshold is computed for MPEG<sup>3</sup>-1 layer 1. Given an input signal  $s[n]$  quantized with  $b$  bits, we obtain the normalized signal  $x[n]$  as

$$x[n] = \frac{s[n]}{N2^{b-1}} \quad (7.33)$$

where  $N = 512$  is the length of the DFT. Then, using a Hanning window,

$$w[n] = 0.5 - 0.5 \cos(2\pi n / N) \quad (7.34)$$

we obtain the log-power spectrum as

$$P[k] = P_0 + 10 \log_{10} \left( \sum_{n=0}^{N-1} w[n] x[n] e^{-j2\pi nk / N} \right) \quad (7.35)$$

where  $P_0$  is the playback SPL, which, in the absence of any volume information, is defined as 90 dB.

*Tonal* components are identified in Eq. (7.35) as local maxima, which exceed neighboring components within a certain bark distance by at least 7 dB. Specifically, bin  $k$  is tonal if and only if

$$P[k] > P[k \pm 1] \quad (7.36)$$

and

$$P[k] > P[k \pm l] + 7 \text{ dB} \quad (7.37)$$

where  $1 < l \leq \Delta_k$ , and  $\Delta_k$  is given by

$$\Delta_k = \begin{cases} 2 & 2 < k < 63 & (170\text{Hz} - 5.5\text{kHz}) \\ 3 & 63 \leq k < 127 & (5.5\text{kHz}, 11\text{kHz}) \\ 6 & 127 \leq k \leq 256 & (11\text{kHz}, 22\text{kHz}) \end{cases} \quad (7.38)$$

so that the power of that tonal masker is computed as the sum of the power in that bin and its left and right adjacent bins:

$$P_{TM}[k] = 10 \log_{10} \left( \sum_{j=-1}^1 10^{0.1P[k+j]} \right) \quad (7.39)$$

The noise maskers are computed from as the sum of power spectrum of the remaining frequency bins  $\bar{k}$  in a critical band not within a neighborhood  $\Delta_k$  of the tonal maskers:

<sup>3</sup> MPEG (Moving Picture Experts Group) is the nickname given to a family of International Standards for coding audiovisual information.

$$P_{NM}[\bar{k}] = 10 \log_{10} \left( \sum_j 10^{0.1P[j]} \right) \quad (7.40)$$

where  $j$  spans a critical band.

To compute the overall masked threshold we need to sum all masking thresholds contributed by each frequency bin  $i$ , which is approximately equal to the maximum (see Chapter 2):

$$T[k] = \max \left( T_h[k], \max_i (T_i[k]) \right) \quad (7.41)$$

In Chapter 2 we saw that whereas temporal postmasking can last from 50 to 300 ms, temporal premasking tends to last about 5 ms. This is also important because when a frequency transform is quantized, the blocking effects of transform's coders can introduce noise above the temporal premasking level that can be audible, since 1024 points corresponds to 23 ms at a 44-kHz sampling rate. To remove this pre-echo distortion, audible in the presence of castanets and other abrupt transient signals, subband filtering has been proposed, whose time constants are well below the 5-ms premasking time constant.

### 7.3.2. Transform Coders

We now use the *Adaptive Spectral Entropy Coding of High Quality Music Signals* (ASPEC) algorithm, which is the basis for the MPEG1 Layer 1 audio coding standard [24], to illustrate how transform coders work. The DFT coefficients are grouped into 128 subbands, and 128 scalar quantizers are used to transmit all the DFT coefficients. It has been empirically found that a difference of less than 1 dB between the original amplitude and the quantized value cannot be perceived. Each subband  $j$  has a quantizer having  $k_j$  levels and step size of  $T_j$  as

$$k_j = 1 + 2 \times \text{rnd} \left( P_j / T_j \right) \quad (7.42)$$

where  $T_j$  is the quantized JND threshold,  $P_j$  is the quantized magnitude of the largest real or imaginary component of the  $j^{\text{th}}$  subband, and  $\text{rnd}(\ )$  is the nearest integer rounding function. Entropy coding is used to encode the coefficients of that subband. Both  $T_j$  and  $P_j$  are quantized on a dB scale using 8-bit uniform quantizers with a 170-dB dynamic range, thus with a step size of 0.66 dB. Then they are transmitted as side information.

There are two main methods of obtaining a frequency-domain representation:

1. Through subband filtering via a filterbank (see Chapter 5). When a filterbank is used, the bandwidth of each band is chosen to increase with frequency following a perceptual scale, such as the Bark scale. As shown in Chapter 5, such filterbanks yield perfect reconstruction in the absence of quantization.

2. Through frequency-domain transforms. Instead of using a DFT, higher efficiency can be obtained by the use of an MDCT (see Chapter 5).

The exact implementation of the MPEG1 Layer 1 standard is much more complicated and beyond the scope of this book, though it follows the main ideas described here; the same is true for the popular MPEG1 layer III, also known as MP3. Implementation details can be found in [42].

### 7.3.3. Consumer Audio

Dolby Digital, MPEG, DTS and the Perceptual Audio Coder (PAC) [28] are all audio coders based on frequency-domain coding. Except for MPEG-1, which supports only stereo signals, the rest support multichannel.

Dolby Digital is multichannel digital audio, using lossy AC-3 [54] coding technology from original PCM with a sample rate of 48 kHz at up to 24 bits. The bit rate varies from 64 to 448 kbps, with 384 being the normal rate for 5.1 channels and 192 the normal rate for stereo (with or without surround encoding). Most Dolby Digital decoders support up to 640 kbps. Dolby Digital is the format used for audio tracks on almost all Digital Video/Versatile Discs (DVD). A DVD-5 with only one surround stereo audio stream (at 192 kbps) can hold over 55 hours of audio. A DVD-18 can hold over 200 hours.

MPEG was established in 1988 as part of the joint ISO (International Standardization Organization) / IEC (International Electrotechnical Commission) Technical Committee on Information technology. MPEG-1 was approved in 1992 and MPEG-2 in 1994. Layers I to III define several specifications that provide better quality at the expense of added complexity. MPEG-1 audio is limited to 384 kbps. MPEG1 Layer III audio [23], also known as MP3, is very popular on the Internet, and many compact players exist.

MPEG-2 Audio, one of the audio formats used in DVD, is multichannel digital audio, using lossy compression from 16-bit linear PCM at 48 kHz. Tests have shown that for nearly all types of speech and music, at a data rate of 192 kbps and over, on a stereo channel, scarcely any difference between original and coded versions was observable (ranking of coded item > 4.5), with the original signal needing 1.4 Mbps on a CD (reduction by a factor of 7). One advantage of the MPEG Audio technique is that future findings regarding psychoacoustic effects can be incorporated later, so it can be expected that today's quality level using 192 kbps will be achievable at lower data rates in the future. A variable bit rate of 32 to 912 kbps is supported for DVDs.

DTS (Digital Theater Systems) Digital Surround is another multi-channel (5.1) digital audio format, using lossy compression derived from 20-bit linear PCM at 48 kHz. The compressed data rate varies from 64 to 1536 kbps, with typical rates of 768 and 1536 kbps.

### 7.3.4. Digital Audio Broadcasting (DAB)

*Digital Audio Broadcasting* (DAB) is a means of providing current AM and FM listeners with a new service that offers: sound quality comparable to that of compact discs, increased

service availability (especially for reception in moving vehicles), flexible coverage scenarios, and high spectrum efficiency.

Different approaches have been considered for providing listeners with such a service. Currently, the most advanced system is one commonly referred to as Eureka 147 DAB, which has been under development in Europe under the Eureka Project EU147 since 1988. Other approaches include various American in-band systems (IBOC, IBAC, IBRC, FMDigital, and FMeX) still in development, as well as various other systems promising satellite delivery, such as WorldSpace and CD Radio, still in development as well. One satellite-delivery system called MediaStar (formerly Archimedes) proposes to use the Eureka 147 DAB signal structure, such that a single receiver could access both terrestrial and satellite broadcasts.

DAB has been under development since 1981 at the Institut für Rundfunktechnik (IRT) and since 1987 as part of a European research project (Eureka 147). The Eureka 147 DAB specification was standardized by the European Telecommunications Standards Institute (ETSI) in February 1995 as document ETS 300 401, with a draft second edition issued in June 1996. In December 1994, the International Telecommunication Union—Radiocommunication (ITU-R) recommended that this technology, referred to as Digital System A, be used for implementing DAB services.

The Eureka 147 DAB signal consists of multiple carriers within a 1.536-MHz channel bandwidth. Four possible modes of operation define the channel coding configuration, specifying the total number of carriers, the carrier spacing, and also the guard interval duration. Each channel provides a raw data rate of 2304 kbps; after error protection, a useful data rate of anywhere between approximately 600 kbps up to 1800 kbps is available to the service provider, depending on the user-specified multiplex configuration. This useful data rate can be divided into an infinite number of possible configurations of audio and data programs. All audio programs are individually compressed using MUSICAM (MPEG-1 Layer II).

For each useful bit,  $1\frac{1}{3}$  ... 4 bits are transmitted. This extensive redundancy makes it possible to reconstruct the transmitted bit sequence in the receiver, even if part of it is disrupted during transmission (FEC—forward error correction). In the receiver, error concealment can be carried out at the audio reproduction stage, so that residual transmission errors which could not be corrected do not always cause disruptive noise.

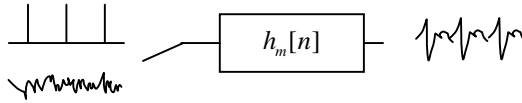
## 7.4. CODE EXCITED LINEAR PREDICTION (CELP)

The use of linear predictors removes redundancy in the signal, so that coding of the residual signal can be done with simpler quantizers. We first introduce the LPC vocoder and then introduce coding of the residual signal with a very popular technique called CELP.

### 7.4.1. LPC Vocoder

A typical model for speech production is shown in Figure 7.6, which has a source, or excitation, driving a linear time-varying filter. For voiced speech, the excitation is an impulse train spaced  $P$  samples apart. For unvoiced speech, the source is white random noise. The filter

$h_m[n]$  for frame  $m$  changes at regular intervals, say every 10 ms. If this filter is represented with linear predictive coding, it is called an *LPC vocoder* [3].



**Figure 7.6** Block diagram of an LPC vocoder.

In addition to transmitting the gain and LPC coefficients, the encoder has to determine whether the frame is voiced or unvoiced, as well as the pitch period  $P$  for voiced frames.

The LPC vocoder produces reasonable quality for unvoiced frames, but often results in somewhat mechanical sound for voiced sounds, and a buzzy quality for voiced fricatives. More importantly, the LPC vocoder is quite sensitive to voicing and pitch errors, so that an accurate pitch tracker is needed for reasonable quality. The LPC vocoder also performs poorly in the presence of background noise. Nonetheless, it can be highly intelligible. The Federal Standard 1015 [55], proposed for secure communications, is based on a 2.4-kbps LPC vocoder.

It's also possible to use linear predictive coding techniques together with Huffman coding [45] to achieve lossless compression of up to 50%.

## 7.4.2. Analysis by Synthesis

*Code Excited Linear Prediction* (CELP) [5] is an umbrella for a family of techniques that quantize the LPC residual using VQ, thus the term *code excited*, using analysis by synthesis. In addition CELP uses the fact that the residual of voiced speech has periodicity and can be used to predict the residual of the current frame. In CELP coding the LPC coefficients are quantized and transmitted (feedforward prediction), as well as the codeword index. The prediction using LPC coefficients is called *short-term prediction*. The prediction of the residual based on pitch is called *long-term prediction*. To compute the quantized coefficients we use an *analysis-by-synthesis* technique, which consists of choosing the combination of parameters whose reconstructed signal is closest to the analysis signal. In practice, not all coefficients of a CELP coder are estimated in an analysis-by-synthesis manner.

We first estimate the  $p^{\text{th}}$ -order LPC coefficients from the samples  $x[n]$  for frame  $t$  using the autocorrelation method, for example. We then quantize the LPC coefficients to  $(a_1, a_2, \dots, a_p)$  with the techniques described in Section 7.4.5. The residual signal  $e[n]$  is obtained by inverse filtering  $x[n]$  with the quantized LPC filter

$$e[n] = \sum_{i=1}^p a_i x[n-i] \quad (7.43)$$

Given the transfer function of the LPC filter

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - \sum_{i=1}^p a_i z^{-i}} = \sum_{i=0}^{\infty} h_i z^{-i} \quad (7.44)$$

we can obtain the first  $M$  coefficients of the impulse response  $h[n]$  of the LPC filter by driving it with an impulse as

$$h[n] = \begin{cases} 1 & n = 0 \\ \sum_{i=1}^n a_i h[n-i] & 0 < n < p \\ \sum_{i=1}^p a_i h[n-i] & p \leq n < M \end{cases} \quad (7.45)$$

so that if we quantize a frame of  $M$  samples of the residual  $\mathbf{e} = (e[0], e[1], \dots, e[M-1])^T$  to  $\mathbf{e}_i = (e_i[0], e_i[1], \dots, e_i[M-1])^T$ , we can compute the reconstructed signal  $\hat{x}_i[n]$  as

$$\hat{x}_i[n] = \sum_{m=0}^n h[m] e_i[n-m] + \sum_{m=n+1}^{\infty} h[m] e[n-m] \quad (7.46)$$

where the second term in the sum depends on the residual for previous frames, which we already have. Let's define signal  $r_0[n]$  as the second term of Eq. (7.46):

$$r_0[n] = \sum_{m=n+1}^{\infty} h[m] e[n-m] \quad (7.47)$$

which is the output of the LPC filter when there is no excitation for frame  $t$ . The important thing to note is that  $r_0[n]$  does not depend on  $e_i[n]$

It is convenient to express Eqs. (7.46) and (7.47) in matrix form as

$$\hat{\mathbf{x}}_i = \mathbf{H} \mathbf{e}_i + \mathbf{r}_0 \quad (7.48)$$

where matrix  $\mathbf{H}$  corresponds to the LPC filtering operation with its memory set to 0:

$$\mathbf{H} = \begin{bmatrix} h_0 & 0 & \cdots & 0 & 0 \\ h_1 & h_0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ h_{M-1} & h_{M-2} & \cdots & h_0 & 0 \\ h_M & h_{M-1} & \cdots & h_1 & h_0 \end{bmatrix} \quad (7.49)$$

Given the large dynamic range of the residual signal, we use gain-shape quantization, where we quantize the gain and the gain-normalized residual separately:

$$\mathbf{e}_i = \lambda \mathbf{c}_i \quad (7.50)$$

where  $\lambda$  is the gain and  $\mathbf{c}_i$  is the codebook entry  $i$ . This codebook is known as the *fixed codebook* because its vectors do not change from frame to frame. Usually the size of the codebook is selected as  $2^N$  so that full use is made of all  $N$  bits. Codebook sizes typically vary from 128 to 1024. Combining Eq. (7.48) with Eq. (7.50), we obtain

$$\hat{\mathbf{x}}_i = \lambda \mathbf{H} \mathbf{c}_i + \mathbf{r}_0 \quad (7.51)$$

The error between the original signal  $\mathbf{x}$  and the reconstructed signal  $\hat{\mathbf{x}}_i$  is

$$\boldsymbol{\varepsilon} = \mathbf{x} - \hat{\mathbf{x}}_i \quad (7.52)$$

The optimal gain  $\lambda$  and codeword index  $i$  are the ones that minimize the squared error between the original signal and the reconstructed<sup>4</sup> signal:

$$E(i, \lambda) = |\mathbf{x} - \hat{\mathbf{x}}_i|^2 = |\mathbf{x} - \lambda \mathbf{H} \mathbf{c}_i - \mathbf{r}_0|^2 = |\mathbf{x} - \mathbf{r}_0|^2 + \lambda^2 \mathbf{c}_i^T \mathbf{H}^T \mathbf{H} \mathbf{c}_i - 2\lambda \mathbf{c}_i^T \mathbf{H}^T (\mathbf{x} - \mathbf{r}_0) \quad (7.53)$$

where the term  $|\mathbf{x} - \mathbf{r}_0|^2$  does not depend on  $\lambda$  or  $i$  and can be neglected in the minimization. For a given  $\mathbf{c}_i$ , the gain  $\lambda_i$  that minimizes Eq. (7.53) is given by

$$\lambda_i = \frac{\mathbf{c}_i^T \mathbf{H}^T (\mathbf{x} - \mathbf{r}_0)}{\mathbf{c}_i^T \mathbf{H}^T \mathbf{H} \mathbf{c}_i} \quad (7.54)$$

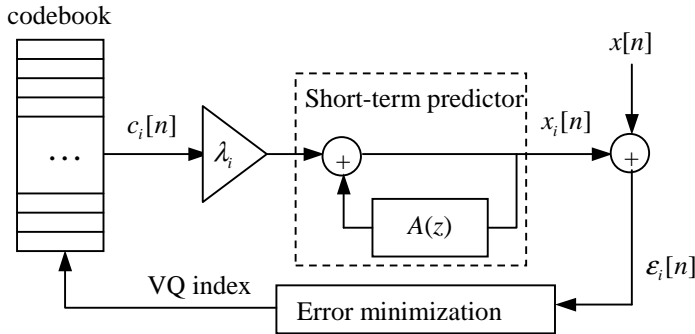
Inserting Eq. (7.54) into (7.53) lets us compute the index  $j$  as the one that minimizes

$$j = \arg \min_i \left\{ -\frac{(\mathbf{c}_i^T \mathbf{H}^T (\mathbf{x} - \mathbf{r}_0))^2}{\mathbf{c}_i^T \mathbf{H}^T \mathbf{H} \mathbf{c}_i} \right\} \quad (7.55)$$

So we first obtain the codeword index  $j$  according to Eq. (7.55) and then the gain  $\lambda_j$  according to Eq. (7.54), which is scalarly quantized to  $\hat{\lambda}_j$ . Both codeword index  $j$  and  $\hat{\lambda}_j$  are transmitted. In the algorithm described here, we first chose the quantized LPC coefficients  $(a_1, a_2, \dots, a_p)$  independently of the gains and codeword index, and then we chose the codeword index independently of the quantized gain  $\hat{\lambda}_j$ . This procedure is called *open-loop* estimation, because some parameters are obtained independently of the others. This is shown in Figure 7.7. *Closed-loop* estimation [49] means that all possible combinations of quantized parameters are explored. Closed-loop is more computationally expensive but yields lower squared error.

<sup>4</sup> A beginner's mistake is to find the codebook index that minimizes the squared error of the residual. This does not minimize the difference between the original signal and the reconstructed signal.





**Figure 7.7** Analysis-by-synthesis principle used in a basic CELP.

### 7.4.3. Pitch Prediction: Adaptive Codebook

The fact that speech is highly periodic during voiced segments can also be used to reduce redundancy in the signal. This can be done by predicting the residual signal  $e[n]$  at the current vector with samples from the past residual signal shifted a pitch period  $t$ :

$$e[n] = \lambda_i^a e[n-t] + \lambda_i^f c_i^f[n] = \lambda_i^a c_i^a[n] + \lambda_i^f c_i^f[n] \quad (7.56)$$

Using the matrix framework we described before, Eq. (7.56) can be expressed as

$$\mathbf{e}_{ii} = \lambda_i^a \mathbf{c}_i^a + \lambda_i^f \mathbf{c}_i^f \quad (7.57)$$

where we have made use of an *adaptive codebook* [31], where  $\mathbf{c}_i^a$  is the adaptive codebook entry  $j$  with corresponding gain  $\lambda^a$ , and  $\mathbf{c}_i^f$  is the fixed or stochastic codebook entry  $i$  with corresponding gain  $\lambda^f$ . The adaptive codebook entries are segments of the recently synthesized excitation signal

$$\mathbf{c}_i^a = (e[-t], e[1-t], \dots, e[M-1-t])^T \quad (7.58)$$

where  $t$  is the delay which specifies the start of the adaptive codebook entry  $t$ . The range of  $t$  is often between 20 and 147, since this can be encoded with 7 bits. This corresponds to a range in pitch frequency between 54 and 400 Hz for a sampling rate of 8 kHz.

The contribution of the adaptive codebook is much larger than that of the stochastic codebook for voiced sounds. So we generally search for the adaptive codebook first, using Eq. (7.58) and a modified version of Eqs. (7.55), (7.54), replacing  $i$  by  $t$ . Closed-loop search of both  $t$  and gain here often yields a much larger error reduction.

### 7.4.4. Perceptual Weighting and Postfiltering

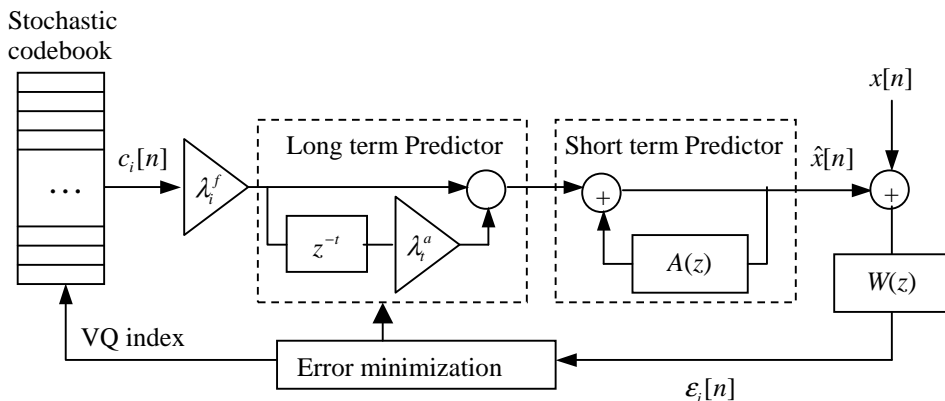
The objective of speech coding is to reduce the bit rate while maintaining a perceived level of quality; thus, minimization of the error is not necessarily the best criterion. A perceptual weighting filter tries to shape the noise so that it gets masked by the speech signal (see Chapter 2). This generally means that most of the quantization noise energy is located in spectral regions where the speech signal has most of its energy. A common technique [4] consists in approximating this perceptual weighting with a linear filter

$$W(z) = \frac{A(z/\beta)}{A(z/\gamma)} \quad (7.59)$$

where  $A(z)$  is the predictor polynomial

$$A(z) = 1 - \sum_{i=1}^p a_i z^{-i} \quad (7.60)$$

Choosing  $\gamma$  and  $\beta$  so that  $0 < \gamma < \beta \leq 1$ , implies that the roots of  $A(z/\beta)$  and  $A(z/\gamma)$  will move closer to the origin of the unit circle than the roots of  $A(z)$ , thus resulting in a frequency response with wider resonances. This perceptual filter therefore deemphasizes the contribution of the quantization error near the formants. A common choice of parameters is  $\beta = 1.0$  and  $\gamma = 0.8$ , since it simplifies the implementation. This filter can easily be included in the matrix  $\mathbf{H}$ , and a CELP coder incorporating the perceptual weighting is shown in Figure 7.8.



**Figure 7.8** Diagram of a CELP coder. Both long-term and short-term predictors are used, together with a perceptual weighting.

Despite the perceptual weighting filter, the reconstructed signal still contains audible noise. This filter reduces the noise in those frequency regions that are perceptually irrelevant without degrading the speech signal. The postfilter generally consists of a short-term postfilter to emphasize the formant structure and a long-term postfilter to enhance the periodicity

of the signal [10]. One possible implementation follows Eq. (7.59) with values of  $\beta = 0.5$  and  $\gamma = 0.75$ .

### 7.4.5. Parameter Quantization

To achieve a low bit rate, all the coefficients need to be quantized. Because of its coding efficiency, vector quantization is the compression technique of choice to quantize the predictor coefficients. The LPC coefficients cannot be quantized directly, because small errors produced in the quantization process may result in large changes in the spectrum and possibly unstable filters. Thus, equivalent representations that guarantee stability are used, such as reflection coefficients, log-area ratios, and the line spectral frequencies (LSF) described in Chapter 6. LSF are used most often, because it has been found empirically that they behave well when they are quantized and interpolated [2]. For 8 kHz, 10 predictor coefficients are often used, which makes using a single codebook impractical because of the large dimension of the vector. Split-VQ [43] is a common choice, where the vectors are divided into several subvectors, and each is vector quantized. Matrix quantization can also be used to exploit the correlation of these subvectors across consecutive time frames. *Transparent quality*, defined as average spectral distortion below 1 dB with no frames above 4 dB, can be achieved with fewer than 25 bits per frame.

A frame typically contains around 20 to 30 milliseconds, which at 8 kHz represents 160–240 samples. Because of the large vector dimension, it is impractical to quantize a whole frame with a single codebook. To reduce the dimensionality, the frame is divided into four or more nonoverlapping sub-frames. The LSF coefficients for each subframe are linearly interpolated between the two neighboring frames.

A typical range of the pitch prediction for an 8-kHz sampling rate goes from 2 to 20 ms, from 20 to 147 samples, 2.5 ms to 18.375 ms, which can be encoded with 7 bits. An additional bit is often used to encode fractional delays for the lower pitch periods. These fractional delays can be implemented through upsampling as described in Chapter 5. The subframe gain of the adaptive codebook can be effectively encoded with 3 or 4 bits. Alternatively, the gains of all sub-frames within a frame can be encoded through VQ, resulting in more efficient compression.

The fixed codebook can be trained from data using the techniques described in Chapter 4. This will offer the lowest distortion for the training set but doesn't guarantee low distortion for mismatched test signals. Also, it requires additional storage, and full search increases computation substantially.

Since subframes should be approximately white, the codebook can be populated from samples of a white process. A way of reducing computation is to let those noise samples be only +1, 0, or -1, because only additions are required. Codebooks of a specific type, known as *algebraic codebooks* [1], offer even more computational savings because they contain many 0s. Locations for the 4 pulses per subframe under the G.729 standard are shown in Table 7.3.

Full search can efficiently be done with this codebook structure. Algebraic codebooks can provide almost as low distortion as trained codebooks can, with low computational complexity.

**Table 7.3** Algebraic codebooks for the G.729 standard. Each of the four codebooks has one pulse in one possible location indicated by 3 bits for the first three codebooks and 4 bits for the last codebook. The sign is indicated by an additional bit. A total of 17 bits are needed to encode a 40-sample subframe.

Amplitude	Positions
$\pm 1$	0, 5, 10, 15, 20, 25, 30, 35
$\pm 1$	1, 6, 11, 16, 21, 26, 31, 36
$\pm 1$	2, 7, 12, 17, 22, 27, 32, 37
$\pm 1$	3, 8, 13, 18, 23, 28, 33, 38 4, 9, 14, 19, 24, 29, 34, 39

#### 7.4.6. CELP Standards

There are many standards for speech coding based on CELP, offering various points in the bit-rate/quality plane, mostly depending on when they were created and how refined the technology was at that time.

Voice over Internet Protocol (Voice over IP) consists of transmission of voice through data networks such as the Internet. H.323 is an umbrella standard which references many other ITU-T recommendations. H.323 provides the system and component descriptions, call model descriptions, and call signaling procedures. For audio coding, G.711 is mandatory, while G.722, G.728, G.723.1, and G.729 are optional. G.728 is a low-delay CELP coder that offers toll quality at 16 kbps [9], using a feedback 50<sup>th</sup>-order predictor, but no pitch prediction. G.729 [46] offers toll quality at 8 kbps, with a delay of 10 ms. G.723.1, developed by DSP Group, including Audiocodes Ltd., France Telecom, and the University of Sherbrooke, has slightly lower quality at 5.3 and 6.3 kbps, but with a delay of 30 ms. These standards are shown in Table 7.4.

**Table 7.4** Several CELP standards used in the H.323 specification used for teleconferencing and voice streaming through the internet.

Standard	Bit Rate (kbps)	MOS	Algorithm	H.323	Comments
G.728	16	4.0	No pitch prediction	Optional	Low -delay
G.729	8	3.9	ACELP	Optional	
G.723.1	5.3, 6.3	3.9	ACELP for 5.3k	Optional	

In 1982, the Conference of European Posts and Telegraphs (CEPT) formed a study group called the Groupe Spécial Mobile (GSM) to study and develop a pan-European public land mobile system. In 1989, GSM responsibility was transferred to the European Telecommunication Standards Institute (ETSI), and the phase I GSM specifications were published in 1990. Commercial service was started in mid 1991, and by 1993 there were 36 GSM networks in 22 countries, with 25 additional countries considering or having already selected GSM. This is not only a European standard; South Africa, Australia, and many Middle and Far East countries have chosen GSM. The acronym GSM now stands for Global System for Mobile telecommunications. The GSM group studied several voice coding algorithms on the basis of subjective speech quality and complexity (which is related to cost, processing delay, and power consumption once implemented) before arriving at the choice of a Regular Pulse Excited–Linear Predictive Coder (RPE-LPC) with a Long Term Predictor loop [56]. Neither the original full-rate at 13 kbps [56] nor the half-rate at 5.6 kbps [19] achieves toll quality, though the enhanced full-rate (EFR) standard based on ACELP [26] has toll quality at the same rates.

The *Telecommunication Industry Association* (TIA) and the *Electronic Industries Alliance* (EIA) are organizations accredited by the *American National Standards Institute* (ANSI) to develop voluntary industry standards for a wide variety of telecommunication products. TR-45 is the working group within TIA devoted to mobile and personal communication systems. Time Division Multiple Access (TDMA) is a digital wireless technology that divides a narrow radio channel into framed time slots (typically 3 or 8) and allocates a slot to each user. The TDMA Interim Standard 54, or TIA/EIA/IS54, was released in early 1991 by both TIA and EIA. It is available in North America at both the 800-MHz and 1900-MHz bands. IS54 [18] at 7.95 kbps is used in North America's TDMA (Time Division Multiple Access) digital telephony and has quality similar to the original full-rate GSM. TDMA IS-136 is an update released in 1994.

**Table 7.5** CELP standards used in cellular telephony.

Standard	Bit Rate (kbps)	MOS	Algorithm	Cellular	Comments
Full-rate GSM	13	3.6	VSELP RTE-LTP	GSM	
EFR GSM	12.2	4.5	ACELP	GSM	
IS-641	7.4	4.1	ACELP	PCS1900	
IS-54	7.95	3.9	VSELP	TDMA	
IS-96a	max 8.5	3.9	QCELP	CDMA	Variable-rate

*Code Division Multiple Access* (CDMA) is a form of *spread spectrum*, a family of digital communication techniques that have been used in military applications for many years. The core principle is the use of noiselike carrier waves, and, as the name implies, bandwidths much wider than that required for simple point-to-point communication at the same data rate. Originally there were two motivations: either to resist enemy efforts to jam

the communications (anti-jam, or AJ) or to hide the fact that communication was even taking place, sometimes called low probability of intercept (LPI). The service started in 1996 in the United States, and by the end of 1999 there were 50 million subscribers worldwide. IS-96 QCELP [14], used in North America's CDMA, offers variable-rate coding at 8.5, 4, 2 and 0.8 kbps. The lower bit rate is transmitted when the coder detects background noise. TIA/EIA/IS-127-2 is a standard for an enhanced variable-rate codec, whereas TIA/EIA/IS-733-1 is a standard for high-rate. Standards for CDMA, TDMA, and GSM are shown in Table 7.5.

*Third generation (3G)* is the generic term used for the next generation of mobile communications systems. 3G systems will provide enhanced services to those—such as voice, text, and data—predominantly available today. The Universal Mobile Telecommunications System (UMTS) is a part of ITU's International Mobile Telecommunications (IMT)-2000 vision of a global family of third-generation mobile communications systems. It has been assigned to the frequency bands 1885–2025 and 2110–2200 MHz. The first networks are planned to launch in Japan in 2001, with European countries following in early 2002. A major part of 3G is General Packet Radio Service (GPRS), under which carriers charge by the packet rather than by the minute. The speech coding standard for CDMA2000, the umbrella name for the third-generation standard in the United States, is expected to gain approval late in 2000. An adaptive multi rate wideband speech codec has also been proposed for the GSM's 3G [16], which has five modes of operation from 24 kbps down to 9.1 kbps.

While most of the work described above uses a sampling rate of 8 kHz, there has been growing interest in using CELP techniques for high bandwidth and particularly in a scalable way so that a basic layer contains the lower frequency and the higher layer either is a full-band codec [33] or uses a parametric model [37].

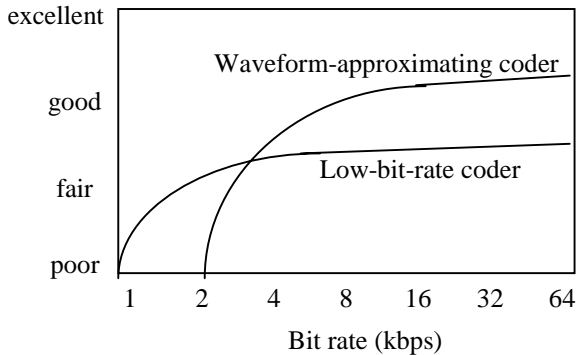
## 7.5. LOW-BIT RATE SPEECH CODERS

In this section we describe a number of low-bit-rate speech coding techniques including the mixed-excitation LPC vocoder, harmonic coding, and waveform interpolation. These coding techniques are also used extensively in speech synthesis.

*Waveform-approximating coders* are designed to minimize the difference between the original signal and the coded signal. Therefore, they produce a reconstructed signal whose SNR goes to infinity as the bit rate increases, and they also behave well when the input signal is noisy or music. In this category we have the scalar waveform coders of Section 7.2, the frequency-domain coders of Section 7.3, and the CELP coders of Section 7.4.

Low-bit-rate coders, on the other hand, do not attempt to minimize the difference between the original signal and the quantized signal. Since these coders are designed to operate at low bit rates, their SNR does not generally approach infinity even if a large bit rate is used. The objective is to compress the original signal with another one that is perceptually equivalent. Because of the reliance on an inaccurate model, these low-bit-rate coders often distort the speech signal even if the parameters are not quantized. In this case, the distortion can consist of more than quantization noise. Furthermore, these coders are more sensitive to the presence of noise in the signal, and they do not perform as well on music.

In Figure 7.9 we compare the MOS of waveform approximating coders and low-bit-rate coders as a function of the bit rate. CELP uses a model of speech to obtain as much prediction as possible, yet allows for the model not to be exact, and thus is a waveform-approximating coder. CELP is a robust coder that works reasonably well when the assumption of only a clean speech signal breaks either because of additive noise or because there is music in the background. Researchers are working on the challenging problem of creating more scalable coders that offer best performance at all bit rates.



**Figure 7.9** Typical subjective performance of waveform-approximating and low-bit-rate coders as a function of the bit rate. Note that waveform-approximating coders are a better choice for bit rates higher than about 3 kbps, whereas parametric coders are a better choice for lower bit rates. The exact cutoff point depends on the specific algorithms compared.

### 7.5.1. Mixed-Excitation LPC Vocoder

The main weakness of the LPC vocoder is the binary decision between voiced and unvoiced speech, which results in errors especially for noisy speech and voiced fricatives. By having a separate voicing decision for each of a number of frequency bands, the performance can be enhanced significantly [38]. The new proposed U.S. Federal Standard at 2.4 kbps is a Mixed Excitation Linear Prediction (MELP) LPC vocoder [39], which has a MOS of about 3.3. This exceeds the quality of the older 4800-bps federal standard 1016 [8] based on CELP. The bit rate of the proposed standard can be reduced while maintaining the same quality by jointly quantizing several frames together [57]. A hybrid codec that uses MELP in strongly voiced regions and CELP in weakly voiced and unvoiced regions [53] has shown to yield lower bit rates. MELP can also be combined with the waveform interpolation technique of Section 7.5.3 [50].

### 7.5.2. Harmonic Coding

Sinusoidal coding decomposes the speech signal [35] or the LP residual signal [48] into a sum of sinusoids. The case where these sinusoids are harmonically related is of special in-

terest for speech synthesis (see Chapter 16), so we will concentrate on it in this section, even though a similar treatment can be followed for the case where the sinusoids are not harmonically related. In fact, a combination of harmonically related and nonharmonically related sinusoids can also be used [17]. We show in Section 7.5.2.2 that we don't need to transmit the phase of the sinusoids, only the magnitude.

As shown in Chapter 5, a periodic signal  $\tilde{s}[n]$  with period  $T_0$  can be expressed as a sum of  $T_0$  harmonic sinusoids

$$\tilde{s}[n] = \sum_{l=0}^{T_0-1} A_l \cos(nl\omega_0 + \phi_l) \quad (7.61)$$

whose frequencies are multiples of the fundamental frequency  $\omega_0 = 2\pi/T_0$ , and where  $A_l$  and  $\phi_l$  are the sinusoid amplitudes and phases, respectively. If the pitch period  $T_0$  has fractional samples, the sum in Eq. (7.61) includes only the integer part of  $T_0$  in the summation. Since a real signal  $s[n]$  will not be perfectly periodic in general, we have a modeling error

$$e[n] = s[n] - \tilde{s}[n] \quad (7.62)$$

We can use short-term analysis to estimate these parameters from the input signal  $s[n]$  at frame  $k$ , in the neighborhood of  $t = kN$ , where  $N$  is the frame shift:

$$s_k[n] = s[n]w_k[n] = s[n]w[kN - n] \quad (7.63)$$

if we make the assumption that the sinusoid parameters for frame  $k$  ( $\omega_0^k$ ,  $A_l^k$  and  $\phi_l^k$ ) are constant within the frame.

At resynthesis time, there will be discontinuities at unit boundaries, due to the block processing, unless we specifically smooth the parameters over time. One way of doing this is with overlap-add method between frames  $(k - 1)$  and  $k$ :

$$\hat{s}[n] = w[n]\tilde{s}^{k-1}[n] + w[n - N]\tilde{s}^k[n - N] \quad (7.64)$$

where the window  $w[n]$  must be such that

$$w[n] + w[n - N] = 1 \quad (7.65)$$

to achieve perfect reconstruction. This is the case for the common Hamming and Hanning windows.

This harmonic model [35] is similar to the classic filterbank, though rather than the whole spectrum we transmit only the fundamental frequency  $\omega_0$  and the amplitudes  $A_l$  and phases  $\phi_l$  of the harmonics. This reduced representation doesn't result in loss of quality for a frame shift  $N$  that corresponds to 12 ms or less. For unvoiced speech, using a default pitch of 100 Hz results in acceptable quality.

### 7.5.2.1. Parameter Estimation

For simplicity in the calculations, let's define  $\tilde{s}[n]$  as a sum of complex exponentials



$$\tilde{s}[n] = \sum_{l=0}^{T_0-1} A_l \exp\{j(nl\omega_0 + \phi_l)\} \quad (7.66)$$

and perform short-time Fourier transform with a window  $w[n]$

$$\tilde{S}_w(\omega) = \sum_{l=0}^{T_0-1} A_l e^{j\phi_l} W(\omega - l\omega_0) \quad (7.67)$$

where  $W(\omega)$  is the Fourier transform of the window function. The goal is to estimate the sinusoid parameters as those that minimize the squared error:

$$E = |S(\omega) - \tilde{S}_w(\omega)|^2 \quad (7.68)$$

If the main lobes of the analysis window do not overlap, we can estimate the phases  $\phi_l$  as

$$\phi_l = \arg S(l\omega_0) \quad (7.69)$$

and the amplitudes  $A_l$  as

$$A_l = \frac{|S(l\omega_0)|}{W(0)} \quad (7.70)$$

For example, the Fourier transform of a  $(2N + 1)$  point rectangular window centered around the origin is given by

$$W(\omega) = \frac{\sin((2N + 1)\omega / 2)}{\sin(\omega / 2)} \quad (7.71)$$

whose main lobes will not overlap in Eq. (7.67) if  $2T_0 < 2N + 1$ : i.e., the window contains at least two pitch periods. The implicit assumption in the estimates of Eqs. (7.69) and (7.70) is that there is no spectral leakage, but a rectangular window does have significant spectral leakage, so a different window is often used in practice. For windows such as Hanning or Hamming, which reduce the leakage significantly, it has been found experimentally that these estimates are correct if the window contains at least two and a half pitch periods.

Typically, the window is centered around 0 (nonzero in the interval  $-N \leq n \leq N$ ) to avoid numerical errors in estimating the phases.

Another implicit assumption in Eqs. (7.69) and (7.70) is that we know the fundamental frequency  $\omega_0$  ahead of time. Since, in practice, this is not the case, we can estimate it as the one which minimizes Eq. (7.68). This pitch-estimation method can generate pitch doubling or tripling when a harmonic falls within a formant that accounts for the majority of the signal's energy.

Voiced/unvoiced decisions can be computed from the ratio between the energy of the signal and that of the reconstruction error

$$SNR = \frac{\sum_{n=-N}^N |s[n]|^2}{\sum_{n=-N}^N |s[n] - \tilde{s}[n]|^2} \quad (7.72)$$

where it has been empirically found that frames with SNR higher than 13 dB are generally voiced and lower than 4 dB unvoiced. In between, the signal is considered to contain a mixed excitation. Since speech is not perfectly stationary within the analysis frame, even noise-free periodic signals will yield finite SNR.

For unvoiced speech, a good assumption is to default to a pitch of 100 Hz. The use of fewer sinusoids leads to perceptual artifacts.

Improved quality can be achieved by using an analysis-by-synthesis framework [17, 34] since the closed-loop estimation is more robust to pitch-estimation and voicing decision errors.

### 7.5.2.2. Phase Modeling

An impulse train  $e[n]$ , a periodic excitation, can be expressed as a sum of complex exponentials

$$e[n] = T_0 \sum_{k=-\infty}^{\infty} \delta[n - n_0 - kT_0] = \sum_{l=0}^{T_0-1} e^{j(n-n_0)\omega_0 l} \quad (7.73)$$

which, if passed through a filter  $H(\omega) = A(\omega) \exp \Phi(\omega)$ , will generate

$$s[n] = \sum_{l=0}^{T_0-1} A(l\omega_0) \exp \{ j[(n - n_0)\omega_0 l + \Phi(l\omega_0)] \} \quad (7.74)$$

Comparing Eq. (7.66) with (7.74), the phases of our sinusoidal model are given by

$$\phi_l = -n_0 \omega_0 l + \Phi(l\omega_0) \quad (7.75)$$

Since the sinusoidal model has too many parameters to lead to low-rate coding, a common technique is to not encode the phases. In Chapter 6 we show that if a system is considered minimum phase, the phases can be uniquely recovered from knowledge of the magnitude spectrum.

The magnitude spectrum is known at the pitch harmonics, and the remaining values can be filled in by interpolation: e.g., linear or cubic splines [36]. This interpolated magnitude spectrum can be approximated through the real cepstrum:

$$|\tilde{A}(\omega)| = c_0 + 2 \sum_{k=1}^K c_k \cos(k\omega) \quad (7.76)$$

and the phase, assuming a minimum phase system, is given by

$$\tilde{\Phi}(\omega) = -2 \sum_{k=1}^K c_k \sin(k\omega) \quad (7.77)$$

The phase  $\phi_0(t)$  of the first harmonic between frames  $(k-1)$  and  $k$  can be obtained from the instantaneous frequency  $\omega_0(t)$

$$\phi_0(t) = \phi_0((k-1)N) + \int_{(k-1)N}^t \omega_0(t) dt \quad (7.78)$$

if we assume the frequency  $\omega_0(t)$  in that region to vary linearly between frames  $(k-1)$  and  $k$ :

$$\omega_0(t) = \omega_0^{k-1} + \frac{\omega_0^k - \omega_0^{k-1}}{N} t \quad (7.79)$$

and insert Eq. (7.79) into (7.78), evaluating at  $t = kN$ , to obtain

$$\phi_0^k = \phi_0(kN) = \phi_0((k-1)N) + (\omega_0^{k-1} + \omega_0^k)(N/2) \quad (7.80)$$

the phase of the sinusoid at  $\omega_0$  as a function of the fundamental frequencies at frames  $(k-1)$ ,  $k$  and the phase at frame  $(k-1)$ :

$$\phi_l^k = \Phi^k(l\omega_0) + l\phi_0^k \quad (7.81)$$

The phases computed by Eqs. (7.80) and (7.81) are a good approximation in practice for perfectly voiced sounds. For unvoiced sounds, random phases are needed, or else the reconstructed speech sounds buzzy. Voiced fricatives and many voiced sounds have an aspiration component, so that a mixed excitation is needed to represent them. In these cases, the source is split into different frequency bands and each band is classified as either voiced or unvoiced. Sinusoids in voiced bands use the phases described above, whereas sinusoids in unvoiced bands have random phases.

### 7.5.2.3. Parameter Quantization

To quantize the sinusoid amplitudes, we can use an LPC fitting and then quantize the line spectral frequencies. Also we can do a cepstral fit and quantize the cepstral coefficients. To be more effective, a mel scale should be used.

While these approaches help in reducing the number of parameters and in quantizing those parameters, they are not the most effective way of quantizing the sinusoid amplitudes. A technique called *Variable-Dimension Vector Quantization* (VDVQ) [12] has been devised to address this. Each codebook vector  $\mathbf{c}_i$  has a fixed dimension  $N$  determined by the length of the FFT used. The vector of sinusoid amplitudes  $\mathbf{A}$  has a dimension  $l$  that depends on the number of harmonics and thus the pitch of the current frame. To compute the distance between  $\mathbf{A}$  and  $\mathbf{c}_i$ , the codebook vectors are resampled to a size  $l$  and the distance is computed between two vectors of dimension  $l$ . Euclidean distance of the log-amplitudes is often used. In this method, only the distance at the harmonics is evaluated instead of the distance at the points in the envelope that are actually not present in the signal. Also, this technique does

not suffer from inaccuracies of the model used, such as the inability of linear predictive coding to model nasals.

### 7.5.3. Waveform Interpolation

The main idea behind waveform interpolation (WI) [29] is that the pitch pulse changes slowly over time for voiced speech. During voiced segments, the speech signal is nearly periodic. WI coders can operate as low as 2.4 kbps.

Starting at an arbitrary time instant, it is easy to identify a first pitch cycle  $x_1[n]$ , a second  $x_2[n]$ , a third  $x_3[n]$ , and so on. We then express our signal  $x[n]$  as a function of these pitch cycle waveforms  $x_m[n]$

$$x[n] = \sum_{m=-\infty}^{\infty} x_m[n - t_m] \tag{7.82}$$

where  $P_m = t_m - t_{m-1}$  is the pitch period at time  $t_m$  in samples, and the pitch cycle is a windowed version of the input

$$x_m[n] = w_m[n]x[n] \tag{7.83}$$

—for example, with a rectangular window. To transmit the signal in a lossless fashion we need to transmit all pitch waveforms  $x_m[n]$ .

If the signal is perfectly periodic, we need to transmit only one pitch waveform  $x_m[n]$  and the pitch period  $P$ . In practice, voiced signals are not perfectly periodic, so that we need to transmit more than just one pitch waveform. On the other hand, voiced speech is nearly periodic, and consecutive pitch waveforms are very similar. Thus, we probably do not need to transmit all, and we could send every other pitch waveform, for example.

It is convenient to define a two-dimensional surface  $u[n, l]$  (shown in Figure 7.10) such that the pitch waveform  $x_m[n]$  can be obtained as

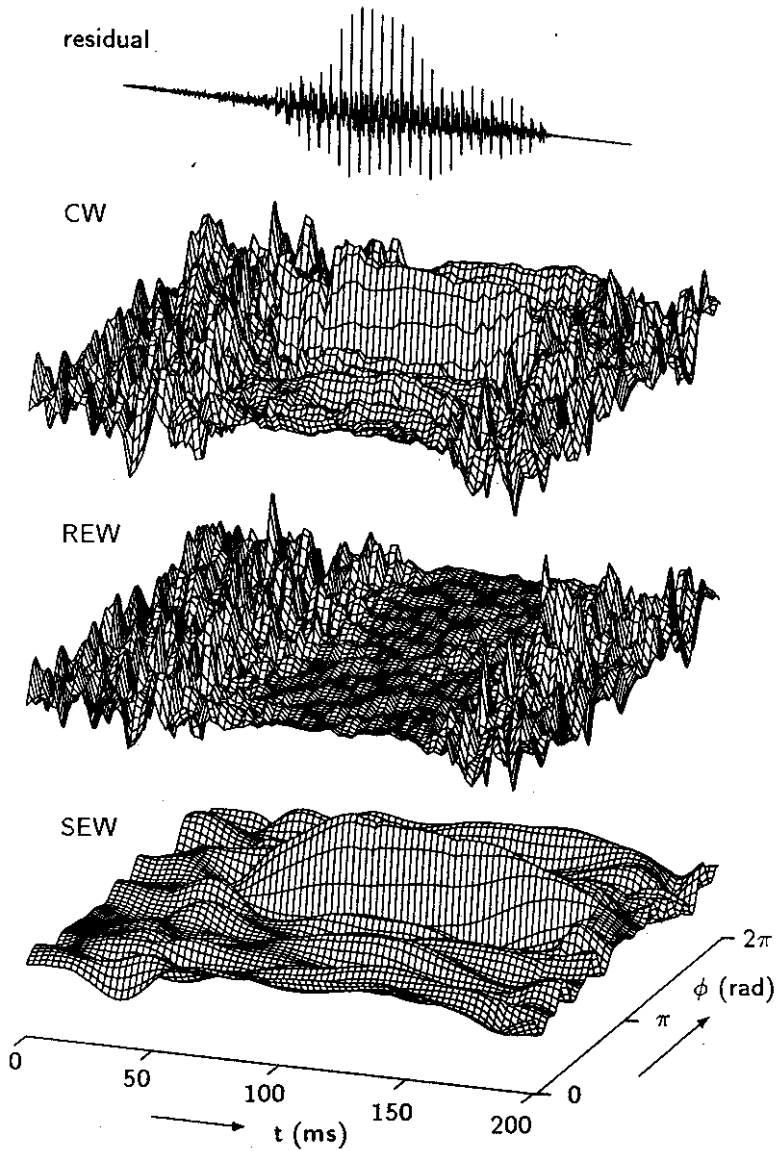
$$x_m[n] = u[n, t_m] \tag{7.84}$$

so that  $u[n, l]$  is defined for  $l = t_m$ , with the remaining points been computed through interpolation. A frequency representation of the pitch cycle can also be used instead of the time pitch cycle.

This surface can then be sampled at regular time intervals  $l = sT$ . It has been shown empirically that transmitting the pitch waveform  $x_s[n]$  about 40 times per second (a 25-ms interval is equivalent to  $T = 200$  samples for an  $F_s = 8000$  Hz sampling rate) is sufficient for voiced speech. The so-called *slowly evolving waveform* (SEW)  $\tilde{u}[n, l]$  can be generated by low-pass filtering  $u[n, l]$  along the  $l$ -axis:

$$x_s[n] = \tilde{u}[n, sT] = \frac{\sum_m h[sT - t_m]u[n, t_m]}{\sum_m h[sT - t_m]} \tag{7.85}$$

where  $h[n]$  is a low-pass filter and  $x_s[n]$  is a sampled version of  $\tilde{u}[n, l]$ .



**Figure 7.10** LP residual signal and its associated surface  $u(t, \phi)$ . In the  $\phi$  axis we have a normalized pitch pulse at every given time  $t$ . Decomposition of the surface into a slowly evolving waveform and a rapidly evolving waveform (After Kleijn [30], reprinted by permission of IEEE).

The decoder has to reconstruct each pitch waveform  $x_m[n]$  from the SEW  $x_s[n]$  by interpolation between adjacent pitch waveforms, and thus the name *waveform interpolation (WI) coding*:

$$\tilde{w}_m[n] = \tilde{u}[n, t_m] = \frac{\sum_s h[t_m - sT] w_s[n]}{\sum_s h[t_m - sT]} \quad (7.86)$$

If the sampling period is larger than the local pitch period ( $T > P_m$ ), perfect reconstruction will not be possible, and there will be some error in the approximation

$$x_m[n] = \tilde{x}_m[n] + \hat{x}_m[n] \quad (7.87)$$

or alternatively in the two-dimensional representation

$$u[n, l] = \tilde{u}[n, l] + \hat{u}[n, l] \quad (7.88)$$

where  $\hat{x}_m[n]$  and  $\hat{u}[n, l]$  represent the *rapidly evolving waveforms (REW)*.

Since this technique can also be applied to unvoiced speech, where the concept of pitch waveform doesn't make sense, the more general term *characteristic waveform* is used instead. For unvoiced speech, an arbitrary *period* of around 100 Hz can be used.

For voiced speech, we expect the rapidly varying waveform  $\hat{u}[n, l]$  in Eq. (7.88) to have much less energy than the slowly evolving waveform  $\tilde{u}[n, l]$ . For unvoiced speech the converse is true:  $\hat{u}[n, l]$  has more energy than  $\tilde{u}[n, l]$ . For voiced fricatives, both components may be comparable and thus we want to transmit both.

In Eqs. (7.85) and (7.86) we need to average characteristic waveforms that have, in general, different lengths. To handle this, all characteristic waveforms are typically normalized in length prior to the averaging operation. This length normalization is done by padding with zeros  $x_m[n]$  to a certain length  $M$ , or truncating  $x_m[n]$  if  $P_m > M$ . Another possible normalization is done via linear resampling. This decomposition is shown in Figure 7.10.

Another representation uses the Fourier transform of  $x_m[n]$ . This case is related to the harmonic model of Section 7.5.2. In the harmonic model, a relatively long window is needed to average the several pitch waveforms within the window, whereas this waveform interpolation method has higher time resolution. In constructing the characteristic waveforms we have implicitly used a rectangular window of length one pitch period, but other windows can be used, such as a Hanning window that covers two pitch periods. This frequency-domain representation offers advantages in coding both the SEW and the REW, because properties of the human auditory system can help reduce the bit rate. This decomposition is often done on the LPC residual signal.

In particular, the REW  $\hat{u}[n, l]$  has the characteristics for noise, and as such only a rough description of its power spectral density is needed. At the decoder, random noise is generated with the transmitted power spectrum. The spectrum of  $\hat{u}[n, l]$  can be vector quantized to as few as eight shapes with little or no degradation.

The SEW  $\tilde{u}[n, l]$  is more important perceptually, and for high quality the whole shape needs to be transmitted. Higher accuracy is desired at lower frequencies so that a perceptual

frequency scale (mel or Bark) is often used. Since the magnitude of  $\tilde{u}[n,l]$  is perceptually more important than the phase, for low bit rates the phase of the SEW is not transmitted. The magnitude spectrum can be quantized with the VDVQ described in Section 7.5.2.3.

To obtain the characteristic waveforms, the pitch needs to be computed. We can find the pitch period such that the energy of the REW is minimized. To do this we use the approaches described in Chapter 6. Figure 7.11 shows a block diagram of the encoder and Figure 7.12 of the decoder.

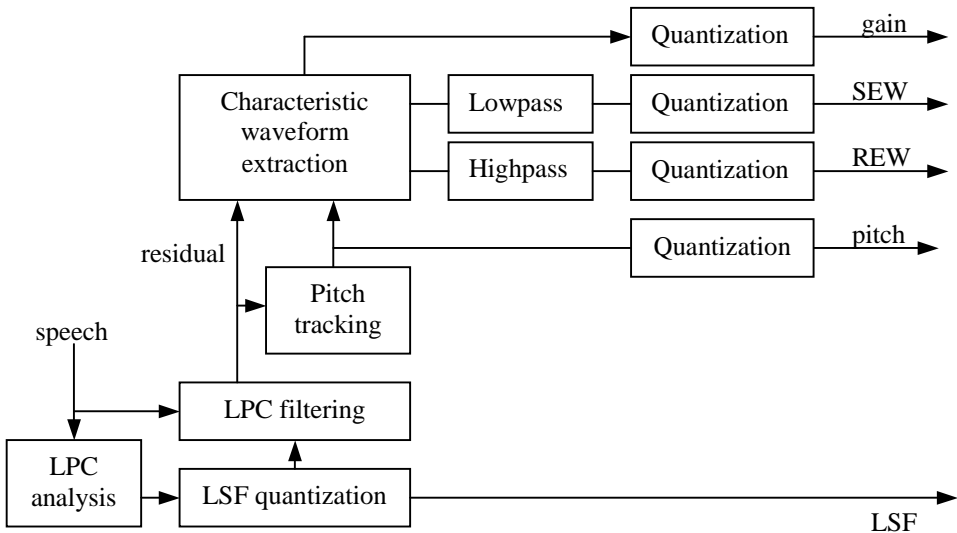


Figure 7.11 Block diagram of the WI encoder.

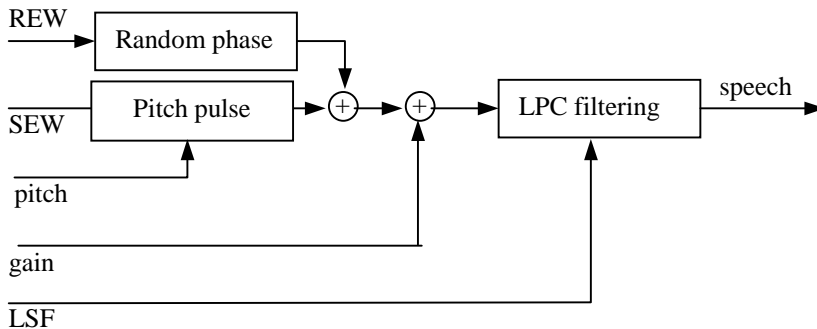


Figure 7.12 Block diagram of the WI decoder.

Parameter estimation using an analysis-by-synthesis framework [21] can yield better results than the open-loop estimation described above.

## 7.6. HISTORICAL PERSPECTIVE AND FURTHER READING

This chapter is only an introduction to speech and audio coding technologies. The reader is referred to [27, 32, 41, 52] for coverage in greater depth. A good source of the history of speech coding can be found in [20].

In 1939, Homer Dudley of AT&T Bell Labs first proposed the channel vocoder [15], the first analysis-by-synthesis system. This vocoder analyzed slowly varying parameters for both the excitation and the spectral envelope. Dudley thought of the advantages of bandwidth compression and information encryption long before the advent of digital communications.

PCM was first conceived in 1937 by Alex Reeves at the Paris Laboratories of AT&T, and it started to be deployed in the United States Public Switched Telephone Network in 1962. The digital compact disc, invented in the late 1960s by James T. Russell and introduced commercially in 1984, also uses PCM as coding standard. The use of  $\mu$ -law encoding was proposed by Smith [51] in 1957, but it wasn't standardized for telephone networks (G.711) until 1972. In 1952, Schouten et al. [47] proposed delta modulation and Cutler [11] invented differential PCM. ADPCM was developed by Barnwell [6] in 1974.

Speech coding underwent a fundamental change with the development of linear predictive coding in the early 1970s. Atal [3] proposed the LPC vocoder in 1971, and then CELP [5] in 1984. The majority of coding standards for speech signals today use a variation on CELP.

Sinusoidal coding [35] and waveform interpolation [29] were developed in 1986 and 1991, respectively, for low-bit-rate telephone speech. Transform coders such as MP3 [23], MPEG II, and Perceptual Audio Coder (PAC) [28] have been used primarily in audio coding for high-fidelity applications.

Recently, researchers have been improving the technology for cellular communications by trading off source coding and channel coding. For poor channels more bits are allocated to channel coding and fewer to source coding to reduce dropped calls. Scalable coders that have different layers with increased level of precision, or bandwidth, are also of great interest.

## REFERENCES

- [1] Adoul, J.P., *et al.*, "Fast CELP Coding Based on Algebraic Codes," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1987, Dallas, TX pp. 1957-1960.
- [2] Atal, B.S., R.V. Cox, and P. Kroon, "Spectral Quantization and Interpolation for CELP Coders," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1989, Glasgow pp. 69-72.
- [3] Atal, B.S. and L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," *Journal of the Acoustical Society of America*, 1971, **50**, pp. 637-655.
- [4] Atal, B.S. and M.R. Schroeder, "Predictive Coding of Speech Signals and Subjective Error Criteria," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1979, **ASSP-27**(3), pp. 247-254.
- [5] Atal, B.S. and M.R. Schroeder, "Stochastic Coding of Speech at Very Low Bit Rates," *Proc. Int. Conf. on Comm.*, 1984, Amsterdam pp. 1610-1613.



- [6] Barnwell, T.P., *et al.*, *Adaptive Differential PCM Speech Transmission*, 1974, Rome Air Development Center.
- [7] Benvenuto, N., G. Bertocci, and W.R. Daumer, "The 32-kbps ADPCM Coding Standard," *AT&T Technical Journal*, 1986, **65**, pp. 12-22.
- [8] Campbell, J.P., T.E. Tremain, and V.C. Welch, "The DoD 4.8 kbps Standard (Proposed Federal Standard 1016)" in *Advances in Speech Coding*, B. Atal, V. Cuperman, and A. Gersho, eds. 1991, pp. 121-133, Kluwer Academic Publishers.
- [9] Chen, J.H., *et al.*, "A Low-Delay CELP Coder for the CCITT 16 kbps Speech Coding Standard," *IEEE Journal on Selected Areas Communications*, 1992, **10**(5), pp. 830-849.
- [10] Chen, J.H. and A. Gersho, "Adaptive Postfiltering for Quality Enhancement of Coded Speech," *IEEE Trans. on Speech and Audio Processing*, 1995, **3**(1), pp. 59-71.
- [11] Cutler, C.C., *Differential Quantization for Communication Signals*, , 1952, US Patent 2,605,361.
- [12] Das, A. and A. Gersho, "Variable Dimension Vector Quantization," *IEEE Signal Processing Letters*, 1996, **3**(7), pp. 200-202.
- [13] Daumer, W.R., *et al.*, "Overview of the 32kbps ADPCM Algorithm," *Proc. IEEE Global Telecomm*, 1984 pp. 774-777.
- [14] DeJaco, P.J.A., W. Gardner, and C. Lee, "QCELP: The North American CDMA Digital Cellular Variable Speech Coding Standard," *Proc. Workshop on Speech Coding for Telecommunications*, 1993, Sainte Adele, Quebec pp. 5-6.
- [15] Dudley, H., "The Vocoder," *Bell Labs Record*, 1939, **17**, pp. 122-126.
- [16] Erdmann, C., *et al.*, "An Adaptive Rate Wideband Speech Codec with Adaptive Gain Re-Quantization," *IEEE Workshop on Speech Coding*, 2000, Delavan, Wisconsin.
- [17] Etemoglu, C.O., V. Cuperman, and A. Gersho, "Speech Coding with an Analysis-by-Synthesis Sinusoidal Model," *Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1371-1374.
- [18] Gerson, I.A. and M.A. Jasiuk, "Vector Sum Excited Linear Prediction (VSELP)" in *Advances in Speech Coding*, B.S. Atal, V. Cuperman, and A. Gersho, eds. 1991, Boston, MA, pp. 69-79, Kluwer Academic Publishers.
- [19] Gerson, I.A. and M.A. Jasiuk., "Techniques for Improving the Performance of CELP-type Speech Coders," *IEEE Journal Selected Areas Communications*, 1991, **10**(5), pp. 858-865.
- [20] Gold, B. and N. Morgan, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, 2000, New York, John Wiley.
- [21] Gottesman, O. and A. Gersho, "High Quality Enhanced Waveform Interpolative Coding at 2.8 kbps," *Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1363-1366.
- [22] Greefkes, J.A., "A Digitally Companded Delta Modulation Modem for Speech Transmission," *Proc. Int. Conf. on Communications*, 1970 pp. 7.33-7.48.
- [23] ISO, *Coding of Moving Pictures and Associated Audio - Audio Part*, 1993, Int. Standards Organization.
- [24] ISO/IEC, *Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbps, Part 3: Audio (MPEG-1)*, 1992, Int. Standards Organization.
- [25] ITU-T, *Methods for Subjective Determination of Transmission Quality*, 1996, Int. Telecommunication Unit.
- [26] Jarvinen, K., *et al.*, "GSM Enhanced Full Rate Speech Codec," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1997, Munich, Germany pp. 771-774.
- [27] Jayant, N.S. and P. Noll, *Digital Coding of Waveforms*, 1984, Upper Saddle River, NJ, Prentice Hall.

- [28] Johnston, J.D., *et al.*, "ATT Perceptual Audio Coding (PAC)" in *Audio Engineering Society (AES) Collected Papers on Digital Audio Bit Rate Reduction*, N. Gilchrist and C. Grewin, eds. 1996, pp. 73-82.
- [29] Kleijn, W.B., "Continuous Representations in Linear Predictive Coding," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, Toronto, Canada pp. 201-204.
- [30] Kleijn, W.B. and J. Haagen, "Transformation and Decomposition of the Speech Signal for Coding," *IEEE Signal Processing Letters*, 1994, **1**, pp. 136-138.
- [31] Kleijn, W.B., D.J. Krasinski, and R.H. Ketchum, "An Efficient Stochastically Excited Linear Predictive Coding Algorithm for High Quality Low Bit Rate Transmission of Speech," *Speech Communication*, 1988, **7**, pp. 305-316.
- [32] Kleijn, W.B. and K.K. Paliwal, *Speech Coding and Synthesis*, 1995, Amsterdam, Netherlands, Elsevier.
- [33] Koishida, K., V. Cuperman, and A. Gersho, "A 16-KBIT/S Bandwidth Scalable Audio Coder Based on the G.729 Standard," *Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1149-1152.
- [34] Li, C. and V. Cuperman, "Analysis-by-Synthesis Multimode Harmonic Speech Coding at 4 kbps," *Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1367-1370.
- [35] McAulay, R.J. and T.F. Quateri, "Speech Analysis/Synthesis Based on a Sinusoidal Representation," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1986, **34**, pp. 744-754.
- [36] McAulay, R.J. and T.F. Quateri, "Sinusoidal Coding" in *Speech Coding and Synthesis*, W.B. Kleijn and K.K. Paliwal, eds. 1995, pp. 121-174, Elsevier.
- [37] McCree, A., "A 14 kbps Wideband Speech Coder with a Parametric Highband Model," *Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1153-1156.
- [38] McCree, A.V. and T.P. Barnwell, "Improving the Performance of a Mixed-Excitation LPC Vocoder in Acoustic Noise," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1992, San Francisco pp. II-137-138.
- [39] McCree, A.V., *et al.*, "A 2.4 kbit/s MELP Coder Candidate for the New U.S. Federal Standard," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1996, Atlanta, GA pp. 200-203.
- [40] Paez, M.D. and T.H. Glisson, "Minimum Squared-Error Quantization in Speech," *IEEE Trans. on Comm*, 1972, **20**, pp. 225-230.
- [41] Painter, T. and A. Spanias, "A Review of Algorithms for Perceptual Coding of Digital Audio Signals," *Proc. Int. Conf. on DSP*, 1997 pp. 179-205.
- [42] Painter, T. and A. Spanias, "Perceptual Coding of Digital Audio," *Proc. of IEEE*, 2000(April), pp. 451-513.
- [43] Paliwal, K.K. and B. Atal, "Efficient Vector Quantization of LPC Parameters at 24 Bits/Frame," *IEEE Trans. on Speech and Audio Processing*, 1993, **1**(1), pp. 3-14.
- [44] Prevez, M.A., H.V. Sorensen, and J.V.D. Spiegel, "An Overview of Sigma-Delta Converters," *IEEE Signal Processing Magazine*, 1996, **13**(1), pp. 61-84.
- [45] Robinson, T., *Simple Lossless and Near-Lossless Waveform Compression*, 1994, Cambridge University Engineering Department.
- [46] Salami, R., C. Laflamme, and B. Bessette, "Description of ITU-T Recommendation G.729 Annex A: Reduced Complexity 8 kbps CS-ACELP Codec," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1997, Munich, Germany pp. 775-778.
- [47] Schouten, J.S., F.E. DeJager, and J.A. Greefkes, *Delta Modulation, a New Modulation System for Telecommunications*, 1952, Phillips, pp. 237-245.

- [48] Shlomot, E., V. Cuperman, and A. Gersho, "Combined Harmonic and Waveform Coding of Speech at Low Bit Rates," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1998, Seattle, WA pp. 585-588.
- [49] Singhal, S. and B.S. Atal, "Improving Performance of Multi-Pulse LPC Coders at Low Bit Rates," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1984, San Diego pp. 1.3.1-1.3.4.
- [50] Skoglund, J., R. Cox, and J. Collura, "A Combined WI and MELP Coder at 5.2KBPS," *Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1387-1390.
- [51] Smith, B., "Instantaneous Companding of Quantized Signals," *Bell Systems Technical Journal*, 1957, **36**(3), pp. 653-709.
- [52] Spanias, A.S., "Speech Coding: A Tutorial Review," *Proc. of the IEEE*, 1994, **82**(10), pp. 1441-1582.
- [53] Stachurski, J. and A. McCree, "A 4 kbps Hybrid MELP/CELP Coder with Alignment Phase Encoding and Zero Phase Equalization," *Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1379-1382.
- [54] Todd, C., "AC-3: Flexible Perceptual Coding for Audio Transmission and Storage," *Audio Engineering Society 96th Convention*, 1994.
- [55] Tremain, T.E., *The Government Standard Linear Predictive Coding Algorithm*, in *Speech Technology Magazine*, 1982. pp. 40-49.
- [56] Vary, P., *et al.*, "A Regular-Pulse Excited Linear Predictive Code," *Speech Communication*, 1988, **7**(2), pp. 209-215.
- [57] Wang, T., *et al.*, "A 1200 BPS Speech Coder Based on MELP," *Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, Turkey pp. 1375-1378.