International Journal of Control
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713393989

Properties of neural networks with applications to modelling non-linear dynamical systems

S. A. Billings [a]; H. B. Jamaluddin [a]; S. Chen [b]
[a] Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, SI, U.K
[b] Department of Electrical Engineering, University of Edinburgh, Edinburgh, EH, U.K

Online Publication Date: 01 January 1992
To cite this Article: Billings, S. A., Jamaluddin, H. B. and Chen, S. (1992) 'Properties of neural networks with applications to modelling non-linear dynamical systems', International Journal of Control, 55:1, 193 - 224
To link to this article: DOI: 10.1080/00207179208934232
URL: http://dx.doi.org/10.1080/00207179208934232

INT. J. CONTROL, 1992, VOL. 55, NO. 1, 193–224

Properties of neural networks with applications to modelling non-linear dynamical systems

S. A. BILLINGS†, H. B. JAMALUDDIN† and S. CHEN‡

Properties of neural network performance are investigated by studying the modelling of non-linear dynamical systems. Network complexity, node selection, prediction and the effects of noise are studied and some new metrics of performance are introduced. The results are illustrated with both simulated and industrial examples.

1. Introduction

Neural networks have recently become an enormously fashionable area of research and have been applied in many diverse areas such as speech processing, pattern recognition and non-linear model fitting. In most of these cases the neural network is trained to represent the data set using some learning algorithm. Ideally the weights which define the strength of connection between the neurons in the network should converge to yield a neural network architecture which can emulate the mechanisms which produced the data set. This process clearly involves learning a mathematical description of the system and can therefore be studied as a system identification problem. The advantage of this interpretation is that all the fundamental results of estimation theory which have been developed over many decades (Nahi 1969, Goodwin and Payne 1977, Ljung and Söderström 1983) can be employed to study rigorously both the properties and the performance of neural networks.

Previous studies (Lapedes and Farber 1987, Narendra and Parthasarathy 1990, Uhrig and Guo, 1989, Bhat *et al.* 1990) have successfully demonstrated the power of neural networks when employed to model complex non-linear mechanisms, but most of these have not used the results of estimation theory to interpret the results obtained. Although system identification has traditionally been based on estimating the coefficients of linear models using variants of least squares or maximum likelihood algorithms (Goodwin and Payne 1977, Ljung and Söderström 1983) the fundamental concepts of estimation, bias, prediction and model validation are applicable to a much wider class of problems including multilayered networks.

The present study is an attempt to introduce to the neural network community some of the basic concepts from estimation theory and to show how these can be used to measure interpret and improve network performance. Questions are addressed, such as whether it is important to consider the assignment of input nodes, whether network performance improves with increasing network complexity, whether it is better to assign the network nodes as lagged inputs only or as a

Received 5 March 1991.

† Department of Automatic Control and Systems Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, U.K.

‡ Department of Electrical Engineering, University of Edinburgh, Mayfield Road, Edinburgh EH9 3JL, U.K.

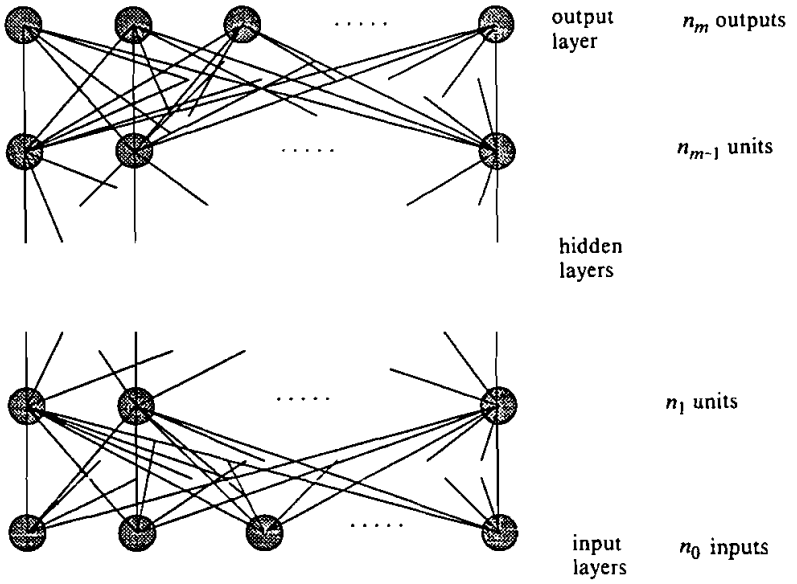

Figure 1. A multilayered neural network.

combination of lagged inputs and lagged outputs, whether a one-step-ahead prediction is a sufficient metric of network performance, whether model validity tests are useful, whether it is possible to detect when the network is sufficient to represent the data set how noise on the measurements affects network performance, the question of bias, etc. Throughout, simulated non-linear models and real data sets are used to illustrate the ideas. The recursive prediction error learning algorithm (Chen *et al.* 1990 a, b, Billings *et al.* 1991) is used to update the network weights simply because this has much better convergence properties compared with the back-propagation algorithm. It is important to note, however, that most of the results should be applicable for alternative learning mechanisms and should apply to the range of problems to which neural networks have been applied and not just to modelling non-linear dynamical systems.

2. Feedforward neural networks

2.1. *Neural network architectures*

A neural network is a massively parallel, interconnected network of elementary units called neurons. Inputs to each neuron are combined and the neuron produces an output if the sum of the inputs exceeds an internal threshold value. A feedforward neural network is made up of layers of neurons between the input and output layers, called hidden layers, with connections between neurons of intermediate layers. The general structure of a multilayered neural network is illustrated in Fig. 1. The input layer usually acts as an input data holder which distributes inputs to the first hidden layer. The signals flow from the input layer to the output layer. A basic element of the network, the ith neuron in the lth layer is shown in Fig. 2. A neuron performs two functions. These functions are the combining function and the activation function.

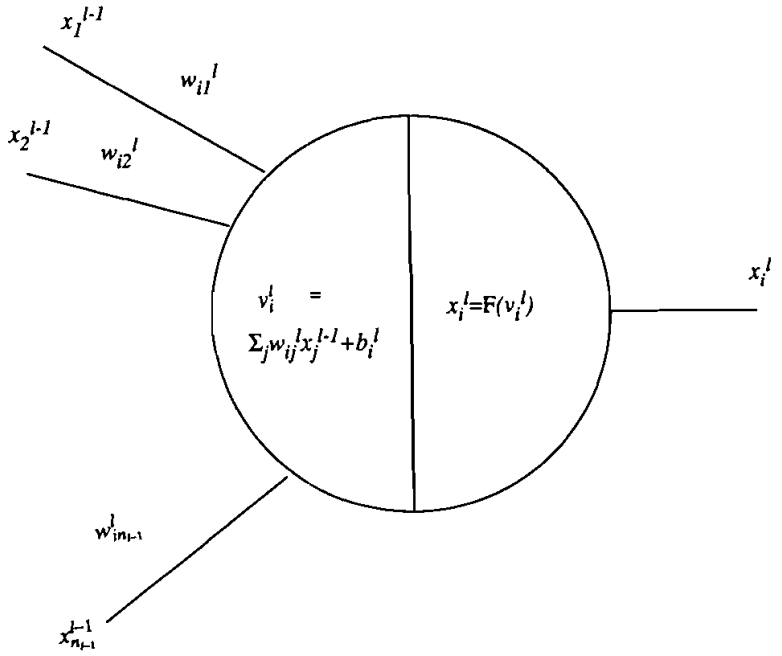

Figure 2. A hidden neuron i of layer l.

The combining function produces an activation for the neuron, $v_i^l(t)$, defined as

$$v_i^l(t) = \sum_{j=1}^{n_{l-1}} w_{ij}^l x_j^{l-1}(t) + b_i^l \tag{1}$$

where w_{ij}^l is the weight connection between the jth neuron of the $(l-1)$th layer and the ith neuron of the lth layer, b_i^l is the threshold of the neuron and n_{l-1} is the number of neurons in the $(l-1)$th layer. The activation function performs a non-linear transformation to give the output, $x_i^l(t)$

$$x_i^l(t) = F(v_i^l(t)) \tag{2}$$

where $F(\cdot)$ is called the non-linear transformation or activation function.

The inputs propagate forward through the network. By combining (1) and (2) the output of each neuron can be expressed as

$$x_i^l(t) = F\left(\sum_{j=1}^{n_{l-1}} w_{ij}^l x_j^{l-1}(t) + b_i^l\right) \tag{3}$$

where $x_i^l(t)$ is the output of the ith neuron in the lth layer for $i = 1, ..., n_l$ and $l = 1, ..., m$. With these definitions the mth layer becomes the output layer and the input layer can be labelled as the zero layer as illustrated in Fig. 1. Thus n_0 and n_m refer to the numbers of network inputs and outputs respectively. For convenience $x_i^0(t)$ and $x_i^m(t)$ will often be denoted as $x_i(t)$ and $\hat{y}_i(t)$.

For the neurons in the hidden layers, the activation function is often chosen to be

$$F(v(t)) = \frac{1}{1 + \exp(-v(t))} \tag{4}$$

In system modelling applications it is common for the dynamic range of output data to be greater than 1, the activation function of the output nodes is therefore chosen to be linear. Thus, the ith output node performs a weighted sum of its inputs as follows

$$\hat{y}_i(t) = \sum_{j=1}^{n_{m-1}} w_{ij}^m x_j^{m-1}(t) \tag{5}$$

Based on the results of Cybenko (1989) and Funahashi (1989), only one hidden-layer networks are considered in the present study, that is $m = 2$. All the results will be presented for cause and effect systems and the single-input single-output (SISO) case only. This has been done to enhance the clarity and interpretation of the results. It is important to emphasize however that all the results are applicable to the multi-input multi-output (MIMO) case with an obvious extension of notation. The SISO restriction means that only one output neuron, that is $n_2 = 1$ is required and the index i in (5) can be removed. With these restrictions the output of the network is therefore given by

$$\hat{y}(t) = \sum_{i=1}^{n_1} w_i^2 x_i^1(t) = \sum_{i=1}^{n_1} w_i^2 F\left(\sum_{j=1}^{n_0} w_{ij}^1 x_j(t) + b_j^1 \right) \tag{6}$$

where $x(t) = [x_1(t) \ \ldots \ x_{n_0}(t)]^T$ is the input vector to the network. The elements of the input vector are the input variables to be assigned at the input nodes of the network. The ws and bs are the parameters to be estimated and these form the elements of Θ, the parameter vector defined as $\Theta = [\theta_1 \ \theta_2 \ \ldots \ \theta_{n_\theta}]^T$ where $n_\theta = n_1 + n_1(n_0 + 1)$ is the number of unknown parameters. The objective of training the neural network model is to determine Θ such that $\hat{y}(t)$ is as close to the desired output $y(t)$ as possible. The discrepancy between $y(t)$ and $\hat{y}(t)$

$$\varepsilon(t) = y(t) - \hat{y}(t) \tag{7}$$

is called the prediction error or the residuals.

2.2. *Training algorithms*

Resurgence of research into the use and application of multilayered neural networks is due in part to the development of the back-propagation learning algorithm. The back-propagation algorithm was first proposed by Werbos in his Ph.D. thesis and further developed by Rumelhart and McClelland (1986). Back-propagation is a steepest descent type algorithm where the weight connection between the jth neuron of the $(l-1)$th layer, and the ith neuron of the lth layer, and the threshold of the ith neuron of the lth layer are respectively updated according to

$$w_{ij}^l(t) = w_{ij}^l(t-1) + \Delta w_{ij}^l(t) \tag{8 a}$$

$$b_i^l(t) = b_i^l(t-1) + \Delta b_i^l(t) \tag{8 b}$$

with the increment $\Delta w_{ij}^l(t)$ and $\Delta b_i^l(t)$ given by

$$\Delta w_{ij}^l(t) = \eta_w \rho_i^l(t) x_j^{l-1}(t) + \alpha_w \Delta w_{ij}^l(t-1) \tag{9 a}$$

$$\Delta b_i^l(t) = \eta_b \rho_i^l(t) + \alpha_b \Delta b_i^l(t-1) \tag{9 b}$$

where the subscripts w and b represent the weight and threshold respectively, α_w and α_b are momentum constants which determine the influence of past parameter changes on the current direction of movement in the parameter space, η_w and η_b

represent the learning rates and $\rho_i^l(t)$ is the error signal of the ith neuron of the lth layer which is back-propagated in the network. Since the activation function of the output neuron is linear, the error signal at the output nodes is

$$\rho_i^m(t) = y_i(t) - \hat{y}_i(t) \tag{10 a}$$

and for the neurons in the hidden layer

$$\rho_i^l(t) = F'(v_i^l(t)) \sum_k \rho_k^{l+1}(t) w_{ki}^{l+1}(t-1) \quad l = m-1, ..., 2, 1 \tag{10 b}$$

where $F'(v)$ is the first derivative of $F(v)$ with respect to v.

Similar to other steepest descent-type algorithms, the back-propagation algorithm suffers from a slow convergence rate, it may become trapped at local minima and can be sensitive to user selectable parameters (Brady *et al.* 1989, Sutton 1986, Billings *et al.* 1991). Some of these properties can be improved by modifying the algorithm (Leonard and Kramer 1990, Schmidhuber 1989).

A new recursive prediction error (RPE) algorithm which is a Gauss–Newton type algorithm has been proposed (Chen *et al.* 1990 a, Billings *et al.* 1991) as an alternative to back-propagation. The RPE algorithm is given by

$$\varepsilon(t) = y(t) - \hat{y}(t) \tag{11}$$

$$R(t) = R(t-1) + \gamma(t)[\boldsymbol{\Psi}(t)\boldsymbol{\Lambda}^{-1}\boldsymbol{\Psi}^{\mathrm{T}}(t) + \beta I - R(t-1)] \tag{12}$$

$$\hat{\boldsymbol{\Theta}}(t) = \hat{\boldsymbol{\Theta}}(t-1) + \gamma(t)R^{-1}(t)\boldsymbol{\Psi}(t)\boldsymbol{\Lambda}^{-1}\varepsilon(t) \tag{13}$$

where $\hat{\boldsymbol{\Theta}}(t)$ is the estimate of the parameter vector, β is a non-negative small scalar, $\gamma(t)$ is the gain at sample t and $\boldsymbol{\Psi}(t)$ is the first derivative of the one-step-ahead prediction of the model with respect to the estimated parameters, that is

$$\boldsymbol{\Psi}(t, \boldsymbol{\Theta}) = \left[\frac{d\hat{y}(t, \boldsymbol{\Theta})}{d\boldsymbol{\Theta}}\right]^{\mathrm{T}} \tag{14}$$

To avoid inversion of R at every iteration, in practice (12) and (13) are implemented in the equivalent form (with $\beta = 0$ and $\boldsymbol{\Lambda} = I$)

$$P(t) = \frac{1}{\lambda(t)}\{P(t-1) - P(t-1)\boldsymbol{\Psi}(t)[\lambda(t)I + \boldsymbol{\Psi}^{\mathrm{T}}(t)P(t-1)\boldsymbol{\Psi}(t)]^{-1}\boldsymbol{\Psi}^{\mathrm{T}}(t)P(t-1)\} \tag{15}$$

$$\hat{\boldsymbol{\Theta}}(t) = \hat{\boldsymbol{\Theta}}(t-1) + P(t)\boldsymbol{\Psi}(t)\varepsilon(t) \tag{16}$$

where

$$P(t) = \gamma(t)R^{-1}(t), \quad \lambda(t) = \frac{\gamma(t-1)}{\gamma(t)}(1 - \gamma(t))$$

and $\lambda(t)$ is called the forgetting factor.

The derivation, implementation and properties of this algorithm have been reported in Billings *et al.* (1991). The RPE algorithm has superior convergence properties compared with back-propagation and will be used to train all the networks in the present study.

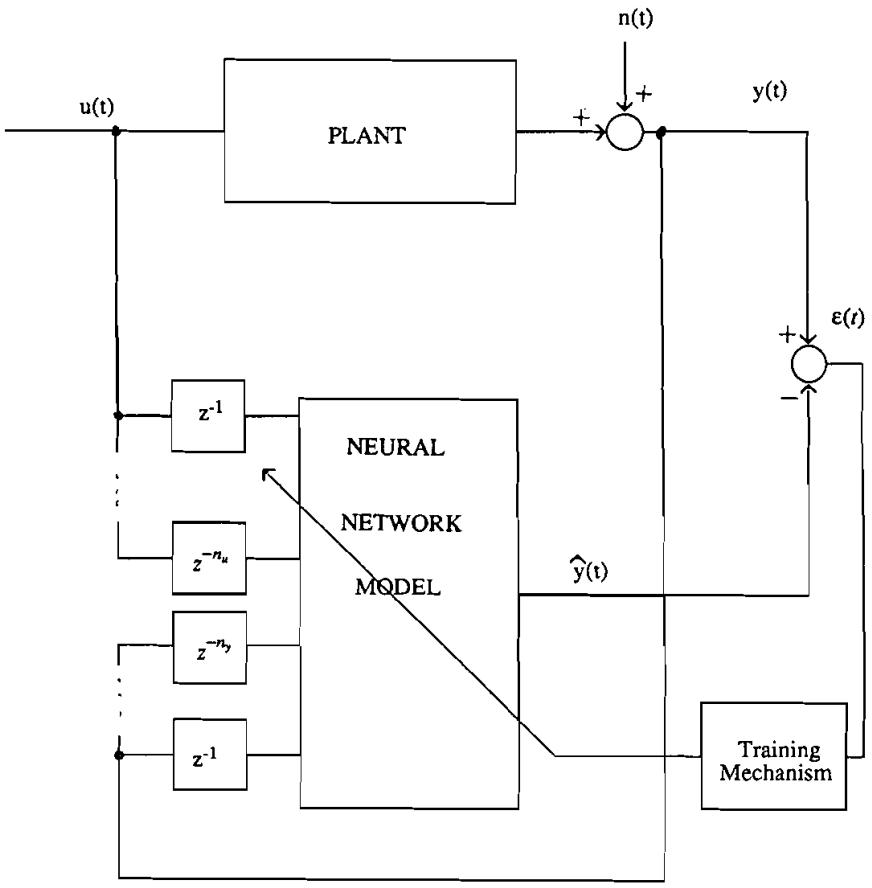

Figure 3. Non-linear system identification using a neural network model.

3. Properties of neural networks

3.1. *Network expansions*

If a neural network model is employed to represent a dynamical system it is important to establish the form of expansion that the network provides because this will determine if it is necessary to consider the correct assignment of network input nodes. The number of input nodes specifies the dimension of the network inputs. In the system identification context, the network input vector consists of lagged system inputs and outputs as illustrated in Fig. 3. The assignment of network input nodes then becomes the task of selecting the correct lag orders of the system input and output to form the network input vector.

The investigation of this property is straightforward and consists of expanding the activation function of the hidden nodes. Assuming for example that the activation function of each neuron is a sigmoidal function as defined in (4), this can be expressed as

$$F(v(t)) = \frac{1}{1+\exp(-v(t))} = \tfrac{1}{2} + \tfrac{1}{4}v(t) - \tfrac{1}{48}v^3(t) + \tfrac{1}{480}v^5(t) + \ldots \qquad (17)$$

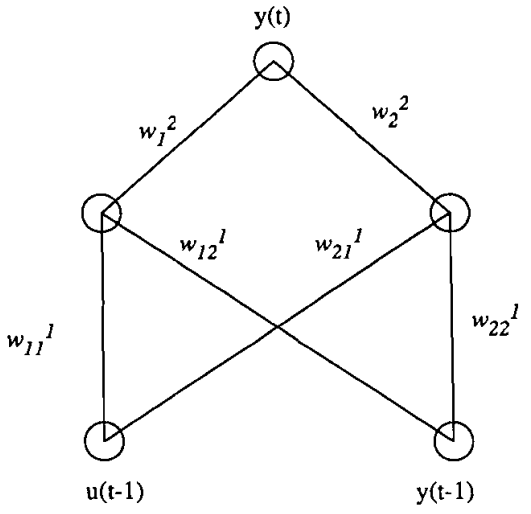

Figure 4. A one-hidden-layer network with two input nodes and two hidden nodes.

The input–output relationship for the one-hidden-layered network with two input nodes and two hidden nodes illustrated in Fig. 4, for example, is given by

$$\begin{aligned} y(t) = {} & w_1^2 F(w_{11}^1 u(t-1) + w_{12}^1 y(t-1) + b_1^1) \\ & + w_2^2 F(w_{21}^1 u(t-1) + w_{22}^1 y(t-1) + b_2^1) \end{aligned} \tag{18}$$

Expanding (18) using (17) yields

$$\begin{aligned} y(t) = {} & w_1^2[\tfrac{1}{2} + \tfrac{1}{4}(w_{11}^1 u(t-1) + w_{12}^1 y(t-1) + b_1^1) \\ & - \tfrac{1}{48}(w_{11}^1 u(t-1) + w_{12}^1 y(t-1) + b_1^1)^3 + \ldots] \\ & + w_2^2[\tfrac{1}{2} + \tfrac{1}{4}(w_{21}^1 u(t-1) + w_{22}^1 y(t-1) + b_2^1) \\ & - \tfrac{1}{48}(w_{21}^1 u(t-1) + w_{22}^1 y(t-1) + b_2^1)^3 + \ldots] \end{aligned} \tag{19}$$

Collecting terms gives the representation

$$\begin{aligned} y(t) = {} & a_0 + a_1 u(t-1) + a_2 y(t-1) + a_3 u^2(t-1) + a_4 y^2(t-1) + a_5 u(t-1)y(t-1) \\ & + a_6 u^3(t-1) + a_7 y^3(t-1) + a_8 u^2(t-1)y(t-1) + a_9 y^2(t-1)u(t-1) + \ldots \end{aligned} \tag{20}$$

where the as are parameters which are a function of the ws and bs. It is easy to see how this result would generalize to higher order network expansions and alternative activation functions. Since a_0, a constant term, exists in the expansion of (20), this shows that multilayered neural networks include a mean value in the system representation. Constant terms, which are important in non-linear modelling, may not, therefore, need to be specified as input nodes. As expected, (20) also shows that the network does not generate components of higher order lagged system inputs and outputs which are not specified in the network input nodes, because the output of the neural network is just a static non-linear expansion of the input nodes. This is illustrated schematically in Fig. 3. If insufficient lagged $u(t)$ and $y(t)$ are assigned as input nodes the network cannot generate the missing dynamic terms and this implies

that an inadequate representation will result. It is therefore very important both to consider the selection of network input nodes and to develop methods which detect when network performance is limited by inappropriate node specification. Otherwise a poor model fit, which was caused by a failure to include adequate dynamics in the specification of input nodes, may be incorrectly interpreted as some other effect.

3.2. *Model validation*

Model validity tests are procedures designed to detect the inadequacy of a fitted model. In simulation it is easy to show that given the correct input node assignments and a sufficient number of hidden nodes the network will produce a good model of the system. But in practice the model of the true system will be unknown and the detection of an inadequate fit is therefore more challenging. A poor fit may be caused by incorrect input node assignments, noisy data, a mistake in the programme code, insufficient hidden nodes or several other effects. Whatever the cause the aim of the model validation is to indicate to the analyst that the model fit is incorrect.

If a model of a system is adequate then the residuals or prediction errors $\varepsilon(t)$ should be unpredictable from all linear and non-linear combinations of past inputs and outputs. The derivation of simple tests which can detect these conditions is complex but it can be shown that (Billings and Voon 1986) the following conditions should hold

$$\left.\begin{aligned}
\phi_{\varepsilon\varepsilon}(\tau) &= E[\varepsilon(t-\tau)\varepsilon(t)] = \delta(\tau) \\
\phi_{u\varepsilon}(\tau) &= E[u(t-\tau)\varepsilon(t)] = 0, \quad \forall\,\tau \\
\phi_{u^2\varepsilon}(\tau) &= E[(u^2(t-\tau) - \bar{u}^2(t))\varepsilon(t)] = 0, \quad \forall\tau \\
\phi_{u^2\varepsilon^2}(\tau) &= E[(u^2(t-\tau) - \bar{u}^2(t))\varepsilon^2(t)] = 0, \quad \forall\,\tau \\
\phi_{\varepsilon(\varepsilon u)}(\tau) &= E[\varepsilon(t)\varepsilon(t-1-\tau)u(t-1-\tau)] = 0, \quad \tau \geqslant 0
\end{aligned}\right\} \tag{21}$$

These tests are applicable to the most general case where both process and noise models are estimated (see also § 3.4). In practice normalized correlations are computed. The sampled correlation function between two sequences $\psi_1(t)$ and $\psi_2(t)$ is given by

$$\hat{\phi}_{\psi_1\psi_2}(\tau) = \frac{\sum_{t=1}^{N-\tau} \psi_1(t)\psi_2(t+\tau)}{\left[\sum_{t=1}^{N} \psi_1^2(t) \sum_{t=1}^{N} \psi_2^2(t)\right]^{1/2}} \tag{22}$$

Normalization ensures that all the correlation functions lie in the range $-1 \leqslant \hat{\phi}_{\psi_1\psi_2}(\tau) \leqslant 1$ irrespective of the signal strengths. The correlations will never be exactly zero for all lags and the 95% confidence bands defined as $1{\cdot}96/\sqrt{N}$ are used to indicate if the estimated correlations are significant or not, where N is the data length.

The tests in (21) were derived for the class of analytic non-linear systems (Billings and Voon 1983). Since neural networks may be used to approximate a wider class of systems, it is impossible to definitely state that the correlation tests will always detect all possible non-linear terms that may be part of a neural network model. Whilst a theoretical analysis of the problem would be very complex, all the simulation results do tend to indicate that the tests are a very powerful aid in neural network modelling.

The model validity tests of (21) are simple to compute and will be used throughout this study to detect various defects in network models. Consider the input assignment problem to illustrate these ideas. The system referred to as S1 and defined by

$$\left.\begin{aligned} \bar{y}(t) &= \frac{0{\cdot}4}{1+\exp\{-(0{\cdot}3u(t-1)+0{\cdot}7u(t-2)+0{\cdot}5\bar{y}(t-1)+0{\cdot}1)\}} \\ n(t) &= e(t)+0{\cdot}6e(t-1) \\ y(t) &= \bar{y}(t)+n(t) \end{aligned}\right\} \quad (23)$$

was simulated with a zero mean uniformly distributed white noise input $u(t)$. Realistically it has been assumed that the measured output is corrupted by unknown coloured measurement noise $n(t)$. Notice that the system can be exactly modelled by a one-hidden-layer neural network with one hidden neuron and input node assignment $u(t-1)$, $u(t-2)$ and $y(t-1)$.

Initially the input vector was incorrectly assigned as

$$x(t) = [u(t-1) \quad y(t-1)]^{\mathrm{T}}$$

The network was trained and the model validity tests for this case are illustrated in Fig. 5 which shows that $\phi_{\varepsilon\varepsilon}(\tau) \neq \delta(\tau)$ and $\phi_{u\varepsilon}(\tau)$, $\phi_{u^2\varepsilon^2}(\tau)$ are well outside the 95% confidence bands. This indicates that the fitted model

$$y(t) = \frac{c_1}{1+\exp\{-(c_2u(t-1)+c_3y(t-1)+c_4)\}}$$

is deficient. In this particular example $\phi_{u\varepsilon}(\tau)$ and $\phi_{u^2\varepsilon^2}(\tau)$ suggest that the deficiency

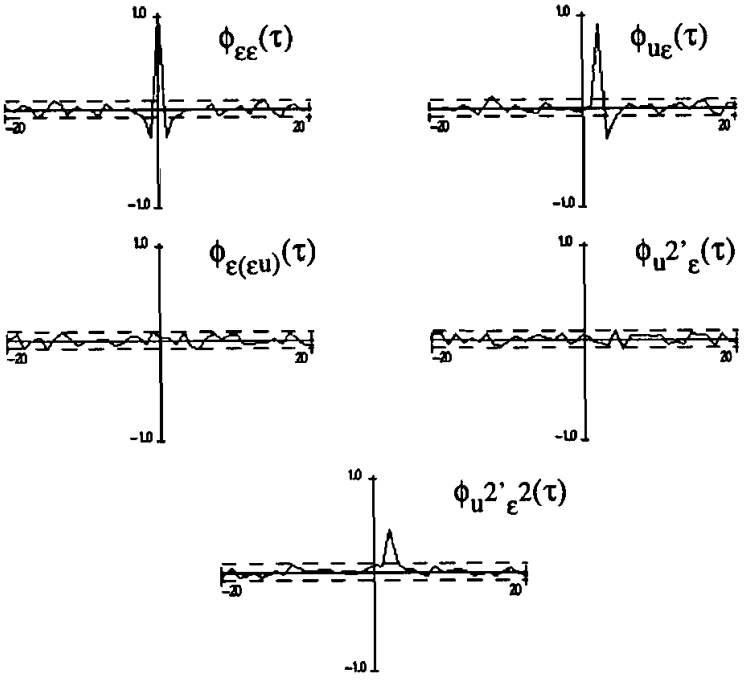

Figure 5. Example S1, incorrect process structure.

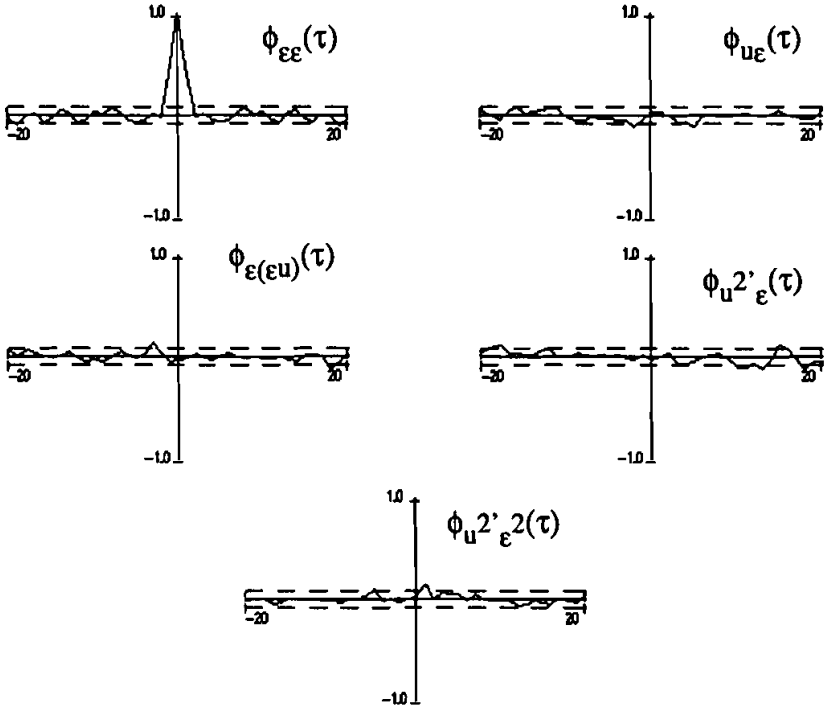

Figure 6. Example S1, correct process structure.

is due to a missing input, probably at lag 2. Redefining the input vector as

$$x(t) = [u(t-1) \quad u(t-2) \quad y(t-1)]^{\mathrm{T}}$$

and retraining the network gave the model validity results illustrated in Fig. 6. All the tests except the autocorrelation of the residuals are now satisfied. In fact $\phi_{\varepsilon\varepsilon}(\tau) \neq \delta(\tau)$ in both Figs 5 and 6, and this indicates that the network is a biased predictor. This is caused because there is correlated noise on the data and both system inputs and outputs have been used as input nodes to the network. This problem will be discussed in detail in the next section.

3.3. *The effect of noise*

It is realistic to assume that most system outputs will be corrupted by noise. Noise may be induced from external and/or internal sources as well as from the measuring instruments themselves. The effects of noise can result in severely biased models. Notice that bias is a subtle concept. Bias does not necessarily mean that the network will predict badly over the data set used to train it. Because the network has been trained by minimizing a cost function, usually some quadratic function of the errors, the output of the network will most probably provide a good prediction over the data set used for estimaton. Whilst this is almost universally used as a metric of network performance it does not mean that the network is a good model of the underlying system. The model may be highly biased. Physically this means that whilst the network will provide good predictions over the data used in training it is valid for that one specific data set and may not provide good predictions for

different data sets. In other words the model obtained is just a curve fit to one data set and is not a model of the underlying system that produced the data. Because the idea of fitting a neural network in the first place is often to produce a model that can be used to simulate the operation of the system, and hence to predict the system response to many different choices of input excitations, it is very important to both detect and if possible to eliminate bias.

A thorough theoretical analysis of these aspects is complex for neural networks because the models are non-linear-in-the-parameters and numerical minimization procedures are used for training. To illustrate the problems in the simplest possible way consider therefore the case of estimating the parameters in the linear model using the least squares method. A general linear regression model is given as

$$y(t) = \sum_{i=1}^{n} p_i(t)\theta_i + \varepsilon(t) \tag{24}$$

where $p_i(t)$ are the regressors, θ_i represent unknown parameters to be estimated and $\varepsilon(t)$ is called the modelling error or the residual. If N data samples are available (24) can be expressed in the matrix form as

$$\mathbf{Y} = \mathbf{P}\boldsymbol{\Theta} + \boldsymbol{\Xi} \tag{25}$$

where

$$\mathbf{Y} = [y(1) \quad \dots \quad y(N)]^{\mathrm{T}}, \quad \mathbf{P} = [p_1 \quad \dots \quad p_n], \quad \boldsymbol{\Xi} = [\varepsilon(1) \quad \dots \quad \varepsilon(N)]^{\mathrm{T}}$$

$$p_i = [p_i[1] \quad \dots \quad p_i(N)]^{\mathrm{T}} \quad \text{and} \quad \boldsymbol{\Theta} = [\theta_1 \quad \dots \quad \theta_n]^{\mathrm{T}}$$

It is well known (Goodwin and Payne 1977) that minimizing the sum of the errors squared yields the least squares estimate

$$\hat{\boldsymbol{\Theta}} = (\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{P}^{\mathrm{T}}\mathbf{Y} \tag{26}$$

The bias of the estimate defined as $E[\hat{\boldsymbol{\Theta}}] - \boldsymbol{\Theta}$ can now be determined. Substituting the true system model (25) into (26) relates the estimated parameter vector to the true parameter vector

$$\begin{aligned}\hat{\boldsymbol{\Theta}} &= (\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{P}^{\mathrm{T}}(\mathbf{P}\boldsymbol{\Theta} + \boldsymbol{\Xi}) \\ &= \boldsymbol{\Theta} + (\mathbf{P}^{\mathrm{T}}\mathbf{P})^{-1}\mathbf{P}^{\mathrm{T}}\boldsymbol{\Xi}\end{aligned} \tag{27}$$

The inverse of $(\mathbf{P}^{\mathrm{T}}\mathbf{P})$ must exist because this was used to obtain the estimate $\hat{\boldsymbol{\Theta}}$ in the first instance. Rearranging (27) gives

$$\mathbf{P}^{\mathrm{T}}\mathbf{P}(\hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta}) = \mathbf{P}^{\mathrm{T}}\boldsymbol{\Xi} \tag{28}$$

It is therefore clear that the estimate will only be unbiased, that is

$$E[\hat{\boldsymbol{\Theta}}] - \boldsymbol{\Theta} = 0 \tag{29}$$

if $E[\mathbf{P}^{\mathrm{T}}\boldsymbol{\Xi}] = 0$. This is a fundamental result from estimation theory which in simple terms says that on average the estimate will only approach the true value if the elements of $\mathbf{P}^{\mathrm{T}}$ and $\boldsymbol{\Xi}$ are uncorrelated such that $E[\mathbf{P}^{\mathrm{T}}\boldsymbol{\Xi}] = 0$. Although the neural network problem cannot be cast into the exact formulation of (25) the principle expressed by this simplified analysis has relevance to the neural network. For the general non-linear model, the parameter estimates obtained using the prediction error algorithm will be unbiased if the prediction error is uncorrelated with all linear and non-linear combinations of past inputs and outputs. This is a well known

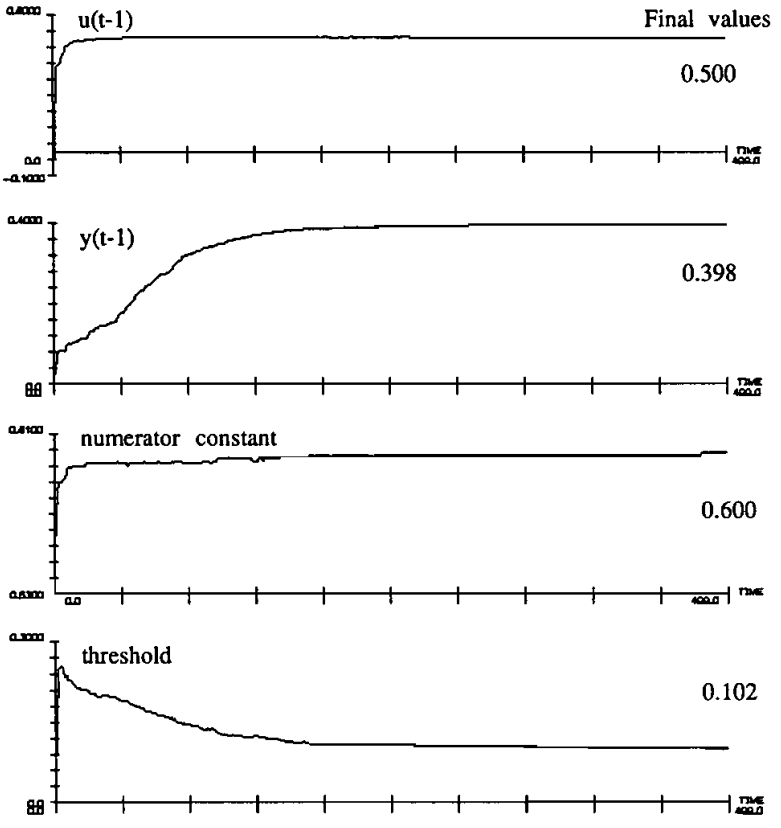

Figure 7. Example S2, noise free, evolution of estimated parameters during training.

property of the prediction error algorithm (Chen *et al.* 1990 a, b). For the linear regression model (24), the prediction error algorithm reduces to the least squares algorithm, and it is only required that the prediction error is uncorrelated with all the regressors to guarantee unbiased estimation. Furthermore, bias can easily be detected by the model validity tests. Recall that, for a general non-linear model, the correlation properties of (21) will only hold if the residuals are unpredictable from all linear and non-linear combinations of past inputs and outputs—the same requirement as for unbiased estimates.

To illustrate these ideas consider the system S2 defined by the model

$$\left.\begin{aligned} \bar{y}(t) &= \frac{0{\cdot}6}{1+\exp\{-(0{\cdot}5u(t-1)+0{\cdot}4\bar{y}(t-1)+0{\cdot}1)\}} \\ y(t) &= \bar{y}(t)+n(t) \end{aligned}\right\} \tag{30}$$

where the system input $u(t)$ is a zero mean uniformly distributed white noise sequence and $n(t)$ is the system noise.

Initially $n(t)$ is set to zero. A network with input vector $x(t) = [u(t-1)\ \ y(t-1)]^{\mathrm{T}}$ was trained and the evolution of the estimated parameters are illustrated in Fig. 7. After 500 representations of input and output pairs for training, the parameters almost converge to the values of 0·500, 0·398, 0·600 and 0·102 for the terms

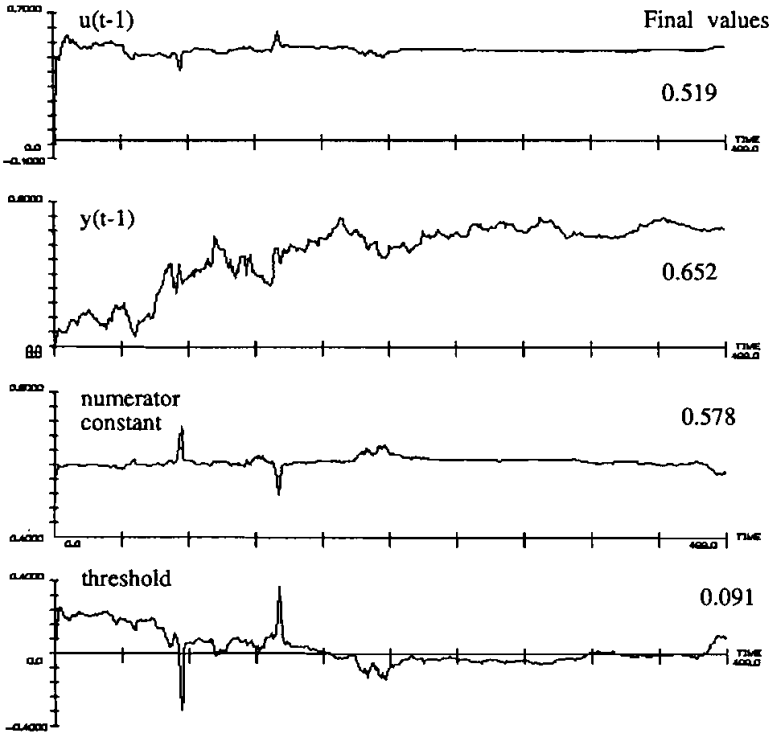

Figure 8. Example S2, evolution of estimated parameters during training for biased model.

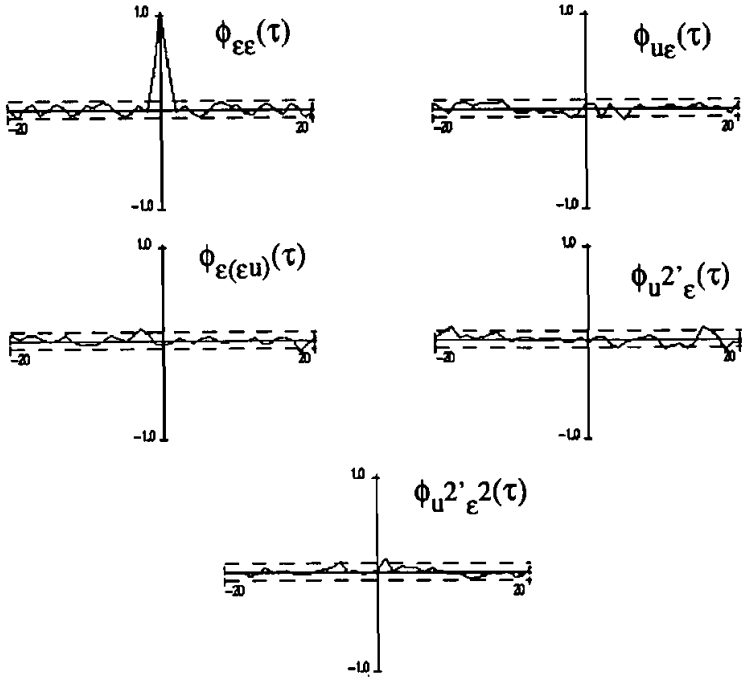

Figure 9. Example S2, correlation tests for network with correct process model.

$u(t-1)$, $y(t-1)$, numerator constant and threshold constant respectively. It is clear that the estimated parameters converge to the true parameter values.

The system was simulated again but this time with coloured noise $n(t) = e(t) + 0{\cdot}6e(t-1)$ where $e(t)$ is zero mean gaussian white noise with a variance of 0·01. The network was retrained and the evolution of the parameters during training is illustrated in Fig. 8. Inspection of Fig. 8 shows that the estimated parameter for the input term $y(t-1)$ deviates significantly from the true value, an estimated value of 0·652 compared with the value of 0·4. The model as expected is biased. This is easy to detect in this case because the true system model is known. In practice, of course, the true system parameters are not available but fortunately the model validity tests can be employed to detect bias. Fig. 9 shows the model validity tests for the identified model. It is seen that $\phi_{\varepsilon\varepsilon}(\tau) \neq \delta(\tau)$, indicating that the residual is correlated with lagged output measurements.

For the linear model identification, an analysis of (28) indicates that the coloured noise will cause bias in the parameters related to lagged output measurements. In general, of course, bias will not be restricted only to the parameters of lagged $y(t)$. It is interesting to see that, for this simple neural network model, the coloured noise also causes bias in the parameter connected to $y(t-1)$. The fact that only one parameter appears to be biased is due to the specific choice of input excitation and the network structure.

The bias in this example will only be eliminated if the residual becomes uncorrelated with past measurements. One way to achieve this is to model the noise. If a linear noise model is used the network architecture takes the form illustrated in Fig. 10 and the noise model parameters can be estimated as part of the network training. A first order noise model should be sufficient in this example and the evolution of the network parameters for this case are illustrated in Fig. 11. Inspection of the model validity tests in Fig. 12 shows that the estimates are now unbiased. The above discussion is only valid for an additive coloured noise source. For more complex noise sources, the analysis will be much more difficult.

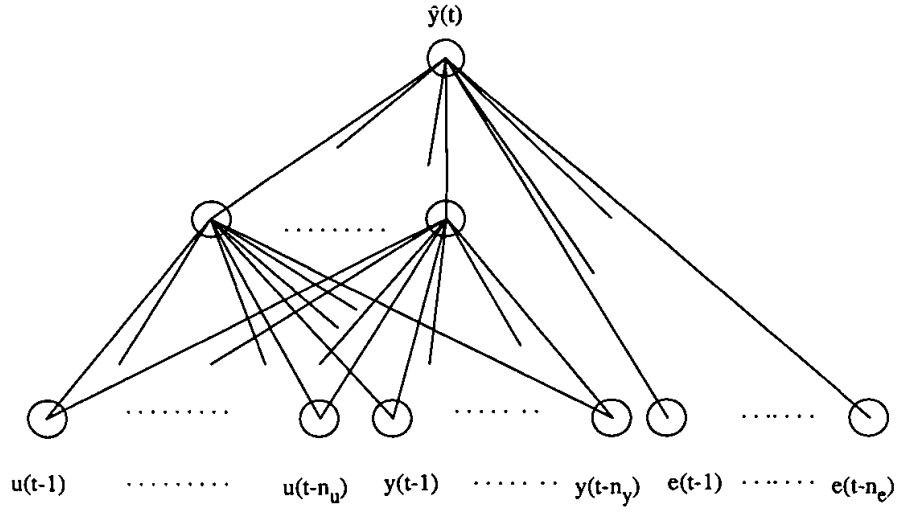

Figure 10. Neural network with a linear noise model.

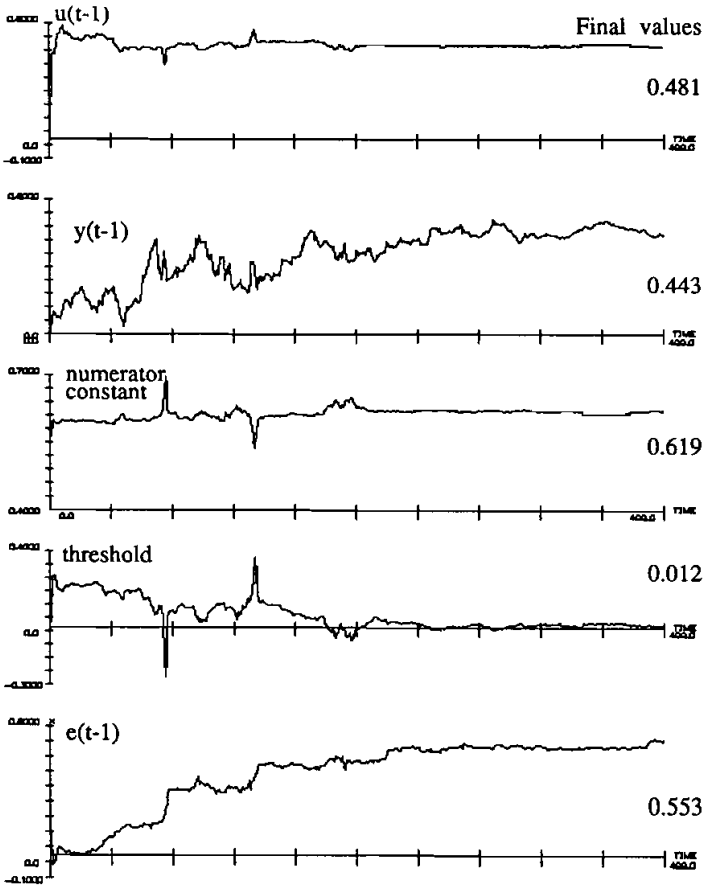

Figure 11. Example S2, evolution of estimated parameters during training for network with correct process and noise model.

It was noted earlier that a biased model may predict well over the estimation set but this is no guarantee that prediction over the other data set will be good. This phenomenon can be illustrated by modelling the data set generated by system S2 using network models with input node specifications which result in both biased and unbiased estimates. An extra 400 data points were generated using a different input sequence than that used to produce the estimation set. This set of data is called the testing set. Using the incorrect input node specification $x(t) = [u(t-1)]^T$, the network was trained over the estimation set and produced excellent prediction over this data. Prediction over the testing set however deteriorated. This is illustrated by the one-step-ahead prediction superimposed on the actual output in Fig. 13 for the data set between points 400 and 900. When the input nodes were specified correctly and a linear first order noise model was used, the trained network produced good predictions over both the estimation and testing sets as illustrated by the one-step-ahead prediction superimposed on the actual output in Fig. 14. The excellent prediction over estimation and test sets in this case is obtained because the trained model is unbiased and has learned the underlying mechanism which produced the data set. The effects of bias which are relatively mild in this simple

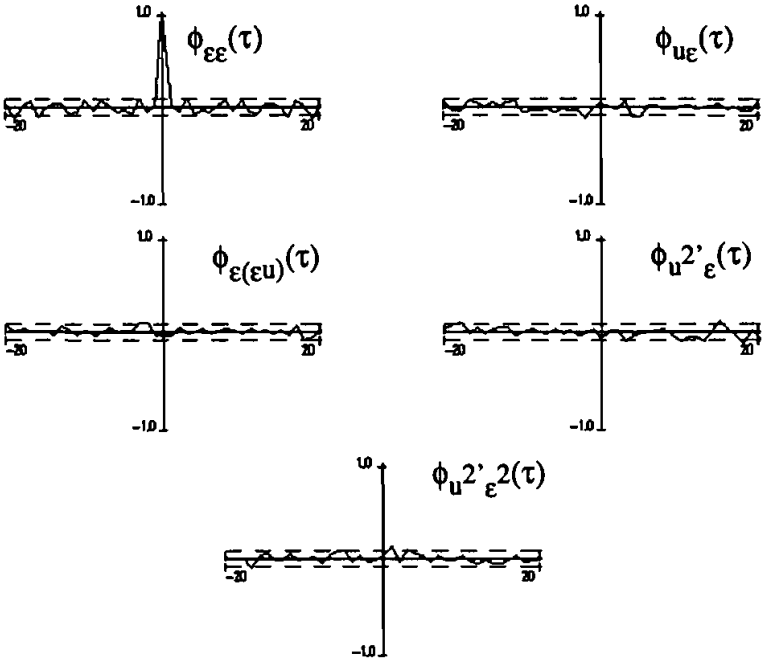

Figure 12. Example S2, correlation tests for network with correct process and noise model.

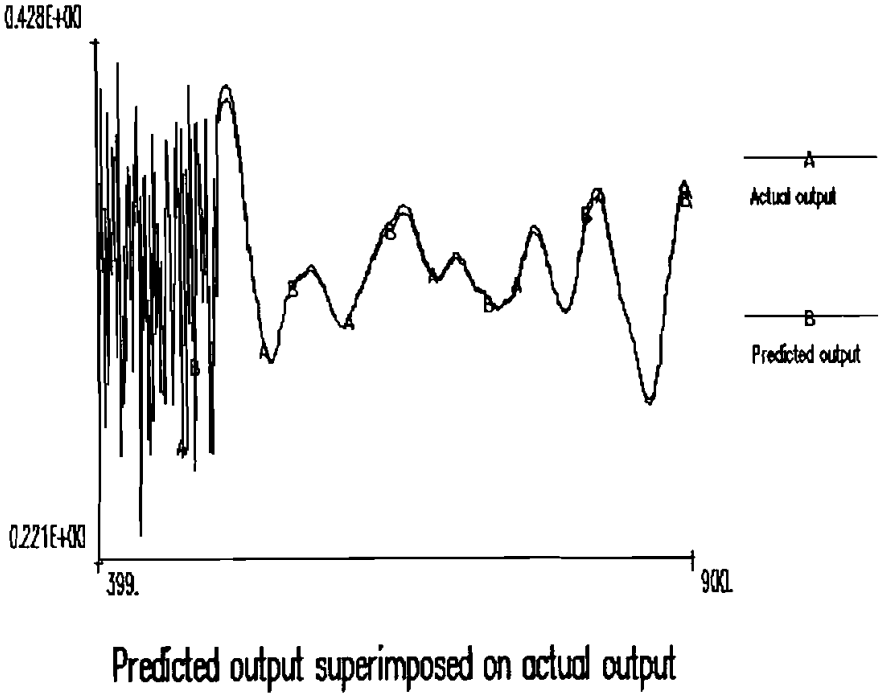

Figure 13. Example S2, one-step-ahead prediction superimposed on system's output for biased model.

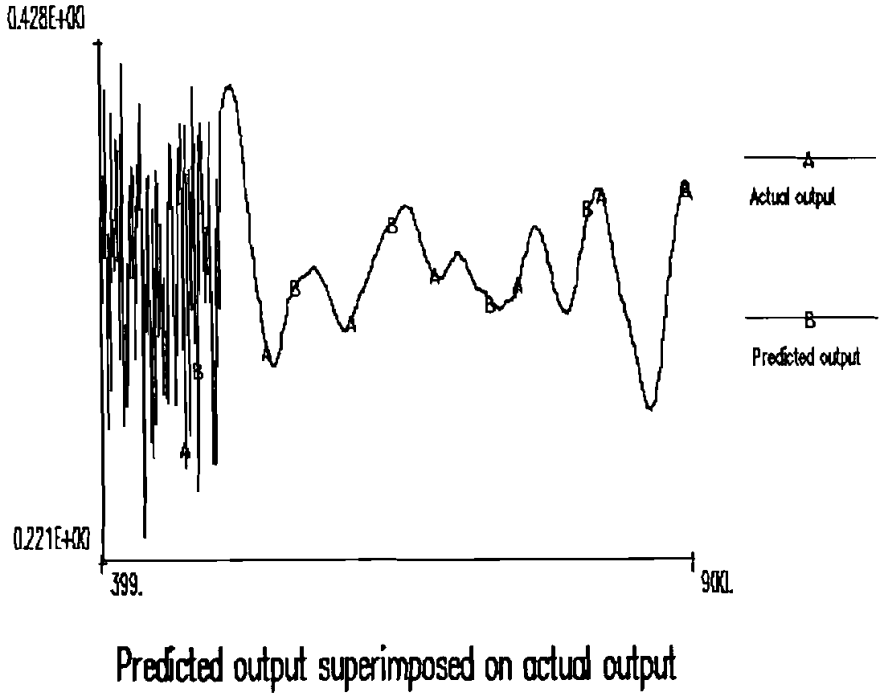

Figure 14. Example S2, one-step-ahead prediction superimposed on system's output for unbiased model.

example can be severe depending on the system and the statistics of the signals involved.

3.4. *Assignment of network nodes*

If records of the input $u(t)$ and output $y(t)$ are available the network input nodes can be assigned as lagged inputs $u(t-j)$, $j \geqslant 0$, lagged outputs $y(t-j)$, $j > 0$ or a combination of lagged inputs and outputs. The choice between these three options is not necessarily straightforward because it can influence both bias and network complexity.

Once again it is easier to introduce the concepts involved using a linear model and then to generalize the results to the neural network case. Consider the simple first order difference equation model

$$\bar{y}(t) = \frac{0{\cdot}5z^{-1}}{1 - 0{\cdot}7z^{-1}}\, u(t) \tag{31}$$

Rearranging (31) gives

$$\bar{y}(t) = 0{\cdot}7\bar{y}(t-1) + 0{\cdot}5u(t-1) \tag{32}$$

Obviously this system can be modelled very concisely by using both lagged inputs and outputs to fit a model of the form

$$\bar{y}(t) = a_1\bar{y}(t-1) + b_1u(t-1)$$

The disadvantage of this approach is that, if the output is corrupted by noise so that

$$y(t) = \bar{y}(t) + e(t) \tag{33}$$

fitting a model based on the measurements $y(t)$ and $u(t)$ will yield biased estimates. This follows because substitution shows that

$$y(t) = 0{\cdot}7y(t-1) + 0{\cdot}5u(t-1) + e(t) - 0{\cdot}7e(t-1) \tag{34}$$

and even if the noise $e(t)$ is zero mean and white it appears in the model as a coloured sequence such that from (28), $E[\mathbf{P}^{\mathrm{T}}\boldsymbol{\Xi}] \neq 0$ and hence the estimates will be biased. To obtain unbiased estimates, a noise model is required.

Alternatively the system can be modelled using lagged inputs only. Applying long division to (31) shows that in this case the system can be expressed as

$$\begin{aligned} y(t) = {} & 0{\cdot}5u(t-1) + 0{\cdot}35u(t-2) + 0{\cdot}245u(t-3) + 0{\cdot}1715u(t-4) \\ & + 0{\cdot}12005u(t-5) + \ldots \end{aligned} \tag{35}$$

The sequence will converge providing the roots of the denominator (poles) are within the unit circle but there may be many, many terms in the model. The disadvantage is that many more parameters must now be estimated to describe the same system. The advantage is that if the output is corrupted by noise, (35), then

$$\begin{aligned} y(t) = {} & 0{\cdot}5u(t-1) + 0{\cdot}35u(t-2) + 0{\cdot}245u(t-3) + 0{\cdot}1715u(t-4) \\ & + 0{\cdot}12005u(t-5) + \ldots + e(t) \end{aligned} \tag{36}$$

and since the noise will almost always be independent of the input, even if it is coloured, $E[\mathbf{P}^{\mathrm{T}}\boldsymbol{\Xi}] = 0$ and the estimates will be unbiased. Increased complexity in the model is therefore balanced by the fact that there will be no need to fit a noise model because the estimates will, under the conditions specified above, always be unbiased.

Conversely the model could be expanded in terms of outputs only. This offers no real advantages because many terms will be needed to describe the system and the effects of noise will be as discussed above for the model in (34) so bias will also be a problem.

These results will also apply to neural networks assuming the noise is additive at the system output. Assigning network input nodes as lagged inputs only will yield unbiased estimates without the need to fit noise models at the expense of possibly very complex architectures and hence slow training.

Consider, for example, a network similar to that illustrated in Fig. 4 but with input nodes defined by $u(t-1)$ and $u(t-2)$. Using the sigmoidal activation function of (17) will yield the input–output expansion

$$\begin{aligned} y(t) = {} & a_0 + a_1u(t-1) + a_2u(t-2) + a_3u^2(t-1) + a_4u^2(t-2) + a_5u(t-1)u(t-2) \\ & + a_6u^3(t-1) + a_7u^3(t-2) + a_8u^2(t-1)u(t-2) + a_9u(t-1)u^2(t-2) + \ldots \end{aligned} \tag{37}$$

This has the form of a Volterra Series. Notice, however, that if the true system model included a $y^2(t-1)$ term say, attempts to approximate this by an expansion of inputs only using (37) may result in very complex networks.

The predictive accuracy of the model can be computed by defining the normalized root mean square of the residuals as an error index

$$error\ index = \left[\frac{\Sigma(\hat{y}(t) - y(t))^2}{\Sigma y^2(t)}\right]^{1/2} \tag{38}$$

A comparison of the predictive accuracy of networks with input nodes assigned as lagged $u(\cdot)$s only and as a combination of lagged $u(\cdot)$s and $y(\cdot)$s can be illustrated by the simulation study of system S3 described by

$$\left.\begin{aligned} \bar{y}(t) &= \frac{0{\cdot}3}{1 + \exp\{-(0{\cdot}3u(t-1) + 0{\cdot}6\bar{y}^2(t-1) + 0{\cdot}1)\}} \\ n(t) &= e(t) + 0{\cdot}5e(t-1) \\ y(t) &= \bar{y}(t) + n(t) \end{aligned}\right\} \tag{39}$$

A uniformly distributed zero mean white noise sequence $u(t)$ of variance 1·0 was used to generate 500 data points. The output data was corrupted by coloured noise generated using a zero mean Gaussian white noise sequence of variance 0·01. A one-hidden-layer neural network with one hidden node was used to model this system. The values of the error index for input vector specification consisting of only lagged $u(\cdot)$s and input vector specification consisting of lagged $u(\cdot)$s and lagged $y(\cdot)$s, are shown in the Table. For the input vector consisting of lagged $u(\cdot)$s and lagged $y(\cdot)$s, an additional first order noise model was also fitted as part of the network. The results show that a neural network model with input vector specification consisting of only lagged $u(\cdot)$s with nine input nodes produced a predictive accuracy equivalent to a neural network model with input vector specification consisting of both lagged $u(\cdot)$s and lagged $y(\cdot)$s with only two input nodes. Considerable predictive accuracy is obtained when both lagged $u(\cdot)$s and lagged $y(\cdot)$s are used as input nodes but at the expence of fitting a noise model to obtain an adequate overall model for the system.

If only lagged $u(\cdot)$s are used as the input vector it is no longer appropriate to test all five model validity checks defined by (21) because even if the noise is correlated (i.e. $\phi_{\varepsilon\varepsilon}(\tau) \neq \delta(\tau)$) the model should still be unbiased. Thus, if the structure of the model or network is such that unbiased estimates will be obtained even when the noise is correlated it can be shown that (Billings and Voon, 1986) the estimate will be unbiased providing

$$\left.\begin{aligned} \phi_{u\varepsilon}(\tau) &= 0 \\ \phi_{u^2\cdot\varepsilon^2}(\tau) &= 0 \\ \phi_{u^2\cdot\varepsilon}(\tau) &= 0 \end{aligned}\right\}, \quad \forall\,\tau \tag{40}$$

n_0	Lagged $u(\cdot)$s and $y(\cdot)$s†	Lagged $u(\cdot)$s only
2	0.393	0.400
3	0.379	0.400
5	0.375	0.396
7	0.377	0.398
9	0.373	0.393

†An additional first-order noise model is used.

Error index comparison for example S3.

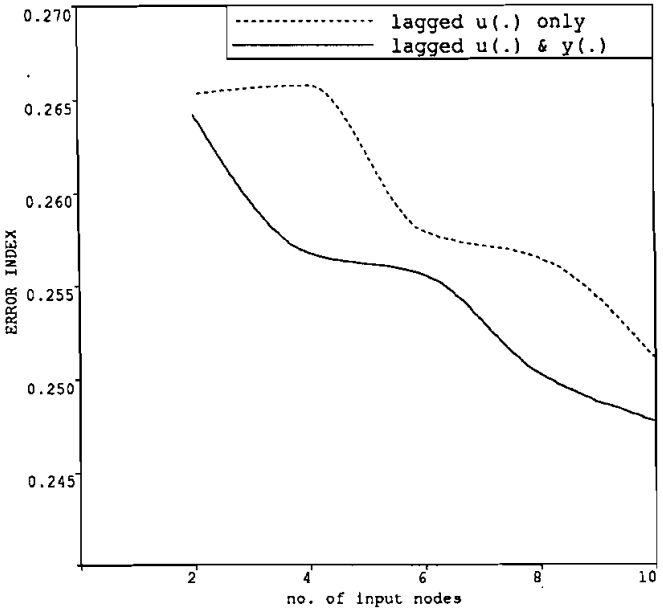

Figure 15. Example S4. A comparison of predictive accuracy for network with different input node assignment.

To illustrate this idea consider simulated system S4 defined by

$$\left.\begin{aligned} \bar{y}(t) &= \frac{0{\cdot}6}{1+\exp\{-(0{\cdot}5u(t-1)+0{\cdot}6\bar{y}(t-4)+0{\cdot}4\bar{y}(t-8)+0{\cdot}1)} \\ y(t) &= \bar{y}(t)+n(t) \end{aligned}\right\} \tag{41}$$

The output data was generated using a uniformly distributed white noise sequence $u(t)$ of variance 1·0 and a coloured noise $n(t)$.

Networks were trained using a range of input nodes defined using only lagged $u(\cdot)$s and using both lagged $u(\cdot)$s and lagged $y(\cdot)s$. The plot of the error indices versus the number of input nodes is illustrated in Fig. 15. A smaller error index is obtained when both lagged $u(\cdot)$s and lagged $y(\cdot)$s are used as input node specification.

If only lagged $u(\cdot)$s are used as input nodes, all the three correlation tests, (40) appear to be satisfied when $n_0 \geqslant 9$. The correlation tests for $n_0 = 7$, 8, 9 and 10 are shown in Fig. 16. Notice that $\phi_{\varepsilon\varepsilon}(\tau)$ and $\phi_{\varepsilon(\varepsilon u)}(\tau)$ in Fig. 16 are not relevant for this case.

When both lagged $u(\cdot)$s and lagged $y(\cdot)$s are used as input nodes with $x = [u(t-1)\ y(t-1)]^T$, an underspecified case, the appropriate correlation test are illustrated in Fig. 17. The cross-correlation between the input and the residuals $\phi_{u\varepsilon}(\tau)$ is well outside the confidence intervals at lags 4 and 8 correctly indicating that the network is deficient. However, using the correct node assignments $x = [u(t-1)\ y(t-4)\ y(t-8)]^T$ and a first order linear noise model yields an unbiased network model as indicated by the correlation tests in Fig. 18.

Assigning network input nodes as lagged inputs and outputs may considerably reduce network complexity and hence increase the possibility of rapid learning and

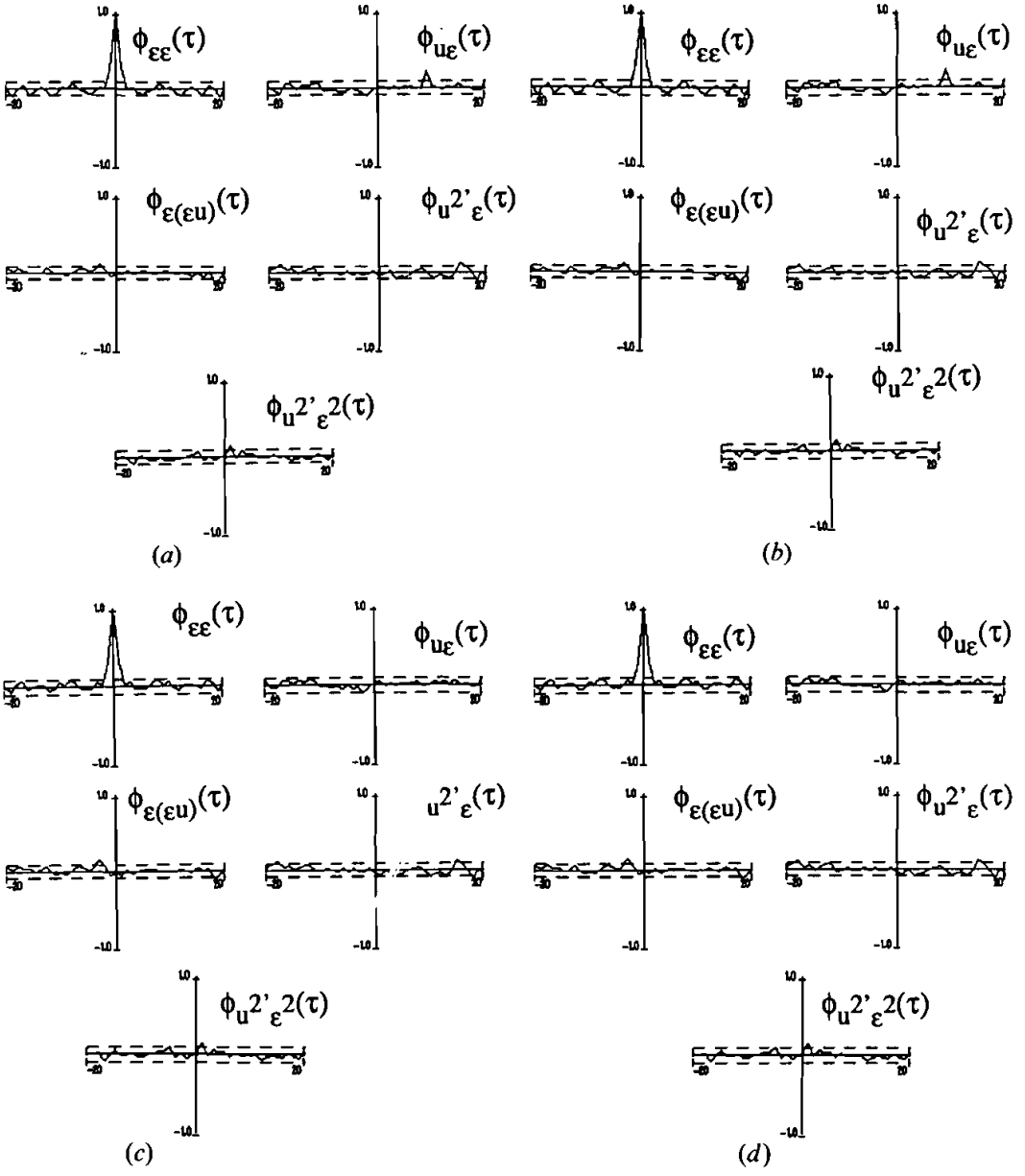

Figure 16. Example S4. Correlation tests for network using only lagged $u(\cdot)$'s as inputs: (*a*) $n_u = 7$; (*b*) $n_u = 8$; (*c*) $n_u = 9$; (*d*) $n_u = 10$.

adaptation but unless noise models are estimated the network will provide a biased representation of the system.

This analysis may, however, be far too simplistic if the assumption of additive noise is violated. There is no reason why noise should be purely additive at the output, it can arise internally at various locations in a system. If the system is linear this does not create any additional problems because, by superposition, the noise can be translated to be additive at the output. But when the system is non-linear

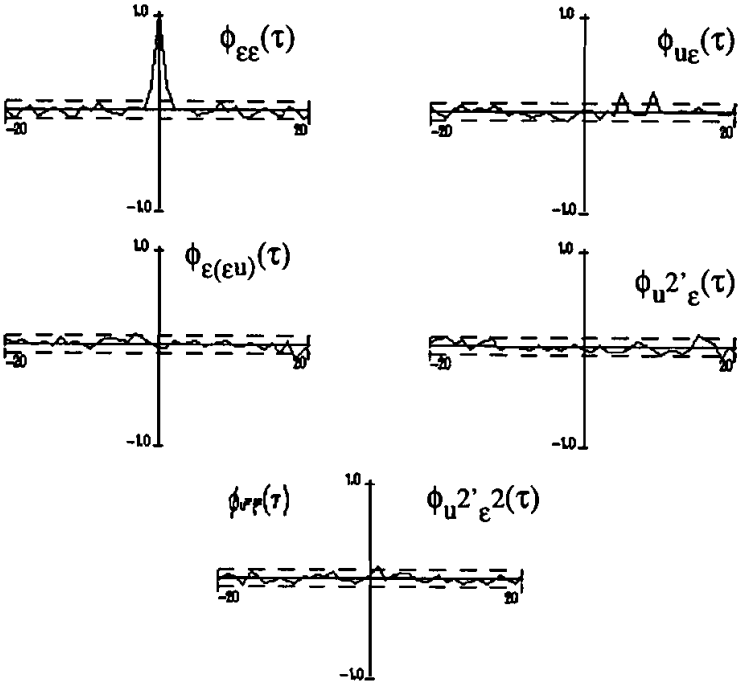

Figure 17. Example S4. Correlation tests for network with input specification $n_u = n_y = 1$.

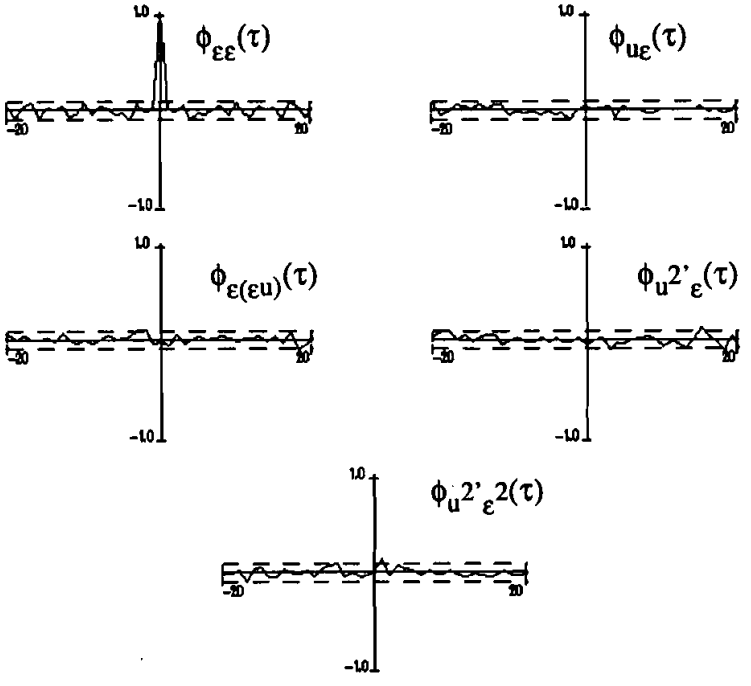

Figure 18. Example S4. Correlation tests for network with input specification $x(t) = [u(t-1)\ y(t-4)\ y(t-8)]^T$ and $n_e = 1$.

internal noise can induce cross-product terms or non-linear expansions between the input, output and noise. If this occurs the induced noise terms will be highly correlated with the input such that unless these are modelled explicitly or accommodated in some other way the network will be a biased representation of the system even if only lagged inputs are assigned as network input nodes.

Ideally the model validity tests should detect all these deficiencies in network performance including bias due to internal noise. The cause of the bias will however be different for different assignments of network input nodes. Consequently the full five tests defined by (21) should be satisfied if $u(\cdot)$s and $y(\cdot)$s are used as network input nodes but only the subset of the tests defined by (40) are necessary if the network is expanded in terms of $u(\cdot)$s only.

3.5. *Network complexity*

Another important question to ask is; does network performance improve with increasing network complexity? In other words, is it important to consider input node assignment, number of hidden nodes, etc, or should we just build the biggest network that our computer can accommodate.

If the estimated model is linear-in-the-unknown parameters then the loss function or sum of the residuals squared will monotonically decrease as the number of terms in the model is increased. After a while though the increased model complexity is not justified by the insignificant decrease in the loss function and the model essentially becomes just a high dimensional curve fit to the data set used for estimation. This means that whilst the model will predict well over the estimation set it is not a good representation of the underlying system.

This phenomenon typically becomes evident if the loss function of the estimated model is computed over a sequence of data which has not been used to fit the model. Such a data set is usually called the prediction or testing set. A plot of the loss function computed over the testing set often displays a minimum when the estimated model structure coincides with the true system order or structure. Overfitting, therefore, is undesirable. This distinction between performance over estimation and testing set is well known in system identification and is referred to as generalization for neural networks. Broomhead and Lowe (1988) provide a good discussion of this for the case of radial basis function networks. In linear system identification, model overfitting can be avoided by several approaches—one of which is to fit models of increasing order and to select the model of minimal complexity which just satisfies the model validity tests.

Extending these ideas to neural networks in general is not straightforward because the models provided are non-linear-in-the-parameters. Overfitting however is likely to produce misleading results because even if the loss function reduces as network complexity is increased this does not necessarily mean that the model provided is a better representation of the system. This is the worst possible result because, as in the case of bias, the effect is not easy to detect. Consider examples S1 and S2 introduced in § 3.2 and 3.3 respectively to illustrate some of the ideas. Initially only the noise free case ($n(t) = 0$) will be considered.

A one-hidden-layer neural network with one hidden node will be used to model both these systems. The number of input nodes was increased from 1 to 5 with the node specifications $[u(t-1)]$, $[u(t-1)\ y(t-1)]$, $[u(t-1)\ u(t-2)\ y(t-1)]$, $[u(t-1)\ u(t-2)\ y(t-1)\ y(t-2)]$, $[u(t-1)\ u(t-2)\ u(t-3)\ y(t-1)\ y(t-2)]$ to produce

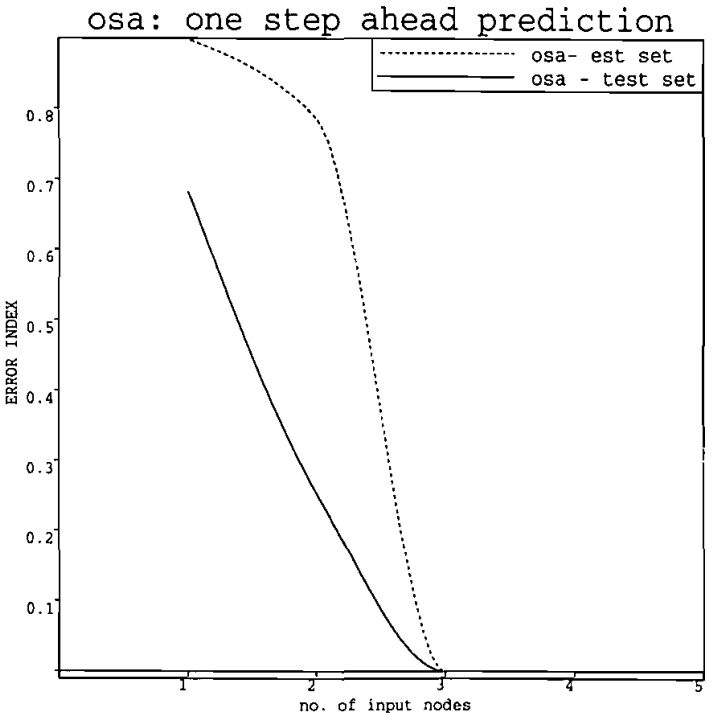

Figure 19. Example S1. Increasing the number of input nodes.

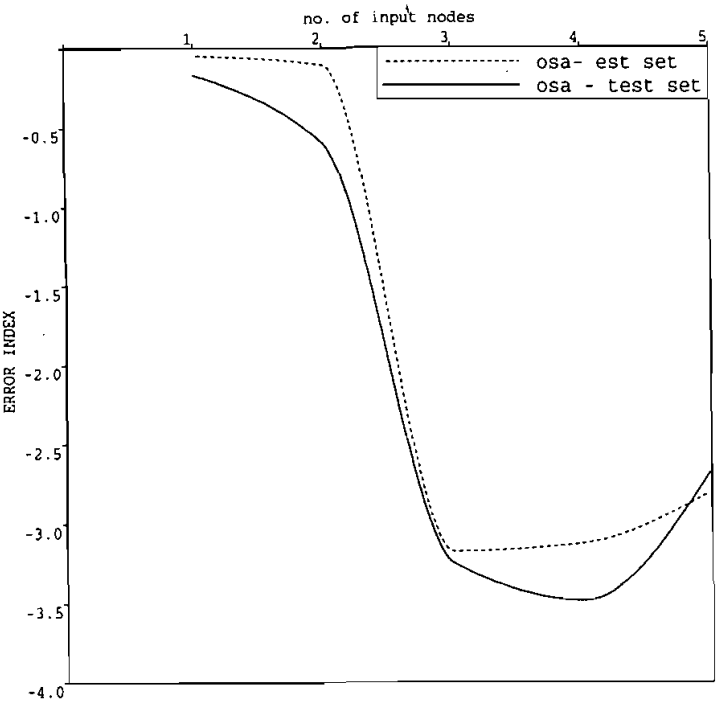

Figure 20. Example S1. Increasing the number of input nodes, error index using logarithmic scale.

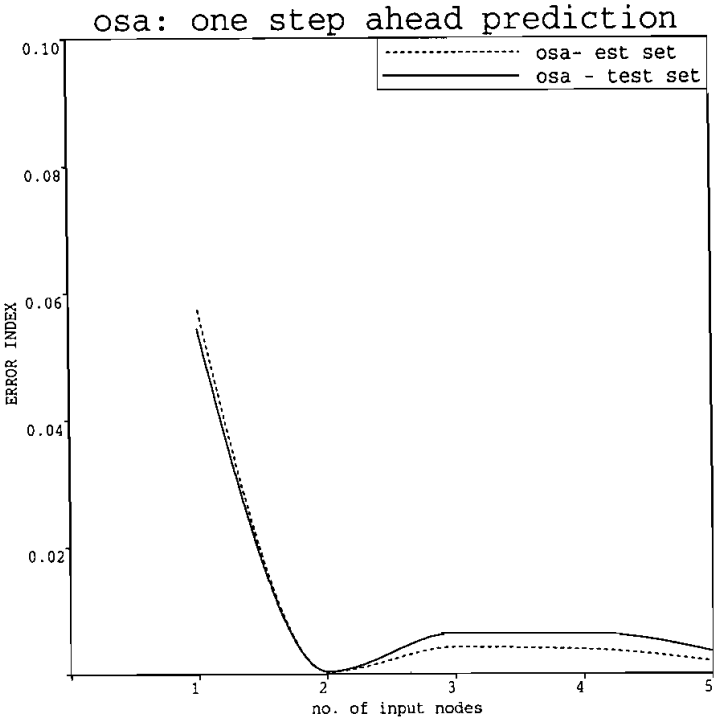

Figure 21. Example S2. Increasing the number of input nodes.

error indexes for the estimation and test sets shown in Fig. 19 for system S1. Notice that a minimum error index value occurs when the number of input nodes is 3, which corresponds to the correct input vector specification $[u(t-1)\ u(t-2)\ y(t-1)]$. This is shown more clearly if the error index is plotted using a logarithmic scale as illustrated in Fig. 20.

The plots of the error index values versus the number of input nodes for Example S2 are shown in Fig. 21. The minimum value of error index occurs when the number of input nodes is 2 which corresponds to input vector specfication $[u(t-1)\ y(t-1)]$.

When the output data are corrupted by noise, the plots of error indexes against the number of input nodes are shown in Figs 22 and 23 for S1 and S2 respectively. The clarity of the results in the noise-free case is now disguised by the effects of the noise. Fortunately the model validity tests can be used to aid the selection of an appropriate network structure.

The correlation tests for S1 for the number of input nodes (n_0) equal to 1, 2, ..., 5 are shown in Fig. 24. When $n_0 \geqslant 3$, all the tests are within the confidence intervals except $\phi_{\varepsilon\varepsilon}(\tau)$ suggesting that $n_0 = 3$ should provide an acceptable process model structure. Estimation of a noise model will eliminate the bias as shown in § 3.3.

The correlation tests for S2 as the number of input nodes is increased from 1 to 5 are illustrated by Fig. 25. All the correlation tests except $\phi_{\varepsilon\varepsilon}(\tau)$ are satisfied when $n_0 \geqslant 2$ indicating that the process model can be adequately represented by a network with input vector specification $[u(t-1)\ y(t-1)]$. Estimation of a noise model should reduce $\phi_{\varepsilon\varepsilon}(\tau)$ to an impulse to yield an unbiased representation of the system.

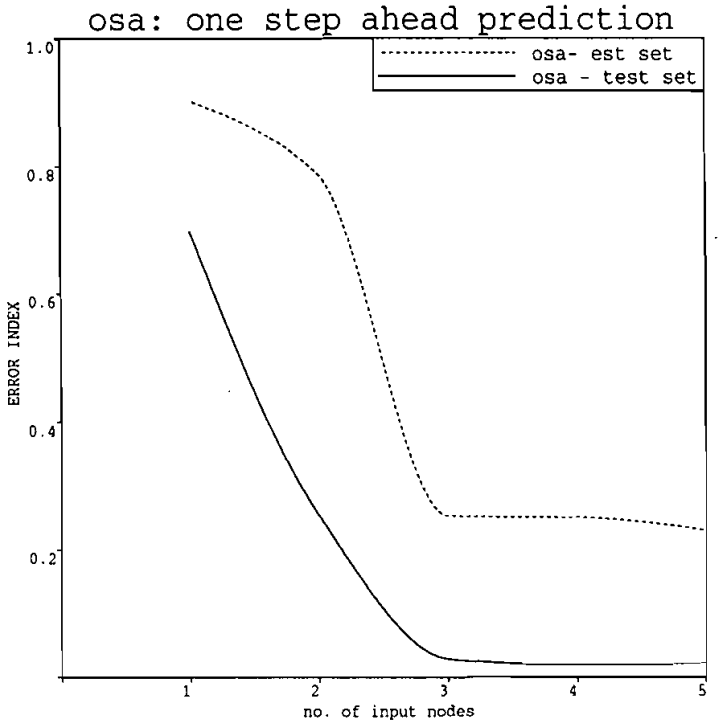

Figure 22. Example S1. Increasing the number of input nodes for the output of the system corrupted by noise.

3.6. *One-step-ahead versus model predicted output*

The common measure of predictive accuracy considered by many authors is computed using the one-step-ahead prediction of the system outputs. This was defined in (6) but can be expressed in the more general form as

$$\hat{y}(t) = f(u(t-1), ..., u(t-n_u), y(t-1), ..., y(t-n_y)) \tag{42}$$

where $f(\cdot)$ is a non-linear function. It is not surprising that often $\hat{y}(t)$ will be a relatively good prediction of $y(t)$ over the estimation set even if the model is biased because the model was estimated by minimizing the prediction errors. An additional and often much better metric of the predictive capability of the fitted model is to compute the model predicted output defined by

$$\hat{y}_d(t) = f(u(t-1), ..., u(t-n_u), \hat{y}_d(t-1), ..., \hat{y}_d(t-n_y)) \tag{43}$$

Notice that if only lagged inputs are used to assign network input nodes then

$$\hat{y}(t) = \hat{y}_d(t) \tag{44}$$

The identification of a liquid level system will be used to illustrate these concepts. Data was obtained from a large pilot scale liquid level system with a conical tank

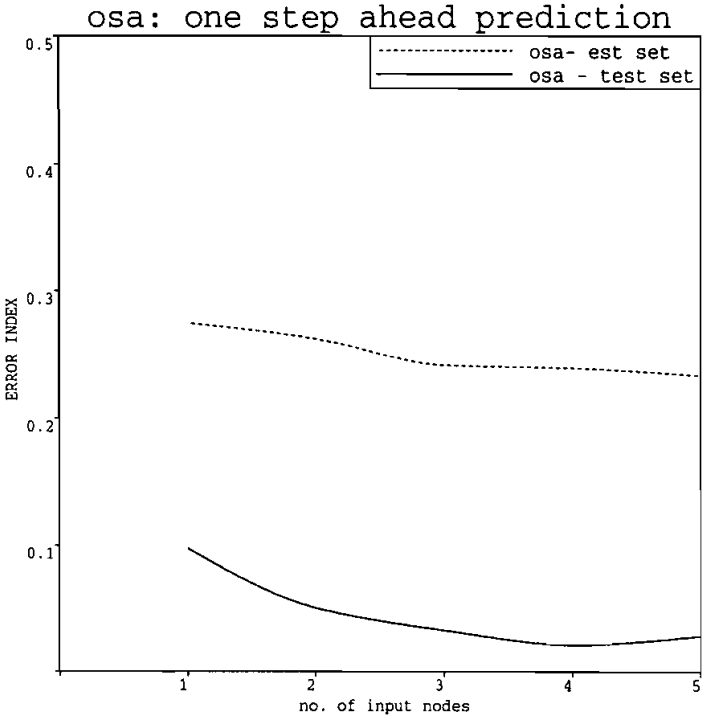

Figure 23. Example S2. Increasing the number of input nodes for the output of the system corrupted by noise.

which induced non-linear effects. A one-hidden-layer network with 3 hidden nodes was used to model this system. The number of input nodes was increased by specifying the input vector from $n_u = n_y = 1$ to $n_u = n_y = 9$. The plot of the error index against the number of input nodes is shown in Fig. 26. When $n_u = n_y = 3$ both the one-step-ahead prediction and the model predicted output are good. But for $n_u = n_y > 6$, the model predicted outputs deteriorate although the one-step-ahead predictions remain good. When the network involves a large number of parameters which have to be trained, the parameters may not converge in one pass through the data. The use of one-step-ahead predictions, as illustrated by this example, may not reveal the inadequacy of the trained network. The model predicted outputs appear to be far more sensitive and indicate that the model is not adequate because the parameters have not converged. Retraining the networks for $n_u = n_y = 4$, 5, 6 and 7, with two passes through the data and for $n_u = n_y = 8$ and 9 with three passes produced improved model predicted outputs as illustrated in Fig. 27.

3.7. *Linear model fitting*

Although one of the main advantages of neural networks it that they can be used to describe very complex non-linear relationships between signals it is interesting to consider what would happen if the system which generated the data happened to

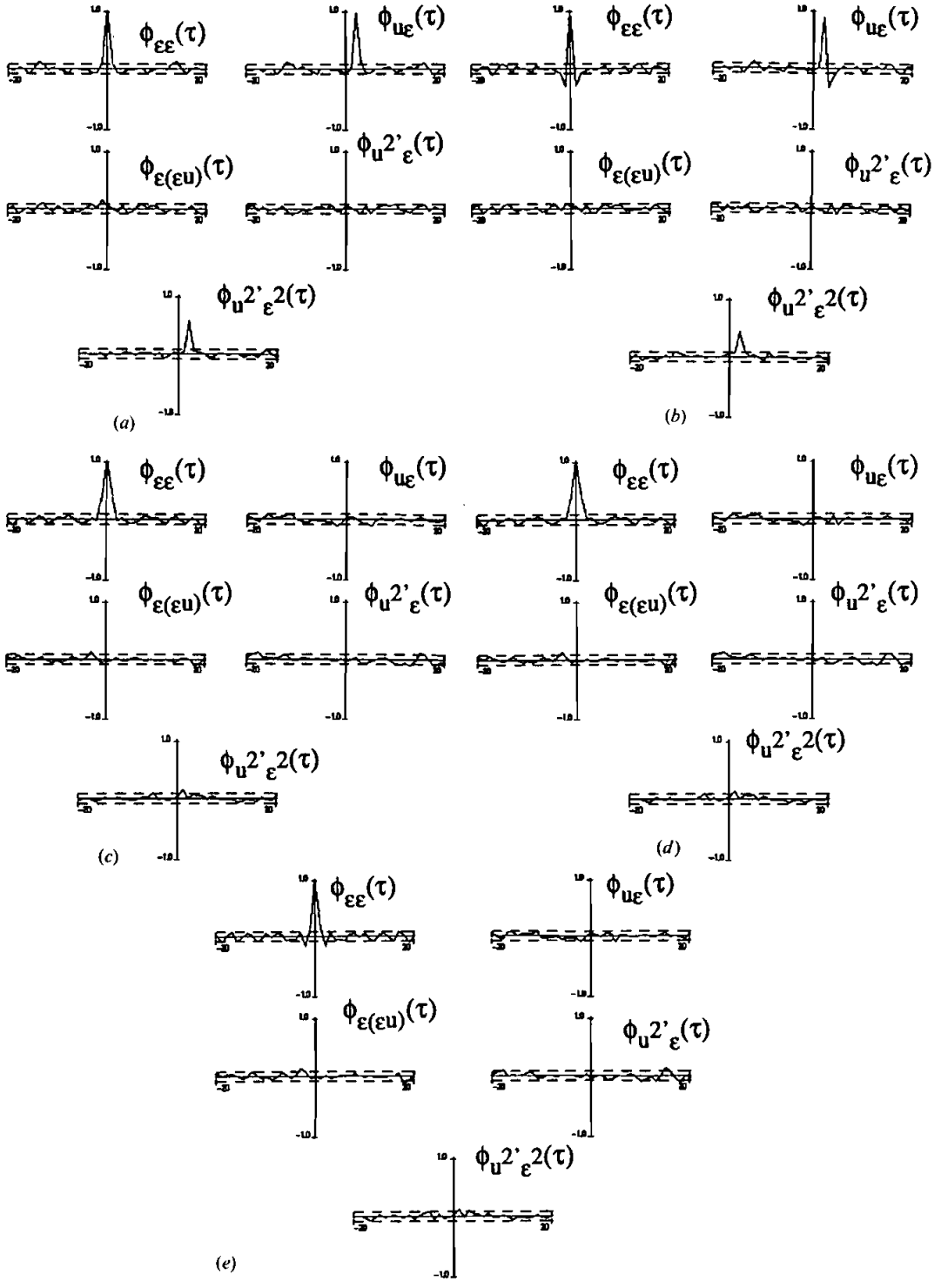

Figure 24. Example S1. Correlation tests for network with input nodes using both lagged $u(\cdot)$s and lagged $y(\cdot)$s as the number of input nodes is increased; (*a*) $n_0 = 1$; (*b*) $n_0 = 2$; (*c*) $n_0 = 3$; (*d*) $n_0 = 4$; (*e*) $n_0 = 5$.

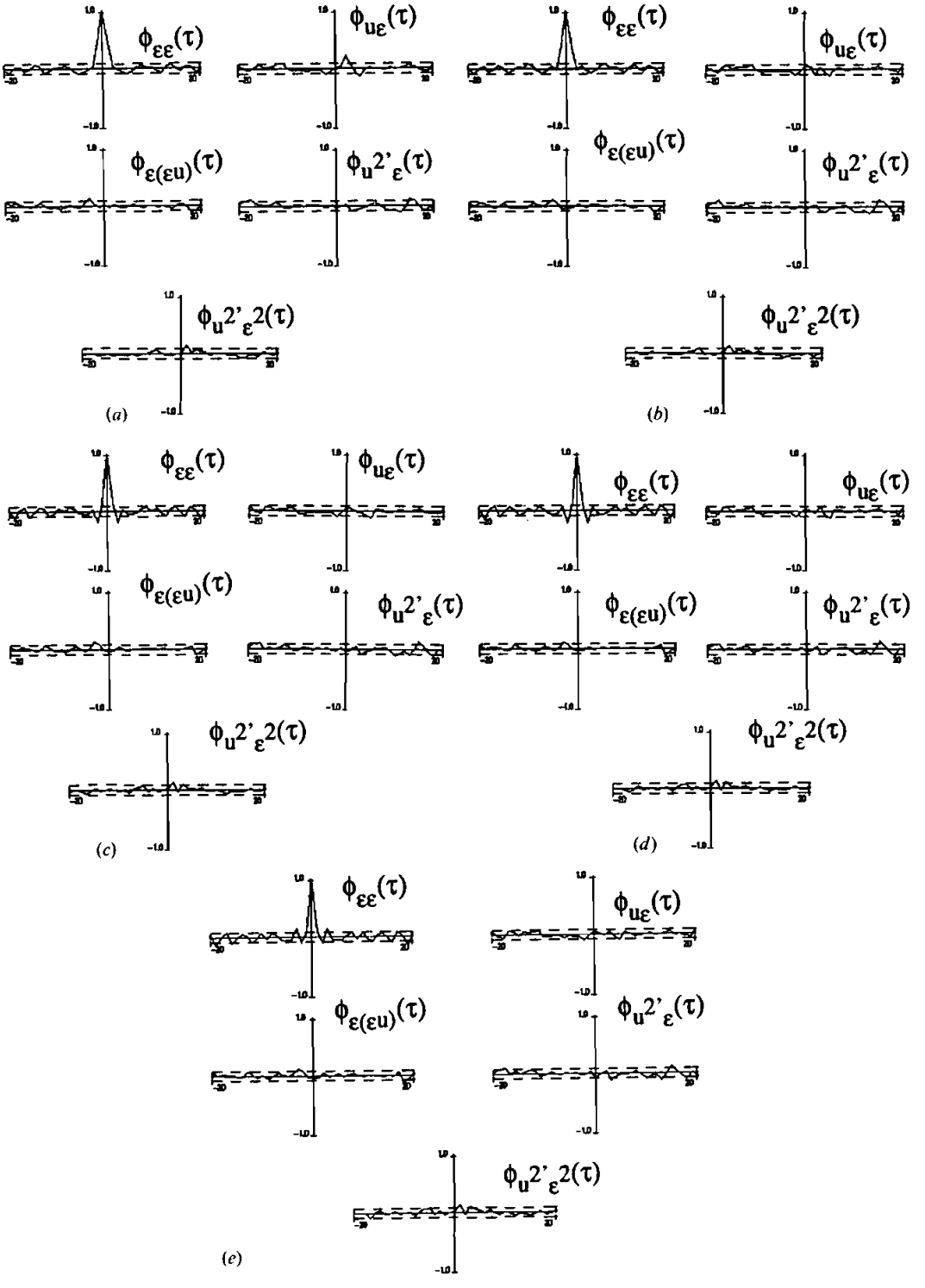

Figure 25. Example S2. Correlation tests for a network with input nodes using both lagged $u(\cdot)$s and lagged $y(\cdot)$s as the number of input nodes is increased: (*a*) $n_0 = 1$; (*b*) $n_0 = 2$; (*c*) $n_0 = 3$; (*d*) $n_0 = 4$; (*e*) $n_0 = 5$.

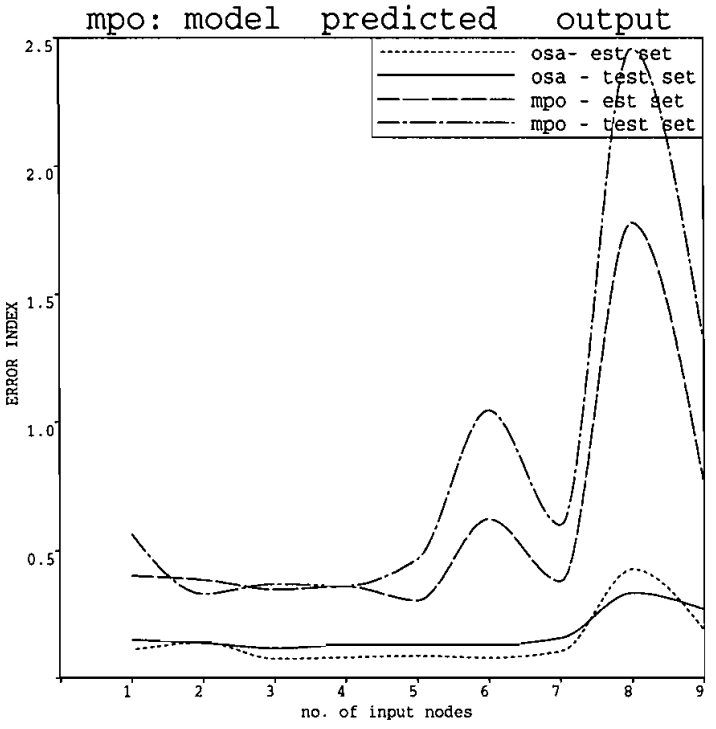

Figure 26. The plots of error indices for liquid level system training with number of passes equal 1 for all input nodes specification.

be linear. In this case the neural network will force a non-linear model to fit the data. This can then be considered in the context of the idea of § 3.5 because the network will be overfitting. This is undesirable because if the input–output signals are gaussianly distributed—a reasonable assumption from the Central Limit Theorem—then a linear system minimizing the mean squared prediction errors will provide the best estimate from the class of all linear and non-linear estimators. In other words, under these conditions, fitting a non-linear relationship can never improve the performance of the model. It may therefore be useful to augment the network with some purely linear branches to alleviate this problem.

4. Conclusions

Various aspects of neural network performance have been investigated using results and ideas from system identification and estimation theory. It has been shown that while it is easy to train a neural network which predicts well over the estimation set this does not necessarily mean that the network provides an adequate description of the underlying mechanism which generated the data. It is difficult to get definitive analytical results in this area but the model validity tests introduced in this study do appear to provide a useful metric of network performance. Although the results have been presented in the context of non-linear model estimation they should be applicable to neural network modelling in general.

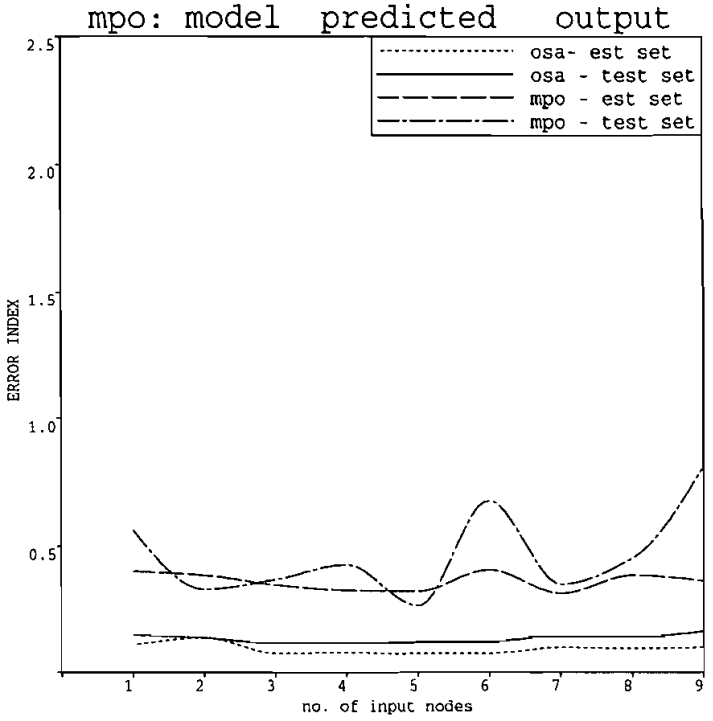

Figure 27. The plots of error indices for liquid level system training with different number of passes for different input nodes specification.

ACKNOWLEDGMENTS

SAB gratefully acknowledges support for this work from the UK Science and Engineering Research Council.

REFERENCES

BILLINGS, S. A., JAMALUDDIN, H. B., and CHEN, S., 1991, A comparison of the backpropagation and recursive prediction error algorithms for training neural networks. *Mechanical Systems and Signal Processing*, **5**, 233–255.

BILLINGS, S. A., and VOON, W. S. F., 1983, Structure detection and model validity tests in the identification of nonlinear systems. *Proceedings of I.E.E., Part D*, **130**, 193–199.

BILLINGS, S. A., and VOON, W. S. F., 1986, Correlation based model validity tests for non-linear models. *International Journal of Control*, **44**, 235–244.

BHAT, N. V., MINDERMAN, Jr., P. A., MCAVOY, T., and WANG, N. S., 1990, Modelling chemical process systems via neural computation. *I.E.E.E. Control Systems Magazine*, April 1990.

BRADY, M. L., RAGHAVAN, R., and SLAWNY, J., 1989, Back propagation fails to separate where perceptrons succeed. *I.E.E.E. Transactions on Circuits and Systems*, **36**, 665–674.

BROOMHEAD, D. S., and LOWE, D., 1988, Multivariable functional interpolation and adaptive networks. *Complex Systems*, **2**, 321–355.

CHEN, S., BILLINGS, S. A., and GRANT, P. M., 1990 a, Non-linear systems identification using neural networks. *International Journal of Control*, **51**, 1191–1214.

CHEN, S., COWAN, C. F. N., BILLINGS, S. A., and GRANT, P. M., 1990 b, A parallel recursive prediction error algorithm for training layered neural networks. *International Journal of Control*, **51**, 1215–1228.

CYBENKO, G., 1989, Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, **2**, 303–314.

FUNAHASHI, K., 1989, On the approximate realisation of continuous mappings by neural networks. *Neural Networks*, **2**, 183–192.

GOODWIN, G. C., and PAYNE, R. L., 1977, *Dynamic System Identification: Experiment Design and Data Analysis* (New York: Academic Press).

LAPEDES, A., and FARBER, R., 1987, Nonlinear Signal Processing using Neural Networks: Prediction and System Modelling. Preprint LA-UR-87-2662.

LEONARD, J., and KRAMER, M. A., 1990, Improvement of the backpropagation algorithm for training neural networks. *Computers in Chemical Engineering.*, **14**, 337–341.

LJUNG, L., and SÖDERSTRÖM, T., 1983, *Theory and Practice of Recursive Identification*, (Cambridge, Mass: MIT Press).

NAHI, N. E., 1969, *Estimation Theory and Applications*, (New York: Krieger).

NARENDRA, K. S., and PARTHASARATHY, K., 1990, Identification and control of dynamical systems using neural networks. *I.E.E.E. Transactions on Neural Networks*, **1**, 4–27.

RUMELHART, D. E., and MCCLELLAND, J. L., (editor), 1986, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations* (Cambridge, Mass: MIT Press).

SCHMIDHUBER, J., 1989, Accelerated learning in back-propagation nets. (In *Connectionism in Perspective*, edited by R. Pfeifer, Z. Schreter, F. Fogelman-Soulie and L. Steele (Amsterdam: Elsevier North-Holland).

SUTTON, R. S., 1986, Two problems with backpropagation and other steepest-descent learning procedures for networks. Eighth Annual Conference of the Cognitive Science Society (Hillsdale, NJ: Lawrence Erlbaum Associates), pp. 823–831.

UHRIG, R. E., and GUO, Z., 1989, Use of neural networks to identify transient operating conditions in nuclear power plants. *Application of Artificial Intelligence VII*, SPIE **1095**, 851–856.