CHAPTER 4

MODELS OF NONLINEAR SYSTEMS

This chapter describes the different nonparametric structures used to model nonlinear systems and demonstrates how these model structures evolved from the linear models presented in Chapter 3. As before, a running example will be used to illustrate the different model structures and the nature of the insights they provide. The chapter concludes with a presentation of the Wiener–Bose model as a general structure for nonlinear modeling and shows how the other models described in the chapter may be regarded as special cases of the Wiener–Bose structure.

Methods for the identification of nonlinear models will be presented in Chapters 6–8, grouped according to estimation approach rather than model type.

4.1 THE VOLTERRA SERIES

Chapter 3 demonstrated that a linear system can be modeled by its impulse response, $h(\tau)$, and how because of superposition the response to an arbitrary input can be predicted using the convolution integral

$$y(t) = \int_0^T h(\tau)u(t-\tau)\,d\tau$$

Suppose, instead that the output of a system, $\mathbf{N}(u(t))$, is given by the convolution-like expression

$$\mathbf{N}(u(t)) = \int_0^T \int_0^T \mathbf{h}^{(2)}(\tau_1, \tau_2)u(t-\tau_1)u(t-\tau_2)\,d\tau_1\,d\tau_2 \tag{4.1}$$

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

What kind of system is this? The response to an impulse, $\delta(t)$, is given by

$$\begin{aligned}\mathbf{N}(\delta(t)) &= \int_0^T \int_0^T \mathbf{h}^{(\mathbf{2})}(\tau_1, \tau_2)\delta(t-\tau_1)\delta(t-\tau_2)\,d\tau_1\,d\tau_2 \\ &= \int_0^T \mathbf{h}^{(\mathbf{2})}(\tau_1, t)\delta(t-\tau_1)\,d\tau_1 \\ &= \mathbf{h}^{(\mathbf{2})}(t,t) \qquad (4.2)\end{aligned}$$

but the response to the scaled impulse, $\alpha\delta(t)$, is

$$\begin{aligned}\mathbf{N}(\alpha\delta(t)) &= \int_0^T \int_0^T \mathbf{h}^{(\mathbf{2})}(\tau_1, \tau_2)\alpha\delta(t-\tau_1)\alpha\delta(t-\tau_2)\,d\tau_1\,d\tau_2 \\ &= \alpha^2\mathbf{h}^{(\mathbf{2})}(t,t) \qquad (4.3)\end{aligned}$$

Thus, if the input is multiplied by a scale factor, α, the output is scaled by its square, α^2, that is,

$$\mathbf{N}(\alpha \cdot u(t)) = \alpha^2 \cdot \mathbf{N}(u(t))$$

The system does not obey the scaling property and thus it is not linear.

In fact, this is a *second-order* nonlinear system, since it operates on two "copies" of the input. As a result, scaling the input by a constant, α, generates an output scaled by the square of the constant, α^2. In general, an order n nonlinear system operates on n copies of the input; scaling the input by α will scale the output by α^n. In this context a "zero-order" nonlinear system produces a constant output, regardless of the input applied. Furthermore, a linear system can be regarded as a "first-order" nonlinear system that operates on one copy of the input and therefore generates an output that scales linearly with the input magnitude.

These ideas led Volterra (1959) to generalize the linear convolution concept to deal with nonlinear systems by replacing the single impulse response with a series of multidimensional integration kernels. The resulting Volterra series is given by

$$\boxed{\begin{aligned}y(t) = \sum_{q=0}^{\infty} \int_0^{\infty} \cdots \int_0^{\infty} & \mathbf{h}^{(\mathbf{q})}(\tau_1, \ldots, \tau_q)u(t-\tau_1)\ldots \\ & \ldots u(t-\tau_q)\,d\tau_1 \ldots d\tau_q \qquad (4.4)\end{aligned}}$$

The summation index, q, is the "order" of the nonlinear term. As with polynomial coefficients, the superscript in parentheses indicates the order of the term (kernel in this case). For example, setting $q = 0$ gives the zero-order term, which produces a constant output, irrespective of the input,

$$y^{(0)}(t) = \mathbf{h}^{(\mathbf{0})} \qquad (4.5)$$

Setting $q = 1$ gives the first-order, or linear, term generated by the linear convolution integral

$$y^{(1)}(t) = \int_0^T \mathbf{h^{(1)}}(\tau)u(t-\tau)\,d\tau \tag{4.6}$$

It may be tempting to think of the first-order kernel, $\mathbf{h^{(1)}}(\tau)$, as the system's impulse response. However, this will not be the case in general, since a system's impulse response will contain contributions from all its Volterra kernels. For example, the response to a unit impulse input of a system having both zero- and first-order terms will be the sum of the outputs of both kernels,

$$\mathbf{h^{(0)}} + \mathbf{h^{(1)}}(t)$$

rather than the response of the "linear kernel,"

$$\mathbf{h^{(1)}}(t)$$

Indeed, a system's first-order kernel will be equal to its impulse response only when all other kernels are equal to zero—that is, when the system is linear.

The second-order Volterra kernel describes nonlinear interactions between two copies of the input via the generalized convolution:

$$y^{(2)}(t) = \int_0^\infty \int_0^\infty \mathbf{h^{(2)}}(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2)\,d\tau_1\,d\tau_2 \tag{4.7}$$

Note that the second-order Volterra kernel is symmetric,

$$\mathbf{h^{(2)}}(\tau_1,\tau_2) = \mathbf{h^{(2)}}(\tau_2,\tau_1)$$

since interchanging the two indices is equivalent to interchanging the two copies of the input, $u(t-\tau_1)$ and $u(t-\tau_2)$. Furthermore, a single impulse can be thought of as a coincident pair of impulses. Thus, the diagonal elements of the second-order kernel, where $\tau_1 = \tau_2$, describe the component of the response that scales with the input magnitude squared.

Similarly, the nth-order Volterra kernel describes nonlinear interactions among n copies of the input. The kernel is symmetric with respect to any exchanges or permutations of integration variables, $\tau_1, \tau_2, \ldots, \tau_n$, and the kernel's diagonal ($\tau_1 = \tau_2 = \cdots = \tau_n$) corresponds to that component of the impulse response that scales with the nth power of the input amplitude.

4.1.1 The Finite Volterra Series

So far, the Volterra series has been presented in continuous time, and its output is computed using a series of generalized convolution integrals. Chapter 3 showed that IRFs can be represented in both continuous (3.8) and discrete (3.9) time. It is possible to compute the continuous-time convolution integral numerically, but it is more efficient to represent the system in discrete time and replace the integrals with summations. The same is true for the Volterra series, which can be represented in discrete time as

$$\boxed{y(t) = \sum_{q=0}^{\infty}(\Delta_t)^q \sum_{\tau_1=0}^{\infty} \cdots \sum_{\tau_q=0}^{\infty} \mathbf{h}^{(\mathbf{q})}(\tau_1, \ldots, \tau_q)u(t-\tau_1)\ldots u(t-\tau_q) \qquad (4.8)}$$

Henceforth, the sampling increment, Δ_t, will be taken as 1 to simplify the notation.

Furthermore, for any practical model, the maximum kernel order, Q, and the kernel memory length, T, must be finite. Since the sampling rate, $1/\Delta_t$, is also finite, the overall model will be finite-dimensional. The finite, discrete Volterra series model is given by

$$y(t) = \sum_{q=0}^{Q} \sum_{\tau_1=0}^{T-1} \cdots \sum_{\tau_q=0}^{T-1} \mathbf{h}^{(\mathbf{q})}(\tau_1, \ldots, \tau_q)u(t-\tau_1)\ldots u(t-\tau_q) \qquad (4.9)$$

4.1.1.1 Example: Peripheral Auditory Model

The application of the Volterra series will be illustrated by using it to model the nonlinear response of the cochlear nerve to an auditory input. Pfeiffer (1970) modeled this response using the structure shown in Figure 4.1; this consisted of two linear bandpass filters separated by a half-wave rectifier. He showed that this model reproduced important features of the peripheral auditory response, including two-tone suppression, and used it to study the effects of various inputs on the identification of linearized models of the auditory system. Although Pfeiffer did not associate the model's elements with underlying mechanisms, Swerup (1978) subsequently suggested that the first band-pass filter arose from the frequency selectivity of the basilar membrane while the hair cells were responsible for the rectification. Similar models have been used to describe other sensory encoders (French and Korenberg, 1991; French et al., 1993; Juusola et al., 1995).

This section will illustrate the Volterra series representation of this model. Similar illustrations will be provided for all model structures described in this chapter.

To transform the model of Figure 4.1 into a Volterra series representation, the nonlinearity must first be expressed as a polynomial. This was done by fitting a Tchebyshev polynomial to a half-wave rectifier over the output range expected for the first linear element, [−0.25 0.25]. An eighth-order polynomial was used since it provided the best compromise between model accuracy and complexity.

The LNL cascade model of Figure 4.1 was then converted to a Volterra series model using the method to be described in Section 4.3.3. Figure 4.2 shows the resulting first- and second-order Volterra kernels. Since the system is causal, kernel elements will be zero

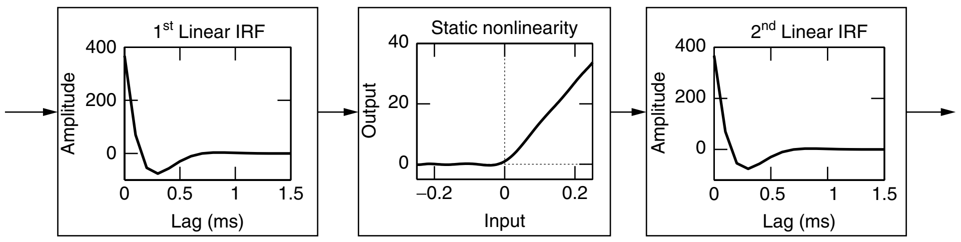

Figure 4.1 LNL cascade model of signal processing in the peripheral auditory system.

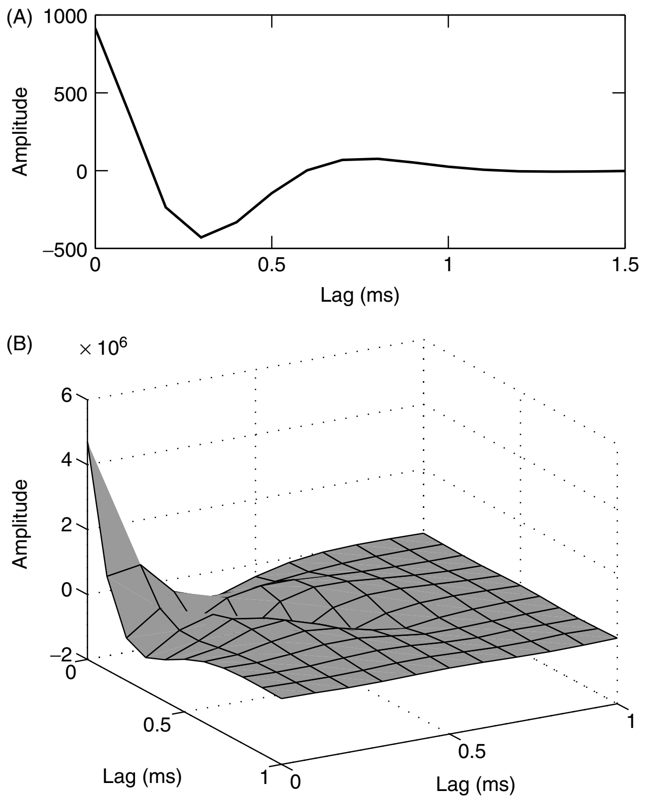

Figure 4.2 Volterra kernels computed for the peripheral auditory model. (A) First-order kernel. (B) Second-order kernel.

for any argument $(\tau_1 \ldots \tau_n)$ less than zero. Thus, the first-order kernel is displayed for positive lags only while only the first quadrant of the second-order kernel, where both τ_1 and τ_2 are ≥ 0, is shown. Moreover, as discussed above, the second-order kernel, shown in Figure 4.2B, is symmetric about its diagonal.

A full representation of this model would require kernels of order 0 through 8. However, there are no practical methods to display kernels of order greater than 3. Moreover, high-order Volterra kernels are not useful for computation/simulation purposes because the high-order convolutions involved are very expensive computationally. For example, using equation (4.9) to compute the output of a second-order (i.e., $q = 2$) kernel requires T^2 multiplications and T^2 additions per data point, or $2NT^2$ flops overall. The three-dimensional convolution needed to compute the output of a third-order kernel involves $2NT^3$ flops. Taking maximum advantage of the symmetry in the kernels (see Section 4.1.3 below) could reduce these flop counts to NT^2 and $NT^3/3$, respectively. Nevertheless, the computational burden still scales with T^q, where q is the kernel order.

Dimensions The peripheral auditory model was used to demonstrate a mechanism that explained several observed phenomena. It is unlikely that the scalings actually represent any physiological system. However, for the sake of argument, it will be assumed that the input signal is the voltage measured by a microphone and that the output is the instantaneous firing frequency of an auditory fiber. Thus, for purposes of discussion, the input will be measured in volts (V) and the output will be expressed in terms of pulses per second (pps).

The zero-order kernel of the model must produce an output, in pps, regardless of the input. Thus, the zero-order kernel must have the same dimensions as the output, namely pps.

As with the linear IRF, the first-order kernel is multiplied by the input and is integrated with respect to time. Thus dimensions of the first-order kernel must be pps/(Vs), which simplifies to 1/(Vs^2). A similar argument leads to the conclusion that the second-order kernel must have dimensions of pps/(Vs)^2, or 1/(V^{2}s^3). Since the dimensions associated with the Volterra kernels can easily become unwieldy, they are often simply reported as "amplitude," leaving the reader to determine the appropriate dimensions.

Assigning the correct dimensions to the elements of block structured models, such as that shown in Figure 4.1, involves subtleties that will be discussed in Section 4.3.

4.1.2 Multiple-Input Systems

The terms of the single-input Volterra series of equation (4.9) involve products of lagged inputs, $u(t-\tau_1)\cdot u(t-\tau_2)\dots\cdot u(t-\tau_q)$, multiplied by kernel values, $\mathbf{h}^{(\mathbf{q})}(\tau_1, \tau_2, \dots, \tau_q)$. Marmarelis and Naka (1974) extended this structure to model multiple-input systems by formulating a multiple-input Volterra series in which each term contains products of lagged values of one or more inputs.

Consider an order q nonlinear system with m inputs, $u_1(t), u_2(t), \dots, u_m(t)$. The first-order terms, analogous to those of equation (4.9), will involve only a single lagged input. Thus, there will be m first-order kernels, one for each input, with the same form as for the single-input model. Thus,

$$y^{(1)}(t) = \sum_{i=1}^{m} \left\{ \sum_{\tau=0}^{T-1} \mathbf{h}_{\mathbf{i}}^{(\mathbf{1})}(\tau) u_i(t-\tau) \right\} \tag{4.10}$$

where $\mathbf{h}_{\mathbf{i}}^{(\mathbf{1})}$ is the first-order kernel associated with the ith input, $u_i(t)$.

The second-order terms for the single-input system of equation (4.9) are given by

$$y^{(2)}(t) = \sum_{\tau_1=0}^{T-1} \sum_{\tau_2=0}^{T-1} \mathbf{h}^{(\mathbf{2})}(\tau_1, \tau_2) u(t-\tau_1) u(t-\tau_2)$$

The second-order terms for a multiple-input system are analogous, except that the pairs of lagged inputs in each product need not come from the same input. This gives rise to two types of second-order terms: self-terms, involving the same input (e.g., $u_1(t-\tau_1)u_1(t-\tau_2)$), and cross-terms, involving different inputs (e.g., $u_1(t-\tau_1)u_2(t-\tau_2)$). Thus, for an m input system, the complete second-order term would be

$$y^{(2)}(t) = \sum_{i=1}^{m} \left\{ \sum_{\tau_1=0}^{T-1} \sum_{\tau_2=0}^{T-1} \mathbf{h}_{\mathbf{ii}}^{(\mathbf{2})}(\tau_1, \tau_2) u_i(t-\tau_1) u_i(t-\tau_2) \right.$$
$$\left. + \sum_{j=i+1}^{m} \sum_{\tau_1=0}^{T-1} \sum_{\tau_2=0}^{T-1} \mathbf{h}_{\mathbf{ij}}^{(\mathbf{2})}(\tau_1, \tau_2) u_i(t-\tau_1) u_j(t-\tau_2) \right\} \tag{4.11}$$

The "self-terms" are generated by *self-kernels*, $\mathbf{h}_{\mathbf{ii}}^{(2)}$, that act on a single input. The "cross-terms" are produced by *cross-kernels*, $\mathbf{h}_{\mathbf{ij}}^{(2)}$, that act on *different* inputs.

Note that the second-order self-kernels, $\mathbf{h}_{\mathbf{ii}}^{(2)}(\tau_1, \tau_2)$ are symmetric, since interchanging the lag values corresponds to interchanging two copies of the same input. However, the cross-kernels, $\mathbf{h}_{\mathbf{ij}}^{(2)}(\tau_1, \tau_2)$, are *not* symmetric, in general, since interchanging two lag values is equivalent to interchanging two *different* input signals.

The complete multiple-input Volterra structure may be illustrated by considering a second-order system with two inputs: $u_1(t)$ and $u_2(t)$. Its Volterra series will include a zero-order kernel, two first-order kernels (one for each input), two second-order self-kernels, and one second-order cross-kernel, as illustrated in Figure 4.3. The overall output will be given by

$$
\begin{aligned}
y = h^{(0)} &+ h_1^{(1)} * u_1 + h_2^{(1)} * u_2 \\
&+ h_{11}^{(2)} * u_1 * u_1 + h_{12}^{(2)} * u_1 * u_2 + h_{22}^{(2)} * u_2 * u_2
\end{aligned}
$$

where the asterisk denotes a convolution integral and multiple asterisks denote multiple convolutions.

In principal, this structure may be extended for higher-order systems. However, as the system order and/or the number of inputs grows, the number of kernels explodes. By counting all permutations and combinations, it can be shown that an m input system will have $(m+q)!/m!q!$ kernels of order q. As a result, the multiple-input Volterra series approach is only practical for low-order systems with two or at most three inputs.

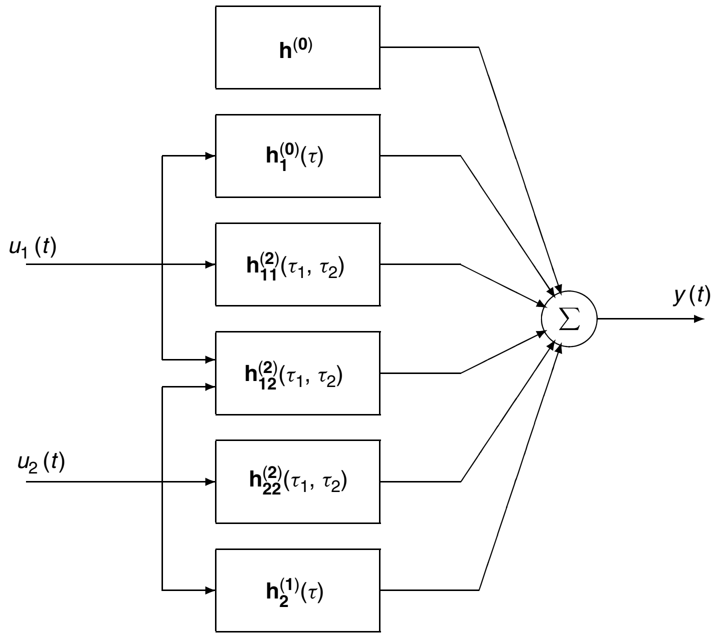
$\mathbf{h}^{(0)}$
$\mathbf{h}_1^{(0)}(\tau)$
$u_1(t)$
$\mathbf{h}_{11}^{(2)}(\tau_1, \tau_2)$
Σ
$y(t)$
$\mathbf{h}_{12}^{(2)}(\tau_1, \tau_2)$
$u_2(t)$
$\mathbf{h}_{22}^{(2)}(\tau_1, \tau_2)$
$\mathbf{h}_2^{(1)}(\tau)$

Figure 4.3 The elements of a two-input, second-order Volterra series model.

4.1.3 Polynomial Representation

There is a direct correspondence between Volterra series models and the multiple-input power series, $\mathcal{M}$. Consider, for example, the second-order Volterra kernel, whose output is given by

$$y^{(2)}(t) = \sum_{\tau_1,\tau_2=0}^{T-1} \mathbf{h}^{(2)}(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2)$$

which can be written as

$$y^{(2)}(t) = \sum_{\tau_1,\tau_2=0}^{T-1} \mathbf{h}^{(2)}(\tau_1,\tau_2)\mathcal{M}^{(2)}(u(t-\tau_1),u(t-\tau_2)) \tag{4.12}$$

The change in notation demonstrates that each element in $\mathbf{h}^{(2)}(\tau_1,\tau_2)$ corresponds to the coefficient associated with a polynomial term, since

$$\mathcal{M}^{(2)}(u(t-\tau_1),u(t-\tau_2)) = u(t-\tau_1)\cdot u(t-\tau_2)$$

Recall that the Volterra kernels are symmetric. This is also true for the terms of the multiple-input power series. For example,

$$\mathcal{M}^{(2)}(u(t-\tau_1),u(t-\tau_2)) = \mathcal{M}^{(2)}(u(t-\tau_2),u(t-\tau_1))$$

Therefore summing over a triangular domain, rather than the usual rectangular domain of the second-order convolution, will eliminate the redundant symmetric terms. Thus, equation (4.12) can be rewritten as

$$y^{(2)}(t) = \sum_{\tau_1=0}^{T-1}\sum_{\tau_2=\tau_1}^{T-1} c^{(2)}_{\tau_1,\tau_2}\mathcal{M}^{(2)}(u(t-\tau_1)u(t-\tau_2)) \tag{4.13}$$

where the coefficients, $c^{(2)}_{\tau_1,\tau_2}$, are related to the elements of the second-order Volterra kernel by

$$\begin{aligned} c^{(2)}_{\tau_k,\tau_k} &= \mathbf{h}^{(2)}(\tau_k,\tau_k) \\ c^{(2)}_{\tau_j,\tau_k} &= 2\mathbf{h}^{(2)}(\tau_j,\tau_k), \qquad j \neq k \end{aligned}$$

Thus, each unique polynomial term, $\mathcal{M}^{(2)}(u(t-\tau_1),u(t-\tau_2))$, has a corresponding polynomial coefficient, $c^{(2)}_{\tau_1,\tau_2}$, given by summing all corresponding (symmetrically placed and equal) kernel values.

Applying this change in notation to the entire finite Volterra series (4.9) gives

$$y(t) = \sum_{q=0}^{Q}\sum_{\tau_1=0}^{T-1}\sum_{\tau_2=\tau_1}^{T-1}\cdots\sum_{\tau_q=\tau_{q-1}}^{T-1} c^{(q)}_{\tau_1,\dots,\tau_q}\mathcal{M}^{(q)}(u(t-\tau_1)\dots u(t-\tau_q)) \tag{4.14}$$

where $\mathcal{M}^{(q)}(u(t-\tau_1)\dots u(t-\tau_q))$ is the product of the q lagged inputs $u(t-\tau_1)\dots u(t-\tau_q)$. Note the summation in equation (4.14) takes place over a "triangular" domain in q dimensions, rather than over the rectangular region of the usual formulation (4.9).

Therefore, the order q polynomial coefficient $c^{(q)}_{\tau_1,\ldots,\tau_q}$ will be equal to the sum of the (equal) kernel values for all permutations of the indices $\tau_1, \ldots, \tau_q$.

4.1.4 Convergence Issues(†)

The "traditional wisdom" regarding convergence of Volterra series is that any time-invariant, continuous, nonlinear operator can be approximated by a finite Volterra series. Indeed, this "folk theorem" [to borrow a term from Boyd and Chua (1985)] is not far removed from the early convergence and approximation results, summarized in Rugh (1981). This section will examine convergence issues more rigorously.

It is first necessary to consider the concept of continuity. A loose definition is that an operator is continuous if small changes in the input history lead to only small changes in the output. This can be stated more rigorously as follows. Let $u(t)$ be a bounded, continuous function defined on the real line, $u(t) \in \mathbf{C}(\mathbb{R})$, and let $\epsilon > 0$ be real. Let $v(t) \in \mathbf{C}(\mathbb{R})$ be another bounded continuous function. Define a "small" difference in the previous inputs as

$$\sup_{t \le 0} |u(t) - v(t)| < \delta \tag{4.15}$$

which states that the largest absolute difference between $u(t)$ and $v(t)$, for $t \le 0$, is less than some positive number δ. Then, if $\mathbf{N}$ is a causal, continuous operator, for every $\epsilon > 0$ there will be a positive real number, $\delta > 0$, such that for any bounded, real signal $v(t)$ that satisfies equation (4.15) the difference between the current outputs will be less than ϵ. Thus

$$|\mathbf{N}(u(0)) - \mathbf{N}(v(0))| < \epsilon \tag{4.16}$$

That is, an operator is continuous if for any positive ϵ, there is a number δ such that for any signal $v(t)$ satisfying (4.15), equation (4.16) also holds.

For example, the peak-hold system,

$$\mathbf{N}(u(t)) = \sup_{\tau \le t} u(\tau) \tag{4.17}$$

is a continuous operator. Choosing $\delta = \epsilon$, (4.15) implies (4.16) since a maximum difference of δ in the inputs can lead to a difference of at most δ in their peak values.

Section 4.1.3 showed that the Volterra series may be viewed as a multiple-input, polynomial expansion. Therefore, the Stone–Weierstrass theorem, which describes the approximation properties of polynomial expansions, can be used to establish the approximation properties of the Volterra series. Indeed, the early convergence results were obtained by applying the Stone–Weierstrass theorem directly to the Volterra series. Boyd and Chua (1985) gave a typical statement of the convergence results for continuous nonlinear operators.

Theorem 4.1 Let $\mathbf{K}$ be a compact subset of $\mathbf{L}^2[0, T]$, the space of square integrable functions defined on the closed interval $0 \le t \le T$, and let $\mathbf{N}$ be a causal continuous mapping from $\mathbf{K}$ to $\mathbf{C}[0, T]$, the space of bounded continuous functions on $[0, T]$, that is, $\mathbf{N} : \mathbf{K} \to \mathbf{C}[0, T]$. Then, there is a Volterra series operator, $\hat{\mathbf{N}}$, such that for all $\epsilon > 0$,

$u \in \mathbf{K}$, and $0 \leq t \leq T$, we have

$$|\mathbf{N}(u(t)) - \hat{\mathbf{N}}(u(t))| \leq \epsilon$$

This is a rigorous statement of the ideas stated informally above. For any continuous, time-invariant, nonlinear system $\mathbf{N}$, there is a Volterra series operator, $\hat{\mathbf{N}}$, such that the largest difference between its output, and that of the original system, is bounded by an arbitrarily small ϵ. However, this only holds for inputs belonging to $\mathbf{K}$, a compact subset of $\mathbf{L}^2[0, T]$. This restricts the input set to finite-duration, square-integrable signals. Persistent inputs such as unit steps or sinusoids, typically used for system identification, are not in the input set. Furthermore, the Dirac delta function is not square-integrable and so it is also excluded from the input set. Thus, these convergence results do not hold for many inputs of practical interest.

Any convergence analysis involves trade-offs between the generality of the input set and the model class. Extending the input set to include more useful signals requires placing additional restrictions on the class of models that can be approximated. The problem, then, is to find a combination of useful model classes and input sets for which convergence results can be derived. The foregoing analysis was derived for continuous operators for which a small change in the input history leads to only a small change in the current output. The time at which the input change occurs is not considered, so it is quite possible for a continuous system to have an infinite memory. The peak-hold system, considered above is such a case; it is continuous but has infinite memory. The convergence results stated above were derived for systems with infinite memory, but could only be obtained by severely restricting the input set to finite duration signals.

To expand the input set, Boyd and Chua (1985) developed a stronger notion of continuity, which they termed *fading memory*. This extended the previous definition of continuity by incorporating a weighting function, $w(t)$, which forces the system memory to "fade." The formal development proceeds as follows.

Let $w(t)$ be a decreasing function that maps the positive reals, $\mathbb{R}_+$, to the semi-open interval, (0, 1], that is, $w : \mathbb{R}_+ \to (0, 1]$, and let it have limit zero as t tends to infinity (i.e., $\lim_{t\to\infty} w(t) = 0$). Then, the operator $\mathbf{N}$ will have fading memory if

$$\sup_{t\leq 0} |u(t) - v(t)|w(-t) < \delta \to |\mathbf{N}(u(0)) - \mathbf{N}(v(0))| < \epsilon$$

Loosely translated, this says that small changes in the *weighted* inputs will lead to no more than small changes in the output. The weighting function is decreasing, with limit zero, so the system "forgets" inputs in the distant past. Based on this definition, it is evident that although the peak-hold system is continuous, it does not have fading memory. Static hysteresis is another example of a continuous, nonlinear system that does not have fading memory.

Using this definition, Boyd and Chua (1985) obtained the following convergence result, without invoking the Stone–Weierstrass theorem directly.

Theorem 4.2 Let $\mathbf{K}$ be any uniformly bounded equicontinuous set in $\mathbf{C}(\mathbb{R})$, and let $\mathbf{N}$ be any time-invariant, fading memory operator defined on $\mathbf{K}$. Then, for any $\epsilon > 0$, there is a Volterra operator, $\hat{\mathbf{N}}$, such that for all $u \in \mathbf{K}$, we obtain

$$|\mathbf{N}(u(t)) - \hat{\mathbf{N}}(u(t))| \leq \epsilon$$

Note that, in contrast to Theorem 4.1, this convergence result is not restricted to a finite time interval, $[0, T]$, nor is the input restricted to a compact subset of the input space.

Note that the Volterra series operator, $\hat{\mathbf{N}}$, in Theorem 4.2 is an infinite series of kernels and is infinite-dimensional in general. Boyd and Chua (1985) extended Theorem 4.2 to show that the output of any fading memory system can be approximated to within arbitrary accuracy by a finite-dimensional (i.e., finite-memory and finite-order) Volterra series operator. Of course, finite does not necessarily mean "computationally tractable."

4.2 THE WIENER SERIES

Wiener (1958) addressed the problem of estimating the Volterra series from input–output records using analog computations. He recognized that if the Volterra series were orthogonalized, its coefficients could be estimated sequentially using projections, an operation easily implemented with analog circuitry. Wiener's original derivation focused on this approach, and it used a Gram–Schmidt procedure to perform the orthogonalization in continuous time. The analogous, discrete time derivation forms the basis of most developments of the Wiener series and has been presented in numerous textbooks. This derivation will not be repeated here. The interested reader is referred to Wiener's original manuscript (Wiener, 1958) or to the texts by Marmarelis and Marmarelis (1978) or Schetzen (1981) for the complete derivation. Rather, the major results will be presented and then it will be shown how the Wiener series can be viewed as an orthogonal expansion of the Volterra series.

The Wiener series models a system's response with a set of operators*

$$\boxed{y(t) = \sum_{q=0}^{\infty} G_q[\mathbf{k^{(q)}}(\tau_1, \ldots, \tau_q); u(t'), t' \le t] \qquad (4.18)}$$

which apply a set of kernels, the Wiener kernels $\mathbf{k^{(q)}}$, $q = 0 \ldots \infty$, to the input history, $u(t'), t' \le t$, to generate the output, $y(t)$.

The output of the zero-order operator, $G_0[\mathbf{k^{(0)}}; u(t)]$, is a constant,

$$G_0[\mathbf{k^{(0)}}; u(t)] = \mathbf{k^{(0)}} \qquad (4.19)$$

and therefore is independent of the input.

The output of the first-order Wiener operator is given by

$$G_1[\mathbf{k^{(1)}}(\tau); u(t)] = \sum_{\tau=0}^{T-1} \mathbf{k^{(1)}}(\tau) u(t-\tau) \qquad (4.20)$$

which is the same linear convolution as for the first-order Volterra kernel [see equation (4.6)]. If the input is a zero-mean, Gaussian process, then $G_1[\mathbf{k^{(1)}}(\tau); u(t)]$ will also

*Traditionally, the elements in the Wiener series have been called functionals rather than operators. However, the G_q map functions onto other functions, and are therefore *operators*. Strictly speaking, *functionals* map functions onto values. (Boyd and Chua, 1985).

be a zero-mean Gaussian process; consequently it will be orthogonal to any constant, including the output of the zero-order operator.

The output of the second-order Wiener operator is given by

$$G_2[\mathbf{k^{(2)}}(\tau_1, \tau_2); u(t)] = \sum_{\tau_1=0}^{T-1} \sum_{\tau_2=0}^{T-1} \mathbf{k^{(2)}}(\tau_1, \tau_2) u(t-\tau_1) u(t-\tau_2) - \sigma_u^2 \sum_{\tau=0}^{T-1} \mathbf{k^{(2)}}(\tau, \tau) \tag{4.21}$$

The expected value of the output of G_2 is

$$\begin{aligned} E\Big[G_2[\mathbf{k^{(2)}}(\tau_1, \tau_2); u(t)]\Big] &= \sum_{\tau_1=0}^{T-1} \sum_{\tau_2=0}^{T-1} \mathbf{k^{(2)}}(\tau_1, \tau_2) E[u(t-\tau_1) u(t-\tau_2)] \\ &\quad - \sigma_u^2 \sum_{\tau=0}^{T-1} \mathbf{k^{(2)}}(\tau, \tau) \\ &= 0 \end{aligned}$$

since $E[u(t-\tau_1)u(t-\tau_2)] = \sigma_u^2 \delta_{\tau_1,\tau_2}$. Thus, G_2 will be orthogonal to G_0, and any other constant. Furthermore, all terms in $G_2 \cdot G_1$ will involve products of odd numbers of Gaussian random variables, and therefore it will have expected values of zero. Consequently, the outputs of G_2 and G_1 will be orthogonal also.

Note that the output of an odd-order operator is automatically orthogonal to that of any even-order operator, and vice versa. Thus, constructing a Wiener operator of order q, only requires orthogonalization against operators of order $q - 2m$, for all integers m between 1 and $q/2$. The general form of the order q Wiener operator is given by (Marmarelis and Marmarelis, 1978; Rugh, 1981)

$$\begin{aligned} y^{(q)}(t) &= G_q[\mathbf{k^{(q)}}(\tau_1, \ldots, \tau_q); u(t)] \\ &= q! \sum_{m=0}^{\lfloor q/2 \rfloor} \frac{(-1)^m \sigma_u^m}{m! 2^m (q-2m)!} \sum_{\tau_1=0}^{T-1} \cdots \sum_{\tau_{q-2m}=0}^{T-1} \\ &\quad \times \left(\sum_{j_1=0}^{T-1} \cdots \sum_{j_m=0}^{T-1} \mathbf{k^{(q)}}(\tau_1, \ldots, \tau_{q-2m}, j_1, j_1, \ldots, j_m, j_m) \right) \\ &\quad \times u(t-\tau_1) \ldots u(t-\tau_{q-2m}) \end{aligned} \tag{4.22}$$

which will be orthogonal to all lower-order operators, both even and odd.

4.2.1 Orthogonal Expansion of the Volterra Series

Section 4.1.3 showed that the Volterra series can be regarded as a multiple-input power series. As with any other polynomial expression, this power series can be replaced with an orthogonal polynomial. Indeed, Sections 2.5.1–2.5.2 showed that the use of orthogonal polynomials may avoid the numerical conditioning problems involved in estimating the coefficients of power series. Consequently, it is to be expected that expanding the Volterra series with orthogonal polynomials would have similar advantages. For the Wiener series, where the input is assumed to be white Gaussian noise, the Hermite polynomials are the family of choice. Therefore, this section will expand the Volterra series using Hermite orthogonal polynomials and show that the resulting series is equivalent to the Wiener series.

If the input to a Volterra series is a white, Gaussian process with unit variance, then the polynomial terms, $\mathcal{M}^{(q)}(u(t-\tau_1)\dots u(t-\tau_q))$, in equation (4.14) will involve only products of orthonormal, Gaussian random variables. Consequently, by expanding the Volterra kernels using the multiple-input Grad–Hermite polynomials, discussed in Section 2.5.5, equation (4.14) can be rewritten as

$$y(t) = \sum_{q=0}^{Q}\sum_{\tau_1=0}^{T-1}\cdots\sum_{\tau_q=\tau_{q-1}}^{T-1} \gamma^{(q)}_{\tau_1,\dots,\tau_q}\mathcal{H}^{(q)}(u(t-\tau_1),\dots,u(t-\tau_q)) \tag{4.23}$$

where the terms of different orders will now be orthogonal. Note that $\gamma^{(q)}$ is used to represent the Hermite polynomial coefficients, to distinguish them from $c^{(q)}$, the coefficients of the power-series expansion in equation (4.14). If $u(t)$ is not white, or is non-Gaussian, or doesn't have unit variance, equation (4.23) will remain a valid model but the terms will no longer be mutually orthogonal.

If the input does not have unit variance, it is possible to restore orthogonality by expanding the polynomials using normalized signals and then correcting for the resulting scaling. To do so, make the substitution

$$\mathcal{H}^{(q)}(u(t-\tau_1),\dots,u(t-\tau_q)) \longrightarrow \sigma_u^q\mathcal{H}^{(q)}\left(\frac{u(t-\tau_1)}{\sigma_u},\dots,\frac{u(t-\tau_q)}{\sigma_u}\right) \tag{4.24}$$

resulting in the *normalized form* of the Grad–Hermite polynomials, $\mathcal{H}^{(q)}_{\mathcal{N}}$. For example, the normalized form of the second-order Hermite polynomial is

$$\begin{aligned}\mathcal{H}^{(2)}_{\mathcal{N}}(u(t)) &= \sigma_u^2\mathcal{H}^{(2)}(u(t)/\sigma_u)\\ &= \sigma_u^2((u(t)/\sigma_u)^2-1)\\ &= u^2(t)-\sigma_u^2\end{aligned}$$

where the subscript $\mathcal{N}$ denotes the normalized polynomials.

Thus, for a white, Gaussian signal with variance σ_u^2, the orthogonal expansion will be

$$y(t) = \sum_{q=0}^{Q}\sum_{\tau_1=0}^{T-1}\cdots\sum_{\tau_q=\tau_{q-1}}^{T-1} \gamma^{(q)}_{\tau_1,\dots,\tau_q}\mathcal{H}^{(q)}_{\mathcal{N}}(u(t-\tau_1),\dots,u(t-\tau_q)) \tag{4.25}$$

Note that both the polynomials, $\mathcal{H}^{(q)}_{\mathcal{N}}$, and the coefficients $\gamma^{(q)}$, will depend explicitly on the input variance, σ_u^2, due to the normalization in equation (4.24).

Equation (4.25) can be transformed by scaling and arranging the Hermite polynomial coefficients, $\gamma^{(q)}$, so that they become the elements of the system's Wiener kernels. For example, the output of the second-order Hermite polynomial terms,

$$\begin{aligned} y^{(2)}(t) &= \sum_{\tau_1=0}^{T-1} \sum_{\tau_2=\tau_1}^{T-1} \gamma^{(2)}_{\tau_1,\tau_2} \mathcal{H}^{(2)}_{\mathcal{N}}(u(t-\tau_1), u(t-\tau_2)) \\ &= \sum_{\tau_1=0}^{T-1} \sum_{\tau_2=\tau_1}^{T-1} \gamma^{(2)}_{\tau_1,\tau_2} \left(u(t-\tau_1)u(t-\tau_2) - \sigma_u^2 \delta_{\tau_1,\tau_2} \right) \end{aligned}$$

can be rewritten as the output of a second-order Wiener kernel, $\mathbf{k}^{(2)}(\tau_1, \tau_2)$, as follows:

$$y^{(2)}(t) = \sum_{\tau_1=0}^{T-1} \sum_{\tau_2=0}^{T-1} \mathbf{k}^{(2)}(\tau_1, \tau_2) u(t-\tau_1) u(t-\tau_2) - \sigma_u^2 \sum_{\tau=0}^{T-1} \mathbf{k}^{(2)}(\tau, \tau) \tag{4.26}$$

where the kernel elements are related to the coefficients of the Hermite polynomial by

$$\mathbf{k}^{(2)}(\tau_1, \tau_2) = \begin{cases} \dfrac{\gamma^{(2)}_{\tau_1,\tau_2}}{2}, & \tau_1 < \tau_2 \\ \gamma^{(2)}_{\tau_1,\tau_2}, & \tau_1 = \tau_2 \\ \dfrac{\gamma^{(2)}_{\tau_2,\tau_1}}{2}, & \tau_1 > \tau_2 \end{cases} \tag{4.27}$$

The final term in equation (4.26), which is the term that normalizes the second-order Wiener kernel against all zero-order operators, is generated by the σ_u^2 term in the single-input second-order Hermite polynomial

$$\mathcal{H}^{(2)}_{\mathcal{N}}(u(t)) = u^2(t) - \sigma_u^2$$

Note that the polynomial coefficients corresponding to off-diagonal kernel elements are divided by 2, since their contributions are split between two elements.

This demonstrates that the second-order term in the Hermite expansion of the Volterra series, (4.26), is identical to the second-order Wiener operator (4.21). Similar results hold for the rest of the Wiener series. Thus, the Wiener operators can be viewed as an expansion of the Volterra series onto a basis of Hermite polynomials.

4.2.2 Relation Between the Volterra and Wiener Series

The Volterra kernels for a system may be computed from its Wiener kernels, provided that the complete set is available. Following Rugh (1981), expand equation (4.18) using equation (4.22):

$$\begin{aligned} y(t) &= \sum_{q=0}^{\infty} G_q[\mathbf{k}^{(\mathbf{q})}(\tau_1, \ldots, \tau_q); u(t)] \\ &= \sum_{q=0}^{\infty} q! \sum_{m=0}^{\lfloor q/2 \rfloor} \frac{(-1)^m \sigma_u^m}{m! 2^m (q-2m)!} \sum_{\tau_1=0}^{T-1} \cdots \sum_{\tau_{q-2m}=0}^{T-1} \end{aligned}$$

$$\times \left(\sum_{j_1=0}^{T-1} \cdots \sum_{j_m=0}^{T-1} \mathbf{k^{(q)}}(\tau_1, \ldots, \tau_{q-2m}, j_1, j_1, \ldots, j_m, j_m) \right)$$

$$\times\, u(t-\tau_1)\ldots u(t-\tau_{q-2m}) \tag{4.28}$$

From equation (4.4), it is evident that only the nth-order Volterra kernel is convolved with n shifted copies of the input. Therefore, the kernel can be obtained from equation (4.22) by collecting all terms where $q-2m=n$, since only these will include exactly n (shifted) copies of the input $u(t)$. This gives

$$\mathbf{h^{(n)}}(\tau_1, \ldots, \tau_n) = \sum_{q=0}^{\infty} \frac{(-1)^q (n+2q)!\sigma_u^{2q}}{n!q!2^q}$$

$$\times \sum_{j_1=0}^{T-1} \cdots \sum_{j_q=0}^{T-1} \mathbf{k^{(n+2q)}}(\tau_1, \ldots, \tau_n, j_1, j_1, \ldots, j_q, j_q) \tag{4.29}$$

The inverse relationship also exists (Rugh, 1981). Given all a system's Volterra kernels, the corresponding Wiener kernels are

$$\mathbf{k^{(n)}}(\tau_1, \ldots, \tau_n) = \sum_{q=0}^{\infty} \frac{(n+2q)!\sigma_u^{2q}}{n!q!2^q}$$

$$\times \sum_{j_1=0}^{T-1} \cdots \sum_{j_q=0}^{T-1} \mathbf{h^{(n+2q)}}(\tau_1, \ldots, \tau_n, j_1, j_1, \ldots, j_q, j_q) \tag{4.30}$$

4.2.3 Example: Peripheral Auditory Model — Wiener Kernels

The Wiener kernels of the peripheral auditory model were computed directly from the elements of the LNL model, shown in Figure 4.1, by transforming the static nonlinearity into a Hermite polynomial and applying the methods to be described in Sections 4.3.3 and 4.5.4.*

These Wiener kernels could also have been generated from the model's Volterra kernels using equation (4.30). However, this would have required all the Volterra kernels of the system (of order 0 though 8), and consequently the more direct approach was used.

Figure 4.4 shows the first- and second-order Wiener kernels of the peripheral auditory model, evaluated for high- and low-power inputs. Note that the first-order kernels, shown in Figures 4.4A and 4.4B, do not vary with the input variance. In contrast, the second-order kernels, shown in Figures 4.4C and 4.4D, change with the power level.

*The method described in Section 4.3.3 computes the Volterra kernels of an LNL model. Replacing the power series nonlinearity with an equivalent Hermite polynomial transforms the Volterra kernels into Wiener kernels as shown in Section 4.5.4.

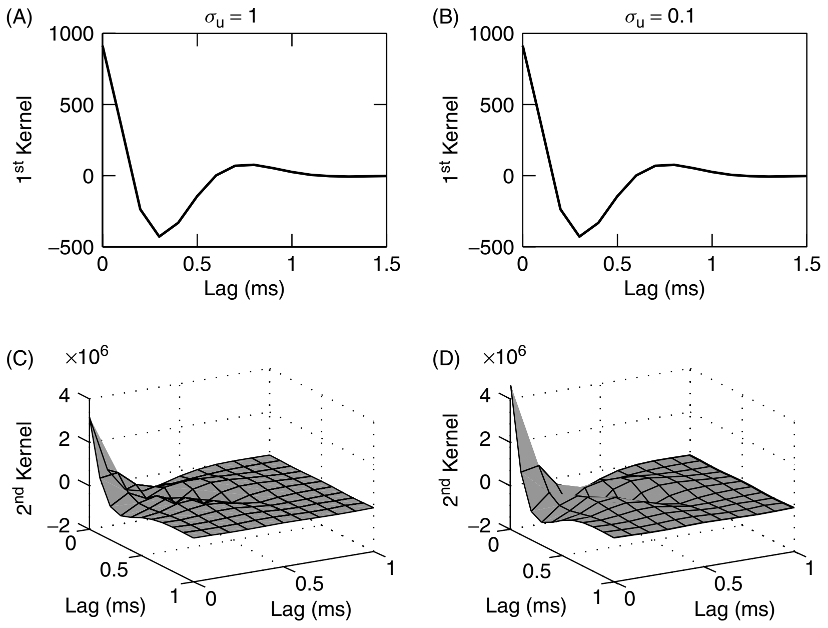

Figure 4.4 First- and second-order Wiener kernels of the peripheral auditory model evaluated at two different input power levels. The first-order kernels for (A) high- and (B) low-input power levels. The second-order kernels for (C) high power and (D) low power.

To understand what this implies, use equation (4.30) to express the first-order Wiener kernel in terms of the Volterra kernels:

$$\mathbf{k}^{(1)}(\tau) = \sum_{j=0}^{\infty} \frac{(1+2j)!\sigma_u^{2j}}{j!2^j} \cdot \sum_{\sigma_1=0}^{T-1} \cdots \sum_{\sigma_j=0}^{T-1} \mathbf{h}^{(1+2j)}(\tau, \sigma_1, \sigma_1, \ldots, \sigma_j, \sigma_j) \tag{4.31}$$

where each term in the sum is multiplied by the input variance, σ_u^2, raised to the jth power. The only term that does not depend on the input amplitude is the $j = 0$ (i.e., first-order) term. Therefore, for the first-order Wiener kernel to be independent of the input variance, all terms in the sum where $j > 0$ must be zero or cancel. Consequently, the observation that the first-order kernel, shown in Figure 4.4, is independent of input variance, suggests that there are no odd-order dynamics in the system, other than the first-order term (i.e., $\mathbf{h}^{(1+2j)} \equiv 0$ for $j \geq 1$). Of course, the observed invariance could result from a coincidental cancellation of all high-order terms for the particular power levels examined. This possibility could be tested by estimating the kernels using additional input power levels.

Using a similar argument, it can be shown that if there are no high-order even dynamics, then the second-order Wiener kernel will be independent of the input power level. Thus, the change in amplitude of the second-order kernel with power level, evident in Figure 4.4, indicates that higher-order even dynamics are present.

The overall scaling of the kernels suggests that the nonlinearity in the system contains high-order even terms but no significant odd terms of order greater than one. This is consistent with the model; the only significant high-order terms in the polynomial used to approximate the half-wave rectifier in the peripheral auditory model were even.

4.2.4 Nonwhite Inputs

The Wiener functionals were orthogonalized for white, Gaussian inputs. However, in many experiments the inputs actually applied are colored, so the Wiener functionals must be reorthogonalized with respect to actual input spectrum. What does this involve?

The zero-order functional is independent of the input and so does not change for a colored input. The first-order functional, given in equation (4.20), also remains unchanged since its output will remain zero-mean and therefore orthogonal to the zero-order functional.

The situation is more complex for the second-order functional whose output must be orthogonal to those of the zero- and first-order functionals. The latter is automatic. However, for the zero- and second-order functionals to be orthogonal, $G_2[\mathbf{k}^{(2)}(\tau_1,\tau_2); u(t)]$ must have zero mean. However, the expected value of the output of the second-order kernel is

$$E\left[\sum_{\tau_1,\tau_2=0}^{T-1} \mathbf{k}^{(2)}(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2)\right] = \sum_{\tau_1,\tau_2=0}^{T-1} \mathbf{k}^{(2)}(\tau_1,\tau_2)\phi_{uu}(\tau_1-\tau_2)$$

Therefore, the second-order Wiener functional for a nonwhite input must be

$$\begin{aligned} G_2[\mathbf{k}^{(2)}(\tau_1,\tau_2); u(t)] &= \sum_{\tau_1,\tau_2=0}^{T-1} \mathbf{k}^{(2)}(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2) \\ &\quad - \sum_{\tau_1,\tau_2=0}^{T-1} \mathbf{k}^{(2)}(\tau_1,\tau_2)\phi_{uu}(\tau_1-\tau_2) \end{aligned} \tag{4.32}$$

Note that for a white input, the input autocorrelation will be a discrete delta function, and so equation (4.32) reduces to equation (4.21). The higher-order Wiener functionals for nonwhite inputs may be derived similarly.

4.3 SIMPLE BLOCK STRUCTURES

Functional expansions, such as the Volterra and Wiener series, can represent a wide variety of systems. However, the expressions are often unwieldy, even for relatively low-order systems. This section describes several simple models, consisting of interconnections of alternating linear dynamic (L) and zero-memory nonlinear (N) subsystems, that provide very efficient descriptions for a more limited class of nonlinear systems.

4.3.1 The Wiener Model

The Wiener (LN) model (Hunter and Korenberg, 1986; Rugh, 1981), shown in Figure 4.5, consists of a linear dynamic element in cascade with a static nonlinearity. If the static

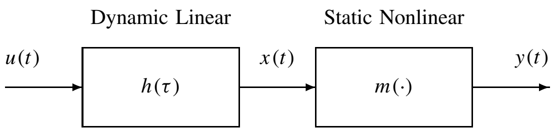

Figure 4.5 Block diagram of a Wiener system.

nonlinearity is represented by a power series

$$m(x(t)) = \sum_{q=0}^{Q} c^{(q)} x^q(t) \tag{4.33}$$

then the output of the Wiener model will be

$$y(t) = \sum_{n=0}^{Q} c^{(q)} \left(\sum_{\tau=0}^{T-1} h(\tau)u(t-\tau) \right)^q \tag{4.34}$$

$$= \sum_{q=0}^{Q} c^{(q)} \left(\sum_{\tau_1=0}^{T-1} \cdots \sum_{\tau_q=0}^{T-1} h(\tau_1)\ldots h(\tau_q) \right.$$
$$\left. \cdot\, u(t-\tau_1)\ldots u(t-\tau_q) \right) \tag{4.35}$$

4.3.1.1 Relationship to the Volterra Series Consider the Volterra series representation of a Wiener cascade where the output of the order q kernel is given by

$$y^{(q)}(t) = \sum_{\tau_1=0}^{T-1} \cdots \sum_{\tau_q=0}^{T-1} \mathbf{h}^{(\mathbf{q})}(\tau_1, \ldots, \tau_q) u(t-\tau_1)\ldots u(t-\tau_q) \tag{4.36}$$

The only term in equation (4.35) involving the product of q copies of the input is

$$c^{(q)} \left(\sum_{\tau_1=0}^{T-1} \cdots \sum_{\tau_q=0}^{T-1} h(\tau_1)\ldots h(\tau_q) u(t-\tau_1)\ldots u(t-\tau_q) \right) \tag{4.37}$$

Equation (4.37) of the Wiener cascade and equation (4.36) of the Volterra series are the only terms involving convolutions of an order q kernel with q copies of the input, $u(t)$. Therefore, they must generate the same output, and consequently the two convolution kernels must be equal. Therefore,

$$\mathbf{h}^{(\mathbf{q})}(\tau_1, \ldots, \tau_q) = c^{(q)} h(\tau_1) h(\tau_2) \ldots h(\tau_q) \tag{4.38}$$

Thus, the order q Volterra kernel of a Wiener system is proportional to the product of q copies of the IRF of its linear element.

4.3.1.2 Arbitrary Division of the Gain The overall gain of the Wiener model depends on both the polynomial coefficients and the gain of the linear element. The distribution of gain is arbitrary; if the gain of the linear element is scaled by k, the Volterra kernels of equation (4.38) will become

$$\begin{aligned}\mathbf{h}^{(\mathbf{q})}(\tau_1, \dots, \tau_q) &= c^{(q)}(kh(\tau_1)) \dots (kh(\tau_q))\\ &= c^{(q)}k^q h(\tau_1) \dots h(\tau_q)\end{aligned}$$

Consequently, if the polynomial coefficients $c^{(q)}$ are scaled by $1/k^q$, then

$$\begin{aligned}\mathbf{h}^{(\mathbf{q})}(\tau_1, \dots, \tau_q) &= \frac{c^{(q)}k^q}{k^q} h(\tau_1) \dots h(\tau_q)\\ &= c^{(q)} h(\tau_1) \dots h(\tau_q)\end{aligned}$$

so the Volterra kernels will remain constant.

Thus, the Wiener model has one degree of freedom that does not affect its input–output behavior. This can make it difficult to compare models obtained with different data sets or under different operating conditions. Consequently, it is customary to normalize the elements of a Wiener model, often by setting the norm of the linear element to 1, and to incorporate this normalization into the model definition.

Due to this arbitrary division of the gain between the linear and nonlinear elements, the dimensions of the intermediate signal, $x(t)$, are also arbitrary. These are sometimes listed as “arbitrary units” (A.U.).

This also affects the dimensions of linear and nonlinear elements. For example, if a Wiener cascade were used to represent the peripheral auditory model, whose input was measured in volts and whose output is measured in pps, the linear element would transform volts into A.U. and therefore have dimensions of A.U./Vs. The nonlinearities abscissa would be measured in A.U., and its ordinate would be measured in pps.

4.3.1.3 Testing Volterra Kernels for the Wiener Structure From equation (4.38) it is evident that any one-dimensional slice, taken parallel to an axis of a Volterra kernel of a Wiener cascade, will be proportional to the impulse response of the linear subsystem. For example, evaluating the third-order kernel of a Wiener system,

$$\mathbf{h}^{(\mathbf{3})}(\tau_1, \tau_2, \tau_3) = c^{(3)} h(\tau_1)h(\tau_2)h(\tau_3)$$

along the one-dimensional slice where τ_1 and τ_2 are held constant, $\tau_1 = k_1, \tau_2 = k_2$, will give

$$\begin{aligned}\mathbf{h}^{(\mathbf{3})}(k_1, k_2, \tau) &= c^{(3)} h(k_1)h(k_2)h(\tau)\\ &= \alpha h(\tau)\end{aligned}$$

This property can be used to test an unknown system for the Wiener structure. It is a necessary condition for the system to have the Wiener structure; if it is not satisfied, then the system cannot be represented by a Wiener cascade. However, it is not a sufficient condition, because kernels of other structures may demonstrate the same behavior.

To illustrate this, consider the Wiener system, shown in Figure 4.6, consisting of a linear IRF followed by an eighth-order polynomial approximation to a half-wave

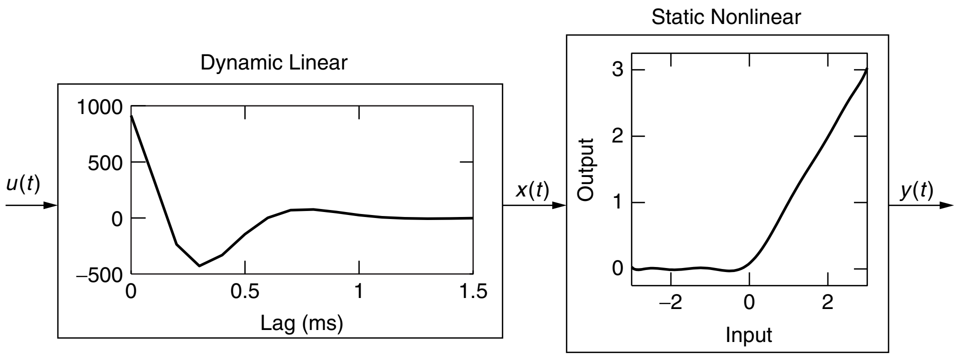

Figure 4.6 A Wiener system having the same first-order Volterra kernel and static nonlinearity as the peripheral auditory processing model shown in Figure 4.1.

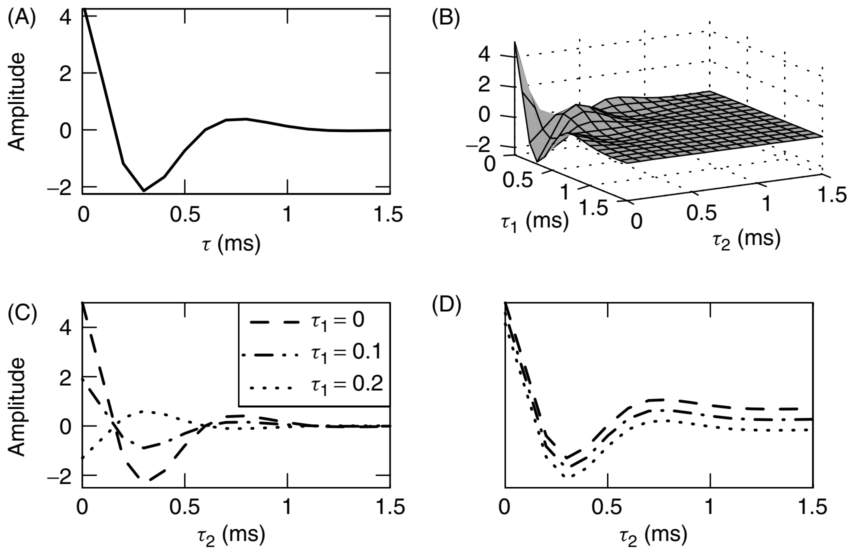

Figure 4.7 Testing for the Wiener structure. (A) First- and (B) second-order Volterra kernels of the Wiener system shown in Figure 4.6. (C) The first three slices ($\tau_1 = 0$, 0.1, 0.2 ms) of the second-order kernel. (D) The same three slices normalized, as well as offset vertically from one and other. Comparing A, D, and Figure 4.6 demonstrates that the first-order kernel and the second-order kernel slices are proportional to the IRF of the linear element.

rectifier. For comparison purposes, the linear IRF was set equal to the first-order kernel of the peripheral auditory processing model used as an example throughout this chapter. Figures 4.7A and 4.7B show the first and second-order Volterra kernels, respectively. Figure 4.7C shows the first three slices ($\tau_1 = 0$, 0.1, 0.2 ms) of the second-order kernel. Figure 4.7D shows the same slices normalized to the same amplitude and polarity and then offset slightly to facilitate comparison. This makes it evident that the three kernel slices are proportional to each other and to the first-order kernel.

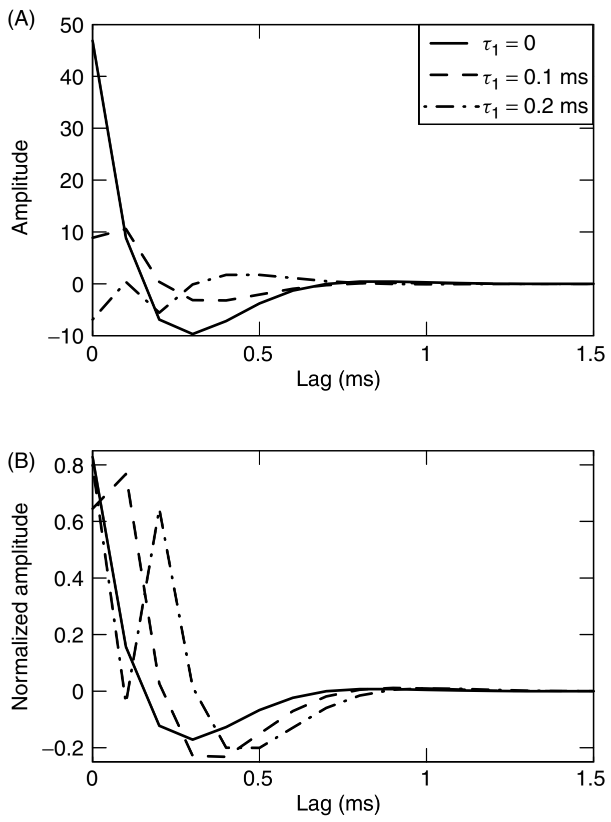

Figure 4.8 (A) The first three slices of the second-order Volterra kernel from Figure 4.2, computed for the peripheral auditory model. (B) The same slices normalized to their peak-to-peak amplitude. They are not proportional to each other and so the system cannot be a Wiener cascade.

4.3.1.4 Example: Peripheral Auditory Model—Test for Wiener Structure

The same analysis will now be applied to the simplified model of peripheral auditory signal processing, shown in Figure 4.2, and examined throughout this chapter. Figure 4.8A shows the first three slices ($\tau_1 = 0$, 0.1, and 0.2 ms, respectively) of the second-order Volterra kernel. Figure 4.8B shows the same slices, rescaled to the same peak-to-peak amplitude. It is apparent that the scaled slices are different and therefore that the kernel slices are not proportional to each other. Consequently, the underlying system cannot have the Wiener structure; this is reassuring since the model actually has a LNL structure.

The Volterra kernels in this example were generated analytically from the model so that any discrepancy between the normalized kernel slices is enough to eliminate the Wiener cascade as a potential model structure. In practical applications, the structural test will be applied to kernel estimates and must account for their statistical uncertainty.

4.3.2 The Hammerstein Model

Figure 4.9 shows the structure of a Hammerstein (NL) model (Hunter and Korenberg, 1986; Rugh, 1981) consisting of a static nonlinearity followed by a dynamic linear system.

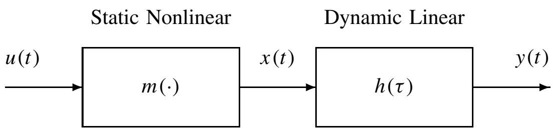

Figure 4.9 Block diagram of a Hammerstein system.

If the nonlinearity can be represented by a power series [see equation (4.33)], the output of a Hammerstein system can be written as

$$y(t) = \sum_{\tau=0}^{T-1} h(\tau) \left\{ \sum_{q=0}^{Q} c^{(q)} u^q(t-\tau) \right\} \tag{4.39}$$

4.3.2.1 Relationship to the Volterra Series To determine the Volterra kernels for this model, manipulate equation (4.39) into a form similar to equation (4.9) by expanding the powers of the input as

$$u^q(t-\tau) = u(t-\tau)u(t-\tau_2)\ldots u(t-\tau_q)\delta_{\tau,\tau_2}\ldots\delta_{\tau,\tau_q}$$

where $\delta_{i,j}$ is a Kronecker delta. Substituting into equation (4.39) yields

$$y(t) = \sum_{n=q}^{Q} c^{(q)} \left(\sum_{\tau_1=0}^{T-1} \cdots \sum_{\tau_q=0}^{T-1} h(\tau_1)u(t-\tau_1)\ldots u(t-\tau_q)\delta_{\tau_1,\tau_2}\ldots\delta_{\tau_1,\tau_q} \right) \tag{4.40}$$

The order q Volterra kernel of the Hammerstein cascade is obtained by equating order q terms of equation (4.40) to those of the finite Volterra series (4.9),

$$\mathbf{h}^{(\mathbf{q})}(\tau_1,\ldots,\tau_q) = c^{(q)}h(\tau_1)\delta_{\tau_1,\tau_2}\delta_{\tau_1,\tau_3}\ldots\delta_{\tau_1,\tau_q} \tag{4.41}$$

As with the Wiener model, there is one redundant degree of freedom in the Hammerstein model. The overall linear gain can be distributed arbitrarily between the linear and nonlinear elements without changing the overall input–output relationship. Consequently, it is once again useful to normalize the elements of a Hammerstein system in some manner and to include the normalization in the model description.

Since the scaling of the nonlinearity is arbitrary, the ordinates of the nonlinearity and of the intermediate signal will both have dimensions of A.U. (arbitrary units). If the peripheral auditory model could be described by a Hammerstein cascade, its linear element would have dimensions of pps/A.U.s.

4.3.2.2 Testing Volterra Kernels for the Hammerstein Structure Equation (4.41) makes it evident that the Volterra kernels of a Hammerstein system are only nonzero along their diagonals ($\tau_1 = \tau_2 = \cdots = \tau_q$). Furthermore, the diagonals of each of the kernels, $\mathbf{h}^{(\mathbf{q})}(\tau_1,\ldots,\tau_1)$, will be proportional to the impulse response of the linear subsystem. These properties may be used to test an unknown system for the Hammerstein structure.

To demonstrate this, consider first the Hammerstein system shown in Figure 4.10. The dynamic linear and static nonlinear elements are the same as for the Wiener cascade

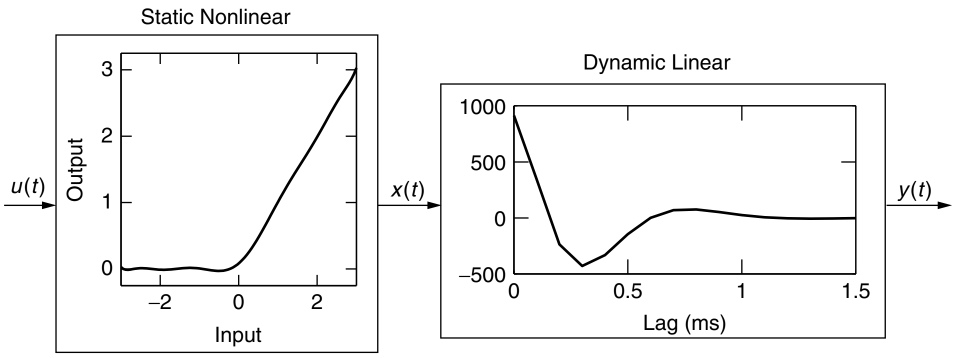

Figure 4.10 Elements of a Hammerstein system with the same first-order Volterra kernel and static nonlinearity as the peripheral auditory processing model in Figure 4.1.

of Figure 4.6, but their order is reversed. Figure 4.11 shows the first- and second-order Volterra kernels computed for this model. Note that the second-order kernel is zero everywhere except along the diagonal elements, where $\tau_1 = \tau_2$. Moreover, Figure 4.11B demonstrates that the diagonal elements are proportional to the first-order kernel. Furthermore, both the diagonal and the first-order kernel are proportional to the IRF of the linear element in Figure 4.10.

4.3.2.3 Example: Peripheral Auditory Model—Test for Hammerstein Structure For comparison purposes examine the second-order Volterra kernel of the peripheral auditory processing model shown in Figure 4.2. It is clear that many of the off-diagonal values are much different than zero. This clearly demonstrates that this system cannot be described as a Hammerstein cascade, which is again reassuring since the model has a LNL structure.

4.3.3 Sandwich or Wiener–Hammerstein Models

Another common block structure is the LNL, or sandwich model (Korenberg and Hunter, 1986; Rugh, 1981), comprising two linear systems separated by a static nonlinearity, as shown in Figure 4.12. This model is also called the Wiener–Hammerstein model or the Korenberg–Billings model. The output of this system may be determined by convolving the output of a Wiener system, (4.34), with a linear impulse response. Thus

$$y(t) = \sum_{\sigma=0}^{T-1} g(\sigma) \sum_{q=0}^{Q} c^{(q)} \left(\sum_{\tau=0}^{T-1} h(\tau)u(t-\sigma-\tau) \right)^q \tag{4.42}$$

4.3.3.1 Relationship to the Volterra Series The Volterra kernels of the LNL model may be obtained by rearranging (4.42) to have the form of (4.9). To do so, change the order of the summations to sum over q first, and then make the substitution $\nu = \sigma + \tau$:

$$y(t) = \sum_{q=0}^{Q} c^{(q)} \sum_{\sigma=0}^{T-1} g(\sigma) \left(\sum_{\nu=\sigma}^{\sigma+T-1} h(\nu-\sigma)u(t-\nu) \right)^q$$

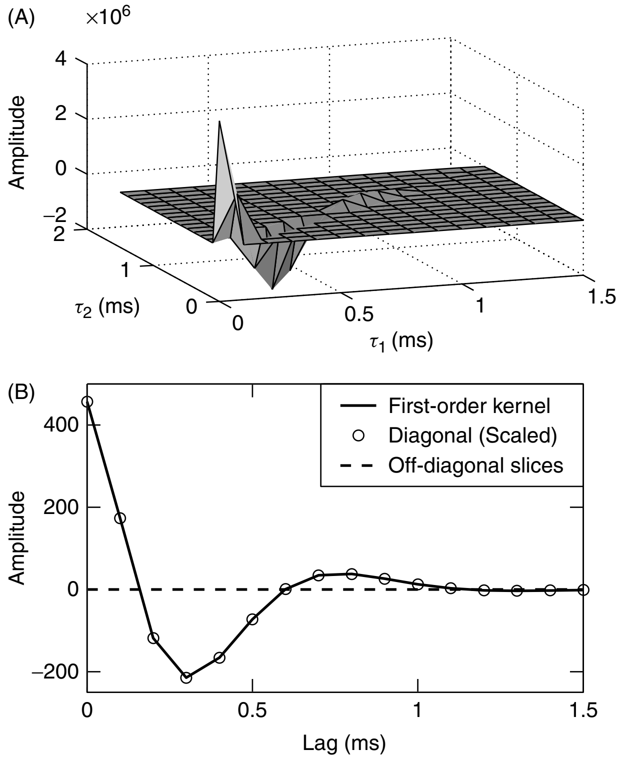

Figure 4.11 (A) The second-order Volterra kernel of the Hammerstein system shown in Figure 4.10. The kernel elements are zero everywhere except along the diagonal. (B) The first-order Volterra kernel superimposed on a scaled copy of the diagonal of the second-order kernel. The dashed line is an off-diagonal slice, which is zero.

$$\begin{tikzcd}
{} \arrow[r, "u(t)"] & \boxed{h(\tau)} \arrow[r, "x(t)"] & \boxed{m(\cdot)} \arrow[r, "w(t)"] & \boxed{g(\tau)} \arrow[r, "y(t)"] & {}
\end{tikzcd}$$

Figure 4.12 Block diagram of a LNL cascade model, also known as a sandwich model, a Wiener–Hammerstein model, and a Korenberg–Billings model.

Finally, rewrite the exponent, q, as a product of q terms:

$$y(t) = \sum_{q=0}^{Q} c^{(q)} \sum_{\sigma=0}^{T-1} g(\sigma) \left(\sum_{\nu_1=\sigma}^{\sigma+T-1} \cdots \sum_{\nu_q=\sigma}^{\sigma+T-1} h(\nu_1 - \sigma) \cdot \ldots \right.$$
$$\left. \cdot\, h(\nu_q - \sigma) u(t-\nu_1) \ldots u(t-\nu_q) \right) \tag{4.43}$$

Equating the order q terms in equations (4.43) and (4.9) yields the Volterra kernels of the LNL system as

$$\mathbf{h}^{(\mathbf{q})}(\tau_1, \dots, \tau_q) = c^{(q)} \sum_{\sigma=0}^{T-1} g(\sigma)h(\tau_1 - \sigma)h(\tau_2 - \sigma)\dots h(\tau_q - \sigma) \tag{4.44}$$

4.3.3.2 Relationship to the Wiener Series The Wiener kernels of an LNL cascade can be computed from its Volterra kernels, given by equation (4.44), using equation (4.30). Thus,

$$\begin{aligned}\mathbf{k}^{(\mathbf{q})}(\tau_1, \dots, \tau_q) = \sum_{j=0}^{\lfloor\frac{Q-q}{2}\rfloor} \frac{(q+2j)!\sigma_u^{2j}}{q!j!2^j} \sum_{\sigma_1=0}^{T-1} \dots \sum_{\sigma_j=0}^{T-1} c^{(q+2j)} \\ \times \sum_{i=0}^{T-1} g(i)h(\tau_1 - i)\dots h(\tau_q - i) \\ \cdot h(\sigma_1 - i)h(\sigma_1 - i)\dots h(\sigma_j - i)h(\sigma_j - i)\end{aligned}$$

Exchange the order of the summations to give

$$\begin{aligned}\mathbf{k}^{(\mathbf{q})} = \sum_{j=0}^{\lfloor\frac{Q-q}{2}\rfloor} \frac{(q+2j)!\sigma_u^{2j}}{q!j!2^j} c^{(q+2j)} \sum_{i=0}^{T-1} g(i)h(\tau_1 - i)\dots h(\tau_q - i) \\ \cdot \sum_{\sigma_1=0}^{T-1} \dots \sum_{\sigma_j=0}^{T-1} h(\sigma_1 - i)h(\sigma_1 - i)\dots h(\sigma_j - i)h(\sigma_j - i)\end{aligned}$$

Note that T, the memory length of the LNL cascade, is equal to the sum of the memory lengths of the two linear filters. Thus, if i is chosen such that $g(i) \neq 0$, then summing $h(\sigma - i)$ for $\sigma = 0 \dots T - 1$ will include all nonzero elements of $h(\tau)$. Therefore, the sums over $\sigma_1 \dots \sigma_j$ can be performed independently of the sum over i, as follows:

$$\begin{aligned}\mathbf{k}^{(\mathbf{q})}(\tau_1, \dots, \tau_q) = \sum_{j=0}^{\lfloor\frac{Q-q}{2}\rfloor} \frac{(q+2j)!\sigma_u^{2j}}{q!j!2^j} c^{(q+2j)} \left(\sum_{\sigma=0}^{T-1} h^2(\sigma)\right)^j \\ \times \sum_{i=0}^{T-1} g(i)h(\tau_1 - i)\dots h(\tau_q - i)\end{aligned} \tag{4.45}$$

Everything before the final summation is independent of the lags $\tau_1 \dots \tau_n$, and thus it is constant for each kernel. Therefore, by comparing equations (4.45) and (4.44), it can be seen that the Wiener kernels of an LNL cascade are proportional to its Volterra kernels. Furthermore, since the Wiener and Hammerstein cascades are special cases of the LNL model, the same observation will apply to them as well.

4.3.3.3 Testing Kernels for the LNL Structure As with the Wiener and Hammerstein structures, it is possible to derive a test that provides a necessary but not sufficient condition for the LNL structure. Let $\mathbf{h^{(1)}}(\tau)$ and $\mathbf{h^{(2)}}(\tau_1, \tau_2)$ be the first- and second-order Volterra kernels of an LNL cascade with linear elements $h(\tau)$ and $g(\tau)$, as shown in Figure 4.12. Let the two linear elements be causal with memory lengths, T_1 and T_2, respectively, so that

$$\begin{aligned} h(\tau) &= 0 \qquad \text{for } \tau < 0 \text{ or } \tau \geq T_1 \\ g(\tau) &= 0 \qquad \text{for } \tau < 0 \text{ or } \tau \geq T_2 \end{aligned}$$

Thus, the memory length of the cascade is $T = T_1 + T_2$.

The *second-order marginal kernel* is the sum of all slices of the second-order kernel taken parallel to one axis and is given by

$$\begin{aligned} k(\tau) &= \sum_{i=0}^{T-1} \mathbf{h^{(2)}}(\tau, i) \\ &= \sum_{i=0}^{T-1} \sum_{\sigma=0}^{T-1} c^{(2)} g(\sigma) h(\tau - \sigma) h(i - \sigma) \end{aligned}$$

Given the memory lengths of the individual elements, the extent of the summations may be reduced to include only nonzero terms:

$$k(\tau) = \sum_{\sigma=0}^{T_2-1} \sum_{i=\sigma}^{T-1} c^{(2)} g(\sigma) h(\tau - \sigma) h(i - \sigma)$$

Make the change of variables, $\gamma = i - \sigma$, to get

$$\begin{aligned} k(\tau) &= \sum_{\sigma=0}^{T_2-1} \sum_{\gamma=0}^{T_1-1} c^{(2)} g(\sigma) h(\tau - \sigma) h(\gamma) \\ &= \left(\frac{c^{(2)}}{c^{(1)}} \sum_{\gamma=0}^{T_1-1} h(\gamma) \right) \mathbf{h^{(1)}}(\tau) \end{aligned}$$

Thus, for a LNL system, the first-order Volterra kernel will be proportional to the sum of the one-dimensional slices of the second-order Volterra kernel taken parallel to one axis, the marginal second-order kernel. That is,

$$\mathbf{h^{(1)}}(\tau) \quad \alpha \quad k(\tau) = \sum_{i=0}^{T-1} \mathbf{h^{(2)}}(\tau, i) \tag{4.46}$$

As before, this is not a sufficient condition; satisfying (4.46) does not guarantee that the underlying system is an LNL cascade. However, if the kernels do not satisfy equation (4.46), then the system cannot be a LNL cascade.

Note also that the Wiener and Volterra kernels of an LNL cascade are proportional to each other, so this test may be applied to a system's Wiener kernels as well. Similarly, the tests for the Wiener and Hammerstein cascade structures can be applied to either the Volterra or Wiener kernels.

4.3.3.4 Example: Peripheral Auditory Model—Test for LNL Structure The simplified model of the peripheral auditory system, shown in Figure 4.1 and used as a running example throughout this chapter, consists of two bandpass filters separated by a half-wave rectifier (Pfeiffer, 1970; Swerup, 1978) and thus is a LNL cascade. Figure 4.13 illustrates the application of the LNL structure test, defined by equation (4.46), to the Volterra kernels of this model shown in Figure 4.2. Figures 4.13A and 4.13B show the first- and second-order Volterra kernels. Figure 4.13C shows the first four slices ($\tau_1 = 0$, 1, 2, and 3 ms) of the second-order kernel; note that none of these are proportional to the first-order kernel. Figure 4.13D presents the results of the LNL structural test. The solid line is the normalized marginal kernel, the sum of all nonzero slices of the second-order kernel. The circles represent the normalized first-order kernel that is seen to be identical to the normalized marginal kernel. Thus, these Volterra kernels are consistent with an LNL structure, which is reassuring since the peripheral auditory model is indeed an LNL cascade.

4.3.4 NLN Cascades

Figure 4.14 shows the final cascade model to be considered in detail; the NLN model consisting of a linear system, $h(\tau)$, sandwiched between two static nonlinearities, $m_1(\cdot)$

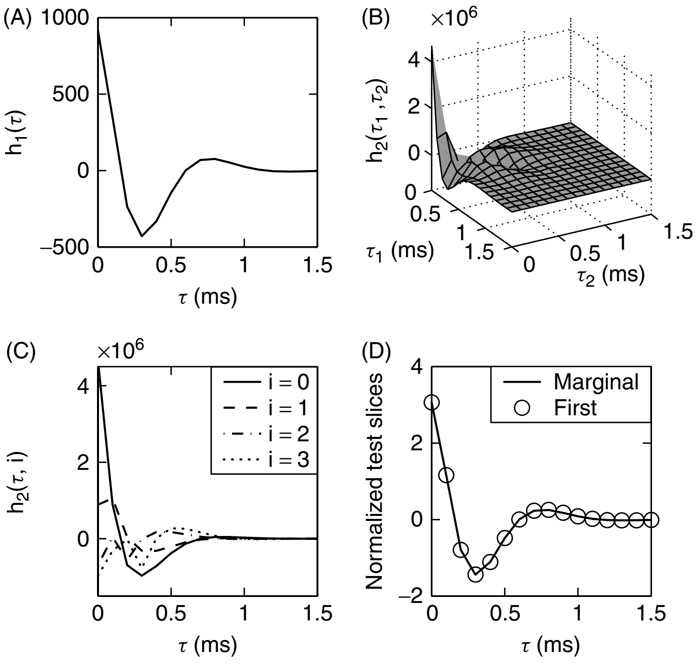

Figure 4.13 LNL structure test of the peripheral auditory processing model. (A) First- and (B) second-order Volterra kernels. (C) The first four slices of the second-order kernel. (D) The marginal second-order kernel superimposed on the first-order kernel. Both slices have been normalized. The marginal kernel is proportional to the first-order kernel, indicating that the system could be an LNL cascade.

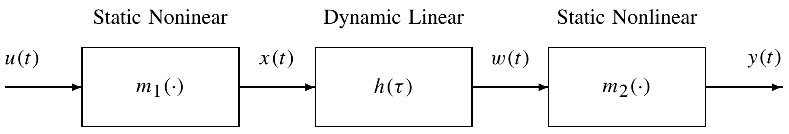

Figure 4.14 Block diagram of an NLN cascade model, comprising two static nonlinearities separated by a dynamic linear system.

and $m_2(\cdot)$. The output can be written as

$$y(t) = \sum_{q_2=0}^{Q_2} c_2^{(q_2)} \left(\sum_{q_1=0}^{Q_1} c_1^{(q_1)} \sum_{\tau=0}^{T-1} h(\tau) u^{q_1}(t-\tau) \right)^{q_2} \tag{4.47}$$

4.3.4.1 Relationship to the Volterra Series Deriving the Volterra kernels for this structure is complex because the effects of the two nonlinearities interact. To compute the Volterra kernels, the output (4.47) must be separated into terms corresponding to each nonlinearity order. The kernels that generate these terms can then be computed. To do this, first define the intermediate signal $x_q(t)$ as

$$x_q(t) = c_1^{(q)} \sum_{\tau=0}^{T-1} h(\tau) u^q(t-\tau) \tag{4.48}$$

the result of filtering the output of the order q term of the nonlinearity, $m_1(\cdot)$, with the linear IRF, $h(\tau)$. The output of the NLN cascade can be written in terms of the $x_q(t)$ as

$$y(t) = \sum_{q=0}^{Q_2} c_2^{(q)} \left(x_0(t) + x_1(t) + x_2(t) + \cdots + x_{Q_1}(t) \right)^q \tag{4.49}$$

Expanding the qth exponent in (4.49) gives

$$\begin{aligned} \left(x_0(t) + \cdots + x_{Q_1}(t) \right)^q = \Bigg(& x_0^q(t) + \binom{q}{1} x_1(t) x_0^{q-1}(t) \\ & + \binom{q}{1} x_2(t) x_0^{q-1}(t) + \cdots + x_{Q_1}^q(t) \Bigg) \end{aligned} \tag{4.50}$$

where $\binom{q}{1} = q$, the binomial coefficient, is the number of ways of choosing one element from a set of q elements. Thus, each term of the power series in equation (4.49) contains a term of the form $x_0^q(t)$ that contributes to the zero-order output. Thus, the output of the zero-order kernel of the NLN cascade is

$$y^{(0)}(t) = \sum_{q=0}^{Q_2} c_2^{(q)} x_0^q \tag{4.51}$$

Note that the constant x_0 is given by

$$x_0 = c_1^{(0)} \sum_{\tau=0}^{T-1} h(\tau) \tag{4.52}$$

To obtain the first-order output term, note that the only first-order term in equation (4.50) is given by the product of $x_1(t)$ with $q - 1$ copies of $x_0(t)$. Thus, to obtain the first-order contribution, it is only necessary to expand the binomial

$$c_2^{(q)}(x_0(t) + x_1(t))^q$$

since the higher-order terms, $x_2(t) \dots x_{Q_1}(t)$, will contribute only to higher-order outputs and can be neglected. Thus, the only first-order term [in each term in equation (4.49)] is

$$\binom{q}{1} c_2^{(q)} x_0^{q-1}(t) x_1(t)$$

Note that every term in the power series (4.49), for $q > 0$, includes a first-order output contribution. Summing these first-order terms gives the output of the first-order Volterra kernel,

$$y^{(1)}(t) = x_1(t) \sum_{q=1}^{Q_2} q c_2^{(q)} x_0^{q-1}(t) \tag{4.53}$$

so the first-order kernel is

$$\mathbf{h}^{(1)}(\tau) = c_1^{(1)} \left(\sum_{q=1}^{Q_2} q c_2^{(q)} x_0^{q-1} \right) h(\tau) \tag{4.54}$$

Thus, the first-order Volterra kernel of the NLN cascade is proportional to the IRF of its linear element.

Things are more complicated for the second-order kernel. As with the derivation of the first-order output, higher-order terms in equation (4.50) may be neglected since they do not contribute to the second-order output. Thus, only the trinomial

$$c_2^{(q)}(x_0(t) + x_1(t) + x_2(t))^q$$

need be considered. Expand that and note that the result will contain two types of second-order terms:

$$\binom{q}{1} x_2(t) x_0^{q-1}(t) \qquad \text{for } q \geq 1$$

$$\binom{q}{2} x_1^2(t) x_0^{q-2}(t) \qquad \text{for } q \geq 2$$

Thus, the second-order kernel output will be a weighted sum of $x_1^2(t)$ and $x_2(t)$:

$$y^{(2)}(t) = x_2(t) \sum_{q=1}^{Q_2} q c_2^{(q)} x_0^{q-1} + x_1^2(t) \sum_{q=2}^{Q_2} \binom{q}{2} c_2^{(q)} x_0^{q-2} \tag{4.55}$$

Since $x_1^2(t)$ is the output of a Wiener cascade,

$$x_1^2(t) = \left(c_1^{(1)}\right)^2 \sum_{\tau_1=0}^{T-1} \sum_{\tau_2=0}^{T-1} h(\tau_1)h(\tau_2)u(t-\tau_1)u(t-\tau_2)$$

and $x_2(t)$ is the output of a Hammerstein cascade,

$$x_2(t) = c_1^{(2)} \sum_{\tau_1=0}^{T-1} \sum_{\tau_2=0}^{T-1} h(\tau_1)\delta(\tau_1-\tau_2)u(t-\tau_1)u(t-\tau_2)$$

the second-order kernel will have a "Wiener plus Hammerstein" structure,

$$\begin{aligned}\mathbf{h^{(2)}}(\tau_1,\tau_2) &= \left(\left(c_1^{(1)}\right)^2 \sum_{q=1}^{Q_2} q c_2^{(q)} x_0^{q-1}\right) h(\tau_1)h(\tau_2) \\ &\quad + \left(c_1^{(2)} \sum_{q=2}^{Q_2} \binom{q}{2} c_2^{(q)} x_0^{q-2}\right) h(\tau_1)\delta(\tau_1-\tau_2) \qquad (4.56)\end{aligned}$$

The derivation for the third-order kernel is more complicated still since it comprises the weighted sum of three kernel types. As before, consider the expansion of

$$c_2^{(q)}(x_0(t)+x_1(t)+x_2(t)+x_3(t))^q$$

and collect all third-order terms

$$\binom{q}{1} x_3(t)x_0^{q-1}(t) \qquad \text{for } q \geq 1$$

$$\binom{q}{2} x_1(t)x_2(t)x_0^{q-2}(t) \qquad \text{for } q \geq 2$$

$$\binom{q}{3} x_1^3(t)x_0^{q-3}(t) \qquad \text{for } q \geq 3$$

The first and last terms will make "Hammerstein-like" and "Wiener-like" contributions to the kernel, as for the second-order case. The second term generates the output

$$\sum_{\tau_1=0}^{T-1} \sum_{\tau_2=0}^{T-1} \sum_{\tau_3=0}^{T-1} h(\tau_1)h(\tau_2)\delta(\tau_2-\tau_3)u(t-\tau_1)u(t-\tau_2)u(t-\tau_3) \qquad (4.57)$$

and a contribution to the kernel that is proportional to

$$h(\tau_1)h(\tau_2)\delta(\tau_2-\tau_3) + h(\tau_2)h(\tau_1)\delta(\tau_1-\tau_3) + h(\tau_3)h(\tau_2)\delta(\tau_1-\tau_2)$$

which is obtained by making the kernel in (4.57) symmetric.

Similar, but ever more complex, manipulations are required to derive the higher-order kernels; there is no simple formula for determining them. Consequently, since all the

Volterra kernels must be known to determine the Wiener kernels [see equation (4.30)], there is also no simple relation for the Wiener kernels of an NLN model.

4.3.5 Multiple-Input Multiple-Output Block Structured Models

Block structured models can be generalized to represent multiple-input multiple-output (MIMO) systems by replacing the single-input single-output linear filters and static nonlinearities with equivalent MIMO elements. One approach would be to use a bank of IRFs, one for each combination of input and output signals. However, it will usually be more efficient to use the state-space model structure, described in Section 3.4, since dynamics common to multiple pathways need only be represented once.

4.4 PARALLEL CASCADES

The simple block structured models described in the previous section can represent some high-order nonlinear systems very efficiently—when they are appropriate. However, many systems cannot be represented accurately by a simple cascade model. For example, Figure 4.15 illustrates a block-structured model of human ankle stiffness dynamics (Kearney et al., 1997) that incorporates two parallel pathways: one linear and one LNL. The Wiener and/or Volterra kernels of this model will not pass the structural tests for the Wiener, Hammerstein, or LNL cascades presented above.

In such cases, a parallel cascade model structure may combine the compactness of block structured models with the generality of functional expansions. The concept underlying the parallel cascade model is to represent the overall system response as the sum of the outputs from a parallel set of simple block structured models.

Figure 4.16 illustrates a parallel cascade model that uses LNL systems for the individual pathways. The overall output is equal to the sum of the outputs of all pathways. That is,

$$y(t) = \sum_{k=1}^{P} \left\{ \sum_{j=0}^{T-1} g_k(j) \sum_{q=0}^{Q} c_k^{(q)} \left(\sum_{\tau=0}^{T-1} h_k(\tau) u(t-j-\tau) \right)^q \right\} \tag{4.58}$$

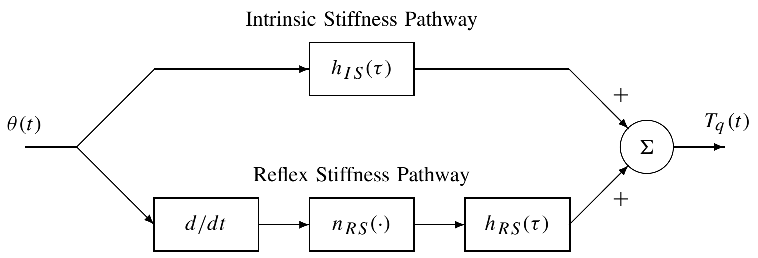

Figure 4.15 Block structured model of the dynamic stiffness of the human ankle, defined as the dynamic relationship between the ankle angle, θ, and the ankle torque, T_q. The model includes both intrinsic (upper pathway) and reflex (lower pathway) components.

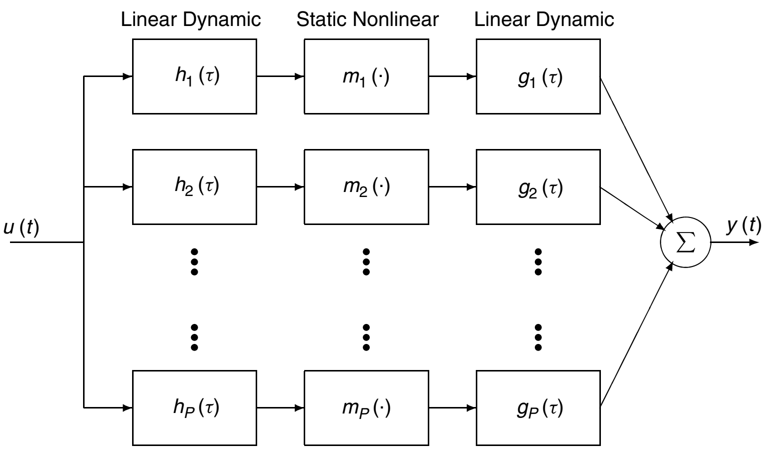

Figure 4.16 A parallel sum of LNL cascade models.

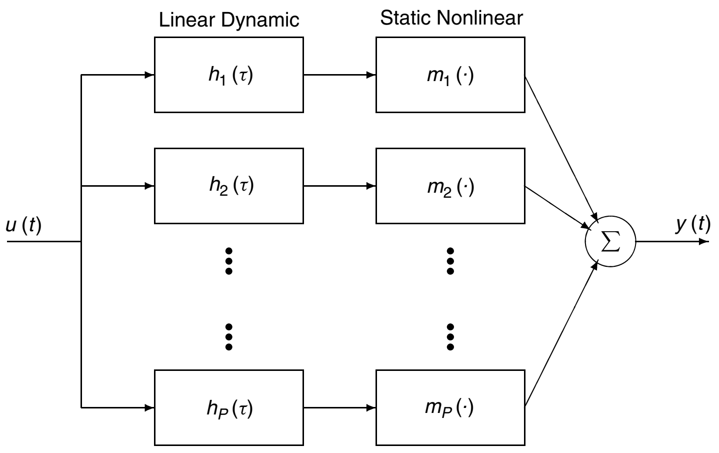

Figure 4.17 Parallel cascade made up of Wiener systems.

Using this and equation (4.44), it is evident that the Volterra kernels of a parallel LNL cascade model will be equal to the sum of the kernels of the individual pathways.

$$\mathbf{h}^{(\mathbf{q})}(\tau_1, \dots, \tau_q) = \sum_{k=1}^{P} \left\{ c_k^{(q)} \sum_{j=0}^{T-1} g_k(j) h_k(\tau_1 - j) \dots h_k(\tau_q - j) \right\} \tag{4.59}$$

Figure 4.17 shows another, simpler, parallel cascade that uses Wiener systems for the parallel pathways. The overall output is the sum of the outputs of all pathways, so that, as before, the Volterra kernels may be obtained by summing the kernels from each pathway. By collecting terms of the same order, it can be seen that the order q kernel is

$$\mathbf{h}^{(\mathbf{q})}(\tau_1, \dots, \tau_q) = \sum_{k=1}^{P} c_k^{(q)} h_k(\tau_1) h_k(\tau_2) \dots h_k(\tau_q) \tag{4.60}$$

4.4.1 Approximation Issues(†)

Palm (1979) provided the theoretical basis for the parallel cascade model by showing that the output of any continuous, discrete-time, finite-dimensional system can be approximated by a parallel LNL cascade to within an arbitrarily small error. Korenberg (1991) extended these results by showing that the output of any discrete-time, continuous, finite-dimensional system could be approximated, in the mean-square sense, to within an arbitrarily small error by a parallel Wiener cascade model. He showed further that the Volterra kernels of a discrete-time, continuous, finite-dimensional system could be represented exactly by a finite sum of Wiener cascades (Korenberg, 1991). Thus parallel cascade models can represent not only the input–output behavior of nonlinear systems but also their Volterra kernels. Note that the Wiener system is simply a special case of the LNL cascade, so this result also proved that LNL cascades can represent any nonlinear system, thereby extending Palm's (Palm, 1979) earlier result.

4.4.1.1 Example: Peripheral Auditory Model — Parallel Cascade Figure 4.18 shows the first three paths of a parallel Wiener cascade representation of the peripheral auditory model, which is actually an LNL cascade. The parallel cascade model is not

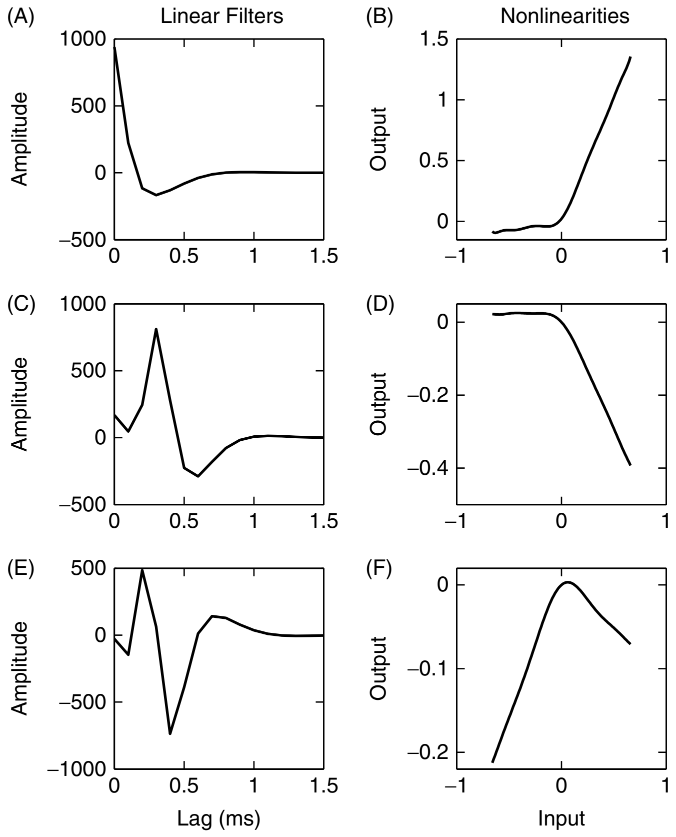

Figure 4.18 First three paths of a parallel Wiener cascade representation of the peripheral auditory model.

unique, so there can be no unique transformation from the Volterra kernels to a parallel cascade model. The cascade shown in Figure 4.18 was generated using the eigenvector algorithm, to be described in Section 8.2.4.

Figures 4.19A–4.19F show the first- and second-order Volterra kernels determined using the first one, two, and three pathways. Figures 4.19G and 4.19H show the "true" kernels computed directly from the model. Note that none of the IRFs in the cascade resemble the first-order kernel. Nevertheless, the estimated first-order kernel, shown in

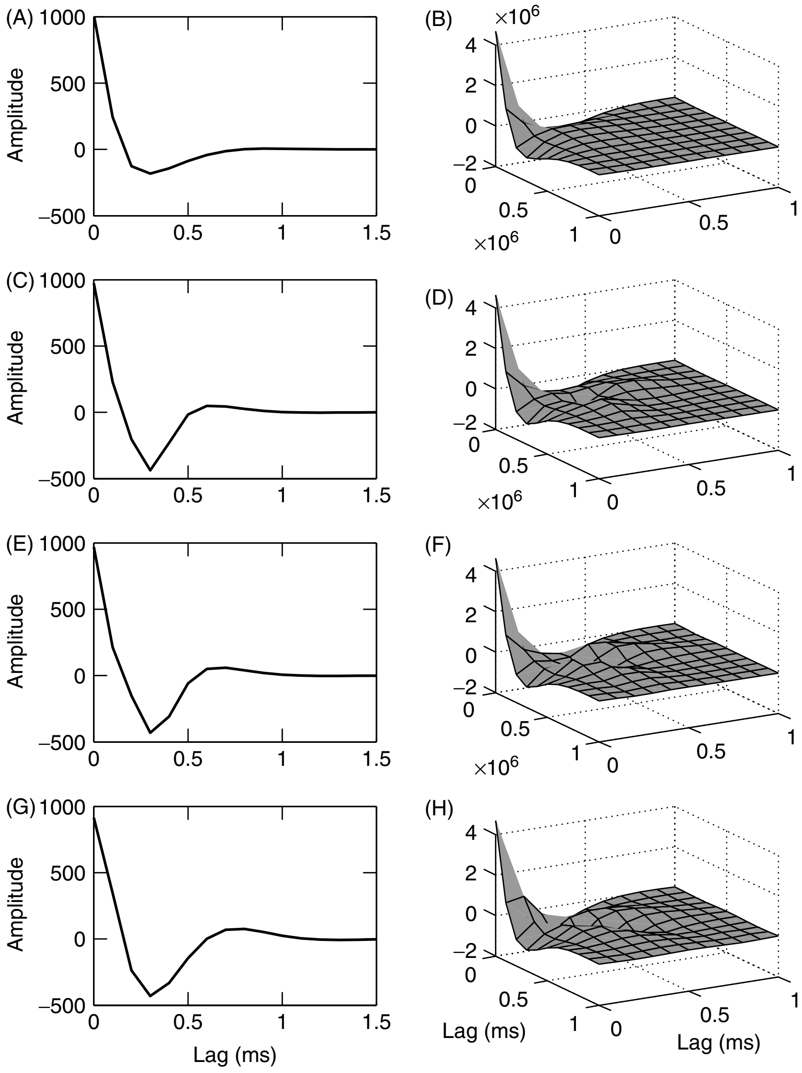

Figure 4.19 Volterra kernels of the parallel cascade model of the peripheral auditory model in Figure 4.18. (A) First- and (B) second-order kernels computed from the first pathway. (C) First- and (D) second-order Volterra kernels computed from the first two pathways. (E) First- and (F) second-order Volterra kernels computed from the first three pathways. (G) First- and (H) second-order Volterra kernels computed from the original model.

the left column, rapidly approaches the shape and size of the true first-order kernel. Similarly, the second-order kernel approaches that of the original model.

4.5 THE WIENER–BOSE MODEL

Figure 4.20 shows the structure of the Wiener–Bose model (Marmarelis, 1989; Paulin, 1993), also known as the generalized Wiener model (Marmarelis, 1987b, 1989) or the Volterra series approximator (Boyd and Chua, 1985). The Wiener–Bose model consists of a bank of linear filters, whose outputs are transformed by a multiple-input static nonlinearity. Sections 4.5.3–4.5.5 will demonstrate that the Wiener and Volterra functional expansions, as well as the parallel Wiener cascade, are special cases of the Wiener–Bose model. Consequently, the Wiener–Bose model provides a common conceptual framework for the analysis of the model structures discussed in this chapter and for the identification methods to follow in Chapters 6–8.

To compute the output of the Wiener–Bose model, let $x_k(t)$ represent the output of the kth filter in the bank,

$$x_k(t) = \sum_{\tau=0}^{T-1} h_k(\tau)u(t-\tau) \tag{4.61}$$

Represent the static nonlinearity, $m(\cdot, \dots, \cdot)$, as a multiple-input polynomial of degree Q,

$$m(x_1, \dots, x_P) = \sum_{q=0}^{Q} \sum_{k_1=1}^{P} \sum_{k_2=k_1}^{P} \cdots \sum_{k_q=k_{q-1}}^{P} c^{(q)}_{k_1,k_2,\dots,k_q} \cdot \mathcal{M}^{(q)}(x_{k_1}(t), x_{k_2}(t), \dots, x_{k_q}(t)) \tag{4.62}$$

where $\mathcal{M}^{(q)}(x_1, \dots, x_q)$, the qth-order multiple-input polynomial term with arguments x_1 through x_q, defined in Section 2.5.5, is simply the product of its arguments. Then, the output of the Wiener–Bose model is given by

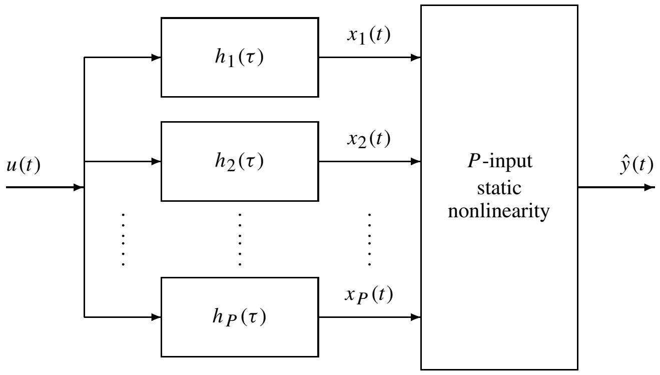

Figure 4.20 Block diagram of a Wiener–Bose model. This structure has also been called the generalized Wiener model and the Volterra series approximator.

$$y(t) = \sum_{q=0}^{Q} \sum_{k_1=1}^{P} \sum_{k_2=k_1}^{P} \cdots \sum_{k_q=k_{q-1}}^{P} c^{(q)}_{k_1,k_2,\ldots,k_q} x_{k_1}(t) \cdot x_{k_2}(t) \cdot \ldots \cdot x_{k_q}(t) \tag{4.63}$$

4.5.1 Similarity Transformations and Uniqueness

Recall from Section 4.3.1 that the Wiener cascade contains an extra degree of freedom, since the overall gain may be distributed arbitrarily between the linear dynamic element and the static nonlinearity. The Wiener–Bose model contains similar redundancies. The FIR filter bank can be transformed by multiplying it by any nonsingular matrix without changing the input–output behavior, provided that the output nonlinearity is corrected appropriately.

To facilitate subsequent manipulations, define the following terms:

1. $\mathbf{u(t)} = [u(t)\ u(t-1)\ \ldots\ u(t-T+1)]^T$, a vector containing lagged samples of the input, $u(t)$.
2. $\mathbf{G}$, a $T \times P$ matrix* whose columns contain the IRFs in the filter-bank,

$$\mathbf{G} = \begin{bmatrix} h_1(0) & h_2(0) & \ldots & h_P(0) \\ h_1(1) & h_2(1) & \ldots & h_P(1) \\ \vdots & \vdots & \vdots & \vdots \\ h_1(T-1) & h_2(T-1) & \ldots & h_P(T-1) \end{bmatrix} \tag{4.64}$$

3. $\mathbf{x(t)} = [x_1(t)\ x_2(t)\ \ldots\ x_P(t)]^T$, a vector containing the filter outputs at time t.

Multiplying $\mathbf{u(t)}$ by a transposed column of $\mathbf{G}$ is equivalent to a discrete convolution (3.9), so $\mathbf{x(t)}$ may be written as a matrix–vector product:

$$\mathbf{x(t)} = \mathbf{G}^T\mathbf{u(t)} \tag{4.65}$$

Now, if $\mathbf{G}$ is multiplied by any nonsingular $P \times P$ matrix, $\mathbf{T}$, the IRF output vector becomes

$$\begin{aligned}(\mathbf{GT})^T\mathbf{u(t)} &= \mathbf{T}^T\mathbf{G}^T\mathbf{u(t)} \\ &= \mathbf{T}^T\mathbf{x(t)}\end{aligned}$$

The effect of this transformation can be removed by scaling the polynomial coefficients appropriately. To see this, consider the first-order terms in equation (4.63), whose output is obtained by setting $q = 1$ in the first summation. These first-order terms are

$$y^{(1)}(t) = \mathbf{x(t)}^T\mathbf{c^{(1)}} \tag{4.66}$$

where $\mathbf{c^{(1)}} = [c_1^{(1)}\ c_2^{(1)}\ \ldots\ c_P^{(1)}]^T$ is a column vector of first-order polynomial coefficients.

The equivalent relation for the transformed system is

$$y_T^{(1)}(t) = (\mathbf{T}^T\mathbf{x(t)})^T\mathbf{c^{(1)}}$$

where the subscript T is used to denote the transformed system.

*The logical symbol for this matrix would be **H**, but that is already used to denote the Hessian.

Therefore, replacing $\mathbf{c^{(1)}}$ with $\mathbf{T}^{-1}\mathbf{c^{(1)}}$ will correct the effects of the filter-bank transformation on the first-order terms,

$$\begin{aligned} y^{(1)}(t) &= (\mathbf{T}^T\mathbf{x(t)})^T\mathbf{T}^{-1}\mathbf{c^{(1)}} \\ &= \mathbf{x(t)}^T\mathbf{T}\mathbf{T}^{-1}\mathbf{c^{(1)}} \\ &= \mathbf{x(t)}^T\mathbf{c^{(1)}} \end{aligned}$$

Next, consider the output of the second-order terms. As with the first-order terms, express the output using vector and matrix multiplications,

$$y^{(2)}(t) = \mathbf{x}^T\mathbf{(t)}\mathbf{C^{(2)}}\mathbf{x(t)} \tag{4.67}$$

where $\mathbf{C^{(2)}}$ is a $P \times P$ matrix containing the coefficients of the second-order polynomial terms. To determine how the second-order polynomial coefficients enter into the matrix $\mathbf{C^{(2)}}$, expand the matrix-vector products in equation (4.67) as

$$y^{(2)}(t) = \sum_{i=1}^{P}\sum_{j=1}^{P} x_i(t)x_j(t)\mathbf{C^{(2)}(i, j)} \tag{4.68}$$

Compare this to the second-order terms in (4.63), which are given by

$$y^{(2)}(t) = \sum_{i=1}^{P}\sum_{j=i}^{P} c_{i,j}^{(2)} x_i(t)x_j(t) \tag{4.69}$$

Notice that the second summation in equation (4.69) runs from i to P, whereas both summations in equation (4.68) run over the full range from 1 to P. Thus, when $i \neq j$, the polynomial coefficient, $c_{i,j}^{(2)}$ in equation (4.69), must be split between matrix elements, $\mathbf{C^{(2)}}(i, j)$ and $\mathbf{C^{(2)}}(j, i)$, in equation (4.68). Thus,

$$\mathbf{C^{(2)}} = \begin{bmatrix} c_{1,1}^{(2)} & \frac{1}{2}c_{1,2}^{(2)} & \cdots & \frac{1}{2}c_{1,P}^{(2)} \\ \frac{1}{2}c_{1,2}^{(2)} & c_{2,2}^{(2)} & \cdots & \frac{1}{2}c_{2,P}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{2}c_{1,P}^{(2)} & \frac{1}{2}c_{2,P}^{(2)} & \cdots & c_{P,P}^{(2)} \end{bmatrix} \tag{4.70}$$

If the matrix of IRFs is transformed by $\mathbf{T}$, the second-order output terms will be

$$\begin{aligned} y_T^{(2)}(t) &= (\mathbf{T}^T\mathbf{x(t)})^T\mathbf{C^{(2)}}(\mathbf{T}^T\mathbf{x(t)})^T \\ &= \mathbf{x}^T\mathbf{(t)}\mathbf{T}\mathbf{C^{(2)}}\mathbf{T}^T\mathbf{x(t)} \end{aligned}$$

To compensate for this transformation, replace $\mathbf{C^{(2)}}$ with $\mathbf{T}^{-1}\mathbf{C^{(2)}}\mathbf{T}^{-T}$, giving

$$\begin{aligned} y^{(2)}(t) &= (\mathbf{T}^T\mathbf{x(t)})^T(\mathbf{T}^{-1}\mathbf{C^{(2)}}\mathbf{T}^{-T})(\mathbf{T}^T\mathbf{x(t)}) \\ &= \mathbf{x}^T\mathbf{(t)}\mathbf{C^{(2)}}\mathbf{x(t)} \end{aligned}$$

Similar corrections can be developed for the third- and higher-order kernel outputs.

Thus, the Wiener–Bose representation is not unique; there are an infinite number of combinations of basis vectors, h_k, and polynomial coefficients that represent the system equivalently. However, it may be possible to find a transformation, $\mathbf{T}$, that reduces the polynomial coefficients associated with one or more filters to zero. Such filters may be removed from the filter bank, reducing the complexity of the model without changing its accuracy.

4.5.2 Approximation Issues(†)

Boyd and Chua (1985) analyzed the approximation properties of the Wiener–Bose model, and proved the following theorem:

Theorem 4.3 The approximation, $\hat{\mathbf{N}}$, for any time-invariant fading memory operator given in Theorem 4.2 from Section 4.1.4 can be realized as follows:

$$\dot{\mathbf{x}}(\mathbf{t}) = \mathbf{A}\mathbf{x}(\mathbf{t}) + \mathbf{b}u(t) \tag{4.71}$$

$$y(t) = m(\mathbf{x}(\mathbf{t})) \tag{4.72}$$

where $\mathbf{x}(\mathbf{t})$ and $\mathbf{b}$ are P-dimensional vectors, for some finite P, $\mathbf{A}$ is an exponentially stable $P \times P$ matrix, and $m(\cdot) : \mathbb{R}^P \to \mathbb{R}$ is a polynomial.

The connection with the Wiener–Bose model becomes evident when it is noted that equation (4.71) is a state-space description of a single-input, $u(t)$, P-output linear state-space system. This is equivalent to a bank of P separate filters, $h_1(\tau), h_2(\tau), \ldots, h_P(\tau)$; ideally these should be chosen to have orthogonal outputs, $x_1(t), x_2(t), \ldots, x_P(t)$. Similarly, the multiple-input static nonlinearity, m, is often expanded using orthogonal polynomials such as the Grad–Hermite polynomials discussed in Section 2.5.5.

4.5.3 Volterra Kernels of the Wiener–Bose Model

The Volterra kernels of a Wiener–Bose model may be determined using a procedure similar to that employed for the other block structures. From equation (4.9), it is evident that the output of the order q Volterra kernel involves a q-dimensional convolution of the kernel with q copies of the input. Thus, only terms in the Wiener–Bose model (4.63) that involve the product of q signals will contribute to the order q kernel. Therefore, the polynomial coefficient degree determines the order of the Volterra kernel to which it contributes.

The combined output of all first-order polynomial terms in a Wiener–Bose model is given by:

$$\begin{aligned} y^{(1)}(t) &= \sum_{k=1}^{P} c_k^{(1)} x_k(t) \\ &= \sum_{k=1}^{P} c_k^{(1)} \sum_{\tau=0}^{T-1} h_k(\tau) u(t-\tau) \\ &= \sum_{\tau=0}^{T-1} \left(\sum_{k=1}^{P} c_k^{(1)} h_k(\tau) \right) u(t-\tau) \end{aligned} \tag{4.73}$$

Comparing this with equation (4.6), it is evident that the first-order Volterra kernel is given by

$$\mathbf{h^{(1)}}(\tau) = \sum_{k=1}^{P} c_k^{(1)} h_k(\tau) \tag{4.74}$$

Similarly, the symmetric second-order Volterra kernel of the Wiener–Bose model is

$$\mathbf{h^{(2)}}(\tau_1, \tau_2) = \sum_{k_1=1}^{P} \sum_{k_2=k_1}^{P} \frac{c_{k_1,k_2}^{(2)}}{2} \big(h_{k_1}(\tau_1) h_{k_2}(\tau_2) + h_{k_2}(\tau_1) h_{k_1}(\tau_2)\big) \tag{4.75}$$

Analogous expressions can be developed for higher-order Volterra kernels.

4.5.4 Wiener Kernels of the Wiener–Bose Model

This section will develop the relationship between a system's Wiener–Bose model and its Wiener kernels. The Wiener orthogonalization depends explicitly on the input, so it will be assumed to be Gaussian, white noise, with variance σ_u^2. The analysis proceeds in a manner analogous to that used for the Volterra series except that the static nonlinearity is expressed as a Grad–Hermite polynomial, rather than as a multiple-input power series. The output of this model, analogous to equation (4.63), is

$$y(t) = \sum_{q=0}^{Q} \sum_{k_1=1}^{P} \sum_{k_2=k_1}^{P} \cdots \sum_{k_q=k_{q-1}}^{P} \gamma_{k_1,k_2,\dots,k_q}^{(q)} \mathcal{H}^{(q)}(x_{k_1}(t), x_{k_2}(t), \dots, x_{k_q}(t)) \tag{4.76}$$

where $\gamma_{k_1,k_2,\dots,k_q}^{(q)}$ is the coefficient of the qth-order Hermite polynomial in the inputs $x_{k_1}(t), \dots, x_{k_q}(t)$.

For the expansion to be orthogonal, the filter outputs, $x_k(t)$, must be orthonormal:

$$E[x_i(t)x_j(t)] = \begin{cases} 1, & i = j \\ 0, & \text{otherwise} \end{cases}$$

Since the input is white noise, this leads to the following condition on the filter bank:

$$h_i^T h_j = \begin{cases} \dfrac{1}{\sigma_u^2}, & i = j \\ 0, & \text{otherwise} \end{cases}$$

This can be achieved by computing either the QR factorization or the SVD of **G**, the matrix of impulse responses.

Thus, for a white-noise input and for a filter bank whose IRFs are orthonormal, the terms in equation (4.76) will be orthogonal. There will then be a close relationship between the Hermite polynomial coefficients and the Wiener kernels.

Consider first the $q = 0$ term, which leads to the constant output $y^{(0)}(t) = \gamma^{(0)}$. This has the form of a zero-order Wiener functional and thus will be orthogonal to the outputs of all higher-order Wiener functionals. Similarly, all higher-order Grad–Hermite polynomial terms will have zero-mean outputs, and therefore they will be orthogonal to any constant and thus to any zero-order Wiener operator. Thus, $\gamma^{(0)}$ is the only polynomial term that contributes to the zero-order Wiener kernel, and

$$\mathbf{k}^{(\mathbf{0})} = \gamma^{(0)} \tag{4.77}$$

The same argument can be used to relate the first-order polynomial terms and Wiener kernel. For $q = 1$, we have

$$\begin{aligned} y^{(1)}(t) &= \sum_{k=1}^{P} \gamma_k^{(1)} \mathcal{H}^{(1)}(x_k(t)) \\ &= \sum_{k=1}^{P} \gamma_k^{(1)} x_k(t) \end{aligned}$$

But, $x_k(t) = h_k(\tau) * u(t)$. Thus,

$$y^{(1)}(t) = \sum_{\tau=0}^{T-1} \sum_{k=1}^{P} \gamma_k^{(1)} h_k(\tau) u(t-\tau)$$

which has the same form as the first-order Wiener operator.

Thus, $y^{(1)}(t)$ is orthogonal to the output of any higher-order (or zero-order) Wiener functional and is orthogonal to the outputs of all other Grad–Hermite polynomial terms. Hence, $y^{(1)}(t)$ must be the output of the first-order Wiener kernel, so that

$$\mathbf{k}^{(\mathbf{1})}(\tau) = \sum_{k=1}^{P} \gamma_k^{(1)} h_k(\tau) \tag{4.78}$$

Similarly, the output of the $q = 2$ term is given by

$$\begin{aligned} y^{(2)}(t) &= \sum_{k_1=1}^{P} \sum_{k_2=k_1}^{P} \gamma_{k_1,k_2}^{(2)} \mathcal{H}^{(2)}(x_{k_1}(t), x_{k_2}(t)) \\ &= \sum_{k_1=1}^{P} \sum_{k_2=k_1}^{P} \gamma_{k_1,k_2}^{(2)} x_{k_1}(t) x_{k_2}(t) - \sum_{k=1}^{P} \gamma_{k,k}^{(2)} \end{aligned} \tag{4.79}$$

Using an equation analogous to equation (4.75), construct a potential kernel,

$$\mathbf{k}^{(\mathbf{2})}(\tau_1, \tau_2) = \sum_{k_1=1}^{P} \sum_{k_2=k_1}^{P} \frac{\gamma_{k_1,k_2}^{(2)}}{2} \big(h_{k_1}(\tau_1) h_{k_2}(\tau_2) + h_{k_2}(\tau_1) h_{k_1}(\tau_2)\big) \tag{4.80}$$

and consider the second term in the second-order Wiener operator (4.21),

$$\sum_{\tau=0}^{T-1} \mathbf{k}^{(\mathbf{2})}(\tau, \tau) = \sum_{\tau=0}^{T-1} \sum_{k_1=1}^{P} \sum_{k_2=k_1}^{P} \gamma_{k_1,k_2}^{(2)} h_{k_1}(\tau) h_{k_2}(\tau)$$

$$= \sum_{k_1=1}^{P} \sum_{k_2=k_1}^{P} \gamma_{k_1,k_2}^{(2)} \sum_{\tau=0}^{T-1} h_{k_1}(\tau) h_{k_2}(\tau)$$

$$= \sum_{k=1}^{P} \gamma_{k,k}^{(2)}$$

Thus, equation (4.79) has the same form as a second-order Wiener operator, and equation (4.80) defines the second-order Wiener kernel of the Wiener–Bose model. In principle, this argument may be continued for all higher-order polynomial coefficients and Wiener kernels.

4.5.5 Relationship to the Parallel Cascade Model

It is apparent that a parallel cascade of Wiener systems is a special case of the Wiener–Bose model in which the nonlinearity contains no cross-terms. The parallel Wiener cascade model generates the outputs of these cross-terms by adding extra pathways. For such situations the full Wiener–Bose model will provide a more concise representation than an equivalent parallel Wiener cascade structure.

To examine the relationship between these two structures, consider the example shown in Figure 4.21, a Wiener–Bose model with two linear filters and a third-order static-nonlinearity. The linear filters, $h_1(\tau)$ and $h_2(\tau)$, have outputs $x_1(t)$ and $x_2(t)$, respectively. The system output is

$$y(t) = x_1^3(t) + x_1(t)x_2(t) - x_2(t) \tag{4.81}$$

Now, construct a parallel Wiener cascade model of this system. The first term in equation (4.81) can be generated by a Wiener path, $(g_1(\tau), m_1(\cdot))$, with $g_1(\tau) = h_1(\tau)$ as its linear IRF and $m_1(w_1) = w_1^3$ as its static nonlinearity. Similarly, the last term can be generated by a Wiener path, $(g_2(\tau), m_2(\cdot))$, with $g_2(\tau) = h_2(\tau)$ as its linear IRF and $m_2(w_2) = -w_2$ as its static nonlinearity. A third Wiener cascade will be needed to generate the cross-term, $x_1(t)x_2(t)$.

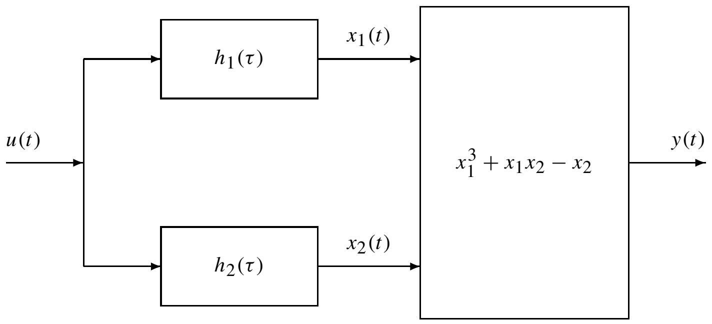

Figure 4.21 Wiener–Bose model used in the examples in Section 4.5.5. The elements are shown in the top row of Figure 4.23.

To do so let $g_3(\tau) = h_1(\tau) + h_2(\tau)$ and square its output to give

$$(x_1(t) + x_2(t))^2 = x_1^2(t) + 2x_1(t)x_2(t) + x_2^2(t)$$

This generates the needed cross-terms but also creates two extra terms, $x_1^2(t)$ and $x_2^2(t)$. These must be removed by adjusting the remaining nonlinearities, $m_1(\cdot)$ and $m_2(\cdot)$. Thus, the elements of the parallel Wiener cascade are

$$\begin{aligned} g_1(\tau) &= h_1(\tau) & m_1(w_1) &= -0.5w_1^2 + w_1^3 \\ g_2(\tau) &= h_2(\tau) & m_2(w_2) &= -w_2 - 0.5w_2^2 \\ g_3(\tau) &= h_1(\tau) + h_2(\tau) & m_3(w_3) &= \quad 0.5w_3^2 \end{aligned} \tag{4.82}$$

Summing the outputs of the three paths gives the output of the parallel cascade:

$$y(t) = m_1(w_1(t)) + m_2(w_2(t)) + m_3(w_3(t)) \tag{4.83}$$

These signals and elements are shown in Figure 4.22, which illustrates this equivalent parallel Wiener cascade representation of the Wiener–Bose model shown in Figure 4.21.

Note that this the parallel Wiener cascade representation is not unique. For example, $g_3(\tau)$ can be any linear combination: $\alpha h_1(\tau) + \beta h_2(\tau)$, with $\alpha\beta \neq 0$, provided that appropriate corrections are made to the nonlinearities in other pathways. Similarly, there is an extra degree of freedom in the first-order polynomial terms. Any one may be chosen arbitrarily, provided that the first-order terms of the other two nonlinearities are corrected.

It is also possible to convert a parallel Wiener cascade model into a Wiener–Bose structure. First, construct a filter bank with linearly independent IRFs (i.e., no IRF is a linear combination of the remaining IRFs in the filter bank). This may be done using QR factorization. Let $\mathbf{g_1} \ldots \mathbf{g_3}$ be vectors containing the impulse responses $g_1(\tau) \ldots g_3(\tau)$, and compute:

$$\begin{bmatrix} \mathbf{g_1} & \mathbf{g_2} & \mathbf{g_3} \end{bmatrix} = \mathbf{QR} = \begin{bmatrix} \mathbf{q_1} & \mathbf{q_2} & \mathbf{q_3} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}$$

where the columns of $\mathbf{Q}$ are orthogonal, so that $\mathbf{q_i}^T\mathbf{q_j} = \delta_{i,j}$. In this example, $\mathbf{g_3}$ was a linear combination of $\mathbf{g_1}$ and $\mathbf{g_2}$, so $r_{33} = 0$, and the filter bank needs to contain only two IRFs, $q_1(\tau)$ and $q_2(\tau)$.

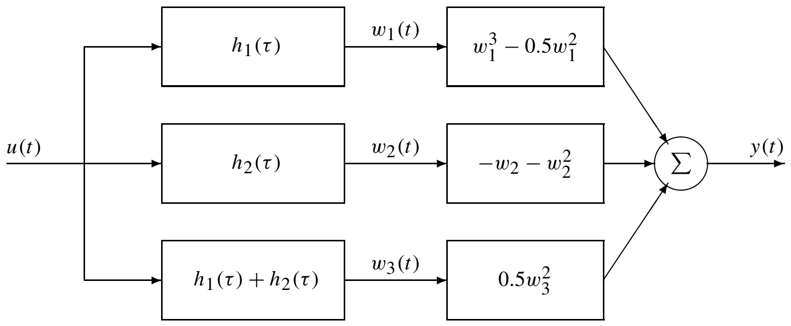

Figure 4.22 A parallel Wiener cascade structure that is equivalent to the Wiener–Bose model of Figure 4.21.

Let $x_1(t)$ and $x_2(t)$ be the outputs of the IRFs $q_1(\tau)$ and $q_2(\tau)$, respectively. Then, by superposition, we obtain

$$\begin{bmatrix}\mathbf{w_1} & \mathbf{w_2} & \mathbf{w_3}\end{bmatrix} = \begin{bmatrix}\mathbf{x_1} & \mathbf{x_2}\end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23}\end{bmatrix} \tag{4.84}$$

To construct the nonlinearity for the Wiener–Bose model, substitute equation (4.84) into equation (4.83) so the output is given by

$$\begin{aligned} y(t) &= -0.5w_1^2(t) + w_1^3(t) - w_2(t) - 0.5w_2^2(t) + 0.5w_3^2(t) \\ &= -r_{12}x_1(t) - r_{22}x_2(t) + 0.5\left(r_{13}^2 - r_{12}^2 - r_{11}^2\right)x_1^2(t) \qquad (4.85) \\ &+ (r_{13}r_{23} - r_{12}r_{22})\,x_1(t)x_2(t) \\ &+0.5\left(r_{23}^2 - r_{22}^2\right)x_2^2(t) + r_{11}^3x_1^3(t) \qquad (4.86) \end{aligned}$$

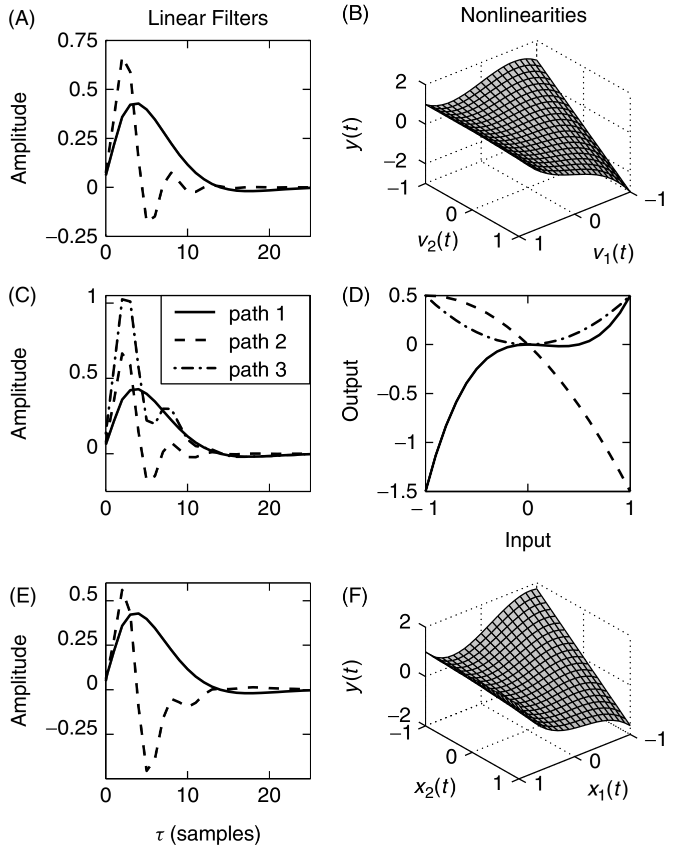

Figure 4.23 Three equivalent representations of a nonlinear system. (A) The IRFs of the linear elements ($h_1(\tau)$, solid, $h_2(\tau)$, dashed) and (B) the two-input nonlinearity of the Wiener–Bose model of Figure 4.21. (C) The IRFS and (D) Single-input nonlinearities of an equivalent parallel Wiener cascade. (E) The IRFs and (F) two-input nonlinearity of an equivalent Wiener–Bose model with an orthogonal filter bank.

If the IRFs in the original Wiener–Bose system were orthonormal, then $\mathbf{h_i}^T\mathbf{h_j} = \delta_{i,j}$, $r_{12} = 0$, and $r_{13} = r_{23} = 1$; the nonlinearity in equation (4.86) would then reduce to that in the original system (4.81). In all other cases, the nonlinearity will be different, because the filter bank will have been orthogonalized.

Figure 4.23 shows the three systems discussed in this example. The elements of the original Wiener–Bose model are illustrated in Figure 4.21A, which shows the IRFs of the two linear elements, $h_1(\tau)$ and $h_2(\tau)$, and in Figure 4.23B, which is a 3-D perspective plot showing the output of the static nonlinearity as a function of its two inputs. The elements of the equivalent parallel Wiener cascade model are shown next; the IRFs of the linear elements in Figure 4.23C and the SISO static nonlinearities in Figure 4.23D. Finally, Figures 4.23E and 4.23F show the elements of the Wiener–Bose model derived from the parallel cascade.

4.6 NOTES AND REFERENCES

1. See Volterra (1959) for the original derivation of the Volterra series.
2. Wiener's original derivation, in continuous time, assuming a Brownian motion input, can be found in Wiener (1958).
3. Schetzen wrote a thorough review article (Schetzen, 1981) on the Wiener series, as well as a textbook describing the Volterra and Wiener series (Schetzen, 1980).
4. Rugh (1981) contains descriptions of the Volterra series, the simple cascade models, and a derivation of the Wiener series, including the transformations that relate the Wiener and Volterra kernels.
5. Marmarelis and Marmarelis (1978) is the classic reference for Wiener series modeling and identification.

4.7 THEORETICAL PROBLEMS

1. Consider a system comprising the series connection of $\mathbf{h^{(n)}}(\tau_1, \ldots, \tau_n)$, an nth-order Volterra kernel, followed by $\mathbf{g^{(m)}}(\tau_1, \ldots, \tau_m)$.

(a) Write an expression that computes the output of this cascade of nonlinear systems.

(b) Show that the combined system is of order mn, and compute its Volterra kernel. (*Hint*: Use the derivation for the Volterra kernels of an LNL cascade as a guide.)

2. Write an expression, analogous to equations (4.74) and (4.75), for the third-order Volterra kernel of a Wiener–Bose model.

3. Repeat the derivation of the parallel Wiener cascade, presented in Section 4.5.5, but let $g_3(\tau) = \alpha h_1(\tau) + \beta h_2(\tau)$, for any nonzero constants α and β. Let the first-order polynomial coefficient in $m_1(\cdot)$ be $c_1^{(1)} = \gamma$, and compute the remaining first-order coefficients so that the parallel Wiener cascade is still equivalent to the Wiener–Bose model in the example.

4.8 COMPUTER EXERCISES

Run the function `ch4/examples.m`. This will create the example systems: `ch4_wiener`,`ch4_hammerstein`, and `ch4_cascade`.

1. Use the Wiener cascade `ch4_wiener` and

(a) Generate a 1000-point white Gaussian input (as an NLDAT object), and compute the system's output.

(b) Convert the model into a Volterra series, and plot the first- and second-order kernels. Use the Volterra series to compute the response to the Gaussian input. How long did it take? How does this compare to the time required by the Wiener cascade?

(c) Convert the model into a Wiener series, using several different values for the input variance. Does the size of the zero- to second-order kernels change with σ? Alter the polynomial coefficients in the nonlinearity, and recompute the Wiener kernels, again using several input power levels.

2. Repeat the previous problem, but use the Hammerstein cascade, `ch4_hammerstein`.

3. Plot the elements of `ch4_cascade`, and compute its Volterra kernels. Transform the model into a Wiener series, using $\sigma = 1$. Truncate the Wiener series by removing all but the zero-, first-, and second-order kernels. Transform the truncated Wiener series back into a Volterra series. What happened? Repeat this experiment using a different input power level.

4. Compute and plot the second-order Volterra kernel of `ch4_structure`. Extract a few slices of the kernel, and plot them superimposed. Are they proportional to each other? Compute and plot the first-order Volterra kernel of `ch4_structure`. Is it proportional to the slices of the second-order kernel? Is `ch4_structure` a Wiener system? Confirm your answer by examining its properties (i.e., type `get(ch4_structure)`).