Biol. Cybern. 71, 187–195 (1994)

Biological Cybernetics

Calculation of the Volterra kernels of non-linear dynamic systems using an artificial neural network

Jonathan Wray, Gary G. R. Green

Department of Physiological Sciences, The Medical School, Framlington Place, Newcastle upon Tyne, NE2 4HH, UK

Received: 4 June 1993/Accepted in revised form: 2 March 1994

Abstract. The Volterra series is a well-known method of describing non-linear dynamic systems. A major limitation of this technique is the difficulty involved in the calculation of the kernels. More recently, artificial neural networks have been used to produce black box models of non-linear dynamic systems. In this paper we show how a certain class of artificial neural networks are equivalent to Volterra series and give the equation for the nth order Volterra kernel in terms of the internal parameters of the network. The technique is then illustrated using a specific non-linear system. The kernels obtained by the method described in the paper are compared with those obtained by a Toeplitz matrix inversion technique.

1 Introduction

Many non-linear systems exist within the realms of biology. However, due to the nature of some of these systems, it has proved difficult to produce and solve analytic equations which describe them. An alternative to mathematical analysis (which produces analytic equations describing the workings of a system) is to take a general black box model and obtain the specific parameters which enable the model to describe a system. Two such methods, which are closely related to each other, are the Volterra and Wiener series (Marmarelis and Marmarelis 1978; Schetzen 1980; Rugh 1981). These functional approaches to system identification have found many applications in neurobiology. For example, the Volterra series has been used to characterise the receptive fields of auditory neurones (Aertsen and Johannesma 1981), in the modelling of movement detection (Poggio and Reichardt 1973) and in the analysis of network models of lateral inhibition (Nabet and Pinter 1992). The Wiener series has been used in the analysis of neurone chains (Marmarelis and Naka 1972), in the determination of the non-linear properties of cells in the visual cortex (Emerson et al. 1992) and in the investigation of the dynamics of cockroach ocellar neurones (Mizunami et al. 1986). The Volterra series can be shown to be a general solution of non-linear differential equations using differential algebraic approaches (Fliess et al. 1983). However, at present no general methods exist to calculate the Volterra kernels for non-linear dynamic systems, although they can be calculated for systems whose order is known and finite (Schetzen 1965). This method is not generally applicable since the order of a system is not typically known a priori. Conversely, general methods do exist to calculate the Wiener kernels, although these can be highly inaccurate (Palm and Poggio 1977). This paper tackles the problem of Volterra kernel calculation for systems of unknown order. First, Volterra and Wiener series modelling is introduced in detail, and some associated problems are discussed. A method is developed to calculate the Volterra kernels for non-linear systems of unknown order, via a proof demonstrating the equivalence between a certain neural network architecture and the finite memory, discrete, Volterra series. The method is compared to the currently most accurate method of Wiener kernel production: Toeplitz matrix inversion (Korenberg and Hunter 1990). The comparison is effected by using both methods to calculate the kernels for a system that can be described purely by its second-order kernel. The Wiener and Volterra kernels for such a system are identical (Marmarelis and Marmarelis 1978). The new method produces a substantial improvement in accuracy, judged both by comparison of the kernels and by the kernels' ability to predict the system response to new stimuli.

2 Volterra and Wiener modelling

The Volterra approach characterises a system as a mapping between two function spaces, which represent the input and output spaces of that system. The Volterra series is an extension of the Taylor series representation

Correspondence to: J. Wray, The Neurosciences Institute, 3377 North Torrey Pines Court, La Jolla, CA 92037, USA

to cover dynamic systems and has the general form

$$y(t) = h_0 + \int_{-\infty}^{\infty} h_1(\tau_1)u(t-\tau_1)\,\mathrm{d}\tau_1 + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2)\,\mathrm{d}\tau_1\,\mathrm{d}\tau_2 + \cdots + \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} h_n(\tau_1,\ldots,\tau_n)u(t-\tau_1)\ldots u(t-\tau_n)\,\mathrm{d}\tau_1\ldots\mathrm{d}\tau_n + \ldots \quad (1)$$

where $y(t)$ is the output of the system at time t, $u(t)$ is the input at time t, and $h_n(\tau_1,\ldots,\tau_n)$ is the nth order Volterra kernel. Equation (1) has an alternative functional form,

$$y(t) = \mathscr{H}_1[u(t)] + \mathscr{H}_2[u(t)] + \cdots + \mathscr{H}_n[u(t)] + \ldots \quad (2)$$

where

$$\mathscr{H}_n[u(t)] = \int_{-\infty}^{\infty}\ldots\int_{-\infty}^{\infty} h_n(\tau_1,\ldots,\tau_n)u(t-\tau_1)\ldots u(t-\tau_n)\,\mathrm{d}\tau_1\ldots\mathrm{d}\tau_n$$

and is known as the nth order Volterra operator.

Volterra (1959) showed that such a series is capable of representing any analytic, time-invariant system. However, two major problems exist with this approach to system identification. The first is due to the fact that the Volterra series is a polynomial approximation to a system. It is a direct extension of the Taylor series to systems with memory, so the well-known limitations of the Taylor series apply equally to the Volterra series. The second problem relates to calculating the kernels for a specific system. The work of Wiener (1958) was seen as a way to overcome these problems. In function approximation a standard technique (Cheney 1982) is to use sums of orthogonal polynomials of different orders. This method overcomes two of the major problems associated with Taylor series approximation. First, it allows the approximation of non-analytic functions, which is not possible with a Taylor series since to calculate the series coefficients, the function to be approximated needs to be analytic. Second, it overcomes the problem of truncation. It can be shown (Cheney 1982) that the sum of Chebyshev polynomials, up to degree n, provides the best approximation, in terms of the Chebyshev norm, of degree n. This is not the case for a Taylor series where truncation to degree n will not, in general, give the best approximation of degree n. Wiener extended the use of orthogonal polynomials to functional analysis by modification of the Volterra series to give an expansion made up of orthogonal functionals. This extension produced the Wiener series in which every functional is orthogonal to each other when the input signal is Gaussian white noise. Orthogonality of the functionals means the Wiener series of a specific order is the best possible approximation of the system to that order. In addition, functional orthogonality implies that the kernels can be calculated independently. The method of Wiener has since been improved, and other techniques of Wiener kernel extraction have been developed (Lee and Schetzen 1965; de Boer and Kuyper 1968; Korenberg and Hunter 1990).

A number of limitations of the Wiener approach and the kernel extraction techniques have been reported. One of the most obvious problems is that ideal white noise cannot be generated since it has infinite power, meaning that coloured or band-limited noise has to be used in any practical situation (Marmarelis and Marmarelis 1978). This deviation from the ideal leads to errors in the kernels. Another problem (Palm and Poggio 1977) is that kernel calculation techniques cannot be extended to kernels of degree 3 and above, due to delta functions being produced of the kernels. In addition, errors in kernel calculation occur due to the use of finite length input and output signals (Korenberg and Hunter 1990).

As well as these problems of kernel calculation, the superiority of the Wiener series over the Volterra series has been questioned. In a practical situation *finite* Wiener series must be used (Korenberg and Hunter 1990), and since every finite Wiener series has an equivalent Volterra series (Palm and Poggio 1977), Wiener series offer no advantage in terms of approximation ability. In addition, the ability of the Wiener series to approximate a larger class of functions than a Volterra series is due solely to weakening of the error criterion from a quadratic norm to a Chebyshev norm (Palm and Pöppel 1985). For these reasons the Wiener series is, in principle, no better in a practical situation than a Volterra series. The only practical advantage of the Wiener series is that methods exist to calculate the kernels. In a Volterra series the functionals are not independent, and so to enable their calculation, the order of the system has to be known in advance (Schetzen 1965). In this paper a method for calculating the Volterra kernels for systems of unknown order is presented. This is achieved by effectively calculating them all simultaneously.

One advantage the Volterra series has over the Wiener series lies in the interpretation of the system kernels. The Volterra kernels are a direct extension of the impulse response concept from linear system theory to multiple dimensions. Thus, just as the impulse response of a linear system can be used to describe that system, the kernels, which are a multidimensional analogue of the impulse response, can be used to characterise a non-linear system.

This section introduced Volterra and Wiener series and mentioned some problems that exist in the kernel calculation and the subsequent use of the series in modelling. In the next section we show how an artificial neural network of a certain class is equivalent to a finite memory Volterra series, and we produce the equations necessary to calculate the Volterra kernels from a network. To work in discrete time, the discrete Volterra series needs to be introduced, and is given by

$$y(t) = h_0 + \sum_{\tau_1=0}^{\infty} h_1(\tau_1)u(t-\tau_1) + \sum_{\tau_1=0}^{\infty}\sum_{\tau_2=0}^{\infty} h_2(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2)$$

$$+\cdots+\sum_{\tau_1=0}^{\infty}\cdots\sum_{\tau_n=0}^{\infty} h_n(\tau_1,\ldots,\tau_n)u(t-\tau_1)\ldots u(t-\tau_n)+\ldots \quad (3)$$

where $u(t)$ and $y(t)$ are the input and output at time t, respectively, and $h_n(\tau_1 \ldots \tau_n)$ is the nth order Volterra kernel. It is assumed in this equation that the system is causal and so the delay indicies τ_i start at zero. For a practical system with finite memory the equation becomes

$$y(t)=h_0+\sum_{\tau_1=0}^{T} h_1(\tau_1)u(t-\tau_1)+\sum_{\tau_1=0}^{T}\sum_{\tau_2=0}^{T} h_2(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2)+\ldots+\sum_{\tau_1=0}^{T}\cdots\sum_{\tau_n=0}^{T} h_n(\tau_1,\ldots,\tau_n)u(t-\tau_1)\ldots u(t-\tau_n)+\ldots \quad (4)$$

where T is the memory of the system (i.e. the number of time sampled values needed to describe the dynamics of the system).

3 Artificial neural networks

Artificial neural networks are collections of processing nodes connected together in a network in such a way as to be able to perform a function. One such network, termed a feed-forward network or multilayer perceptron, is shown in Fig. 1. Such a network is trained to perform a certain mapping between its input layer (on the left) and its output by altering the weights associated with each internal connection. These weights are altered by a training algorithm which takes pairs of ideal input/output data and changes the weights to make the network reproduce the mapping described by the data pairs. Many such algorithms exist, but they all attempt to solve the same problem: to find the optimal minimum of a surface in weight space, whose height is given by the total error between the network's actual output and its required output, for all the training pairs. An introduction to neural networks can be found in many texts (Lippmann 1987; Hertz et al. 1990).

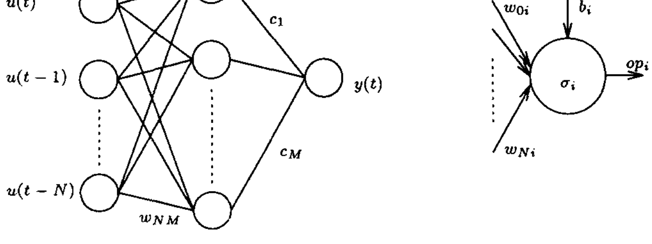

Fig. 1. **a** A neural network architecture for time series prediction. **b** A block diagram of one hidden node of the network shown in **a**

In recent years artificial neural networks have been considered in terms of their function approximation ability, following the theoretical result of a number of authors (Cybenko 1989; Funahashi 1989; Hornik et al. 1989) proving that a feed-forward network with one hidden layer can approximate any continuous function, to any degree of accuracy. One specific architecture, shown in Fig. 1a, has been used in many applications to model non-linear dynamic systems (Hudson et al. 1990; Weigend et al. 1990; Chakraborty et al. 1992). This network is a specific, one-input-variable example of a so-called time-delay neural network (Waibel et al. 1989). The output node of the network represents the current output of the system, and the input nodes represent the inputs to the system sampled in time.

A typical nodal processing function is

$$op_i=\sigma_i\left(b_i+\sum_{j=0}^{N} w_{ji}u(t-j)\right) \quad (5)$$

where op_i is the output from hidden unit i, σ_i is the output function of hidden unit i, w_{ji} is the weight connecting input unit j to hidden unit i, $u(t-j)$ is the input u at delay j, b_i is the bias input to unit i, and $N+1$ is the number of input units. A typical output function, σ, is of sigmoidal shape such as the hyperbolic tangent ($\tanh x$). Figure 1b shows a block diagram of the processing described by (5), occurring at the hidden nodes of the network in Fig. 1a.

In a previous paper (Wray and Green 1991) we have shown how, by considering the output function, such as $\tanh(x)$, in terms of its Taylor series equivalent, the bias input to a node acts to change the output function. Thus, if the hidden nodes have different bias inputs, as is generally the case, we can consider each hidden node as having a different polynomial output function, $p_i(x)$. This can be written as

$$p_i(x)=a_{0i}+a_{1i}x+a_{2i}x^2+\cdots+a_{ni}x^n+\cdots \quad (6)$$

From Fig. 1 if the output unit has no non-linearity, the output of the network can be expressed as

$$y(t)=\sum_{i=1}^{M} c_i p_i(x_i) \quad (7)$$

where c_i is the weight from hidden unit i to the output unit, M is the number of hidden units, and x_i is the weighted sum of inputs into hidden unit i given by

$$x_i=\sum_{j=0}^{N} w_{ji}u(t-j) \quad (8)$$

If (6) and (7) are combined, we get

$$\begin{aligned}y(t)=&\,c_1(a_{01}+a_{11}x_1+a_{21}x_1^2+\cdots)\\&+c_2(a_{02}+a_{12}x_2+a_{22}x_2^2+\cdots)\\&\vdots\\&+c_M(a_{0M}+a_{1M}x_M+a_{2M}x_M^2+\cdots)\end{aligned} \quad (9)$$

now substituting for x_i and gathering like terms we get

$$
\begin{aligned}
y(t) = {} & c_1a_{01} + c_2a_{02} + \cdots + c_Ma_{0M} \\
& + c_1a_{11}\sum_{j=0}^{N} w_{j1}u(t-j) + c_2a_{12}\sum_{j=0}^{N} w_{j2}u(t-j) \\
& + \cdots + c_Ma_{1M}\sum_{j=0}^{N} w_{jM}u(t-j) \\
& + c_1a_{21}\left(\sum_{j=0}^{N} w_{j1}u(t-j)\right)^2 \\
& + c_2a_{22}\left(\sum_{j=0}^{N} w_{j2}u(t-j)\right)^2 + \ldots \\
& + c_Ma_{2M}\left(\sum_{j=0}^{N} w_{jM}u(t-j)\right)^2 + \ldots
\end{aligned}
\tag{10}
$$

$$
\begin{aligned}
= {} & c_1a_{01} + c_2a_{02} + \cdots + c_Ma_{0M} \\
& + \sum_{j=0}^{N} (c_1a_{11}w_{j1} + c_2a_{12}w_{j2} \\
& + \cdots + c_Ma_{1M}w_{jM})u(t-j) \\
& + \sum_{j=0}^{N}\sum_{k=0}^{N} (c_1a_{21}w_{j1}w_{k1} + c_2a_{22}w_{j2}w_{k2} \\
& + \cdots + c_Ma_{2M}w_{jM}w_{kM})u(t-j)u(t-k) + \ldots
\end{aligned}
\tag{11}
$$

Comparing (4) and (11) they are seen to be equivalent, demonstrating that a network of the architecture shown in Fig. 1 is equivalent to a finite memory, discrete infinite Volterra series. The Volterra kernels are given by

$$h_0 = \sum_{i=1}^{M} c_ia_{0i} \tag{12}$$

$$h_1(j) = \sum_{i=1}^{M} c_ia_{1i}w_{ji} \tag{13}$$

$$h_2(j, k) = \sum_{i=1}^{M} c_ia_{2i}w_{ji}w_{ki} \tag{14}$$

and so the general nth order kernel is given by

$$h_n(v_1, v_2, \ldots, v_n) = \sum_{i=1}^{M} c_ia_{ni}w_{v_1i}w_{v_2i}\ldots w_{v_ni} \tag{15}$$

Thus, if a network of the architecture shown in Fig. 1 can be trained to adequately represent the system under study, then the Volterra kernels of all dimensions of that system can be extracted.

Note that the discussion so far has concentrated on single-input Volterra series. The Volterra series representation has been extended to multiple dimensions (Poggio and Reichardt 1973), and similarly the neural network representation can be extended to multiple inputs. Wray (1992) showed how the above derivation can be extended to multiple input variables to allow the calculation of the kernels for multiple-input systems.

3.1 Calculation of hidden node polynomials

One problem of this technique, which is not obvious from the above derivation, is the calculation of the coefficients a_{ji} in the polynomial expansion of the hidden node output functions p_i.

Let the output function of the hidden nodes be tanh(x), although this argument can be developed equally well to apply to any output function. The hyperbolic tangent has the following Maclaurin expansion

$$\tanh(x) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}B_n(2^{4n} - 2^{2n})x^{2n-1}}{(2n)!} \tag{16}$$

where B_n are the Bernoulli numbers given by

$$B_n = \frac{2(2n)!}{(2\pi)^{2n}} \sum_{m=1}^{\infty} \frac{1}{m^{2n}}$$

When this is expanded, the following polynomial is obtained

$$\tanh(x) = x - \frac{1}{3}x^3 + \frac{2}{15}x^5 - \frac{17}{315}x^7 + \cdots \tag{17}$$

If instead of considering the bias as part of the activation function it is considered part of the output function, then it has the effect of 'shifting' the output function along the x-axis (Wray and Green 1991). The output of each hidden node then becomes

$$
\begin{aligned}
p_i(x) = {} & (x + b_i) - \frac{1}{3}(x + b_i)^3 + \frac{2}{15}(x + b_i)^5 \\
& - \frac{17}{315}(x + b_i)^7 + \ldots
\end{aligned}
\tag{18}
$$

This has the effect of altering the coefficients of each power term, which includes making the even terms nonzero. Thus, the coefficients a_{ji} in (6) depend on the expansion of (18). In fact, they are given by

$$a_{ji} = \sum_{k=j}^{\infty} {}^kC_j d_k b_i^{k-j} \tag{19}$$

where ${}^kC_j$ is the combination given by ${}^kC_j = k!/(k-j)!j!$, and d_k is the coefficient of the kth power in the original, non-biased polynomial of the hidden node output function. Thus, the ability to calculate the values of a_{ji}, essential for the calculation of the kernels, depends on the convergence of the series given in (19). The critical factor in the convergence of this series is the bias value b_i. Since the radius of convergence of the Taylor series of tanh x is $\pi/2$, if the bias value goes outside the range $[-\pi/2, \pi/2]$, then (19) diverges. Although typical the values of the weights in a network are less than $\pi/2$, this is not always the case, resulting in a serious problem.

Two possible solutions of this problem are described in the following paragraphs. One is an alternative method of calculating the coefficients of the final biased nodal output functions, and the other involves the use of an alternative function on the output of the hidden nodes.

3.1.1 Alternative calculation method. The alternative calculation method uses the fact that (17) is the Taylor series expansion of tanh (x) around zero. If instead of taking this expansion and using the bias values to alter the coefficients the Taylor series for tanh (x) is evaluated around the bias values, then the polynomials are generated directly, and this problem is overcome. The consideration then becomes the calculation of the Taylor series around the different bias values. This calculation is performed by substituting the values of the individual biases into the equation for the Taylor expansion around an arbitrary point. For example, if hyperbolic tangent output functions are used, then the coefficient a_{ji} of the jth power in the equivalent polynomial for the hidden node output function p_i is given by

$$a_{ji} = \frac{1}{j} \tanh^{(j)}(b_i) \tag{20}$$

where $\tanh^{(j)}(x)$ is the jth derivative of $\tanh(x)$.

Thus, the calculation of the a_{ji} values requires the higher derivatives of the output function used, and these must be calculated analytically. The calculation of these higher derivatives is cumbersome, and if the nodal output functions are changed, to another sigmoid say, then all the derivatives will have to be recalculated. One solution is to use a symbolic manipulation package, such as Mathematica (Wolfram 1988), which has built-in algorithms to produce the power series of any function around a given point. Using these tools, the practical considerations of analytically working out the higher derivatives are removed.

3.1.2 Alternative nodal function. The other solution to this problem is to use an alternative output function in the hidden layers. In a previous paper (Wray and Green 1991) we discussed the use of finite polynomial output functions in the hidden layer to produce a finite polynomial approximation to the training data. The use of finite polynomial output functions can be extended to the architecture of the network given in Fig. 1. If the sigmoidal output function, of whichever form, is replaced by a finite function, such as $e_0 + e_1x + e_2x^2 + e_3x^3 + e_4x^4$, and no bias inputs are used, then the coefficients a_{ji} in (6) are known immediately. The disadvantage of this technique is that a finite order nodal function means that only a limited order Volterra series can be produced, i.e. if a fourth-order nodal function is used, then only a fourth-order Volterra series can be produced. This need not be a problem. If the network, using the alternative functions, learns the training data, and the training data adequately represents the system, then a finite order Volterra series can be used to describe the system.

4 Illustration of kernel extraction

The previous section showed that a neural network with a specific architecture is equivalent to a Volterra series. Equations for the kernels were produced. Thus, if a network can be trained to represent the dynamic behaviour of a system, then the Volterra kernels of that system can be calculated.

To illustrate this technique, and to enable comparison with other methods, the non-linear system used by Korenberg and Hunter (1990) to illustrate their method of kernel extraction using Toeplitz matrix inversion will be used. The system they used was a Wiener cascade model, which consists of a linear dynamic stage followed by a static non-linear stage, as illustrated in Fig. 2.

Korenberg and Hunter chose the linear dynamic stage to be low-pass, underdamped and second order and the static non-linear stage to be a squarer. They do not, however, give any specific details of the linear dynamic system. Consequently, parameters of the system implemented were chosen to give an impulse response that looked like that of the system used by Korenberg and Hunter. The system used has the impulse response given by

$$\mu(t) = \frac{a}{m} \exp(-kt) \sin mt \tag{21}$$

where $a = 2$, $m = 0.3$ and $k = 0.08$ (shown graphically in Fig. 3).

One major advantage of using this Wiener cascade as a test system is that its output can be written analytically in terms of the input

$$y(t) = \left[\int_{-\infty}^{\infty} \mu(t)u(t-\tau)\,\mathrm{d}\tau \right]^2$$
$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mu(\tau_1)\mu(\tau_2)u(t-\tau_1)u(t-\tau_2)\,\mathrm{d}\tau_1\,\mathrm{d}\tau_2 \tag{22}$$

This is a second-order Volterra functional of the input u, and so the system can be described totally by its

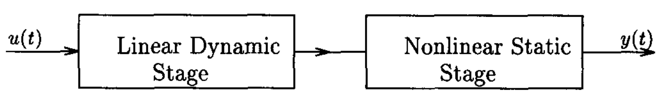

Fig. 2. A Wiener cascade model

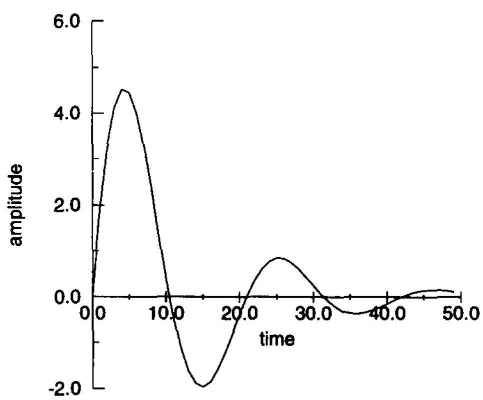

Fig. 3. Impulse response of the linear stage of the Wiener cascade model

second-order kernel which is given by

$$h_2(\tau_1, \tau_2) = \mu(\tau_1)\mu(\tau_2)$$
$$= \frac{a^2}{m^2}\exp(-k(\tau_1 + \tau_2))\sin m\tau_1 \sin m\tau_2 \qquad (23)$$

The function described by (23) is shown graphically in Fig. 4. Comparison to this ideal, analytic kernel will determine the relative accuracy of the kernel extraction techniques.

4.1 Experimental details

For both kernel calculation techniques an input time series and a corresponding output time series are required. For the Toeplitz matrix technique of Korenberg and Hunter, an input signal of Gaussian coloured noise is required. Such a signal was generated using the Box-Muller method of random number generation (Knuth 1981). Repeated calls to this algorithm produces a signal whose amplitude is random and Gaussianly distributed, with a mean of zero and a variance of one. The signal so produced was applied to the non-linear system described above, generating an output time series. In the specific experiment described here, an input time series of 4000 points and the corresponding output signal of 4000 points were used to calculate the Volterra kernels of the Wiener cascade using Toeplitz matrix inversion and the neural network method. The same two 4000-point time series were used in both calculation methods.

The Toeplitz matrix method was implemented using the procedure given by Korenberg and Hunter (1990). The essence of the technique is the solution of a set of linear simultaneous equations relating the autocorrelation function of the input time series to the cross-correlation functions of various orders between the input and output time series. This solution, which involves the inversion of a Toeplitz matrix and some vector multiplication, gives rise to the system kernels.

As mentioned earlier, an artificial neural network can be viewed as an optimisation technique that allows a mapping to be learnt between the input and output spaces represented by the network. This learning is achieved by minimising the error between the network's actual output and its required output over a range of training examples. In this specific case the network, of the form shown in Fig. 1, is required to learn the mapping that predicts the current system ($y(t)$) given the last 50 inputs to the system ($u(t) \ldots u(t-50)$). The training examples used in this experiment are obtained from the input and output time series given to, and obtained from, the system under investigation. A window of 50 successive points of the input trace ($u(t) \ldots u(t-50)$) and the corresponding output value ($y(t)$) comprise one training example. A set of these training examples was constructed by considering all possible windows of 50 points over the whole input trace, producing a total of 3950 training examples. Only 3950 training examples are produced since the initial 50 output points cannot be used because the previous 50 inputs are not known.

In the results presented here the network was trained with the training data set described above, using the back-propagation algorithm with momentum (Rumelhart et al. 1986) with a learn rate $\mu = 0.05$ and a momentum scale factor of $\alpha = 0.5$. The network was presented with examples, drawn at random from the whole training set, many times until the sum squared error between the network's required output and its actual response stopped reducing. This typically took between 1000 and 2000 cycles. After the network training procedure had converged, the first- and second-order Volterra kernels were calculated using (13) and (14). Since in this experiment hyperbolic tangent output functions were used, the a_{ji} values in (13) and (14) were calculated using (20).

4.2 Experimental results

The first-order kernels, which should be identically zero, are shown in Fig. 5. Note that since a theoretical system has been used for the experiment, the system's input and output have no units, and hence the kernels have no units. In a real system, however, the kernels do have units, and so all the kernel graphs shown have the ordinate labelled 'kernel units'. The second-order kernels are shown in Figs. 6 and 7. These figures clearly show the

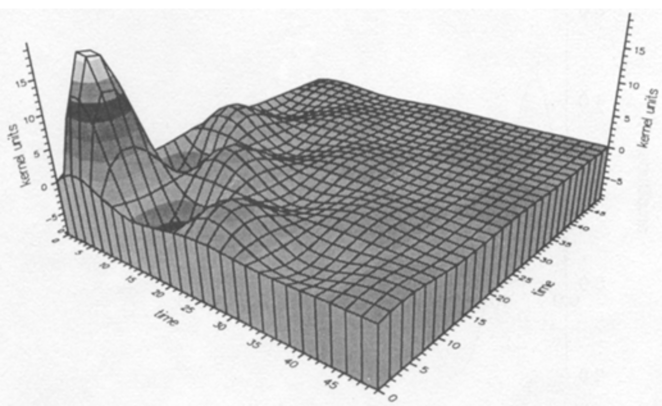

Fig. 4. Ideal second-order kernel for the Wiener cascade model

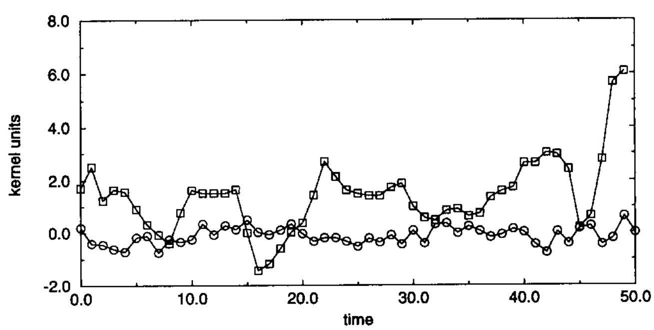

Fig. 5. First-order kernels of the Wiener cascade model calculated using Toeplitz matrix inversion (□) and the neural network method (○)

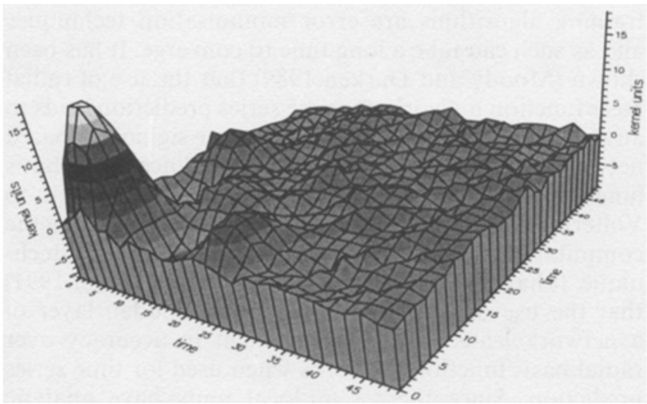

Fig. 6. Second-order kernel for the Wiener cascade model calculated using Toeplitz matrix inversion

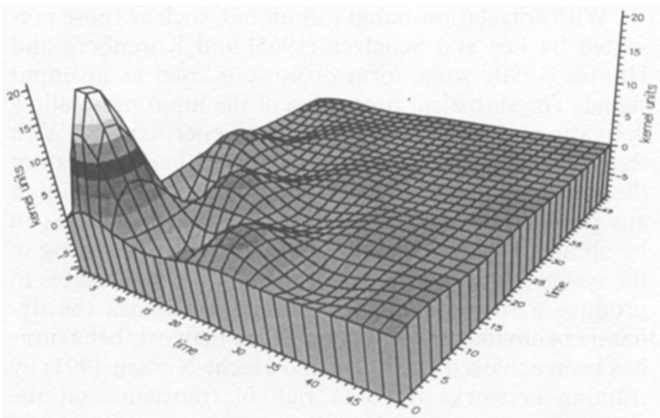

Fig. 7. Second-order kernel for the Wiener cascade calculated using the neural network-based method

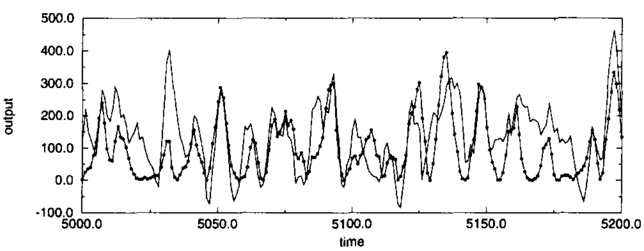

Fig. 8. The output from the system (●) and the kernel calculated using Toeplitz matrix inversion (*line*)

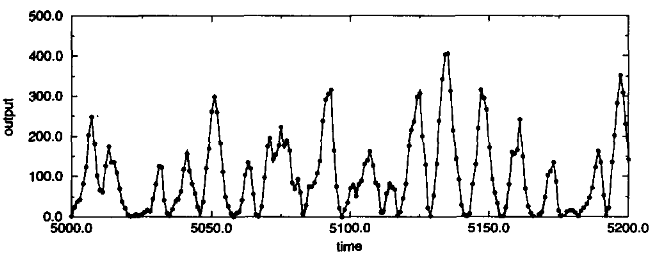

Fig. 9. The output from the system (●) and the kernel calculated using the neural network-based technique (*line*). The two graphs are highly coincident, although slight differences can be detected at some minima and maxima

improved accuracy of the neural network technique over the Toeplitz matrix technique for this specific example.

Although the kernels shown in Figs. 5–7 could be numerically compared to the analytic kernels, a more useful and acid test of the relative accuracy of the kernel calculation techniques is to use the second-order kernels to predict the system's response to a novel stimulus. Figures 8 and 9 show the predicted output for an unseen input using kernels calculated using Toeplitz matrix inversion and the neural network method, respectively. In addition, both graphs show the actual system output (•) for the same input. Clearly, the neural network produces the better prediction, which hardly differs from the actual system output, as seen by the high coincidence of the two graphs in Fig. 9.

5 Discussion

This paper is concerned with the Volterra series representation of non-linear systems. Volterra series are able to represent any analytic, time invariant system (Volterra 1959) and are the general solution of non-linear differential equation models of dynamic systems (Fliess et al. 1983). However, no general methods exist for either constructing differential equation models of non-linear systems or for numerically obtaining the Volterra kernels if the order of the system is unknown a priori.

Artificial neural networks of a certain architecture (depicted in Fig. 1) have been shown to possess general approximation abilities (Cybenko 1989; Funahashi 1989; Hornik et al. 1989) and have been used to model dynamic non-linear systems (Hudson et al. 1990; Weigend et al. 1990; Chakraborty et al. 1992). Neural networks can be viewed as approximation techniques in which an error between the required network output (defined by a number of training examples) and its actual output is minimised by alteration of the network's internal parameters. This paper proves how a network of a certain architecture is equivalent to finite memory, discrete Volterra series. This proof leads directly to equations for the Volterra kernels in terms of network parameters and hence to a method of calculating the Volterra kernels of a system of unknown order.

The technique is illustrated and compared to another kernel calculation technique (Toeplitz matrix inversion of Korenberg and Hunter 1990) by using both methods to calculate the Volterra kernels for a non-linear dynamic system described by a Wiener cascade. An advantage of using such a test system is that it enables the analytic kernels to be calculated. In the original paper describing the Toeplitz matrix technique, the method was shown to be far superior, in terms of accuracy, than the cross-correlation method of Lee and Schetzen (1965). The experiment reported in this paper demonstrates, for the specific system studied, the greater accuracy of the neural network method over the Toeplitz matrix technique, and hence over the cross-correlation-based method of Lee and Schetzen.

There are two possible reasons for this improvement in accuracy. First, the equations for the kernels (12–15) can be evaluated exactly. Thus, if a network can be trained using the available data to produce an adequate approximation to the systems input-output mapping, then an approximation to the Volterra kernels can be calculated exactly. Any errors in the calculation of the kernels are not due to assumptions made in the derivation of the kernel formula being violated but rather to either a poor approximation being produced or the training data not covering the whole space of possible input-output relationships, i.e. the trained network does not adequately represent the systems' behavior. Conversely, the Wiener calculation techniques assume frequently violated statistical properties of the input signal in the derivation of the kernel formula. For example, with the cross-correlation technique (Lee and Schetzen 1965) the equations for the kernels assume that the input signal is infinitely long white noise. This signal is not realisable, and so errors are introduced into the kernels when they are calculated in this way. The second reason for improved accuracy is due to the weakening of the error criteria with the Wiener representation (Palm and Pöppel 1985). The neural network training algorithm minimises the error on a point-by-point basis (L_2 norm), whereas the Wiener representation produces an approximation based on the average error (L_∞ norm). Thus, a more accurate approximation to the system can be generated by the neural network than by a Wiener series.

The experiment described in this paper compares the newly developed neural network technique with that of Toeplitz matrix inversion. In general, however, this comparison is not possible since the Toeplitz matrix method produces the Wiener kernels of a system, whereas the neural network method produces the Volterra kernels. The comparison was only possible because the system used was of second order, which means the Volterra and Wiener kernels are identical (Marmarelis and Marmarelis 1978). Thus, the comparative statements made above concerning accuracy cannot be generalised to systems of higher order. However, the method derived in this paper allows the calculation of the Volterra kernels of systems of any, unknown order, and the experimental results presented demonstrate the technique's accuracy when applied to a specific system. In addition, the technique has been applied successfully to real and modelled neurobiological systems of higher order (Hearne et al. 1993), with close agreement being shown between kernels obtained analytically and numerically (Hearne et al. 1994).

The analysis presented in this paper has been for feed-forward networks with sigmoidal output functions in their hidden nodes. This is only a specific example of the more general case. It can be shown (Wray 1992) that radial basis function networks can be considered equivalent to producing polynomials in their inputs. This is, in fact, true of networks using any analytic function as output functions of their hidden nodes. This means that such networks are equivalent to discrete, finite memory Volterra series and that equations for the Volterra kernels can be derived in terms of the network parameters. This may have implications in overcoming a drawback of this technique: its computational expense. Network training algorithms are error minimisation techniques and as such can take a long time to converge. It has been shown (Moody and Darken 1989) that the use of radial basis function networks for time series prediction leads to an improvement in training speed over sigmoidal-based networks trained by back-propagation. Since radial basis function networks can be shown to be equivalent to Volterra series, the use of such networks may reduce the computational overheads of this kernel calculation technique. It has also been shown (Hartman and Keeler 1991) that the use of semi-local units in the hidden layer of a network leads to an improvement in accuracy over radial basis function networks when used for time series prediction. Since these semi-local units have analytic output functions, networks using such units are also equivalent to a Volterra series. The use of these networks may lead to a lower error after training and hence to more accurate kernels.

With correlation-based techniques, such as those presented by Lee and Schetzen (1965) and Korenberg and Hunter (1990), some form of noise is used as an input signal. The statistical properties of the input noise allow derivation of the formula for the Wiener kernels. With the neural network technique presented here, this is not the case since the kernel formula does not depend upon any property of the input signal used. As long as data can be obtained that represent the input-output mapping of the system, the network training algorithm will be able to produce a trained network that approximates the dynamic behaviour of the system. This network behaviour has been achieved (Lambert and Hecht-Nielsen 1991) by training networks on data rich in transients, on the assumption that enough different transients will represent the overall dynamic behaviour of the system. Thus, the technique can be applied in situations where noise input may be impractical.

Although this paper has presented a method for calculating the Volterra kernels for an arbitrary system, a number of problems inherent to the Volterra and Wiener series still exist. The first concerns the Volterra series and its range of applicability. The method presented in this paper does not remove the constraint that the Volterra approximation cannot be applied to non-analytic systems. The second problem concerns the convergence and truncation of the Volterra series. Since the Volterra functionals are not independent of each other, truncation can affect the approximation ability of the Volterra series quite badly, especially if the convergence properties of the underlying polynomial approximation are not known. A third problem is inherent to both Volterra and Wiener series approximations, and indeed to all methods of system identification, and is concerned with the test signal used. Even though the method presented in this paper does not require a test signal of Gaussian noise, the signal (or signals) used must be such that the input-output mapping of the system under investigation can be adequately described by the data obtained using those inputs. Even if noise is used as an input signal, inferences about system behaviour can only be made over the bandwidth of the noise used. A fourth problem, again inherent to both functional techniques, is that they require the system to be time-invariant. This requires the time scale of the system investigation to be chosen carefully, to ensure stationarity, although an extension of the Volterra formulation to include kernels that vary in time has been provided by Schetzen (1980) and Rugh (1981). A fifth problem which is implicit to both the Volterra and Wiener series is that they can only be used for system identification. Although they can give insight into system dynamics, they reveal nothing about the underlying mechanisms leading to the observed behaviour. The upshot of these limitations is that the Volterra and Wiener techniques cannot be used blindly; properties such as analyticity, time invariance and system bandwidth all have to be considered before application of the functional approaches to biological system identification.

Acknowledgements. This work was supported in part by grants from British Telecom and the Science and Engineering Research Council, UK. J. Wray was supported by the Hunter Memorial Scholarship awarded by the Medical Faculty of the University of Newcastle upon Tyne, UK. We would like to thank Phil Hearne for his many useful discussions.

References

Aertsen AMHJ, Johannesma PIM (1981) The spectro-temporal receptive field: a functional characteristic of auditory neurons. Biol Cybern 42:133–143

Boer E de, Kuyper P (1968) Triggered correlation. IEEE Trans Biomed Eng 15:169–179

Chakraborty K, Mehrotra K, Mohan CK, Ranka S (1992) Forecasting the behavior of multivariate time series using neural networks. Neural Networks 5:961–970

Cheney EW (1982) Introduction to approximation theory, 2nd edn. Chelsea, New York

Cybenko G (1989) Approximation by superpositions of sigmoidal functions. Math Control Signals Syst 2:303–314

Emerson RC, Korenberg MJ, Citron MC (1992) Identification of complex-cell intensive nonlinearities in a cascade model of cat visual cortex. Biol Cybern 66:291–300

Fliess M, Lamnabhi M, Lamnabhi-Lagarrigue F (1983) An algebraic approach to nonlinear functional expansions. IEEE Trans Circuits Syst 30:554–570

Funahashi K (1989) On the approximate realization of continuous mapping by neural networks. Neural Networks 2:183–192

Hartman E, Keeler JD (1991) Predicting the future: advantages of semilocal units. Neural Comput 3:566–578

Hearne PG, Wray J, Sanders DJ, Agar E, Green GGR (1993) The neurone as a nonlinear system: a single compartment study. In: Eeckman FH, Bower JM (eds) Computation and neural systems. Kluwer, Norwell, pp. 19–23

Hearne PG, Manchanda S, Janahmadi M, Thompson IM, Wray J, Sanders DJ, Green GGR (1994) Solutions to Hodgkin-Huxley equations: functional analysis of a molluscan neurone. In: Eeckman FH, Bower JM (eds) Computation and neural systems. II. Kluwer, Norwell

Hertz J, Krogh A, Palmer R (1990) Introduction to the theory of neural computation. Addison-Wesley, Redwood

Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Networks 2:359–366

Hudson JL, Kube M, Adomatis RA, Kevrekidis IG, Lapedes AS, Farber RM (1990) Nonlinear signal processing and system identification: applications to time series from electrochemical reactions. Chem Eng Sci 45:2075–2081

Knuth DE (1981) Seminumerical algorithms. (Art of computer programming, Vol 2). Addison-Wesley, Redwood

Korenberg MJ, Hunter IW (1990) The identification of nonlinear biological systems: Wiener kernel approaches. Ann Biomed Eng 18:629–654

Lambert JM, Hecht-Nielsen R (1991) Application of feedforward and recurrent neural networks to chemical plant predictive modeling. In: Proceedings of International Joint Conference on Neural Networks Seattle, pp 1373–1378

Lee YW, Schetzen M (1965) Measurement of the Wiener kernels of a nonlinear system by cross-correlation. Int J Control 2:237–254

Lippmann RP (1987) An introduction to computing with neural nets. IEEE ASSP Magazine April:4–22

Marmarelis PZ, Marmarelis VZ (1978) Analysis of physiological systems: the white noise approach. Plenum Press, New York

Marmarelis PZ, Naka K (1972) White-noise analysis of a neurone chain: an application of the Wiener theory. Science 175:1276–1278

Mizunami M, Tateda H, Naka K (1986) Dynamics of cockroach ocellar neurons. J Gen Physiol 88:275–292

Moody J, Darken CJ (1989) Fast learning in networks of locally-tuned processing units. Neural Comput 1:281–294

Nabet B, Pinter RB (1992) Multiplicative inhibition and Volterra series expansion. In: Pinter RB, Nabet B (eds) Nonlinear vision: determination of neural receptive fields, function and networks. CRC Press, Boca Raton, pp 475–492

Palm G, Poggio T (1977) The Volterra representation and the Wiener expansion: validity and pitfalls. SIAM J Appl Math 33:195–216

Palm G, Pöppel B (1985) Volterra representation and Wiener-like identification of nonlinear systems: scope and limitations. Q Rev Biophys 18:135–164

Poggio T, Reichardt W (1973) Considerations on models of movement detection. Kybernetik 13:223–227

Rugh WJ (1981) Nonlinear system theory: the Volterra/Wiener approach. Johns Hopkins University Press, Baltimore

Rumelhart DE, McCelland JL, PDP Research Group (1986) Parallel distributed processing: explorations in the microstructure of cognition, Vol 1. MIT Press, Cambridge, Mass

Schetzen M (1965) Measurement of the kernels of a non-linear system of finite order. Int J Control 1:251–263

Schetzen M (1980) The Volterra and Wiener theories of nonlinear systems. Wiley, New York

Volterra V (1959) Theory of functionals and integral and integro-differential equations. Dover, New York

Waibel A, Hanazawa T, Hinton G, Shikano K, Lang K (1989) Phoneme recognition using time-delay neural networks. IEEE Trans Acoustics Speech Signal Process 37:328–339

Weigend AS, Huberman BA, Rumelhart DE (1990) Predicting the future: a connectionist approach. Int J Neural Syst 1:193–209

Wiener N (1958) Nonlinear problems in random theory. MIT Press, Cambridge, Mass

Wolfram S (1988) Mathematica: a system for doing mathematics by computer. Addison-Wesley, Redwood

Wray J (1992) Theory and applications of neural networks. PhD thesis, University of Newcastle upon Tyne, UK

Wray J, Green GGR (1991) Analysis of networks that have learnt control problems. In: Proceedings of IEEE Control-91. Herriot-Watt, UK, pp 261–265