A practical application of neural modelling and predictive control

6

J.T. Evans, J.B. Gomm, D. Williams, P.J.G. Lisboa and Q.S. To

6.1 ABSTRACT

This chapter describes a method of obtaining an accurate neural network model of a non-linear system and its use in a practical neural control scheme. A technique of coding the data presented to a neural network (called spread encoding) is described and it is shown that using this technique greatly improves the accuracy of the neural network model compared to the conventional method of applying a single data value to a single input node of the network. Simulation results are presented and then results are shown of modelling a real laboratory scale process. The resulting neural network model is then incorporated into a neural predictive control strategy and used to predict future process outputs, which are in turn used to calculate process control input values so as to make the process behave in the same way as a specified reference model.

6.2 INTRODUCTION

Recent years have seen a significant increase in interest in the application of artificial neural networks to the modelling and control of systems. This interest is mainly due to the learning capabilities of neural networks, and the fact that multi-layer, feedforward networks can approximate any non-linear function with arbitrary accuracy [1]. This chapter describes the application of the multi-layer perceptron (MLP) neural network, trained using standard back-error propagation, to obtain a representative model of

a physical non-linear process over a wide operational region. Subsequent implementation of the neural network model in a predictive control strategy is described and results are presented of the scheme applied on-line to control a laboratory process.

The usefulness of the neural network model in a predictive control strategy depends strongly on the ability of the model to predict accurately multiple time steps ahead. A NARMAX-type structure [2, 3] for the neural network model is used in these studies, within which lagged process input and output data are used to predict future process outputs. With this type of model the method by which data is represented to the network greatly influences the model prediction accuracy because future predictions rely on previous network outputs that are fed back to the network inputs. Hence, prediction errors can accumulate resulting in poor prediction accuracy. One approach to improve the network accuracy is to include dynamics within the neural network structure as described in [4]. An alternative approach to obtain an accurate model of the system is presented here that utilizes a new data conditioning method whereby each process data value presented to the neural network is distributed over a number of input nodes, rather than just one as is usually the case. The prediction accuracy of a neural network model trained with this method is compared to that of a network trained with the conventional data conditioning method of applying the data for each network input to a single input node [2, 4–6].

When an accurate model of the system under consideration is obtained, it can be used in a variety of control strategies. Some forms of control strategies that have utilized neural networks include the following.

- Internal model control [3]. Here a radial basis function neural network was used to model a non-linear system and it was found that, to achieve a good level of accuracy, a network consisting of a very large number of processing nodes was required.
- Another possible approach of using neural networks in a control strategy is to use direct inverse control as described in [7]. The disadvantage of this method is that the control structure relies heavily on the ability to obtain an inverse model of the system, which may not always be possible.
- A third method is that of model reference control where the performance of the system under closed loop is specified by a stable reference model. Implementation of this control structure is described extensively in [8].

The control strategy implemented in this chapter is that of predictive control which directly utilizes the neural model to predict future process outputs and appropriate input control signals are computed to minimize a specified cost function.

6.3 DATA CONDITIONING

Two forms of data conditioning have been studied in this work and a comparison between the neural network's performance and relative merits in both cases is made in later sections. The standard data conditioning technique is to linearly map the data, between a specified lower and upper bound, for each network input and output node and apply the resulting values to single, individual network nodes [2, 4–6], the effective operational range of the network input and output nodes frequently being between 0.1 and 0.9. This approach will be referred to as single node data mapping, SNDM.

A new method of conditioning the data is proposed whereby each data value is spread over a number of nodes using a sliding Gaussian distribution as shown in Fig. 6.1 (referred to as spread encoding, SE). The range of the data is evenly discretized across the nodes such that each node has a value in the data range assigned to it. The coded data value is represented at the centre of the Gaussian excitatory pattern and is retrieved as a weighted sum of the excitations and data values of each node. The translation equations used for the spread encoding technique are described in Chapter 2 of this book and in [9]. In the results presented, each data value was spread over six nodes.

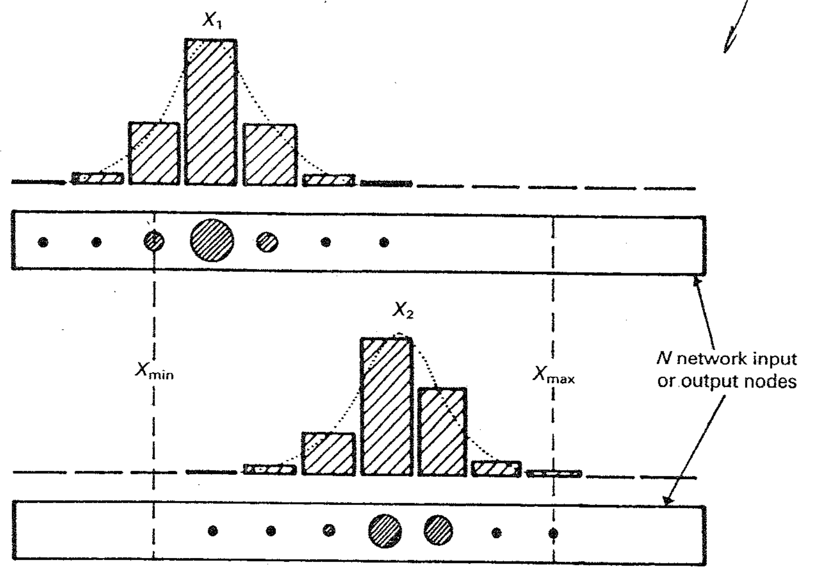

Fig. 6.1 Proposed spread encoding data conditioning technique. The figure illustrates the coding of two data values, x_1 and x_2 $(x_2 > x_1)$, in the range (x_{min}, x_{max}) to N network nodes.

The SE technique has been investigated since, in the area of measurement and process control, the physical variables of interest usually span a wide range of analogue values. If the range of these values is compressed to a single node, as in the case of SNDM, the amount of activity in relevant modes of operation may span only a small fraction of the full range of the processing node. The spread encoding technique overcomes this disadvantage by distributing the information over a number of the network nodes. This technique was also considered since many biological systems work on the principle of spreading the information that they receive, e.g. the retina of the eye.

Results are presented that show that the spread encoding technique greatly improves the accuracy of the neural network to model a process, compared to that of SNDM. One reason for this increase in accuracy could be that when using the spread encoding technique the problem is turned into one of a pattern recognition problem, which is well suited to neural network technology. The spread encoding technique increases the dimensionality of the network weight space, and this may at first seem a serious disadvantage. However, studies indicate that when using the SE technique the neural networks require fewer complete passes of the training data set and give a lower mean squared error than networks trained using the alternative SNDM method, as illustrated in Fig. 6.4 below.

6.4 THE PROCESS

The system considered in this paper is a dual-tank, non-interacting liquid-level system as shown in Fig. 6.2. Both the tanks are of the same size and capacity, the height being 1.25 m and the capacity of each tank being 20 litres. The process has inherent non-linearities and, to be more representative of systems encountered in practice, an unmeasured state in the height of liquid in the top tank. The non-linearity present in this system is due to the square root relationship between the tank outflows and the height of liquid in each tank. The laboratory process uses standard industrial equipment (Fig. 6.2). A pressure measurement, to measure liquid-level height, is taken from the bottom of tank 2. The signal is converted to a suitable voltage (in the range 0–5 V) for A/D conversion via a standard pressure to current converter (3–15 psi, 4–20 mA). The computer-controlled driving output (the control input) is a digital number between 100 and 200, which is converted to a pressure to drive the valve through its full operational range. The neural network and control software for the process were written in the Quick Basic programming language and implemented on an IBM PS/2 model 30 computer.

Initially a mathematical model of the liquid-level system was developed with the same characteristics as the laboratory system. The mathematical

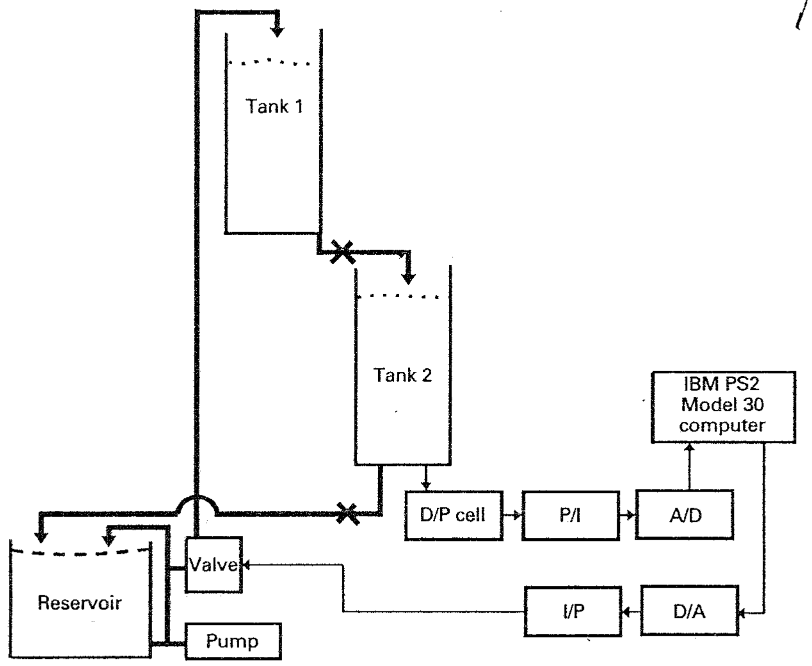

Fig. 6.2 Liquid-level laboratory process.

model of the liquid-level process was represented by the following well-known, non-linear differential equations:

$$\dot{h}_1 = \frac{1}{C_1}\left(K_v u - \frac{\sqrt{h_1}}{R_1}\right) \qquad (6.1)$$

$$\dot{h}_2 = \frac{1}{C_2}\left(\frac{\sqrt{h_1}}{R_1} - \frac{\sqrt{h_2}}{R_2}\right) \qquad (6.2)$$

where h_1 and h_2 are the liquid levels in tanks 1 and 2 respectively, u and h_2 are the process input and output, K_v is the valve gain, C_1, C_2, are the cross-sectional areas of tanks 1 and 2 respectively and R_1, R_2 are the outflow pipe restrictances of tank 1 and tank 2. The mathematical process model was simulated, using a standard continuous simulation package, in order to obtain data for training and validating the neural networks. Subsequently, the same approach was applied to the real process.

6.5 DEVELOPING A NEURAL NETWORK MODEL OF THE SYSTEM

An artificial neural network (ANN) model of the dual-tank, liquid-level system was initially developed prior to developing a control algorithm. In the development of this model, there are a number of issues that need consideration, such as the network topology, network training and validation, and these are discussed in the following subsections.

6.5.1 NEURAL NETWORK TOPOLOGY

It has been assumed that the input–output dynamics of the liquid-level process can be characterized by the general NARX (Non-linear, Auto-Regressive, eXogenous) model defined by:

$$y(t) = F(\,y(t-1), \ldots, y(t-n_a), u(t-k), \ldots, u(t-k-n_b)) + e(t) \qquad (6.3)$$

where F is some unknown non-linear function, y and u are the process outputs and inputs respectively, e is a Gaussian distributed white noise sequence, k is the process deadtime, n_a and n_b are the number of past output and input data used in the above non-linear difference equation. The approach taken was for the neural network model to have the same number of present and past values in its structure as would be taken in NARX input–output modelling. The results are presented for the ANN model structure: process deadtime, $k=1$; $n_a=2$ and $n_b=1$. Hence, two past inputs and two past outputs of the process are used as inputs to the neural network. These values result in the SNDM neural network having four input nodes and one output node. For the SE network, where each data value was mapped over six nodes, the topology consisted of 24 input nodes and six output nodes. In both cases just one hidden layer was used for each network structure. The number of nodes used in the hidden layer was determined empirically [9]. For the SNDM network eight hidden nodes were used and six nodes in the SE network. Each of the hidden and output nodes utilized a sigmoidal activation function. The next step, in the development of the neural network model, was to train the networks to be able to represent the dynamics of the dual-tank, liquid-level system.

6.5.2 NETWORK TRAINING

The neural networks are trained using the back-propagation algorithm, which is a form of the well-known, gradient descent algorithm. There are two algorithm parameters, namely, the gain and momentum which have to be selected to control the iterative process. The values of the gain and momentum were initially set at 0.9 and 0.6 respectively, and were manually reduced as the network training proceeded. If the gain term is initially small, then training the network can take a long time. However, if the initial value of the gain is not reduced during the training period the network may not reach its global minimum, since the large step size causes the network to oscillate around its global minimum

Both the neural networks (SNDM and SE) were trained as one-step-ahead predictors (Fig. 6.3(a)) because this is a more stable training configuration with the back-propagation algorithm than the model configuration [5]. It has been found, in this research, that if a neural network is overtrained, its generalization properties deteriorate when structured as a model of the process (Fig. 6.3(b)). On the other hand, if not trained long enough the generalization properties and accuracy of a network are poor. The approach used to select the optimum amount of training is as described in [9].

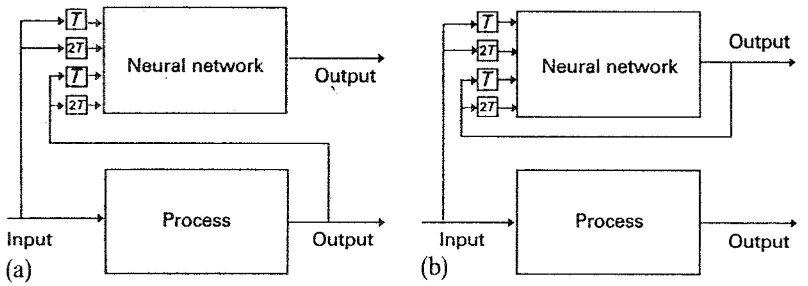

Fig. 6.3 Neural network training and recall configurations for process modelling: (a) predictor structure; (b) model structure.

Two different input–output data sets (D1 and D2), each containing 1000 pairs of input–output values, were generated from simulations of the mathematical model describing the dual-tank, liquid-level process with a random amplitude signal applied to the process input. One of the data sets (D1) is used to train the neural network in the one-step-ahead predictor configuration and the other (D2) is used to test the network's generalization properties in the model configuration at various stages of the training. Figure 6.4 shows the mean squared error of the network outputs for D2 (after decoding and scaling back to the original data range) against the number of complete passes of D1 for both the SNDM and the SE neural networks. The optimum amount of training is indicated by the lowest point in the graphs. Beyond this point it can be seen that the generalization properties as a model degrade for both the SE and SNDM networks. Figure 6.4 also shows that the SE network requires a smaller number of complete passes (300), of the training data, than that required for the SNDM network (700). Furthermore, the mean squared error for the SE network is significantly less than that of the SNDM network.

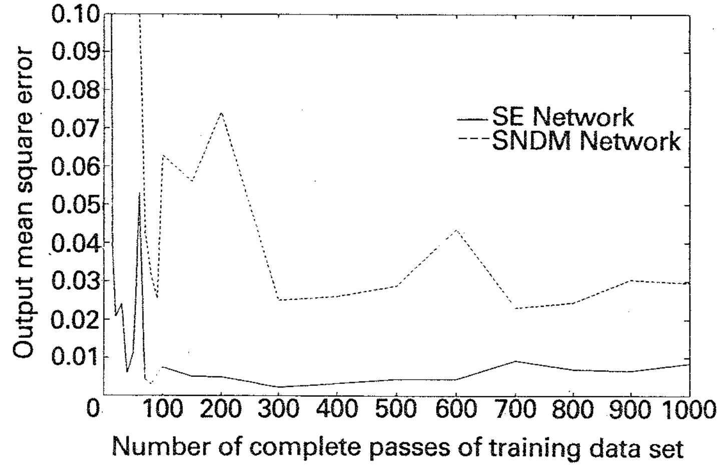

Fig. 6.4 Generalization tests for SNDM and SE networks during training.

6.5.3 MODEL VALIDATION

The neural networks were validated as both a one-step-ahead predictor and also as a model of the process. Figure 6.3(b) shows that when structured as a model, the network inputs consist of the past input data from the process and the past delayed output data from the neural network. Using the model configuration allows the network to be used independently of the process which is a desirable property. The two trained neural networks were validated in both structures on a number of test signals that had not been used in training. These included a random amplitude signal, a pseudo-random binary signal (PRBS) and a set of steps.

Figures 6.5 and 6.6 show the simulation results when recalling both the SE and SNDM networks as one-step-ahead predictors on a set of step responses. The set of steps presents a severe test for both the networks, since it spans a wide region of operation and indicates whether the neural networks have learned the required steady states. It can be seen from Figs 6.5 and 6.6 that both the networks perform adequately, the SE network having a slightly less mean square error than the SNDM network. However, when testing the two neural networks on the set of steps in the model configuration the SE network (Fig. 6.8) performs significantly better than that of the SNDM network in Fig. 6.7. This performance was repeated when the networks were tested with the PRBS and random amplitude signals and is shown in [10].

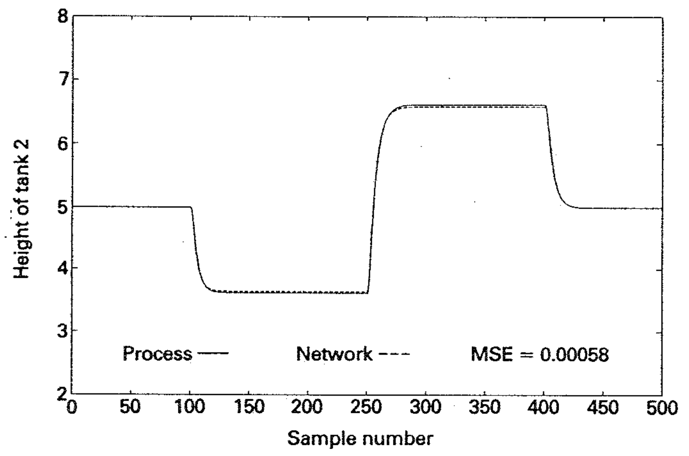

Fig. 6.5 SNDM network tested on a series of step inputs as a one-step-ahead predictor.

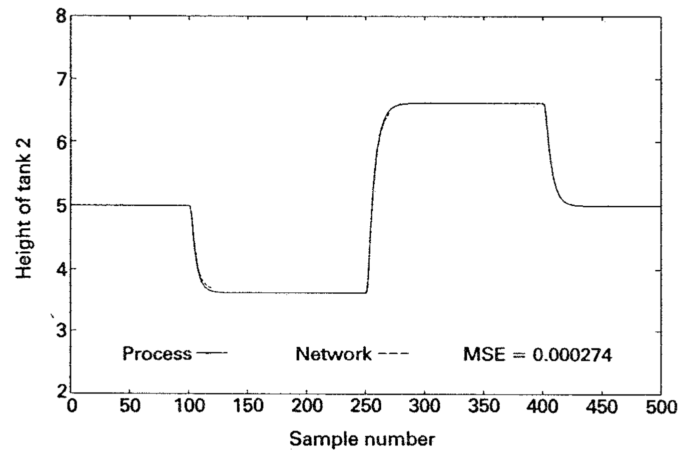

Fig. 6.6 SE network tested on a series of step inputs as a one-step-ahead predictor.

6.5.4 REAL PROCESS RESULTS

The simulation results described above indicate that spread encoding the data presented to the neural network gives a significant improvement in accuracy when using the neural network in a model configuration, and is slightly better when in the one-step-ahead predictor configuration. Hence, the laboratory process was only modelled using the spread encoding technique. Figure 6.9 compares the responses of the real process and that of

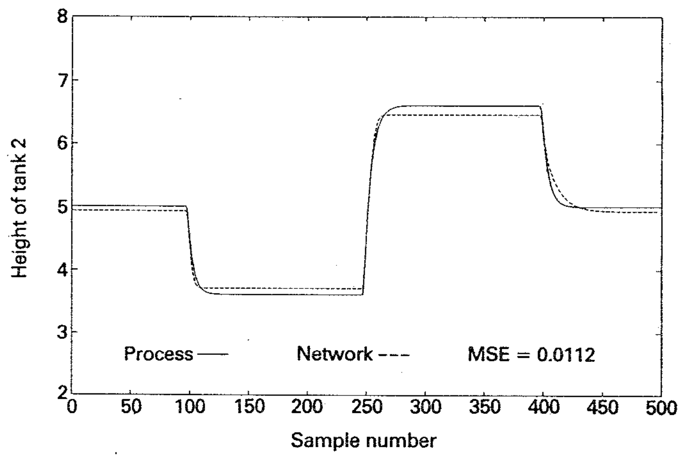

Fig. 6.7 SNDM network tested on a series of step inputs as a model.

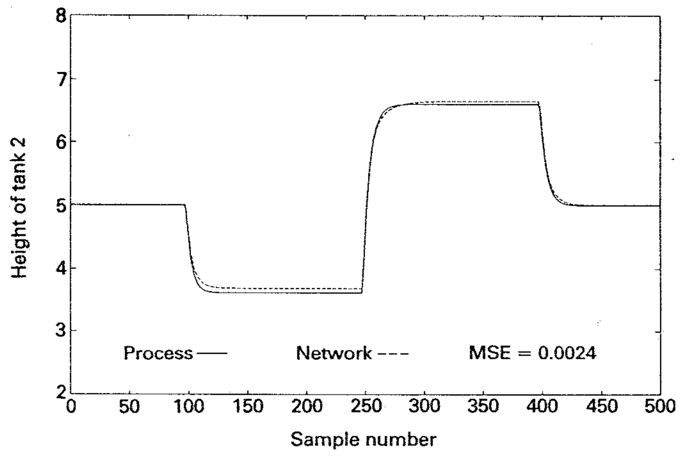

Fig. 6.8 SE network tested on a series of step inputs as a model.

the trained neural network when tested in a model configuration on a PRBS. The neural network accurately simulates the response of the process. It can be seen that on the second sequence of the PRBS signal, the height of liquid in tank 2 does not reach the same level as it did on sequence 1, although it does reach the level on the third sequence. This is indicative of time variations present in real processes and illustrates the differences when using a simulation model and data from a real process. The neural network model of the laboratory process was then recalled on a random amplitude signal as shown in Fig. 6.10, and again an acceptable result is obtained. The

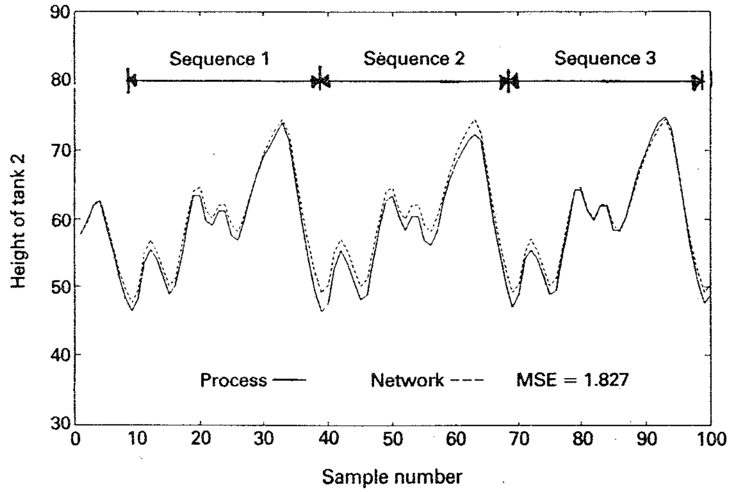

Fig. 6.9 Responses of SE network model and laboratory process to a PRBS.

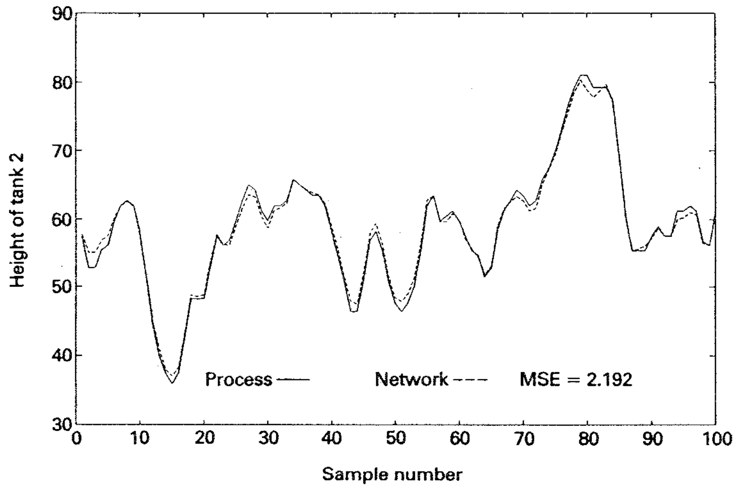

Fig. 6.10 SE network model and laboratory process responses to a random amplitude signal.

neural network model was also successfully validated as a one-step-ahead predictor on the above two signals.

6.6 NEURAL PREDICTIVE CONTROL: A PRELIMINARY STUDY

Having developed a suitable neural network model of the real system, it can be utilized in a control structure. The control structure adopted is illustrated

in Fig. 6.11 and is that of neural predictive control. The neural network model is used to predict future process outputs, and the control signals are chosen so as to minimize a suitable cost function in terms of future output deviations from the required set point values. The control signals are specified by minimizing the following cost function:

$$J=\sum_{k=1}^{N}(y_{\mathrm{ref}}(t+k)-y_{\mathrm{nn}}(t+k))^2+\sum_{k=1}^{N}\lambda(u(t+k-1)-u(t+k-2))^2 \quad (6.4)$$

where y_{ref} and y_{nn} are the outputs of the reference model and neural network respectively, λ is a weighting factor in the interval (0, 1), u is the controller

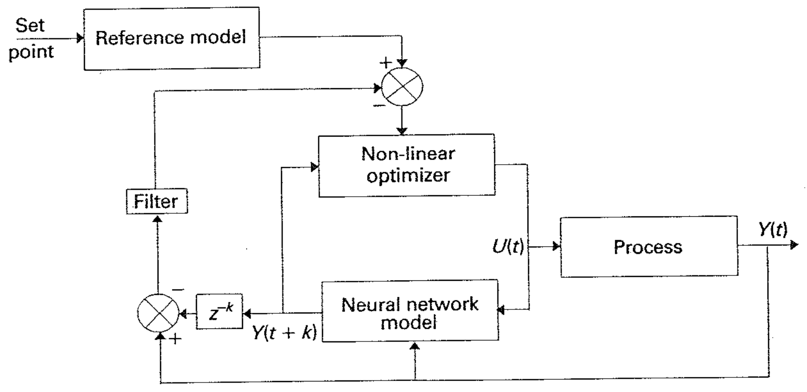

Fig. 6.11 Neural predictive control strategy.

output and N is the prediction horizon. In the results presented only one-step-ahead prediction was considered, hence, $N=1$ in equation (6.4). A reference model of the form:

$$G(s)=\frac{\omega_n^2}{s^2+2c\omega_n s+\omega_n^2} \quad (6.5)$$

was used to specify how the liquid-level process was required to behave. For this study the damping ratio, c, was chosen as 0.7 and ω_n was calculated to obtain a 2% settling time of 7 minutes.

6.6.1 ON-LINE RESULTS

Figure 6.12 shows the required reference model output and the height of the second tank in the system for λ, in equation (6.4), equal to zero, i.e. no constraint on the input value to the valve. It is clearly seen that the process tracks the reference model as required. Figure 6.13 shows the corresponding control input which shows that with λ set to zero the control input is not a

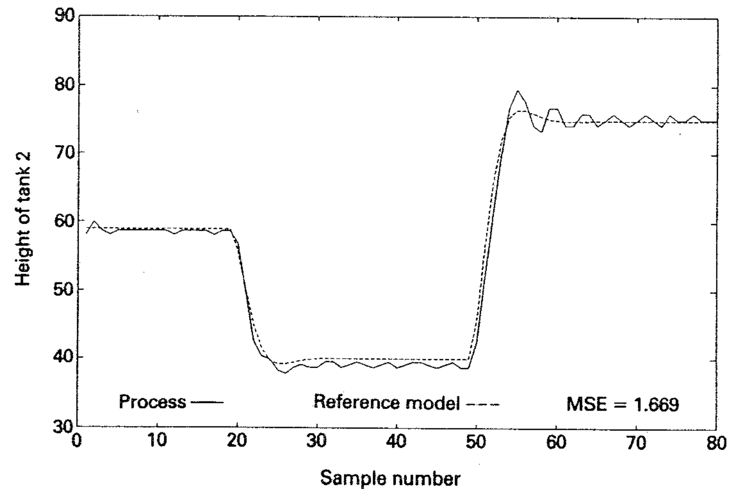

Fig. 6.12 On-line predictive control results for $\lambda=0$ in equation (6.4).

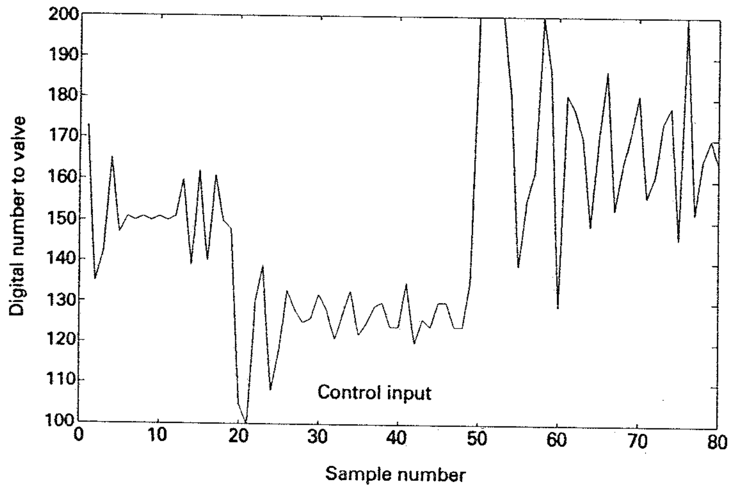

Fig. 6.13 Control input for $\lambda=0$.

desirable one since the valve in the system is continually subjected to excessive movement. The effect of increasing the weighting factor, λ, to 1 can be seen in Figs 6.14 and 6.15. The control valve movement is much smoother but at the expected expense of a poorer control performance.

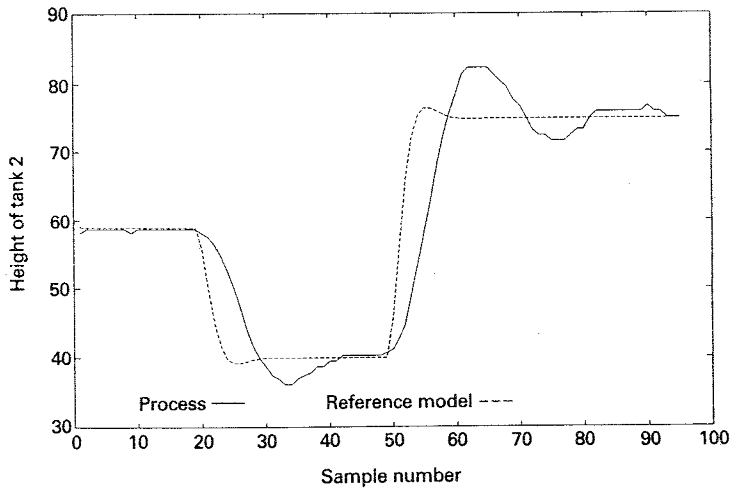

Fig. 6.14 On-line predictive control results for $\lambda=1$.

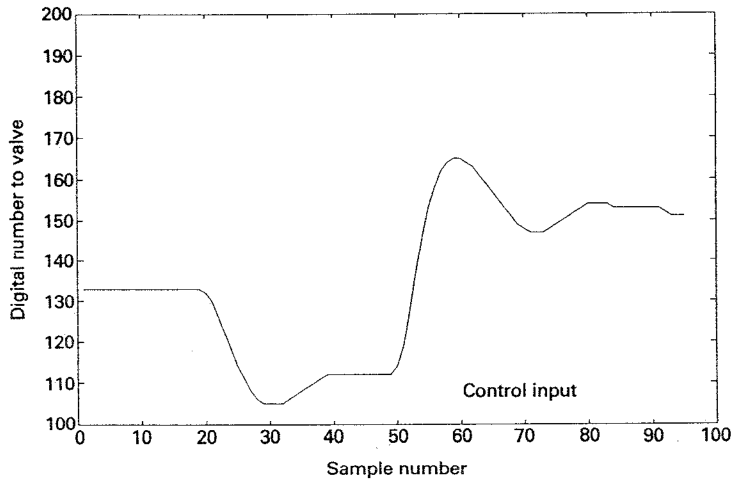

Fig. 6.15 Control input for $\lambda=1$

6.7 CONCLUSIONS AND FURTHER WORK

It has been shown that it is feasible to use artificial neural networks to model non-linear dynamic processes, and using the proposed spread encoding data conditioning technique results in improved accuracy when the neural network is required to act as a process model. The neural network representation of the real process has been successfully incorporated into a predictive control strategy, and on-line results were presented which demonstrate that stable and accurate control is possible.

Work is in progress to investigate multi-step-ahead predictive control, as this should attenuate the excessive movement of the valve which occurred for one-step-ahead prediction when no constraint on the control input was present. Also other control strategies implementing neural networks are being investigated and a comparison of the neural network methods and conventional PID control is to be made. The spread encoding of the process data is also to be implemented on a more complex modelling and control problem.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the Science and Engineering Research Council for financial support of J.T. Evans and Q.S. To, and also Unilever Research Laboratories for CASE support of Q.S. To.

REFERENCES

1. Hornik, K., Stinchcombe, M. and White, H. (1989) Multistage feed-forward networks are universal approximators. *Neural Networks*, **2**, 359–66.
2. Chen, S., Billings, S.A. and Grant, P.M. (1990) Non-linear system identification using neural networks. *Int. J. Control*, **51**(6), 1191–1214.
3. Hunt, K.J. and Sbarbaro, D. (1991) Neural networks for non-linear internal model control. *IEE Proc. D.*, **138**(5), 431–8.
4. Willis, M.J., Di Massimo, C., Montague, G.A., Tham, M.T. and Morris, A.J. (1991) Artificial neural networks in process engineering. *IEE Proc. D.*, **138**(3), 256–66.
5. Narendra, K.S. and Parthasarathy, K. (1990) Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks*, **1**(1), 4–27.
6. Bhat, N.V., Minderman Jr, P.A., McAvoy, T.J. and Sun Wang, N. (1990) Modelling chemical process systems via neural computation. *IEEE Control Systems Magazine*, **April**, 24–9.
7. Barto, A.G. (1990) Connectionist learning for control: an overview. In *Neural Networks for Control* (ed. W. Thomas Miller III, R.S. Sutton and P.J. Werbos). M.I.T. Press, Cambridge, Mass., 5–58.
8. Narendra, K.S. and Parthasarathy, K. (1989) Neural networks and dynamical systems. Part III: Control. Internal Report No. 8909, Yale University.
9. Gomm, J.B., Lisboa, P.J.G., Williams, D., Evans, J.T. and To, Q.S. (1992) Neural networks for accurate modelling of non-linear process dynamics. Submitted to *IEE Proc. D.*, July 1992.
10. Evans, J.T., Gomm, J.B., Williams, D., Lisboa, P.J.G. and To, Q.S. (1992) Non-linear process modelling with artificial neural networks. *Proc. 24th SCS Summer Computer Simulation Conference*, Reno, USA, 27–30 July, pp. 637–41.

A label-driven CMAC intelligent control strategy 7

A.J. Lawrence and C.J. Harris

7.1 ABSTRACT

An *a priori* unknown SISO linear plant, subject to parametric variations, is posed as a control problem for the development of a cerebellar model articulation controller (CMAC) based adaptive control strategy. The plant output is characterized by labels (e.g. rise time, overshoot, steady state error) which serve the purpose of data compression whilst extracting salient features for control. Desired labels are generated from a reference model. Two CMAC modules are required for plant modelling and controller tuning. On-line tuning is performed by issuing a step demand to both the CMAC and reference models, the outputs of which are used to generate a label error vector. A CMAC tuning module maps the label error to a controller gain change and the controller is updated. The CMAC algorithm is capable of learning a wide class of non-linear mappings from training data and hence is not explicitly programmed.

7.2 INTRODUCTION

This work is part of a project which aims to develop an intelligent strategy for the control of aeronautical gas turbine engines. An adaptive control strategy is described and some preliminary results presented.

The cerebellar model articulation controller (CMAC) was developed for solving robot arm articulation problems in which the many degrees of freedom made conventional approaches difficult [1, 2, 3]. CMAC has been successfully applied to robot control problems [4] and pattern recognition, signal processing and adaptive control [5]. CMAC belongs to a class of