4 Wiener Series

4.1 Introduction

Determining the Volterra kernels of an unknown system faces several practical problems: (1) the order of the system underlying the signal being investigated is usually unknown, and (2) the contributions of the individual components (of different order) of the Volterra series are not independent. The first problem is generally an issue if one wants to characterize a system with any type of series approximation, and the second problem can sometimes be resolved by the use of a series with orthogonal components. An example of the latter procedure is the development of the Fourier series; by having orthogonal terms, there are no dependencies between terms and one can determine the coefficients a_i and b_i of the Fourier series sequentially (Chapter 5, van Drongelen, 2007). To address the dependence between components in a series approximation for nonlinear systems, **Norbert Wiener developed an approach where each component in his Volterra-like series is orthogonal to all lower-order ones**. Although within the Wiener series approach one cannot predetermine the order of the system being studied either (problem (1) above), the orthogonality between the terms in the series allows one to determine the Wiener kernels sequentially. Subsequently one can determine their contribution to the signal, and stop the kernel-estimation process at the order where the signal is sufficiently approximated. For practical reasons most studies limit their kernel estimates at either the second order or (less often) at the third order. The third- and higher-order kernels require significant computation and they are difficult to depict.

In this chapter we will first discuss the main differences between the Wiener and Volterra series, after which we will derive the expressions for the zero-, first-, and second-order Wiener kernels, and then finally we will discuss practical methods for determining Wiener kernels for simulated and recorded time series. Applications of these methods will be presented in the form of MATLAB scripts. The last part of this chapter and Fig. 4.6 summarize the mathematical procedures we use to determine the Wiener series. For an extended background on this topic, see Marmarelis and Marmarelis (1978) and the reprint of Schetzen's book (Schetzen, 2006), and for recent engineering-oriented overviews see Westwick and Kearney (2003) and Marmarelis (2004).

Signal Processing for Neuroscientists, A Companion Volume. DOI: 10.1016/B978-0-12-384915-1.00004-8

4.2 Wiener Kernels

Similar to the Volterra series, the Wiener series characterizes the output z of a nonlinear system as the sum of a set of operators G_n dependent on kernels k_n and input x:

$$z(t) = \sum_{n=0}^{N} G_n[k_n; x(t)]$$

This equation is similar to the ones for the Volterra series (Equations (3.9a) and (3.9b)), but although there are many similarities between Volterra and Wiener series, there are also a few crucial differences that allow Wiener operators to be mutually independent. For clarity we first summarize the **three major differences** between the Volterra and Wiener series and then explain further details in the remainder of this chapter (e.g., the exact relationship between Volterra and Wiener kernels is discussed in Section 4.5).

The first major difference is that although a Volterra series usually does not include a zero-order term (see Equation (3.9)), we may define such a term as a constant:

$$H_0[x(t)] = h_0 \tag{4.1a}$$

In contrast, the Wiener series **always** includes a zero-order term. This term is defined as the average output (the DC component) equal to k_0:

$$\boxed{G_0[k_0; x(t)] = k_0} \tag{4.1b}$$

In this equation, k_0 is the zero-order Wiener kernel. We use k_n for the Wiener kernels to distinguish them from the Volterra kernels h_n.

The second major difference is that while individual Volterra operators are homogeneous (see, e.g., Equation (3.6) for a second-order one) (i.e., $H_n[cx(t)] = c^n H_n[x(t)]$), the Wiener operators are nonhomogeneous—for example, the first-order Wiener operator has a first-order and a derived zero-order component:

$$\begin{aligned} G_1[k_1; x(t)] &= g_1[k_1, k_{0(1)}; x(t)] = K_1[x(t)] + K_{0(1)}[x(t)] \\ &= \int_{-\infty}^{\infty} k_1(\tau_1) x(t - \tau_1) \mathrm{d}\tau_1 + k_{0(1)} \end{aligned} \tag{4.2}$$

The subscript 0(1) in $K_{0(1)}$ and $k_{0(1)}$ indicates that these are zero-order members of a first-order nonhomogeneous operator. Specifically, k_1 is the first-order Wiener kernel and $k_{0(1)}$ is the so-called derived Wiener kernel from operator G_1. In general, the Wiener kernels of the type $k_{n(m)}$ with $n < m$ are called derived Wiener kernels

because, as we will see below, they must be derived from Wiener kernel k_m. The notation with the capital G indicates that the operator includes both the kernel and input, while the lower-case notation g differs by explicitly also indicating all of the derived kernels.

The second-order Wiener operator is:

$$\begin{aligned} G_2[k_2;x(t)] &= g_2[k_2,k_{1(2)},k_{0(2)};x(t)] = K_2[x(t)] + K_{1(2)}[x(t)] + K_{0(2)}[x(t)] \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)\mathrm{d}\tau_1\mathrm{d}\tau_2 \\ &\quad + \int_{-\infty}^{\infty} k_{1(2)}(\tau_1)x(t-\tau_1)\mathrm{d}\tau_1 + k_{0(2)} \end{aligned} \tag{4.3}$$

The subscripts 0(2) and 1(2) indicate that these are zero- and first-order members (derived Wiener kernels) of the second-order nonhomogeneous operator G_2, respectively. Kernel k_2 is the second-order Wiener kernel, while $k_{1(2)}$ and $k_{0(2)}$ are derived Wiener kernels from operator G_2. As we will demonstrate below, the rationale for using nonhomogeneous operators relates to their orthogonality.

The third and final major difference is that in the case of a Wiener series we use a special input signal, usually in the form of zero mean Gaussian white noise (GWN) (alternative input signals are discussed in Section 4.7 and Chapter 5). Selection of a special input is critical because it allows us to create a series in which the **operators are orthogonal (uncorrelated) to the lower-order operators**. As we will see in Section 4.3, this property contributes to creating independence between the operators in the series, which will allow us to determine the Wiener kernels sequentially without having to worry about dependency issues.

The first-order Wiener operator is defined so that it is orthogonal to the zero-order Volterra operator:

$$E\{H_0[x(t)]g_1[k_1,k_{0(1)};x(t)]\} = \langle H_0[x(t)]g_1[k_1,k_{0(1)};x(t)]\rangle = 0 \tag{4.4a}$$

In the expression after the equal sign, $\langle\ldots\rangle$ indicates the time average.

> *Note*: $\langle x(t)\rangle$ represents the time average of a signal $x(t)$ over a time interval T. This is an alternative notation for the integral notation: $(1/T)\int_0^T x(t)\mathrm{d}t$.

Equation (4.4a) indicates that we assumed ergodicity so that we may use a time average $\left\langle H_0[x(t)]g_1[k_1,k_{0(1)};x(t)]\right\rangle$ to determine the Expectation $E\{\ldots\}$ of the product of H_0 and g_1. If you need to review the concepts of Expectation and time averages, see section 3.2 and appendix 3.1 in van Drongelen (2007). Details about time averages for GWN are reviewed in Appendix 4.1.

Similarly, the second-order Wiener operator is defined as orthogonal to zero- and first-order Volterra operators:

$$\langle H_0[x(t)]g_2[k_2, k_{1(2)}, k_{0(2)}; x(t)]\rangle = 0 \tag{4.4b}$$

$$\langle H_1[x(t)]g_2[k_2, k_{1(2)}, k_{0(2)}; x(t)]\rangle = 0 \tag{4.4c}$$

To characterize any nonlinear system of order N, this approach is generalized for all Wiener operators; that is, for zero mean GWN input, operator $G_n[k_n; x(t)] = g_n[k_n, k_{n-1(n)}, \ldots, k_{0(n)}; x(t)]$ is defined such that it is orthogonal to ***any*** Volterra operator of a lower order:

$$\langle H_m[x(t)]g_n[k_n, k_{n-1(n)}, \ldots, k_{0(n)}; x(t)]\rangle = 0 \quad \text{for } m < n \tag{4.4d}$$

In the following sections we will achieve orthogonality between the G operators and lower-order Volterra operators by using the so-called Gram−Schmidt technique (for details of this technique, see, e.g., Arfken and Weber, 2005). By defining the Wiener kernels according to this technique, we can determine the kernels of nonlinear systems from lower to higher order without knowledge of the system's components. This procedure is similar to approximating a function or signal with a Fourier series (van Drongelen, 2007, Chapter 5) or a polynomial (Section 2.4). For each kernel (for each order) we can determine its contribution to the output and we can continue to add higher-order terms until we are satisfied with our approximation of the system at hand. The procedure for determining the Wiener kernels as sketched above and their independence to lower-order kernels is summarized in Fig. 4.1. In the following sections we derive the expressions for the first- and second-order Wiener kernels. If you are interested in higher-order components, see Schetzen (2006).

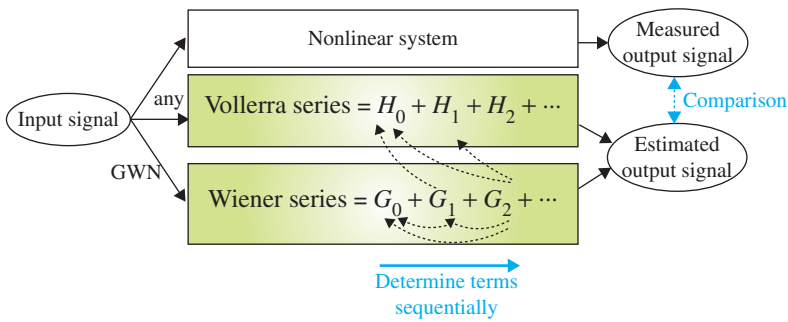

Figure 4.1 Representation of a nonlinear system by Volterra and Wiener series. In contrast to the Volterra operators H_n, the operators G_n in the Wiener series are independent from lower-order operators (stippled arrows). This allows one to determine the Wiener operators and their kernels sequentially and compute their contribution to the estimated output. The comparison with the measured output signal can be used to determine at what order the output is sufficiently approximated.

4.2.1 Derivation of the First-Order Wiener Operator

In the previous section we already identified the zero-order kernel as the signal's DC component in Equation (4.1b). Now we can use orthogonality defined in Equation (4.4) to derive the Wiener kernels k_1 and k_2. Starting with the first-order kernel, we substitute Equations (4.1a) and (4.2) in Equation (4.4a) and find that the following condition must be satisfied:

$$\langle H_0[x(t)]g_1[k_1,k_{0(1)};x(t)]\rangle = 0 \rightarrow$$
$$\left\langle h_0\left[\int_{-\infty}^{\infty} k_1(\tau_1)x(t-\tau_1)\mathrm{d}\tau_1 + k_{0(1)}\right]\right\rangle = h_0\left[\int_{-\infty}^{\infty} k_1(\tau_1)\langle x(t-\tau_1)\rangle \mathrm{d}\tau_1 + k_{0(1)}\right] = 0 \tag{4.5}$$

Note that in the expression after the equal sign we took all constants ($h_0, k_1(\tau_1)$, $k_{0(1)}$) out of the time average operation, such that only the (time-dependent) input time series x remains within the time average brackets $\langle \ldots \rangle$. **Now you will see how convenient it is to have zero mean GWN as input**. Because input x is zero mean GWN, the time average $\langle x(t-\tau_1)\rangle$ is zero. Therefore the integral in Equation (4.5) evaluates to zero and (since h_0 is generally not zero) we find that the orthogonality condition in Equation (4.5) is satisfied when:

$$\boxed{k_{0(1)} = 0} \tag{4.6}$$

Combining this result with Equation (4.2), we find that the first-order Wiener operator is:

$$\boxed{G_1[k_1;x(t)] = g_1[k_1;x(t)] = \int_{-\infty}^{\infty} k_1(\tau_1)x(t-\tau_1)\mathrm{d}\tau_1} \tag{4.7}$$

Note that for a first-order system (without a DC component) k_1 is the UIR (van Drongelen, 2007, Chapter 8). Further, if the input x is zero mean GWN, the output of the first-order (linear) operator G_1 will also be zero mean GWN. Because $\langle x(t-\tau_1)\rangle = 0$, the Expectation or the time average of G_1 is zero: that is, $E\{G_1\} = \langle G_1\rangle = 0$. Therefore G_1 is indeed orthogonal to any constant, such as zero-order operators G_0 and H_0.

4.2.2 Derivation of the Second-Order Wiener Operator

We can obtain the expression for the second-order operator g_2 using a procedure similar to the one we developed for the first-order one. Here we must deal separately with the independence between the second-order Wiener operator and the two lower-order Volterra operators H_0 and H_1, respectively.

4.2.2.1 Orthogonality Between H_0 and g_2

Using Equations (4.1a), (4.3), and (4.4b) we get:

$$\begin{aligned}
&\langle H_0[x(t)]g_2[k_2, k_{1(2)}, k_{0(2)}; x(t)]\rangle \\
&\quad = \left\langle h_0 \left[\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k_2(\tau_1, \tau_2) x(t-\tau_1) x(t-\tau_2) \mathrm{d}\tau_1 \, \mathrm{d}\tau_2 \right.\right. \\
&\qquad \left.\left. + \int_{-\infty}^{\infty} k_{1(2)}(\tau_1) x(t-\tau_1) \mathrm{d}\tau_1 + k_{0(2)} \right] \right\rangle = 0 \\
&\quad = h_0 \left[\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k_2(\tau_1, \tau_2) \big\langle x(t-\tau_1) x(t-\tau_2) \big\rangle \mathrm{d}\tau_1 \, \mathrm{d}\tau_2 \right. \\
&\qquad \left. + \int_{-\infty}^{\infty} k_{1(2)}(\tau_1) \big\langle x(t-\tau_1) \big\rangle \mathrm{d}\tau_1 + k_{0(2)} \right] = 0
\end{aligned} \tag{4.8}$$

As we did in Equation (4.5), we took the constants out of the time average $\langle \ldots \rangle$ such that only the time series x remains within it. Because the input is zero mean GWN with variance σ^2, the average $\langle x(t-\tau_1) \rangle = 0$ and the averaged product of both copies of input x is the autocorrelation R_{xx} (see van Drongelen, 2007, section 8.4.1):

$$\langle x(t-\tau_1) x(t-\tau_2) \rangle = R_{xx}(\tau_2 - \tau_1) = \sigma^2 \delta(\tau_2 - \tau_1).$$

Again we can see how convenient the zero mean GWN input is: the time average $\langle x(t-\tau_1) \rangle$ vanishes and time average $\langle x(t-\tau_1) x(t-\tau_2) \rangle$ can be simplified to the expression for the autocorrelation. (See also Appendix 4.1 for further details on averages of products of Gaussian variables.) Therefore Equation (4.8) becomes:

$$\begin{aligned}
&h_0 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k_2(\tau_1, \tau_2) \underbrace{\langle x(t-\tau_1) x(t-\tau_2) \rangle}_{R_{xx}(\tau_2 - \tau_1)} \mathrm{d}\tau_1 \, \mathrm{d}\tau_2 + h_0 k_{0(2)} \\
&= \sigma^2 h_0 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k_2(\tau_1, \tau_2) \delta(\tau_2 - \tau_1) \mathrm{d}\tau_1 \, \mathrm{d}\tau_2 + h_0 k_{0(2)}
\end{aligned}$$

The double integral on the right-hand side can be evaluated by using the sifting property while evaluating the integral for one of the time constants; here we integrate with respect to τ_2 and get:

$$\sigma^2 h_0 \int_{-\infty}^{\infty} k_2(\tau_1, \tau_1) \mathrm{d}\tau_1 + h_0 k_{0(2)} = 0 \rightarrow \boxed{k_{0(2)} = -\sigma^2 \int_{-\infty}^{\infty} k_2(\tau_1, \tau_1) \mathrm{d}\tau_1} \tag{4.9}$$

In this expression we can see that $k_{0(2)}$ is indeed a *derived* Wiener kernel because it is directly derived from Wiener kernel k_2.

4.2.2.2 Orthogonality Between H_1 and g_2

Subsequently we substitute expression for the first-order Volterra operator (see Equation (3.7)) and Equation (4.3) for the second-order Wiener operator in the orthogonality condition in Equation (4.4c):

$$\begin{aligned}\langle H_1[x(t)]g_2[k_2,k_{1(2)},k_{0(2)};x(t)]\rangle \\ = \left\langle \left[\int_{-\infty}^{\infty} h_1(\upsilon)x(t-\upsilon)\mathrm{d}\upsilon\right]\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)\mathrm{d}\tau_1\,\mathrm{d}\tau_2\right.\right. \\ \left.\left. + \int_{-\infty}^{\infty} k_{1(2)}(\tau_1)x(t-\tau_1)\mathrm{d}\tau_1 + k_{0(2)}\right]\right\rangle \end{aligned} \tag{4.10}$$

The above expression contains three terms. We will first show that the first and third terms always evaluate to zero if the input is zero mean GWN.

The **first** term:

$$\begin{aligned}&\left\langle \left[\int_{-\infty}^{\infty} h_1(\upsilon)x(t-\upsilon)\mathrm{d}\upsilon\right]\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)\mathrm{d}\tau_1\,\mathrm{d}\tau_2\right]\right\rangle \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_1(\upsilon)k_2(\tau_1,\tau_2)\langle x(t-\upsilon)x(t-\tau_1)x(t-\tau_2)\rangle\mathrm{d}\upsilon\,\mathrm{d}\tau_1\,\mathrm{d}\tau_2 = 0\end{aligned} \tag{4.11a}$$

evaluates to zero because of our choice of zero mean GWN as input and the odd product $\langle x(t-\upsilon)x(t-\tau_1)x(t-\tau_2)\rangle = 0$ (Appendix 4.1)—**again taking advantage of our choice of GWN as the input**.

The **third** term in Equation (4.10):

$$\left[\int_{-\infty}^{\infty} h_1(\upsilon)\langle x(t-\upsilon)\rangle\mathrm{d}\upsilon\right]k_{0(2)} = 0 \tag{4.11b}$$

also evaluates to zero because $\langle x(t-\upsilon)\rangle = 0$.

The **second** term in Equation (4.10) is:

$$\begin{aligned}&\left\langle \left[\int_{-\infty}^{\infty} h_1(\upsilon)x(t-\upsilon)\mathrm{d}\upsilon\right]\left[\int_{-\infty}^{\infty} k_{1(2)}(\tau_1)x(t-\tau_1)\mathrm{d}\tau_1\right]\right\rangle \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_1(\upsilon)k_{1(2)}(\tau_1)\langle x(t-\upsilon)x(t-\tau_1)\rangle\mathrm{d}\upsilon\,\mathrm{d}\tau_1\end{aligned} \tag{4.11c}$$

This second term is the only one that contains an even product of $x(t)$ and can be further evaluated using (again) the autocorrelation R_{xx} for the zero mean GWN with variance σ^2; that is, $\langle x(t-\upsilon)x(t-\tau_1)\rangle = R_{xx}(\tau_1-\upsilon) = \sigma^2\delta(\tau_1-\upsilon)$. This gives us:

$$\sigma^2 \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_1(\upsilon)k_{1(2)}(\tau_1)\delta(\tau_1-\upsilon)\mathrm{d}\upsilon\,\mathrm{d}\tau_1 = \sigma^2 \int_{-\infty}^{\infty} h_1(\tau_1)k_{1(2)}(\tau_1)\mathrm{d}\tau_1$$

In the above we evaluate the integral with respect to υ by using the sifting property. Because the first and third terms already evaluate to zero, the second term must be zero in order to satisfy the orthogonality condition in Equation (4.4c). We accomplish this by setting:

$$\boxed{k_{1(2)} = 0} \tag{4.12}$$

Substituting the results we obtained from the orthogonality conditions (in Equations (4.9) and (4.12)) into Equation (4.3), we find the second-order Wiener operator G_2:

$$\begin{aligned} &G_2[k_2; x(t)] = g_2[k_2, k_{1(2)}, k_{0(2)}; x(t)] \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)\mathrm{d}\tau_1\,\mathrm{d}\tau_2 - \underbrace{\sigma^2 \int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\mathrm{d}\tau_1}_{k_{0(2)}} \end{aligned} \tag{4.13}$$

Note that just as the Expectation for G_1 is zero, the expected output of G_2 is also zero:

$$\begin{aligned} &E\left\{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)\mathrm{d}\tau_1\,\mathrm{d}\tau_2 - \sigma^2 \int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\mathrm{d}\tau_1\right\} \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)E\{x(t-\tau_1)x(t-\tau_2)\}\mathrm{d}\tau_1\,\mathrm{d}\tau_2 - \sigma^2 \int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\mathrm{d}\tau_1 \end{aligned} \tag{4.14}$$

As for the time average of an even product of Gaussian variables, once again the Expectation is the autocorrelation: $E\{x(t-\tau_1)x(t-\tau_2)\} = \sigma^2\delta(\tau_1-\tau_2)$. Substituting this into Equation (4.14) gives:

$$\begin{aligned} &\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\sigma^2\delta(\tau_1-\tau_2)\mathrm{d}\tau_1\,\mathrm{d}\tau_2 - \sigma^2 \int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\mathrm{d}\tau_1 \\ &= \sigma^2 \int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\mathrm{d}\tau_1 - \sigma^2 \int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\mathrm{d}\tau_1 = 0 \end{aligned} \tag{4.15}$$

Here we evaluated the term with the double integral using the sifting property. Because the Expectation is zero, G_2 (just as G_1) is orthogonal to any constant including G_0. Even without knowledge about our derivation above, using the same approach, it is straightforward to show that operator G_2 is also designed to be orthogonal to G_1. This orthogonality can be evaluated via the Expectation of the product $E\{G_1, G_2\}$, which contains odd products of the random input variable $x(t)$. **The odd products evaluate to zero (Appendix 4.1), causing the Expectation to vanish**.

In the above we showed that the first-order Wiener operator is orthogonal to the zero-order one, and that the second-order operator is orthogonal to the first- and zero-order ones. We will not elaborate on this here, but in general the Wiener operators are constructed in a way that they are orthogonal to all lower-order ones. Higher-order Wiener kernels will not be derived here, but the derivation follows a similar procedure as described for the zero- to second-order kernels above. Details for these derivations can be found in Schetzen (2006).

4.3 Determination of the Zero-, First- and Second-Order Wiener Kernels

Now that we know how the expressions for the terms in the Wiener series are developed, it is time to examine how we might determine the terms from measured and simulated data sets. Recall also that we can determine the kernels sequentially because of the orthogonality property. The best-known method to establish Wiener kernels from measurements is the cross-correlation method first described by Lee and Schetzen (1965). If we deal with a nonlinear system of order N, and we present a zero mean GWN x at its input, we obtain output z as the sum of the Wiener operators G_n:

$$z(t) = \sum_{n=0}^{N} G_n[k_n; x(t)] \tag{4.16}$$

4.3.1 Determination of the Zero-Order Wiener Kernel

As we extend our results for the first- and second-order operators to all higher-order ones, we find that the Expectation of all Wiener operators G_n, except the zero-order operator G_0, is zero (see the last paragraph in Sections 4.2.1 and 4.2.2.2). Therefore, assuming an ergodic process (allowing the use of time averages for estimating Expectations), we find that the average of output signal z is:

$$\boxed{\langle z(t) \rangle = \sum_{n=0}^{N} \langle G_n[k_n; x(t)] \rangle = G_0[k_0; x(t)] = k_0} \tag{4.17}$$

Thus the zero-order Wiener kernel is obtained from the mean output (i.e., the output's DC component).

4.3.2 Determination of the First-Order Wiener Kernel

Here we show that we can get the first-order Wiener kernel of a system from the cross-correlation between its input x and output z:

$$\begin{aligned}\langle z(t)x(t-\upsilon_1)\rangle &= \langle G_0[k_0;x(t)]x(t-\upsilon_1)\rangle + \langle G_1[k_1;x(t)]x(t-\upsilon_1)\rangle \\ &\quad + \langle G_2[k_2;x(t)]x(t-\upsilon_1)\rangle + \cdots \\ &= \sum_{n=0}^{N}\langle G_n[k_n;x(t)]x(t-\upsilon_1)\rangle\end{aligned} \tag{4.18}$$

Recall that Wiener kernels are defined to be orthogonal to lower-order Volterra kernels. This property may be generalized to **all** lower-order Volterra kernels. Since the delay operator $x(t-\upsilon_1)$ can be presented as a first-order Volterra operator (Appendix 4.2), all Wiener operators G_n with $n \geq 2$ are orthogonal to $x(t-\upsilon_1)$ according to Equation (4.4d). Let us check this property by examining the outcome for $\langle z(t)x(t-\upsilon_1)\rangle$ by determining the outcome for the individual operators $G_0, G_1, G_2, \ldots$

Using Equation (4.1b) for $\boldsymbol{n=0}$:

$$\langle G_0[k_0;x(t)]x(t-\upsilon_1)\rangle = k_0\underbrace{\langle x(t-\upsilon_1)\rangle}_{0} = 0 \tag{4.19a}$$

Using Equation (4.7) for $\boldsymbol{n=1}$:

$$\begin{aligned}\langle G_1[k_1;x(t)]x(t-\upsilon_1)\rangle &= \int_{-\infty}^{\infty} k_1(\tau_1)\underbrace{\langle x(t-\tau_1)x(t-\upsilon_1)\rangle}_{\sigma^2\delta(\tau_1-\upsilon_1)}\mathrm{d}\tau_1 \\ &= \sigma^2\int_{-\infty}^{\infty} k_1(\tau_1)\delta(\tau_1-\upsilon_1)\mathrm{d}\tau_1 = \sigma^2 k_1(\upsilon_1)\end{aligned} \tag{4.19b}$$

For the second-order kernel we already know that $\langle G_1[k_1;x(t)]x(t-\upsilon_1)\rangle$ is zero because $x(t-\upsilon_1)$ can be considered a lower-order Volterra operator (Appendix 4.2). However, let us check the outcome of the cross-correlation anyway.

Using Equation (4.13) for $\boldsymbol{n=2}$:

$$\begin{aligned}&\langle G_2[k_2;x(t)]x(t-\upsilon_1)\rangle \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\underbrace{\langle x(t-\tau_1)x(t-\tau_2)x(t-\upsilon_1)\rangle}_{0}\mathrm{d}\tau_1\,\mathrm{d}\tau_2 \\ &\quad - \sigma^2\int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\underbrace{\langle x(t-\upsilon_1)\rangle}_{0}\mathrm{d}\tau_1 = 0\end{aligned} \tag{4.19c}$$

The above integrals evaluate to zero because they contain time average of odd products of the GWN input x. We now state (without further checking) that the remaining averaged products ($\langle G_n[k_n; x(t)]x(t-\upsilon_1)\rangle$, $n \geq 3$) are zero by using the property of Equation (4.4d). From the three expressions in Equations (4.19a) and (4.19c) we conclude that only the term for $n = 1$ is nonzero; therefore, we can determine the first-order Wiener kernel by combining Equations (4.18) and (4.19b):

$$\langle z(t)x(t-\upsilon_1)\rangle = \sigma^2 k_1(\upsilon_1) \rightarrow \boxed{k_1(\upsilon_1) = \frac{1}{\sigma^2}\langle z(t)x(t-\upsilon_1)\rangle} \tag{4.20}$$

Thus the first-order Wiener kernel can be found by the cross-correlation between input x and output z weighted by the variance of the input.

4.3.3 Determination of the Second-Order Wiener Kernel

Using an analogous procedure for the higher-order Wiener kernels, we can find the second-order kernel by using a (second-order) cross-correlation between output z and now two copies of input x:

$$\begin{aligned}
&\langle z(t)x(t-\upsilon_1)x(t-\upsilon_2)\rangle \\
&= \langle G_0[k_0; x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\rangle + \langle G_1[k_1; x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\rangle \\
&\quad + \langle G_2[k_2; x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\rangle + \cdots = \sum_{n=0}^{N} \langle G_n[k_n; x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\rangle
\end{aligned} \tag{4.21}$$

Because $x(t-\upsilon_1)x(t-\upsilon_2)$ can be presented as a second-order Volterra operator (Appendix 4.2), all Wiener operators G_n with $n \geq 3$ are orthogonal to $x(t-\upsilon_1)x(t-\upsilon_2)$ according to Equation (4.4d).

Using Equation (4.1b) for $\boldsymbol{n = 0}$:

$$\langle G_0[k_0; x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\rangle = k_0 \underbrace{\langle x(t-\upsilon_1)x(t-\upsilon_2)\rangle}_{\sigma^2\delta(\upsilon_1-\upsilon_2)} = k_0\sigma^2\delta(\upsilon_1-\upsilon_2) \tag{4.22a}$$

Using Equation (4.7) for $\boldsymbol{n = 1}$:

$$\langle G_1[k_1; x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\rangle = \int_{-\infty}^{\infty} k_1(\tau_1) \underbrace{\langle x(t-\tau_1)x(t-\upsilon_1)x(t-\upsilon_2)\rangle}_{0} \mathrm{d}\tau_1 = 0 \tag{4.22b}$$

Using Equation (4.13) for $\boldsymbol{n = 2}$:

$$\langle G_2[k_2; x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\rangle$$

$$= \overbrace{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\underbrace{\langle x(t-\tau_1)x(t-\tau_2)x(t-\upsilon_1)x(t-\upsilon_2)\rangle}_{A}\,\mathrm{d}\tau_1\,\mathrm{d}\tau_2}^{\text{I}}$$

$$- \overbrace{\sigma^2\int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\underbrace{\langle x(t-\upsilon_1)x(t-\upsilon_2)\rangle}_{\sigma^2\delta(\upsilon_1-\upsilon_2)}\,\mathrm{d}\tau_1}^{\text{II}} \tag{4.22c}$$

Using Wick's theorem (a theorem that relates higher-order moments to lower-order ones; Appendix 4.1), the average indicated by A in Equation (4.22c) (fourth-order correlation) can be written as:

$$\begin{aligned} A &= \langle x(t-\tau_1)x(t-\tau_2)x(t-\upsilon_1)x(t-\upsilon_2)\rangle \\ &= \underbrace{\langle x(t-\tau_1)x(t-\tau_2)\rangle}_{\sigma^2\delta(\tau_1-\tau_2)}\underbrace{\langle x(t-\upsilon_1)x(t-\upsilon_2)\rangle}_{\sigma^2\delta(\upsilon_1-\upsilon_2)} \\ &\quad + \underbrace{\langle x(t-\tau_1)x(t-\upsilon_1)\rangle}_{\sigma^2\delta(\tau_1-\upsilon_1)}\underbrace{\langle x(t-\tau_2)x(t-\upsilon_2)\rangle}_{\sigma^2\delta(\tau_2-\upsilon_2)} \\ &\quad + \underbrace{\langle x(t-\tau_1)x(t-\upsilon_2)\rangle}_{\sigma^2\delta(\tau_1-\upsilon_2)}\underbrace{\langle x(t-\tau_2)x(t-\upsilon_1)\rangle}_{\sigma^2\delta(\tau_2-\upsilon_1)} \end{aligned}$$

This allows us to separate Part I of the expression in Equation (4.22c) into the following three terms:

(1) $\sigma^4\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\delta(\tau_1-\tau_2)\delta(\upsilon_1-\upsilon_2)\mathrm{d}\tau_1\,\mathrm{d}\tau_2 = \sigma^4\int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\delta(\upsilon_1-\upsilon_2)\mathrm{d}\tau_1$

(2) $\sigma^4\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\delta(\tau_1-\upsilon_1)\delta(\tau_2-\upsilon_2)\mathrm{d}\tau_1\,\mathrm{d}\tau_2 = \sigma^4 k_2(\upsilon_1,\upsilon_2)$

(3) $\sigma^4\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\delta(\tau_1-\upsilon_2)\delta(\tau_2-\upsilon_1)\mathrm{d}\tau_1\,\mathrm{d}\tau_2 = \sigma^4 k_2(\upsilon_2,\upsilon_1) = \sigma^4 k_2(\upsilon_1,\upsilon_2)$

The three integrals above are evaluated using the sifting property. Furthermore, by using the same symmetry property of the Volterra kernels (Section 3.2.1), we have concluded that k_2 is symmetrical and that the terms in 2 and 3 above are identical.

Combining the results for 1−3 in Part I and the integral term Part II in Equation (4.22c) we get:

$$\begin{aligned}\left\langle G_2[k_2;x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\right\rangle &\\ \overbrace{\qquad\qquad\qquad\qquad\qquad\qquad}^{\text{I}} &\\ = 2\sigma^4 k_2(\upsilon_1,\upsilon_2) + \sigma^4 \int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\delta(\upsilon_1-\upsilon_2)\mathrm{d}\tau_1 &\\ \overbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}^{\text{II}} &\\ -\ \sigma^2 \int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\underbrace{\left\langle x(t-\upsilon_1)x(t-\upsilon_2)\right\rangle}_{\sigma^2\delta(\upsilon_1-\upsilon_2)}\mathrm{d}\tau_1 &\end{aligned} \tag{4.22d}$$

The integral terms in the expression above cancel so that the final result becomes:

$$\langle G_2[k_2;x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\rangle = 2\sigma^4 k_2(\upsilon_1,\upsilon_2) \tag{4.22e}$$

According to Equation (4.4d) all Wiener operators for $\boldsymbol{n>2}$ are defined so that their contributions will be zero. This allows us to combine Equations (4.21) with (4.22a), (4.22b), and (4.22e):

$$\langle z(t)x(t-\upsilon_1)x(t-\upsilon_2)\rangle = k_0\sigma^2\delta(\upsilon_1-\upsilon_2) + 2\sigma^4 k_2(\upsilon_1,\upsilon_2) \tag{4.23}$$

Now we decide to ignore the case when $\upsilon_1=\upsilon_2$ and assume that $\upsilon_1\neq\upsilon_2$ so that $\delta(\upsilon_1-\upsilon_2)=0$. In this case the first term on the right-hand side of Equation (4.23) evaluates to zero. Therefore, for the off-diagonal part ($\upsilon_1\neq\upsilon_2$) of the second-order Wiener kernel we have:

$$\boxed{k_2(\upsilon_1,\upsilon_2) = \frac{1}{2\sigma^4}\langle z(t)x(t-\upsilon_1)x(t-\upsilon_2)\rangle \quad \text{for } \upsilon_1\neq\upsilon_2} \tag{4.24}$$

The second-order Wiener kernel is the second-order cross-correlation between output and input weighted by $2\sigma^4$. Our trick to ignore the diagonally located terms may seem a bit strange but in practical applications, **the limitation imposed by $\upsilon_1\neq\upsilon_2$ does not present a problem because we can compute k_2 for delays that are arbitrarily close to $\upsilon_1=\upsilon_2$.**

4.4 Implementation of the Cross-Correlation Method

In this section we present a practical application for finding Wiener kernels associated with a given nonlinear system (Fig. 4.2). MATLAB implementations of this approach are in Pr4_1.m and Pr4_2.m. In principle we can use Equations (4.17), (4.20), and (4.24) to determine the Wiener kernels. However, since our input of random noise is necessarily finite, the subsequent kernels may not be exactly orthogonal. To mitigate the effects of this problem, it is common practice to

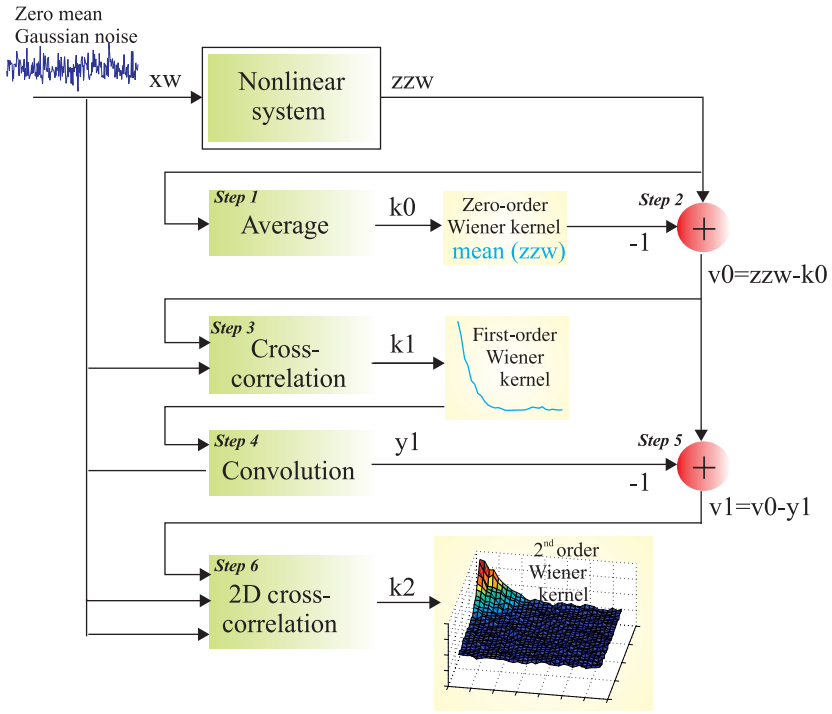

Figure 4.2 Lee−Schetzen cross-correlation method for obtaining the zero-, first-, second-order Wiener kernels of a nonlinear system. Zero mean GWN is used as the input (xw) of a nonlinear system. The average of the output (zzw) is used to estimate the zero-order kernel k0. The residue v0 (= zzw-k0) is then cross-correlated with the input to estimate the first-order kernel k1. Subsequently, the contribution y1 of the first-order kernel is determined by convolving it with the input. Finally, the residue v1 (= v0-y1) is correlated with two copies of the input (2D cross-correlation) for the estimation of k2.

determine the kernels from low to higher orders sequentially while at each step subtracting the contribution of the lower-order kernels from the output (−1, + operations in Fig. 4.2). For example, before computing k_1, it is common practice **to subtract k_0** (the DC component) from the output to obtain a zero-order residue v_0. This residue v_0 ($= z - k_0$), instead of the output z, is then cross-correlated with the input to obtain the first-order kernel k_1 (recall Equation (4.20)):

$$k_1(v_1) = \frac{1}{\sigma^2}\langle v_0(t)x(t - v_1)\rangle \tag{4.25}$$

To estimate the first-order kernel's contribution (y_1) to the output, the first-order kernel k_1 is convolved with the input x: $y_1 = x \otimes k_1$. This first-order contribution y_1 is then subtracted from the zero-order residue v_0 to obtain the first-order residue v_1.

The residue v_1 ($= z - k_0 - y_1$) is now cross-correlated with two copies of the input to estimate the second-order kernel k_2:

$$k_2(\upsilon_1, \upsilon_2) = \frac{1}{2\sigma^4} \langle v_1(t)x(t-\upsilon_1)x(t-\upsilon_2)\rangle \tag{4.26}$$

Note that as in Equation (4.24), the above expression is valid only for off-diagonal values with $\upsilon_1 \neq \upsilon_2$. This procedure is followed in the MATLAB programs and is depicted in Fig. 4.2. The input is variable xw; the output is zzw. The zero- and first-order residues are v0 and v1, respectively. The Wiener kernels are k0, k1, and k2.

An example of a MATLAB implementation can be found in Pr4_1.m and Pr4_2.m. A snippet of the latter is shown here.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%% Estimation of the Wiener kernel estimation using
%%%%%%%%%%%%%%%% the Lee, Schetzen cross-correlation method
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% First create a set of input output using random noise
xw=randn(10000,1);        % create array with Gaussian white noise
xw=xw-mean(xw);
N=length(xw);
st=std(xw);

figure;subplot(2,1,1),plot(xcorr(xw),'k');
title('Autocorrelation of the Input Shows a Random Noise Characteristic');
subplot(2,1,2);hist(xw);
title('Amplitude Distribution of the Input -> Gaussian');

yw_previous1=0;
yw_previous2=0;
for n=1:length(xw);
   ywh1(n)=(A1*yw_previous1+xw(n))/(A1+1);     % the 1st order operator
   yw_previous1=ywh1(n);
   ywh2(n)=(A2*yw_previous2+xw(n))/(A2+1);     % the linear component of
                                               % the 2nd order operator
   yw_previous2=ywh2(n);
   zzw(n)=ywh1(n)+ywh2(n)^2;        % 1st order component+the squarer
end;

figure; hold;
plot(xw,'k');plot(zzw,'r')
title('Input (black) and Output (red) of a Wiener System')
xlabel('Time (ms)');ylabel('Amplitude')
```

```
%%%%%% The Lee Schetzen Cross-correlation Method
%————————————————————————————————————————
% Step 1 (Fig. 4.1): Determine 0-order Wiener kernel
%————————————————————————————————————————
k0=mean(zzw)
y0=ones(1,length(xw))*k0;
% Step 2 (Fig. 4.1): Subtract k0 from the response to find residue % v0———
%————————————————————————————————————————
v0=zzw-k0;
% Step 3 (Fig. 4.1): Estimate k1 by first-order
% cross-correlation of v0 and input
%————————————————————————————————————————

for i=0:T-1
   temp=0;
   for n=i+1:N
      temp=temp+v0(n)*xw(n-i);
   end;
   k1(i+1)=temp/(N*st^2);
end;

figure; plot(k1);
title('first-order Wiener kernel')

% Step 4 (Fig. 4.1): Compute the output of the first-order
%                    Wiener kernel using convolution
%————————————————————————————————————————

for n=1:N;
   temp=0;
   for i=0:min([n-1 T-1]);
      temp=temp+k1(i+1)*xw(n-i);
   end;
   y1(n)=temp;
end;

% Step 5 (Fig. 4.1): Compute the first-order residue
%————————————————————————————————————————
v1=v0-y1;
% Step 6 (Fig. 4.1): Estimate k2 by second-order cross-correlation
%                    of v1 with the input
%————————————————————————————————————————

for i=0:T-1
   for j=0:i
      temp=0;
      for n=i+1:N
```

```
        temp=temp+v1(n)*xw(n-i)*xw(n-j);
      end;
      k2(i+1,j+1)=temp/(2*N*st^4);
      k2(j+1,i+1) = k2(i+1,j+1);
    end;
end;

figure; surf(k2(1:T,1:T));
title('second-order Wiener Kernel');
view(100,50);
```

The MATLAB script Pr4_2.m computes the Wiener kernels for a combined system such as the cascade discussed in the previous chapter (Fig. 3.2C). In this example we use low-pass filters for the linear components and a squarer for the nonlinear one (Fig. 4.3).

In the example in Fig. 4.3, the Lee−Schetzen method is used to determine the Wiener kernels (Fig. 4.3C). Here the kernels are used to predict the output by convolving the input with the kernels and adding up the contributions from each kernel (Fig. 4.3D). It can be seen that the predicted and recorded output match very well; this can be further confirmed when we compute the % variance that is accounted for (VAF) as:

```
VAF = (1-(std(zzw-est)^2)/(std(zzw)^2))*100
```

Here zzw and est are the measured and estimated output, respectively, and std is the MATLAB command to compute the standard deviation.

4.5 Relation between Wiener and Volterra Kernels

To summarize the preceding sections, a principal problem with the Volterra series is the dependence between the convolution-like terms (operators) in the series. This dependence prevents us from determining each term separately; this problem is resolved by Wiener's approach. To achieve the independence between terms, Wiener modified the individual terms in the series (Wiener operators are nonhomogeneous) and adapted the input (zero mean GWN). Volterra operators H_n have Volterra kernels (h_0, h_1, h_2, ...), whereas Wiener operators G_n have Wiener kernels (k_0, k_1, k_2, ...) as well as derived Wiener kernels ($k_{0(1)}$, $k_{0(2)}$, $k_{1(2)}$, ...).

Both Wiener and Volterra kernels are equivalent in the sense that the Wiener kernels can be determined from the Volterra kernels and vice versa. In our examples above we considered the zero- to the second-order kernels; let us assume that we are

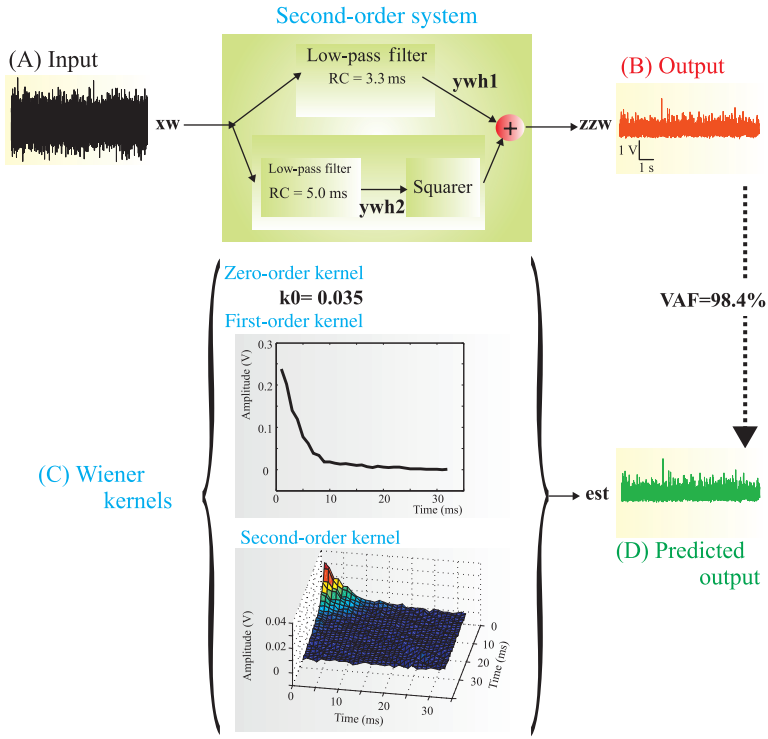

Figure 4.3 Wiener kernels of a second-order system similar to the one depicted in Fig. 3.2C; the example is computed with Pr4_2.m. (A) The input signal xw is GWN. (B) The output signal is zzw. (C) zero-, first- and second-order Wiener kernels computed by the MATLAB script using the procedure depicted in Fig. 4.2. (D) The predicted output est on the basis of the Wiener kernels approximates the measured output well: the variance accounted for VAF is 98.4%.

looking into a second-order system so that these are the only kernels available (all higher-order kernels are zero). In this case we have the following kernel components: k_0, $k_{0(1)}$, $k_{0(2)}$, k_1, $k_{1(2)}$, and k_2. In this example the Volterra kernels h_0–h_2 are:

$$\begin{aligned} h_0 &= k_0 + k_{0(1)} + k_{0(2)} = k_0 + k_{0(2)} \\ h_1 &= k_1 + k_{1(2)} = k_1 \\ h_2 &= k_2 \end{aligned} \tag{4.27}$$

The above equations for h_0–h_2 simplify because $k_{0(1)}$ and $k_{1(2)}$ are zero (see Equations (4.6) and (4.12)). So in a second-order system the relationship between the Wiener and Volterra kernels is fairly straightforward. Had we looked into a higher-order system, for example in a third-order system, we would add $k_{1(3)}$ to h_1 in Equation (4.27). The expressions for h_0 and h_2 remain unaltered because the other derived third-order kernels $k_{0(3)}$ and $k_{2(3)}$ are zero (Schetzen, 2006). Again,

the rationale for this redistribution of kernel components is to create independence between the operators in the series (the condition in Equation (4.4d)). For example, by moving the term $k_{0(2)}$ from the zero-order expression (h_0) to the second-order Wiener operator, we satisfy the independence between the second-order Wiener operator and H_0 (Equation (4.8)). Considering the relationships in Equation (4.27), it is unsurprising that a comparison between our findings for the Wiener kernels k_1 and k_2, obtained with Pr4_2.m (depicted in Fig 4.3C), and the Volterra kernels h_1 and h_2, found in Pr3_2.m from the previous chapter, reveals a close resemblance.

From this chapter we can deduce that to obtain the Volterra kernels, we must know the system's order as well as all the Wiener kernels. In an experimental situation one usually does not know the system's order; at best one could estimate the order by establishing the number of Wiener kernels required to (sufficiently) approximate the system's output signal. In most experimental studies the Wiener kernels (up to the second or third order) and their contributions to the system's output are determined without any further attempt to identify the Volterra kernels.

4.6 Analyzing Spiking Neurons Stimulated with Noise

When studying intracellular or extracellular recordings of a spiking neuron while stimulating it with noise, one might (of course) use the raw output trace (including the action potentials) and relate this to the input as we have done previously (Fig. 4.2). However, instead of using the neuron's raw output signal, we can use alternative methods to represent the action potential activity. In the following discussion we assume that timing is the only relevant information associated with a neuronal spiking event. Methods that consider only spike timing can be applied to both intracellular and extracellular recordings of single cells. When dealing with high levels of spike activity it is common to represent the cell's output as the instantaneous spike rate (defined as (interspike interval)$^{-1}$) plotted vs. time; this procedure is shown in Fig. 1.1. Another frequently used technique is to bin the spike train and plot the number of spikes per bin against the time-stamp of the bin. However, if spike rates are low, both of these methods are impractical because we obtain time series that are either extremely unevenly sampled or too sparsely populated with values other than zeros and ones. In general, if one is interested only in the spike train, it seems reasonable to present the output time series of N spikes occurring at times t_i as a series of delta functions, thereby ignoring small subthreshold fluctuations of the neuronal response or noise in the recordings (chapter 14 in van Drongelen, 2007).

With a little bit of work, the Schetzen correlation method can be adapted to analyze spiking neurons stimulated by GWN. An example for the auditory system was described by Recio-Spinoso et al. (2005). In this study, the auditory system is stimulated by auditory noise and the authors represent the neuron's output y (a spike train of N spikes) as a series of Diracs at times t_i:

$$y(t) = \sum_{i=1}^{N} \delta(t - t_i) \tag{4.28}$$

Following our result in Equation (4.17), the **zero-order Wiener kernel** is the time average of the system's output:

$$k_0 = \langle y(t) \rangle = \left\langle \sum_{i=1}^{N} \delta(t - t_i) \right\rangle \tag{4.29a}$$

The time average $\langle \ldots \rangle$ can be written as an integral over the interval [0,T], divided by epoch length T (that is $(1/T)\int_0^T \cdots$) :

$$\frac{1}{T}\int_0^T \sum_{i=1}^{N} \delta(t - t_i)\mathrm{d}t = \frac{1}{T}\sum_{i=1}^{N}\int_0^T \delta(t - t_i)\mathrm{d}t \tag{4.29b}$$

Here we interchanged the integration and summation operation. The timing t_i for each spike i is between 0 and T, so consequently the Dirac $\delta(t - t_i)$ is located within the [0,T] integration interval and the integral $\int_0^T \delta(t - t_i)\mathrm{d}t$ evaluates to 1 (see Section 2.2.2 in van Drongelen, 2007). Therefore, the expression in Equation (4.29) simply counts the number N of action potentials divided by the time epoch T. Thus the zero-order Wiener kernel evaluates to N/T, which is the neuron's mean firing rate N_0:

$$\boxed{k_0 = \frac{N}{T} = N_0} \tag{4.30}$$

The **first-order Wiener kernel** is given by Equation (4.20):

$$k_1(\tau_1) = \frac{1}{\sigma^2}\langle y(t)x(t - \tau_1)\rangle \tag{4.31}$$

If we rewrite the time average $\langle \ldots \rangle$ as an integral and substitute the output z in Equation (4.20) with the spike time series y (given in Equation (4.28)), we get:

$$\begin{aligned} k_1(\tau_1) &= \frac{1}{\sigma^2}\left[\overbrace{\frac{1}{T}\int_0^T \underbrace{\left(\sum_{i=1}^{N}\delta(t - t_i)\right)}_{\text{output}} \underbrace{x(t - \tau_1)}_{\text{input}}\,\mathrm{d}t}^{\text{Time average}}\right] \\ &= \frac{1}{\sigma^2}\left[\frac{1}{T}\int_0^T \left(\sum_{i=1}^{N}\delta(t - t_i)x(t - \tau_1)\right)\mathrm{d}t\right] \end{aligned} \tag{4.32}$$

In the above we included input x in the summation. Now we again interchange the summation and integration operations:

$$k_1(\tau_1) = \frac{1}{\sigma^2}\left[\frac{1}{T}\sum_{i=1}^{N}\underbrace{\int_0^T \delta(t-t_i)x(t-\tau_1)\mathrm{d}t}_{x(t_i-\tau_1)}\right] = \frac{1}{\sigma^2}\underbrace{\frac{1}{T}N}_{N_0}\underbrace{\frac{1}{N}\sum_{i=1}^{N}x(t_i-\tau_1)}_{R_1(\tau_1)} \tag{4.33}$$

Here we evaluated the integral using the sifting property and multiplied the expression by N/N to allow substitution of $\boldsymbol{R_1(\tau_1)}$**, the reverse-correlation function** (see section 14.5 in van Drongelen, 2007). The reverse-correlation function is also known as the revcor, which can be determined by averaging the stimulus time course that precedes each spike (spike-triggered average). If we think of the zero-order kernel as the time average (mean firing rate) of the system's output, we can conceptualize the first-order Wiener kernel as the average stimulus value some time τ_1 before spike i occurs (i.e., $x(t_i - \tau_1)$). Simplifying notation, we finally get:

$$\boxed{k_1(\tau_1) = \frac{N_0}{\sigma^2}R_1(\tau_1)} \tag{4.34}$$

The **second-order Wiener kernel**, on the other hand, represents the mean of the product of two copies of the input x (at two times τ_1 and τ_2) before the occurrence of a spike. The second-order Wiener kernel as given by Equation (4.24) becomes:

$$k_2(\tau_1, \tau_2) = \frac{1}{2\sigma^4}\langle y(t)x(t-\tau_1)x(t-\tau_2)\rangle \tag{4.35}$$

In Recio-Spinoso et al. (2005), the above equation is corrected by subtracting the zero-order kernel k_0 from the output. This makes sense for the following reasons. As discussed above, subtracting the contribution of lower-order kernels from the output is common practice (Fig. 4.2). In Equation (4.32) we did not correct the output for the first-order kernel estimate because theoretically its contribution should be independent from the zero-order one ($k_{0(1)}$ is zero, Equation (4.6)). However, we do correct for the DC (constant) term in the second-order estimate because a nonzero zero-order component $k_{0(2)}$ does exist (see Equation (4.9)). We will not correct y for the first-order contribution to k_2 because theoretically $k_{1(2)}$ is zero (Equation (4.12)). Therefore, y in Equation

(4.35) can simply be corrected for the zero-order contribution N_0 by using the output y minus the zero-order kernel:

$$y(t) - k_0 = \sum_{i=1}^{N} \delta(t - t_i) - N_0 \tag{4.36}$$

By doing this we get:

$$k_2(\tau_1, \tau_2) = \frac{1}{2\sigma^4} \left\langle \left[\sum_{i=1}^{N} \delta(t - t_i) - N_0 \right] x(t - \tau_1) x(t - \tau_2) \right\rangle \tag{4.37a}$$

Writing the time average in the integral notation, we get:

$$= \frac{1}{2\sigma^4} \left\{ \frac{1}{T} \int_0^T \left[\sum_{i=1}^{N} \delta(t - t_i) - N_0 \right] x(t - \tau_1) x(t - \tau_2) \mathrm{d}t \right\} \tag{4.37b}$$

We can write the expression as two separate integral terms:

$$= \frac{1}{2\sigma^4} \left\{ \frac{1}{T} \int_0^T \sum_{i=1}^{N} \delta(t - t_i) x(t - \tau_1) x(t - \tau_2) \mathrm{d}t - \frac{1}{T} \int_0^T N_0 x(t - \tau_1) x(t - \tau_2) \mathrm{d}t \right\} \tag{4.37c}$$

By changing the integration and summation order in the **first term** and applying the sifting property for the Dirac, we get the following expression for the first term:

$$\frac{1}{2\sigma^4} \frac{1}{T} \sum_{i=1}^{N} \underbrace{\int_0^T \delta(t - t_i) x(t - \tau_1) x(t - \tau_2) \mathrm{d}t}_{x(t_i - \tau_1) x(t_i - \tau_2)} = \frac{1}{2\sigma^4} \frac{1}{T} \sum_{i=1}^{N} x(t_i - \tau_1) x(t_i - \tau_2) \tag{4.38a}$$

As we did with the first-order kernel earlier, we can multiply by N/N to simplify notation by using the expression for the **second-order reverse correlation** $R_2(\tau_1, \tau_2) = (1/N) \sum_{i=1}^{N} x(t_i - \tau_1) x(t_i - \tau_2)$. Finally, the first term in Equation (4.37c) simplifies to:

$$\frac{1}{2\sigma^4} \underbrace{\frac{1}{T} N}_{N_0} \underbrace{\frac{1}{N} \sum_{i=1}^{N} x(t_i - \tau_1) x(t_i - \tau_2)}_{R_2(\tau_1, \tau_2)} = \frac{N_0}{2\sigma^4} R_2(\tau_1, \tau_2) \tag{4.38b}$$

The **second term** in Equation (4.37c) is:

$$= \frac{1}{2\sigma^4}\frac{1}{T}\int_0^T -N_0 x(t-\tau_1)x(t-\tau_2)\mathrm{d}t$$

$$= -\frac{N_0}{2\sigma^4}\underbrace{\frac{1}{T}\int_0^T x(t-\tau_1)x(t-\tau_2)\mathrm{d}t}_{\phi(\tau_2-\tau_1)} = -\frac{N_0}{2\sigma^4}\phi(\tau_2-\tau_1) \tag{4.39}$$

The expression $(1/T)\int_0^T x(t-\tau_1)x(t-\tau_2)\mathrm{d}t$ is the autocorrelation $\phi(\tau_2-\tau_1)$ of the input noise. Note that unlike the variable t_i (representing the spike times) in the expression for the reverse correlation R_2, the time variable t is continuous in ϕ. Combining the results for the first and second terms we finally get:

$$\boxed{k_2(\tau_1,\tau_2) = \frac{N_0}{2\sigma^4}[R_2(\tau_1,\tau_2) - \phi(\tau_2-\tau_1)]} \tag{4.40a}$$

The above approach was used by Recio-Spinoso et al. (2005) to determine the first- and second-order Wiener kernels of different types of auditory nerve fibers. An example of the second-order kernel for a so-called low-characteristic frequency nerve fiber is shown in Fig. 4.4. If the input is zero mean GWN, we have $\phi(\tau_2-\tau_1) = \sigma^2\delta(\tau_2-\tau_1)$. Then, because we decided to ignore values where $\tau_1 = \tau_2$, as we did in Equation (4.24), we get:

$$\boxed{k_2(\tau_1,\tau_2) = \frac{N_0}{2\sigma^4}R_2(\tau_1,\tau_2) \quad \text{for } \tau_1 \neq \tau_2} \tag{4.40b}$$

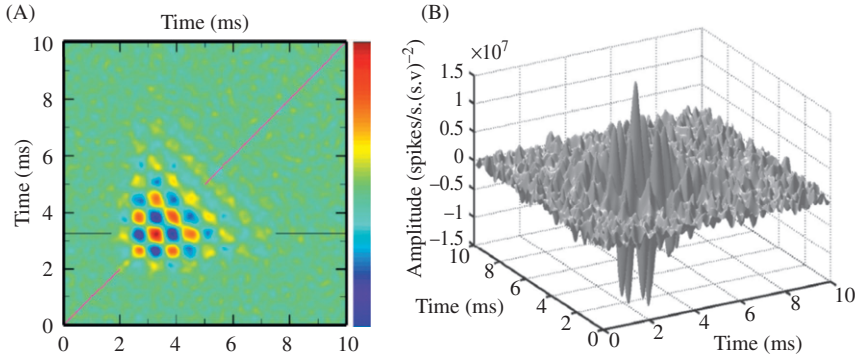

Figure 4.4 Example of a second-order kernel of an auditory low-characteristic frequency nerve fiber. (A) A 2D color-coded presentation of k_2. (B) The corresponding 3D plot of k_2. (*Panel A color in electronic version.*) (*From Recio-Spinoso et al., 2005*)

4.7 Nonwhite Gaussian Input

Zero mean GWN was selected as input signal for the determination of the Wiener series. In real applications, however, this is not feasible because the bandwidth of the noise is limited. In some cases, the bandwidth of the noise at the input may be wide enough relative to the bandwidth that is relevant for the system under investigation that we may consider the noise as white. However, there are situations where such an assumption is not valid. In these cases the input noise is band-limited (colored). The effect of using colored noise as input will be analyzed and discussed in the following paragraphs.

Recall that in Equation (4.40a) we left the noise autocorrelation term $\varphi(\tau_2 - \tau_1)$ in the expression. In Equation (4.40b), under the condition that the input is zero mean GWN, we ignored the correlation term because $\sigma^2\delta(\tau_2 - \tau_1)$ evaluates to zero for $\tau_1 \neq \tau_2$. In general, when we consider systems, the noise presented at the input may be zero mean and Gaussian, but nonwhite (Gaussian colored noise [GCN]). The term "white" indicates that all frequencies are equally present in the noise signal, while in colored noise not all frequencies are equally present (i.e., we are dealing with filtered white noise). The filter effect has a direct consequence on the autocorrelation of the noise input (Fig. 4.5A and B). However, both colored and white noise may be Gaussian, a property that is related to their amplitude distribution (Fig. 4.5C and D).

In the following we assume that we have determined the zero-order kernel as the mean output and that we deal only with demeaned signals for input and output. Under this assumption, the cross-correlation φ_{xz} between GCN input x and output z can be developed similar to the procedure shown in Equations (4.18)−(4.20):

$$\begin{aligned}\phi_{xz}(\upsilon_1) &= \langle z(t)x(t-\upsilon_1)\rangle = \sum_{n=0}^{N}\langle G_n[k_n; x(t)]x(t-\upsilon_1)\rangle \\ &= \int_{-\infty}^{\infty} k_1(\tau_1)\langle x(t-\tau_1)x(t-\upsilon_1)\rangle \mathrm{d}\tau_1 = \int_{-\infty}^{\infty} k_1(\tau_1)\phi_{xx}(\tau_1-\upsilon_1)\mathrm{d}\tau_1\end{aligned} \tag{4.41}$$

The above shows that the cross-correlation ϕ_{xz} is the convolution of the first-order kernel k_1 with the input autocorrelation φ_{xx}:

$$\phi_{xz} = k_1 \otimes \phi_{xx} \tag{4.42a}$$

Therefore k_1 can be obtained from the associated deconvolution. In the frequency domain convolution and deconvolution can be simplified to multiplication and division, respectively (section 8.3.2 in van Drongelen, 2007). The equivalent of Equation (4.42a) in the frequency domain therefore is:

$$\Phi_{xz} = K_1\Phi_{xx} \rightarrow K_1 = \Phi_{xz}/\Phi_{xx} \tag{4.42b}$$

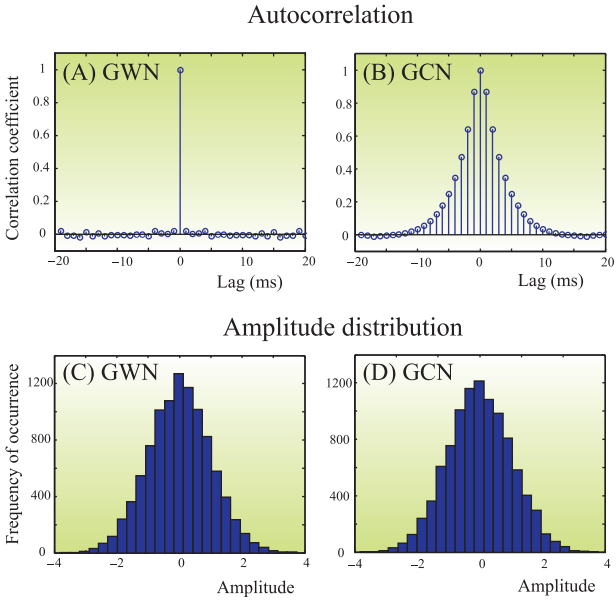

Figure 4.5 Autocorrelations (A and B) and amplitude distributions (C and D) of sampled Gaussian noise signals. The cases for GWN are depicted in (A) and (C). The same properties for colored (filtered) noise are shown in (B) and (D).

Here Φ_{XZ}, Φ_{XX}, and K_1 are the Fourier transforms of ϕ_{XZ}, ϕ_{XX}, and k_1, respectively. Now recall that the cross- and autocorrelation in the frequency domain can also be expressed as products (section 8.4.2 in van Drongelen, 2007) X^*Z and X^*X (where X and Z are the Fourier transforms of x and z, respectively, and * indicates the complex conjugate). Substituting these expressions for cross- and autocorrelation we get:

$$K_1 = X^*Z/X^*X \tag{4.42c}$$

In real applications we can use this expression to determine K_1 by averaging Φ_{XZ} (X^*Z) and Φ_{XX} (X^*X) for each frequency f over a number of epochs:

$$K_1(f) = \langle X(f)^* Z(f) \rangle / \langle X(f)^* X(f) \rangle \tag{4.42d}$$

Here the angle brackets $\langle \ldots \rangle$ indicate the average procedure in the frequency domain. Note the similarities and differences between this expression and the one for coherence (section 8.5 in van Drongelen, 2007). The inverse Fourier transform of K_1 in Equation (4.42d) gives k_1 for a nonlinear system with GCN input. A similar development for the second-order kernel gives us:

$$K_2(f_1,f_2) = \frac{\langle X(f_1)^* X(f_2)^* Z(f_1+f_2) \rangle}{2\langle X(f_1)^* X(f_1) \rangle \langle X(f_2)^* X(f_2) \rangle} \tag{4.43}$$

and taking the inverse Fourier transform of the above expression then gives k_2.

4.8 Summary

As we demonstrated in this chapter, the determination of the Wiener kernels can be obtained from input−output correlation. These scenarios for GWN input are depicted in Fig. 4.6, both for the case with continuous output (Fig. 4.6A and B) and for spike train outputs (Fig. 4.6C−F).

Panels (A) and (B) in Fig. 4.6 show first- and second-order correlation procedures: the multiplication of $z(t)x(t-\tau_1)$ and $z(t)x(t-\tau_1)x(t-\tau_2)$, respectively. Because the cross-correlations are determined by the integration of these products, one may envision moving the multiplications over the signal while summing

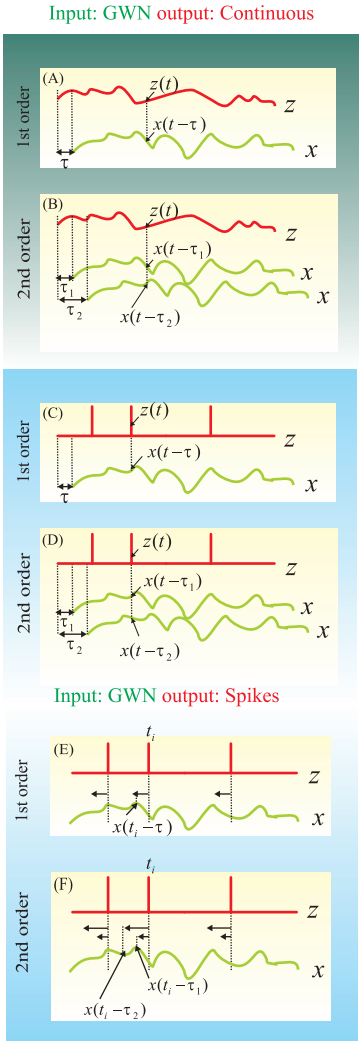

Figure 4.6 Summary diagrams of the characterization of a system with GWN input. Diagrams of the cross-correlation procedures for systems with a continuous output (A, B) or spike train output (C−F). (A), (C), and (E) show the first-order case and (B), (D), and (F) represent the second-order procedure. (C) and (E) depict two alternative visualizations for obtaining the first-order cross-correlation for systems with spiking output. In (C), the input is shifted by amount τ, whereas in (E), $x(t-\tau)$ at time $t = t_i$ is directly determined without shifting x (represented by the left-pointing arrow). For the spike output case, this procedure in (E) can be followed (as an alternative to the standard procedure in C) since the cross-correlation product is zero when there is no spike at time t_i. The analogous alternatives for determining the second-order correlation is shown in (D) and (F). See text for further explanation.

(integrating) the resulting products. The delays τ, τ_1, τ_2 can be visualized by shifting input x relative to output z (Fig. 4.6A and B).

If the system's output z is a spike train, as shown in (C) and (D), the correlations required to compute the kernels are identical: that is, the input can be shifted relative to the output to obtain $x(t-\tau)$, $x(t-\tau_1)$, and $x(t-\tau_2)$. However, this procedure can also be depicted as reverse correlations of each spike at time t_i, as shown in (E) and (F). Instead of shifting the input as we have just depicted, the reverse-correlation procedure is shown here with left-pointing arrows. Note that this is just another way of representing the shifts τ, τ_1, τ_2, and that it is not essentially different from the visualization in (C) and (D). However, the fact that we only consider the products $z(t)x(t-\tau)$ and $z(t)x(t-\tau_1)x(t-\tau_2)$ at t_i is essentially different from the case when we have a system with continuous output (as depicted in (A) and (B)) and is caused by the fact that we model the spike train with a series of Diracs. In between the spikes (i.e., in between the unit-impulse functions), the output $z(t)$ is considered zero and the products $z(t)x(t-\tau)$ and $z(t)x(t-\tau_1)x(t-\tau_2)$ vanish.

From the examples in this and the previous chapter, it may be clear that computing the kernels in the series can be a demanding task computationally. Recently, Franz and Schölkopf (2006) described an alternative method to estimate Volterra and Wiener series. Details of their approach are beyond the scope of this text, but the essence is to consider discrete systems only (which is not really a limitation if one wants to compute the series). In this case, the Volterra or Wiener series operators can be replaced by functions for which the parameters (the kernel parameters) can be estimated with regression techniques (see Section 2.4.1 for an example of a regression procedure). This approach is computationally more efficient than the Lee and Schetzen (1965) cross-correlation method (described here in Sections 4.3 and 4.4) and makes the estimation of high-order kernels feasible. An example of an application of this method to EEG is described in Barbero et al. (2009).

Appendix 4.1

Averages of Gaussian Random Variables

In this appendix we discuss averages of GWN variables because their properties are important for the development of the Wiener series approach (especially in demonstrating that the operators are orthogonal to lower-order operators). Because it is beyond the scope of this text to provide a detailed proof of all properties presented here, for further background see appendix A of Schetzen (2006). The relationship between higher- and lower-order moments, which we will discuss below, is also known as Wick's theorem (see, e.g., Zinn-Justin, 2002).

Let us consider ergodic and zero mean GWN represented by variable x: that is, the expected value of x can be replaced by its time average, which is zero (zero mean):

$$E\{x(t-\tau)\} = \langle x(t-\tau)\rangle = 0 \tag{A4.1.1}$$

The product $\langle x(t-\tau_1)x(t-\tau_2)\rangle$ is equal to the autocorrelation and also to the autocovariance (because the noise is zero mean):

$$\langle x(t-\tau_1)x(t-\tau_2)\rangle = \sigma^2\delta(\tau_1-\tau_2) \tag{A4.1.2}$$

Because this may not be immediately apparent, let us define $t-\tau_1 = T$ and $\tau_2-\tau_1 = \tau$. We can now rewrite the autocorrelation in Equation (A4.1.2) as $\langle x(T)x(T-\tau)\rangle$. In the case where $\tau = 0$ ($\tau_2 = \tau_1$), we get the expression $\langle x(T)x(T)\rangle = \langle x(T)^2\rangle = E\{x(T)^2\}$. For GWN with zero mean, this is the definition of the variance σ^2 of signal x (see section 3.2 in van Drongelen, 2007, on statistical moments). Again, since we are dealing with GWN (which gives us a random signal x), two different samples of x are uncorrelated; that is, for $\tau \neq 0$ ($\tau_2 \neq \tau_1$), $x(T)$ is not correlated with $x(T-\tau)$. This means that:

$$\langle x(T)x(T-\tau)\rangle = E\{x(T)x(T-\tau)\} = 0 \quad \text{for } \tau \neq 0.$$

Combining the above findings for $\tau_2 = \tau_1$ and $\tau_2 \neq \tau_1$ we can use the expression in Equation (A4.1.2) with the Dirac delta function. Let us look into an example in which we scale the correlation coefficient between ± 1. A scatter plot showing correlation for a GWN signal is shown in Fig. A4.1.1. The plot of the signal against itself with zero lag ($\tau = 0$) is depicted in Fig. A4.1.1A and obviously all points lie on the $y = x$ line, corresponding to a correlation coefficient of one. An example for a delay of $\tau = 1$ is shown in Fig. A4.1.1B; here the points are distributed in all directions corresponding to the absence of correlation (correlation coefficient of zero). This behavior is confirmed in a plot of the autocorrelation of GWN: we have a correlation coefficient of one for a lag τ of zero and a correlation coefficient of zero otherwise (see also Fig. 4.5A).

The findings from the paragraph above can be generalized to evaluate higher-order products between GWN signals (Schetzen, 2006). All averages of **odd** products evaluate to zero (e.g., $\langle x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)\rangle = 0$), while it can be shown

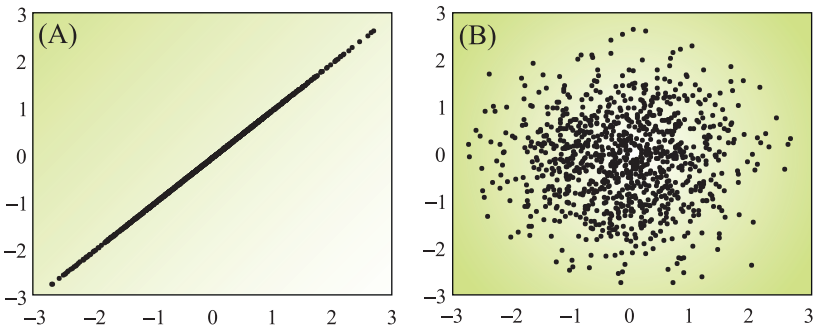

Figure A4.1.1 Correlation for $y(t)$, a digitized GWN signal of 1000 points. (A) A plot of $y(t)$ vs. $y(t)$. (B) A plot of $y(t+1)$ vs. $y(t)$.

that higher-order **even** products are equal to the sum of all distinct pair-wise products. For example:

$$\langle x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)x(t-\tau_4)\rangle = \langle x(t-\tau_1)x(t-\tau_2)\rangle\langle x(t-\tau_3)x(t-\tau_4)\rangle + \langle x(t-\tau_1)x(t-\tau_3)\rangle\langle x(t-\tau_2)x(t-\tau_4)\rangle + \langle x(t-\tau_1)x(t-\tau_4)\rangle\langle x(t-\tau_2)x(t-\tau_3)\rangle \quad \text{(A4.1.3)}$$

If you are interested in the formal proof of the above generalizations for the odd and even products, please see Appendix A in Schetzen (2006).

Appendix 4.2

Delay System as Volterra Operator

We used a specific delay operator earlier for creating the Hilbert transform in Chapter 1. Here we will comment on delay operators in general. Creation of a delay υ_1 in $x(t)$ is an operation by which we obtain $x(t-\upsilon_1)$; this operation can be considered a 1D, first-order Volterra operator (Fig. A4.2.1A). Higher-dimensional (2D and 3D) delay systems can be represented by second- and third-order Volterra systems (Fig. A4.2.1B and C), etc. The 1D operator D_1 can be characterized by the notation:

$$D_1[x(t)] = x(t-\upsilon_1) \quad \text{(A4.2.1)}$$

Because this is a first-order system, this operation can be represented by a convolution:

$$D_1[x(t)] = x(t-\upsilon_1) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)\mathrm{d}\tau \quad \text{(A4.2.2)}$$

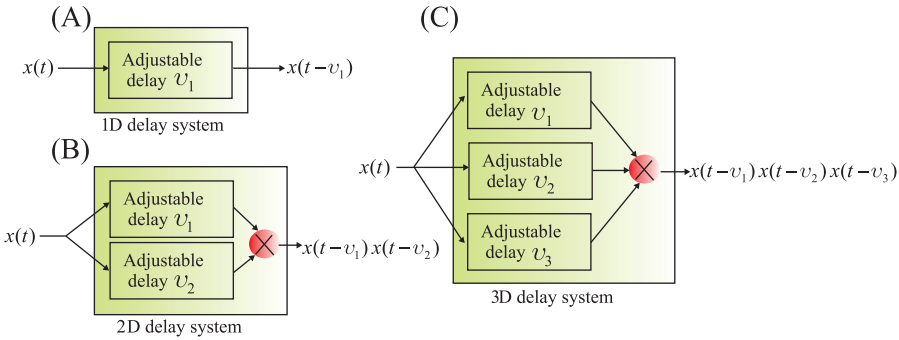

Figure A4.2.1 Examples of delay systems as Volterra operators.

From Equation (A4.2.2) we may conclude that the weighting function (the UIR) of the 1D system is $h(\tau) = \delta(\tau - \upsilon_1)$, thus resulting in:

$$x(t - \upsilon_1) = \int_{-\infty}^{\infty} \delta(\tau - \upsilon_1) x(t - \tau) \mathrm{d}\tau \tag{A4.2.3}$$

Similarly, the delay operators for 2D operator D_2 and 3D operator D_3 can be defined as $D_2[x(t)] = x(t - \upsilon_1)x(t - \upsilon_2)$ and $D_3[x(t)] = x(t - \upsilon_1)x(t - \upsilon_2)x(t - \upsilon_3)$, respectively. In Fig. A4.2.1B and C we can see that each of the delays in the higher-dimensional system is a first-order operator. In the second-order (2D) system the UIRs are $\delta(\tau - \upsilon_1)$ and $\delta(\tau - \upsilon_2)$; in the third-order delay system, the UIRs are $\delta(\tau - \upsilon_1)$, $\delta(\tau - \upsilon_2)$, and $\delta(\tau - \upsilon_3)$. Similar to Equation (A4.2.3), these operations can be represented with the convolution-like integrals of the Volterra series (see Equation (3.4)); for example, in the 2D case:

$$x(t - \upsilon_1)x(t - \upsilon_1) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \underbrace{\delta(\tau_1 - \upsilon_1)\delta(\tau_2 - \upsilon_2)}_{\text{2nd order Volterra kernel } h_2(\upsilon_1, \upsilon_2)} x(t - \tau_1)x(t - \tau_2)\mathrm{d}\tau_1 \, \mathrm{d}\tau_2 \tag{A4.2.4}$$

where the second-order Volterra kernel is:

$$h_2(\upsilon_1, \upsilon_2) = \delta(\tau_1 - \upsilon_1)\delta(\tau_2 - \upsilon_2) \tag{A4.2.5}$$

In the 3D case the third-order Volterra kernel for a delay system is:

$$h_3(\upsilon_1, \upsilon_2, \upsilon_3) = \delta(\tau_1 - \upsilon_1)\delta(\tau_2 - \upsilon_2)\delta(\tau_3 - \upsilon_3) \tag{A4.2.6}$$

Similarly, we can extend this approach to an n-dimensional delay operator:

$$h_n(\upsilon_1, \upsilon_2, \ldots, \upsilon_n) = \delta(\tau_1 - \upsilon_1)\delta(\tau_2 - \upsilon_2)\ldots\delta(\tau_n - \upsilon_n) \tag{A4.2.7}$$