Annals of Biomedical Engineering, Vol. 19, pp. 429–455, 1991
Printed in the USA. All rights reserved.

0090-6964/91 $3.00 + .00
Copyright © 1991 Pergamon Press plc

Parallel Cascade Identification and Kernel Estimation for Nonlinear Systems

Michael J. Korenberg

Department of Electrical Engineering
Queen's University
Kingston, Ontario, Canada

(Received 3/16/91)

We consider the representation and identification of nonlinear systems through the use of parallel cascades of alternating dynamic linear and static nonlinear elements. Building on the work of Palm and others, we show that any discrete-time finite-memory nonlinear system having a finite-order Volterra series representation can be exactly represented by a finite number of parallel LN cascade paths. Each LN path consists of a dynamic linear system followed by a static nonlinearity (which can be a polynomial). In particular, we provide an upper bound for the number of parallel LN paths required to represent exactly a discrete-time finite-memory Volterra functional of a given order. Next, we show how to obtain a parallel cascade representation of a nonlinear system from a single input-output record. The input is not required to be Gaussian or white, nor to have special autocorrelation properties. Next, our parallel cascade identification is applied to measure accurately the kernels of nonlinear systems (even those with lengthy memory), and to discover the significant terms to include in a nonlinear difference equation model for a system. In addition, the kernel estimation is used as a means of studying individual signals to distinguish deterministic from random behaviour, in an alternative to the use of chaotic dynamics. Finally, an alternate kernel estimation scheme is presented.

Keywords—*Parallel cascades, Kernel estimation, Nonlinear systems, Identification.*

INTRODUCTION

This article concerns the representation and identification of a nonlinear system by parallel cascades of alternating dynamic linear (L) and static nonlinear (N) elements. For present purposes, the nonlinear system is defined in discrete time, is time invariant, causal, has finite memory, and is continuous, in that small changes in the system input result in small changes in the system output.

Palm (21) has shown that any such system can be uniformly approximated, to an

Acknowledgments—I particularly wish to thank Professor Hun H. Sun and Professor Fred J. Looft for organizing the mini-symposium on physiological system identification and modeling, in Philadelphia, November 1990, out of which this article grew. This research was sponsored by the Natural Sciences and Engineering Research Council of Canada.

Address correspondence to Michael J. Korenberg, Department of Electrical Engineering, Queen's University, Kingston, Ontario, K7L 3N6, Canada.

arbitrary degree of accuracy, by a sum of a sufficient number of LNL cascades. Each LNL cascade comprises a dynamic linear system followed by a static nonlinearity which is, in turn, followed by a dynamic linear system. Palm (21) actually allowed the nonlinear system under approximation to have finite anticipation, in addition to finite memory, but in the present article we consider approximation of nonanticipatory (i.e., causal or physically realizable) nonlinear systems. Previously, Kolmogoroff (7) had shown that a finite sum of NLNL cascades could be used to represent exactly a very wide class of continuous nonlinear systems.

The capability of parallel LNL cascades to represent a wide variety of nonlinear systems is a very useful result, but Palm (21) did not describe a method for obtaining this representation. Korenberg (8,9,13) proposed a procedure for building up such a parallel cascade representation for any nonlinear system having a Wiener functional expansion. The individual cascade paths of the parallel arrangement were identified one at a time. It was shown that, for the procedure to be general, it sufficed if each cascade path comprised a dynamic linear system followed by a static nonlinearity. However, it was also pointed out that the procedure could be extended very simply by adding a second dynamic linear system following the static nonlinearity.

The parallel cascade identification method (8,9,13) does not require use of a Gaussian input. The method made it possible to construct accurate mimetic models of nonlinear systems, even those with high-order nonlinearities and lengthy memory. Indeed, the parallel cascade method was significantly faster and more accurate than representing the nonlinear system by a Wiener (23) functional expansion and estimating the kernels in the expansion by the Lee-Schetzen (16) cross-correlation technique. Moreover, once the parallel cascade representation had been constructed, it could be rearranged into a corresponding functional expansion. This enabled accurate estimation of the Wiener or the Volterra kernels in the obtained expansion. Kernel estimation via the parallel cascade method is accurate even for lengthy system memories (e.g., up to 150 lags). While the method does not require a Gaussian or white input signal, nor one having special autocorrelation properties, the kernel estimates may depend on the choice of input. Thus, a Gaussian input is used to obtain Wiener kernel estimates.

Recently, Shi and Sun (22) have considered the representation of nonlinear systems by parallel cascades. They point out that a discrete-time, finite-memory, finite-order Volterra series can be exactly represented by the sum of a finite number of LNL cascades. Earlier, Palm (21) had shown that the kernels of such a Volterra series could each be exactly expressed as a finite sum of separable functions.

In the next section we show that, in fact, the sum of a finite number of LN cascades suffices to represent exactly any discrete-time, finite-memory, finite-order Volterra series. [Such an LN cascade is frequently referred to as a Wiener model (6).] Moreover, we provide an upper bound for the number of LN cascades required to represent exactly a Volterra functional of given order. We also point out that the parallel cascade representation of a given system is not unique. In subsequent sections, we describe the parallel cascade identification approach (8,9,13) and illustrate its ability to estimate accurately the kernels of a system with lengthy memory. The method is also used to discover the significant terms to include in a difference equation model for a nonlinear system. Also, we show how the kernel estimation can be applied to individual signals to distinguish random noise from chaos produced by deterministic systems. Finally, an alternate scheme for estimating kernels is presented.

REPRESENTATION OF DISCRETE-TIME FINITE-MEMORY VOLTERRA FUNCTIONALS BY PARALLEL LN CASCADES

Consider first a *continuous-time*, time-invariant, finite-memory, causal nonlinear system which is a continuous functional of its input. It follows from the work of Frechet (3) that such a system can be uniformly approximated over a uniformly bounded equicontinuous set of input signals, to an arbitrary degree of accuracy, by a Volterra series of sufficient but finite order.

Next, for the *discrete-time* nonlinear system, the corresponding approximation result follows immediately from the Stone-Weierstrass theorem (2,21). The finite-order Volterra series which is capable of uniformly approximating the (finite-memory) nonlinear system over a uniformly bounded set of input signals has the form

$$y_s(n) = k_0 + \sum_{m=1}^{M} V_m \ , \qquad n = 0,1,\ldots \tag{1}$$

where M is the order of the series and, for $m \geq 1$, the mth order Volterra functional is

$$V_m = \sum_{i_1=0}^{R} \ldots \sum_{i_m=0}^{R} k_m(i_1,\ldots,i_m)x(n-i_1)\ldots x(n-i_m) \ . \tag{2}$$

The zero-order functional is the constant k_0, x in Eq. 2 is the system input, k_m is the mth order symmetric Volterra kernel, and $(R+1)$ is the memory length (since the series output $y_s(n)$ depends on input delays from 0 to R lags).

We will show that each Volterra functional of Eq. 2 can be exactly represented by a finite sum of LN (i.e., dynamic linear, static nonlinear) cascades. We will also find an upper bound for the number of LN cascades required to represent a given order functional.

The zero- and first-order functionals are each exactly represented by a trivial case of a single LN cascade.

To represent the second-order functional ($m = 2$ in Eq. 2), consider first the sum of two particular LN cascades. (Slices of the second-order functional will be constructed from these cascade pairs.) The first system in each of the two cascades is linear and has the delta (discrete impulse) response

$$h_i(j) = k_2(j,A) + (-1)^i C\delta(j-A) \ , \qquad i = 1, 2 \ . \tag{3}$$

Here A is a constant selected from $0,\ldots,R$, $C \neq 0$ is a real constant, and the delta function $\delta(j) = 0, j \neq 0$, and $\delta(0) = 1$. The corresponding outputs of the linear systems in the two cascades are

$$u_i(n) = \sum_{j=0}^{R} h_i(j)x(n-j) \tag{4}$$

where $i = 1, 2$.

The second system in each cascade is a (static nonlinear) squarer defined by

$$\begin{aligned} z_i &= F_i(u_i) \\ &= (-1)^i u_i^2/(4C) \ , \qquad i = 1, 2 \ . \end{aligned} \tag{5}$$

It follows from Eqs. 3 through 5 that the sum of the two cascades has output

$$\begin{aligned} z(n) &= z_1(n) + z_2(n) \\ &= \sum_{j=0}^{R} k_2(j,A)x(n-j)x(n-A) \ . \end{aligned} \tag{6}$$

Suppose each *pair* of cascades is defined using a different value of A (in Eq. 3) from $0,\ldots,R$. Then the sum of the $(R+1)$ cascade pairs (corresponding to $A = 0,\ldots,R$) will exactly represent the second-order Volterra functional, so that at most $2(R+1)$ cascades of LN type are required. Note that this parallel array of LN cascades is not a unique representation of the second-order functional: for example, any real $C \neq 0$ can be used in Eqs. 3 and 5 in defining each cascade pair. In addition, notice that we have shown equivalently that the second-order kernel ($m = 2$ in Eq. 2) can be exactly expressed as

$$k_2(i_1,i_2) = \sum_{l=1}^{2(R+1)} (-1)^l f_l(i_1)f_l(i_2)$$

which is a little stronger than the separability result of Palm (21).

To represent exactly the third-order Volterra functional ($m = 3$ in Eq. 2), consider first the sum of four LN cascades. The first (linear) system in each of the cascades has delta response respectively:

$$h_1(j) = k_3(j,A_1,A_2) - C_1\delta(j-A_1) - C_2\delta(j-A_2) \tag{7}$$

$$h_2(j) = k_3(j,A_1,A_2) + C_1\delta(j-A_1) - C_2\delta(j-A_2) \tag{8}$$

$$h_3(j) = k_3(j,A_1,A_2) - C_1\delta(j-A_1) + C_2\delta(j-A_2) \tag{9}$$

$$h_4(j) = k_3(j,A_1,A_2) + C_1\delta(j-A_1) + C_2\delta(j-A_2) \ . \tag{10}$$

Here A_1 and A_2 are constants chosen from $0,\ldots,R$; $C_1 \neq 0$, $C_2 \neq 0$ are real constants.

Let $u_i(n)$ (Eq. 4), $i = 1,\ldots,4$, be the corresponding outputs of the linear systems in the cascades. The second (static nonlinear) system in the cascades is defined respectively by

$$z_1 = u_1^3/(EC_1C_2) \tag{11}$$

$$z_2 = -u_2^3/(EC_1C_2) \tag{12}$$

$$z_3 = -u_3^3/(EC_1C_2) \tag{13}$$

$$z_4 = u_4^3/(EC_1C_2) \ . \tag{14}$$

In Eqs. 11 through 14, $E = 12$ if $A_1 \neq A_2$ and $E = 24$ if $A_1 = A_2$.

It follows from Eqs. 4 and 7 through 14 that the sum of the four LN cascades has output

$$\begin{aligned} z(n) &= \sum_{i=1}^{4} z_i(n) \\ &= 2\sum_{j=0}^{R} k_3(j,A_1,A_2)x(n-j)x(n-A_1)x(n-A_2) \ , \qquad A_1 \neq A_2 \\ &= \sum_{j=0}^{R} k_3(j,A,A)x(n-j)x^2(n-A) \ , \qquad A_1 = A_2 = A \ . \end{aligned}$$

In general, in defining a set of *four* LN cascades, select A_1 from $0,\ldots,R$ and A_2 from $A_1,\ldots,R$, for a total of $(R+1)(R+2)/2$ sets. The sum over these sets (of four cascades each) will represent exactly the third-order Volterra functional, so that at most $2(R+1)(R+2)$ cascades of LN type are required. Clearly, since $C_1 \neq 0$ and $C_2 \neq 0$ are arbitrary real constants, the parallel LN cascade representation of the third-order functional is not unique. Notice we have shown that the third-order kernel ($m = 3$ in Eq. 2) can be exactly expressed as

$$k_3(i_1,i_2,i_3) = \sum_{l=1}^{2(R+1)(R+2)} g_l(i_1)g_l(i_2)g_l(i_3) \ .$$

Fourth- and higher-order functionals can be represented exactly by parallel LN cascades in an analogous fashion. For the mth order functional ($m \geq 1$) in Eq. 2, it requires at most

$$2^{m-1} \cdot \binom{R+m-1}{m-1}$$

of such cascades for exact representation.

Note that in the representation of a discrete-time, finite-memory, finite-order Volterra series by parallel cascades, an arbitrary number of the cascade paths can be chosen at will. For example, in each of the first 10 parallel paths, one may choose any finite number of arbitrarily selected finite-memory linear systems and (polynomial) static nonlinearities. The difference between the original Volterra series and the 10 parallel paths is itself a discrete-time, finite-memory, finite-order Volterra series, and so can be represented exactly by a parallel array of LN cascades. This parallel array plus the 10 parallel paths will exactly represent the original Volterra series. This further illustrates that the parallel cascade representation of the Volterra series is not unique.

Recall that, by the Stone-Weierstrass theorem, a discrete-time, finite-memory Volterra series of sufficient order can (over a uniformly bounded set of input signals) uniformly approximate to an arbitrary accuracy any time-invariant, causal, finite-memory, discrete-time nonlinear system which is a continuous functional of its input. Hence, such a nonlinear system can also be approximated to an arbitrary degree of accuracy by a sufficient number of LN cascades in parallel. One method for constructing such a parallel array (8,9,13) is described in the next section.

PARALLEL CASCADE IDENTIFICATION

Suppose that the nonlinear system to be identified has input $x(n)$ and output $y(n)$, $n = 0, \dots, T$. We assume that the input x is sufficiently "rich" to make it possible to solve for least-square estimates of the kernels k_m in Eqs. 1, 2; that is, unique values for the kernels which minimize the mean-square error of approximating $y(n)$ by the Mth order series of Eq. 1. However, we do not solve for such kernel estimates directly. Instead, we construct a parallel cascade approximation for the given nonlinear system. Assume that the system output depends on input delays from 0 to R (e.g., as in Eqs. 1, 2) so that $(R + 1)$ is the memory length.

Parallel cascade identification (8,9,13) begins by approximating the nonlinear system with a first cascade of alternating dynamic linear and static nonlinear systems. The unidentified residue (i.e., the difference between the nonlinear system output and the cascade output) is treated as the output of a new nonlinear system. Another cascade is identified to approximate the new nonlinear system, the resulting residue is computed, and so on. Assume that the original nonlinear system has a Wiener (23) functional expansion. This displays convergence in the mean-square sense, a weaker convergence than the uniform convergence discussed earlier for continuous functionals. Then we will show that the nonlinear system can be approximated to an arbitrary degree of accuracy in the mean-square sense by a sum of a sufficient number of the individually identified cascades.

The estimation of a cascade may be outlined as follows. Begin the cascade with a linear system whose delta response is defined using a first- or higher-order cross-correlation of the input with the residue. Thus, the delta response is set equal either to the first-order cross-correlation or to a slice of a higher-order cross-correlation to which delta functions are added or subtracted at diagonal points (as in Eqs. 3 and 7 through 10). After defining the linear system, its output is computed (as in Eq. 4), and then a static nonlinearity (e.g., a polynomial) is best-fit to the residue. The procedure may be continued by adding a second linear system (8) following the static nonlinearity. For example, in implementing the parallel cascade identification (8), Mo and Elkasabgy (20) followed the static nonlinearity with a linear differential equation best-fit to the residue. Once a cascade has been estimated, the new residue is calculated, and then a further cascade can be estimated analogously. The procedure will be illustrated next when each cascade consists of a dynamic linear system followed by a static nonlinearity. However, it will be appreciated that any number of alternating dynamic linear and static nonlinear systems can be added to a cascade path (8,9,13).

Outline of One Version of Algorithm

Let $y_i(n)$ be the residue remaining after adding the ith cascade path to the parallel cascade model. Thus, $y_0(n) = y(n)$. Let $z_i(n)$ be the output of the ith cascade. Hence,

$$y_i(n) = y_{i-1}(n) - z_i(n) \ , \quad i = 1, 2, \ldots \ . \tag{15}$$

Consider fitting the ith cascade ($i \geq 1$) to the residue $y_{i-1}(n)$. Begin by defining the delta response $h_i(j)$ of the dynamic linear system in the cascade. Let $\phi_{xy_{i-1}}$ and $\phi_{xxy_{i-1}}$ denote the first- and second-order cross-correlations of the input with the residue, computed over the portion of the recording extending from $n = R$ to $n = T$:

$$\phi_{xy_{i-1}}(j) \triangleq \frac{1}{T - R + 1} \sum_{n=R}^{T} y_{i-1}(n)x(n-j) \tag{16}$$

$$\phi_{xxy_{i-1}}(j_1, j_2) \triangleq \frac{1}{T - R + 1} \sum_{n=R}^{T} y_{i-1}(n)x(n-j_1)x(n-j_2) \ . \tag{17}$$

Set $h_i(j)$ to be one of

$$p_1(j) = \phi_{xy_{i-1}}(j) \tag{18}$$

$$p_2(j) = \phi_{xxy_{i-1}}(j, A) \pm C\delta(j - A) \ . \tag{19}$$

Whether Eq. 18 or Eq. 19 will be used to define $h_i(j)$ is decided at random. If Eq. 19 is used, then the integer A is randomly selected from $0, \ldots, R$, the sign of the δ term is chosen at random, and C is adjusted to tend to zero as the mean-square of the residue approaches zero. For example, set

$$C = \frac{\overline{y_{i-1}^2(n)}}{\overline{y^2(n)}} \tag{20}$$

where (here and elsewhere), the overbar denotes time-average over the portion of the record extending from $n = R$ to $n = T$ (as in Eqs. 16, 17). Since the nonlinear system to be identified is assumed to have finite memory lasting up to R lags, therefore $h_i(j) = 0, j > R$. Note that, in addition to Eqs. 18 and 19, slices of third- or higher-order cross-correlations may be used to define $h_i(j)$ with δ-functions added or subtracted at diagonal values, analogous to Eqs. 7 through 10. This is discussed later.

Once the delta response $h_i(j)$ of the linear system has been determined, calculate its output $u_i(n)$ via Eq. 4. To obtain the static nonlinearity, best-fit a polynomial having input $u_i(n)$ to the residue $y_{i-1}(n)$ over $n = R, \ldots, T$. The cascade output is then

$$z_i(n) = F_i(u_i) = \sum_{l=0}^{M} a_{il} u_i^l \tag{21}$$

and the new residue $y_i(n)$ is given by Eq. 15.

Hence, the polynomial coefficients a_{il} minimize the mean-square of the new residue over $n = R, \ldots, T$, and therefore

$$\begin{aligned}\overline{y_i^2(n)} &= \overline{(y_{i-1}(n) - z_i(n))^2} \\ &= \overline{y_{i-1}^2(n)} - \overline{z_i^2(n)} \ .\end{aligned} \tag{22}$$

Thus, the reduction in mean-square error (mse) be adding the ith cascade equals the mean-square of the output of the ith cascade.

We can then repeat the procedure to fit the residue $y_i(n)$ by an $(i + 1)$th cascade, having output $z_{i+1}(n)$. Note that for $i = 1,2, \ldots$

$$r = (\overline{z_{i+1}^2(n)}/\overline{y_i^2(n)})^{1/2} \tag{23}$$

is the correlation of $z_{i+1}(n)$ and $y_i(n)$ over $n = R, \ldots, T$. If the residue were independent zero-mean Gaussian noise, then for sufficiently large T

$$|r| < 2/\sqrt{T - R + 1} \tag{24}$$

with probability of about 0.95. Hence, before accepting a given candidate for the $(i + 1)$th cascade, one may *optionally* require that

$$\overline{z_{i+1}^2(n)} > \frac{4}{T - R + 1} \overline{y_i^2(n)} \ . \tag{25}$$

This requirement (a standard correlation test) helps to avoid choosing unnecessary cascades which are merely fitting noise. Note that $4/(T - R + 1)$ on the right side of Eq. 25 can be replaced by other factors (corresponding to different confidence intervals), or by the square of the critical correlation coefficient value for sample size $T - R + 1$.

If the test is used and a candidate fails to satisfy the immediately above inequality, then we may construct and test a new candidate for the $(i + 1)$th cascade. Begin by redefining the delta response $h_{i+1}(j)$ of the linear system in the cascade. This may be done (analogous to the right sides of Eqs. 18, 19) via a first-order cross-correlation of the input with the residue y_i, or using a slice of a second- or higher-order cross-correlation with δ-functions added or subtracted at diagonal values. Then the static nonlinearity (e.g., polynomial) is estimated by best-fitting the residue, and the new candidate cascade may be tested for compliance with the inequality shown in Eq. 25.

Parallel cascade development may be stopped when a specified number of cascades have been added or tested, or when the mse has been made sufficiently small. Termination may also occur when no remaining candidate cascade can cause a reduction in mse exceeding a small threshold level.

Outline of Proof of Convergence of Algorithm

In obtaining the ith cascade, the static nonlinearity represented by Eq. 21 is best-fit to the residue y_{i-1}. Since the polynomial's constant term a_{i0} is a least-square estimate, it follows that

$$\bar{z}_i = \bar{y}_{i-1}\ , \qquad i \geq 1$$

$$\bar{z}_i = \bar{y}_{i-1} = 0\ , \qquad i \geq 2\ ,$$

in view of Eq. 15.

Recall from Eq. 22 that each cascade added to the parallel array reduces the mse by an amount equalling the mean-square of that cascade output. Clearly, the mse of the parallel array approximation can be reduced at most to zero. Hence, if the parallel cascade development is continued sufficiently long, eventually no remaining candidate cascade can reduce the mean-square of the residue more than a negligible amount. Suppose that when this occurs, the existing residue is $y_{i-1}(n)$. Thus, i is sufficiently large (i.e., a sufficient number of parallel cascade paths have already been added), so no matter which candidate is considered for the ith cascade, the mean-square $(\overline{z_i^2(n)})$ of that cascade's output is negligible.

Suppose that the first-order cross-correlation on the right side of Eq. 18 is used to define $h_i(j)$ (for the linear system in the cascade). As just noted, the corresponding $\overline{z_i^2(n)}$ can be made arbitrarily small by choosing i sufficiently large. We will now show that this implies that the first-order cross-correlation of the input with the residue can itself be made arbitrarily small.

First, note that if the system to be identified has order (i.e., degree) of nonlinearity of at least 2, then M in Eq. 21 will be chosen to be greater than or equal to 2. Thus, the mse reduction, which equals $\overline{z_i^2(n)}$, cannot be less than the reduction in mse if only the linear term $a_{i1}u_i$ were best-fit to the residue y_{i-1}. Least-square fitting the linear term alone would cause a reduction in mse equalling

$$\text{Reduction (linear)} = \frac{(\overline{y_{i-1}(n)u_i(n)})^2}{\overline{u_i^2(n)}} \tag{26}$$

which must be less than or equal to $\overline{z_i^2(n)}$. Since the first-order cross-correlation on the right side of Eq. 18 is used to define $h_i(j)$, it follows from Eq. 4 and the Schwarz inequality that

$$\overline{u_i^2(n)} \leq \overline{y^2(n)} \left(\sum_{j=0}^{R} \overline{x^2(n-j)} \right)^2 .$$

Again, from Eqs. 4 and 18,

$$\overline{y_{i-1}(n)u_i(n)} = \sum_{j=0}^{R} \phi^2_{xy_{i-1}}(j)\ .$$

Hence, it follows from Eq. 26 that

$$\overline{z_i^2(n)} \geq \frac{\left(\sum_{j=0}^{R} \phi_{xy_{i-1}}^2(j)\right)^2}{\overline{y^2(n)}\left(\sum_{j=0}^{R} \overline{x^2(n-j)}\right)^2}$$

and since $\overline{z_i^2(n)}$ can be made arbitrarily small, then so can $|\phi_{xy_{i-1}}(j)|$ for $j = 0,\ldots,R$.

Next, suppose that the right side of Eq. 19 (which employs a slice of a second-order cross-correlation) is used to define $h_i(j)$ for the ith cascade. Again, the resulting mse reduction, equalling $\overline{z_i^2(n)}$, is negligible, since by assumption we have reached the stage where no further cascade addition can reduce the mse significantly. Thus, whether the δ-term is added or subtracted on the right side of Eq. 19, and for each value of A chosen therein, the corresponding $\overline{z_i^2(n)}$ for the cascade is negligible. Recall that this reduction in mse results from best-fitting $z_i(n)$ (defined by Eq. 21 with $M \geq 2$) to the residue. Such a reduction cannot be less than the following mse reduction, which results from best-fitting only the quadratic term $a_{i2}u_i^2$ to the residue:

$$\text{Reduction (quadratic)} = \frac{(\overline{y_{i-1}(n)u_i^2(n)})^2}{\overline{u_i^4(n)}} \ . \tag{27}$$

Thus, the right side of Eq. 27 must be less than or equal to $\overline{z_i^2(n)}$, which can be made arbitrarily small. Using Eq. 4 and the Schwarz inequality, it is easy to show that the denominator $\overline{u_i^4(n)}$ in Eq. 27 is less than a bound which depends only on x and y. This implies that the numerator on the right side of Eq. 27 can be made arbitrarily small. Now, from Eq. 4,

$$\begin{aligned}\overline{y_{i-1}(n)u_i^2(n)} &= \sum_{j_1=0}^{R}\sum_{j_2=0}^{R} \phi_{xxy_{i-1}}(j_1,A)\phi_{xxy_{i-1}}(j_2,A)\phi_{xxy_{i-1}}(j_1,j_2) \\ &+ C^2\phi_{xxy_{i-1}}(A,A) \pm 2C\sum_{j=0}^{R}\phi_{xxy_{i-1}}^2(j,A)\end{aligned} \tag{28}$$

where the sign of the last term on the right side matches the sign of the δ-term in Eq. 19. Since the magnitude of the right side of Eq. 28 can be made arbitrarily small for either sign of the last term, therefore

$$\sum_{j=0}^{R}\phi_{xxy_{i-1}}^2(j,A)$$

can be made arbitrarily small. Since this is true for each value of A, therefore $|\phi_{xxy_{i-1}}(j,A)|$ can be made arbitrarily small for $j = 0,\ldots,R$ and $A = 0,\ldots,R$.

Next, suppose that $h_i(j)$ is defined using a slice of a third-order cross-correlation of the input with the residue; that is, set $h_i(j)$ equal to

$$p_3(j) = \phi_{xxxy_{i-1}}(j,A_1,A_2) \pm C_1\delta(j-A_1) \pm C_2\delta(j-A_2) \ . \tag{29}$$

In Eq. 29, A_1 is randomly selected from $0, \ldots, R$ and A_2 from $A_1, \ldots, R$, and the sign of each δ-term is chosen at random. The real constants $C_1 \neq 0$ and $C_2 \neq 0$ each tend to zero as the mean-square of the residue approaches zero (for example, as in Eq. 20). Notice the similarity between Eq. 29 and Eqs. 7 through 10.

As noted, the mse reduction, equalling $\overline{z_i^2(n)}$, can be made arbitrarily small (for sufficiently large i). This is true irrespective of the signs of the δ-terms in Eq. 29, and the values chosen for A_1, A_2, in defining $h_i(j)$. Assume that in defining the static nonlinearity via Eq. 21, we choose the polynomial degree $M \geq 3$. Then it may be shown that, for sufficiently large i, $|\phi_{xxxy_{i-1}}(j, A_1, A_2)|$ can be made arbitrarily small for $j = 0, \ldots, R$; $A_1 = 0, \ldots, R$; and $A_2 = 0, \ldots, R$.

In summary, the delta response $h_i(j)$ of the linear system in the cascade can be defined using the right side of Eqs. 18, 19, 29 (and analogous expressions involving slices of higher-order cross-correlations) chosen at random. If the parallel cascade development is continued sufficiently long, we will eventually achieve a residue $y_{i-1}(n)$ whose cross-correlations (with the input) are arbitrarily small, up to order M, the polynomial degree in Eq. 21.

Assume we have noise-free conditions and that the nonlinear system to be identified can be very accurately approximated, in the mean-square sense, by a Volterra series of order not exceeding M. Then the residue y_{i-1} can itself be very accurately approximated by a Volterra series of order M. Hence, the fact that $\bar{y}_{i-1} = 0$ and that the cross-correlations $\phi_{xy_{i-1}}$, $\phi_{xxy_{i-1}}, \ldots$ (up to Mth order) can be made arbitrarily small for sufficiently large i implies that the mean-square of the residue $y_{i-1}(n)$ can be made arbitrarily small for sufficiently large i. Hence, the sum of the individually obtained cascades can approximate the given nonlinear system to an arbitrary degree of accuracy, in the mean-square sense. Note that this result does not require that the input x to the nonlinear system be Gaussian or white, nor have special probability density or autocorrelation properties.

Further Details of the Algorithm

Kernel Estimation. Suppose that the parallel array has been developed until no further cascade can reduce the mse more than a negligible amount. (The mse itself may not be negligible due to noise contamination or because the polynomial degree M in Eq. 21 was not chosen sufficiently large.) The identified parallel cascade array can be rearranged into a corresponding Volterra series of order M. This particular Volterra series will approximate the nonlinear system with very nearly the minimum mean-square error out of all series of order M and memory length $R + 1$. (This is because the residue mean is zero and the input-residue cross-correlations up to order M have been made very nearly zero.) Thus, the kernels in the obtained Volterra series are very nearly least-square estimates.

Suppose the nonlinear system to be identified has a Volterra series representation (Eq. 1) of order M and memory length $R + 1$. Then in the absence of noise contamination, the kernels obtained via parallel cascade will be very close to the actual Volterra kernels of the system. Note that there will be one value estimated for the zero-order kernel k_0, $(R + 1)$ values for the first-order kernel k_1, $(R + 1)(R + 2)/2$ distinct values for the second-order kernel k_2, and so on. The parallel cascade method (8,9,13) is particularly suited to estimating kernels of systems with lengthy memory. For example, if the nonlinear system had a second-order Volterra series

($M = 2$ in Eq. 1) with memory length ($R + 1$) equal to 150, then the 11,476 distinct kernel values could readily be estimated via the parallel cascade method. (See an example following for a memory length of 100.) However, attempting to obtain the kernel values by direct least-squares estimation would instead require the numerically formidable inversion of a 11,476 × 11,476 symmetric matrix, which is not Toeplitz or near Toeplitz to enable rapid inversion. The advantage of the parallel cascade approach becomes even more pronounced in estimating third- and higher-order kernels.

Suppose that the input x is a zero-mean white Gaussian process, and that an Mth order Volterra series approximation is obtained via parallel cascade. Then the Mth and ($M - 1$)th order kernel estimates obtained will closely approximate the Wiener kernels of corresponding order. These estimates tend to be significantly more accurate than those found by the cross-correlation method (16), when the record length is relatively short.

Modeling High-Order Nonlinear Systems. Aside from its use in kernel estimation, parallel cascade identification is convenient for modeling systems with high-order nonlinearities. This is because approximating a higher-order nonlinear system via a parallel cascade merely requires increasing the degree of the (polynomial) static nonlinearities in the cascade paths, and/or the number of static nonlinearities in a given path. The key advantage is that the nonlinearities in the parallel cascade array always occur as static functions. Hence, their estimation is far faster than computing the higher-order cross-correlations required to approximate higher-order nonlinear systems via the Wiener-Lee-Schetzen functional expansion approach. Note that while the static nonlinearities in the cascade paths have been represented by polynomials above, other sets of basis functions, such as gate functions, could equally well be used.

Recall that each cascade began with a linear system whose delta response was defined above using a slice of a cross-correlation (first- or higher-order) of the input with the residue. First, notice that when a higher-order cross-correlation is used, only a single slice need be computed, rather than the entire cross-correlation as in the Lee-Schetzen (16) approach. Second, note that slices of cross-correlations of first-, second-, and possibly third-order may sometimes suffice (in defining the linear systems in the cascade paths) even when the system to be identified has much higher-order nonlinearities. This is because the static nonlinearity in the path allows the cascade to approximate higher-order nonlinear components of the residue even if a lower-order cross-correlation was used in defining the linear system preceding the static nonlinearity.

More on Developing a Cascade Path. In place of slices of input-residue cross-correlations, one may define the linear system at the beginning of a cascade using a set of basis functions such as the Laguerre functions, sinusoids, exponentially decaying sinusoids, and exponentials. Thus, the delta response of the linear system can randomly be set equal either to one of the basis functions or to sums and/or differences of two or more of the functions.

Whichever way the linear system (in, say, the ith cascade) is defined, its output u_i is next calculated. Then least-square procedures are used to fit a static nonlinearity (having input u_i) to the residue (8). Previously, Korenberg (8) pointed out that this parallel cascade identification could trivially be extended by adding a second linear system following the static nonlinearity in the cascade under development. Mo and

Elkasabgy (20) noted that this significantly accelerated the rate of convergence in simulated examples. In their simulations, least-square procedures were used to fit a differential equation (to the residue) for the second linear system. The process of adding static nonlinear and dynamic linear elements to the cascade may be continued indefinitely (9,13), and of course each linear system can be defined in terms of its delta response or as a difference equation. Note that once the first linear system has been determined, the same approach is used each time thereafter to add a new element to the cascade under construction. Namely, the output, say v_i, of the last block in the cascade is calculated, then least-square procedures are used to fit the new element (having input v_i) to the residue.

Alternately, instead of adding the additional systems one at a time to the cascade, one may adapt (9) a relaxation technique (1) to parallel cascade identification. Thus, suppose only the linear system at the beginning of the cascade has been determined. We may then add a static nonlinearity followed by a second linear system, and iteratively update one based on the current estimate of the other. For example, best-fit the static nonlinearity, with the delta response of the second linear system initially set equal to a δ-function. Then best-fit the second linear system based on the current estimate of the static nonlinearity. Next, update the estimate of the static nonlinearity based on the latest estimate of the second linear system, and so on in an alternating procedure to hone the estimates of the two systems. A variation of this (6) can be used to fit the initial pair of linear and static nonlinear systems in any path.

Alternately, a nonlinear mean-square-error minimization technique can be used to fit the current residue with the "best" LN or LNL model (8) in the path under construction. Then the new residue is computed and the minimization technique is used to fit another LN or LNL cascade, etc.

Multiinput Multioutput System Identification. The parallel cascade identification (8,9,13) can readily be extended to model multiinput multioutput systems. Suppose that the nonlinear system to be identified has inputs $x_1(n)$, $x_2(n)$ and output $y(n)$. Then, when a new path is to be added to the parallel array, one of x_1,x_2 is chosen at random to be the input to the cascade. The cascade may then be constructed as set out earlier. In addition, we can introduce cross-products of x_1,x_2 into the parallel array. For example, for the ith cascade, we may define the first linear system using a slice of a cross-correlation of the residue y_{i-1} with both x_1 and x_2. Thus, let

$$\phi_{x_1x_2y_{i-1}}(j_1,j_2) \triangleq \frac{1}{T-R+1}\sum_{n=R}^{T} y_{i-1}(n)x_1(n-j_1)x_2(n-j_2) \ . \tag{30}$$

If x_1 is to be the input to the ith cascade, set the delta response $h_i(j)$ of the first linear system to equal

$$p_4(j) = \phi_{x_1x_2y_{i-1}}(j,A) \tag{31}$$

where A is chosen at random from $0,\ldots,R$. Next, calculate the output u_i of the linear system using Eq. 4 with $x = x_1$. Then let

$$w_i(n) = u_i(n) \pm Cx_2(n-A) \tag{32}$$

where the sign is chosen at random, and C is adjusted to tend to zero as the mean-square of the residue approaches zero (for example, as in Eq. 20). Next, we can best-fit a static nonlinearity (having input w_i) to the residue, and subsequently proceed as for any other cascade.

Whether Eq. 18, 19, 29, or 31 will be used to define $h_i(j)$ is decided at random. In addition, $h_i(j)$ can be defined using a slice of a higher-order cross-correlation (than in Eq. 30) of the residue with both x_1 and x_2.

The identified parallel array can be rearranged to obtain estimates of the self- and cross-kernels for the nonlinear system.

Difference Equation Development via Parallel Cascade. This procedure is particularly useful for discovering the significant terms to include in a difference equation model for a nonlinear system with unknown structure. For example, if x and y are respectively the single input and output, we can construct a parallel array having two inputs $x_1(n) = x(n)$, $x_2(n) = y(n-1)$, and one output $y(n)$. Once the parallel cascade array has been identified (and rearranged into a functional expansion format), the resulting self- and cross-kernels reveal the significant terms for a difference equation model. (See an example following.) This exploits the ability of parallel cascade identification to handle lengthy memories (which here means many lagged values of both x and y) and effectively perform a global search for significant terms.

Once the significant terms (or a much reduced set of candidate terms) have been earmarked by this process, their coefficients should be determined by other procedures; see, for example, (11,12). This is because the parallel cascade development is conveniently continued just long enough for the resulting kernel shapes to reveal the significant terms, but not the precise coefficient values.

This procedure can clearly also be used to discover the significant terms to include in multiinput, multioutput difference equation models.

Distinguishing Chaos from Random Noise. A great deal of attention (4,17) has been paid to the problem of distinguishing between chaotic behaviour due to a deterministic system and noise due to random processes. Parallel cascade identification and kernel estimation can provide a useful approach (10) to this problem. Suppose $y(n)$, $n = 0, \ldots, T$ is given time-series data and we wish to determine whether $y(n)$ is deterministic chaos or random noise. One approach to doing this is to treat a delayed version of y as the system input, and the original (undelayed) signal y as the system output; then identify a parallel cascade approximation for the created system. This exploits the ability of parallel cascade identification to model readily high-order nonlinear systems, which may be necessary to "copy" the created system, particularly if a lengthy delay is used in creating the system input.

Suppose that the correlation test of Eq. 25 is used as a requirement for accepting a candidate cascade into the parallel array. If the signal $y(n)$ is chaos, then the identified parallel cascade array will have a smaller mse (as a percentage of the variance of y), and more cascade paths will be chosen than if y were independent noise. Indeed, if y were merely independent noise, the expected performance (for the number of candidate cascades tested) can readily be calculated and compared with the actual result of the parallel cascade identification.

In addition, kernel estimates can be obtained from the identified parallel array. If y is due to a deterministic system, the obtained kernels characterize the signal and may detect subtle changes in it (10). For example, we may wish to study changes in a monitored physiological variable due to disease or aging, or due to an administered drug or other treatment. (Note that the kernels obtained will of course depend [10] on the amount of delay used in creating the input signal.) If y were independent noise, the kernel estimates would tend to be very noisy and negligibly small.

Finally, note that both the identified parallel cascade and the kernel estimates can be utilized to predict future values for a chaotic signal.

EXAMPLES

Kernel Estimation

The system to be identified was a second-order Volterra series ($M = 2$ in Eq. 1) with a memory length ($R + 1$) equal to 100. Hence one zeroth-, 100 first-, and 5050 distinct second-order kernel values were to be estimated, via the cross-correlation (16) and parallel cascade (8,9,13) methods. A second-order series was chosen because then the corresponding Volterra and Wiener kernels of first and second order are equal. A zero-mean Gaussian white-noise input was used to generate 10,000 input-output data pairs.

The identified parallel cascade model had an mse of 0.053%. The actual first-order kernel and the close parallel cascade estimate are shown in Fig. 1. The second-order kernel (Fig. 2) was also estimated accurately (Fig. 3) by the parallel cascade method. (The parallel cascade kernel estimates can be made even more accurate simply by adding more cascades to the parallel array.) The cross-correlation first-order (Fig. 4) and second-order (Fig. 5) kernel estimates are significantly less precise. Cross-correlation estimates do approach the actual kernel values with increasing record length. However, even with a record of 100,000 data pairs, cross-correlation estimates do not attain the accuracy of the parallel cascade results in Figs. 1 and 3. Parallel cascade estimation of kernels is robust; see (13) for performance when noise contaminates the system output.

Note that attempting to solve for the kernel values by direct least-squares estimation would here entail inversion of a 5151×5151 symmetric matrix, which is neither Toeplitz nor near Toeplitz.

Determining the Significant Terms in a Difference Equation Model

The test system was a nonlinear difference equation which had been identified by McIlroy (19) to model a simulated communications channel. By searching through a set of candidates, he found (in the order shown) 15 significant terms, including a constant, and x, y, xx, yy (but no xy) terms:

$$\begin{aligned}y(n) = &-0.636507 + 1.08717x(n-24) + 0.914799x(n-23)\\ &- 0.698645x(n-27) + 0.967739x(n-25) + 0.787682x(n-22)\\ &- 0.630834x(n-28) + 0.160929x(n-30) - 0.209741x(n-32)\\ &+ 0.397268x(n-21) - 0.117377x(n-29) - 0.49215y(n-2)\\ &+ 0.114922y(n-5) + 0.129476x(n-24)x(n-23) + 0.211136y^2(n-1)\ .\end{aligned}$$

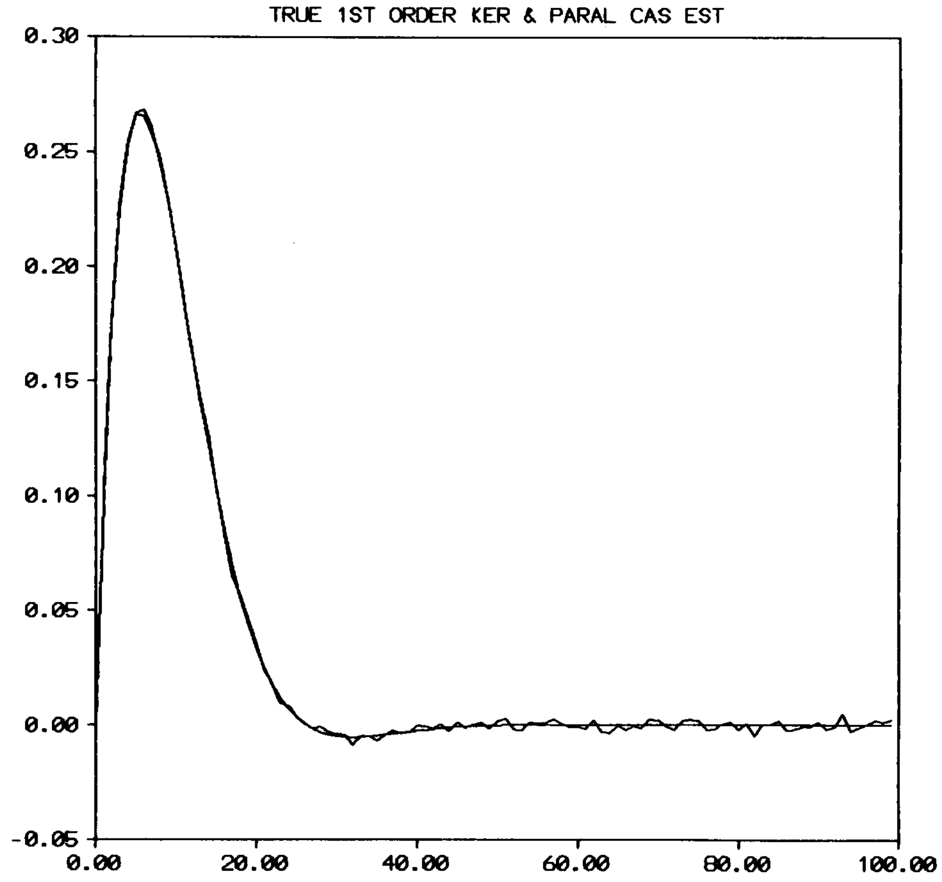

FIGURE 1. Actual first-order Wiener kernel and parallel cascade estimate, obtained using 10,000 points of white Gaussian input.

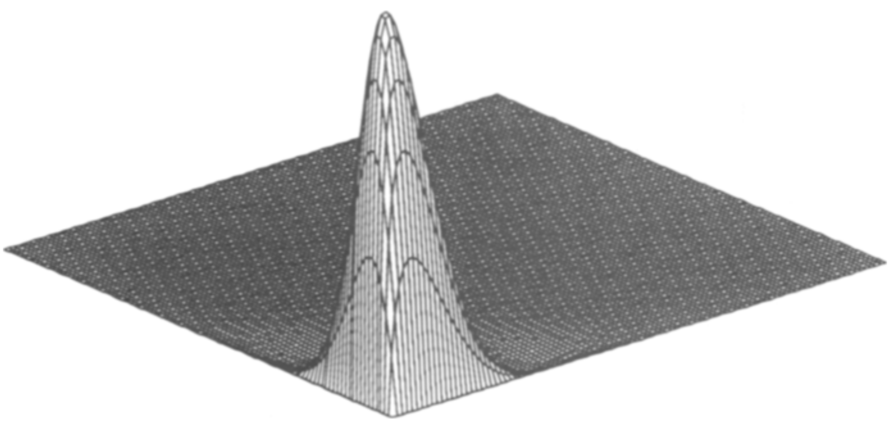

FIGURE 2. Actual second-order Wiener kernel.

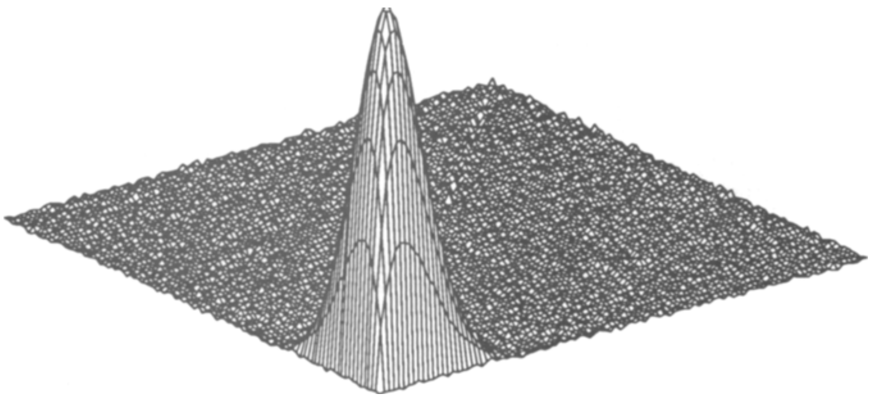

FIGURE 3. Parallel cascade estimate of second-order Wiener kernel, obtained using 10,000 points of white Gaussian input.

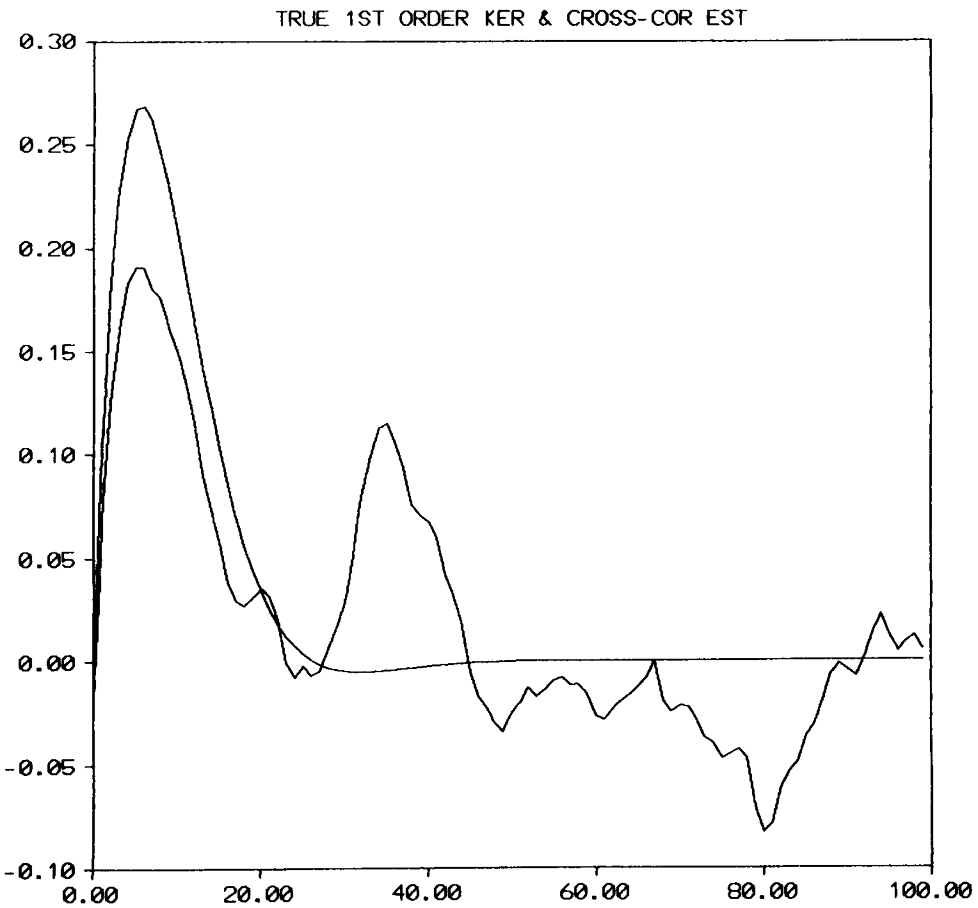

FIGURE 4. Actual first-order Wiener kernel and cross-correlation estimate, obtained using 10,000 points of white Gaussian input.

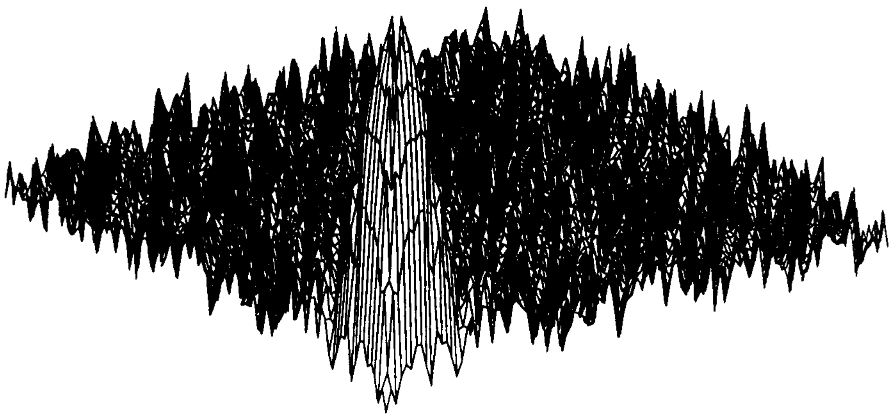
FIGURE 5. Cross-correlation estimate of second-order Wiener kernel, obtained using 10,000 points of white Gaussian input.

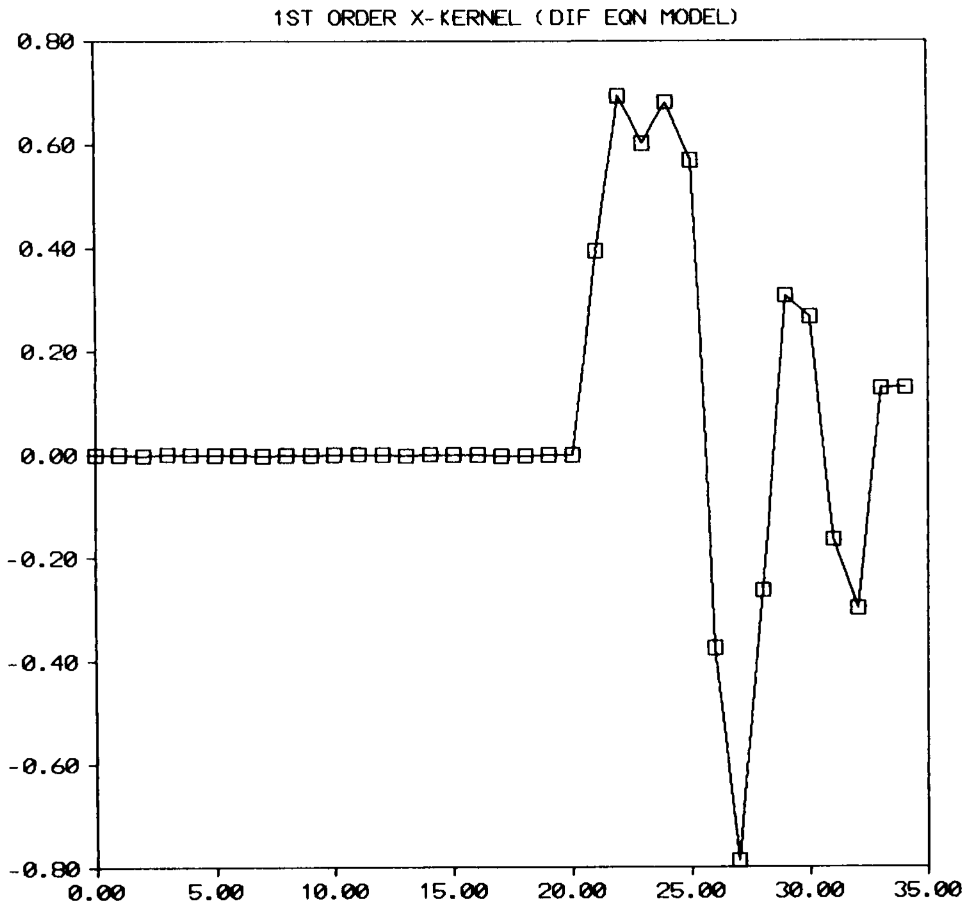

FIGURE 6. First-order x kernel, giving an indication of the significant linear x terms for (nonlinear) difference equation model.

This difference equation was used in the present example to generate 2000 input-output data pairs. The input was white noise uniformly distributed between -1.5 and 1.5.

With this input-output data, parallel cascade was utilized to identify a difference equation model having lags in x from 0 to 34, and lags in y from 1 to 35. The procedure set out earlier (multiinput/output and difference equation via parallel cascade) was employed, except that cross-product terms were not introduced.

The linear x terms selected by parallel cascade are shown in the first-order x kernel in Fig. 6. Note that this kernel indicates (correctly) that there are no significant linear x terms with a lag less than 21. The first-order y kernel (Fig. 7) does not as clearly reveal the significant terms, but does appear to rule out, for example, linear y terms with lags 16, 17, 20, 22, 28, 30, 35 (this is clearer from the actual kernel values). Since linear x and y terms are relatively few in total, there is little inconvenience in retaining as candidates any terms which are not clearly ruled out by this parallel cascade screening.

Parallel cascade is very useful in revealing the significant xx and yy terms, where

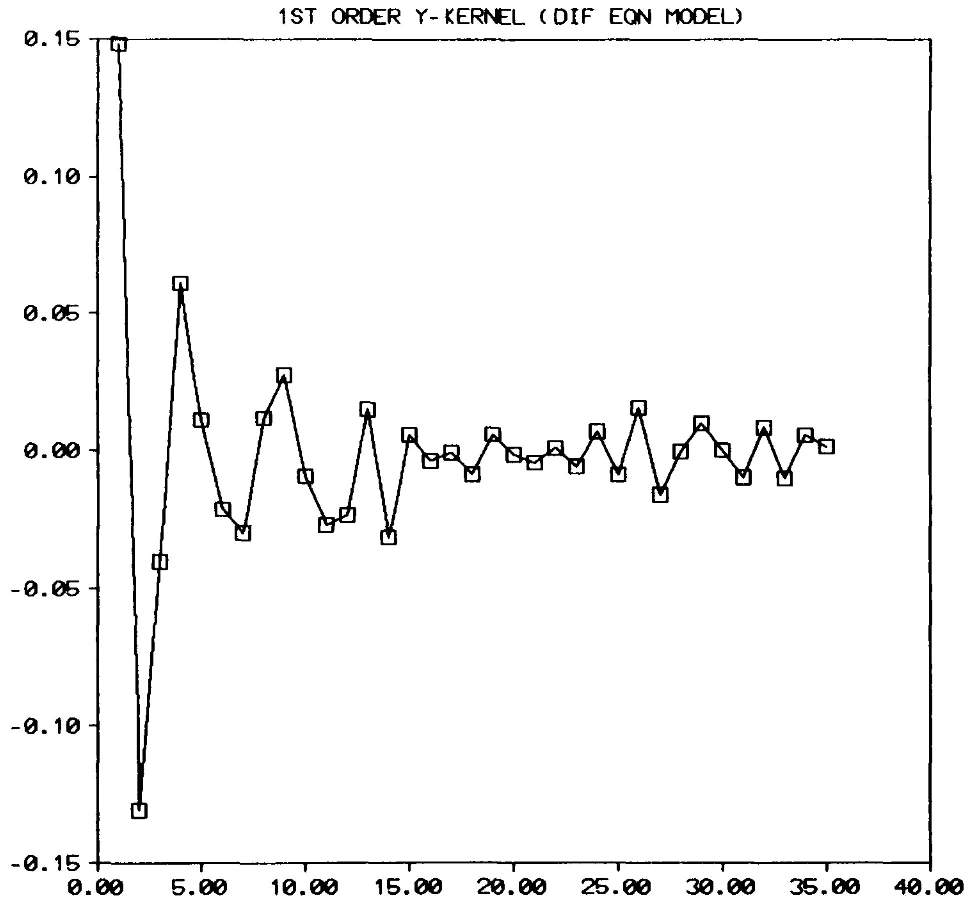

FIGURE 7. First-order y kernel, giving an indication of the significant linear y terms for (nonlinear) difference equation model.

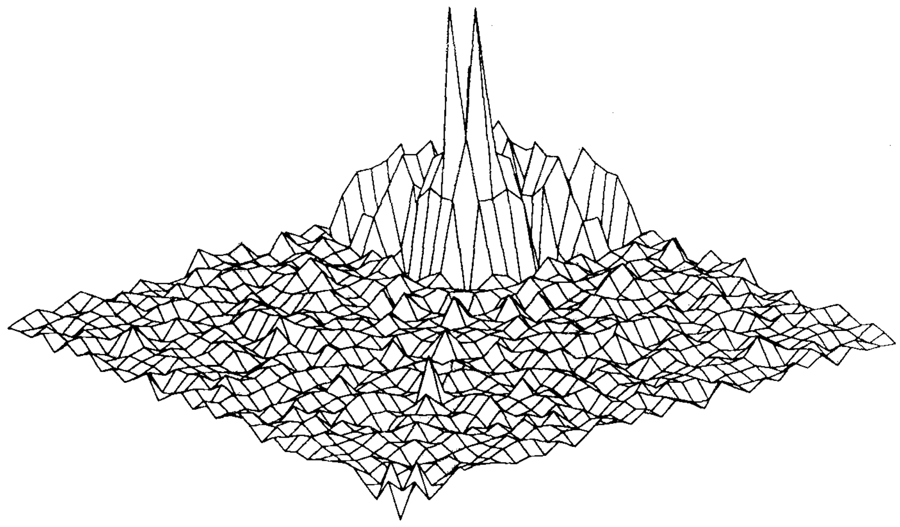

FIGURE 8. The (second-order) *xx* kernel, giving an indication of the significant *xx* terms for nonlinear difference equation model.

there are 630 candidates of each type. The (second-order) *xx* kernel (Fig. 8) correctly had its largest magnitude at the lag pair (23,24), and there were 67 other distinct terms with a kernel magnitude of at least 10% of the largest kernel magnitude. The (second-order) *yy* kernel (Fig. 9) correctly had its largest magnitude at (1,1), and only eight other distinct terms had a kernel magnitude at least 10% of this size. Thus, when there are many candidate terms of a given class, parallel cascade can pick out a much reduced subset of terms to explore.

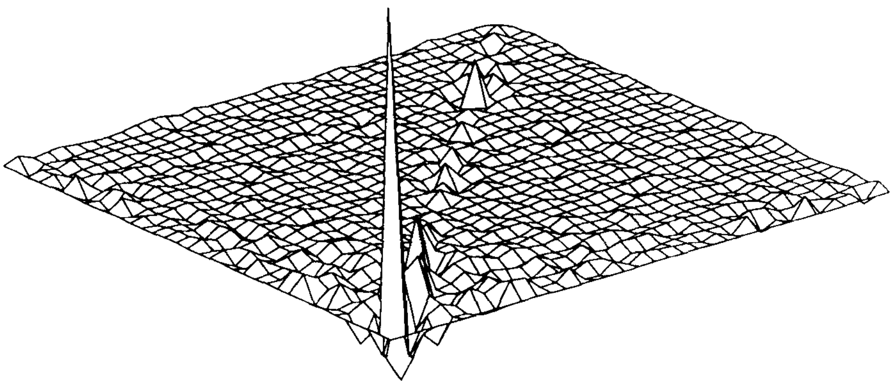

FIGURE 9. The (second-order) *yy* kernel, giving an indication of the significant *yy* terms for nonlinear difference equation model.

ALTERNATE METHOD FOR KERNEL ESTIMATION

Over the past 20 years, kernel estimation has played an increasingly important part in mathematically characterizing the functioning of nonlinear physiological systems; see in particular the book by Marmarelis and Marmarelis (18). A major breakthrough in the calculation of Wiener kernels (23) was the cross-correlation method of Lee and Schetzen (16). The following method (11,12,14) of kernel estimation is briefly presented here to clarify a few points about its operation.

For convenience, we will illustrate the method by estimating the kernels in a second-order Volterra series approximation of a nonlinear system ($M = 2$ in Eq. 1, except that the memory length is now R rather than $R + 1$):

$$z_s(n) = k_0 + \sum_{i=0}^{R-1} k_1(i)x(n-i) + \sum_{i_1=0}^{R-1}\sum_{i_2=0}^{R-1} k_2(i_1,i_2)x(n-i_1)x(n-i_2) \qquad n = 0, 1, \ldots, T \,. \tag{33}$$

To estimate the kernels k_0, k_1, k_2, begin by rewriting Eq. 33 as follows (15):

$$z_s(n) = \sum_{m=0}^{P} a(m)q_m(n) \,. \tag{34}$$

Here $P = R + R(R+1)/2$, and for $n = R, \ldots, T$, $q_o(n) = 1$,

$$q_m(n) = x(n - m + 1) \,, \qquad m = 1, \ldots, R \,. \tag{35}$$

For $m = R + 1, \ldots, P$, the $q_m(n)$ are defined as follows (colons are used below to separate portions of the code which should be set out on separate lines for increased readability):

```
$m = R$: FOR $J1 = 0$ TO $R - 1$: FOR $J2 = J1$ TO $R - 1$
$m = m + 1$: FOR $n = R$ TO $T$
$q_m(n) = x(n - J1)x(n - J2)$                                  (36)
NEXT $n$: NEXT $J2$: NEXT $J1$
```

The coefficients $a(m)$ in Eq. 34 are directly related to the kernels in Eq. 33, and are found to minimize the mse

$$e = \overline{(y(n) - z_s(n))^2}$$

computed over the record portion $n = R, \ldots, T$. To estimate the $a(m)$, we may use a Cholesky factorization, for example, via the following pseudocode:

```
D(0,0) = 1: FOR $m = 1$ TO $P$
        $D(m,0) = \overline{q_m(n)}$                                   (37)

FOR $r = 1$ TO $m$
        $D(m,r) = \overline{q_m(n)q_r(n)}$                             (38)

NEXT $r$: NEXT $m$: FOR $j = 0$ TO $P - 1$
FOR $m = j + 1$ TO $P$: $DD = D(m,j)/D(j,j)$: FOR $r = j + 1$ TO $m$
        $D(m,r) = D(m,r) - D(r,j) \cdot DD$
NEXT $r$: NEXT $m$: NEXT $j$
FOR $j = 0$ TO $P - 1$: FOR $m = j + 1$ TO $P$
        $D(m,j) = D(m,j)/D(j,j)$
NEXT $m$: NEXT $j$
        $G(0) = \overline{y(n)}$                                       (39)

FOR $m = 1$ TO P
        $G(m) = \overline{y(n)q_m(n)}$                                 (40)
NEXT $m$: FOR $j = 0$ TO $P - 1$: FOR $m = j + 1$ TO $P$
        $G(m) = G(m) - D(m,j) \cdot G(j)$
NEXT $m$: NEXT $j$: FOR $m = 1$ TO $P$
        $G(m) = G(m)/D(m,m)$
        NEXT $m$ .
```

Other codes, for example, the Cholesky outer product version [see Golub and Van Loan (5)], may also be used. However the code set out above has proved particularly robust in extensive testing.

The $a(m)$ can now be calculated from the $G(m)$ and the $D(m,j)$ according to the following formula (15):

$$a(m) = \sum_{i=m}^{P} G(i)\nu(i) \tag{41}$$

where

$$\nu(m) = 1 \tag{42}$$

$$\nu(i) = -\sum_{r=m}^{i-1} D(i,r)\nu(r) , \qquad i = m+1,\ldots,P . \tag{43}$$

Hence, the zero- and first-order kernels are

$$k_0 = a(0)$$

$$k_1(i) = a(i+1) , \qquad i = 0,\ldots,R-1 .$$

The second-order kernel $k_2(I1,I2)$ can be obtained as follows:

```
$m = R$: FOR $I1 = 0$ TO $R - 1$
FOR $I2 = I1$ TO $R - 1$: $m = m + 1$: $k_2(I1,I2) = a(m)$
IF $I1 \neq I2$ THEN $k_2(I1,I2) = 0.5\, k_2(I1,I2)$
    NEXT $I2$: NEXT $I1$ .
```

This yields the (best) second-order Volterra series approximation, with mse

$$\overline{y^2(n)} - \sum_{m=0}^{P} G^2(m)D(m,m) \ .$$

Replacing P with R in the foregoing expression gives the mse of the best first-order Volterra series approximation (having memory length R). Replacing P with R in Eqs. 41 and 43 yields the $a(m)$ (and hence the corresponding zero- and first-order kernels) for the best first-order Volterra series approximation.

The time-average on the right side of Eq. 39 is, clearly, the output mean, computed over the portion of the record $n = R, \ldots, T$. The time-average on the right side of Eq. 40 is, for $m = 1, \ldots, R$, the first-order input output cross-correlation $\phi_{xy}(m - 1)$ and for $m = R + 1, \ldots, P$, the second-order cross-correlation $\phi_{xxy}(J1,J2)$. These cross-correlations are defined as in Eqs. 16 and 17 with y_{i-1} replaced by y.

A key part of the algorithm (11,12,14) is the efficient calculation of the time-averages on the right sides of Eqs. 37 and 38, and to overlook this aspect is to miss what makes the algorithm fast. The time-average in Eq. 37 is, for $m = 1$, the input mean (as computed over $n = R, \ldots, T$), and for $m = 2, \ldots, R$ can be calculated from the mean using the relation

$$\overline{q_m(n)} = \overline{x(n)} + \frac{1}{T - R + 1} \sum_{j=0}^{m-2} [x(R - j - 1) - x(T - j)] \ . \tag{44}$$

Eq. 44 simply "corrects" for the finite record length. This correction, and in fact all the corrections for finite record length, can be carried out recursively. For example,

$$\overline{q_1(n)} = \overline{x(n)} \tag{45}$$

$$\overline{q_m(n)} = \overline{q_{m-1}(n)} + \frac{1}{T - R + 1} [x(R - m + 1) - x(T - m + 2)] \qquad m = 2, \ldots, R \ . \tag{46}$$

However, in correcting the more complicated time-averages for the finite record length, recursive calculations tend to be less accurate, and so efficient nonrecursive schemes are also pointed out following and in refs. (11,14).

The remaining time-averages in Eqs. 37 and 38 can be calculated efficiently from the input autocorrelations of first, second, and third order, respectively ϕ_{xx}, ϕ_{xxx}, and ϕ_{xxxx}, defined analogously to earlier correlations (e.g., for first and second order, replace y_{i-1} in Eqs. 16 and 17 with x). This method of calculating the required time-averages is much faster than computing them independently.

For example, in view of Eq. 36, the time-average in Eq. 38 for $m = R + 1, \ldots, P$ and $r = R + 1, \ldots, m$ has the form

$$\overline{q_m(n)q_r(n)} = \overline{x(n-J1)x(n-J2)x(n-J3)x(n-J4)} \ . \tag{47}$$

If this time-average were independently computed, for $J1 = 0, \ldots, R-1$, $J2 = J1, \ldots, R-1$, $J3 = J2, \ldots, R-1$, and $J4 = J3, \ldots, R-1$, then the number of multiplications would be about $R^4T/4$! Instead, the time-average is readily calculated from $\phi_{xxxx}(J4-J1, J3-J1, J2-J1)$ by correcting for the finite record length, and then the total number of multiplications (for required values of $J1$, $J2$, $J3$, $J4$) is about $R^3T/3$! If the memory length R is 60, then we do about 1/15 of the multiplications otherwise required.

Referring to Eq. 36, define the function $DL(J2,J1)$ which gives the value of m corresponding to given values of $J1$, $J2$:

```
$m = R$: FOR $J1 = 0$ TO $R-1$: FOR $J2 = J1$ TO $R-1$: $m = m+1$
            $DL(J2,J1) = m$                                        (48)
        NEXT $J2$: NEXT $J1$ .
```

Certain of the required time-averages on the right side of Eqs. 37 and 38 are exactly given by the autocorrelations ϕ_{xx}, ϕ_{xxx}, ϕ_{xxxx} and may be specified as follows:

```
FOR $J = 0$ to $R-1$: $m = DL(J,0)$
$\overline{q_m(n)} = \phi_{xx}(J)$
$\overline{q_{J+1}(n)q_1(n)} = \phi_{xx}(J)$
        NEXT $J$
FOR $J1 = 0$ TO $R-1$: $m1 = DL(J1,0)$
FOR $J2 = J1$ TO $R-1$: $m2 = DL(J2,0)$: $m3 = DL(J2,J1)$
$\overline{q_{m1}(n)q_{J2+1}(n)} = \phi_{xxx}(J2,J1)$
$\overline{q_{m2}(n)q_{J1+1}(n)} = \phi_{xxx}(J2,J1)$
$\overline{q_{m3}(n)q_1(n)} = \phi_{xxx}(J2,J1)$
NEXT $J2$: NEXT $J1$
FOR $J1 = 0$ TO $R-1$: $m1 = DL(J1,0)$
FOR $J2 = J1$ TO $R-1$: $m2 = DL(J2,0)$: $m21 = DL(J2,J1)$
FOR $J3 = J2$ TO $R-1$: $m3 = DL(J3,0)$: $m31 = DL(J3,J1)$: $m32 = DL(J3,J2)$
$\overline{q_{m32}(n)q_{m1}(n)} = \phi_{xxxx}(J3,J2,J1)$
$\overline{q_{m31}(n)q_{m2}(n)} = \phi_{xxxx}(J3,J2,J1)$
$\overline{q_{m3}(n)q_{m21}(n)} = \phi_{xxxx}(J3,J2,J1)$
NEXT $J3$: NEXT $J2$: NEXT $J1$ .
```

Depending on the program setup, it may be important to note that $m3$ can be less than, equal to, or greater than $m21$.

The remaining time-averages required in Eqs. 37 and 38 can be calculated from the autocorrelations ϕ_{xx}, ϕ_{xxx}, ϕ_{xxxx} by correcting for the finite record length. Consider the time-averages derived from ϕ_{xx}. These time-averages can be calculated recursively using the relation:

$$\overline{x(n-J1)x(n-J2)} = \overline{x(n-J1+1)x(n-J2+1)} + \left\{\frac{1}{T-R+1} \times [x(R-J1)x(R-J2) - x(T-J1+1)x(T-J2+1)]\right\} . \tag{49}$$

The following pseudocode carries out this recursive calculation and assigns the result to each of the time-averages having the same value:

```
FOR $J = 0$ TO $R - 1$: $A1(J) = \Phi_{xx}(J)$: NEXT $J$
FOR $J1 = 1$ TO $R - 1$: FOR $J2 = J1$ TO $R - 1$: $m = DL(J2,J1)$
$A1(J2 - J1) = A1(J2 - J1) + (x(R - J1)x(R - J2)$
                                        $- x(T - J1 + 1)x(T - J2 + 1))/(T - R + 1)$
$\overline{q_m(n)} = A1(J2 - J1)$
$\overline{q_{J2+1}(n)q_{J1+1}(n)} = A1(J2 - J1)$
NEXT $J2$: NEXT $J1$ .
```

Time-averages derived from ϕ_{xxx} can be calculated recursively using a relation analogous to Eq. 49. This can be expressed in the following pseudocode, which again assigns the result to every time-average having the same value.

```
FOR $J1 = 0$ TO $R - 1$: FOR $J2 = J1$ TO $R - 1$
$A2(J2,J1) = \Phi_{xxx}(J2,J1)$: NEXT $J2$: NEXT $J1$
FOR $J1 = 1$ TO $R - 1$: FOR $J2 = J1$ TO $R - 1$: $m21 = DL(J2,J1)$
FOR $J3 = J2$ TO $R-1$: $m31 = DL(J3,J1)$: $m32 = DL(J3,J2)$
$A2(J3 - J1,J2 - J1) = A2(J3 - J1,J2 - J1)$
                        $+ x(R - J1)x(R - J2)x(R - J3)/(T - R + 1)$
$A2(J3 - J1,J2 - J1) = A2(J3 - J1,J2 - J1) - [x(T - J1 + 1)$
                        $\times x(T - J2 + 1)x(T - J3 + 1)]/(T - R + 1)$
$\overline{q_{m21}(n)q_{J3+1}(n)} = A2(J3 - J1,J2 - J1)$
$\overline{q_{m31}(n)q_{J2+1}(n)} = A2(J3 - J1,J2 - J1)$
$\overline{q_{m32}(n)q_{J1+1}(n)} = A2(J3 - J1,J2 - J1)$
NEXT $J3$: NEXT $J2$: NEXT $J1$ .
```

Finally, time-averages derived from ϕ_{xxxx} can be calculated recursively via a formula analogous to Eq. 49, which can be expressed in the following pseudocode:

```
FOR $J1 = 0$ TO $R - 1$: FOR $J2 = J1$ TO $R - 1$: FOR $J3 = J2$ TO $R - 1$
$A3(J3,J2,J1) = \phi_{xxxx}(J3,J2,J1)$: NEXT $J3$: NEXT $J2$: NEXT $J1$
FOR $J1 = 1$ TO $R - 1$: FOR $J2 = J1$ TO $R - 1$: $m21 = DL(J2,J1)$
FOR $J3 = J2$ TO $R - 1$: $m31 = DL(J3,J1)$: $m32 = DL(J3,J2)$
FOR $J4 = J3$ TO $R - 1$: $m41 = DL(J4,J1)$: $m42 = DL(J4,J2)$
                              $m43 = DL(J4,J3)$
$A3(J4 - J1,J3 - J1,J2 - J1) = A3(J4 - J1,J3 - J1,J2 - J1) + [x(R - J1)$
                              $\times x(R - J2)x(R - J3)x(R - J4)]/(T - R + 1)$
$A3(J4 - J1,J3 - J1,J2 - J1) = A3(J4 - J1,J3 - J1,J2 - J1)$
                              $- [x(T - J1 + 1)x(T - J2 + 1)x(T - J3 + 1)$
                              $\times x(T - J4 + 1)]/(T - R + 1)$
$\overline{q_{m43}(n)q_{m21}(n)} = A3(J4 - J1,J3 - J1,J2 - J1)$
$\overline{q_{m42}(n)q_{m31}(n)} = A3(J4 - J1,J3 - J1,J2 - J1)$
$\overline{q_{m41}(n)q_{m32}(n)} = A3(J4 - J1,J3 - J1,J2 - J1)$
NEXT $J4$: NEXT $J3$: NEXT $J2$: NEXT $J1$ .
```

Again, depending on the setup of the program, it may be important to note that $m41$ may be less than, equal to, or greater than $m32$.

The time-averages in Eqs. 37 and 38 can also be obtained nonrecusively, but still efficiently. For example, time-averages derived from ϕ_{xxx} can be calculated via the formula (where $1 \le J1 \le J2 \le J3$):

$$\overline{x(n - J1)x(n - J2)x(n - J3)}$$
$$= \phi_{xxx}(J3 - J1, J2 - J1) + \left\{ \frac{1}{T - R + 1} \sum_{l=1}^{J1} \left[x(R - l)x(R - l + J1 - J2)x(R - l + J1 - J3) - \left(x(T + 1 - l)x(T + 1 - l + J1 - J2)x(T + 1 - l + J1 - J3) \right) \right] \right\} .$$

Analogous formulas exist for calculating time-averages derived from ϕ_{xx} and ϕ_{xxxx}, and the foregoing pseudocode can readily be modified to utilize these nonrecursive formulas. It is emphasized that the pseudocode presented here is merely illustrative, and it is trivial to set down more efficient code. Note that the terms $q_m(n)$ need not be created or stored. Rather, only time-averages involving these terms are required, and can be obtained efficiently and accurately as illustrated above.

CONCLUSION

There are many variations possible in parallel cascade identification (8,9,13). The essential feature (8) is to approximate the nonlinear system by a first cascade, then approximate the residue using a second cascade, and so on. This enables the cascade paths in the parallel array to be identified one at a time. As we have seen, under broad conditions the nonlinear system can be approximated to an arbitrary accuracy, in the mean-square sense, by a sum of the individually obtained cascades. Note that in place of some or all of the cascade paths, one could substitute a parallel array of simple nonlinear difference equations (8), which are successively fit to the (updated) residue one at a time.

REFERENCES

1. Banyasz, C.S.; Haber, R.; Keviczky, L. Some estimation methods for nonlinear discrete time identification. IFAC Symp. Ident. Sys. Param. Est. 3:793-802; 1973.
2. Dieudonné, J. Foundations of modern analysis. Berlin, Heidelberg, New York: Springer; 1976.
3. Frechet, M. Sur les fonctionnelles continues. Annales Scientifiques de l'Ecole Normal Superieure 27:193-219; 1910.
4. Glass, L.; Mackey, M.C. From clocks to chaos. Princeton: Princeton University; 1988.
5. Golub, G.H.; Van Loan, C.F. Matrix computations (2nd ed.). Baltimore: Johns Hopkins Univ. Press; 1989.
6. Hunter, I.W.; Korenberg, M.J. The identification of nonlinear biological systems: Wiener and Hammerstein cascade models. Biol. Cybern. 55:135-144; 1986.
7. Kolmogoroff, A.N. On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition (Russian). Dokl. Akad. Nauk. SSSR 114:953-956; 1957; AMS Transl. 2:55-59; 1963.
8. Korenberg, M.J. Statistical identification of parallel cascades of linear and nonlinear systems. IFAC Symp. Ident. Sys. Param. Est. 1:580-585; 1982.
9. Korenberg, M.J. Functional expansions, parallel cascades and nonlinear difference equations. In: Marmarelis, V.Z. ed. Advanced methods of physiological system modeling. Los Angeles: USC Biomedical Simulations Resource, Vol. 1; 1987: pp. 221-240.
10. Korenberg, M.J. Exact orthogonal estimation of kernels with biological applications. IEEE Montech Conference on Biomedical Technologies, pp. 27-32; 1987.
11. Korenberg, M.J. Identifying nonlinear difference equation and functional expansion representations: The fast orthogonal algorithm. Ann. Biomed. Eng. 16:123-142; 1988.
12. Korenberg, M.J. A robust orthogonal algorithm for system identification and time-series analysis. Biol. Cybern. 60:267-276; 1989.
13. Korenberg, M.J. A rapid and accurate method for estimating the kernels of a nonlinear system with lengthy memory. 15th Biennial Symp. Communications. June 1990; Queen's University, Kingston, Canada, pp. 57-60.
14. Korenberg, M.J. Some new approaches to nonlinear system identification and time-series analysis. Proc. 12th Annual International Conference of the IEEE Engineering in Medicine and Biology Society 12(1):20-21; 1990.
15. Korenberg, M.J.; Bruder, S.B.; McIlroy, P.J. Exact orthogonal kernel estimation from finite data records: Extending Wiener's identification of nonlinear systems. Ann. Biomed. Eng. 16:201-214; 1988.
16. Lee, Y.W.; Schetzen, M. Measurement of the Wiener kernels of a non-linear system by cross-correlation. Int. J. Control 2:237-254; 1965.
17. Liebovitch, L.S. Introduction to the properties and analysis of fractal objects, processes, and data. In: Marmarelis, V.Z., ed. Advanced methods of physiological system modeling. New York: Plenum Press, Vol. 2; 1989: pp. 225-239.
18. Marmarelis, P.Z.; Marmarelis, V.Z. Analysis of physiological systems. The white-noise approach. New York: Plenum Press; 1978.
19. McIlroy, P.J.H. Applications of nonlinear systems identification. Kingston, Ontario, Canada: Queen's University; 1986. M.Sc. thesis.
20. Mo, L.; Elkasabgy, N. Elec-841 Report, Dept. Elect. Eng., Queen's Univ., Kingston, Ontario, Canada; 1984.
21. Palm, G. On the representation and approximation of nonlinear systems. Part II: Discrete time. Biol. Cybern. 34:49-52; 1979.
22. Shi, J.; Sun, H.H. Nonlinear system identification via parallel cascaded structure. Proc. 12th Annual International Conference of the IEEE Engineering in Medicine and Biology Society 12(4):1897-1898; 1990.
23. Wiener, N. Nonlinear problems in random theory. New York: Wiley; 1958.