has a z-transform given by

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

by using the sum of the terms of a geometric sequence and making $N \to \infty$. 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

$$h_4[n] = -a^n u[-n-1] \tag{5.30}$$

has a z-transform given by

$$H_4(z) = -\sum_{n=-\infty}^{-1} a^n z^{-n} = -\sum_{n=1}^{\infty} a^{-n} z^n = 1 - \sum_{n=0}^{\infty} a^{-n} z^n \qquad \text{for } |z| < |a| \tag{5.31}$$

$$= 1 - \frac{1}{1-a^{-1}z} = \frac{-a^{-1}z}{1-a^{-1}z} = \frac{1}{1-az^{-1}}$$

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.

Amazon/VB Assets
Exhibit 1012
Page 239

For example,

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

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

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

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

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

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

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

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

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

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

$$h_4[n]=-3^n u[-n-1] \tag{5.37}$$

and the transform of

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

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

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

so that

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

Amazon/VB Assets
Exhibit 1012
Page 240

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 Transforms

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 (see Table 5.5). 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:

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

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:

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

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

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

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

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

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

Amazon/VB Assets
Exhibit 1012
Page 241

5.2.4.2. Power Spectrum and Parseval's Theorem

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

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

where the superscript asterisk (*) means complex conjugate[5] 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

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

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

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

or alternatively

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

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

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

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.

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 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.

[5] If $z = x + jy = Ae^{j\phi}$, its complex conjugate is defined as $z^* = x - jy = Ae^{-j\phi}$.

Amazon/VB Assets
Exhibit 1012
Page 242

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* $X(e^{-j\omega})=X^*(e^{j\omega})$ $\lvert X(e^{j\omega})\rvert$ is even[6] $\mathrm{Re}\{X(e^{j\omega})\}$ is even $\arg\{X(e^{j\omega})\}$ is odd[7] $\mathrm{Im}\{X(e^{j\omega})\}$ is odd	$X(z^*)=X^*(z)$
	$\mathrm{Even}\{x[n]\}$	$\mathrm{Re}\{X(e^{j\omega})\}$	
	$\mathrm{Odd}\{x[n]\}$	$j\,\mathrm{Im}\{X(e^{j\omega})\}$	
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	$R_{xx}[n]=\sum_{m=-\infty}^{\infty}x[m+n]x^*[m]$	$S_{xx}(\omega)=\lvert X(\omega)\rvert^2$	$X(z)X^*(1/z^*)$

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

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

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

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

[6] A function $f(x)$ is called even if and only if $f(x)=f(-x)$.

[7] A function $f(x)$ is called odd if and only if $f(x)=-f(-x)$.

Amazon/VB Assets
Exhibit 1012
Page 243

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

$$<\varphi_k[n],\varphi_l[n]>=\delta[k-l] \tag{5.52}$$

with the inner product defined as

$$<\varphi_k[n],\varphi_l[n]>=\sum_{n=-\infty}^{\infty}\varphi_k[n]\varphi_l^*[n] \tag{5.53}$$

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

$$X[k]=<x[n],\varphi_k[n]> \tag{5.54}$$

as illustrated in Figure 5.10.

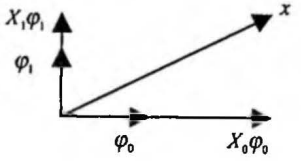


Figure 5.10 Orthonormal expansion of a signal $x[n]$ in a two-dimensional space.

5.3.1. The Discrete Fourier Transform (DFT)

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

$$x_N[n]=x_N[n+N] \tag{5.55}$$

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

$$X_N[k]=\sum_{n=0}^{N-1}x_N[n]e^{-j2\pi nk/N} \qquad 0\le k<N \tag{5.56}$$

$$x_N[n]=\frac{1}{N}\sum_{k=0}^{N-1}X_N[k]e^{j2\pi nk/N} \qquad 0\le n<N \tag{5.57}$$

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).

Amazon/VB Assets
Exhibit 1012
Page 244

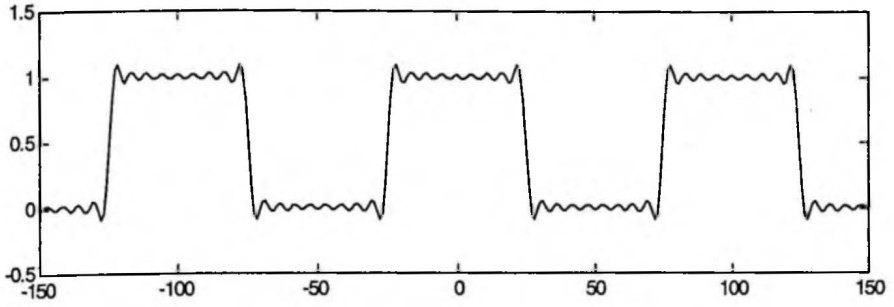


Figure 5.11 Decomposition of a periodic square signal with period 100 samples as a sum of 19 harmonic sinusoids with frequencies $\omega_k = 2\pi k/100$.

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

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.

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

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

which has the following property

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

for all values of $\Delta > 0$.

Amazon/VB Assets
Exhibit 1012
Page 245

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

$$\delta(\omega) = \lim_{\Delta \to 0} d_\Delta(\omega) \tag{5.61}$$

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.

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

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

and similarly

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

so that

$$X(\omega)\delta(\omega - \omega_0) = X(\omega_0)\delta(\omega - \omega_0) \tag{5.64}$$

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

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

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

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

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 ω_0.



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

Amazon/VB Assets
Exhibit 1012
Page 246

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 $x[n]$:

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

so that

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

Amazon/VB Assets
Exhibit 1012
Page 247

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:

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

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.

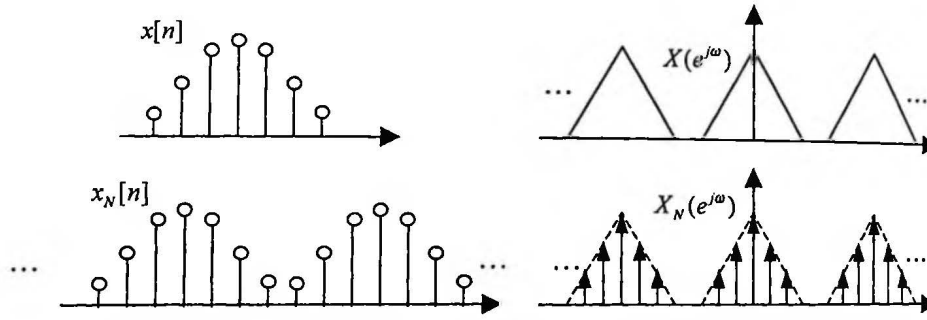


Figure 5.14 Relationships between finite and periodic signals and their Fourier transforms. On one hand, $x[n]$ is a length N discrete signal whose transform $X(e^{j\omega})$ is continuous and periodic with period 2π. On the other hand, $x_N[n]$ is a periodic signal with period N whose transform $X_N(e^{j\omega})$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

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N} = \sum_{n=0}^{N-1} x[n]W_N^{nk} \qquad 0 \le k < N \tag{5.75}$$

Amazon/VB Assets
Exhibit 1012
Page 248

where we have defined for convenience

$$W_N = e^{-j2\pi/N} \tag{5.76}$$

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

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

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 $0 \le k < N/2$, we need to also evaluate them for $N/2 \le k < N$, which is straightforward, since

$$F[k + N/2] = F[k] \tag{5.78}$$

$$G[k + N/2] = G[k] \tag{5.79}$$

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.

Amazon/VB Assets
Exhibit 1012
Page 249

Finally, another possible decomposition is $N = p_1 p_2 \cdots p_L$ with p_l 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.

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

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*.

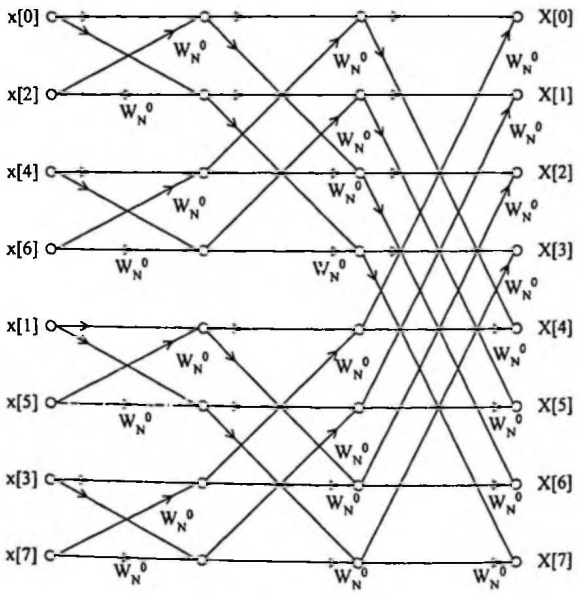


Figure 5.15 Decimation in time radix-2 algorithm for an 8-point FFT.

Amazon/VB Assets
Exhibit 1012
Page 250

```
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;
             xt = y[j];
             y[j] = y[i];
             y[i] = xt;
         }
         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 $n = 2^m$.

Amazon/VB Assets
Exhibit 1012
Page 251

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:

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

take complex conjugate and multiply by j to obtain

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

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

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

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

$$h[n]=f[n]+jg[n]=x[2n]+jx[2n+1] \tag{5.83}$$

whose DFT is

$$H[k]=F[k]+jG[k]=H_R[k]+jH_I[k] \tag{5.84}$$

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

$$H^*[-k]=F^*[-k]-jG^*[-k]=F[k]-jG[k] \tag{5.85}$$

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

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

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

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

Amazon/VB Assets
Exhibit 1012
Page 252

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

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

$$y[n] = x_1[n] \otimes x_2[n] = \sum_{m=0}^{N-1} x_1[m]x_2[n-m] = \sum_{m=<N>} x_1[m]x_2[n-m] \tag{5.89}$$

where $m =< N >$ 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

$$Y[k] = X_1[k]X_2[k] \tag{5.90}$$

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 $0 \le n < N_1$, and $x_2[n] = 0$ outside $0 \le n < N_2$. We know that their regular convolution $y[n] = x_1[n] * x_2[n]$ is zero outside $0 \le N_1 + N_2 - 1$. If we choose an integer N such that $N \ge N_1 + N_2 - 1$, we can define two periodic signals $\tilde{x}_1[n]$ and $\tilde{x}_2[n]$ with period N such that

$$\tilde{x}_1[n] = \begin{cases} x_1[n] & 0 \le n < N_1 \\ 0 & N_1 \le n < N \end{cases} \tag{5.91}$$

$$\tilde{x}_2[n] = \begin{cases} x_2[n] & 0 \le n < N_2 \\ 0 & N_2 \le n < N \end{cases} \tag{5.92}$$

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 $0 \le n < N$, which means that $y[n]$ can be obtained 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 $N \ge 1024$.

Amazon/VB Assets
Exhibit 1012
Page 253

5.3.5. The Discrete Cosine Transform (DCT)

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

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

with its inverse given by

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

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

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

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

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].

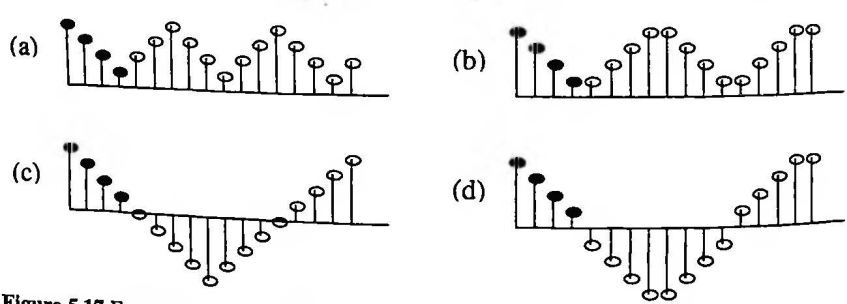


Figure 5.17 Four ways to extend a four-point sequence $x[n]$ to make it both periodic and have even symmetry. The figures in (a), (b), (c) and (d) correspond to the DCT-I, DCT-II, DCT-III and DCT-IV respectively.

Amazon/VB Assets
Exhibit 1012
Page 254

From Eq. (5.95) and (5.96) we see that the DCT-II of a real sequence can be computed with a length-2*N* 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.

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).

5.4.1. The Ideal Low-Pass Filter

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

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

which is the ideal *low-pass* filter because it lets all frequencies below ω_0 pass through unaffected and completely blocks frequencies above ω_0. Using the definition of Fourier transform, we obtain

$$h[n] = \frac{1}{2\pi}\int_{-\omega_0}^{\omega_0} e^{j\omega n} d\omega = \frac{\left(e^{j\omega_0 n} - e^{-j\omega_0 n}\right)}{2\pi jn} = \frac{\sin \omega_0 n}{\pi n} = \left(\frac{\omega_0}{\pi}\right)\text{sinc}(2f_0 n) \tag{5.98}$$

where we have defined the so-called sinc function as

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

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.

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

Amazon/VB Assets
Exhibit 1012
Page 255

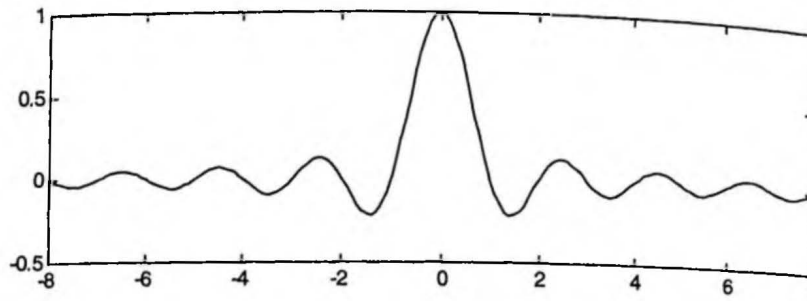


Figure 5.18 A sinc function, which is the impulse response of the ideal low-pass filter with a scale factor.

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

$$h_\pi[n] = u[n] - u[n-N] \tag{5.100}$$

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

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

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

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

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

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

Amazon/VB Assets
Exhibit 1012
Page 256

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):

$$H_\pi(e^{j\omega}) = \frac{1-e^{-j\omega N}}{1-e^{-j\omega}} = \frac{\left(e^{j\omega N/2} - e^{-j\omega N/2}\right)e^{-j\omega N/2}}{\left(e^{j\omega/2} - e^{-j\omega/2}\right)e^{-j\omega/2}}$$
$$= \frac{\sin \omega N/2}{\sin \omega/2} e^{-j\omega(N-1)/2} = A(\omega)e^{-j\omega(N-1)/2} \tag{5.104}$$

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

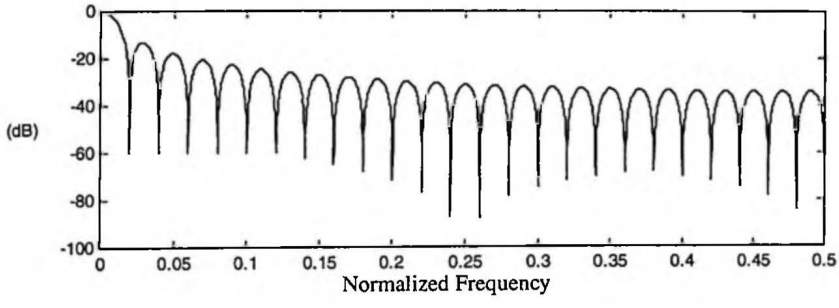


Figure 5.19 Frequency response (magnitude in dB) of the rectangular window with N = 50, which is a digital sinc function.

5.4.2.2. The Generalized Hamming Window

The *generalized Hamming window* is defined as

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

[8] An energy value E is expressed in decibels (dB) as $\overline{E} = 10\log_{10} E$. 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.

Amazon/VB Assets
Exhibit 1012
Page 257

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

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

whose transform is

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

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.

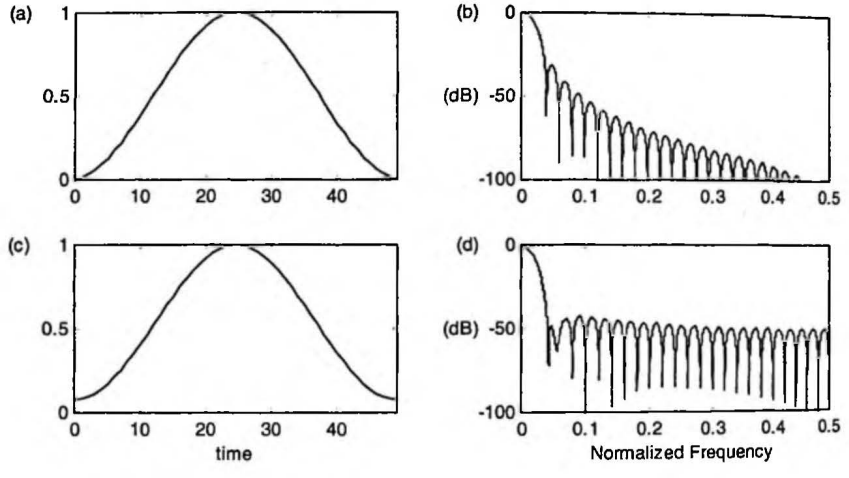


Figure 5.20 (a) Hanning window and (b) the magnitude of its frequency response in dB; (c) Hamming window and (d) the magnitude of its frequency response in dB for $N = 50$.

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.

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:

Amazon/VB Assets
Exhibit 1012
Page 258

$$h[n]=\begin{cases} b_n & 0\le n\le M \\ 0 & otherwise \end{cases} \tag{5.108}$$

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

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

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

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

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

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

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

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

where $A(\omega)$ is a real and even function of ω, since the cosine is an even function, and $A(\omega)$ is a linear combination of cosines. Furthermore, we see that the phase $\arg\left\{H(e^{j\omega})\right\}=L\omega$, which is a linear function of ω, and therefore $h[n]$ is called a *linear-*

Amazon/VB Assets
Exhibit 1012
Page 259

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:

$$h[n] = \pm h^*[M-n] \tag{5.113}$$

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:

$$y[n] = x[n] + \alpha x[n-1] \tag{5.114}$$

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

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

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

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

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

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

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

as shown in Figure 5.21 for various values of α.

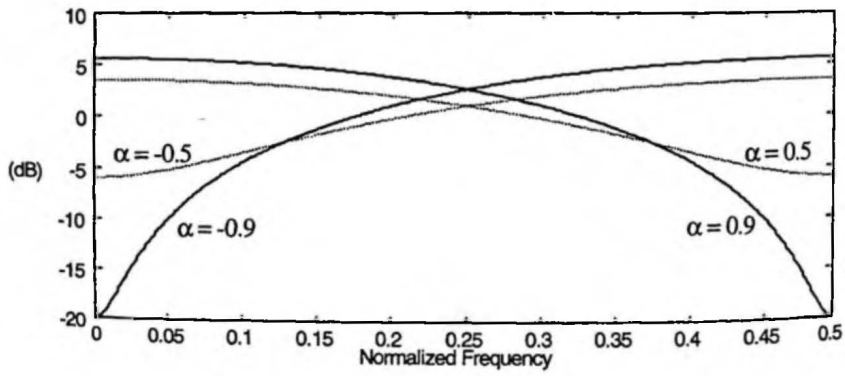


Figure 5.21 Frequency response of the first order FIR filter for various values of α.

Amazon/VB Assets
Exhibit 1012
Page 260

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 α controls the slope of the curve.

5.4.3.3. Window Design FIR Lowpass Filters

The ideal lowpass filter lets all frequencies below ω_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.

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 ω_0 doesn't go away and in fact 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.

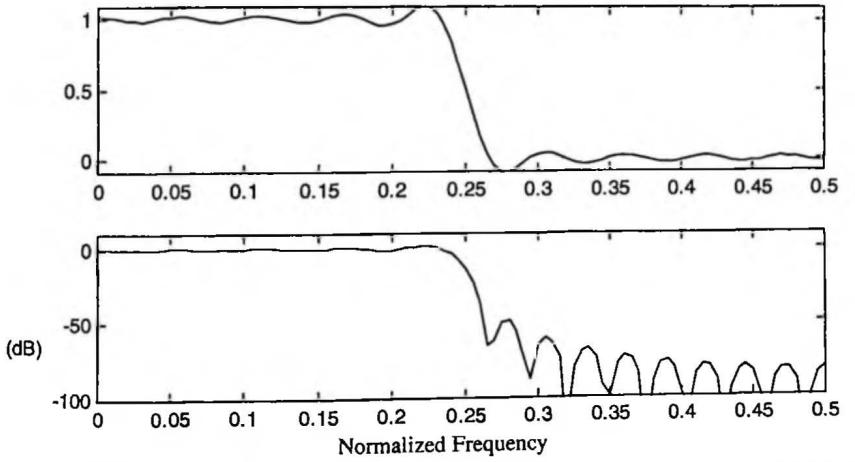


Figure 5.22 Magnitude frequency response of the truncated sinc signal (N=200) for $\omega_0 = \pi/4$. It is an approximation to the ideal low-pass filter, though we see that overshoots are present near the transition. The first graph is linear magnitude and the second is in dB.

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

Amazon/VB Assets
Exhibit 1012
Page 261

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 in Figure 5.22, at the expense of a slower transition.

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 δ_p in the *passband* ($0 \le \omega < \omega_p$), and a minimum attenuation of δ_s in the *stopband* ($\omega_s \le \omega < \pi$), the optimal solution is given by the Parks McClellan algorithm [14].

The transformation

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

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

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

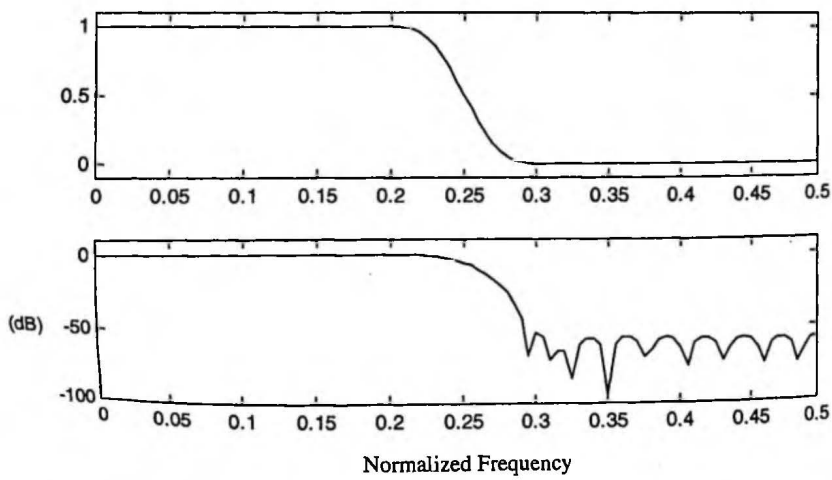


Figure 5.23 Magnitude frequency response of a low-pass filter obtained with the window design method and a Hamming window (N = 200). The first graph is linear magnitude and the second is in dB.

Amazon/VB Assets
Exhibit 1012
Page 262

where $T_n(x)$ is the n^{th}-order *Chebychev* polynomial. The first two Chebychev 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:

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

Using Eq. (5.120), the magnitude response of a linear phase Type-I filter in Eq. (5.112) can be expressed as an L^{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

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

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 ω. 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,

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

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

Amazon/VB Assets
Exhibit 1012
Page 263

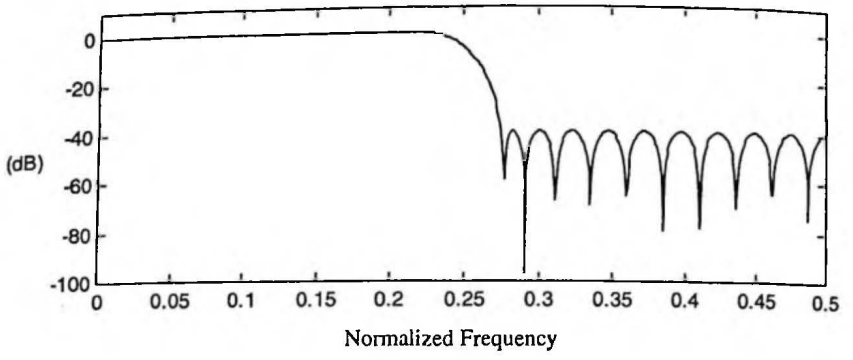


Figure 5.24 Magnitude frequency response of a length-19 lowpass filter designed with the Parks McClellan algorithm.

5.4.4. IIR Filters

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

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

whose z-transform is given by

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

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

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

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

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

Amazon/VB Assets
Exhibit 1012
Page 264

and if $M \geq N$

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

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

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

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

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

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

$$y[n] = Ax[n] + \alpha y[n-1] \tag{5.134}$$

for α real. Its transfer function is given by

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

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:

$$h[n] = A\alpha^n u[n] \tag{5.136}$$

Amazon/VB Assets
Exhibit 1012
Page 265

whose Fourier transform is

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

and magnitude square is given by

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

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

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

If the bandwidth ω_b is also small, we can similarly approximate

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

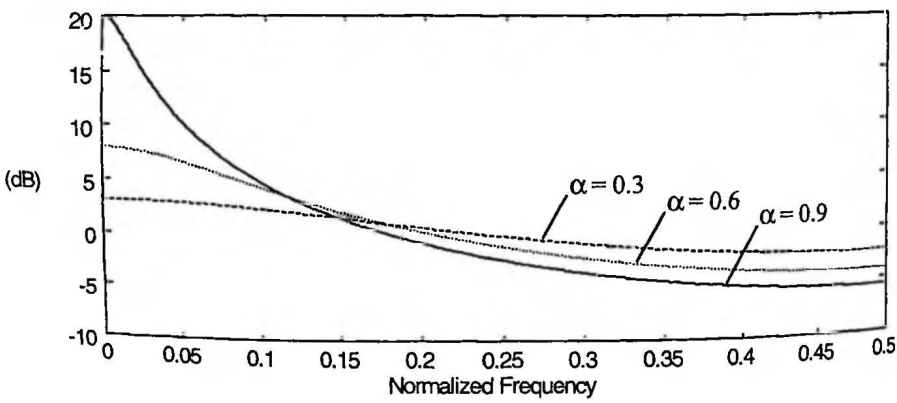


Figure 5.25 Magnitude frequency response of the first-order IIR filter.

Amazon/VB Assets
Exhibit 1012
Page 266

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[9] is smaller than 2% for $b < 0.5$, which corresponds to $0.6 < \alpha < 1$. The relationship with the unnormalized bandwidth B is

$$\alpha = e^{-2\pi B/F_s} \tag{5.141}$$

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

$$y[n] = Ax[n] + a_1 y[n-1] + a_2 y[n-2] \tag{5.142}$$

whose transfer function is given by

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

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

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

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

$$z = e^{-\sigma \pm j\omega_0} \tag{5.145}$$

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

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

In Figure 5.26 we show the magnitude of its Fourier transform for a value of σ and ω_0. We see that the response is centered around ω_0 and is more concentrated for smaller values of

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

Amazon/VB Assets
Exhibit 1012
Page 267

σ. This is a type of *bandpass* filter, since it favors frequencies in a band around ω_0. It is left to the reader as an exercise to show that the bandwidth[10] is approximately 2σ. The smaller the ratio σ/ω_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

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

$$a_2 = -e^{-2\pi B/F_s} \tag{5.148}$$

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.

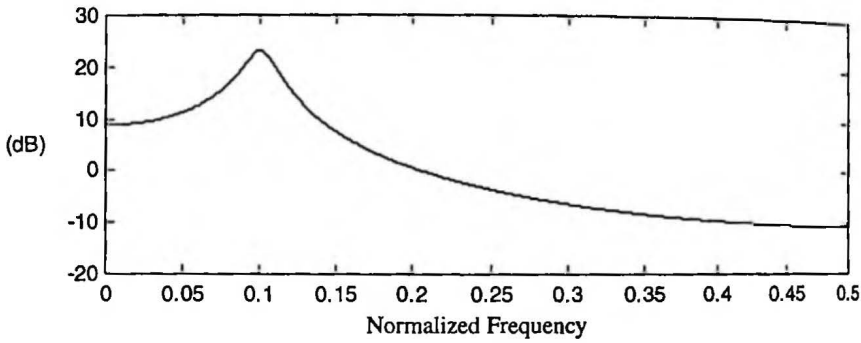


Figure 5.26 Frequency response of the second-order IIR filter for center frequency of $F = 0.1F_s$ and bandwidth $B = 0.01F_s$.

5.5. DIGITAL PROCESSING OF ANALOG SIGNALS

To use 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:

$$x[n] = x(nT) \tag{5.149}$$

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

[10] The bandwidth of a bandpass filter is the region between half maximum magnitude squared values.

Amazon/VB Assets
Exhibit 1012
Page 268

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

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

with its inverse transform being

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

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 $x_p(t)$

$$x_p(t) = x(t)p(t) \tag{5.152}$$

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

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

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

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

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

Amazon/VB Assets
Exhibit 1012
Page 269

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

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

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

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

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

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

From Figure 5.27 it can be seen that if

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

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

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

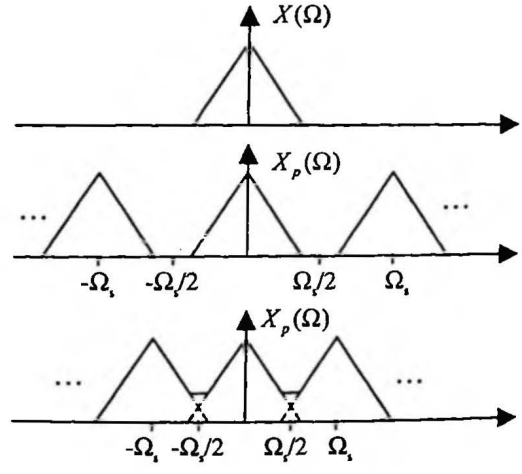


Figure 5.27 $X(\Omega)$, $X_p(\Omega)$ for the case of no aliasing and aliasing.

Amazon/VB Assets
Exhibit 1012
Page 270

where

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

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.

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

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

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

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

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

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

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 is 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

Amazon/VB Assets
Exhibit 1012
Page 271

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

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

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

Amazon/VB Assets
Exhibit 1012
Page 272

$$x_0(t) = h_0(t) * \sum_{n=-\infty}^{\infty} x[n]\delta(t-nT) = \sum_{n=-\infty}^{\infty} x[n]h_0(t-nT) \tag{5.165}$$

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

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

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

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

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.

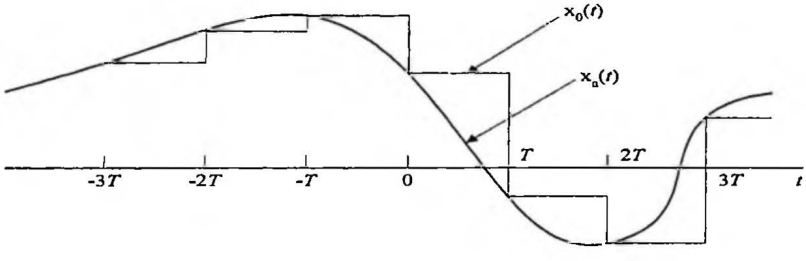


Figure 5.28 Output of a zero-order hold filter.

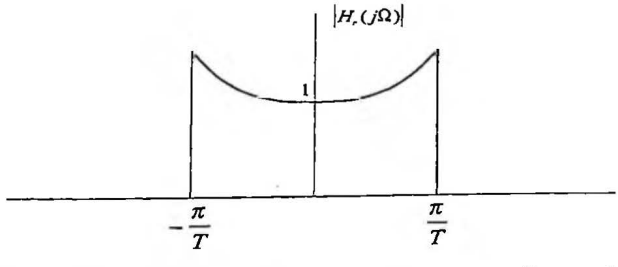


Figure 5.29 Magnitude frequency response of the reconstruction filter used in digital-to-analog converters after a zero-hold filter.

Amazon/VB Assets
Exhibit 1012
Page 273

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.

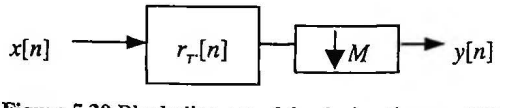


Figure 5.30 Block diagram of the decimation process.

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 $x_l(t)$ at the output of the lowpass filter as

Amazon/VB Assets
Exhibit 1012
Page 274

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

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

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

which can be expressed as

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

where

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

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

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

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

Amazon/VB Assets
Exhibit 1012
Page 275

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

which, inserted into Eq. (5.174), gives

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

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

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

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

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

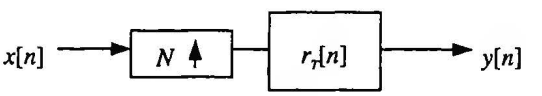


Figure 5.31 Block diagram of the interpolation process.

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

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

where

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

Amazon/VB Assets
Exhibit 1012
Page 276

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.

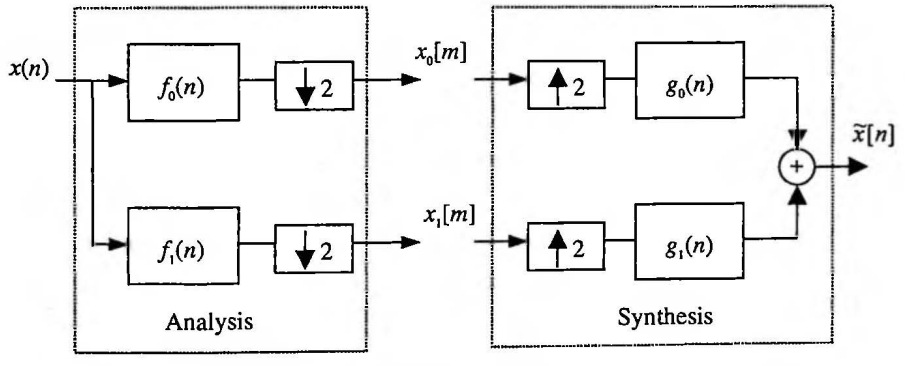


Figure 5.32 Two-band filterbank.

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.

Amazon/VB Assets
Exhibit 1012
Page 277

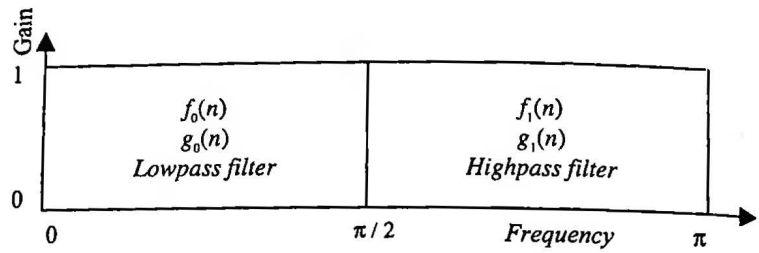


Figure 5.33 Ideal frequency responses of analysis and synthesis filters for the two-band filter-bank.

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 $x[n]$.

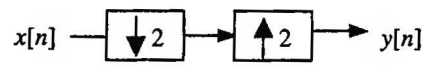


Figure 5.34 Cascade of a downsampler and an upsampler.

The z-transform of the output is given by

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

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

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

Amazon/VB Assets
Exhibit 1012
Page 278

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

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

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 $f_0[n]$:

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

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

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

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. (5.187) 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 $h[n]$

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

Amazon/VB Assets
Exhibit 1012
Page 279

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[11] 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 δ 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 δ to $p[0]$ so that we can guarantee that $P(e^{j\omega}) \geq 0$ for all ω 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-resolution analysis such as that of Figure 5.35, are possible with bands of different bandwidths (see Figure 5.36).

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.

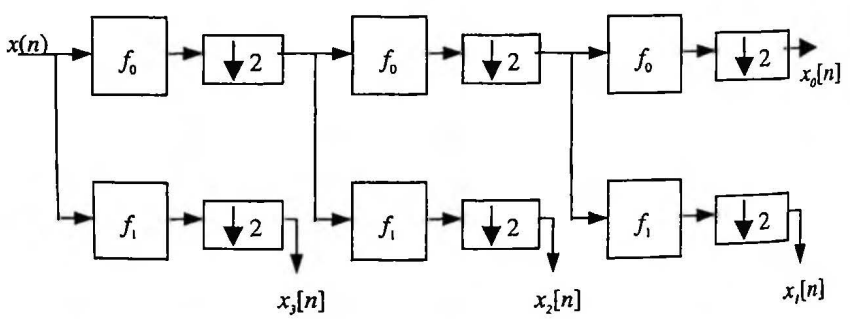


Figure 5.35 Analysis part of a multi-resolution filterbank designed with conjugate quadrature filters. Only $f_0[n]$ needs to be specified.

[11] A filter with N taps is a filter of length N.

Amazon/VB Assets
Exhibit 1012
Page 280

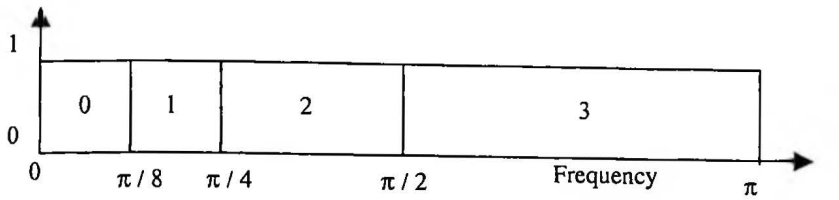


Figure 5.36 Ideal frequency responses of the multi-resolution filterbank of Figure 5.35. Note that $x_0[n]$ and $x_1[n]$ occupy 1/8 of the total bandwidth.

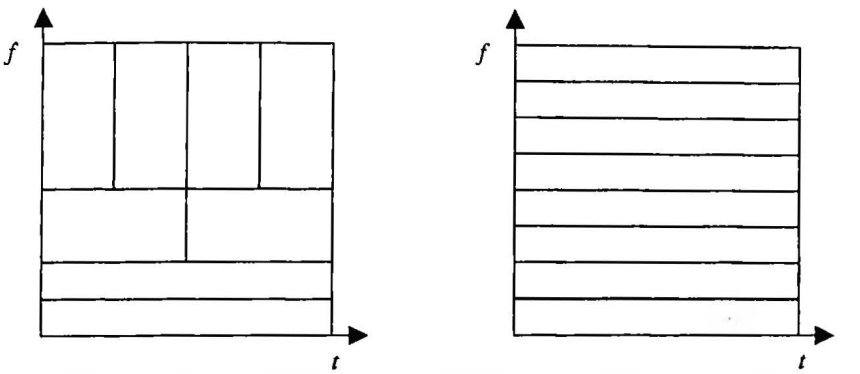


Figure 5.37 Two different time-frequency tilings: the non-uniform filterbank and that obtained through a short-time Fourier transform. Notice that the area of each tile is constant.

5.7.3. The DFT 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 $x_m[n]$

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

where $x_m[n]$ is obtained as

$$x_m[n] = x[n]w_m[n] \tag{5.191}$$

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

$$\sum_{m=-\infty}^{\infty} w_m[n] = 1 \qquad \forall n \tag{5.192}$$

Amazon/VB Assets
Exhibit 1012
Page 281

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

$$w_m[n] = w[n - Mm] \tag{5.193}$$

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

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

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

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

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

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

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

$$X_m[k] = \tilde{X}_k[mM] \tag{5.197}$$

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

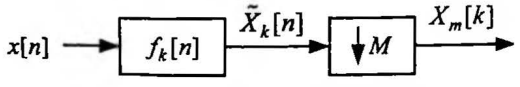


Figure 5.38 Fourier analysis used to build a linear filter.

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

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

where $h[n]$ has been defined as

$$h[n] = \begin{cases} 1/N & 0 \le n < N \\ 0 & \text{otherwise} \end{cases} \tag{5.199}$$

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

Amazon/VB Assets
Exhibit 1012
Page 282

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

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

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

$$g_k[n] = h[n] e^{j\omega_k n} \tag{5.201}$$

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

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

which, inserted into Eq. (5.200), yields

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

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 $\tilde{x}[n] = x[n]$.

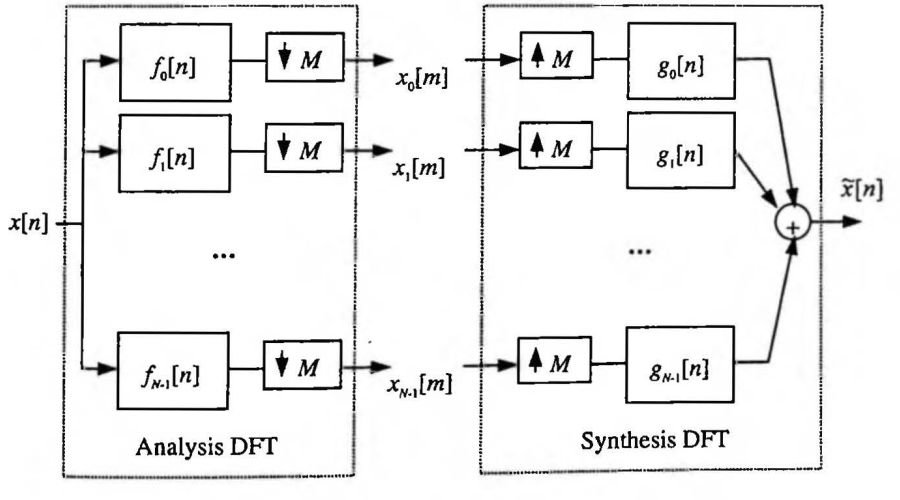


Figure 5.39 A filterbank with N analysis and synthesis filters.

Amazon/VB Assets
Exhibit 1012
Page 283

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

$$w[n] = w_a[n]w_s[n] \tag{5.204}$$

so that the analysis filters are given by

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

and the synthesis filters by

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

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

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

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

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

whose frequency response can be seen in Figure 5.40.

Amazon/VB Assets
Exhibit 1012
Page 284

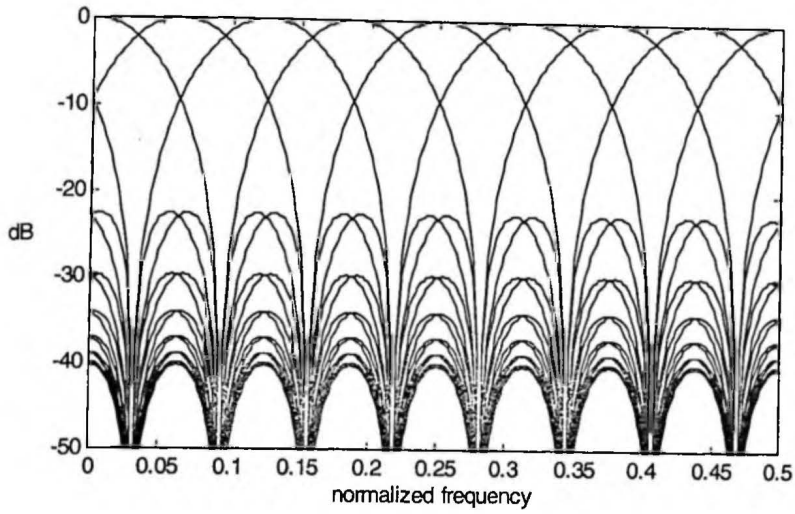


Figure 5.40 Frequency response of the Lapped Orthogonal Transform filterbank.

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

$$h_k^{N/2}[n] = \sqrt{2} h_k^N[2n] \tag{5.209}$$

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.

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):

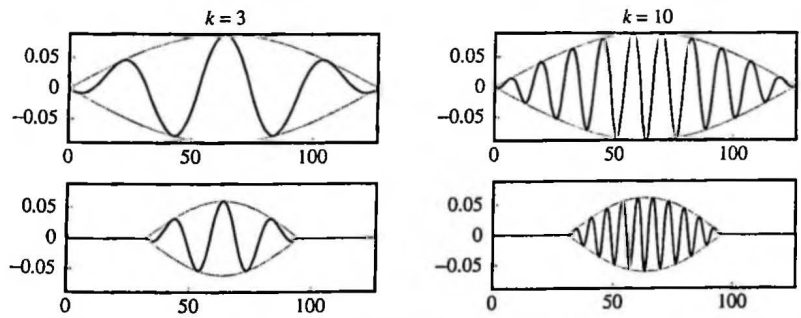


Figure 5.41 Iterations of the wavelet $h_k^N[n]$ for several values of k and N.

Amazon/VB Assets
Exhibit 1012
Page 285

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

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

$$h^2[n] + h^2[n+M] = 1 \tag{5.211}$$

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

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

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 5.1

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

$$\mathbf{x}[n] = \cos[\omega n + \boldsymbol{\varphi}] \tag{5.213}$$

where $\boldsymbol{\varphi}$ 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.

Amazon/VB Assets
Exhibit 1012
Page 286

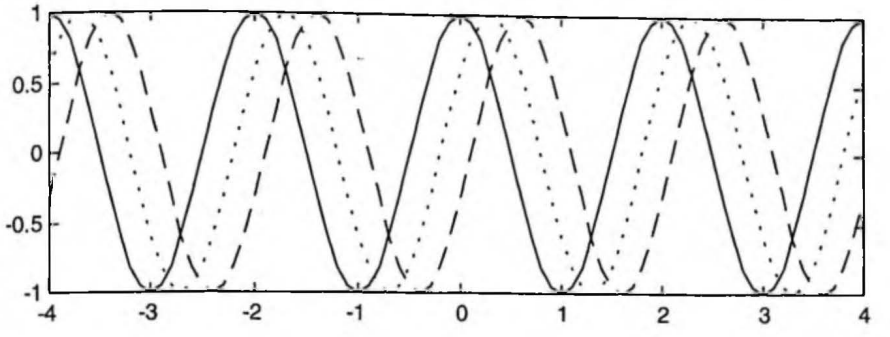


Figure 5.42 Several realizations of a sinusoidal random process with a random phase.

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*

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

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

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

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

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

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

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

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:

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

Amazon/VB Assets
Exhibit 1012
Page 287

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

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

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

Example 5.2

Let's look at the following sinusoidal random process

$$\mathbf{x}[n] = \mathbf{r}\cos[\omega n + \varphi] \tag{5.220}$$

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

$$\mu_x[n] = E\{\mathbf{r}\cos[\omega n + \varphi]\} = E\{\mathbf{r}\}E\{\cos[\omega n + \varphi]\} = 0 \tag{5.221}$$

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

$$E\{\cos[\omega n + \varphi]\} = \int_{-\pi}^{\pi} \cos[\omega n + \varphi]\frac{1}{2\pi}d\varphi = 0 \tag{5.222}$$

Its autocorrelation is given by

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

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

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

which is a consequence of the identity

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

Amazon/VB Assets
Exhibit 1012
Page 288

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

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

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:

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

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

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

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:

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

from which Eq. (5.228) is derived.

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

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

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

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

Amazon/VB Assets
Exhibit 1012
Page 289

Independent processes. If two processes $\mathbf{x}[n]$ and $\mathbf{y}[n]$ are such that the random variables $\mathbf{x}[n_1],\mathbf{x}[n_2],\cdots,\mathbf{x}[n_m]$, and $\mathbf{y}[n_1'],\mathbf{y}[n_2'],\cdots,\mathbf{y}[n_m']$ 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 $\mathbf{x}[n_1],\mathbf{x}[n_2],\cdots,\mathbf{x}[n_m]$ are jointly Gaussian for any m and $n_1,n_2,\cdots,n_m$. 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 5.2.1. 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^{th}-order density of an SSS process must be such that

$$f(x_1,\cdots,x_m;n_1,\cdots,n_m)=f(x_1,\cdots,x_m;n_1+l,\cdots,n_m+l) \tag{5.235}$$

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:

$$f(x,n)=f(x) \tag{5.236}$$

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

$$f(x_1,x_2;n_1,n_2)=f(x_1,x_2;m) \qquad m=n_1-n_2 \tag{5.237}$$

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:

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

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

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

Amazon/VB Assets
Exhibit 1012
Page 290

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

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

is independent of n.

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

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

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

From Eqs. (5.226), (5.238), and (5.239) we can compute the autocovariance as

$$C[m] = R[m] - |\mu|^2 \tag{5.242}$$

and its correlation coefficient as

$$r[m] = C[m]/C[0] \tag{5.243}$$

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

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

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

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*:

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

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

Amazon/VB Assets
Exhibit 1012
Page 291

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

$$\mathbf{x}[n] = a[n] + \mathbf{v}[n] \tag{5.247}$$

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 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

$$E\{\mathbf{x}[n]\} = \mu \tag{5.248}$$

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

$$\lim_{N\to\infty} \mu_N = \mu \tag{5.249}$$

where μ_N is the time average

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

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

$$E\{\mu_N\} = \mu \tag{5.251}$$

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

$$\lim_{N\to\infty} \sigma_N^2 = 0 \tag{5.252}$$

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

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

Amazon/VB Assets
Exhibit 1012
Page 292

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

$$\lim_{m\to\infty} C_{xx}[m] = 0 \tag{5.254}$$

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

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

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

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

The cross-correlation between input and output is given by

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

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.

Amazon/VB Assets
Exhibit 1012
Page 293

5.8.4. Power Spectral Density

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

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

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:

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

Amazon/VB Assets
Exhibit 1012
Page 294

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:

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

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

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

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

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

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

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

The energy of the output process is

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

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

$$0 \le \lim_{c \to 0} \frac{1}{2c} \int_{\omega_0 - c}^{\omega_0 + c} S_{xx}(\omega) d\omega = S_{xx}(\omega_0) \tag{5.270}$$

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 ω. Equation (5.269) also explains the name power spectral density, because $S_{xx}(\omega)$ represents the density of power at any given frequency ω.

5.8.5. Noise

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

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

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

Amazon/VB Assets
Exhibit 1012
Page 295

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

$$C_{xx}[n] = R_{xx}[n] = q\delta[n] \tag{5.272}$$

which has a flat power spectral density

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

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 ω. 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 Schafer [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 17th 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 18th 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),

Amazon/VB Assets
Exhibit 1012
Page 296

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 19th 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*™, 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 20th 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.

Amazon/VB Assets
Exhibit 1012
Page 297

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 in 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. 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.

Amazon/VB Assets
Exhibit 1012
Page 298

[9] Oliver, B.M., J.R. Pierce, and C. Shannon, "The Philosophy of PCM," *Proc. Institute of Radio Engineers*, 1948, **36**, pp. 1324-1331.

[10] Oppenheim, A.V., R.W. Schafer, and J.R. Buck, *Discrete-Time Signal Processing*, 2nd ed., 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*, 3rd ed., 1991, New York, McGraw-Hill.

[13] Parks, T.W. and C.S. Burrus, *Digital Filter Design*, 1987, New York, NY, 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 Conversión 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, CA, pp. 27.1.1-27.1.4.

Amazon/VB Assets
Exhibit 1012
Page 299

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 $x[n]$.

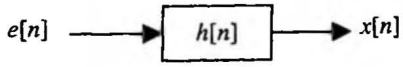


Figure 6.1 Basic source-filter model for speech signals.

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

Amazon/VB Assets
Exhibit 1012
Page 301

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.

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.

Amazon/VB Assets
Exhibit 1012
Page 302

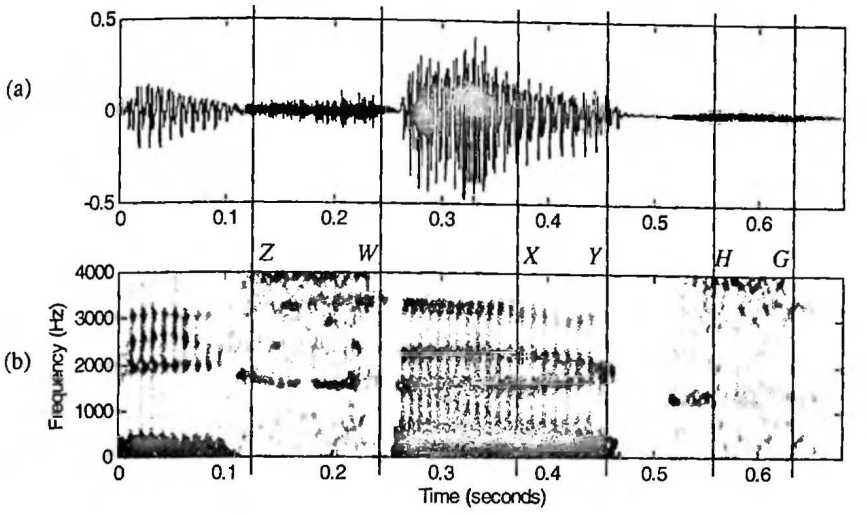


Figure 6.2 (a) Waveform with (b) its corresponding wideband spectrogram. Darker areas mean higher energy for that time and frequency. Note the vertical lines spaced by pitch periods.

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

$$x_m[n] = x[n]w_m[n] \tag{6.1}$$

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:

$$w_m[n] = w[m-n] \tag{6.2}$$

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

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

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

Amazon/VB Assets
Exhibit 1012
Page 303

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

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

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

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

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

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

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 λ such that

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

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

Amazon/VB Assets
Exhibit 1012
Page 304

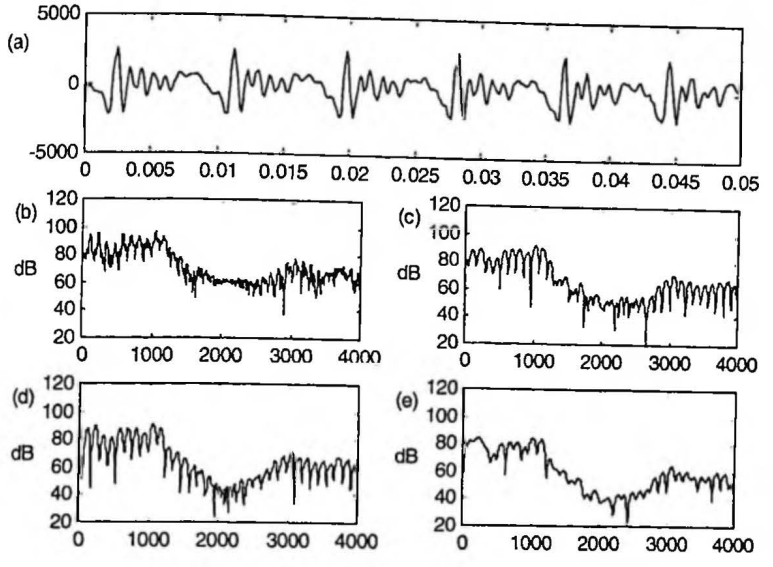


Figure 6.3 Short-time spectrum of male voiced speech (vowel /ah/ with local pitch of 110Hz): (a) time signal, spectra obtained with (b) 30 ms rectangular window and (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15 ms Hamming window. The window lobes are not visible in (e), since the window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

$F_0 = 50\,\text{Hz}$ 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 40 ms, 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^{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.

Amazon/VB Assets
Exhibit 1012
Page 305

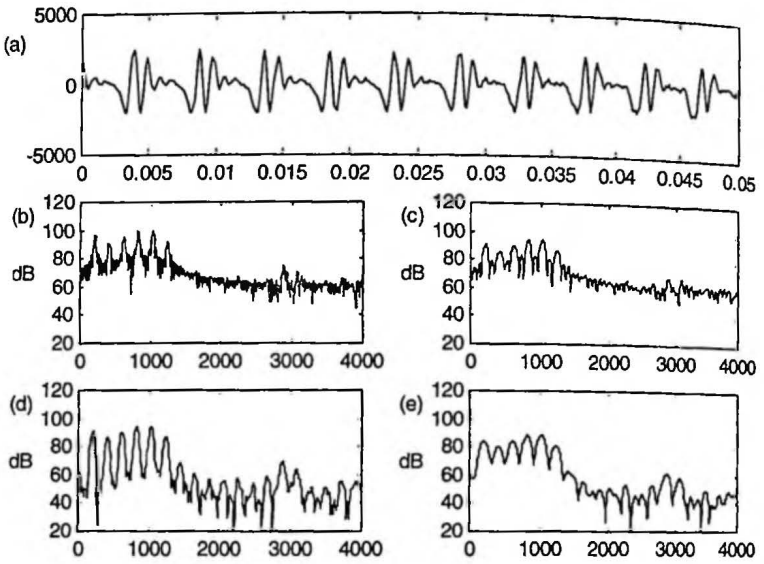


Figure 6.4 Short-time spectrum of female voiced speech (vowel /aa/ with local pitch of 200Hz): (a) time signal, spectra obtained with (b) 30 ms rectangular window and (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15 ms Hamming window. In all cases the window lobes are visible, since the 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

Amazon/VB Assets
Exhibit 1012
Page 306

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.

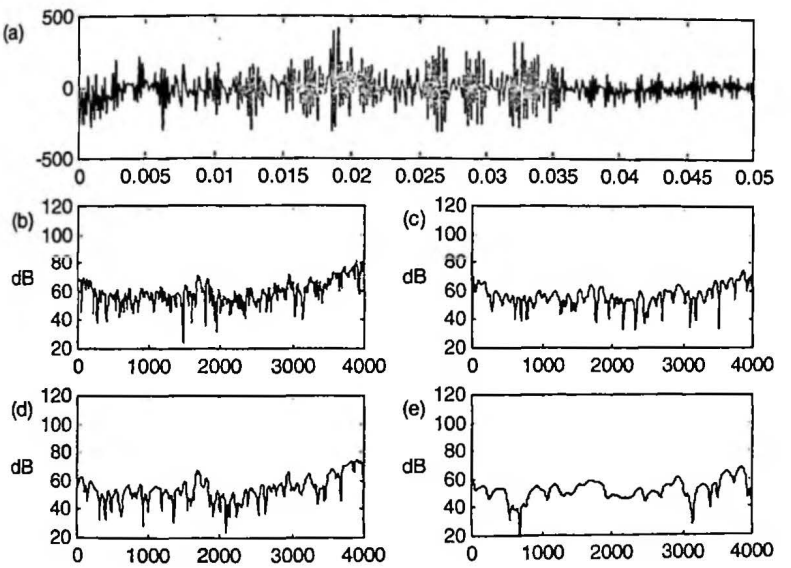


Figure 6.5 Short-time spectrum of unvoiced speech: (a) time signal, (b) 30 ms rectangular window, (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15 ms Hamming window.

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

$$\log|X[k]|^2 = \log\left(X_r^2[k] + X_i^2[k]\right) \tag{6.8}$$

Amazon/VB Assets
Exhibit 1012
Page 307

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.

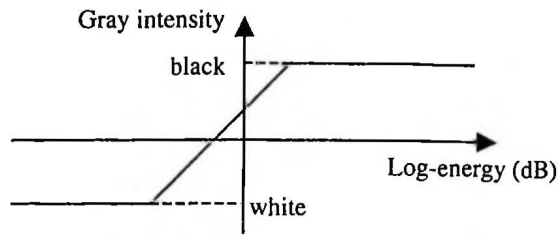


Figure 6.6 Conversion between log-energy values (in the x-axis) and gray scale (in the y-axis). Larger log-energies correspond to a darker gray color. There is a linear region for which more log-energy corresponds to darker gray, but there is saturation at both ends. Typically there is 40 to 60 dB between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

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.

Amazon/VB Assets
Exhibit 1012
Page 308

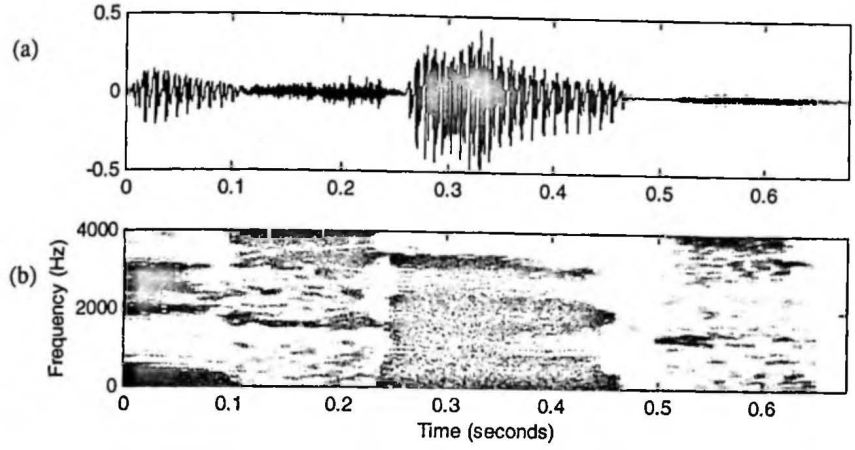


Figure 6.7 Waveform (a) with its corresponding narrowband spectrogram (b). Darker areas mean higher energy for that time and frequency. The harmonics can be seen as horizontal lines spaced by fundamental frequency. The corresponding wideband spectrogram can be seen in Figure 6.2.

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

Amazon/VB Assets
Exhibit 1012
Page 309

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.

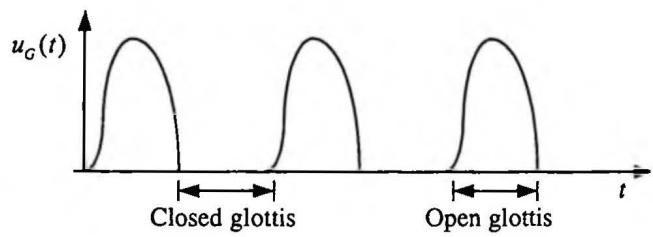


Figure 6.8 Glottal excitation: volume velocity is zero during the closed-phase, during which the vocal cords are closed.

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

Amazon/VB Assets
Exhibit 1012
Page 310

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:

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

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, ρ is the density of air in the tube, c is the velocity of sound, and A is the cross-sectional area of the tube.

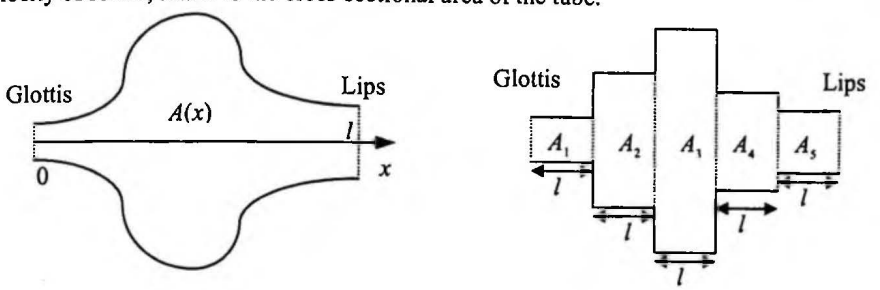


Figure 6.9 Approximation of a tube with continuously varying area $A(x)$ as a concatenation of 5 lossless acoustic tubes.

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

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

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}: $0 \le x \le l$. The reader can prove that this is indeed the solution by substituting Eq. (6.10) into (6.9).

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

$$r_k = \frac{A_{k+1} - A_k}{A_{k+1} + A_k} \tag{6.11}$$

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

$$-1 \le r_k \le 1 \tag{6.12}$$

Amazon/VB Assets
Exhibit 1012
Page 311

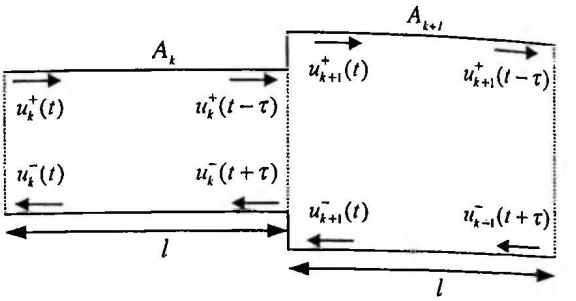


Figure 6.10 Junction between two lossless tubes.

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:

$$V(z)=\frac{U_L(z)}{U_G(z)}=\frac{0.5z^{-N/2}(1+r_G)\prod_{k=1}^{N}(1+r_k)}{\begin{bmatrix}1 & -r_G\end{bmatrix}\left(\prod_{k=1}^{N}\begin{bmatrix}1 & -r_k\\ -r_k z^{-1} & z^{-1}\end{bmatrix}\right)\begin{bmatrix}1\\0\end{bmatrix}} \tag{6.13}$$

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

$$U_L(z)=P_L(z)/Z_L \tag{6.14}$$

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 $F_s=1/T$:

$$N=\frac{2LF_s}{c} \tag{6.15}$$

Amazon/VB Assets
Exhibit 1012
Page 312

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

$$Z_L(z) \approx R_0(1 - z^{-1}) \tag{6.16}$$

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.

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 3 kHz. 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.

Amazon/VB Assets
Exhibit 1012
Page 313

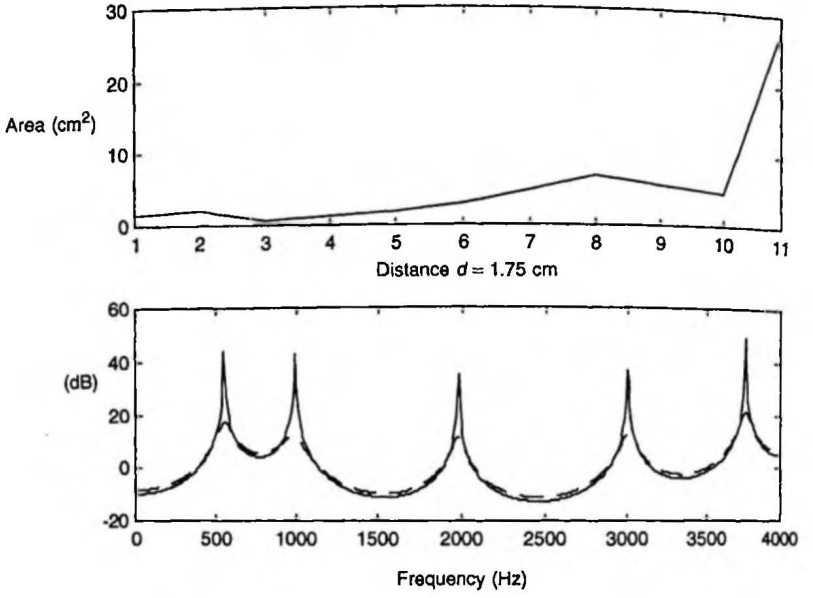


Figure 6.11 Area function and frequency response for vowel /a/ and its approximation as a concatenation of 10 lossless tubes. A reflection coefficient at the load of $k = 0.72$ (dotted line) is displayed. For comparison, the case of $k = 1.0$ (solid line) is also shown.

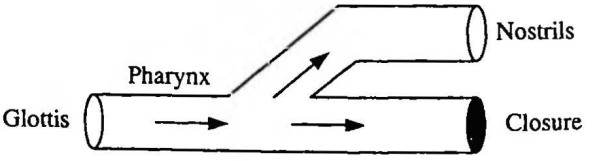


Figure 6.12 Coupling of the nasal cavity with the oral cavity.

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

$$P_L(z) = U_L(z)Z_L(z) = U_G(z)V(z)Z_L(z) \tag{6.17}$$

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.

Amazon/VB Assets
Exhibit 1012
Page 314

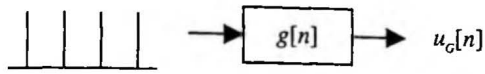


Figure 6.13 Model of the glottal excitation for voiced sounds.

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.

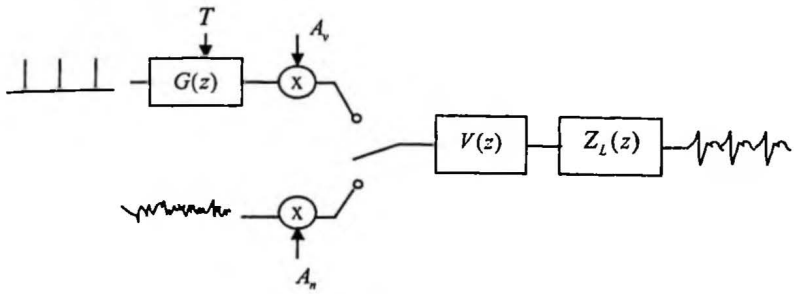


Figure 6.14 General discrete-time model of speech production. The excitation can be either an impulse train with period T and amplitude A_v driving a filter $G(z)$ or random noise with amplitude A_n.

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.

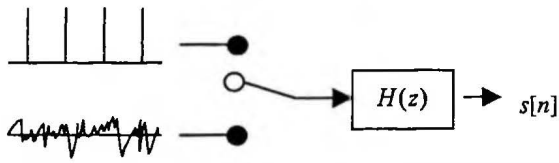


Figure 6.15 Source-filter model for voiced and unvoiced speech.

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).

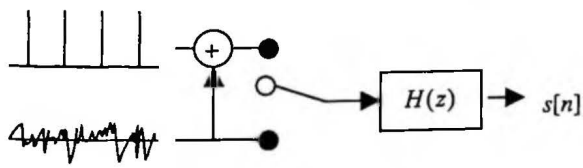


Figure 6.16 A mixed excitation source-filter model of speech.

Amazon/VB Assets
Exhibit 1012
Page 315

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 in Chapter 5, we state that the z-transform of $x[n] = a^n u[n]$ is

$$X(z) = \sum_{n=0}^{\infty} a^n z^{-n} = \frac{1}{1 - az^{-1}} \quad \text{for} \quad |a| < |z| \tag{6.18}$$

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.

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

$$H(z) = \frac{X(z)}{E(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} = \frac{1}{A(z)} \tag{6.19}$$

where p is the order of the LPC analysis. The *inverse filter* $A(z)$ is defined as

$$A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k} \tag{6.20}$$

Taking inverse z-transforms in Eq. (6.19) results in

$$x[n] = \sum_{k=1}^{p} a_k x[n-k] + e[n] \tag{6.21}$$

Linear predictive coding gets its name from the fact that it predicts the current sample as a linear combination of its past p samples:

$$\tilde{x}[n] = \sum_{k=1}^{p} a_k x[n-k] \tag{6.22}$$

Amazon/VB Assets
Exhibit 1012
Page 316

The prediction error when using this approximation is

$$e[n] = x[n] - \tilde{x}[n] = x[n] - \sum_{k=1}^{p} a_k x[n-k] \tag{6.23}$$

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:

$$x_m[n] = x[m+n] \tag{6.24}$$

We define the short-term prediction error for that segment as

$$E_m = \sum_n e_m^2[n] = \sum_n \left(x_m[n] - \tilde{x}_m[n]\right)^2 = \sum_n \left(x_m[n] - \sum_{j=1}^{p} a_j x_m[n-j] \right)^2 \tag{6.25}$$

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:

$$< \mathbf{e}_m, \mathbf{x}_m^i >= \sum_n e_m[n] x_m[n-i] = 0 \qquad 1 \le i \le p \tag{6.26}$$

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

$$\sum_n x_m[n-i] x_m[n] = \sum_{j=1}^{p} a_j \sum_n x_m[n-i] x_m[n-j] \qquad i = 1, 2, \ldots, p \tag{6.27}$$

For convenience, we can define the correlation coefficients as

$$\phi_m[i,j] = \sum_n x_m[n-i] x_m[n-j] \tag{6.28}$$

so that Eqs. (6.27) and (6.28) can be combined to obtain the so-called *Yule-Walker* equations:

Amazon/VB Assets
Exhibit 1012
Page 317

$$\sum_{j=1}^{p} a_j \phi_m[i,j] = \phi_m[i,0] \qquad i = 1,2,\ldots,p \tag{6.29}$$

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:

$$E_m = \sum_n x_m^2[n] - \sum_{j=1}^{p} a_j \sum_n x_m[n] x_m[n-j] = \phi[0,0] - \sum_{j=1}^{p} a_j \phi[0,j] \tag{6.30}$$

It is convenient to define a normalized prediction error $u[n]$ with unity energy

$$\sum_n u_m^2[n] = 1 \tag{6.31}$$

and a gain G, such that

$$e_m[n] = G u_m[n] \tag{6.32}$$

The gain G can be computed from the short-term prediction error

$$E_m = \sum_n e_m^2[n] = G^2 \sum_n u_m^2[n] = G^2 \tag{6.33}$$

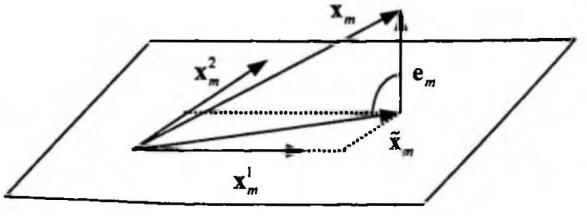


Figure 6.17 The orthogonality principle. The prediction error is orthogonal to the past samples.

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.

Amazon/VB Assets
Exhibit 1012
Page 318

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:

$$E_m = \sum_{n=0}^{N-1} e_m^2[n] \tag{6.34}$$

so that $\phi_m[i,j]$ in Eq. (6.28) becomes

$$\phi_m[i,j] = \sum_{n=0}^{N-1} x_m[n-i]x_m[n-j] = \sum_{n=-i}^{N-1-i} x_m[n]x_m[n+i-j] = \phi_m[j,i] \tag{6.35}$$

and Eq. (6.29) becomes

$$\begin{pmatrix} \phi_m[1,1] & \phi_m[1,2] & \phi_m[1,3] & \cdots & \phi_m[1,p] \\ \phi_m[2,1] & \phi_m[2,2] & \phi_m[2,3] & \cdots & \phi_m[2,p] \\ \phi_m[3,1] & \phi_m[3,2] & \phi_m[3,3] & \cdots & \phi_m[3,p] \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \phi_m[p,1] & \phi_m[p,2] & \phi_m[p,3] & \cdots & \phi_m[p,p] \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \cdots \\ a_p \end{pmatrix} = \begin{pmatrix} \phi_m[1,0] \\ \phi_m[2,0] \\ \phi_m[3,0] \\ \cdots \\ \phi_m[p,0] \end{pmatrix} \tag{6.36}$$

which can be expressed as the following matrix equation

$$\mathbf{\Phi a} = \psi \tag{6.37}$$

where the matrix $\mathbf{\Phi}$ 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 $\mathbf{\Phi}$ is expressed as

$$\mathbf{\Phi = VDV'} \tag{6.38}$$

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 $\mathbf{\Phi}$ can be expressed as

$$\phi[i,j] = \sum_{k=1}^{j} V_{ik} d_k V_{jk} \qquad 1 \le j < i \tag{6.39}$$

or alternatively

$$V_{ij} d_j = \phi[i,j] - \sum_{k=1}^{j-1} V_{ik} d_k V_{jk} \qquad 1 \le j < i \tag{6.40}$$

and for the diagonal elements

Amazon/VB Assets
Exhibit 1012
Page 319

$$\phi[i,i] = \sum_{k=1}^{i} V_{ik} d_k V_{ik} \tag{6.41}$$

or alternatively

$$d_i = \phi[i,i] - \sum_{k=1}^{i-1} V_{ik}^2 d_k , \qquad i \geq 2 \tag{6.42}$$

with

$$d_1 = \phi[1,1] \tag{6.43}$$

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

$$\mathbf{VY} = \psi \tag{6.44}$$

with

$$\mathbf{Y} = \mathbf{DV'a} \tag{6.45}$$

or alternatively

$$\mathbf{V'a} = \mathbf{D^{-1}Y} \tag{6.46}$$

Therefore, given matrix $\mathbf{V}$ and Eq. (6.44), $\mathbf{Y}$ can be solved recursively as

$$Y_i = \psi_i - \sum_{j=1}^{i-1} V_{ij} Y_j , \qquad 2 \leq i \leq p \tag{6.47}$$

with the initial condition

$$Y_1 = \psi_1 \tag{6.48}$$

Having determined $\mathbf{Y}$, Eq. (6.46) can be solved recursively in a similar way

$$a_i = Y_i / d_i - \sum_{j=i+1}^{p} V_{ji} a_j , \qquad 1 \leq i < p \tag{6.49}$$

with the initial condition

$$a_p = Y_p / d_p \tag{6.50}$$

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.

Amazon/VB Assets
Exhibit 1012
Page 320

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 $0 \le n < N$:

$$x_m[n] = x[m+n]w[n] \tag{6.51}$$

with $w[n]$ being a window (such as a Hamming window) which is 0 outside the interval $0 \le n < N$. With this assumption, the corresponding prediction error $e_m[n]$ is non-zero over the interval $0 \le n < N+p$, and, therefore, the total prediction error takes on the value

$$E_m = \sum_{n=0}^{N+p-1} e_m^2[n] \tag{6.52}$$

With this range, Eq. (6.28) can be expressed as

$$\phi_m[i,j] = \sum_{n=0}^{N+p-1} x_m[n-i]x_m[n-j] = \sum_{n=0}^{N-1-(i-j)} x_m[n]x_m[n+i-j] \tag{6.53}$$

or alternatively

$$\phi_m[i,j] = R_m[i-j] \tag{6.54}$$

with $R_m[k]$ being the autocorrelation sequence of $x_m[n]$:

$$R_m[k] = \sum_{n=0}^{N-1-k} x_m[n]x_m[n+k] \tag{6.55}$$

Combining Eqs. (6.54) and (6.29), we obtain

$$\sum_{j=1}^{p} a_j R_m[|\, i-j\,|] = R_m[i] \tag{6.56}$$

which corresponds to the following matrix equation

$$\begin{pmatrix} R_m[0] & R_m[1] & R_m[2] & \cdots & R_m[p-1] \\ R_m[1] & R_m[0] & R_m[1] & \cdots & R_m[p-2] \\ R_m[2] & R_m[1] & R_m[0] & \cdots & R_m[p-3] \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ R_m[p-1] & R_m[p-2] & R_m[p-3] & \cdots & R_m[0] \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \cdots \\ a_p \end{pmatrix} = \begin{pmatrix} R_m[1] \\ R_m[2] \\ R_m[3] \\ \cdots \\ R_m[p] \end{pmatrix} \tag{6.57}$$

Amazon/VB Assets
Exhibit 1012
Page 321

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

$$E^0 = R[0] \tag{6.58}$$

2. Iteration. For $i = 1, \cdots, p$ do the following recursion:

$$k_i = \left(R[i] - \sum_{j=1}^{i-1} a_j^{i-1} R[i-j] \right) / E^{i-1} \tag{6.59}$$

$$a_i^i = k_i \tag{6.60}$$

$$a_j^i = a_j^{i-1} - k_i a_{i-j}^{i-1}, \qquad 1 \le j < i \tag{6.61}$$

$$E^i = (1 - k_i^2) E^{i-1} \tag{6.62}$$

3. Final solution:

$$a_j = a_j^p \qquad 1 \le j \le p \tag{6.63}$$

where the coefficients k_i, called *reflection coefficients*, are bounded between –1 and 1 (see Section 6.3.2.1.3). 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

$$r[j] = R[j] / R[0] \tag{6.64}$$

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

$$V^i = \frac{E^i}{R[0]} = 1 - \sum_{j=1}^{i} a_j r[j] \tag{6.65}$$

The normalized prediction error is, using Eqs. (6.62) and (6.65),

$$V^p = \prod_{i=1}^{p} (1 - k_i^2) \tag{6.66}$$

Amazon/VB Assets
Exhibit 1012
Page 322

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

$$e^i[n] = x[n] - \sum_{k=1}^{i} a_k^i x[n-k] \tag{6.67}$$

whose z-transform is given by

$$E^i(z) = A^i(z)X(z) \tag{6.68}$$

with $A^i(z)$ being defined by

$$A^i(z) = 1 - \sum_{k=1}^{i} a_k^i z^{-k} \tag{6.69}$$

which, combined with Eq. (6.61), results in the following recursion:

$$A^i(z) = A^{i-1}(z) - k_i z^{-i} A^{i-1}(z^{-1}) \tag{6.70}$$

Similarly, we can define the so-called *backward prediction error* as

$$b^i[n] = x[n-i] - \sum_{k=1}^{i} a_k^i x[n+k-i] \tag{6.71}$$

whose z-transform is

$$B^i(z) = z^{-i} A^i(z^{-1})X(z) \tag{6.72}$$

Now combining Eqs. (6.68), (6.70), and (6.72), we obtain

$$E^i(z) = A^{i-1}(z)X(z) - k_i z^{-i} A^{i-1}(z^{-1})X(z) = E^{i-1}(z) - k_i z^{-1} B^{i-1}(z) \tag{6.73}$$

whose inverse z-transform is given by

$$e^i[n] = e^{i-1}[n] - k_i b^{i-1}[n-1] \tag{6.74}$$

Also, substituting Eq. (6.70) into (6.72) and using Eq. (6.68), we obtain

$$B^i(z) = z^{-1} B^{i-1}(z) - k_i E^{i-1}(z) \tag{6.75}$$

whose inverse z-transform is given by

$$b^i[n] = b^{i-1}[n-1] - k_i e^{i-1}[n] \tag{6.76}$$

Amazon/VB Assets
Exhibit 1012
Page 323

Equations (6.74) and (6.76) define the forward and backward prediction error sequences for an i^{th}-order predictor in terms of the corresponding forward and backward prediction errors of an $(i-1)^{th}$-order predictor. We initialize the recursive algorithm by noting that the 0^{th}-order predictor is equivalent to using no predictor at all; thus

$$e^0[n] = b^0[n] = x[n] \tag{6.77}$$

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, hence its name.

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

$$E^i = \sum_{n=0}^{N-1} \left(e^i[n]\right)^2 \tag{6.78}$$

by substituting Eq. (6.74) in (6.78), taking the derivative with respect to k_i, and equating to 0:

$$k_i = \frac{\sum_{n=0}^{N-1} e^{i-1}[n] b^{i-1}[n-1]}{\sum_{n=0}^{N-1} \left(b^{i-1}[n-1]\right)^2} \tag{6.79}$$

Using Eqs. (6.67) and (6.71), it can be shown that

$$\sum_{n=0}^{N-1} \left(e^{i-1}[n]\right)^2 = \sum_{n=0}^{N-1} \left(b^{i-1}[n-1]\right)^2 \tag{6.80}$$

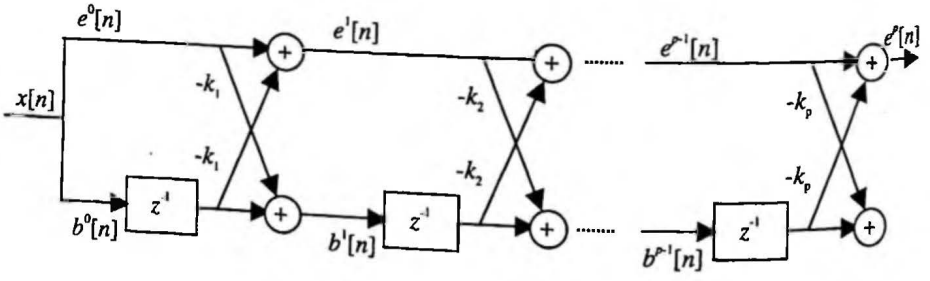


Figure 6.18 Block diagram of the lattice filter.

Amazon/VB Assets
Exhibit 1012
Page 324

since minimization of both yields identical Yule-Walker equations. Thus Eq. (6.79) can be alternatively expressed as

$$k_l = \frac{\sum_{n=0}^{N-1} e^{l-1}[n]b^{l-1}[n-1]}{\sqrt{\sum_{n=0}^{N-1}\left(e^{l-1}[n]\right)^2 \sum_{n=0}^{N-1}\left(b^{l-1}[n-1]\right)^2}} = \frac{<\mathbf{e}^{l-1}, \mathbf{b}^{l-1}>}{\left|\mathbf{e}^{l-1}\right|\left|\mathbf{b}^{l-1}\right|} \tag{6.81}$$

where we have defined the vectors $\mathbf{e}^l = \left(e^l[0]\cdots e^l[N-1]\right)$ and $\mathbf{b}^l = \left(b^l[0]\cdots b^l[N-1]\right)$. The inner product of two vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$<\mathbf{x},\mathbf{y}>=\sum_{n=0}^{N-1} x[n]y[n] \tag{6.82}$$

and its norm as

$$|\mathbf{x}|^2 =<\mathbf{x},\mathbf{x}>=\sum_{n=0}^{N-1} x^2[n] \tag{6.83}$$

Equation (6.81) has the form of a normalized cross-correlation function, and, therefore, the reflection coefficients are also called *partial correlation coefficients* (PARCOR). As with any normalized cross-correlation function, the k_l coefficients are bounded by

$$-1 \le k_l \le 1 \tag{6.84}$$

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_l are also called *reflection coefficients*.

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 be sufficient to cause k_l 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.

Amazon/VB Assets
Exhibit 1012
Page 325

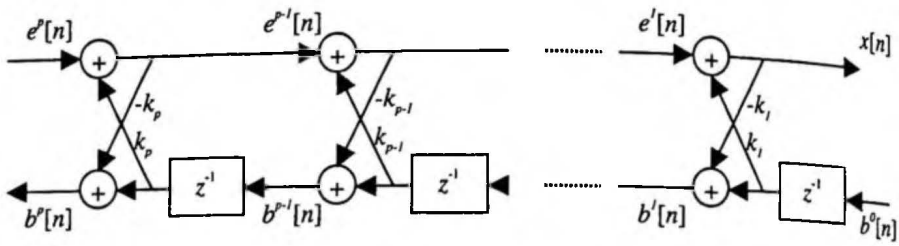


Figure 6.19 Inverse lattice filter used to generate the speech signal, given its residual.

6.3.3. Spectral Analysis via LPC

Let's now analyze the frequency-domain behavior of the LPC analysis by evaluating

$$H(e^{j\omega}) = \frac{G}{1-\sum_{k=1}^{p} a_k e^{-j\omega k}} = \frac{G}{A(e^{j\omega})} \tag{6.85}$$

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).

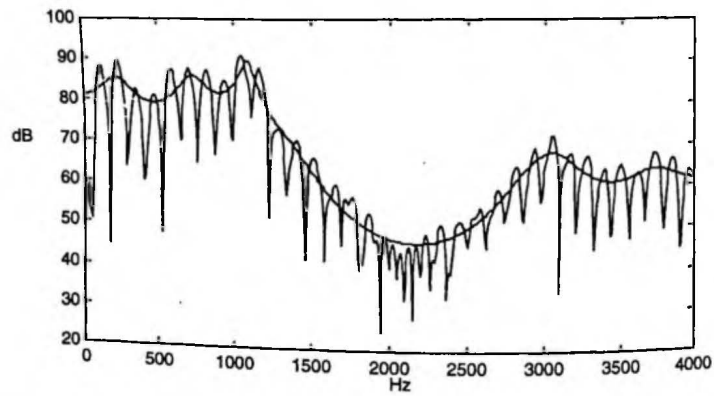


Figure 6.20 LPC spectrum of the */ah/* phoneme in the word *lives* of Figure 6.3. Used here are a 30-ms Hamming window and the autocorrelation method with $p = 14$. The short-time spectrum is also shown.

Amazon/VB Assets
Exhibit 1012
Page 326

For the autocorrelation method, the squared error of Eq. (6.52) can be expressed, using Eq. (6.85) and Parseval's theorem, as

$$E_m = \frac{G^2}{2\pi}\int_{-\pi}^{\pi}\frac{|X_m(e^{j\omega})|^2}{|H(e^{j\omega})|^2}d\omega \tag{6.86}$$

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.

The prediction order is not known for arbitrary speech, so we need to set it to balance spectral detail with estimation errors.

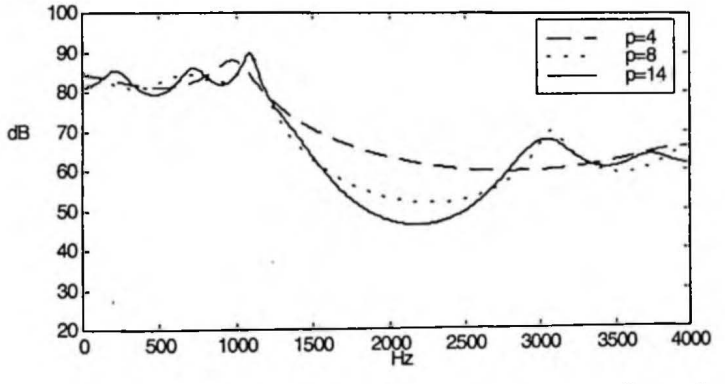


Figure 6.21 LPC spectra of Figure 6.20 for various values of the predictor order p.

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 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 we

Amazon/VB Assets
Exhibit 1012
Page 327

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 robotic, 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.

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.

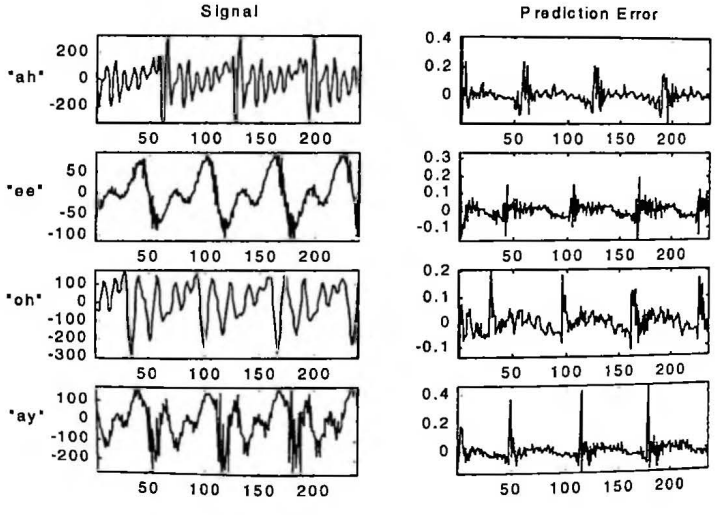


Figure 6.22 LPC prediction error signals for several vowels.

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

Amazon/VB Assets
Exhibit 1012
Page 328

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.

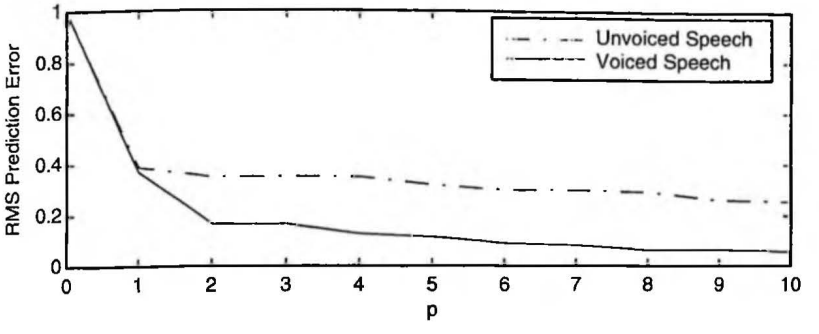


Figure 6.23 Variation of the normalized prediction error with the number of prediction coefficients p for the voiced segment of Figure 6.3 and the unvoiced speech of Figure 6.5. The autocorrelation method was used with a 30 ms Hamming window, and a sampling rate of 8 kHz.

6.3.5. Equivalent Representations

There are a number of alternate useful representations of the predictor coefficients. The most important are the line spectrum frequencies, 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

$$P(z) = A(z) + z^{-(p+1)} A(z^{-1}) \tag{6.87}$$

$$Q(z) = A(z) - z^{-(p+1)} A(z^{-1}) \tag{6.88}$$

To gain insight on these roots, look at a second-order predictor filter with a pair of complex roots:

$$A(z) = 1 - a_1 z^{-1} - a_2 z^{-2} = 1 - 2\rho_0 \cos(2\pi f_0) z^{-1} + \rho_0^2 z^{-2} \tag{6.89}$$

Amazon/VB Assets
Exhibit 1012
Page 329