Pergamon

S0098-1354(96)00003-8

Computers chem. Engng Vol. 21, No. 3, pp. 327–346, 1997

0098-1354/97 $17.00+0.00

EXPERIMENT DESIGN CONSIDERATIONS FOR NON-LINEAR SYSTEM IDENTIFICATION USING NEURAL NETWORKS

S. K. DOHERTY, J. B. GOMM* and D. WILLIAMS

Control Systems Research Group, School of Electrical and Electronic Engineering, Liverpool John Moores University, Byrom Street, Liverpool L3 3AF, U.K.

(Received 11 February 1994; revised 9 November 1995)

Abstract—Although the non-linear modelling capability of neural networks is widely accepted there remain many issues to be addressed relating to the design of a successful identification experiment. In particular, the choices of process excitation signal, data sample time and neural network model structure all contribute to the success, or failure, of a neural network's ability to reliably approximate the dynamic behaviour of a process. This paper examines the effects of these design considerations in an application of a multi-layered perceptron neural network to identifying the non-linear dynamics of a simulated pH process. The importance of identification experiment design for obtaining a network capable of both accurate single step and long range predictions is illustrated. The use of model parsimony indices, model validation tests and histogram analysis of training data for design of a neural network identification experiment are investigated. Copyright © 1996 Elsevier Science Ltd

1. INTRODUCTION

System identification procedures for parametric models frequently assume the process is both linear and time-invariant. After specification of a model structure, process input/output data is then used to calculate the model parameters by solving a least squares or maximum likelihood criterion. The effects of process non-linearities and time variations can be compensated by periodically updating the model parameters on-line. Alternatively, process non-linearities can be represented by an array of linear models spanning the operating range. While these methodologies can often provide acceptable solutions, there remain many real world systems where the identified model can be improved or where linear representation is even prone to failure. A main reason for this is that it is often not possible to adequately represent system characteristics such as non-linearity, time delay, saturation, and overall complexity within this linear modelling framework, as is the case with many chemical processes. In these situations a single model representation capable of encompassing the overall non-linear dynamic process behaviour would be preferred.

Over the past decade, there has been a resurgence of interest into the field of artificial neural networks in such diverse areas as pattern recognition, financial forecasting and signal processing (e.g. Widrow *et al.*, 1988; Hopfield, 1985). In control engineering, this interest has focused on the modelling and control of non-linear dynamic systems. Neural networks offer the promise of a modelling tool capable of emulating complex systems. A neural network can be used to approximate the non-linear input/output dynamics of a process based on a time history of process data. It is now widely accepted that the proven ability of certain neural network architectures to represent non-linear mappings, such as the multi-layered perceptron (Cybenko, 1989; Hornick *et al.*, 1989), enables a non-linear neural network model to be incorporated within a control algorithm. Indeed, the modelling abilities of neural networks have been demonstrated (Bhat and McAvoy, 1990; Bhat *et al.*, 1990; Chen *et al.*, 1990a, 1990b; Billings *et al.*, 1992) and neural networks have been used successfully to control non-linear processes, both in simulation (Saint Donat *et al.*, 1991; Hernández and Arkun, 1992; Chen and Khalil, 1992) and on-line (Evans *et al.*, 1993). Many of the proposed neural control strategies (e.g. Model Predictive Control, Internal Model Control and Model Reference Control (Hunt *et al.*, 1992)) incorporate a neural network model directly into the control system. Hence, a valid neural network process model is a keystone to the performance of these control systems, the quality of control being largely determined by the representation accuracy of the model.

The attractive benefit of the neural network approach is that the authentic development of the network enables an accurate representation to be obtained by training the network using examples of process input/output data. Hence, precise understanding and development of a rigorous mathematical model is not necessary. However, the successful training of a neural network model can require a significant amount of experimental data, and the resulting models are unreliable outside the operating regimes of this training data. Another limitation of this

* Author to whom all correspondence should be addressed

approach is that the input/output relationship of the neural network does not contain any physical knowledge about the plant which could be useful in some applications.

A system identification experiment, either for linear or non-linear modelling, involves the following design considerations:-

(i) Selecting a process excitation signal
(ii) Choosing a data sample time
(iii) Selecting the model structure (for a neural network this concerns the selection of the input layer topology)
(iv) Validating the resulting model

These elements of an identification experiment are important as they ultimately determine the validity of the resulting model. While the non-linear modelling capabilities of neural networks have been widely demonstrated, the difficulties in developing, and the limitations of, such models have not been sufficiently addressed. Little has been reported concerning the effects of the above choices and no established methodology exists for their selection. Billings *et al.* (1992) have made some important studies in this area and proposed some techniques for (iii) and (iv) but the remaining two choices were not investigated.

This paper describes investigations into the effects of the above design issues (i)–(iii) and the use of a range of techniques which can aid their selection, with the aim of developing guidelines for identification experiment design which will consequently improve the validity of the resulting neural network model. The selection techniques studied include metrics from linear system identification theory, histogram analysis and some novel metrics. Methods for neural network model validation are also examined. The investigations are carried out in a study of the development of a multi-layered perceptron neural network model of pH in a continuous stirred tank reactor (CSTR). This process is strongly non-linear and is therefore well suited to assessing the non-linear modelling capabilities of neural networks.

2. NEURAL NETWORKS FOR NON-LINEAR SYSTEM IDENTIFICATION

This section describes the neural network architecture, training algorithm, network model configuration and a typical process excitation signal used in these studies. The section provides the necessary background to the investigations described later in section 4.

2.1. Multi-layered perceptron neural network and training algorithm

A multi-layered perceptron neural network consisting of an input layer, one hidden layer and an output layer, with sigmoidal activation functions in the hidden and output layer nodes, was used throughout this investigation. Only one hidden layer was used because of the proven non-linear approximation capabilities of multi-layered perceptron networks for this case (Cybenko, 1989; Hornick *et al.*, 1989). The input/output mapping for such a network with a single output $\hat{y}$, is of the form...

$$\hat{y}=F\left[\sum_{j=1}^{n_h} w_{j1}^2 F\left(\sum_{i=1}^{n_i} w_{ij}^1 x_i + w_{0j}^1\right) + w_{01}^2\right] \tag{1}$$

where w_{ij}^l is the weight connecting the i^{th} input to the j^{th} node in the l^{th} layer, x_i is the i^{th} network input, w_{oj}^l is the bias weight for the j^{th} node in the l^{th} layer, n_h is the number of neurons in the hidden layer and n_i is the number of inputs to the network. The sigmoidal activation function, F, is given by...

$$F(z)=\frac{1}{1+e^{-z}} \tag{2}$$

The multi-layered perceptron neural network is trained using a supervised learning procedure and the back-propagation algorithm with momentum (Rumelhart *et al.*, 1986) was used in these studies. Back-propagation minimises a quadratic cost function (J) of the error between the neural network output and the target output, y, over the entire set of example data vectors...

$$J=\sum_{t=1}^{N}\left[y(t)-\hat{y}(t)\right]^2 \tag{3}$$

where N is the number of data vectors. A single data vector is selected randomly from the set of training examples and presented to the neural network. The weight update at iteration t is given by...

$$w_{ij}^l(t)=w_{ij}^l(t-1)-\eta\frac{\partial J}{\partial w_{ij}^l(t-1)}+\alpha\Delta w_{ij}^l(t) \tag{4}$$

where $w_{ij}^l(t)$ is a weight at iteration t, $\Delta w_{ij}^l(t)$ is the previous weight update, η is the learning rate and α is the momentum. The neural network is said to have been trained for one epoch when all the training examples have been used once. Often, a multi-layered perceptron must be trained for hundreds of epochs before the cost function converges.

Initialisation of the network weights and parameter values for the back-propagation algorithm were based on conventional practice in implementing the multi-layered perceptron and the experience of the authors. Initial weights were selected randomly from a uniform distribution in the interval $[-0.1,0.1]$. The learning rate and the momentum are usually chosen in the range $[0,1]$. The learning rate determines the step size of the gradient

descent back-propagation algorithm, and the momentum adds a proportion of the previous weight update to allow the search to escape from local minima which may be present in the cost function surface. In the initial stages of learning, a large learning rate and momentum can be used to accelerate the learning process. However, it is well known that for back-propagation to fully converge the learning rate and momentum should approach zero. Hence, in this work the learning rate and momentum were exponentially decreased towards zero from large initial values ($\eta = 0.9$, $\alpha = 0.6$) using an exponential time constant of 200 epochs.

Input/output data to the multi-layered perceptron was conditioned using a widely used technique of scaling each network input and output variable to a fixed range. A range of slightly less than [0,1] is usually used for data scaling because the output of the sigmoid activation function, eqn. (2), asymptotically approaches zero or one hence, a range of [0.1,0.9] was used in these studies. Each scaled data value is then assigned to a single input or output neuron of the multi-layered perceptron. This data conditioning method was used in all of the investigations reported, except for those in section 4.4.3 where an alternative data conditioning method is described. During training the back-propagation cost function was calculated using scaled target output data, and during recall the network output was re-scaled back to the original variable range to obtain real world prediction values.

2.2. *Network configuration for non-linear modelling*

The input/output mapping of equation (1) is a static function, and hence, if a multi-layered perceptron neural network is to be employed as a process model, dynamics must be incorporated into the network. A general model structure which is suitable for this purpose is the NARX (Non-linear Auto-Regressive eXogenous) model (Leontaritis and Billings, 1985) which, for a SISO system, is defined as...

$$y(t)=f\Big[y(t-1),\ldots,y(t-n_y),u(t-k),\ldots,u(t-k-n_u+1)\Big]+e(t) \quad (5)$$

where $u(t)$ and $y(t)$ are the process input and output at time t, $f[]$ is a non-linear function to be identified, $e(t)$ is the equation error, k is the process deadtime ($k \geq 1$) and n_u and n_y are the number of delayed process inputs and outputs included in the model.

A neural network is configured in a NARX model structure by assigning the network input vector $x = [y(t-1),\ldots, y(t-n_y),u(t-k),\ldots, u(t-k-n_u+1)]$ and training the network to provide a one step ahead prediction of the process output at time t, $\hat{y}(t)$. When the network has been trained, it can then be used to predict $\hat{y}(t+1)$ based on process data available at time t.

2.3. *Random amplitude process excitation signal*

A neural network should be trained with dynamically rich data which covers the whole of the required process operating region. A random amplitude signal is commonly used as the process excitation signal to generate open loop data for network training. This signal consists of a uniformly distributed random variable applied to the process input at each clock period and is more likely to exercise the process over the desired operating range than a binary signal (Pottmann and Seborg, 1992), such as a pseudo-random binary sequence (PRBS) which is widely employed for linear system identification. A random amplitude signal is specified by its clock period, which should be a multiple of the sample time so that the process input is constant between consecutive samples, and by its amplitude range, which may be expressed as a percentage maximum deviation from a steady state value.

3. SIMULATED CSTR PROCESS

In the CSTR investigated, shown in Fig. 1, acetic acid (CH_3COOH) of concentration C_A flows into the tank at flow rate F_A, and is neutralised by sodium hydroxide (NaOH) of concentration C_B which flows into the tank at rate F_B. The process equations and parameter values used are given in the appendix. A simulation of the process using this first principle model was implemented in the Advanced Continuous Simulation Language (ACSL). F_A was held constant and a random amplitude signal was superimposed on a steady state, $\bar{F}_B$, to provide dynamic input/output data (F_B and pH) for neural network training and model validation.

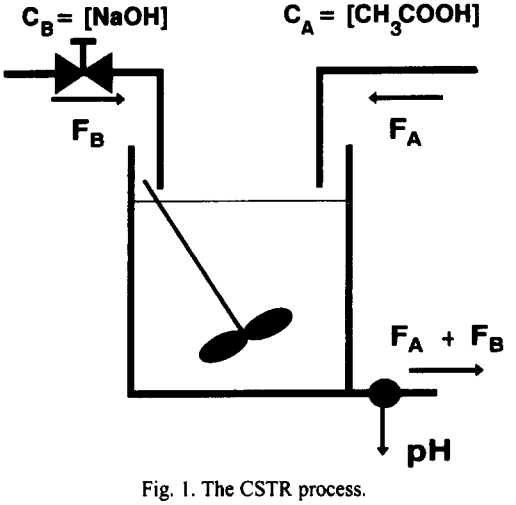


Fig. 1. The CSTR process.

4. DESIGN ISSUES IN NEURAL NETWORK MODEL DEVELOPMENT

Investigations into the design of a neural network system identification experiment are described in this section in an application of a multi-layered perceptron neural network to model the simulated pH CSTR process. Techniques which may assist the decisions required to be made in the development of a neural network model and methods for model validation are studied. The investigations described are not exhaustive, since to undertake a study of all possible combinations of the design parameters would be an extremely involved task. Therefore, decisions were made, based on the results obtained, to confine the studies to a more tractable form. The design issues and the techniques used are discussed in a more general framework to extend their applicability to the identification of other processes.

A starting point for the parameter values of the excitation signal and data sample time were taken from previous work for neural network identification of the pH CSTR process (Bhat and McAvoy, 1990). These were a 10% random amplitude signal with a clock period of 240 s and a sample time of 24 s. Networks were trained with data generated from the CSTR simulation with these settings, unless otherwise specified. In order to make the training conditions as equitable as possible, the data length was fixed at 1000 input/output training vectors and neural network training was terminated after 1000 epochs, since after 1000 epochs the learning rate and momentum were decayed to near zero values (see section 2.1). In all cases it was observed that this was sufficient to obtain convergence of the network weights to a minimum of the cost function, equation (3), and that further training did not decrease the cost function.

4.1. Model structure selection

It is important to consider the NARX model structure for each individual problem because the prediction accuracy can be degraded by inappropriate structure specification. For the SISO NARX model of equation (5), the structure is specified by the values of k, n_u and n_y. If the system deadtime, k, is unknown it can be included in the search for a model structure or may be determined experimentally (e.g. from step responses). Although, for a fixed deadtime the model structure is specified by two parameters, n_u and n_y, to perform an exhaustive search of the combinations of n_u and n_y is impractical, since to evaluate structures of sizes $n_u = 1,\ldots,N_U$, $n_y = 1,\ldots,N_Y$ would necessitate the training and testing of $(2^{N_U}-1)(2^{N_Y}-1)$ neural networks (e.g. $N_U = N_Y = 5$ gives 225 possible model structures). This effort can be greatly curtailed by setting $n_u = n_y = n$, where n is termed the model order, giving n possible model structures as is common in linear system identification (Isermann, 1980).

There are two potential pitfalls in the selection of the NARX model order. The first is assigning a model order which is too low to describe the system dynamics. Since the neural network performs a static non-linear expansion of the NARX model input terms, equation (1), it cannot generate any system lags of higher order than the model order. In this case, the neural network is deprived of important information which it cannot compensate for and hence, the resulting model will be deficient.

If the model order is increased, it should provide an improvement in the neural network model accuracy. Therefore, the potential problem of under-parameterising the NARX input vector could be avoided by choosing a large model order. However, this may not be the best solution because the performance of a neural network can also be adversely affected by over-parameterisation of the NARX input vector. For neural networks, over-parameterisation often results in inadequate generalisation abilities-the network can perform well when tested with the training data, but has a poor performance when tested with novel data-because the neural network can overfit the training data. This can result in a neural network functioning like a look-up table of the training data rather than representing the underlying system which produced the data. Furthermore, enlarging a network increases the number of local minima in the cost function surface. This means that the learning algorithm may have greater difficulty in converging to the desired global minimum, and the training of the network is more sensitive to the initial state of the weights.

Model parsimony is recognised as an important criterion for a dynamic model because, although the underlying system may be very complex, from a set of candidate models a parsimonious model will contain the minimum number of parameters which approximates the system performance. The choice of a parsimonious model is a well established principle of linear system identification and consequently complicated dynamic systems are frequently represented by model orders lower than the actual physical system. From the above discussion on neural network parameterisation, it is evident that model parsimony is an important issue and a particularly relevant criteria for the selection of a neural network model.

The number of nodes in the neural network hidden layer also affects model parsimony. This study did not attempt to optimise the number of hidden layer nodes. Pollard *et al.* (1992) reported that, provided a network incorporated sufficient nodes in the hidden layer, the mapping accuracy of a neural network was relatively insensitive to the number of hidden layer nodes over a wide range. This observation was also noted in these

studies and thus, the size of the hidden layer was fixed experimentally at 15 which was sufficient to enable convergence of all networks tested. Fixing the size of the hidden layer also enables the effects of increasing the model order to be more clearly observed, as any improvements in the network prediction accuracy will be more directly attributed to changes in the input layer rather than the hidden layer.

4.1.1. Comparing neural network performance to learn training data. To establish a suitable NARX model order for a particular system, neural networks of increasing model order can be trained and their performance on the training data compared using the loss function (or mean squared error), *LF*...

$$LF = \frac{1}{N} \sum_{t=1}^{N} \varepsilon^2(t) \tag{6}$$

where the one step ahead prediction error, $\epsilon(t) = y(t) - \hat{y}(t)$ and N is the data length. Figure 2a shows the LF on the same training data for neural network models with different model orders $n = 1,..., 10$ and a delay $k = 1$. A delay of one sample between input and output for the CSTR process was selected based on knowledge that the process was perfectly mixed. This was also confirmed in step response tests on the process. The 10th order model exhibits the lowest *LF*, however, this model may not be the best choice, because there is a trade off between the model complexity (i.e. size) and accuracy. A small decrease in the *LF* may be rejected if it is at the expense of enlarging the model size. Thus, the decision procedure for selecting a parsimonious model using the *LF* is to decide for each increase in model order whether any reductions in the *LF* are worth the expense of a larger model. This decision can be assisted by examining the trend in the *LF* (Fig. 2b). This indicates a second order model which causes a large reduction in the *LF* compared to a first order model but subsequent decreases in the *LF* caused by increasing model orders are not comparable.

The difficult trade off between model accuracy and complexity can be clarified by using model parsimony indices from linear estimation theory, such as Akaike's Final Prediction Error (AFPE) and Akaike's Information Criterion (AIC) (Akaike, 1974)...

$$AFPE = \frac{1 + \frac{n_w}{N}}{1 - \frac{n_w}{N}} LF \tag{7}$$

$$AIC = \ln\left(\frac{N}{2} LF\right) + \frac{2n_w}{N} \tag{8}$$

where n_w = number of model parameters (weights in a neural network). Hence, both the *AIC* and *AFPE* are weighted functions of the *LF* which penalise for reductions in the prediction errors at the expense of increasing model complexity (i.e. model order and number of parameters). Strict application of the *AFPE*

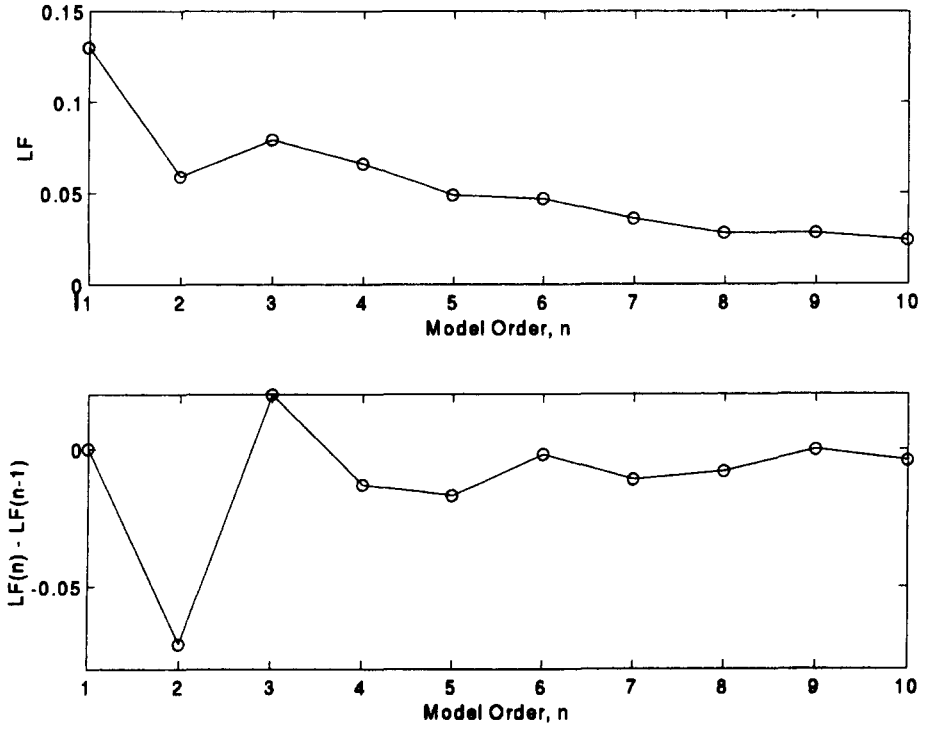


Fig. 2. Comparison of neural networks with different NARX model orders. (a) loss functions, (b) trend in loss functions.

and *AIC* means that the model structure with the minimum *AFPE* or *AIC* is selected as a parsimonious structure. However, in practice, engineering judgement may need to be exercised when using these metrics.The corresponding *AFPE*s and *AIC*s for the *LF*s in Fig. 2 are shown in Fig. 3 where it is observed that up to a 6th order model both indices point to the selection of a 2nd order model as having the minimum AFPE and AIC. However, a strict application of the indices would select a 10th order model because this exhibits the lowest of both indices for all the model structures compared. Based on engineering judgement, a 10th order model would be considered to be too complex and a 2nd order model would be preferred without significant loss of accuracy. Although, in this case, the *AFPE* and *AIC* do not provide a clear indication of a particular model, the interpretation of the *AFPE* and *AIC* results described does provide further support for the choice of a 2nd order model indicated by the *LF*.

Once the network input topology has been chosen, the number of hidden layer nodes could then be improved by comparing networks trained with different complements of hidden nodes, using the *LF*, *AFPE* and *AIC* to select a parsimonious architecture.

4.1.2. Comparing neural network generalisation capabilities. A neural network model which has identified the system dynamics should be capable of predicting the system response to a range of test signals. Thus, an alternative technique for choosing a NARX model order is to compare the performance of different neural network models on data sets that were not used during the training phase. This method can also be used for model validation as discussed in section 4.4.1.

When several models are tested on a range of different test signals, it is desirable to derive a single figure of merit for each model so that model accuracy can be easily and fairly compared. Suppose M neural network models are to be compared by testing them using L test signals. A single performance index (*PI*) for each model can be defined as the sum of the loss functions for each test signal...

$$PI_m = \sum_{i=1}^{L} LF_m^i,\ m = 1,\ldots,M \tag{9}$$

where m is the index of the model tested and LF_m^i is the loss function of the m^{th} model when tested with the i^{th} signal.However, this simple summation does not take into account the relative magnitudes of the loss functions compared to other models. An alternative index, which enables a comparison of the relative performance of different models on the same test signals, can be formulated by weighting the loss functions by the sum of all the model loss functions for each test signal. This gives the normalised performance index (*NPI*)...

$$NPI_m = \sum_{i=1}^{L} \frac{LF_m^i}{\sum_{m=1}^{M} LF_m^i}. \tag{10}$$

Figure 4 shows the performance and normalised per-

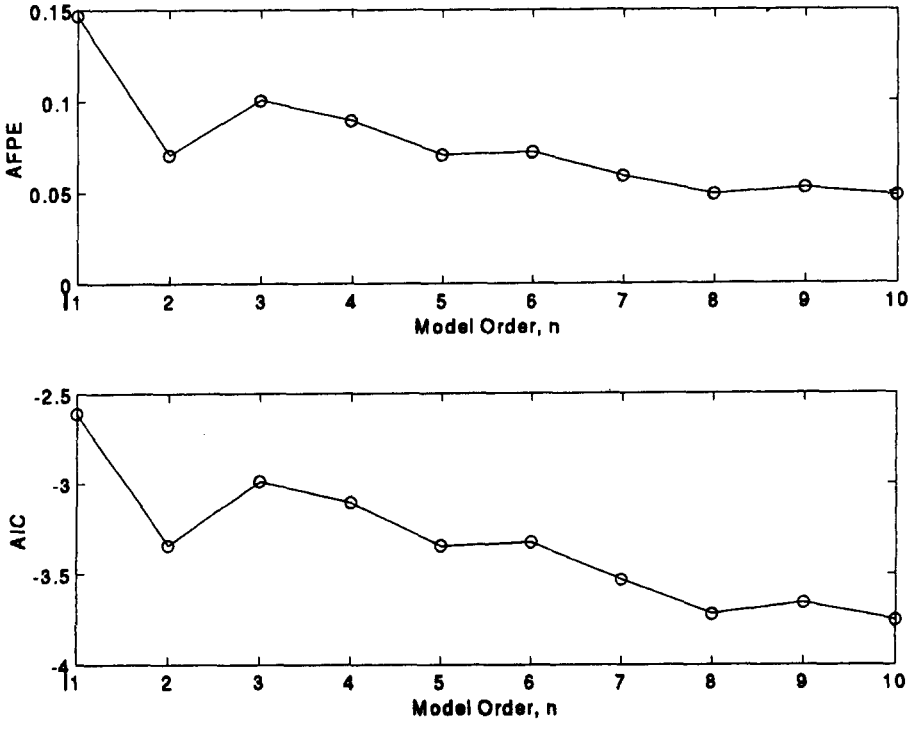


Fig. 3. Comparison of neural networks with different NARX model orders. (a) AFPE, (b) AIC.

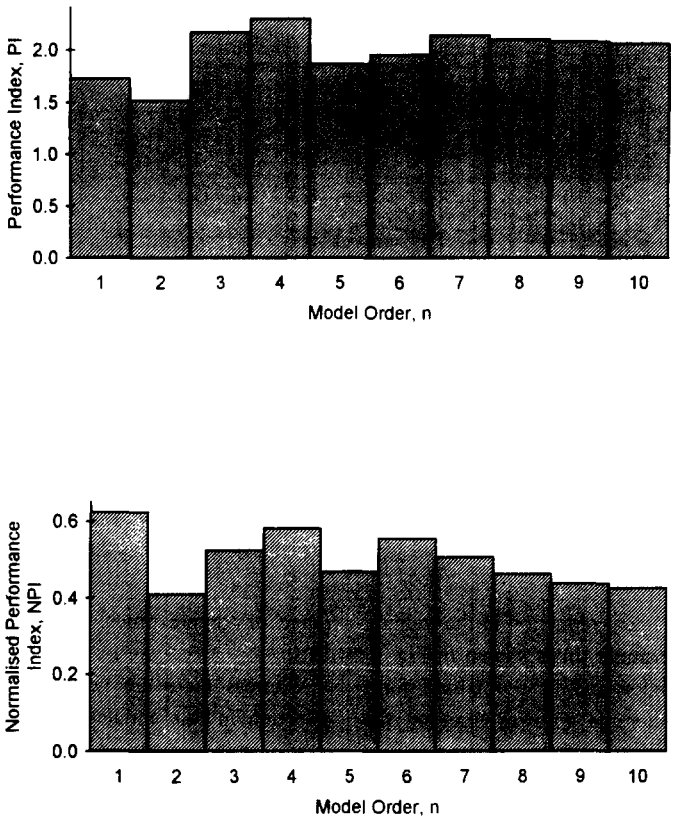


Fig. 4. Comparison of neural networks with different NARX model orders. (a) performance index, (b) normalised performance index.

formance indices for the ten neural network models used in the previous section when tested on five test signals consisting of different random amplitude signals and steady state data. The test signals used were the same as those designed for model validation (section 4.4.1). The performance index clearly indicates a second order model structure as exhibiting the best generalisation performance on the test signals used. The choice of a second order model is also indicated by the normalised performance index.

Comparing the generalisation capabilities of different models, in conjunction with the *LF*, *AFPE* and *AIC*, can assist the choice of a neural network model. Based on the results presented for the two approaches, a second order model would be chosen as a parsimonious neural network model structure for representing the pH process.

4.2. Data sample time

The choice of data sample time should, by the sampling theorem, correspond to a sampling frequency of at least twice that of the highest process frequency range to be identified. Other than this fundamental requirement for the sample time, little is known at present of how the sample time affects the performance of a neural network model. This section describes investigations into the effects of the data sample time on neural network model performance for the pH process when trained with data generated by a conventional random amplitude excitation signal.

Figure 5 shows the evolution of the training cost function for identically structured networks (2^{nd} order NARX models with 15 hidden layer nodes as chosen in section 4.1) which were trained with the same random amplitude excitation signal, but sampled at different rates. The training cost function is an indicator of how well the learning phase is proceeding and it is often monitored to determine a suitable point for terminating training or to flag any convergence problems. Hence, Fig. 5 suggests that the network learns the training data more accurately for faster sample times. However, the training cost function is akin to the loss function on the training data, in that neither give an indication of a network's generalisation ability, and therefore the magnitude of the final value to which the cost function converges should not be taken as an indication of model validity. Figure 6 shows the performance index and the normalised performance index for these neural network models when tested with the cross-validation signals described in section 4.4.1. In calculating the performance indices, all models were tested on the same window of real time and consequently the number of testing vectors for each of the test signals was appropriately increased for decreasing sample times.

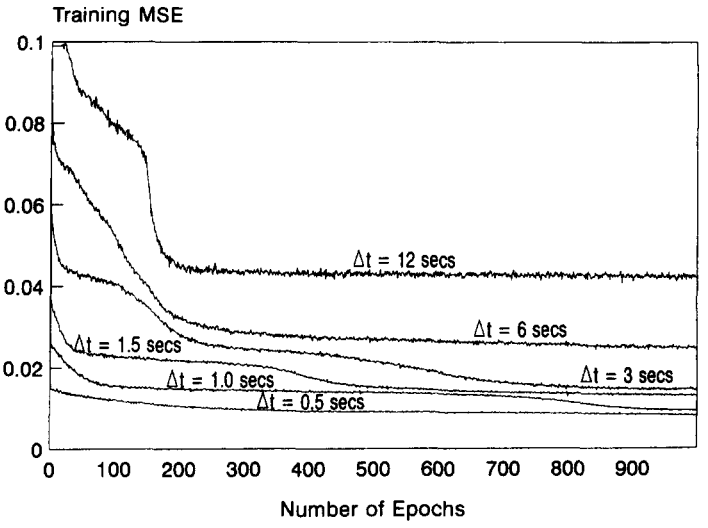


Fig. 5. Training loss functions for 2nd order neural network models employing different sample times.

In contrast to the consistent reduction in the training cost function for faster sample times shown in Fig. 5, both the performance index and its normalised form (Fig. 6) illustrate that there is a lower limit on the selection of sample time. A sample time of 1 s achieves the lowest values of the performance indices for the sample times investigated. Figure 6 also shows that satisfactory network performance can be achieved with a range of sample times, in this case between 0.5 and 12 s.

Whereas there are no restrictions on sample time selection in this simulation exercise, the presence of process noise can impose limits on the choice of sample time for the modelling of a real plant. Reducing the data sample time increases the effect of high frequency process noise which can result in a poor model. Another

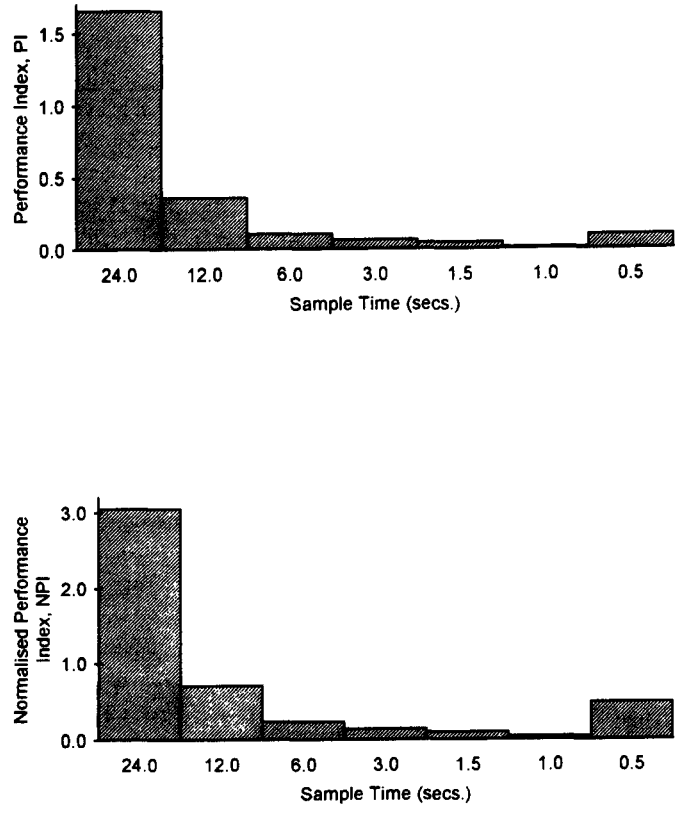


Fig. 6. Comparison of 2nd order neural network models identified with different sample times. (a) performance index, (b) normalised performance index.

application specific factor, which may restrict the choice of sample time in practice, is computational constraints because, if the identified model is to be implemented on-line, the calculations must be performed in less time than the sample interval. Often it is required that calculations are completed in a small fraction of the sample interval, such as in a model based control algorithm. These practical considerations should be taken into account when choosing a sample time, in conjunction with the main objective of achieving an accurate process model.

4.3. Excitation signals

The objective of process excitation is to generate input/output process data which contains sufficient information for a neural network to identify the non-linear process dynamics over the entire operating range. As described in section 2.3, a random amplitude signal is commonly employed to achieve this. However, there is no reason why this signal should achieve adequate excitation of the non-linear dynamics of all processes. Additionally, there is no established method for assessing training data adequacy at present. The effects of excitation signal for different data sample times are investigated in this section. The use of histogram analysis for examining the process output data density is proposed as a simple method for assessing the adequacy of the training data.

The strong non-linearity inherent in the pH process is characterised by its steady state titration curve (Fig. 7), where the gain changes by a factor of 150 between operating points A and B. One of the obstacles to accurately modelling such a process can be obtaining output data in this high gain region for network training. When the pH process is excited by a standard random amplitude signal, little output data is generated in the vicinity of operating point A, where the process gain is a maximum. The bowl shaped data distribution of Fig. 8 illustrates the lack of output data between pH 7 and 10, and a neural network model which results from training with such data can have large prediction errors in this region.

One practical way of improving the uneven distribution of the training data is to force the signal through the region of maximum process gain on each clock pulse. The resulting 'forced' random amplitude signal, depicted in Fig. 9, has a uniformly distributed input in the two intervals above and below a threshold level which was chosen as the process input at maximum steady state gain. While the data distribution still appears to be very uneven, there is a threefold increase in the data density between pH 7 and 10 when compared to the output data density generated by a standard random amplitude signal (Fig. 8).

A further improvement in the output data distribution (Fig. 10) can be realised by modifying the forced random amplitude signal, by reducing its amplitude for a fraction of the identification experiment. Half way through the identification experiment the magnitude of this 'modified' random amplitude signal was reduced from 10% to 0.5%. The central value of the modified random amplitude signal was chosen as the input value which corresponded to the point of maximum process gain, and hence, the reduced amplitude of the input signal provides additional output data in this high gain region.

Three identically structured networks (2^{nd} order NARX models with 15 hidden layer nodes) were trained using a standard, a forced and a modified random amplitude signal respectively and this was repeated with different data sample times. Figure 11 shows the performance indices for the trained networks when evaluated using the same test signals of section 4.4.1. The performance index (Fig. 11a) illustrates that network model accuracy can be improved by training with

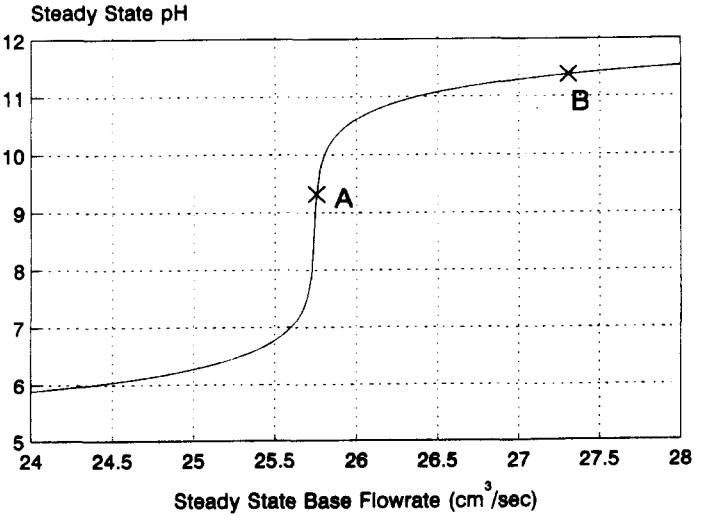


Fig. 7. CSTR process titration curve.

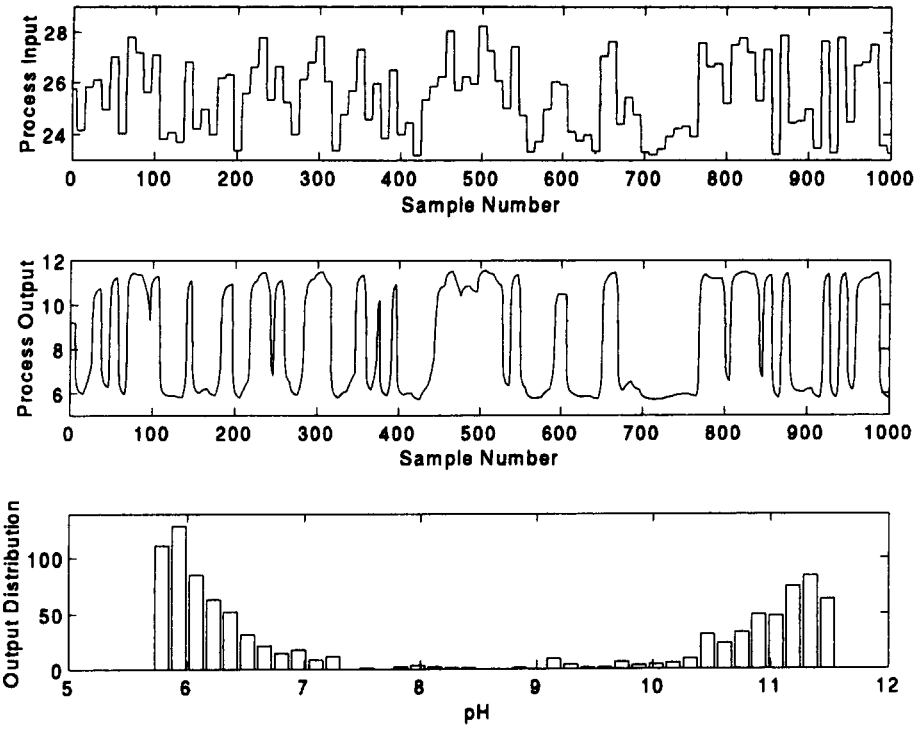


Fig. 8. Process input, output and output data histogram for a conventional random amplitude excitation signal.

alternative process excitation signals. The extent of the improvement corresponds to the provision of additional training data in the pH 7–10 region, where data was previously sparse, with the lowest performance index for each sample time occurring for networks trained with the modified random amplitude signal. This general trend is also reflected in the normalised performance index (Fig. 11b), although the forced random amplitude signal for a 24 s sample time does not show a decrease in the index. This case illustrates the difference between the two performance indices. While the network trained with the forced random amplitude signal had lower loss functions

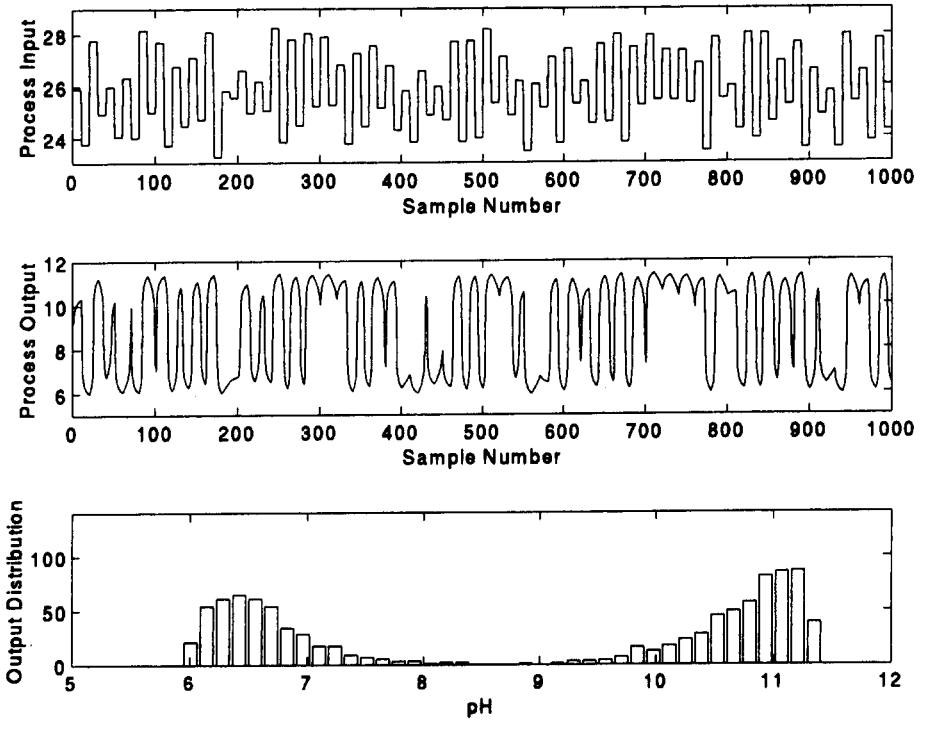


Fig. 9. Process input, output and output data histogram for a forced random amplitude excitation signal.

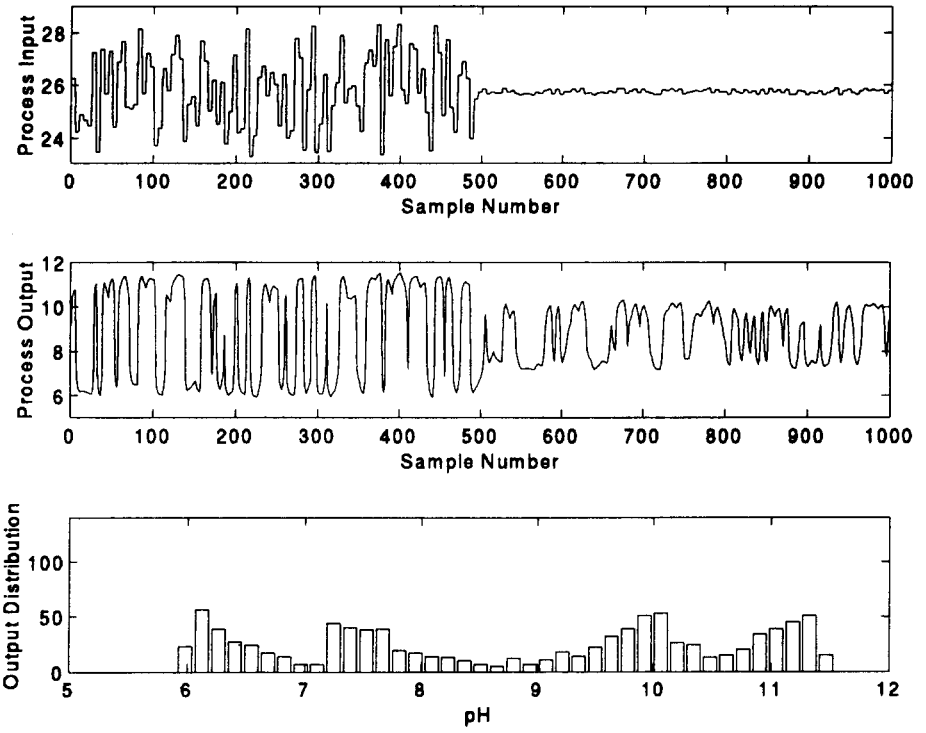


Fig. 10. Process input, output and output data histogram for a modified random amplitude excitation signal.

than the standard excitation signal for four of the five test signals, there was a relatively high loss function for the steady state test data which is highlighted in the normalised performance index.

Implementing the forced or modified random amplitude signals on-line would require knowledge of the process steady state characteristic. Indeed, the modified random amplitude signal may not be implementable on-line since the plant actuators may not have the necessary rangeability. Nevertheless, some adjustment of the excitation signal should be possible to enhance the training data distribution in deficient regions indicated by the process output data histogram and thus, effect an improved neural network model.

4.4. Model validation

The purpose of model validity tests is to give an indication of the adequacy of a fitted model. A poor neural network model may occur for many different reasons, such as incorrect model order selection or insufficient hidden layer nodes. More fundamentally, the identification data may be deficient in some way, which may result from coloured process noise, an unsuitable data sample time or process excitation signal. This section presents three techniques for validating neural network models-cross-validation, correlation tests and multi-step ahead prediction. Results are presented to demonstrate the use of these methods and the improvements in model performance which can be achieved by the design of a good identification experiment.

4.4.1. Cross-validation. Model accuracy over the training data set is not reliable evidence of model validity since it gives no indication of a model's generalisation ability. Hence, a common method of validating a neural network model is to split a data set into two, using one part to train the network and the other part to validate it. However, if the data set has been collected when process operating conditions are comparable, then the test set may be so similar to the training set, with regard to operating range and frequency content, that the model's generalisation abilities are not properly tested. For instance, the process may often operate at or near steady state conditions, but these low frequencies may not be present in the testing data. This can be avoided by testing the neural network using several disparate test signals which test the model over different operating and/or frequency ranges. These test signals can be realised by adjusting the magnitude range and/or clock period of the random amplitude excitation signal. This testing of a neural network's generalisation capabilities is referred to as cross-validation in the system identification field.

In this work, the test signals used for cross-validation of neural network models were generated by three different random amplitude signals with clock periods of 72, 120 and 240 s, a random amplitude signal whose amplitude was greatly reduced thus, testing the model predictions solely in the region of maximum process gain (operating point A in Fig. 7), and steady state data. These disparate test signals were selected to compre-

hensively test the generalisation capabilities of the neural networks, which is a fundamental requirement of a useful dynamic model.

One approach for assessing the generalisation capabilities of neural networks when tested on a range of test data sets is to use the performance and normalised performance indices of eqns (7) and (8), respectively. This is a flexible method for examining neural network performance and its use for selection of model structure, sample time and excitation signal has been demonstrated in sections 4.1.2, 4.2 and 4.3. The effects of the design issues considered in this paper on the generalisation capabilities of a neural network model are illustrated in Table 1, which shows the performance indices for two network models trained with different parameter values in the identification experiments. Network model 1 was a 5th order model trained with the initial settings for excitation signal and sample time and network model 2

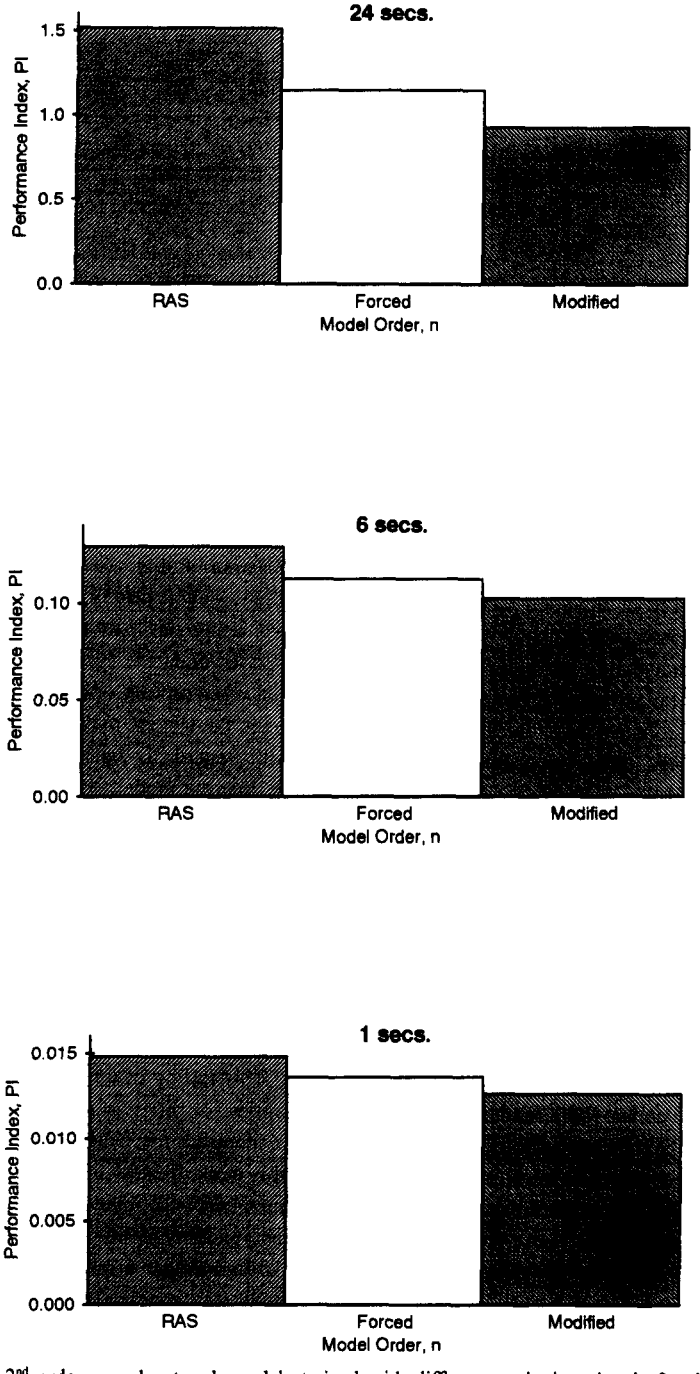


Fig. 11. Comparison of 2nd order neural network models trained with different excitation signals for three sample times. (a) performance index.

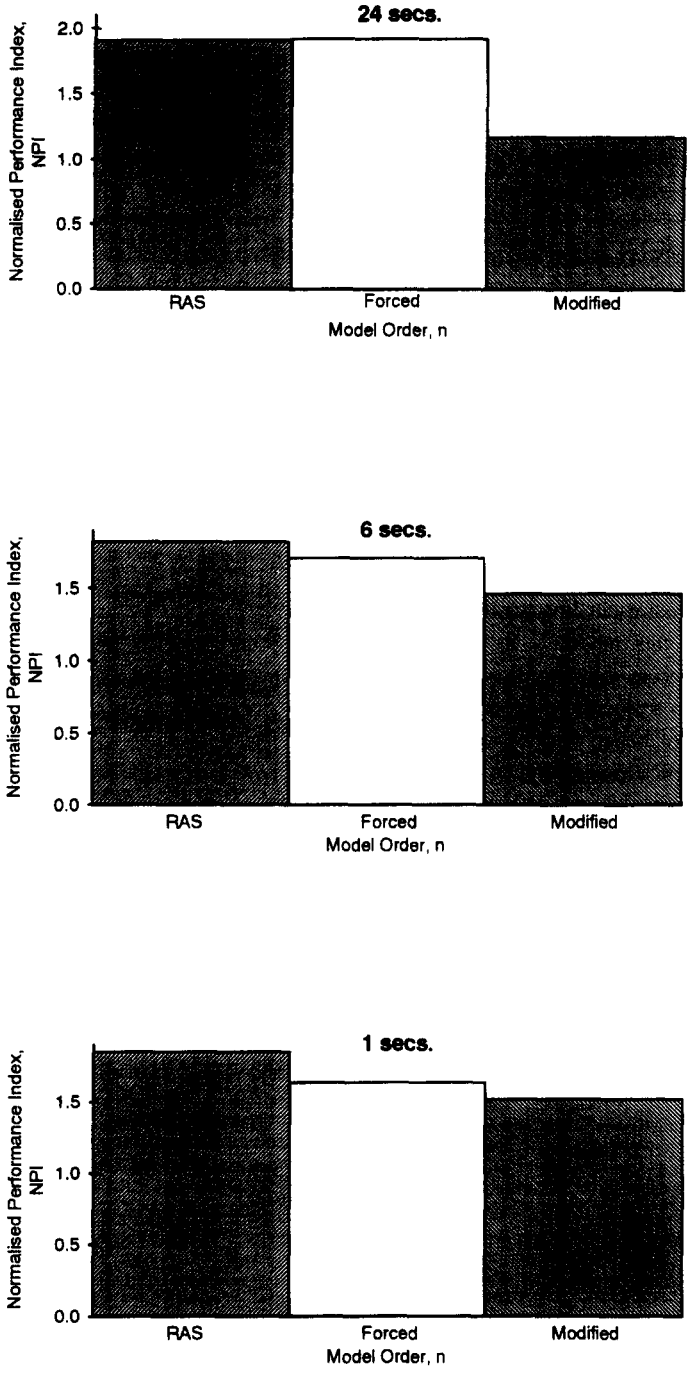


Fig. 11. Comparison of 2nd order neural network models trained with different excitation signals for three sample times. (b) normalised performance index.

Table 1. Comparison of two neural network models with different identification experiment parameters

Network Model	1	2
Identification experiment parameters	Standard random amplitude signal, 240 sec clock period 24 sec sample time 5^{th} order	Forced random amplitude signal 240 sec clock period 1 sec sample time 2^{nd} order
Performance Index, PI	1.52	0.014
Normalised Performance Index, NPI	4.7	0.3
Max. prediction error	3.4 pH units	0.84 pH units

was a 2nd order model trained with a forced random amplitude signal and a 1 sec sample time. The design parameters for network model 2 were chosen based on the results described in sections 4.1, 4.2 and 4.3. It is clear that a good choice of the design parameters can result in notable improvements in the process representation accuracy of a neural network model as shown by the significantly lower performance indices of network model 2 compared to network model 1. These improvements may also be achieved with a much smaller network as demonstrated in these results.

A widely used alternative to the performance indices is to visually examine the performance of a model in plots of the model and process output on test data. However, it can be illustrated that this alone is not a reliable test. Figure 12a shows the predictions and actual process output of network model 1 on a test random amplitude signal. From this graph it would appear that the model is able to generalise satisfactorily. However, an examination of the corresponding prediction errors, Fig. 12b, shows that this model is in fact a poor representation of the process with large prediction errors exceeding 3 pH units in a process output range of 5–12 pH units. The effects of better choices for the design parameters in the identification experiment are apparent in Fig. 13 which shows typical predictions of network model 2 on an unseen test random amplitude signal. The prediction errors are noticeably smaller than that of the larger 5th order neural network model (Fig. 12) and demonstrate the improved prediction accuracy of this model.

4.4.2. Correlation tests

A set of statistical tests have been proposed as an alternative method of verifying the adequacy of a model. They were introduced by Billings and Voon (1986) for non-linear models and extended to neural networks (Chen *et al.*, 1990a). For a perfectly unbiased model, its prediction errors (ϵ) should be uncorrelated with all linear and non-linear combinations of past values of themselves and of past model inputs (u). Five tests have been proposed as a test of non-linear model adequacy (Billings and Voon, 1986)...

$$\Phi_{\epsilon\epsilon}(\tau)=E[\epsilon(t)\epsilon(t-\tau)]=\begin{cases}0, \forall \tau \neq 0\\ 1, \tau=0\end{cases}$$

$$\Phi_{u\epsilon}(\tau)=E[u(t)\epsilon(t-\tau)]=0, \forall \tau$$

$$\Phi_{u^2\epsilon}(\tau)=E[(u^2(t)-\bar{u}^2)\epsilon(t-\tau)]=0, \forall \tau$$

$$\Phi_{u^2\epsilon^2}(\tau)=E[(u^2(t)-\bar{u}^2)\epsilon^2(t-\tau)]=0 \forall \tau$$

$$\Phi_{\epsilon(\epsilon u)}(\tau)=E[\epsilon(t)\epsilon(t-1-\tau)u(t-1-\tau)]=0 \forall \tau \geq 0 \quad (11)$$

where $E[X]$ is the expected value of X, τ is the lag and $\bar{u}^2(t)$ is the mean of $u^2(t)$. To implement these tests, u and

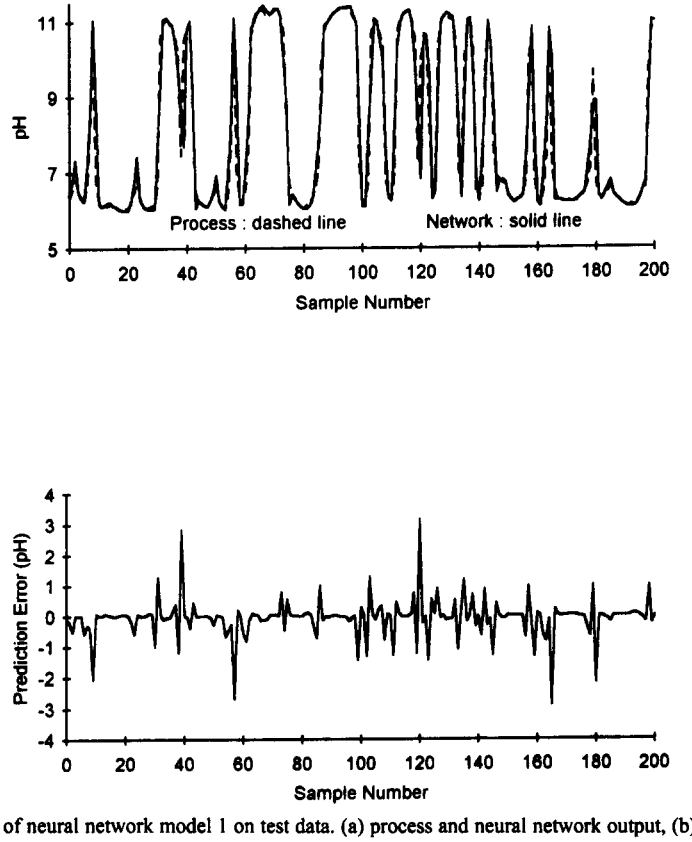


Fig. 12. Predictions of neural network model 1 on test data. (a) process and neural network output, (b) prediction errors.

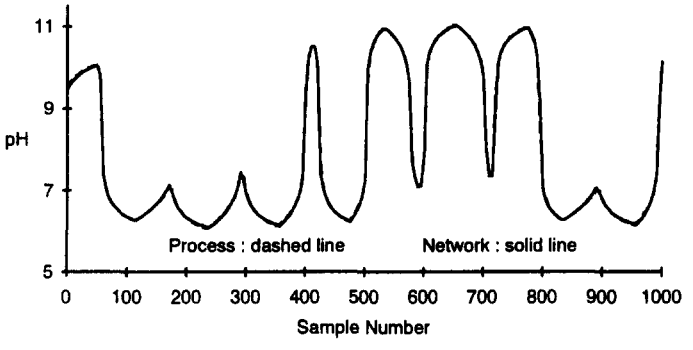
ϵ are normalised to give zero mean sequences of unit variance. The sampled cross-correlation function between two such data sequences $v(t)$ and $r(t)$ is then calculated as...

$$\hat{\Phi}_{vr}(\tau)=\frac{\sum_{t=1}^{N-\tau} v(t)r(t+\tau)}{\left[\sum_{t=1}^{N} v^2(t)\sum_{t=1}^{N} r^2(t)\right]^{1/2}} \qquad (12)$$

where N is the number of data samples. The tests are satisfied if they fall within the 95% confidence limits from Normal distribution tables of $\pm 1.96/\sqrt{N}$.

Practical results from the application of these correlation tests suggest that they should not be interpreted rigidly (see for example Chen *et al.*, 1990a, 1990b). This is because of the assumption of additive white noise in the model structure of equation (5) and the fact that the tests were derived for a class of analytic non-linear systems which is a subset of a wider class of non-linear systems that neural networks can approximate. Nevertheless, the tests can be usefully applied to NARX structured neural networks to give an indication of the validity of the model. In addition it has been demonstrated that the tests may also provide an indication of any deficient terms in the network input structure signalled by test failures at a particular lag (Billings *et al.*, 1992).

These correlation tests are intended for model validation of systems subject to random disturbances. Therefore, to implement these tests in the simulation studies and to make the simulation more realistic, a zero mean Gaussian distributed random sequence, representing pH measurement error, was added to the output of the process. The variance of the noise sequence was set to give an output signal-to-noise ratio of 10%, which was considered to be in the region of a typical value that may occur in practice. Initially, neural network models of 1st to 5th order were trained with the same noisy data set collected from the simulation using the initial values for the design parameters. Results from applying the correlation tests to the trained networks were inconclusive with four of the five cross-correlation functions exceeding the confidence limits at seemingly random values of lag. Figure 14 shows the correlation tests for the 5th order model structure, and is typical of the test results observed for the other model orders. For clarity, only four of the five test results are shown. These results confirm the poor validity of these networks which was caused by the inappropriate selection of the experiment design parameters for training data excitation signal and data sample time.



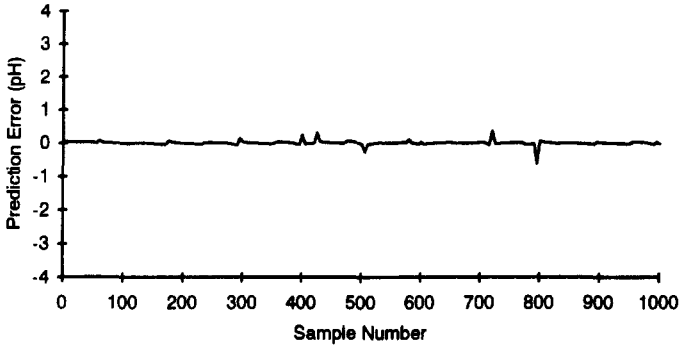


Fig. 13. Predictions of neural network model 2 on test data. (a) process and neural network output, (b) prediction errors.

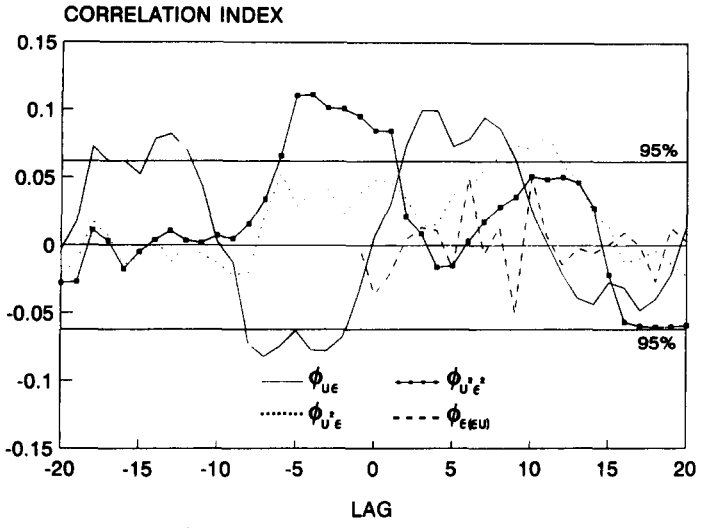


Fig. 14. Correlation test results for a 5th order neural network model identified with initial design parameter settings.

Figure 15 depicts four of the five correlation test results for a network trained with the experiment design parameters set based on the results described in previous sections. The network was configured as a 2nd order NARX model and the training data was generated by a forced random amplitude excitation signal sampled at 1 sec intervals. While one of the tests does exceed a 95% confidence limit, these results further substantiate that this neural network model is more representative of the process dynamics than the earlier correlation tests for the 5th order neural network (Fig. 14). Additionally, the correlation test results for this 2nd order model were better than those for other 1st–5th order neural network models which were also trained with same data. Hence, these statistical tests reinforce confidence in the validity of this neural network model.

4.4.3. Multi-step ahead prediction. Throughout this paper, neural network models have been assessed on their capability to predict the current value of the process output based on past values of the process input and output. The networks were trained in this one step ahead configuration and it is therefore natural to validate them accordingly. In some circumstances, however, it may be necessary for the model to predict more than one sample interval into the future. For instance, it may be desirable to use the model independently from the process so that a controller could be tuned in simulation. Multi-step ahead prediction is also necessary to implement neural predictive control schemes (Hunt *et al.*, 1992), which minimise a cost function over a moving horizon of several steps ahead. In these situations it is beneficial to also validate a neural network model by examining its multi-step ahead prediction performance.

Long range neural network predictions are made by feeding back the delayed neural network output, $\hat{y}$, to the network input. Thus, from equation (5), to predict P

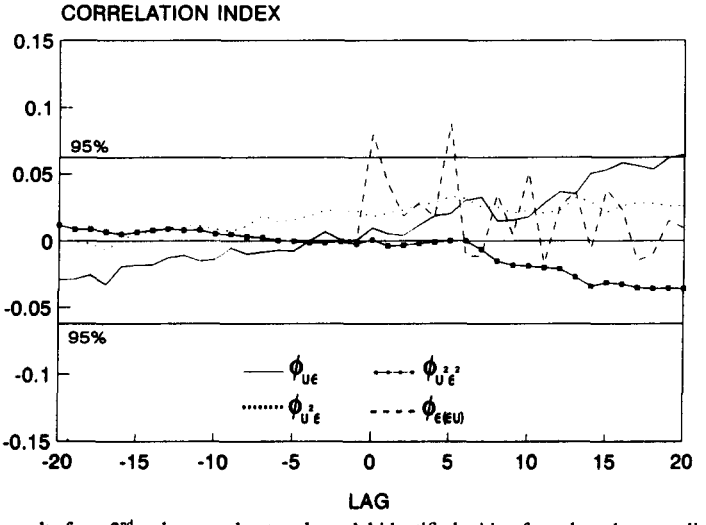


Fig. 15. Correlation test results for a 2nd order neural network model identified with a forced random amplitude signal and 1 s data sample time.

network input. Thus, from equation (5), to predict P steps ahead at time t, the neural network makes P separate one step ahead predictions, where the i^{th} prediction would be of the form...

$$\hat{y}(t+i)=f(\hat{y}(t+i-1),\ldots,\hat{y}(t+i-n_y), u(t+i-k),\ldots,u(t+i-k-n_u+1)) \quad (13)$$

For a P step ahead prediction, the neural network is recurred $P-1$ times. This feedback operation can cause an accumulation of prediction errors and a consequent deterioration of prediction accuracy. It is possible for the multi-step ahead prediction errors to become so large as to render the model unusable for any practical purpose.

The accumulation of prediction errors when operating a neural network model recursively can be reduced by using an alternative data conditioning method of spread encoding (Gomm *et al.*, 1994). In contrast to the scaling technique used throughout the investigations described above, each data value is represented as the mean value of a sliding Gaussian pattern of excitations, in the range [0.1,0.9], over several nodes at the network input and output. To spread encode a continuous valued variable $r \in [r_{min}, r_{max}]$ to N_s network nodes, each node is assigned a value, a_i, linearly spaced by a distance, δ, within the range of r. The excitation of each node, x_i $i=1,\ldots N_s$, is found by integrating a Gaussian probability density function, $\phi(a-r)$, over each class interval...

$$x_i(r)=\frac{1}{a_i}\int_{a_i-\delta/2}^{a_i+\delta/2}\phi(a-r)\,da \quad (14)$$

The algorithm is formulated so that decoding of the network output back to the original variable range is achieved as a normalised weighted summation of the output node activations...

$$\hat{y}=\frac{\sum_{i=1}^{N_s}a_i x_i(r)}{\sum_{i=1}^{N_s}x_i(r)} \quad (15)$$

To demonstrate the multi-step ahead prediction performance of neural networks trained with different sample times, three neural networks were trained in a 2nd order NARX configuration; one with each input/output data value distributed over six nodes using the spread encoding technique (this resulted in a network with 24 input nodes and six output nodes); the other neural networks with the input/output data conditioned using the conventional scaling method. The training data for the spread encoded neural network was generated by a forced random amplitude signal sampled at 1 sec intervals and the training data for the conventional neural networks was the same signal sampled at 1 sec and 24 s intervals respectively. All training data records contained the same number of training vectors to enable a fair comparison of the networks. The real time length of the training data for the conventional neural network with a 24 s sample time was therefore 24 times that for the other two networks.

Figure 16 illustrates the improvements in multi-step ahead performance which can be achieved using the spread encoding technique. The loss functions on a test data set for the neural network models trained with a 1 s sample time are shown for different prediction horizons. Both networks achieve a similar prediction accuracy for one step ahead predictions. However, as the prediction horizon is increased, the neural network using conventional scaled data exhibits continual deterioration in its prediction accuracy. In contrast, the prediction accuracy of the spread encoded network shows a significantly lower increase in the loss function for increasing prediction horizon and saturates for large prediction horizons.

The predictions for the same test signal of the conventional network, trained with a sample time of 24 secs., and the spread encoded network are compared in Figs 17 and 18. The conventionally trained neural network is predicting one step ahead, whereas the spread encoded neural network is predicting 24 steps ahead (and is therefore operating recursively 23 times for each

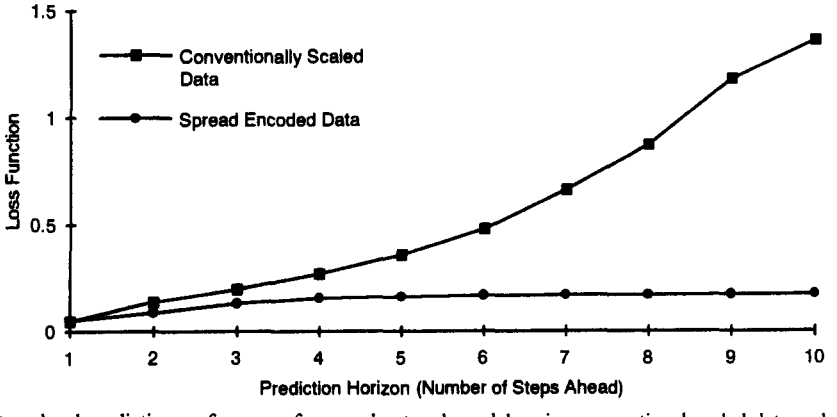


Fig. 16. Multi-step ahead prediction performance for neural network models using conventional scaled data and spread encoded data.

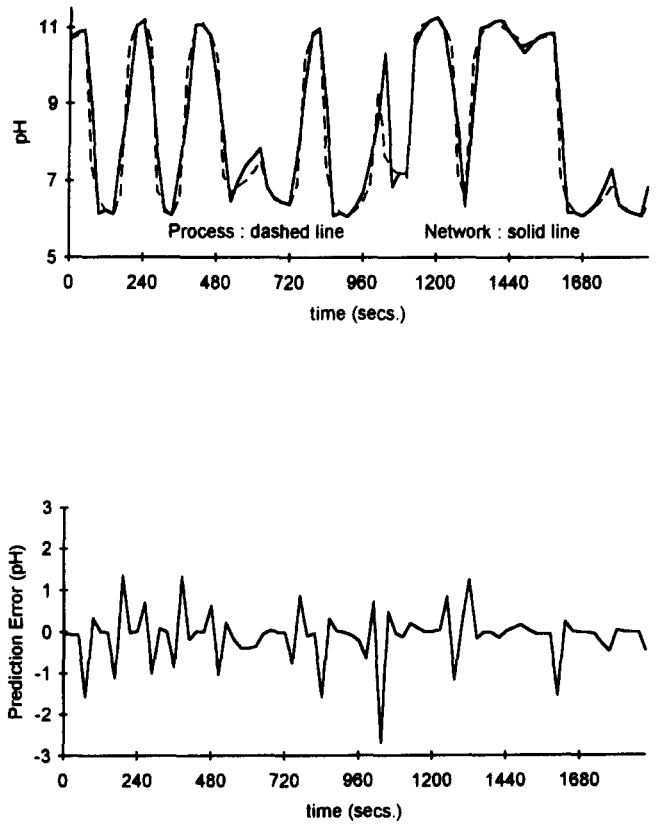


Fig. 17. Predictions of a 2nd order neural network model using conventional scaled data and a 24 sec data sample time, predicting one step ahead. (a) process and neural network output, (b) prediction errors.

the spread encoded neural network so that the two model predictions are compared at the same instants in time. The *LF* for the spread encoded neural network (*LF*=0.062, maximum prediction error =1.6 pH units) was 6 times smaller than the one step ahead neural network (*LF* =0.386, maximum prediction error=2.7 pH units). The improved performance of the spread encoded neural network further illustrates the effects of sample time on a neural network model. The use of a faster sample time for this model enables the network to provide significantly better predictions at 24 s intervals than the one step ahead network trained with data sampled at this rate, despite its recursive mode of operation.

5. CONCLUSIONS

Experiment design considerations for neural network identification of non-linear processes have been presented. It is stressed that although neural networks are commonly regarded as black-box models, this does not preclude the engineer from exerting considerable influence over the outcome of an identification experiment. The choices for data sample time, excitation signal and model structure all influence the performance of an identified neural network model. These design issues were discussed and their effects illustrated through simulation studies. The use of a range of techniques to assist in the design of a neural network identification experiment and for validation of a neural network model were demonstrated.

Acknowledgements—The authors would like to thank Liverpool John Moores University and British Nuclear Fuels plc (Sellafield, U.K.) for financial support of S.K. Doherty and this research project.

REFERENCES

Akaike H., A new look at the statistical model identification. *IEEE Trans.* **AC-19**, 716–722 (1974).

Bhat N. and T.J. McAvoy, Use of neural nets for dynamic modelling and control of chemical process systems. *Computers Chem. Engng.* **14**(4/5), 573–583 (1990).

Bhat N., P.A. Minderman, T.J. McAvoy and N.S. Wang, Modelling chemical process systems via neural computation. *IEEE Control Sys. Magazine* **April**, 24–30 (1990).

Billings S.A. and W.S.F. Voon, Correlation based model validity tests for non-linear models. *Int. J. Control* **44**, 235–244 (1986).

Billings S.A., H.B. Jamaluddin and S. Chen, Properties of neural networks with applications to modelling non-linear dynamical systems. *Int. J. Control* **55**, 193–224 (1992).

Chen S., S.A. Billings and P.M. Grant, Non-linear system identification using neural networks. *Int. J. Control* **51**, 1191–1214 (1990a).

Chen S., S.A. Billings, C.F.N. Cowan and P.M. Grant, Practical identification of NARMAX models using radial basis functions. *Int. J. Control* **52**, 1327–1350 (1990b).

Chen F.C. and H.K. Khalil, Adaptive control of nonlinear

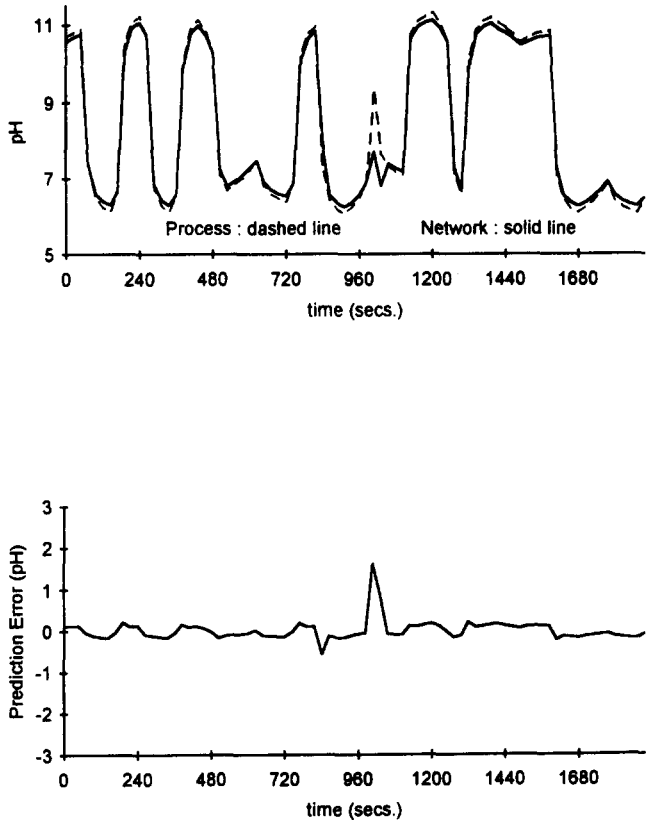


Fig. 18. Predictions of a 2nd order neural network model using spread encoded data and a 1 sec data sample time, predicting 24 steps ahead. (a) process and neural network output, (b) prediction errors.

systems using neural networks. *Int. J. Control* **55**, 1299–1317 (1992).

G. Cybenko, Continuous value neural networks with two hidden layers are sufficient. *Math. Contr. Signal and Sys.* **2**, 303–314 (1989).

Evans J.T., Gomm J.B., Williams D., Lisboa P.J.G. and To Q.S., A practical application of neural modelling and predictive control. In *Application of Neural Networks to Modelling and Control*, (Eds, Page, G.F., Gomm, J.B. and Williams, D.), 74-88. Chapman and Hall, U.K. 1993.

Gomm J.B., P.J.G. Lisboa, D. Williams and J.T. Evans, Accurate multi-step-ahead predictions of non-linear systems with the MLP neural network using spread encoding. *Trans. Ints. Meas. Control* **16**, 203–213 (1994).

Hernández E. and Y. Arkun, Study of the control relevant properties of backpropagation neural network models of nonlinear dynamical systems. *Computers Chem. Engng.* **16**, 227–240 (1992).

Hopfield, J. J., Neural computation of decisions in optimization problems. *Biology. Cybern.* **52**, 141–152 (1985).

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

Hunt K. J., D. Sbarbaro, R. Zbikowski and P. J. Gawthrop, Neural networks for control systems—a survey. *Automatica* **28**(6), 1083–1112 (1992).

Isermann R., Practical aspects of process identification. *Automatica* **16**, 575–587 (1980).

Leontaritis I. J. and S. A. Billings, Input–output parametric models for non-linear systems: 1. deterministic non-linear systems: 2. stochastic non-linear systems. *Int. J. Control* **41**, 303–344 (1985).

McAvoy T. J., E. Hsu and S. Lowenthal, Dynamics of pH in controlled stirred tank reactors. *Ind. Eng. Chem. Process Des. Dev.*, **11**, 68–70 (1972).

Pollard J. F., M. R. Broussard, D. B. Garrison and K. Y. San, Process identification using neural networks. *Computers Chem. Engng.* **16**, 253–270 (1992).

Pottmann M. and D. E. Seborg, Identification of non-linear processes using reciprocal multiquadric functions. *J. Proc. Control* **2**(4), 189–203 (1992).

Rumelhart D. E., G. E. Hinton and R. J. Williams, Learning internal representations by error propagation. in *Parallel Distributed Processing* (Rumelhart D. E. and McClelland J. L., Eds.), MIT Press, Cambridge, M.A. (1986).

Saint Donat J., N. Bhat and T. J. McAvoy, Neural net based model predictive control. *Int. J. Control* **54**(6), 1453–1468 (1991).

Widrow, B., R. G. Winter and R. A. Baxter, Layered neural nets for pattern recognition. *IEEE Trans. Acoust. Speech, Signal Processing* **36**(7), 1109–1118 (1988).

6. APPENDIX

The simulated pH CSTR process equations (McAvoy *et al.*, 1972) and parameter values are given below.

$$pH\Delta - LOG_{10}[H^+]$$

$$\alpha=[CH_3COOH]+[AC^-]$$

$$\beta=[Na^+]$$

Acetate balance...

$$F_A C_A - (F_A + F_B\alpha) = V\frac{d\alpha}{dt}$$

Sodium balance...

$$F_B C_B - (F_A + F_B)\beta = V\frac{d\beta}{dt}$$

Acetic acid equilibrium...

$$[AC^-][H^+] = K_A[CH_3COOH]$$

Water equilibrium...

$$[H^+][OH^-] = K_W$$

Electroneutrality...

$$\beta + [H^+] = [OH^-] + [AC^-]$$

[X], concentration of X
K_A, dissociation constant of CH_3COOH (1.8×10^{-5})
K_B, dissociation constant of NaOH (1.0)
K_W, dissociation constant of water (1.0×10^{-14})
V, CSTR volume (3000 l)
F_A, flow rate of CH_3COOH (4.05 l/s)
F_B, flow rate of NaOH (5.75 l/s at steady state)
C_A, concentration of CH_3COOH (0.3178 M)
C_B, concentration of NaOH (0.05 M)
pH, effluent pH (9.18 at steady state)