CHAPTER 2

BACKGROUND

This chapter will review a number of important mathematical results and establish the notation to be used throughout the book. Material is drawn from diverse areas, some of which are not well known and thus extensive references are provided with each section.

2.1 VECTORS AND MATRICES

Many of the techniques presented in this text use numerical methods derived from linear algebra. This section presents a brief overview of some important results to be used in the chapters that follow. For a more thorough treatment, the reader should consult the canonical reference by Golub and Van Loan (1989).

Vectors will be represented using lowercase, boldface letters. The same letter, in lightface type, will be used for the elements of the vector, subscripted with its position in the vector. Thus, an M element vector will be written as follows:

$$\boldsymbol{\theta} = \left[\theta_1 \; \theta_2 \; \dots \; \theta_M\right]^T$$

where the superscript T denotes transposition (i.e., $\boldsymbol{\theta}$ is a column vector).

Bold uppercase letters will denote matrices. Depending on the context, individual elements of a matrix will be referenced by placing the row and column number in parentheses or by using a lowercase, lightface letter with a double subscript. Thus, both $\mathbf{R}(i,j)$ and $r_{i,j}$ represent the element in the ith row and jth column of the matrix $\mathbf{R}$. MATLAB's convention will be used for referencing rows and columns of a matrix, so that $\mathbf{R}(:, j)$ will be the jth column of $\mathbf{R}$.

Two matrix decompositions will be used extensively: the QR factorization and the singular value decomposition (SVD). The QR factorization takes a matrix $\mathbf{X}$ (i.e., either

Identification of Nonlinear Physiological Systems, By David T. Westwick and Robert E. Kearney
ISBN 0-471-27456-9 © 2003 Institute of Electrical and Electronics Engineers

square or tall) and constructs

$$\mathbf{X} = \mathbf{QR}$$

where $\mathbf{R}$ is upper triangular, so that $r_{i,j} = 0$ for $i > j$, and $\mathbf{Q}$ is an orthogonal matrix, $\mathbf{Q}^\mathbf{T}\mathbf{Q} = \mathbf{I}$. Note that the columns of $\mathbf{Q}$ are said to be orthonormal (i.e., orthogonal and normalized); however, the matrix itself is said to be orthogonal.

The *singular value decomposition* (SVD) takes a matrix, $\mathbf{X}$, of any size and shape and replaces it with the product,

$$\mathbf{X} = \mathbf{USV}^\mathbf{T}$$

where $\mathbf{S}$ is a diagonal matrix with non-negative diagonal elements. By convention, the elements of $\mathbf{S}$, the *singular values*, are arranged in decreasing order along the diagonal. Thus, $s_{1,1} \geq s_{2,2} \geq \cdots \geq s_{n,n}$. $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices containing the left and right singular vectors, respectively.

Throughout the book, the approximate computational cost of algorithms will be given in *flops*. A flop, or floating point operation, represents either the addition or the multiplication of two floating point numbers. Note that a flop was originally defined as a floating point multiplication optionally followed by a floating point addition. Thus, the first edition of Golub and Van Loan (1989) used this earlier definition, whereas subsequent editions used the later definition. Indeed, they Golub and Van Loan (1989) noted that supercomputer manufacturers were delighted by this change in terminology, since it gave the appearance of an overnight doubling in performance. In any case, the precise definition is not important to this text, so long as it is applied consistently; the *flop count* will be used only as a first-order measurement of the computation requirements of algorithms.

2.2 GAUSSIAN RANDOM VARIABLES

Gaussian random variables, and signals derived from them, will play a central role in much of the development to follow. A Gaussian random variable has the probability density (Bendat and Piersol, 1986; Papoulis, 1984)

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.1}$$

that is completely defined by two parameters: the mean, μ, and variance, σ^2. By convention, the mean, variance, and other statistical moments, are denoted by symbols subscripted by the signal they describe. Thus,

$$\begin{aligned} \mu_x &= E[x] \\ \sigma_x^2 &= E\left[(x-\mu_x)^2\right] \end{aligned}$$

Figure 2.1 shows a single realization of a Gaussian signal, the theoretical probability density function (PDF) of the process that generated the signal, and an estimate of the PDF derived from the single realization.

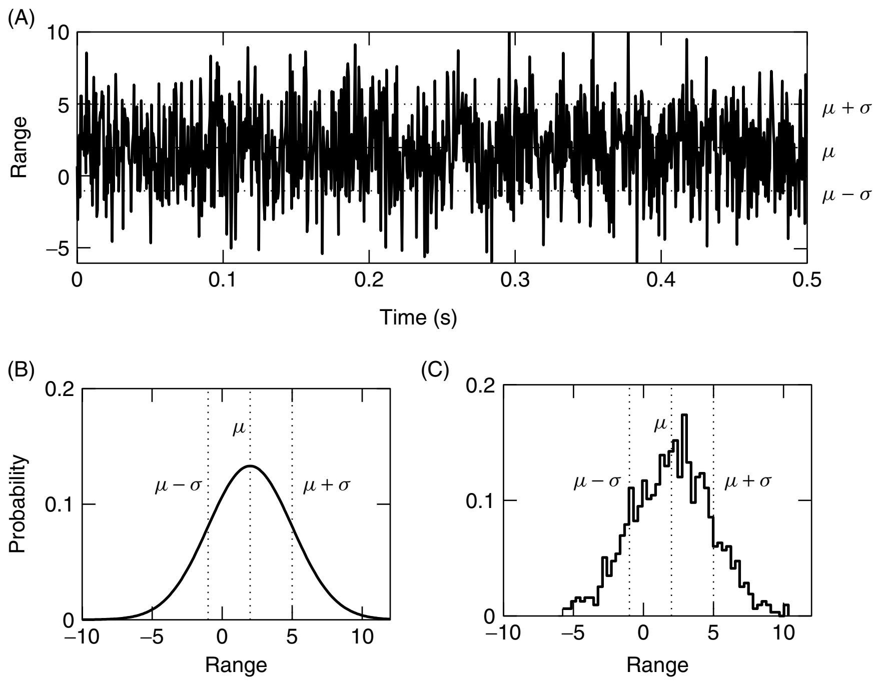

Figure 2.1 Gaussian random variable, x, with mean $\mu_x = 2$ and standard deviation $\sigma_x = 3$. (A) One realization of the random process: $x(t)$. (B) The ideal probability distribution of the sequence x. (C) An estimate of the PDF obtained from the realization shown in A.

2.2.1 Products of Gaussian Variables

The probability density, $f(x)$ defined in equation (2.1), of a Gaussian random variable is completely specified by its first two moments; all higher-order moments can be derived from them. Furthermore, filtering a Gaussian signal with a linear system produces an output that is also Gaussian (Bendat and Piersol, 1986; Papoulis, 1984). Consequently, only the first two moments are required for linear systems analysis. However, the situation is more complex for nonlinear systems since the response to a Gaussian input is not Gaussian; rather it is the sum of products of one or more Gaussian signals. Consequently, the expected value of the product of n zero-mean, Gaussian random variables, $E[x_1x_2 \dots x_n]$, is an important higher-order statistic that will be used frequently in subsequent chapters.

The expected value of the product of an odd number of zero-mean Gaussian signals will be zero. However, the result is more complex for the product of an even number of Gaussian variables. The expected value may be obtained as follows (Bendat and Piersol, 1986):

1. Form every distinct pair of random variables and compute the expected value of each pair. For example, when n is 4, the expected values would be

$$E[x_1x_2],\ E[x_1x_3],\ E[x_1x_4],\ E[x_2x_3],\ E[x_2x_4],\ E[x_3x_4]$$

2. Form all possible distinct combinations, each involving $n/2$ of these pairs, such that each variable is included exactly once in each combination. For $n = 4$, there

are three combinations

$$(x_1x_2)(x_3x_4),\ (x_1x_3)(x_2x_4),\ (x_1x_4)(x_2x_3)$$

3. For each combination, compute the product of the expected values of each pair, determined from step 1. Sum the results of all combinations to get the expected value of the overall product. Thus, for $n = 4$, the combinations are

$$E[x_1x_2x_3x_4] = E[x_1x_2]E[x_3x_4] + E[x_1x_3]E[x_2x_4] + E[x_1x_4]E[x_2x_3] \tag{2.2}$$

Similarly, when n is 6:

$$E[x_1x_2x_3x_4x_5x_6] = E[x_1x_2]E[x_3x_4]E[x_5x_6] + E[x_1x_2]E[x_3x_5]E[x_4x_6] + \cdots$$

For the special case where all signals are identically distributed, this yields the relation

$$\begin{aligned} E[x_1 \cdot x_2 \cdot \ldots \cdot x_n] &= (1 \cdot 3 \cdot 5 \cdot \ldots \cdot n-1)E[x_ix_j]^{n/2} \\ &= \frac{n!}{2^{n/2}(n/2)!}E[x_ix_j]^{n/2} \qquad i \neq j \end{aligned} \tag{2.3}$$

2.3 CORRELATION FUNCTIONS

Correlation functions describe the sequential structures of signals. In signal analysis, they can be used to detect repeated patterns within a signal. In systems analysis, they are used to analyze relationships between signals, often a system's input and output.

2.3.1 Autocorrelation Functions

The autocorrelation function characterizes the sequential structure of a signal, $x(t)$, by describing its relation to a copy of itself shifted by τ time units. The correlation will be maximal at $\tau = 0$ and change as the lag, τ, is increased. The change in correlation as a function of lag characterizes the sequential structure of the signal.

Three alternative correlation functions are used commonly: the autocorrelation function, the autocovariance function, and the autocorrelation-coefficient function.

To illustrate the relationships between these functions, consider a random signal, $x(t)$, and let $x_0(t)$ be a zero-mean, unit variance random variable derived from $x(t)$,

$$x_0(t) = \frac{x(t) - \mu_x}{\sigma_x} \tag{2.4}$$

The *autocorrelation* function of the signal, $x(t)$, is defined by

$$\phi_{xx}(\tau) = E[x(t-\tau)x(t)] \tag{2.5}$$

Figure 2.2 illustrates time records of several typical signals together with their autocorrelations. Evidently, the autocorrelations reveal structures not apparent in the time records of the signals.

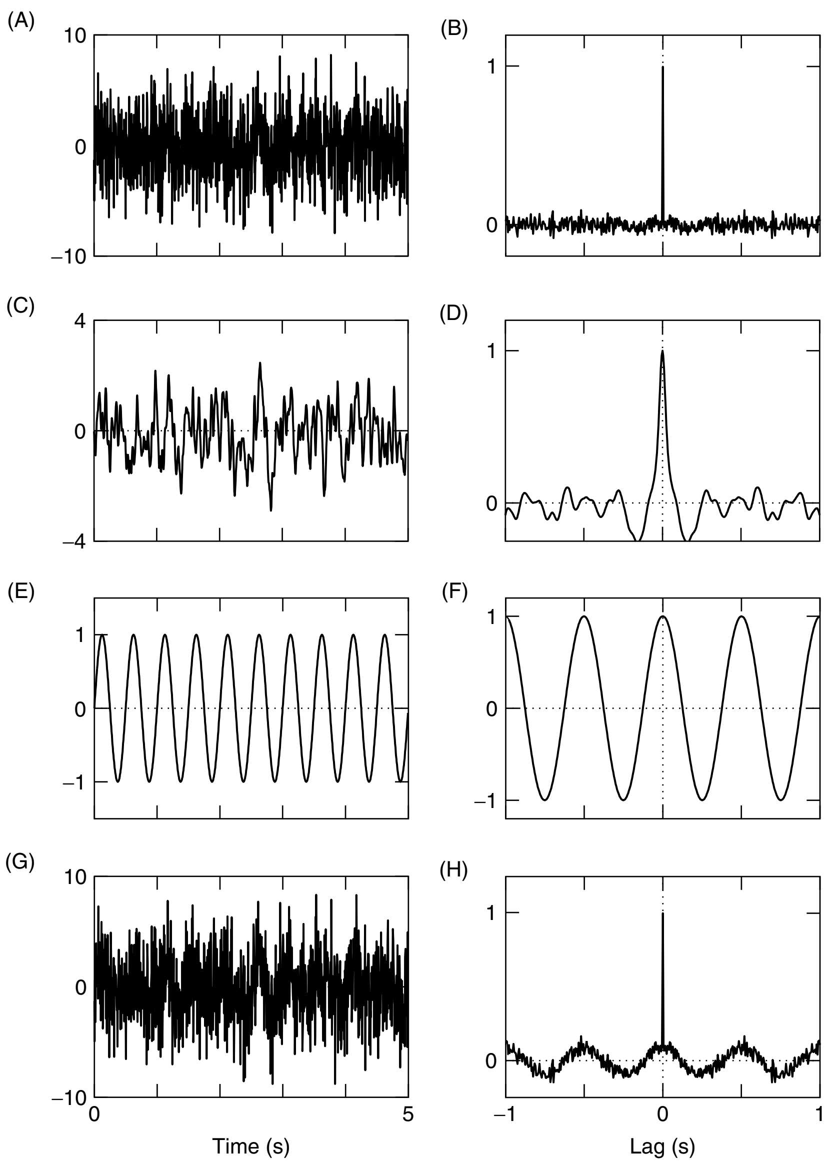

Figure 2.2 Time signals (left column) and their autocorrelation coefficient functions (right column). (A, B) Low-pass filtered white-noise signal. (C, D) Low-pass filtered white noise with a lower cutoff. The resulting signal is smoother and the autocorrelation peak is wider. (E, F) Sine wave. The autocorrelation function is also a sinusoid. (G, H) Sine wave buried in white noise. The sine wave is more visible in the autocorrelation than in the time record.

Substituting equation (2.4) into equation (2.5) gives

$$\phi_{xx}(\tau) = E[(\sigma_x x_0(t-\tau) + \mu_x)(\sigma_x x_0(t) + \mu_x)] \tag{2.6}$$

$$= \sigma_x^2 E[x_0(t-\tau)x_0(t)] + \mu_x^2 \tag{2.7}$$

Thus, the autocorrelation, ϕ_{xx}, at any lag, τ, depends on both the mean, μ_x, and the variance, σ_x^2, of the signal.

In many applications, particularly where systems have been linearized about an operating point, signals will not have zero means. It is common practice in these cases to remove the mean before calculating the correlation, resulting in the *autocovariance* function:

$$\begin{aligned} C_{xx}(\tau) &= E[(x(t-\tau)-\mu_x)(x(t)-\mu_x)] \\ &= \phi_{xx}(\tau) - \mu_x^2 \end{aligned} \tag{2.8}$$

Thus, if $\mu_x = 0$, the autocorrelation and autocovariance functions will be identical.

At zero lag, the value of the autocovariance function is

$$\begin{aligned} C_{xx}(0) &= E[(x(t)-\mu_x)^2] \\ &= \sigma_x^2 \end{aligned}$$

which is the signal's variance. Dividing the autocovariance by the variance gives the *autocorrelation coefficient* function,

$$\begin{aligned} r_{xx}(\tau) &= \frac{C_{xx}(\tau)}{C_{xx}(0)} \\ &= E[x_0(t-\tau)x_0(t)] \end{aligned} \tag{2.9}$$

The values of the autocorrelation coefficient function may be interpreted as correlations in the statistical sense. Thus the autocorrelation coefficient function ranges from 1 (i.e., complete positive correlation) to 0 (i.e., no correlation), through −1 (i.e., complete negative correlation).

It is not uncommon, though it can be very confusing, for the autocovariance function and the autocorrelation coefficient function to be referred to as the autocorrelation function. The relationships between the different autocorrelation functions are illustrated in Figure 2.3.

Finally, note that all the autocorrelation formulations are even and thus are symmetric about zero lag:

$$\begin{aligned} \phi_{xx}(-\tau) &= \phi_{xx}(\tau) \\ C_{xx}(-\tau) &= C_{xx}(\tau) \\ r_{xx}(-\tau) &= r_{xx}(\tau) \end{aligned} \tag{2.10}$$

2.3.2 Cross-Correlation Functions

Cross-correlation functions measure the sequential relation between two signals. It is important to remember that two signals may each have considerable sequential structure yet have no correlation with each other.

The *cross-correlation* function between two signals $x(t)$ and $y(t)$ is defined by

$$\phi_{xy}(\tau) = E[x(t-\tau)y(t)] \tag{2.11}$$

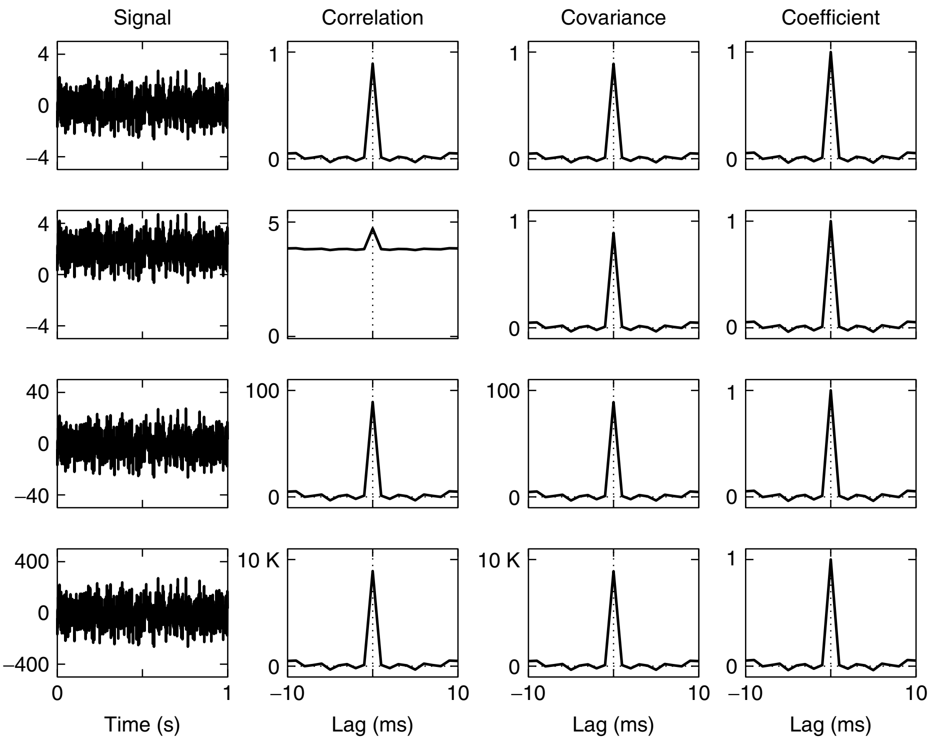

Figure 2.3 Examples of autocorrelation functions. The first column shows the four time signals, the second column shows their autocorrelation functions, the third column shows the corresponding autocovariance functions, and the fourth column the equivalent autocorrelation coefficient functions. First Row: Low-pass filtered sample of white, Gaussian noise with zero mean and unit variance. Second Row: The signal from the first row with an offset of 2 added. Note that the mean is clearly visible in the autocorrelation function but not in the auto-covariance or autocorrelation coefficient function. Third Row: The signal from the top row multiplied by 10. Bottom Row: The signal from the top row multiplied by 100. Scaling the signal changes the values of the autocorrelation and autocovariance function but not of the autocorrelation coefficient function.

As before, removing the means of both signals prior to the computation gives the *cross-covariance* function,

$$\begin{aligned} C_{xy}(\tau) &= E[(x(t-\tau)-\mu_x)(y(t)-\mu_y)] \\ &= \phi_{xy}(\tau) - \mu_x\mu_y \end{aligned} \tag{2.12}$$

where μ_x is the mean of $x(t)$, and μ_y is the mean of $y(t)$. Notice that if either $\mu_x = 0$ or $\mu_x = 0$, the cross-correlation and the cross-covariance functions will be identical.

The *cross-correlation coefficient* function of two signals, $x(t)$ and $y(t)$, is defined by

$$r_{xy}(\tau) = \frac{C_{xy}(\tau)}{\sqrt{C_{xx}(0)C_{yy}(0)}} \tag{2.13}$$

The value of the cross-correlation coefficient function at zero lag, $r_{xy}(0)$, will be unity only if the two signals are identical to within a scale factor (i.e., $x(t) = ky(t)$). In this case, the cross-correlation coefficient function will be the same as the autocorrelation coefficient function of either signal.

As with the autocorrelation coefficient function, the values of the cross-correlation coefficient function can be interpreted as correlations in the statistical sense, ranging from complete positive correlation (1) through 0 to complete negative correlation (-1). Furthermore, the same potential for confusion exists; the cross-covariance and cross-correlation coefficient functions are often referred to simply as cross-correlations.

The various cross-correlation formulations are neither even nor odd, but do satisfy the interesting relations:

$$\phi_{xy}(\tau) = \phi_{yx}(-\tau) \tag{2.14}$$

and

$$\phi_{xy}(\tau) \leq \sqrt{\phi_{xx}(0)\phi_{yy}(0)} \tag{2.15}$$

Finally, consider the cross-correlation between $x(t)$ and $y(t)$ where

$$y(t) = \alpha x(t - \tau_0) + v(t)$$

that is, $y(t)$ is a delayed, scaled version of $x(t)$ added to an uncorrelated noise signal, $v(t)$. Then,

$$\phi_{xy}(\tau) = \alpha\phi_{xx}(\tau - \tau_0)$$

That is, the cross-correlation function is simply the autocorrelation of the input signal, $x(t)$, displaced by the delay τ_0, and multiplied by the gain α. As a result, the lag at which the cross-correlation function reaches its maximum provides an estimate of the delay.

2.3.3 Effects of Noise

Frequently, the auto- and cross-correlations of the signals $x(t)$ and $y(t)$ must be estimated from measurements containing additive noise. Let

$$\begin{aligned} w(t) &= x(t) + n(t) \\ z(t) &= y(t) + v(t) \end{aligned}$$

where $n(t)$ and $v(t)$ are independent of $x(t)$ and $y(t)$ and of each other.

First, consider the autocorrelation of one signal,

$$\begin{aligned} \phi_{zz}(\tau) &= E\left[(y(t-\tau) + v(t-\tau))(y(t) + v(t))\right] \\ &= \phi_{yy}(\tau) + \phi_{yv}(\tau) + \phi_{vy}(\tau) + \phi_{vv}(\tau) \end{aligned}$$

The terms $\phi_{yv} \equiv \phi_{vy} \equiv 0$ will disappear because y and v are independent. However, the remaining term, ϕ_{vv}, is the autocorrelation of the noise sequence and will not be zero. As a result, additive noise will bias autocorrelation estimates,

$$\phi_{zz}(\tau) = \phi_{yy}(\tau) + \phi_{vv}(\tau)$$

In contrast, for the cross-correlation function

$$\begin{aligned}\phi_{wz}(\tau) &= E[w(t)z(t+\tau)]\\ &= E\left[(x(t-\tau)+n(t-\tau))(y(t)+v(t))\right]\\ &= \phi_{xy}(\tau)+\phi_{xv}(\tau)+\phi_{ny}(\tau)+\phi_{nv}(\tau)\end{aligned}$$

and $\phi_{xv} \equiv \phi_{ny} \equiv \phi_{nv} \equiv 0$, since the noise signals are independent of each other by assumption. Thus,

$$\phi_{wz}(\tau) = \phi_{xy}(\tau)$$

and estimates of the cross-correlation with additive noise will be unbiased.

2.3.4 Estimates of Correlation Functions

Provided that $x(t)$ and $y(t)$ are realizations of ergodic processes,* the expected value in equation (2.11) may be replaced with an infinite time-average:

$$\phi_{xy}(\tau) = \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} x(t-\tau)y(t)\,dt \tag{2.16}$$

In any practical application, $x(t)$ and $y(t)$ will be finite-length, discrete-time signals. Thus, it can be assumed that $x(t)$ and $y(t)$ have been sampled every Δ_t units from $t = 0, \Delta_t, \dots, (N-1)\Delta_t$, giving the samples $x(i)$ and $y(i)$ for $i = 1, 2, \dots, N$. By using rectangular integration, the cross-correlation function (2.16) can be approximated as

$$\hat{\phi}_{xy}(\tau) = \frac{1}{N-\tau}\sum_{i=\tau}^{N} x(i-\tau)y(i) \tag{2.17}$$

This is an unbiased estimator, but its variance increases with lag τ. To avoid this, it is common to use the estimator:

$$\hat{\phi}_{xy}(\tau) = \frac{1}{N}\sum_{i=\tau}^{N} x(i-\tau)y(i) \tag{2.18}$$

which is biased, because it underestimates correlation function values at long lags, but its variance does not increase with lag τ.

Similar estimators of the auto- and cross-covariance and correlation coefficient functions may be constructed. Note that if N is large with respect to the maximum lag, the biased and unbiased estimates will be very similar.

*Strictly speaking, a deterministic signal, such as a sinusoid, is nonstationary and is certainly not ergodic. Nevertheless, computations based on time averages are routinely employed with both stochastic and deterministic signals. Ljung (1999) defines a class of *quasi-stationary* signals, together with an alternate expected value operator, to get around this technicality.

2.3.5 Frequency Domain Expressions

The discrete Fourier transform of a discrete-time signal of length N is defined as

$$U(f) = \mathfrak{F}(u(t)) = \sum_{t=1}^{N} u(t)e^{-2\pi jft/N} \tag{2.19}$$

Note that in this definition, f takes on integer values $f = 0, 1, \ldots, N-1$. If Δ_t is the sampling increment in the time domain, then the sampling increment in the frequency domain will be

$$\Delta_f = \frac{1}{N\Delta_t}$$

The inverse Fourier transform is given by*,

$$u(t) = \mathfrak{F}^{-1}(U(f)) = \frac{1}{N}\sum_{f=1}^{N} U(f)e^{2\pi jft/N} \tag{2.20}$$

Direct computation of either equation (2.19) or equation (2.20) requires about $4N^2$ flops, since the computations involve complex numbers. However, if the fast Fourier transform (FFT) algorithm (Bendat and Piersol, 1986; Oppenheim and Schafer, 1989; Press et al., 1992) is used, the cost can be reduced to about $N\log_2(N)$ flops (Oppenheim and Schafer, 1989). Note that these figures are for real valued signals. FFTs of complex-valued signals will require twice as many flops.

The Fourier transform of the autocorrelation function is called the power spectrum. For a discrete-time signal,

$$\begin{aligned} S_{uu}(f) &= \sum_{\tau=0}^{N-1} \phi_{uu}(\tau)e^{-2\pi jf\tau/N} \\ &= E\left[\sum_{\tau=0}^{N-1} u(t)u(t-\tau)e^{-2\pi jf\tau/N}\right] \end{aligned} \tag{2.21}$$

This expression, like the definition of the autocorrelation, is in terms of an expected value. To estimate the power spectrum, $u(t)$ is treated as if it were ergodic, and the expected value is replaced with a time average. There are several ways to do this.

One possibility is to estimate the correlation in the time domain using a time average (2.18) and then Fourier transform the result:

$$\mathfrak{F}\left(\hat{\phi}_{uu}(\tau)\right) = \frac{1}{N}\sum_{\tau=0}^{N}\sum_{i=0}^{N} u(i-\tau)u(i)e^{-j2\pi\tau f/N}$$

Multiplying by $e^{-j2\pi(i-i)f/N} = 1$ and then simplifying gives

$$\begin{aligned} \mathfrak{F}\left(\hat{\phi}_{uu}(\tau)\right) &= \frac{1}{N}\sum_{\tau=0}^{N} u(i-\tau)e^{j2\pi(i-\tau)f/N}\sum_{i=0}^{N} u(i)e^{-j2\pi if/N} \\ &= \frac{1}{N}U^*(f)U(f) \end{aligned} \tag{2.22}$$

*Some authors (Ljung, 1999) include a factor of $1/\sqrt{N}$ in both the forward and inverse Fourier transform.

Taking the inverse Fourier transform of (2.22) yields (2.18), the biased estimate of the cross-correlation. In practice, correlation functions are often computed this way, using the FFT to transform to and from the frequency domain.

An alternative approach, the *averaged periodogram* (Oppenheim and Schafer, 1989), implements the time average differently. Here, the signal is divided into D segments of length N_D, and the ensemble of segments is averaged to estimate the expected value. Thus, the averaged periodogram spectral estimate is

$$\hat{S}_{uu}(f) = \frac{1}{DN_D} \sum_{d=1}^{D} U_d^*(f) U_d(f) \tag{2.23}$$

where $U_d(f)$ is the Fourier transform of the dth segment of N_D points of the signal $u(t)$, and the asterisk, $U^*(f)$, denotes the complex conjugate.

It is common to overlap the data blocks to increase the number of blocks averaged. Since the FFT assumes that the data are periodic, it is also common to window the blocks before transforming them. The proper selection of the window function, and of the degree of overlap between windows, is a matter of experience as well as trial and error. Further details can be found in Bendat and Piersol (1986).

The averaged periodogram (2.23) is the most commonly used nonparametric spectral estimate, and it is implemented in MATLAB's `spectrum` command. Parametric spectral estimators have also been developed and are described in Percival and Walden (1993).

2.3.5.1 The Cross-Spectrum

The Fourier transform of the cross-correlation function is called the cross-spectrum,

$$S_{uy}(f) = \sum_{\tau=0}^{N-1} \phi_{uy}(\tau) e^{-2\pi j f \tau / N} \tag{2.24}$$

Replacing the autocorrelation with the cross-correlation in the preceding derivations gives the Fourier transform of the cross-correlation,

$$\mathfrak{F}\left(\hat{\phi}_{uy}(\tau)\right) = \frac{1}{N} U^*(f) Y(f) \tag{2.25}$$

and an averaged periodogram estimate of the cross-spectrum,

$$\hat{S}_{uy}(f) = \frac{1}{DN_D} \sum_{d=1}^{D} U_d^*(f) Y_d(f) \tag{2.26}$$

2.3.6 Applications

The variance of a biased correlation estimate (2.18) is proportional to $1/N$ and thus decreases as N increases. Furthermore, the effect of the bias is a scaling by a factor of $N/(N-\tau)$, which decreases as N increases with respect to τ. Thus, in general the length of a correlation function should be much shorter than the data length from which it is estimated; that is, $N \gg \tau$. As a rule of thumb, correlation functions should be no more than one-fourth the length of the data and should never exceed one-half of the data length.

Autocovariance functions determined from stochastic signals tend to "die out" at longer lags. The lag at which the autocovariance function has decreased to values that cannot be distinguished from zero provides a subjective measure of the extent of the sequential structure in a signal, or its "memory."

In contrast, if there is an underlying periodic component, the autocorrelation function will not die out but will oscillate at the frequency of the periodic component. If there is substantial noise in the original signal, the periodicity may be much more evident in the correlation function than in the original data. This periodicity will be evident in the autocorrelation function at large lags, after the contribution from the noise component has decayed to zero. An example of this is presented in the bottom row of Figure 2.2.

A common use for the cross-covariance function, as illustrated in the middle panel of Figure 2.4, is to estimate the delay between two signals. The delay is the lag at which

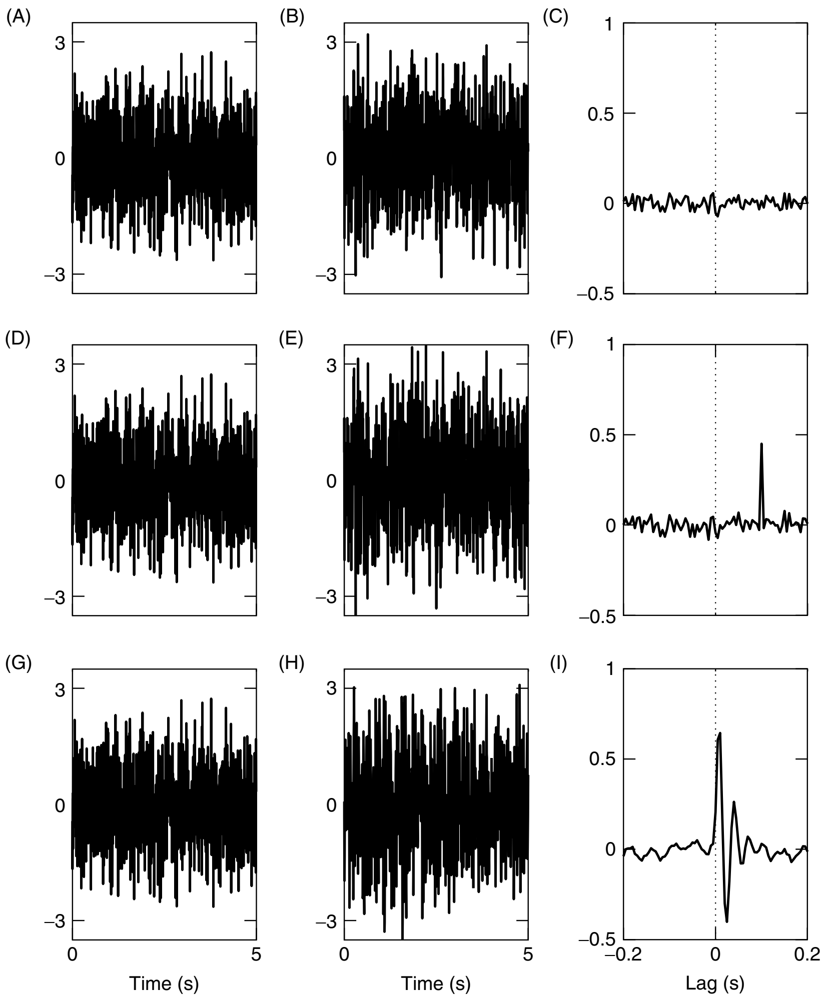

Figure 2.4 Examples of the cross-correlation coefficient function. Top Row: The signals in A and B are uncorrelated with each other. Their cross-correlation coefficient function, C, is near zero at all lags. Middle Row: E is a delayed, scaled version of D with additive noise. The peak in the cross-correlation coefficient function, F, indicates the delay between input and output. Bottom Row: G is a low-pass filtered version of H, also with additive noise. The filtering appears as ringing in I.

the cross-covariance function is maximal. For example, the delay between two signals measured at two points along a nerve can be determined from the cross-covariance function (Heetderks and Williams, 1975). It should be remembered that dynamics can also give rise to delayed peaks in the cross-correlation function.

2.3.7 Higher-Order Correlation Functions

Second-order cross-correlation functions will be represented by ϕ with three subscripts. Thus,

$$\phi_{xxy}(\tau_1, \tau_2) = E[x(t-\tau_1)x(t-\tau_2)y(t)] \tag{2.27}$$

while the second-order autocorrelation function is

$$\phi_{xxx}(\tau_1, \tau_2) = E[x(t-\tau_1)x(t-\tau_2)x(t)] \tag{2.28}$$

Note that there is some confusion in the literature about the terminology for this function. Some authors (Korenberg, 1988) use the nomenclature "second-order," as does this book; others have used the term "third-order" (Marmarelis and Marmarelis, 1978) to describe the same relation.

2.4 MEAN-SQUARE PARAMETER ESTIMATION

Given a model structure and a set of input–output measurements, it is often desirable to identify the parameter vector that generates the output that "best approximates" the measured output. Consider a model, $\mathbf{M}$, with a parameter set, $\boldsymbol{\theta}$, whose response to the input $u(t)$ will be written

$$\hat{y}(\boldsymbol{\theta}, t) = \mathbf{M}(\boldsymbol{\theta}, u(t))$$

A common definition for the "best approximation" is that which minimizes the mean-square error between the measured output, $z(t)$, and the model output, $\hat{y}(\boldsymbol{\theta}, t)$:

$$\mathrm{MSE}(\mathbf{M}, \boldsymbol{\theta}, u(t)) = E\left[\left(z(t) - \hat{y}(\boldsymbol{\theta}, t)\right)^2\right] \tag{2.29}$$

If the signals are ergodic, then this expectation can be evaluated using a time average over a single record. In discrete time, this results in the summation:

$$V_N(\mathbf{M}, \boldsymbol{\theta}, u(t)) = \frac{1}{N}\sum_{t=1}^{N}\left(z(t) - \hat{y}(\boldsymbol{\theta}, t)\right)^2 \tag{2.30}$$

which is often referred to as the "mean-square error," even though it is computed using a time average rather than an ensemble average.

Note that the MSE depends on the model structure, $\mathbf{M}$, the parameter vector, $\boldsymbol{\theta}$, and the test input, $u(t)$. For a particular structure, the goal is to find the parameter vector, $\hat{\boldsymbol{\theta}}$, that minimizes (2.30). That is (Ljung, 1999):

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} V_N(\boldsymbol{\theta}, u(t)) \tag{2.31}$$

In general, there is no closed-form solution to this minimization problem, so the "parameter space" must be searched using iterative methods as discussed in Chapter 8.

2.4.1 Linear Least-Squares Regression

If the model output, $\hat{y}(t)$, is a linear function of the parameters, the model may be formulated as a matrix equation (Beck and Arnold, 1977),

$$\hat{\mathbf{y}}(\boldsymbol{\theta}) = \mathbf{U}\boldsymbol{\theta} \tag{2.32}$$

where $\hat{\mathbf{y}}$ is an N element vector containing $\hat{y}(t)$, and $\mathbf{U}$ is a matrix with N rows (one per data point) and M columns (one per model parameter). Equation (2.30) then becomes

$$\begin{aligned} V_N(\boldsymbol{\theta}) &= \frac{1}{N}\left(\mathbf{z} - \mathbf{U}\boldsymbol{\theta}\right)^T\left(\mathbf{z} - \mathbf{U}\boldsymbol{\theta}\right) \\ &= \frac{1}{N}\left(\mathbf{z}^T\mathbf{z} - 2\boldsymbol{\theta}^T\mathbf{U}^T\mathbf{z} + \boldsymbol{\theta}^T\mathbf{U}^T\mathbf{U}\boldsymbol{\theta}\right) \end{aligned}$$

which may be solved analytically as follows. First, differentiate with respect to $\boldsymbol{\theta}$ to get the gradient

$$\frac{\partial V_N}{\partial \boldsymbol{\theta}} = \frac{2}{N}(\mathbf{U}^T\mathbf{U}\boldsymbol{\theta} - \mathbf{U}^T\mathbf{z})$$

The minimum mean-square error solution, $\hat{\boldsymbol{\theta}}$, is found by setting the gradient to zero and solving:

$$\begin{aligned} \mathbf{U}^T\mathbf{U}\hat{\boldsymbol{\theta}} &= \mathbf{U}^T\mathbf{z} \\ \hat{\boldsymbol{\theta}} &= (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{z} \end{aligned} \tag{2.33}$$

Thus, for any model structure where the output is *linear in the parameters*, as in equation (2.32), the optimal parameter vector may be determined directly by from equations (2.33), called the normal equations.* Many of the model structures examined in this text are linear in their parameters even though they describe nonlinear systems. Thus, solutions of the normal equations and their properties are fundamental to much of what will follow.

2.4.1.1 Example: Polynomial Fitting Consider, for example, the following problem. Given N measurements of an input signal, $u_1, u_2, \ldots, u_N$ and output $z_1, z_2, \ldots, z_N$, find the third-order polynomial that best describes the relation between u_j and z_j. To do so, assume that

$$\begin{aligned} y_j &= c^{(0)} + c^{(1)}u_j + c^{(2)}u_j^2 + c^{(3)}u_j^3 \\ z_j &= y_j + v_j \end{aligned}$$

*In the MATLAB environment, the normal equation, (2.33), can be solved using the "left division" operator. $\hat{\theta} = \mathbf{U}\backslash\mathbf{z}$.

where v is a zero-mean, white Gaussian noise sequence. First, construct the regression matrix

$$\mathbf{U} = \begin{bmatrix} 1 & u_1 & u_1^2 & u_1^3 \\ 1 & u_2 & u_2^2 & u_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & u_N & u_N^2 & u_N^3 \end{bmatrix} \tag{2.34}$$

Then, rewrite the expression for $\mathbf{z}$ as the matrix equation

$$\mathbf{z} = \mathbf{U}\boldsymbol{\theta} + \mathbf{v} \tag{2.35}$$

where $\boldsymbol{\theta} = \left[c^{(0)}\ c^{(1)}\ c^{(2)}\ c^{(3)}\right]^T$, and use equations (2.33) to solve for $\boldsymbol{\theta}$.

2.4.2 Properties of Estimates

The solution of equations (2.33) gives the model parameters that provide the minimum mean-square error solution. It is important to know how close to the true parameters these estimated values are likely to be. Consider the following conditions:

1. The model structure is correct; that is, there is a parameter vector such that the system can be represented exactly as $\mathbf{y} = \mathbf{U}\boldsymbol{\theta}$.
2. The output, $\mathbf{z} = \mathbf{y} + \mathbf{v}$, contains additive noise, $v(t)$, that is zero-mean and statistically independent of the input, $u(t)$.

If conditions 1 and 2 hold, then the expected value of the estimated parameter vector is

$$\begin{aligned} E[\hat{\boldsymbol{\theta}}] &= E[(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{z}] \\ &= \boldsymbol{\theta} + E[(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{v}] \end{aligned} \tag{2.36}$$

However, since $\mathbf{v}$ is independent of $\mathbf{u}$, it will be independent of all the columns of $\mathbf{U}$ and so

$$E[\hat{\boldsymbol{\theta}}] = \boldsymbol{\theta} \tag{2.37}$$

That is, the least-squares parameter estimate is *unbiased* (Beck and Arnold, 1977).

The covariance matrix for these parameter estimates is

$$\mathbf{C}_{\hat{\boldsymbol{\theta}}} = E[(\hat{\boldsymbol{\theta}} - E[\hat{\boldsymbol{\theta}}])(\hat{\boldsymbol{\theta}} - E[\hat{\boldsymbol{\theta}}])^T] \tag{2.38}$$

$$= (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T E[\mathbf{v}\mathbf{v}^T]\mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1} \tag{2.39}$$

Usually, equation (2.39) cannot be evaluated directly since $E[\mathbf{v}\mathbf{v}^T]$, the covariance matrix of the measurement noise, is not known. However, if the measurement noise is white, then the noise covariance in equation (2.39) reduces to $E[\mathbf{v}\mathbf{v}^T] = \sigma_v^2\mathbf{I_N}$, where σ_v^2 is the variance of $v(t)$, and $\mathbf{I_N}$ is an $N \times N$ identity matrix. Furthermore, an unbiased estimate of the noise variance (Beck and Arnold, 1977) is given by

$$\hat{\sigma}_v^2 = \frac{1}{N-M}\sum_{t=1}^{N}(z(t) - \hat{y}(t))^2 \tag{2.40}$$

where M is the number of model parameters. Thus the covariance matrix for the parameter estimates, $\mathbf{C}_{\hat{\boldsymbol{\theta}}}$, reduces to

$$\mathbf{C}_{\hat{\boldsymbol{\theta}}} = \hat{\sigma}_v^2 (\mathbf{U}^T\mathbf{U})^{-1} \tag{2.41}$$

which can be evaluated directly.

2.4.2.1 Condition of the Hessian Instead of using probabilistic considerations, as in the previous section, it is also instructive to examine the numerical properties of the estimate. Thus, instead of taking an expected value over a hypothetical ensemble of records, consider the sensitivity of the computed estimate, $\hat{\boldsymbol{\theta}}$, to changes in the single, finite-length, input–output record, $[u(t)\ z(t)]$.

From the normal equations (2.33), it is evident that the error in the parameter estimate is

$$\begin{aligned} \tilde{\boldsymbol{\theta}} &= \boldsymbol{\theta} - \hat{\boldsymbol{\theta}} \\ &= \boldsymbol{\theta} - (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{z} \\ &= -(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{v} \end{aligned} \tag{2.42}$$

where $v(t)$ is the noise in the measured output, $\mathbf{z} = \mathbf{U}\boldsymbol{\theta} + \mathbf{v}$.

The error is the product of two terms. The first term, $(\mathbf{U}^T\mathbf{U})^{-1}$, is the inverse of the *Hessian*, denoted $\mathbf{H}$, an $M \times M$ matrix containing the second-order derivatives of the cost function with respect to the parameters.

$$\mathbf{H}(i, j) = \frac{\partial^2 V_N(\boldsymbol{\theta})}{\partial\theta_i \partial\theta_j} = \mathbf{U}^T\mathbf{U} \tag{2.43}$$

Furthermore, if the measurement noise is white, the inverse of the Hessian is proportional to the parameter covariance matrix, $\mathbf{C}_{\hat{\boldsymbol{\theta}}}$, given in equation (2.41).

Let $\boldsymbol{\nu} = \mathbf{U}^T\mathbf{v}$ be the second term in equation (2.42). Substituting these two expressions gives

$$\tilde{\boldsymbol{\theta}} = -\mathbf{H}^{-1}\boldsymbol{\nu}$$

Now, consider the effect of a small change in $\boldsymbol{\nu}$ on the parameter estimate. The Hessian is a non-negative definite matrix, so its singular value decomposition (SVD) (Golub and Van Loan, 1989) can be written as

$$\mathbf{H} = \mathbf{V}\mathbf{S}\mathbf{V}^T$$

where $\mathbf{V} = [\mathbf{v_1}\ \mathbf{v_2}\ \dots\ \mathbf{v_M}]$ is an orthogonal matrix, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$, and $\mathbf{S}$ is a diagonal matrix, $\mathbf{S} = \operatorname{diag}[s_1, s_2 \dots, s_M]$, where $s_1 \geq s_2 \geq \cdots \geq s_M \geq 0$. Using this, the Hessian can be expanded as

$$\mathbf{H} = \sum_{i=1}^{M} s_i \mathbf{v_i}\mathbf{v_i}^T$$

and the estimation error, $\tilde{\boldsymbol{\theta}}$, becomes

$$\tilde{\boldsymbol{\theta}} = -\sum_{i=1}^{M} \frac{1}{s_i} \mathbf{v_i} \mathbf{v_i}^T \boldsymbol{\nu} \tag{2.44}$$

Notice that if the noise term, $\boldsymbol{\nu}$, changes in the direction parallel to the kth singular vector, $\mathbf{v_k}$, then the change in $\hat{\boldsymbol{\theta}}$ will be multiplied by $1/s_k$. Consequently, the ratio of the largest to smallest singular values will determine the relative sensitivity of the parameter estimates to noise. This ratio is referred to as the *condition number* (Golub and Van Loan, 1989) of the matrix and ideally should be close to 1.

2.5 POLYNOMIALS

Polynomials provide a convenient means to model the instantaneous, or *memoryless*, relationship between two signals. Section 2.4.1.1 demonstrated that fitting a polynomial between two data sequences can be done by solving a linear least-squares problem.

2.5.1 Power Series

The power series is a simple polynomial representation involving a sum of monomials in the input signal,

$$\begin{aligned} m(u) &= \sum_{q=0}^{Q} c^{(q)} u^q \\ &= \sum_{q=0}^{Q} c^{(q)} \mathcal{M}^{(q)}(u) \\ &= c^{(0)} + c^{(1)} u + c^{(2)} u^2 + c^{(3)} u^3 + \cdots \end{aligned}$$

where the notation $\mathcal{M}^{(q)}(u) = u^q$ is introduced to prepare for the discussion of orthogonal polynomials to follow. Uppercase script letters will be used throughout to represent polynomial systems.

The polynomial coefficients, $c^{(q)}$, can be estimated by solving the linear least-squares problem defined by equations (2.34) and (2.35). This approach may give reasonable results provided that there is little noise and the polynomial is of low order. However, this problem often becomes badly conditioned, resulting in unreliable coefficient estimates. This is because in power-series formulations the Hessian will often have a large condition number; that is the ratio of the largest and smallest singular values will be large. As a result, the estimation problem is ill-conditioned, since as equation (2.44) shows, small singular values in the Hessian will amplify errors in the coefficient estimates. The large condition number arises for two reasons:

1. The columns of $\mathbf{U}$ will have widely different amplitudes, particularly for high-order polynomials, unless $\sigma_u \approx 1$. As a result, the singular values of $\mathbf{U}$, which are the square roots of the singular values of the Hessian, will differ widely.

2. The columns of **U** will not be orthogonal. This is most easily seen by examining the Hessian, $\mathbf{U}^T\mathbf{U}$, which will have the form

$$\mathbf{H} = N \begin{bmatrix} 1 & E[u] & E[u^2] & \dots & E[u^q] \\ E[u] & E[u^2] & E[u^3] & \dots & E[u^{q+1}] \\ E[u^2] & E[u^3] & E[u^4] & \dots & E[u^{q+2}] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ E[u^q] & E[u^{q+1}] & E[u^{q+2}] & \dots & E[u^{2q}] \end{bmatrix}$$

Since **H** is not diagonal, the columns of **U** will not be orthogonal to each other. Note that the singular values of **U** can be viewed as the lengths of the semiaxes of a hyperellipsoid defined by the columns of **U**. Thus, nonorthogonal columns will stretch this ellipse in directions more nearly parallel to multiple columns and will shrink it in other directions, increasing the ratio of the axis lengths, and hence the condition number of the estimation problem (Golub and Van Loan, 1989).

2.5.2 Orthogonal Polynomials

Ideally, the regressors should be mutually orthogonal, so the Hessian will be diagonal with elements of similar size. This can be achieved by replacing the power series basis functions with another polynomial function $\mathcal{P}^{(q)}(u)$

$$m(u) = \sum_{q=0}^{Q} c^{(q)}\mathcal{P}^{(q)}(u) \tag{2.45}$$

chosen to make the estimation problem well-conditioned. The objective is to find a polynomial function that makes the Hessian diagonal. That is, the (i, j)th element of the Hessian should be

$$\begin{aligned} \mathbf{H}(i, j) &= N \cdot E[\mathcal{P}^{(i+1)}(u)\mathcal{P}^{(j+1)}(u)] \\ &= \delta_{i,j} \end{aligned}$$

which demonstrates that the terms of $\mathcal{P}^{(q)}(u)$ must be orthonormal.

The expected value of a function, g, of a random variable, u, with probability density $f(u)$, is given by (Bendat and Piersol, 1986; Papoulis, 1984)

$$E[g(u)] = \int_{-\infty}^{\infty} f(u)g(u)\,du$$

Consequently, the expected values of the elements of the Hessian will be

$$E[\mathcal{P}^{(i)}(u)\mathcal{P}^{(j)}(u)] = \int_{-\infty}^{\infty} f(u)\mathcal{P}^{(i)}(u)\mathcal{P}^{(j)}(u)\,du$$

This demonstrates that a particular polynomial basis function, $\mathcal{P}^{(q)}(u)$, will be orthogonal only for a particular input probability distribution. Thus each polynomial family will be orthogonal for a particular input distribution. Figure 2.5 shows the basis functions corresponding to three families of polynomials: the ordinary power series, as well as the Hermite and Tchebyshev families of orthogonal polynomials to be discussed next.

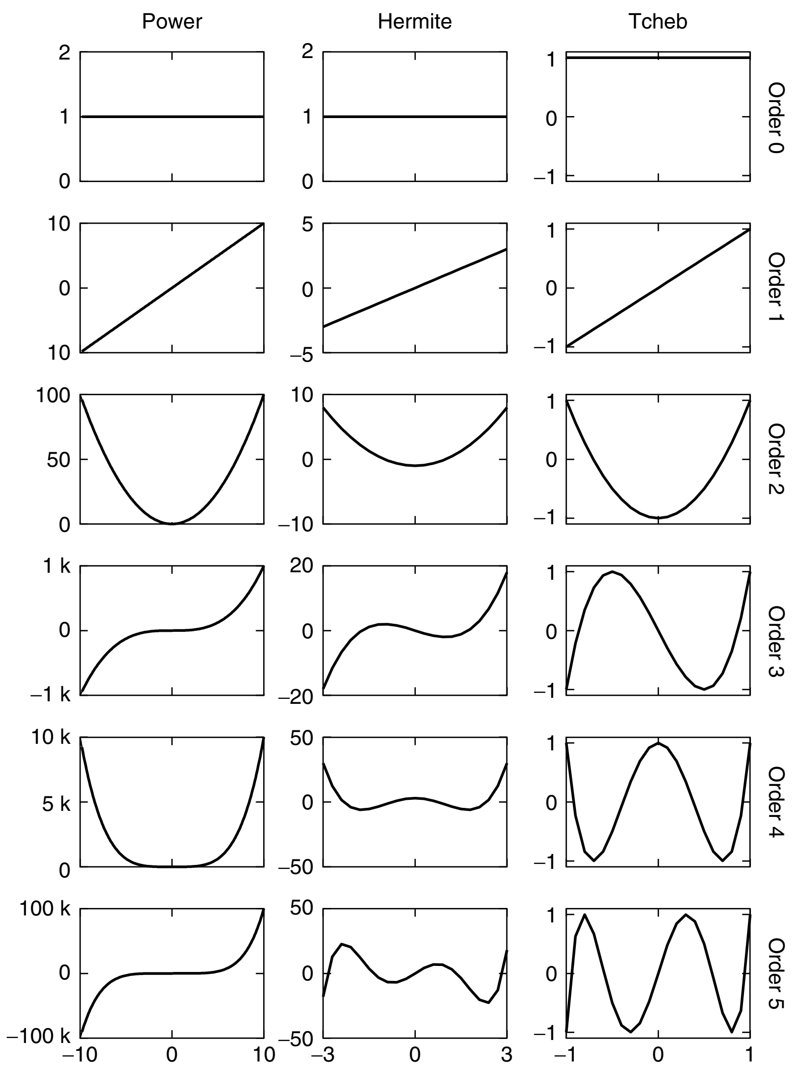

Figure 2.5 Power series, Hermite and Tchebyshev polynomials of orders 0 through 5. Left Column: Power series polynomials over the arbitrarily chosen domain [−10 10]. Middle Column: Hermite polynomials over the domain [−3 3] corresponding to most of the range of the unit-variance, normal random variable. Right Column: Tchebyshev polynomials over their full domain [−1 1].

2.5.3 Hermite Polynomials

The Hermite polynomials $\mathcal{H}^{(q)}(u)$ result when the orthogonalization is done for inputs with a zero-mean, unit-variance, Gaussian distribution. Thus, Hermite polynomials are constructed such that

$$\int_{-\infty}^{\infty} \exp(-u^2)\mathcal{H}^{(i)}(u)\mathcal{H}^{(j)}(u)\,du = 0 \qquad \text{for } i \neq j \tag{2.46}$$

Using equation (2.46) and the results for the expected value of products of Gaussian variables (2.3), the Hermite polynomials can be shown to be

$$\mathcal{H}^{(n)}(u) = n! \sum_{m=0}^{\lfloor n/2 \rfloor} \frac{(-1)^m}{m! 2^m (n-2m)!} u^{(n-2m)} \tag{2.47}$$

The first four elements are

$$\begin{aligned}
\mathcal{H}^{(0)}(u) &= 1 \\
\mathcal{H}^{(1)}(u) &= u \\
\mathcal{H}^{(2)}(u) &= u^2 - 1 \\
\mathcal{H}^{(3)}(u) &= u^3 - 3u
\end{aligned}$$

The Hermite polynomials may also be generated using the recurrence relation,

$$\mathcal{H}^{(k+1)}(u) = u\mathcal{H}^{(k)}(u) - k\mathcal{H}^{(k-1)}(u) \tag{2.48}$$

Note that these polynomials are only orthogonal for zero-mean, unit variance, Gaussian inputs. Consequently, input data are usually transformed to zero mean and unit variance before fitting Hermite polynomials. The transformation is retained as part of the polynomial representation and used to transform any other inputs that may be applied to the polynomial.

2.5.4 Tchebyshev Polynomials

A Gaussian random variable can take on any value between $-\infty$ and $+\infty$ (although the probability of attaining the extreme values is negligible). Real data, on the other hand, always have a finite range since they are limited by the dynamic range of the recording apparatus, if nothing else. Thus, it is logical to develop a set of polynomials that are orthogonal over a finite range.

The Tchebyshev polynomials, $\mathcal{T}$, are orthogonalized over the range $[-1\ 1]$ for the probability distribution $(1-u^2)^{-1/2}$.

$$\int_{-1}^{1} \frac{\mathcal{T}^{(i)}(u)\mathcal{T}^{(j)}(u)}{\sqrt{1-u^2}} du = \begin{cases} \frac{\pi}{2}\delta_{i,j} & \text{for } i \neq 0, j \neq 0 \\ \pi & \text{for } i = j = 0 \end{cases} \tag{2.49}$$

This probability density tends to infinity at ± 1, and thus does not correspond to the PDF of any realistic data set. Thus, the Tchebyshev polynomials will not be exactly orthogonal for any data. However, all Tchebyshev polynomial basis functions are bounded between -1 and $+1$ for inputs between -1 and $+1$ (see Figure 2.5). In addition, each goes through all of its local extrema (which are all ± 1) in this range. Thus, although the regressors will not be exactly orthogonal, they will have similar variances, and so the estimation problem should remain well-scaled. Hence, Tchebyshev polynomials are used frequently since they usually lead to well-conditioned regressions.

The general expression for the order-q Tchebyshev polynomial is

$$\mathcal{T}^{(q)}(u) = \frac{q}{2} \sum_{m=0}^{\lfloor q/2 \rfloor} \frac{(-1)^m}{q-m} \binom{q-m}{m} (2u)^{q-2m} \tag{2.50}$$

The first four Tchebyshev polynomials are

$$\begin{aligned}\mathcal{T}^{(0)}(u) &= 1\\ \mathcal{T}^{(1)}(u) &= u\\ \mathcal{T}^{(2)}(u) &= 2u^2 - 1\\ \mathcal{T}^{(3)}(u) &= 4u^3 - 3u\end{aligned}$$

The Tchebyshev polynomials are also given by the recursive relationship

$$\mathcal{T}^{(q+1)}(u) = 2u\mathcal{T}^{(q)}(u) - \mathcal{T}^{(q-1)}(u) \tag{2.51}$$

In practice, input data are transformed to $[-1\ 1]$ prior to fitting the coefficients, and the scale factor is retained as part of the polynomial representation.

2.5.5 Multiple-Variable Polynomials

Polynomials in two or more variables will also be needed to describe nonlinear systems. To establish the notation for this section, consider the types of terms involved in a multivariable power series. The order-zero term will be a constant, as in the single variable case. Similarly, the first-order terms will include only a single input, raised to the first power, and thus will have the same form as their single-input counterparts,

$$\mathcal{M}^{(1)}(u_k) = u_k$$

However, the second-order terms will involve products of two inputs, either two copies of the same signal or two distinct signals. Thus, the second-order terms will be of two types,

$$\begin{aligned}\mathcal{M}^{(2)}(u_k, u_k) &= u_k^2\\ \mathcal{M}^{(2)}(u_j, u_k) &= u_j u_k\end{aligned}$$

For example, the second-order terms in a three-input, second-order polynomial will involve three single-input terms, u_1^2, u_2^2, and u_3^2, and three two-input terms, u_1u_2, u_1u_3, and u_2u_3. To remain consistent with the notation for single-variable polynomials, let

$$\mathcal{M}^{(2)}(u_k) = \mathcal{M}^{(2)}(u_k, u_k) = u_k^2$$

Similarly, the order-q terms will involve from one to q inputs, raised to powers such that the sum of their exponents is q. For example, the third-order terms in a three-input polynomial are

$$\begin{array}{ccccc} u_1^3 & u_2^3 & u_3^3 & u_1^2u_2 & u_1^2u_3\\ u_2^2u_1 & u_2^2u_3 & u_3^2u_1 & u_3^2u_2 & u_1u_2u_3\end{array}$$

2.5.5.1 *Orthogonal Multiple-Input Polynomials*

Next, consider the construction of orthogonal, multiple-input polynomials. Let $u_1, u_2, \ldots, u_n$ be mutually orthonormal, zero-mean Gaussian random variables. These may be measured signals that just happen to be orthonormal, but will more likely result from orthogonalizing and normalizing a set of more general (Gaussian) signals with a QR factorization or SVD.

For Gaussian inputs, it is reasonable to start with Hermite polynomials. The order-zero term will be a constant, since it is independent of the inputs. Similarly, since the first-order terms include only one input, they must be equivalent to their single-input counterparts:

$$\mathcal{H}^{(1)}(u_k) = u_k \tag{2.52}$$

Next, consider the first type of second-order term, involving a second-degree function of a single input. These terms will have the form

$$\begin{aligned}\mathcal{H}^{(2)}(u_k, u_k) &= \mathcal{H}^{(2)}(u_k)\\ &= u_k^2 - 1\end{aligned} \tag{2.53}$$

and will be orthogonal to order-zero terms since

$$E[\mathcal{H}^{(2)}(u_k, u_k)\mathcal{H}^{(0)}(u_j)] = E[(u_k^2 - 1)\cdot 1]$$

and $E[u_k^2 - 1] = 0$. They will also be orthogonal to all first-order terms, since

$$\begin{aligned}E[\mathcal{H}^{(2)}(u_k, u_k)\mathcal{H}^{(1)}(u_j)] &= E[(u_k^2 - 1)u_j]\\ &= E[u_k^2 u_j] - E[u_j]\end{aligned}$$

Both terms in this expression are products of odd numbers of zero-mean Gaussian random variables, and so their expected values will be zero. Furthermore, any two second-order, single-input terms will be orthogonal. To see why, note that

$$\begin{aligned}E[\mathcal{H}^{(2)}(u_k, u_k)\mathcal{H}^{(2)}(u_j, u_j)] &= E[(u_k^2 - 1)(u_j^2 - 1)]\\ &= E[u_k^2 u_j^2] - E[u_j^2] - E[u_k^2] + 1\end{aligned}$$

The inputs are orthonormal, by assumption, and so $E[u_k^2] = E[u_j^2] = 1$. Furthermore, from equation (2.2) we obtain

$$E[u_k^2 u_j^2] = E[u_k^2]E[u_j^2] + 2E[u_k u_j]^2 = 1$$

and so $\mathcal{H}^{(2)}(u_k, u_k)$ and $\mathcal{H}^{(2)}(u_j, u_j)$ will be orthogonal (for $j \neq k$).

Now, consider second-order terms involving different inputs. The corresponding Hermite polynomial is simply the product of corresponding first-order terms,

$$\begin{aligned}\mathcal{H}^{(2)}(u_j, u_k) &= \mathcal{H}^{(1)}(u_j)\mathcal{H}^{(1)}(u_k) \qquad j \neq k\\ &= u_j u_k\end{aligned} \tag{2.54}$$

Since $E[u_k u_j] = 0$, this term will be orthogonal to the constant, order-zero term. It will also be orthogonal to any odd-order term because the resulting expectations will involve an odd number of Gaussian random variables. It remains to show that this term is orthogonal to the other second-order Hermite polynomial terms.

First consider the two-input terms,

$$\begin{aligned}E[\mathcal{H}^{(2)}(u_i, u_j)\mathcal{H}^{(2)}(u_k, u_l)] &= E[u_i u_j u_k u_l]\\ &= E[u_i u_j]E[u_k u_l] + E[u_i u_k]E[u_j u_l]\\ &\quad + E[u_i u_l]E[u_j u_k]\end{aligned}$$

Provided that we have neither $i = k, j = l$ nor $i = l, j = k$ (in which case the terms would be identical), each pair of expectations will contain at least one orthogonal pair of signals, and therefore equal zero. Thus, any distinct pair of two-input second-order terms will be orthogonal to each other.

Next consider the orthogonality of the two-input term to a single-input, second-order Hermite polynomial.

$$\begin{aligned} E[\mathcal{H}^{(2)}(u_i, u_j)\mathcal{H}^{(2)}(u_k, u_k)] &= E[u_i u_j (u_k^2 - 1)] \\ &= E[u_i u_j u_k^2] - E[u_i u_j] \\ &= E[u_i u_j]E[u_k^2] - 2E[u_i u_k]E[u_j u_k] - E[u_i u_j] \end{aligned}$$

Since u_i and u_j are orthogonal, the first and third terms will be zero. Similarly, u_k is orthogonal to at least one of u_i and u_j, so the second term will also be zero. Thus, equations (2.53) and (2.54) define two-input polynomials orthogonalized for orthonormal, zero-mean Gaussian inputs.

Similarly, there will be three types of third-order terms corresponding to the number of distinct inputs. The single-input terms are the same as the third-order, single-input Hermite polynomial term,

$$\begin{aligned} \mathcal{H}^{(3)}(u_k, u_k, u_k) &= \mathcal{H}^{(3)}(u_k) \\ &= u_k^3 - 3u_k \end{aligned} \tag{2.55}$$

Terms with distinct inputs are generated using products of lower-order single-input Hermite polynomials,

$$\begin{aligned} \mathcal{H}^{(3)}(u_j, u_j, u_k) &= \mathcal{H}^{(2)}(u_j)\mathcal{H}^{(1)}(u_k) \qquad j \neq k \\ &= (u_j^2 - 1)u_k \end{aligned} \tag{2.56}$$

$$\begin{aligned} \mathcal{H}^{(3)}(u_i, u_j, u_k) &= \mathcal{H}^{(1)}(u_i)\mathcal{H}^{(1)}(u_j)\mathcal{H}^{(1)}(u_k) \qquad i \neq j \neq k \\ &= u_i u_j u_k \end{aligned} \tag{2.57}$$

Higher-order terms are constructed in a similar fashion resulting in an orthogonal series known as the Grad–Hermite (Barrett, 1963) polynomial series. The same approach may be used to construct multiple-input polynomials based on the Tchebyshev polynomials.

2.6 NOTES AND REFERENCES

1. Bendat and Piersol (1986) and Papoulis (1984) provide more detailed discussions of probability theory and first-order correlation functions.
2. More information concerning higher-order correlation functions can be found in Marmarelis and Marmarelis (1978).
3. There are many texts dealing with linear regression. In particular, Beck and Arnold (1977) and Seber (1977) are recommended. Discussions more relevant to system identification can be found in Ljung (1999) and Söderström and Stoica (1989).

4. Beckmann (1973) contains a detailed discussion of several orthogonal polynomial systems, including both the Tchebyshev and Hermite polynomials. The classic reference for multiple-input, orthogonal polynomials is Barrett (1963).

2.7 PROBLEMS

1. Let x and y be zero-mean Gaussian random variables with variances σ_x^2 and σ_y^2, and covariance $E[xy] = \sigma_{xy}^2$. Evaluate

$$E[(xy)^2] \quad E[x^3 y] \quad E[x^3 y^2]$$

2. Let $x(t)$ and $n(t)$ be mutually independent zero-mean Gaussian signals with variances σ_x^2 and σ_n^2, respectively. Let $y(t) = 3x(t - \tau) + n(t)$. What is the cross-correlation between $x(t)$ and $y(t)$? What is the cross-correlation between $x(t)$ and $z(t) = x^3(t - \tau) + n(t)$?

3. Show that if x is a zero-mean unit variance Gaussian random variable, then the Hessian associated with fitting Hermite polynomials is given by

$$\mathbf{H}(i, j) = i!\delta_{i,j}$$

where $\delta_{i,j}$ is a Kronecker delta.

4. Prove the recurrence relation (2.51).

5. Prove the recurrence relation (2.48).

2.8 COMPUTER EXERCISES

1. Use the MATLAB NLID toolbox to create a RANDV object describing a zero-mean Gaussian random variable with a standard deviation of 0.5. Use it to generate a NLDAT object containing 1000 points with this distribution. Suppose that this signal is an unrectified EMG, measured in millivolts, and was sampled at 1000 Hz. Set the domain and range of your NLDAT object accordingly.

Plot the NLDAT object (signal) as a function of time, and check that the axes are labeled properly. Use PDF to construct a histogram for this signal. Construct the theoretical probability density, using PDF, and your RANDV object. Vary the number of bins used to compute the histogram, and compare the result with the theoretical distribution. How does the histogram change if you use 10,000 data points, or 100?

How many of the 1000 points in your original signal are greater than 1 (i.e., 2σ). How many are less than -1?

2. Use the RANDV object from question 1 to generate an NLDAT object containing white Gaussian noise. Use equation (2.3), and the variance of the original signal, to predict the expected value of the fourth power of this signal. Check the validity of this prediction by raising the signal to the fourth power and computing its mean. Try changing the length of the signal. How many points are necessary before equation (2.3) yields an accurate prediction of the mean of the fourth power of a Gaussian RV? Repeat this with the sixth and eighth power of the signal.

3. Generate an NLDAT object containing 10,000 samples of zero-mean, unit variance, white Gaussian noise. Assume that this signal was sampled at 200 Hz, and set the domain increment accordingly.

Compute and plot the autocorrelation between lags of 0 and 1 s. What is the amplitude of the peak at 0 lag? Estimate the standard deviation of the remaining points. Compute this ratio for various record lengths.

Filter your signal using a second-order Butterworth low-pass filter, with a cutoff frequency of 10 Hz, and plot the autocorrelation of the resulting signal. Compute the ratio of the peak, at zero lag, to the standard deviation of the points far distant from the peak. Where does the correlation first fall below the standard deviation of the "distant" points?

4. Generate a 10-s sample of a 20 Hz, 5 V peak to peak, square wave, sampled at 500 Hz. Compute its autocorrelation for lags of 0–0.5 s, and plot the result.

Generate a 10-s sample of zero-mean white Gaussian noise, with a standard deviation of 10 V, sampled at 500 Hz. Add this to the square wave, and plot the result as a function of time. Compute and plot the autocorrelation function of the noisy square wave.

Filter the noise record with a fourth-order low-pass filter with a 20 Hz cutoff. Add the low-pass filtered noise to the square wave. Plot the resulting signal, and compute and plot its autocorrelation.

5. Generate an NLDAT object containing 10,000 samples of zero-mean, unit variance, white Gaussian noise, and compute and plot its autocorrelation. Next, compute and plot the second-order autocorrelation function for this signal.

Square the Gaussian noise signal, and shift and rescale the result so that it has zero mean and unit variance. Compute and plot its first- and second-order autocorrelation functions. Repeat this procedure with the cube of the Gaussian noise signal. What, if anything, does the second-order autocorrelation tell you about the nonlinearity? Does it generalize to higher exponents?

6. Use POLYNOM to create an empty polynomial object. Set the type to "power," and set the coefficients so that it represents

$$y = 1 + 3u - 2u^2$$

Generate an object containing 1000 samples of white Gaussian noise, with mean 1 and variance 2. Apply the polynomial to the signal. Set the mean, standard deviation, and range properties of the polynomial to reflect those of the input data. Transform the polynomial object into a Hermite polynomial (orthogonalized using the statistics of the input NLDAT object). Transform it into a Tchebyshev polynomial. Examine the coefficients in each case. Use the Tchebyshev and Hermite polynomials to transform the input. Compare the results to the output of the power series. Are there any differences?

7. Use POLYNOM to fit polynomials to the input–output data generated in the previous problem. Fit polynomials of type power, hermite, and tcheb to the data, and compare your results. Add noise to the output, so that signal-to-noise ratio is 10 dB. Re-fit the polynomials using the noise corrupted output, and compare with the noise-free results.

Generate another set of input–output data. This time, let the input have a mean 1 and a variance of 4. Use each of the polynomials that you identified to predict the output of this new data set. When does the prediction remain accurate? When does it break down.