ELSEVIER

Fuzzy Sets and Systems 79 (1996) 113–126

FUZZY sets and systems

Enhancing the non-linear modelling capabilities of MLP neural networks using spread encoding

J.B. Gomm[a,*], D. Williams[a], J.T. Evans[a], S.K. Doherty[a], P.J.G. Lisboa[b]

[a] *Control Systems Research Group, School of Electrical and Electronic Engineering, Liverpool John Moores University, Byrom Street, Liverpool L3 3AF, UK*

[b] *Department of Electrical Engineering and Electronics, University of Liverpool, P.O. Box 147, Liverpool, L69 3BX, UK*

Abstract

Two methods for representing data in a multi-layer perceptron (MLP) neural network are described and the resultant ability of networks, trained by the standard back-propagation algorithm, to identify the dynamics of non-linear systems is investigated. One of the data conditioning methods has been widely used in studies of the MLP network and consists of normalising each network input and output variable and applying the normalised data to single network nodes. In the second method, named spread encoding, each network variable is represented as a sliding Gaussian pattern of excitations across several network nodes. The spread encoding technique exhibits similarities with conventional algorithms used in fuzzy logic and a network utilising this method can be considered as a fuzzy-neural type network. Neural networks are configured to represent a non-linear, auto-regressive, exogenous (NARX) input–output model structure and the performance of trained networks is investigated in applications to modelling a real liquid level process unit and a simulation of a highly non-linear chemical process. Results show that using the data normalisation method, a network can provide accurate single-step predictions but is incapable of adequate long-range predictions. In contrast to this, the spread encoding technique significantly enhances the performance of a MLP network model enabling accurate single-step and long-range predictions to be achieved.

Keywords: Fuzzy-neural networks; Non-linear system identification; Non-linear process modelling; Production and process control

1. Introduction

Some artificial neural network architectures exhibit the capability of forming complex mappings between input and output which enable the network to approximate general non-linear mathematical functions. This non-linear mapping feature has prompted the investigation of these neural network architectures for their suitability for identifying the dynamics of non-linear systems. The particular network which is the focus of this paper is the multi-layer perceptron (MLP) neural network, trained by the standard back-propagation algorithm, because it is the most widely used network and its mathematical properties for non-linear function approximation are well documented [4, 9].

A common approach taken to enable a neural network to capture the dynamics of a non-linear

* Corresponding author.

SSDI 0165-0114(95)00294-4

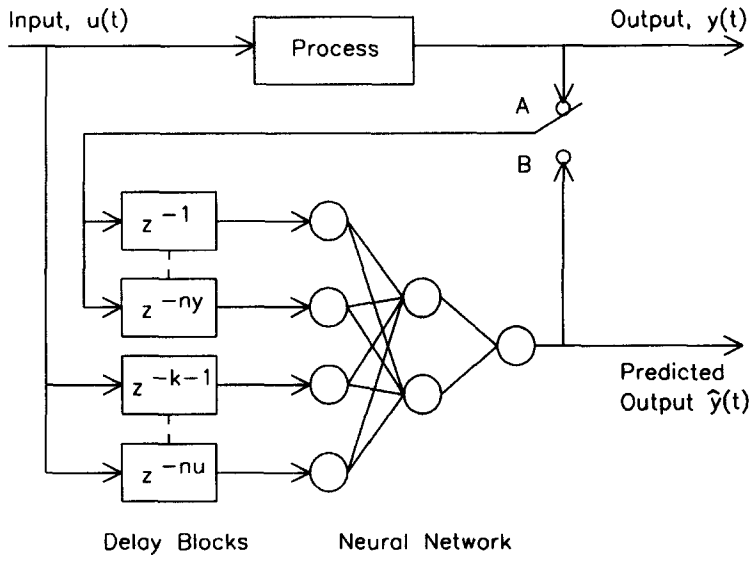

Fig. 1. Two operating modes of a neural network configured in a NARX input–output model structure. Switch in position: (A) one-step-ahead predictor operation; (B) model operation.

system is to configure and train a network to represent a non-linear, auto-regressive, exogenous (NARX) input–output model structure [1, 2]. Here, the network input consists of a moving window of time-delayed system inputs and outputs and the network is trained to predict the system output at the next sampling instant (termed one-step-ahead prediction, Fig. 1). Once trained, single-step predictions of the system output can be obtained from the network by simply supplying it with lagged system data in an identical fashion to that used during network training. However, at any time instant, to obtain predictions of the system output multiple time steps into the future it is necessary to assume that future system inputs are known, and future system outputs are substituted by the network predictions and fed back into the network (termed model operation, Fig. 1).

It is often the case that the resulting neural network model is to be subsequently utilised in a control strategy. In these situations, the overall performance of the control scheme is ultimately dependent on the accuracy of the neural network representation of the controlled dynamic system. For some control structures, such as internal model control [11], a standard feed-forward network operating as a one-step-ahead predictor of the system is adequate. However, in other control schemes, such as model predictive control [8, 22], one-step-ahead prediction is not sufficient and accurate long-range network predictions in the recursive model operating configuration are required. Long-range predictions are also necessary if the identified neural network model is to be used as a system simulation tool because, in this application, the benefits of one-step-ahead prediction are extremely limited.

Employing the neural network training procedure described above to obtain a satisfactory representation of a non-linear dynamical system, therefore requires consideration to be given to the ability of a resulting network to provide accurate long-range predictions. Since the neural network is trained to provide a prediction of the system output a single time step ahead, accurate predictions in this operating mode do not necessarily ensure that the same network is capable of producing accurate long-range predictions in the model operating mode. Furthermore, the presence of feedback in the model operation can result in an accumulation of network errors which significantly deteriorate the long-range prediction accuracy.

This paper describes investigations of utilising a conventional MLP neural network for the identification of NARX models of non-linear dynamical systems with an emphasis on achieving a neural network model capable of accurate long-range predictions. Two methods for representing data in the MLP network are described and the performance of network models utilising these methods is examined. One novel technique of coding network data, investigated in this paper, is based on spreading each network input and output variable across several network nodes using a sliding Gaussian pattern of node excitations. This spread encoding (SE) approach has similarities with data fuzzification techniques where the scalar dimensional space of each variable is fuzzified to a space of higher dimensions. Also, decoding of the network output using the SE method consists of computing a weighted summation of the node excitations which is analogous to the conventional centre of gravity defuzzification technique. Thus, a network utilising spread encoding can be considered as a fuzzy-neural-type network. The SE method is also in keeping with the heritage of neural networks from biological systems where information is often represented by the combined activity of a population of receptors, as in the retina of the eye [16].

Other forms of spread encoding neural network data have been reported. Lin and Lee [13] describe a fuzzy-neural network controller where the data coding is achieved using conventional Gaussian fuzzy membership functions. A triangular form of spread encoding is described by Gent and Sheppard [7] in an application of neural networks to non-linear time-series signal modelling. The spread encoding method used in this paper is formulated differently to these approaches and its application to input–output modelling of non-linear dynamical systems is novel.

The performance of the MLP network with spread encoding is compared to the use of a normalisation method of representing network data. The data normalisation approach has been widely adopted in applications of the MLP network and involves normalising each network input and output variable to a predefined range (usually between zero and one) and applying the normalised data to single network nodes.

The methods used in this research for configuring, training and operating an MLP neural network to obtain both single-step and long-range predictions of non-linear dynamical systems are outlined in Section 2 of this paper. Section 3 describes the data representation methods investigated and applications of the techniques to modelling a laboratory liquid-level process unit and a simulation of a highly non-linear pH process are described in Sections 4 and 5, respectively.

2. Configuring the MLP network for non-linear system identification

The general architecture of the MLP neural network used throughout this work consisted of an input layer, a single hidden layer and an output layer with the sigmoidal activation function used for all of the nodes in the hidden and output layers. The output of the jth node in the lth network layer ($l = 1, 2$) was therefore given by

$$x_j^l = \frac{1}{1 + e^{-(w_j^l x^{l-1} + w_{0j}^l)}}, \tag{1}$$

where w_j^l is a row vector of node weights, x^{l-1} is a column vector of node inputs (i.e. the outputs from the previous $(l-1)$th layer) and w_{0j}^l is a node bias weight. In the case considered with only a single hidden layer, x^0 is the network input vector, x^1 is the hidden node output vector and x^2 is the network output vector. Only one hidden layer of non-linearity was used as it has been proven and widely accepted that this is sufficient for the network to approximate any non-linear, input–output mapping providing an adequate number of hidden nodes are present [4, 9].

To enable a neural network to capture the input–output dynamics of non-linear systems, the networks were configured in the widely applicable NARX model structure (Fig. 1):

$$y(t) = f(y(t-1), \ldots, y(t-n_y), u(t-k-1), \ldots, u(t-k-n_u)) + e(t), \tag{2}$$

where $y(t)$ and $u(t)$ are the sampled system output and input at time t, respectively, k represents the

system delay, n_u and n_y determine the number of lagged system inputs and outputs used in the model structure, $f(\ )$ is the non-linear function to be identified and $e(t)$ is a zero mean white noise sequence representing random system disturbances. Determination of a suitable model structure, i.e. values for n_u, n_y and k, were simplified by setting $n = n_u = n_y$ where n is termed the model order, which is common with linear system identification approaches [12]. To configure the MLP to represent a NARX model, the network inputs are assigned to the delayed system input and output values, $x^0(t) = [y(t-1), \ldots, y(t-n_y), u(t-k-1), \ldots, u(t-k-n_u)]^T$, and the network is trained to provide a one-step-ahead prediction of the system output, $x^2(t) = \hat{y}(t)$, by minimising the following cost function over the training data set:

$$J = \tfrac{1}{2}\sum[y(t) - \hat{y}(t)]^2. \tag{3}$$

Eq. (3) was minimised during training by updating the network weights after the presentation of each input–output data pair using the standard gradient descent back-propagation learning algorithm with momentum term [19]. Hence, the weight update at iteration t was given by

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

where $w_{ij}^l(t)$ is the weight from the ith neuron in the $(l-1)$th layer to the jth neuron in the lth layer, $x_i^{l-1}(t)$ is the output of the ith neuron in the $(l-1)$th layer, $\Delta w_{ij}^l(t)$ is the previous weight change, η and α are the learning rate and momentum, respectively, and are chosen in the range $[0, 1]$.

Prior to network training, all network weights were initialised to small random values determined from a uniformly distributed set of random numbers in the range $[-0.1, 0.1]$. From experience, the learning rate and momentum parameters in the back-propagation algorithm were initially set to $\eta = 0.9$ and $\alpha = 0.6$ and were gradually reduced as training advanced to assist the network convergence. The network training data were also presented in a random order as this has been found to break up serial correlations in the training data leading to significant improvements in the resulting neural network model performance and convergence speed.

Once trained, long-range predictions are obtained by supplying the network with system input data and feeding the delayed network output back into the network, thus operating the network in the model configuration (Fig. 1):

$$\hat{y}(t) = f(\hat{y}(t-1), \ldots, \hat{y}(t-n_y), u(t-k-1), \ldots, u(t-k-n_u)). \tag{5}$$

Training the network to provide a one-step-ahead prediction of the system output enables stable training to be achieved with the back-propagation algorithm. It is possible to train the network in the model configuration; however, this requires the use of other, generally more complex, training algorithms [18, 21]. The error surface is also further complicated when training in the model configuration by the existence of more local minima; hence, convergence to the global minimum of the cost function is less certain than training the network as a one-step-ahead predictor.

3. Data representation methods

The two methods of representing data in the MLP network that are investigated in this paper are described in this section. The first method has been widely adopted in many studies of neural networks applied to control [17, 22], fault diagnosis [10] and non-linear system identification [1, 2, 20]. The method consists of normalising the network data to a prespecified range, usually between zero and one, and applying the normalised data to single nodes at the input and output of the network.

When sigmoidal activation functions are used in the network output nodes, using target network outputs of zero and one for training can result in very large network weights because a large weighted sum is required for the output of the sigmoidal function to reach these limits. Also, as the network outputs approach zero and one during

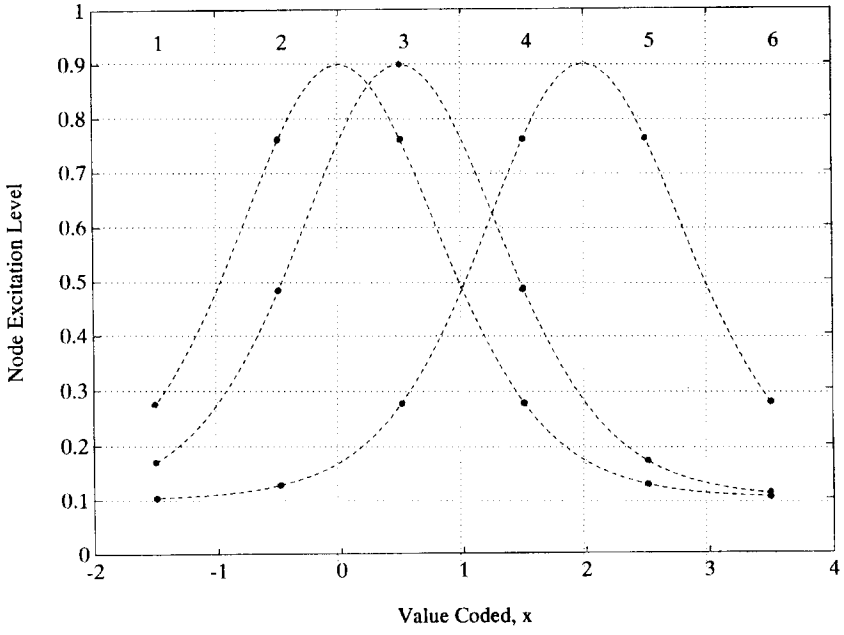

Fig. 2. Node excitations by spread encoding a variable $x = 0, 0.5, 2$ to six network nodes.

training the weights can become frozen because the error gradient, $\partial J/\partial w_{ij}^{l}(t)$ in Eq. (4), tends to zero [10]. Hence, the range of the normalised data is chosen to be slightly less than zero and one and a range of [0.1,0.9] was used throughout these studies. A network using this data representation method will be subsequently referred to as a normalised data (ND) network throughout the remainder of this paper.

The second data conditioning method of spread encoding consists of mapping each network variable, $x \in [x_{\min}, x_{\max}]$, onto a sliding Gaussian activation pattern of N network nodes, which includes additional nodes either side of the variable range to contain overspill resulting from the use of a mapping function with wide support. The level of activation of each node is confined to be in the range [0.1,0.9] for the same reasons as described above for the conventional normalisation technique. Each node is assigned a value, a_i, linearly spaced by a distance, δ, to span the range of x, and the centre of the Gaussian excitation pattern corresponds to the value coded, as shown in Fig. 2.

The spread encoding algorithm is derived by creating a discrete map which represents the mean value of a continuous probability distribution, $\phi(a)$, within each class interval. This then provides a simple mechanism for retrieving the original coded value as a sum of the activity of the node excitations, each weighted by the values at the centres of the class intervals, a_i. For a particular value of x, the excitation of each node is defined by

$$\psi_i(x) = \frac{\int_{a_i-\delta/2}^{a_i+\delta/2} a\phi(a-x)\,\mathrm{d}a}{a_i}, \tag{6}$$

which satisfies the requirement that

$$\sum_{i=1}^{N} a_i\psi_i(x) = \int a\phi(a-x)\,\mathrm{d}a = \bar{a} = x. \tag{7}$$

It is assumed that the distribution $\phi(a)$ has unit area.

The activation of a particular node can be evaluated from (6) by integration by parts:

$$a_i\psi_i(x) = [a\Phi(a-x)]_{a_i-\delta/2}^{a_i+\delta/2} - \int_{a_i-\delta/2}^{a_i+\delta/2} \Phi(a-x)\,\mathrm{d}a, \tag{8}$$

where $\Phi(a)$ is a parent cumulative distribution with $\phi(a) = \Phi'(a)$. In the investigations reported in this paper, the integral term in Eq. (8) was approximated using the first two terms in the trapezium rule resulting in

$$\psi_i(x) \approx \Phi(a_i + \delta/2 - x) - \Phi(a_i - \delta/2 - x) \tag{9}$$

which was found to provide sufficient accuracy in these studies. Further levels of accuracy in the coding technique can be achieved by approximating the integral term using the trapezium rule with more terms per interval.

The relationship between this coding technique and conventional fuzzification techniques is illustrated by considering a first approximation of the integral term in Eq. (8) resulting from a Taylor series expansion of the cumulative function about the interval centre, a_i, and keeping the linear term in the expansion. This leads to

$$\psi_i(x) \approx \phi(a_i - x) \tag{10}$$

which is analogous to the use of membership functions in fuzzy logic [13,23].

The practical advantage of spread encoding put forward in this paper, is that it leads to more accurate models using static feed-forward neural networks than representing normalised physical variables using single nodes. One reason for this is that signal noise is reduced in the spread encoded representations by suitable matching of the coding function with the interval width spanned by each node. It can be shown that, to leading order in δ,

$$\frac{\partial\psi_i(x)}{\partial x} = \delta\phi'(x - a_i) + 0(\delta^2). \tag{11}$$

The noise reduction depends on keeping the slope of this representation smaller than unity.

The spread encoding algorithm was implemented by initially scaling the data to a normalised range where the original data range $r \in [r_{\min}, r_{\max}]$ was represented by $x \in [0, N - 2N_0]$ with N the total number of nodes, N_0 the number of nodes on either side of the variable range and $\delta = 1$. The procedure was also simplified by approximating the cumulative Gaussian distribution function by the sigmoidal function, Eq. (1). The full algorithm used to code and decode a value, r, is given below.

Coding

Step I: Scale r to the normalised range by

$$x = \frac{N - 2N_0}{r_{\max} - r_{\min}}(r - r_{\min}). \tag{12}$$

Step II: Code the data to the N network nodes by

$$\psi_i(x) = \Phi(a_i + 1/2 - x) - \Phi(a_i - 1/2 - x), \quad i = 1, \ldots, N, \tag{13}$$

where

$$a_i = i - N_0 - c \tag{14}$$

and $\Phi(a)$ is the sigmoidal function centred at x:

$$\Phi(a - x) = \frac{1}{1 + \mathrm{e}^{-\beta(a-x)}}. \tag{15}$$

In Eq. (14), c is an offset term which shifts the position of the range limits on the nodes. The width of the node excitations is inversely controlled by the parameter β in Eq. (15).

Step III. Scale the excitation of each node to the range [0.1, 0.9] using a fixed linear relationship.

Decoding

Step I: Apply the inverse of the scaling relationship used in Step III of the coding procedure to descale the node excitations from the range [0.1, 0.9] back to their original range.

Step II: Errors arise in decoding using a straightforward application of Eq. (7) because the node excitations, $\psi_i(x)$, are calculated by an approximation, Eq. (9). The accuracy of the decoding is improved by dividing the weighted sum by the sum of the node excitations. Thus, the network output is decoded back to the normalised range using

$$x = \frac{\sum_{i=1}^{N} a_i\psi_i(x)}{\sum_{i=1}^{N} \psi_i(x)} \tag{16}$$

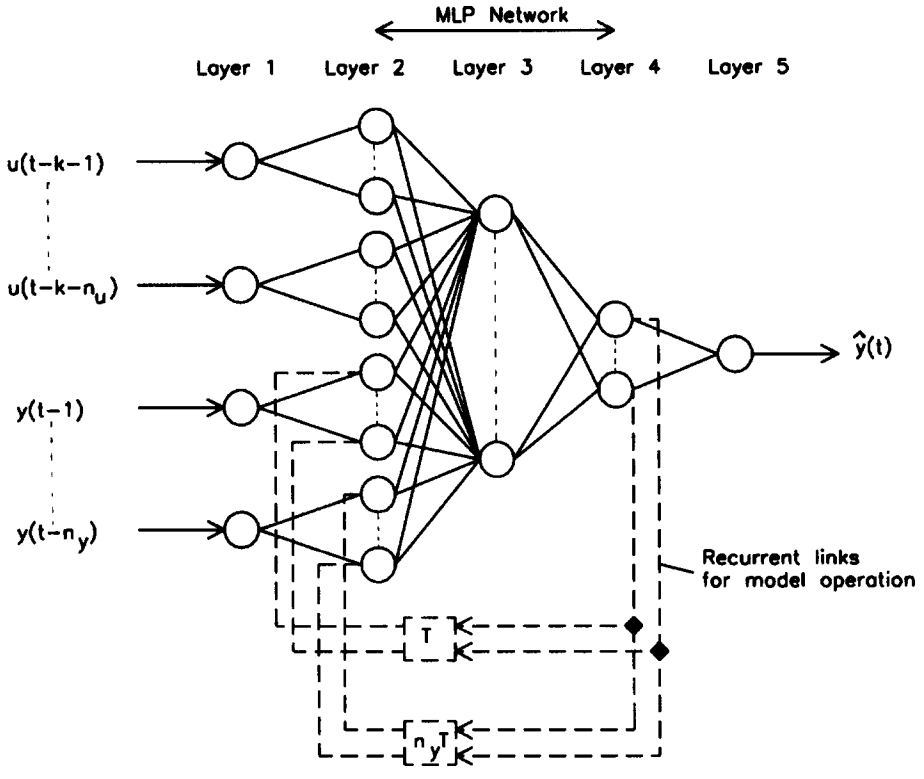

Fig. 3. Representation of an MLP network using spread encoding as a 5-layer fuzzy-neural network.

which is analogous to the conventional centre of gravity defuzzification technique.

Step III: Apply the inverse of the scaling relationship used in Step I of the coding procedure to determine the final decoded value, r.

In this work, the parameters used in the spread encoding algorithm were $N = 6$, $N_0 = 2$, $c = 0.5$ and $\beta = 2$ which were found to provide sufficiently accurate coding and decoding in the applications reported in this paper. Typical node excitations obtained with these parameters are illustrated in Fig. 2 for the coding of a variable $r \in [0, 2]$ which, for the parameter values used, corresponds to the special case $x = r$, Eq. (12).

When the derivative of the sigmoidal function, Eq. (15), is used as the coding function, the slope of the response of node i, Eq. (11), becomes

$$\frac{\partial \psi_i(x)}{\partial x} = \delta\beta^2 \Phi(x - a_i)[1 - \Phi(x - a_i)] \times [1 - 2\Phi(x - a_i)] + 0(\delta^2) \tag{17}$$

which has its maximum value for $\Phi(x - a_i) = 1/2 - \sqrt{3}/6$. The parameters used in this study, therefore, lead to a maximum rate of response of a network node to the coded variable of $\partial\psi_i(x)/\partial x|_{\max} = 0.385 < 1$. The response slope is also generally much smaller than its peak value; hence, this provides some explanation of the reductions is noise and error accumulation observed in practice when using the spread encoding technique.

A MLP neural network with spread encoding can be considered as a 5-layer fuzzy-neural type network as shown in Fig. 3. Layer 1 receives the network data and fuzzifies each input to N nodes in

layer 2 using the spread encoding algorithm. Layers 2–4 comprise the conventional 3-layer MLP network and layer 5 decodes the fuzzified network output of layer 4 back into the original output variable range. During training the MLP network is supplied with the spread encoded input data and is trained to provide an output corresponding to the spread encoded target output for each input pattern presented. After training, recurrency during model operation of the network takes place between layers 4 and 2 as shown.

4. Modelling a liquid level process unit

The liquid level process unit used in these investigations is shown in Fig. 4 and consists of two noninteracting tanks each of 1.2 m in height and 20 l capacity. The process input is the signal to the pneumatic control valve, which controls the input flow rate to the top tank, and the output is the height of liquid in the bottom tank. Standard industrial instrumentation is utilised with the pneumatic control valve driven from a current-to-pressure converter and the level measurement derived from a differential pressure cell; the pressure signal is subsequently converted to a voltage via a pressure-to-current converter. The pressure and current ranges used in the instrumentation are the standard 3–15 psi and 4–20 mA, respectively. The rig is interfaced to a personal computer for data acquisition and digital control. This liquid level laboratory process exhibits some features that are representative of those in industrial processes, such as nonlinearities (including hysteresis in the control valve), time variations and noise, and therefore provides

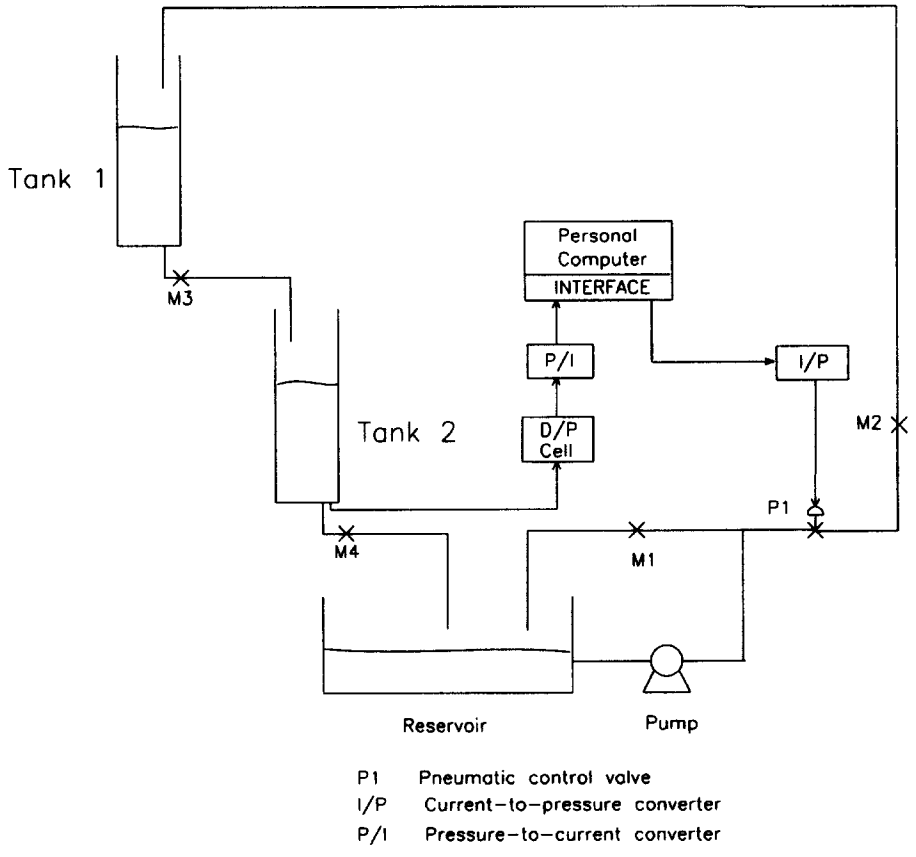

Fig. 4. Liquid level process unit.

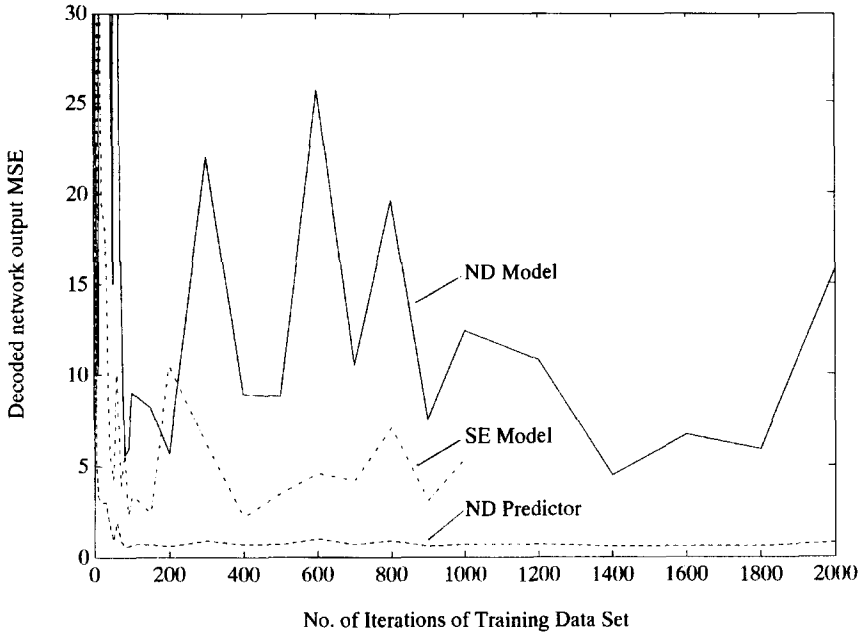

Fig. 5. Training MSEs on liquid level process test data for ND and SE networks.

a practical intermediary test bed for the investigation and development of new techniques for process modelling, monitoring and control.

Input–output data from the process for training the neural network models was collected by applying a random amplitude signal (RAS), consisting of a sequence of uniformly distributed random numbers, to the process input. The input range of the RAS was chosen to ensure that the process was excited well over its full operating range, thus enabling a neural network model trained with the data to adequately capture the non-linear process dynamics. Two data sets, each comprising 300 input–output data samples, were collected using different RASs. One data set was used for training and the other was used to test the generalisation capabilities of a network as the training progressed.

Knowledge obtained from step response tests and gained from the physical structure of the process indicated that the liquid level process exhibited predominantly second-order dynamics and no significant delay, other than one sampling period, between input and output. This knowledge, therefore, provided an initial NARX model structure for the neural network of $n_u = n_y = 2$ and $k = 0$ in Eq. (2). The number of nodes in the hidden layer of each network was determined empirically based on the ability of the networks to learn the training data without the training time becoming excessive. This resulted in MLP network topologies of 4-6-1 (4 inputs, 6 hidden nodes and 1 output node) for the ND network and, with each network variable spread over the activity of 6 nodes, 24-6-6 for the SE network.

Fig. 5 shows the performance of the networks on the test data set during training. The mean square errors (MSE) shown on the graph were computed after decoding the network outputs and rescaling back to the original process output range. Convergence of the one-step-ahead predictor MSEs for both networks was similar and is illustrated in Fig. 5 for the ND network. Both networks converged to approximately the same low MSE with the SE network achieving a slightly smaller MSE than the ND network. In contrast to this, the performance of the networks as process models, where the networks are operated recurrently for the entire test data set and are required to predict the process

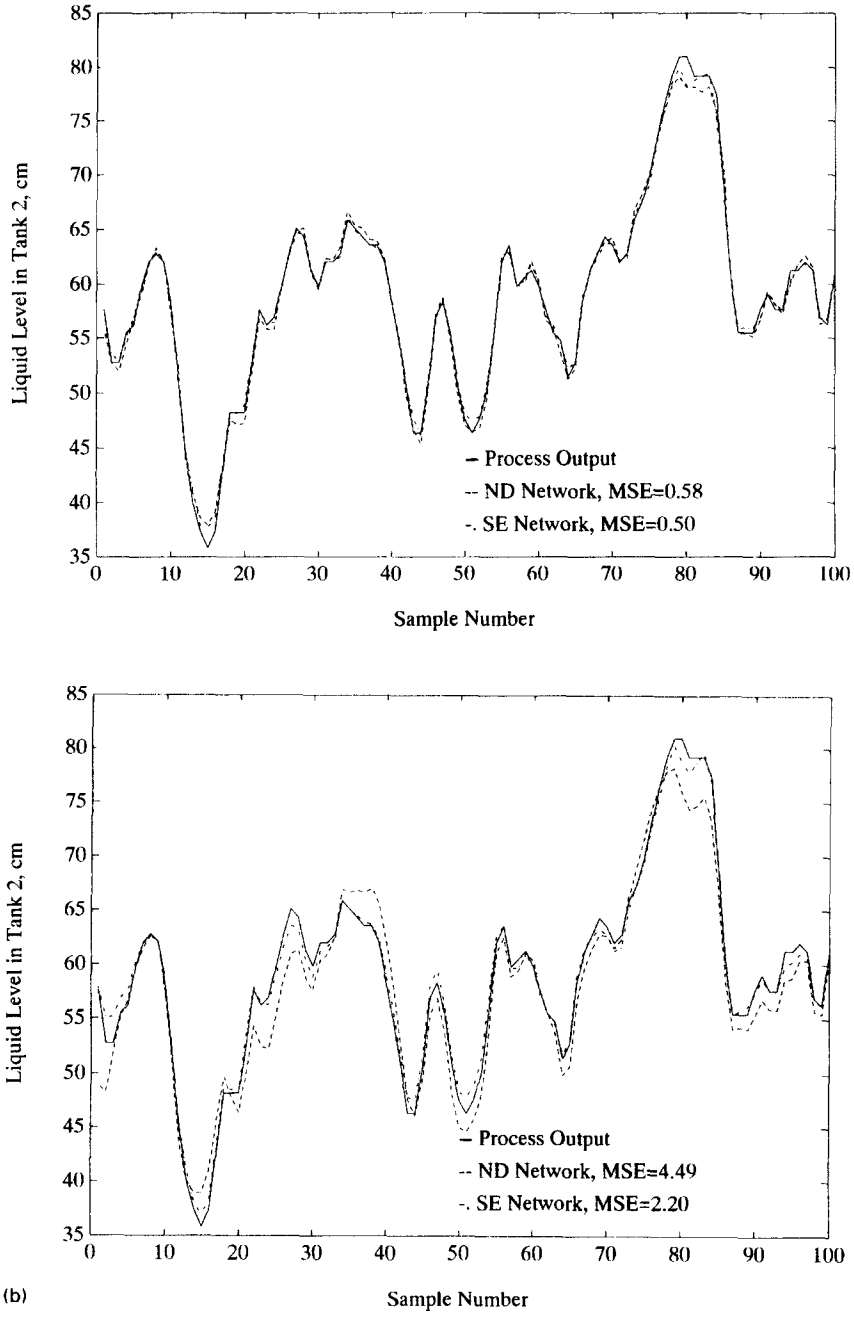

Fig. 6. Performance of ND and SE networks on liquid level process test data: (a) one-step-ahead predictions; (b) model predictions.

output from only the input data, differ significantly. The performance of the SE network as a model is less erratic than the ND network and reaches a much lower MSE in considerably less training iterations. Training of the ND network was continued for 1000 iterations more than the SE network

to reach a lower model MSE for a fairer comparison of the achievable prediction accuracy of the two networks. The training curves in Fig. 5 also illustrate that a low one-step-ahead predictor MSE is not a reliable indicator for accurate model operation. Training for both networks was terminated at the points where the lowest model MSEs were observed, which occurred at 400 iterations for the SE network and 1400 iterations for the ND network.

The results deduced from the network training curves are further illustrated in Fig. 6 which shows the network output predictions when tested on the RAS test data (only 100 of the 300 data points are shown for clarity). These graphs confirm that both networks provide accurate one-step-ahead predictions (Fig. 6(a)); however, only the SE network performs adequately as a process model (Fig. 6(b)). The accurate predictions of the SE network on the test data in both operating modes confirmed the original choice of the neural network NARX model structure as suitable for representing the non-linear dynamics of the liquid level process. The SE network model of the liquid level process has been successfully used in an on-line predictive control scheme to provide long-range predictions which are used to compute optimally the process input for improved non-linear process control [8].

5. Modelling a highly non-linear pH process

The application of the MLP network and data coding techniques to modelling a simulation of a highly non-linear chemical process is described in this section. The process is a continuous stirred tank reactor (CSTR) used for pH control and is shown in Fig. 7. Acetic acid (CH_3COOH) of concentration C_A flows into the tank at a rate F_A and is neutralised by sodium hydroxide (NaOH) of concentration C_B and flow rate F_B. The volume of liquid in the tank is assumed to be constant and perfectly mixed. The mathematical model used for simulating the pH dynamics in the CSTR is based on theoretical material balances, equilibrium and electroneutrality relationships for the system [15].

This chemical process, consisting of a mixture of strong acid and weak base, is highly non-linear and

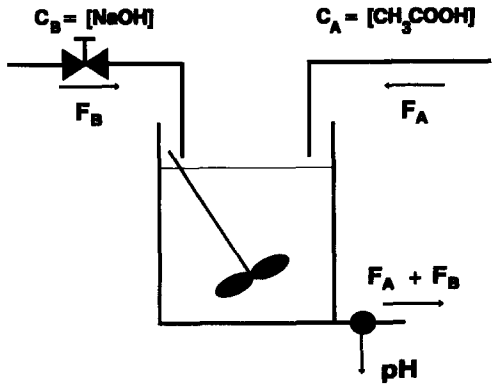

Fig. 7. pH process.

is a severe test for the modelling capabilities of any neural network architecture. The non-linearity is illustrated in Fig. 8 which shows the steady-state process titration curve where a gain change of more than 150 occurs between points A and B on the curve. The steady-state operating point of the process was set in the high gain process region corresponding to point A on the titration curve (Fig. 8).

Data for network training, consisting of 1000 input–output data pairs (F_B and pH), were obtained by maintaining F_A constant and superimposing an excitation signal, with a maximum amplitude of 10%, on a steady-state F_B. Excitation of the process by a standard RAS resulted in poor one-step-ahead predictions of pH from a neural network trained with this data. An examination of the output pH data density by histogram analysis revealed that there were few data in the high gain region of the titration curve (Fig. 8), thus causing poor predictions in this area [5]. An alternative signal was therefore used to excite the process and improve the quality of the training data. This signal was realised by forcing the RAS through the steady-state input flow rate on each clock pulse which provided more pH data in the high gain region and effected improved network predictions.

The selection of a suitable input–output NARX model structure for the neural network was based on the ability of a range of networks with different model orders to emulate the process under a variety of test conditions [6]. The performance of each network was assessed by the use of a performance

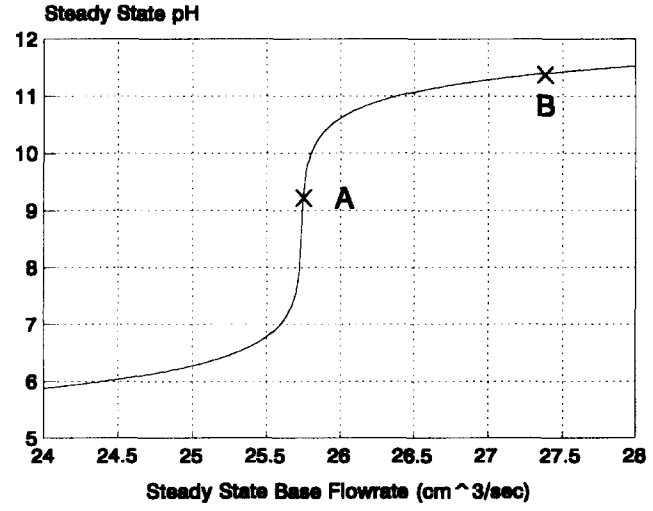

Fig. 8. pH process titration curve.

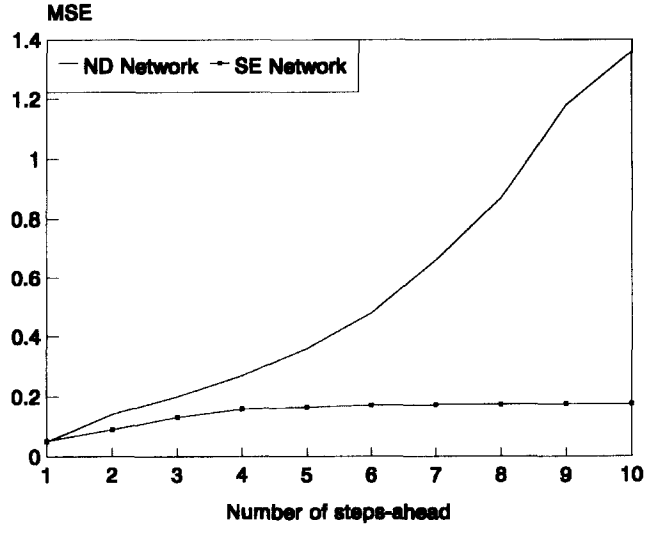

Fig. 9. Effect of prediction horizon on the performance of the ND and SE networks for the pH process.

index (computed as a measure of the overall accuracy of a network for all of the test conditions), statistical metrics (such as Akaike's information criterion [14]) and correlation tests [3]. These studies indicated the most appropriate neural network model structure for the pH process to be a second-order NARX model with no significant delay; hence, $n_u = n_y = 2$ and $k = 0$. The final network topologies used were 4-15-1 for the ND network and 24-15-6 for the SE network. In this investigation both networks were trained for 1000 iterations of the training data set.

The prediction accuracy of both trained networks when tested on a different RAS is illustrated in Fig. 9. The one-step-ahead prediction MSEs of both networks are similar and both networks were

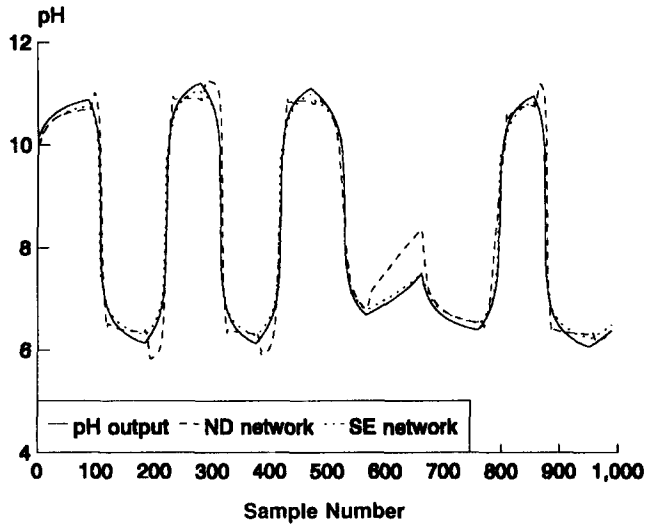

Fig. 10. Ten-step-ahead predictions of ND and SE networks on pH process test data.

capable of predicting the output pH with sufficient accuracy. However, as the prediction horizon increases there is a marked improvement in the long-range prediction accuracy of the SE network compared to the ND network. The MSE of the SE network marginally increases as the prediction horizon rises and, for predictions more than five steps ahead, the MSE settles at a level slightly higher than the MSE for single-step prediction. In contrast, the MSE for the ND network continues to rise sharply as the prediction horizon is increased, demonstrating the inability of this network to provide long-range predictions with any reasonable degree of accuracy. The ability of the SE network to provide good long-range predictions for this process is further illustrated in Fig. 10 which also shows the corresponding poor performance of the ND network.

6. Conclusions

The results from applying the MLP neural network to model real and simulated non-linear dynamical systems demonstrated that significant improvements in the long-range prediction accuracy could be obtained using the SE method of representing network data compared to the conventional normalisation technique. This improvement in performance is generally at the expense of a larger network; however, results from modelling a real process showed that a network employing spread encoding, although larger than a network using normalised data, can require less training iterations to provide reliable long-range predictions. Thus, the use of spread encoding has significant advantages in neural network applications requiring long-range prediction, such as in neural network based predictive control schemes and using networks as a simulation tool for modelling non-linear dynamical systems. Both data representation methods resulted in networks capable of accurate single-step predictions and, in applications where one-step-ahead prediction is sufficient, the data normalisation method may be preferable because it usually results in a smaller network, thus requiring less computation.

Acknowledgements

The authors would like to thank the EPSRC, UK, for financial support of J.T. Evans and the Liverpool John Moores University Research Fund and British Nuclear Fuels Ltd. (Sellafield) for financial support of S.K. Doherty.

References

[1] N.V. Bhat and T.J. McAvoy, Determining model structure for neural models by network stripping, *Comput. Chem. Eng.* **16** (1992) 271–281.

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

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

[4] G. Cybenko, Approximations by superposition of a sigmoidal function, *Math. Control Signal Systems* **2** (1989) 303–314.

[5] S.K. Doherty, J.B. Gomm and D. Williams, Practical considerations on the implementation of neural networks for non-linear system identification, *Proc. IEEE/IMACS Internat Symp. on Signal Processing, Robotics and Neural Nets*, Lille, France (1994) 564–567.

[6] S.K. Doherty, J.B. Gomm, D. Williams and D.C. Eardley, Design issues in applying neural networks to model highly non-linear processes, *Proc. IEE Internat. Conf. Control '94*, Warwick, UK (1994) 1478–1483.

[7] C.R. Gent and C.P. Sheppard, Predicting time series by a fully connected neural network trained by back propagation, *Proc. IEEE Coll. on Neural Networks in Control and Modelling of Industrial Processes*, Polytechnic of Central London, UK (1991) 9/1–9/6.

[8] J.B. Gomm, J.T. Evans, D. Williams and P.J.G. Lisboa, Development of a neural network model based controller for a non-linear process application, *Proc. Workshop on Neural Network Applications and Tools*, Liverpool, England (IEEE Computer Society Press, CA, 1994) 109–117.

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

[10] J.C. Hoskins and D.M. Himmelblau, Artificial neural network models of knowledge representation in chemical engineering, *Comput. Chem. Eng.* **12** (1988) 881–890.

[11] K.J. Hunt and D. Sbarbaro, Neural networks for non-linear internal model control, *IEE Proc. D* **138** (1991) 431–438.

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

[13] C.-T. Lin and C.S.G. Lee, Neural-network-based fuzzy logic control and decision system, *IEEE Trans. Comput.* **40** (1991) 1320–1336.

[14] L. Ljung, *System Identification: Theory for the User* (Prentice-Hall, London, 1987).

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

[16] D. Marr and E. Hildreth, Theory of edge detection, *Proc. Roy. Soc. B* 207 (1980).

[17] K.S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Networks* **1** (1990) 4–27.

[18] S.-Z. Qin, H.-T. Su and T.J. McAvoy, Comparison of four neural net learning methods for dynamic system identification, *IEEE Trans. Neural Networks* **3** (1992) 122–130.

[19] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning representations by back-propagating errors, *Nature* **323** (1986) 533–536.

[20] D. Tsaptsinos, N.A. Jalel and J.R. Leigh, Estimation of state variables of a fermentation process via Kalman filter and neural network, in: G.F. Page, J.B. Gomm and D. Williams, Eds., *Application of Neural Networks to Modelling and Control* (Chapman & Hall, London, 1993) 53–73.

[21] R.J. Williams and D. Zipster, A learning algorithm for continuously running fully recurrent neural networks, *Neural Comput.* **1** (1989) 339–356.

[22] M.J. Willis, G.A. Montague, C. Di Massimo, M.T. Tham and A.J. Morris, Artificial neural networks in process estimation and control, *Automatica* **28** (1992) 1181–1187.

[23] L.A. Zadeh, Fuzzy sets, *Inform. Control* **8** (1965) 338–353.