

This article was downloaded by:

On: 5 January 2010

Access details: *Access Details: Free Access*

Publisher *Taylor & Francis*

Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



International Journal of Systems Science

Publication details, including instructions for authors and subscription information:

<http://www.informaworld.com/smpp/title~content=t713697751>

A recurrent network for dynamic system identification

Sandeep Adwankar ^a; Ravi N. Banavar ^a

^a Systems and Control Engineering, Indian Institute of Technology, Bombay, India

To cite this Article Adwankar, Sandeep and Banavar, Ravi N.(1997) 'A recurrent network for dynamic system identification', International Journal of Systems Science, 28: 12, 1239 – 1250

To link to this Article: DOI: 10.1080/00207729708929481

URL: <http://dx.doi.org/10.1080/00207729708929481>

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: <http://www.informaworld.com/terms-and-conditions-of-access.pdf>

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

A recurrent network for dynamic system identification

SANDEEP ADWANKAR† and RAVI N. BANAVAR†

This paper presents a type of recurrent artificial neural network architecture for identification of an arbitrary, continuous dynamic system. The recurrent network is shown to be stable for a constant input with certain conditions on the parameters of the network. The proposed network has significant advantages over similar models in continuous time nonlinear system identification and is used to identify three nonlinear dynamic systems. Finally, the applicability of the radial basis function networks using the same network architecture to reduce the time-complexity of the training task is presented.

1. Introduction

Real world processes are often characterized by complex nonlinear behaviour and cannot be treated satisfactorily by linear system paradigms. Artificial neural networks (ANN) which are inherently nonlinear and have an arbitrary function approximation property can provide a better means of modelling such complicated nonlinear systems. From the system identification point of view, a feedforward network represents a static nonlinear functional mapping between the input and the output of the network. Hence, identification of a dynamic system can only be possible in the framework of dynamic networks, unless high dimensional inputs through input/output feedback regressors are acceptable.

Dynamic neural networks are those in which the node equations are defined by differential or difference equations. Broadly, the architecture of dynamic networks can be classified as follows.

- (1) Time delay neural network (TDNN). Here, a static network processes the time series data by converting a temporal sequence into a static pattern by unfolding the sequence over time. From the practical point of view the network could be unfolded over only a finite period.
- (2) Recurrent networks. These are nonlinear dynamic systems having output or state feedback. The Hopfield is a fully connected recurrent network

with no hidden layer (Hopfield 1982). Recurrent networks analysed here have hidden nodes since they represent complex functional mapping and arbitrary dynamics.

There are two general concepts of recurrent network training.

- (a) Fixed point learning is aimed at making the network reach the prescribed equilibria or perform steady state matching, e.g. recurrent back-propagation (Hunt *et al.* 1992).
- (b) Trajectory learning which trains the network to follow the desired trajectories in time. As $t \rightarrow \infty$ the trajectory reaches a prescribed steady state, hence it is a generalization of the fixed point learning algorithm.

The significant contributions in trajectory learning are as follows. Back-propagation through time in which the temporal operation of the network is unfolded into a multilayer feedforward network, the topology of which grows by one layer at every time step. The activity levels of the nodes of the network are tracked and errors are back-propagated over the network starting from the last time instant, working back one by one. Real-time recurrent learning (Hush and Horne 1993) for the training of recurrent networks is for online training tasks. Dynamic back-propagation (Narendra and Parthasarathy 1991) computes the gradient of the error function with respect to the parameters of the network by giving a generalized systems approach by dividing any dynamic system into four different classes.

The computational complexity of the above recurrent network training algorithms is considerably higher than

Received 3 February 1997. Revised 28 April 1997. Accepted 8 May 1997.

† Systems and Control Engineering, Indian Institute of Technology, Bombay – 400 076, India.

that of static training procedures like static back-propagation. Two significant contributions which use only a static training procedure for recurrent network training are as follows.

- (a) The series-parallel method (Narendra and Parthasarathy 1990) employs a type of TDNN with output feedback through the second tapped delay line, thus giving delayed values of input and output to a feedforward network and uses static back-propagation to adjust parameters of the network.
- (b) A recent contribution (Olurotimi 1994) is a method of training the recurrent network through static training of the embedded feedforward structure followed by certain transformations to obtain a fully recurrent network.

This paper presents a recurrent network architecture for continuous dynamic system identification. The recurrent network training involves only a static training procedure. The architecture and the associated training scheme have significant advantages over the others prevailing in nonlinear dynamic system identification. The paper is organized as follows. Section 2 presents the recurrent network architecture and proves the stability of the recurrent network in the presence of a constant input. In §3 the training procedure is illustrated with three simulation examples of nonlinear dynamic systems. Section 4 compares the time-complexity of the training task using radial basis function networks in place of feedforward networks using a sigmoidal nonlinearity on the same systems as in §3.

2. System identification

We consider systems of the form

$$\dot{x}(t) = f(x(t), i(t)) \tag{1}$$

where i is the external input to the system ($i(t) \in R^p$), x is the state vector of the system ($x(t) \in R^n$) and f is the unknown continuous vector function.

This is assumed to be the general representation of a dynamic system which is to be identified by the recurrent network. Further, the following assumptions are made about the system.

- (1) The order of the system, n , is known. This is a design choice, although *a priori* knowledge of the system could give a choice close enough to the actual value.
- (2) All the states of the system are either observed through measurements or could be constructed from the output in a phase variable form.

2.1. Network architecture

Given a continuous dynamic system of the form described by (1), the dynamics of the proposed recurrent network are

$$\dot{x}(t) = W_2 g(w_1 x(t) + W_{in} i(t) + b_1) + b_2 \tag{2}$$

where

- W_1 weight matrix from the input state vector $x(t)$ to the hidden layer of the network. $x(t) \in R^n$
- b_1 bias vector input to the hidden layer
- W_2 weight matrix from the hidden layer of the network to the output layer
- b_2 bias vector input to the output layer
- $g(\cdot)$ vector of node output functions
- W_{in} weight matrix from external input, $i(t)$ to the hidden layer of the network $i(t) \in R^p$

Examining (2) it can be noticed that the right-hand side may be viewed as a feedforward network whose inputs are $x(t)$ and $i(t)$. Associated with $x(t)$ and $i(t)$ are the weight matrices $W_1 \in R^{m \times n}$ and $W_{in} \in R^{m \times p}$ respectively. The hidden layer has m nonlinear processing elements $g(\cdot)$ (nodes). The output of the m hidden layer nodes is weighted by the matrix W_2 to give the output vector $\dot{x}(t)$. Hence the feedforward architecture is represented in the form $N_{p+n,m,n}^2$.

The functional approximation property that a feedforward network, with only a single hidden layer having a continuous bounded monotonically increasing nonlinearity, can arbitrarily well approximate any continuous function as well as derivatives of such a function is well proved (Hornik *et al.* 1992). Thus the right-hand side of (2) approximates any continuous function, in this case

$$f(x(t), i(t)) \in N_{p+n,m,n}^2 \tag{3}$$

Hence the dynamic system (2) representing a recurrent network is capable of approximating system (1) and its diagrammatic representation is as in Fig. 1.

2.2. Analysis of the recurrent network

The recurrent network could be analysed in terms of its internal state dynamics. Define

$$z(t) \triangleq W_1 x(t) + W_{in} i(t) + b_1 \tag{4}$$

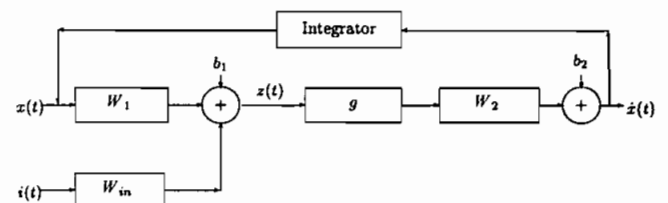


Figure 1. Recurrent network architecture.

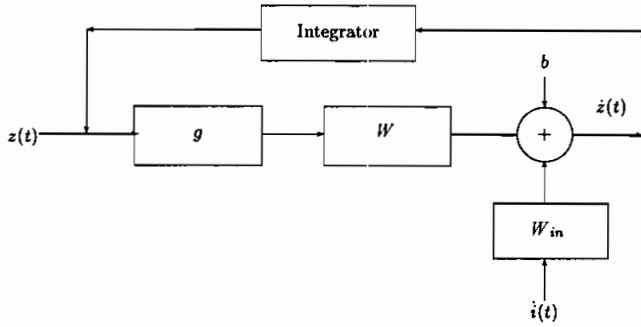


Figure 2. Recurrent network in terms of an internal variable.

for stationary weights, W . Then

$$\dot{z}(t) = W_1 \dot{x}(t) - W_{in} \dot{i}(t) \tag{5}$$

Substituting in (2), yields

$$\dot{z}(t) = W_1 W_2 g(z(t)) + W_{in} \dot{i}(t) + W_1 b_2 \tag{6}$$

Assuming $i(t)$ is differentiable, the above equation can be rewritten as

$$\dot{z}(t) = Wg(z(t)) + W_{in} \dot{i}(t) + b \tag{7}$$

where $W \triangleq W_1 W_2$, $b \triangleq W_1 b_2$. A diagrammatic representation of the recurrence in the intermediate variable is shown in Fig. 2. Note that the derivative of the internal state vector is affected by the derivative of the input vector.

Consider the situation when the input to the system ($i(t)$) is of the form

$$i(t) = \begin{cases} [c_1 c_2 \dots c_p]^T & t \geq 0 \\ \{00 \dots 0\}^T & t < 0 \end{cases}$$

Then (7) reduces to

$$\dot{z}(t) = Wg(z(t)) + b \quad (t \geq 0) \tag{8}$$

which could be represented as in Fig. 3.

Theorem 1: *If W is symmetric, i.e. $W_{(i,j)} = W_{(j,i)} \forall i, j$, g is monotonically increasing and bounded, then the state $z(t)$ of the network moves to an equilibrium as $t \rightarrow \infty$.*

Proof: Equation (8) could be represented as

$$\frac{dz_i(t)}{dt} = \sum_{j=1}^N W_{ij} g_j(z_j(t)) + b_i \quad i = 1, 2, \dots, N \tag{9}$$

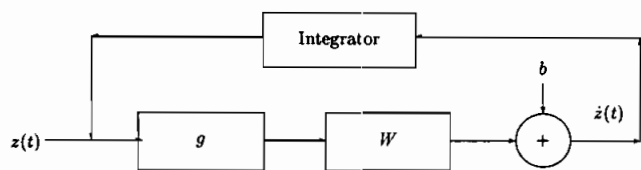


Figure 3. Recurrent network in terms of an internal variable and a constant input.

Now let

$$y_j(t) = g_j(z_j(t)) \quad j = 1, 2, \dots, N \tag{10}$$

Consider the function

$$E(t) \triangleq -\frac{1}{2} y^T(t) W y(t) - y^T(t) b \tag{11}$$

Since both $y(t)$ and b are bounded, the function $E(t)$ is bounded above and below. The differential of this function with respect to time

$$\frac{dE(t)}{dt} = -\frac{1}{2} \left[2 \sum_{i=1}^N \frac{dy_i(t)}{dt} \sum_{j=1}^N W_{ij} y_j(t) \right] - \sum_{i=1}^N \frac{dy_i(t)}{dt} b_i \tag{12}$$

$$\frac{dE(t)}{dt} = - \sum_{i=1}^N \frac{dy_i(t)}{dt} \left[\sum_{j=1}^N W_{ij} y_j(t) + b_i \right] \tag{13}$$

From (9) and (10) it follows

$$\frac{dE(t)}{dt} = - \sum_{i=1}^N \frac{dy_i(t)}{dt} \left[\frac{dz_i(t)}{dt} \right] \tag{14}$$

also from (10)

$$\frac{dE(t)}{dt} = - \sum_{i=1}^N \frac{dg_i}{dz_i} \frac{dz_i(t)}{dt} \left[\frac{dz_i(t)}{dt} \right] \tag{15}$$

$$\frac{dE(t)}{dt} = - \sum_{i=1}^N \frac{dg_i}{dz_i} \left[\frac{dz_i(t)}{dt} \right]^2 \tag{16}$$

As g_i , $i = 1, 2, \dots, N$ is monotonically increasing, its slope is always positive. Hence

$$\frac{dE(t)}{dt} \leq 0$$

and

$$\frac{dE(t)}{dt} = 0$$

only when

$$\frac{dz_i(t)}{dt} = 0 \quad \text{for } i = 1, 2, \dots, N$$

which corresponds to an equilibrium state of the equation. Thus, the above recurrent network always converges to an equilibrium state.

Corollary: *If*

$$\lim_{t \rightarrow \infty} z(t) = c_1$$

then

$$\lim_{t \rightarrow \infty} x(t) = c_2$$

where c_1, c_2 are arbitrary real constants.

Proof: The proof follows from (4), (5) and the fact that the input $i(t)$ is constant. \square

From Theorem 1 and the Corollary, the recurrent network (2) is asymptotically stable under the assumed conditions and will eventually reach an equilibrium point in the neighbourhood of the initial condition. This property also suggests the usefulness of the network in finding the equilibrium points (steady-state fixed point attractor dynamics) of a dynamic system. Thus, the network may converge to different equilibria dependent on the initial condition.

A linear term could be incorporated in (2) as is done in most of the recurrent network literature (Hush and Home 1993) yielding

$$\dot{x}(t) = -ax + W_2g(w_1x(t) + W_{in}i(t) + b_1) + b_2 \quad (17)$$

where a is a constant. However, the inclusion of this linear term does not have any useful role from the stability point of view and also complicates the training procedure. Furthermore, the simulation results here showed that (2) gave superior results compared to (17). Hence, in subsequent work, the linear term is not considered.

The training of the recurrent network proceeds as follows. The right-hand side of (2) is a feedforward network and its parameters are obtained by static back-propagation, with input to the feedforward network as a vector $[i(t) \ x(t)]^T$ and output as vector $[\dot{x}(t)]$. The state vector $[x(t)]$ is obtained by integrating $[\dot{x}(t)]$ and is given as a state vector input to the network thus producing a recurrent network.

2.3. Features of the recurrent network

The salient features of the above introduced architecture and its advantages over other methods are as follows.

- (i) The stability of the recurrent network in the presence of constant input is guaranteed.
- (ii) The recurrent network provides for universal approximation of any dynamic system, provided the static map of the state derivatives is implementable by a feedforward network.
- (iii) The training procedure being static, VCdim considerations (Hush and Home 1993) will lead to approximately determining the number of nodes in the hidden layer, given the number of training patterns, for good generalization of the network.
- (iv) The training procedure is indirect in the sense that the feedforward training is for the derivatives of the states and not the actual output.
- (v) The network training consists of only static training as the mapping to be implemented is only static. Thus, the training procedure requires less memory compared with the recurrent network

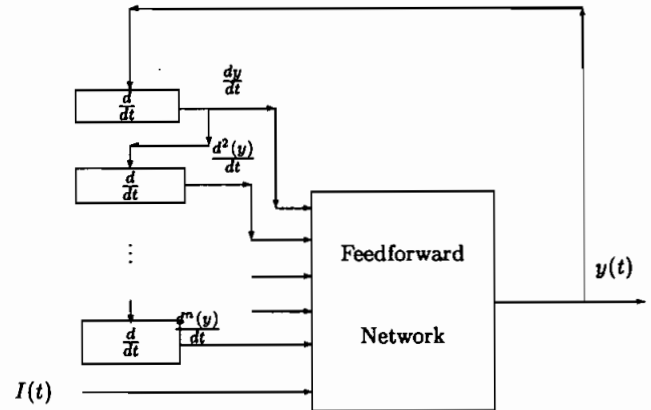


Figure 4. Continuous time identification.

training schemes, like back-propagation through time, which require back-propagation through space as well as through time.

- (vi) The advantages over a similar approach adopted by Olurotimi (1994) are:

- (1) the derivative of the input vector is not required. This is a serious hindrance in real time operation since the derivative is a non-causal operator;
- (2) the matrix inversion of w_1 (required by Olurotimi 1994), the weight matrix from the input state vector to the first hidden layer of the network, which is normally non-square, can be obtained only after certain manipulations.

- (vii) A discrete-time system identification method is proposed by Narendra and Parthasarathy (1990). If the philosophy is extended to the continuous time case, the structure of the model is as shown in Fig. 4 where derivatives of the output are required to be taken n times in the actual operation (n being the order of the system). In real-time operation, the differential operators may operate on noisy outputs of the system, which may not be desirable, while in the proposed method the differential operators are replaced by integration operators.

3. Simulation examples

The system identification method discussed in the previous section was applied to a representative set of dynamic systems. Although parallel and distributed processing features are not available on a serial machine, they still provide a good environment for testing the capabilities of an ANN. The three systems considered were:

- (1) the Bessel differential equation;
- (2) the Van der Pol oscillator;

(3) a bioreactor.

3.1. The Bessel function

The first system illustrated is a Bessel differential equation characterized by a time-varying nonlinear differential equation given by

$$t^2 \ddot{y}(t) + t \dot{y}(t) + (t^2 - \alpha_2)y(t) = 0 \quad (18)$$

The input to this system is order (α) and time (t). The system can be cast in a state-space variable form as

$$\dot{x}_1(t) = x_2(t) \quad (19)$$

$$\dot{x}_2(t) = -\frac{(t^2 - \alpha^2)}{t^2} x_1(t) - \frac{x_2(t)}{t} \quad (20)$$

The phase variable form chosen is advantageous because the feedforward network could be trained for only the n th derivative of the first stage. In the trained recurrent network the input state vector is obtained by integrating the output n times. The feedforward training procedure is thus simplified since there is only one output to train, irrespective of the order of the system.

The training data are composed of:

- (1) the input training set is the string $[\alpha, t, x_1(t), x_2(t)]^T$;
- (2) the output training set is $[x_2(t), \dot{x}_2(t)]$.

The training data are generated for the time interval $7 \leq t \leq 17$ for the orders $\alpha = 1, 3, 5, 7$ and for states $-0.8 \leq x_1, x_2 \leq 0.8$. The input training set is sampled linearly for the entire interval so as to contain the entire range of variation that is required for proper operation of the trained recurrent network. The output of the recurrent network after integration is fed back to the input, and these inputs must be bounded in the range of the input training set or the range of input training set must be large enough to bound all possible future inputs to the network. At the sampling point, each component of the input training set is varied randomly in the vicinity of 10% of its value to avoid monotonicity and to ensure a rich training data which captures even very small details of the system. It was found that such a training data set also speeds up the network training. The network training procedure is illustrated in Fig. 5. The training algorithm used is the standard back-propagation algorithm with a momentum term which is designed to minimize the sum squared error E between the output of the network and the desired output. The update rule for any weight matrix W is

$$w_{ij}(t+1) = w_{ij}(t) - \eta \frac{\partial E}{\partial w_{ij}} + \beta(w_{ij}(t) - w_{ij}(t-1)) \quad (21)$$

where η is the learning rate parameter and β the momentum parameter. The network configuration used was $N_{4,7,1}$. The generalization capability of an ANN can be

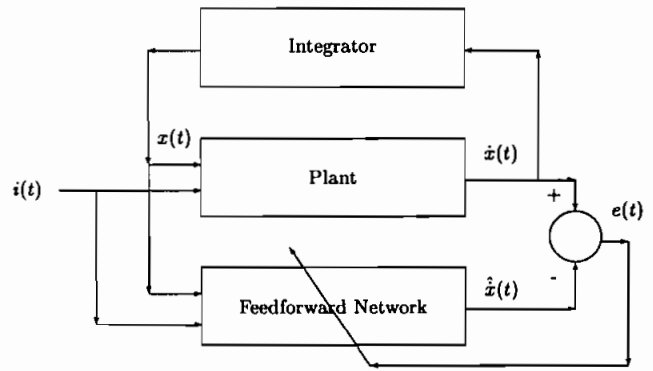


Figure 5. Network training procedure.

assured when the number of training samples exceeds a parameter called the Vapnik-Chervonenkis Dimension (VCdim). The VCdim of the feedforward network, having a single hidden layer and using hard-limiting nonlinearities, is shown to be bounded and is given as

$$VCdim \leq 2N_w \log_2(N_n e) \quad (22)$$

where N_w = no. of weights + no. of thresholds, N_n = no. of nodes and $e = 2.7182$. As a rule of thumb, for good generalization, the number of training patterns should approximately be ten times that of the VCdim value for a feedforward network having continuous nonlinearities (Hush and Horne 1993). The number of nodes in the hidden layer were so chosen as to satisfy the VCdim criterion required for proper generalization of the network, given the number of training patterns.

To hasten the training procedure, hyperbolic tangent sigmoid nonlinearities were used in the hidden layer (Haykin 1994). The batch training procedure was employed to give equal importance to all training patterns in updating of weights. A momentum term was used along with the gradient term to escape from shallow local minima. The learning rate parameter (η) was continuously varied so as to lessen the training time. The network was trained for 14 000 iterations after which the maximum fractional error in the output of the feedforward network was reduced to 0.01. The trained network was cross-validated on a data set which was not used in the training of the network.

The recurrent neural network was simulated in the configuration shown in Fig. 1 and the states were obtained by numerical integration (third-order Runge-Kutta method) of the state derivative. The integration step size was kept small enough to keep the numerical iteration error to a small value for good performance of the recurrent network.

Figure 6 compares the output y of the recurrent network with that obtained from solving the differential equation (18) for various orders. It is evident that the recurrent network identifies the Bessel function very well. The responses are shown for a recurrent network which is trained for data having random noise of 10% of

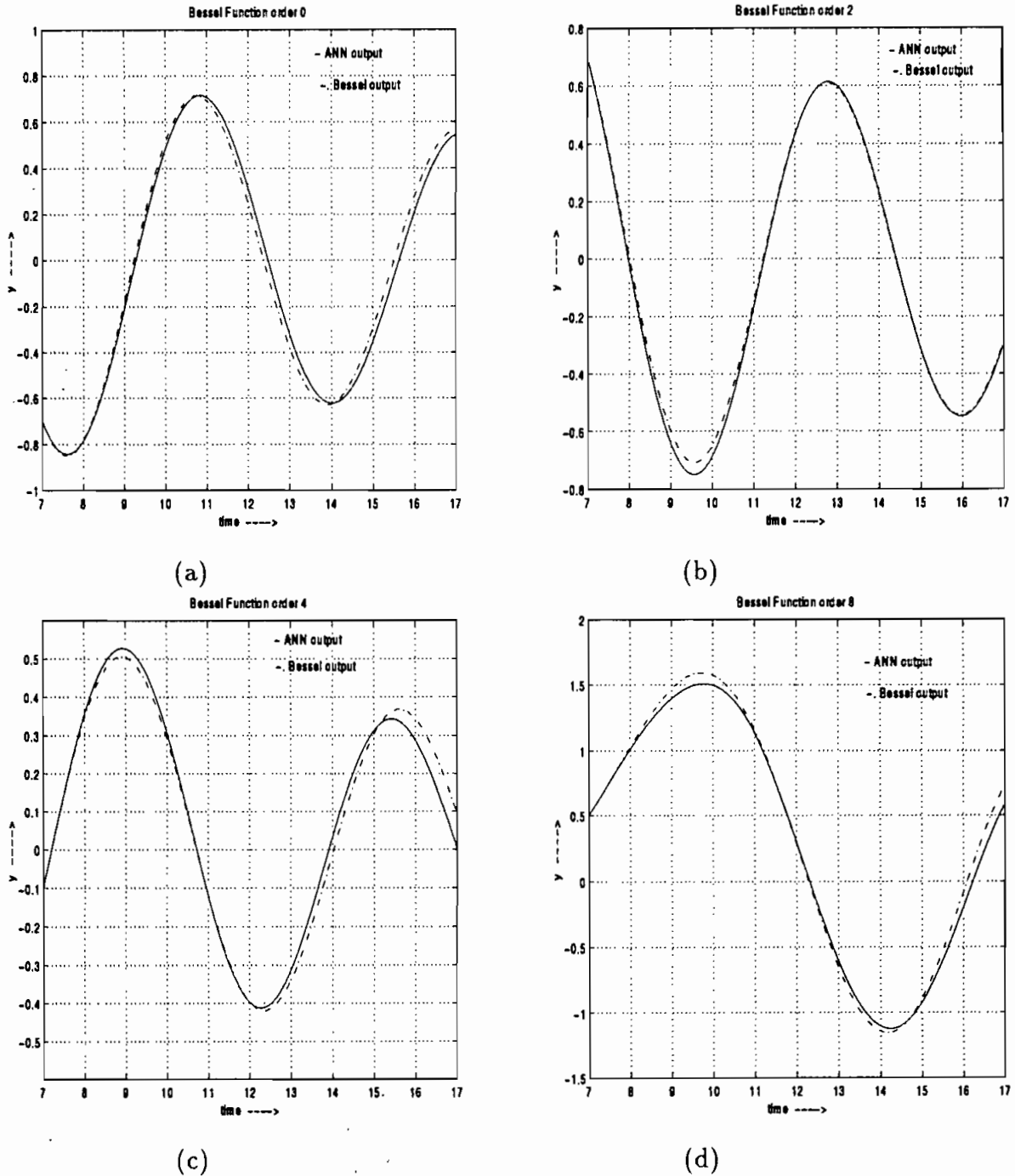
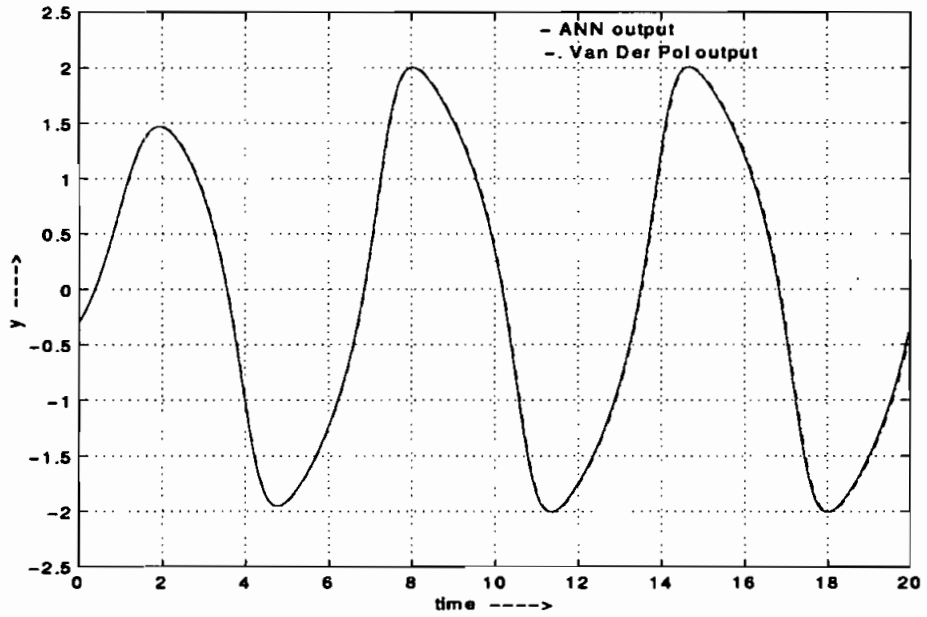


Figure 6. (a) Bessel function Order 0; (b) Order 2; (c) Order 4; (d) Order 8.

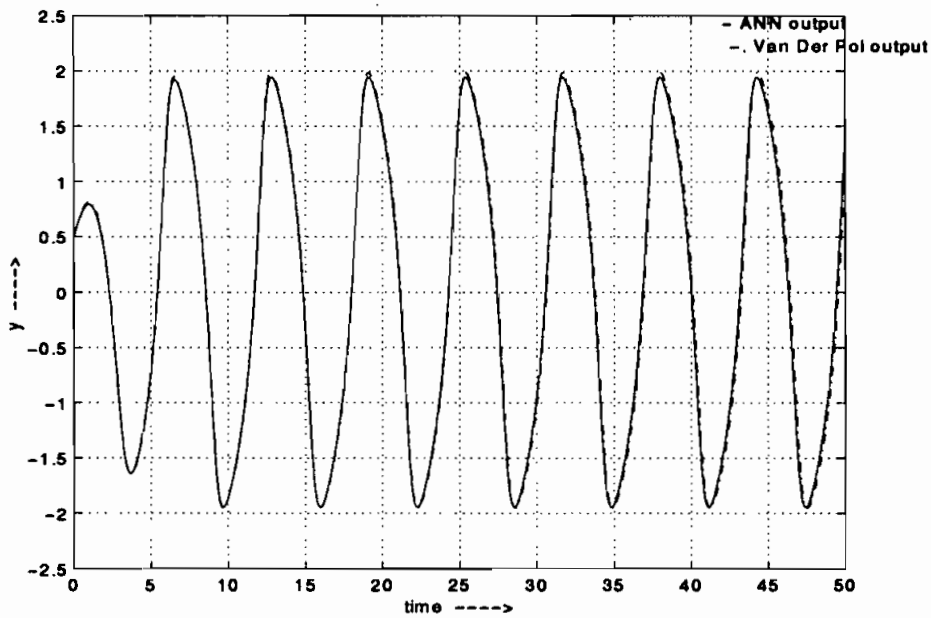
the maximum value of each state. This is to observe the response of the recurrent network to the noise that will be superimposed on the actual data in identification of a practical system. The responses showed the robustness of the network to noise in the training data. Figure 6 also shows the generalization ability of this recurrent network for orders $\alpha = 0, 2, 4, 8$ for which the recurrent network is not trained.

3.2. The Van der Pol oscillator

The next system studied was the well known Van der Pol oscillator which exhibits a stable limit cycle, i.e. system trajectories starting from any non-zero initial states, approach a limit cycle. The system is characterized by the absence of any specific input and shows sustained constant amplitude oscillations for any initial



(a)



(b)

Figure 7. Van der Pol oscillator identification.

condition. The system equations in the state space form are given as

$$\left. \begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -x_1(t) + (1 - x_1^2(t))x_2(t) \end{aligned} \right\} \quad (23)$$

The input to the feedback network is vector $[x_1(t) \ x_2(t)]^T$ and the output of the feedforward network is $[\hat{x}_2(t)]$. A feedforward network $N_{2,5,1}^2$ was

trained for 9000 iterations and was simulated by following the same procedure as outlined for the Bessel function. The output of the recurrent network is plotted with that obtained from (23) for different initial conditions as shown in Fig. 7. The performance of the recurrent network is sensitive to the interval of simulation time since numerical integration errors accumulate over a period of time.

3.3. Identification of a bioreactor

To test the validity and benefits of the network, a real-life dynamic system in the form of a bioreactor (for producing lactic acid) was chosen. The bioreactor is a complex dynamic nonlinear chemical process consisting of a substrate or nutrients and biological cells. Cells consume nutrients and produce product. In the continuously stirred bioreactor considered here, there is an input flow of substrate lactose whose flow rate is equal to outgoing flow rate. The micro-organism *Lactobacillus bulgaricus* converts lactose through a number of intermediate stages eventually to lactic acid. The identification of this system is particularly difficult for the following reasons.

- (1) Biological cells are self-regulatory mechanisms and can adjust their growth rate and production of different products radically depending on environmental conditions like temperature and concentration.
- (2) The process is nonlinear and when coupled with the inhibitory effects of the product and substrate concentration, which themselves are dependent on the pH, the system becomes highly nonlinear.
- (3) The yield coefficient determined by the amount of fresh biomass produced per unit mass of nutrient, if varying during the growth process, will lead to multiplicity, i.e. two flow rates leading to the same cell mass yield and can also lead to a limit cycle under certain conditions of the flow rate (Agarwal *et al.* 1982).

The conventional procedure of obtaining a model consists of fitting a curve through experimentally obtained data points, thus giving a mathematical model. The model of a lactic acid bioreactor obtained by such a method (Venkatesh *et al.* 1993) is as follows:

$$\left. \begin{aligned} \dot{x}(t) &= -x(t)r(t) + \mu x(t) \\ \dot{s}(t) &= (s_0 - s(t))r(t) - (\gamma\mu + 0.95(\alpha\mu + \beta))x(t) \\ \dot{p}(t) &= -p(t)r(t) + (\alpha\mu + \beta)x(t) \\ \mu &= \frac{\mu_{\max}s(t)}{k_s + s(t)} \left(1 - \frac{l^-(t)}{95} \right) \exp(-0.85lh) \\ p(t) &= lh(t) + l^-(t) \\ \frac{l^-(t)}{lh(t)} &= 1.426 \times 10^{pH-4} \\ pH &= 6 \exp(-0.088p(t)) \\ \alpha &= 26.25(pH) - 2.344(pH)^2 - 64 \\ \mu_{\max} &= 1.655(pH) - 0.1426(pH)^2 - 4.09 \end{aligned} \right\} (24)$$

where

x	concentration of the cell mass in the bioreactor (g l^{-1})
s	concentration of the substrate (lactose) (g l^{-1})
p	concentration of the product (lactic acid) (g l^{-1})
lh	concentration of the undissociated lactic acid (g l^{-1})
l^-	concentration of the dissociated lactic acid (g l^{-1})
r	dilution rate (hr^{-1})
k_s	substrate limiting constant (1.2)
α	growth-associated product formation constant
β	non-growth associated product formation constant (0.8)
γ	empirically determined stoichiometric constant for lactose conversion to cell mass (1.12)
μ	specific growth rate
μ_{\max}	Monod's growth constant

The conventional method of obtaining a mathematical model has the following limitations.

- (1) The accuracy of a curve fitting over data points is limited and could not be improved upon.
- (2) The model coefficients vary drastically not only with a change in the micro-organism or the substrate but also with the environmental condition.
- (3) The model does not incorporate some features like micro-organism viability, endogenous respiration, dead cell concentration, or transfer of gases out of liquid phase into gas phase.

Artificial neural networks promise to be very good identifiers as the activity levels could be increased to the desired value by proper choice of the network parameters and extended training. The network could also be subjected to training with the new data available by further running of the bioreactor. Thus, the network could represent the changed features. Further, since the bioreactor is a slow system, changes in the concentration being in the time unit of hours, an online identification is feasible.

The bioreactor was operated at maximum productivity of $p \cdot r = 9.2$ ($\text{g l}^{-1} \text{h}^{-1}$) at a flow rate 0.3 (hr^{-1}). The system model is required at the same flow rate. The outputs of the model are cell concentration, substrate concentration and product concentration. The network was trained with input as $[x(t) \ s(t) \ p(t)]^T$ and with output as $[x(t) \ s(t) \ p(t)]$. The feedforward network $N_{3,15,3}^2$ was trained for 15000 iterations. The data obtained from the actual process, being inadequate, was supplemented with that obtained from (24). When the trained network gave a maximum fractional error in the output as 0.01, it was tested by cross-validation and the training procedure was stopped and a recurrent

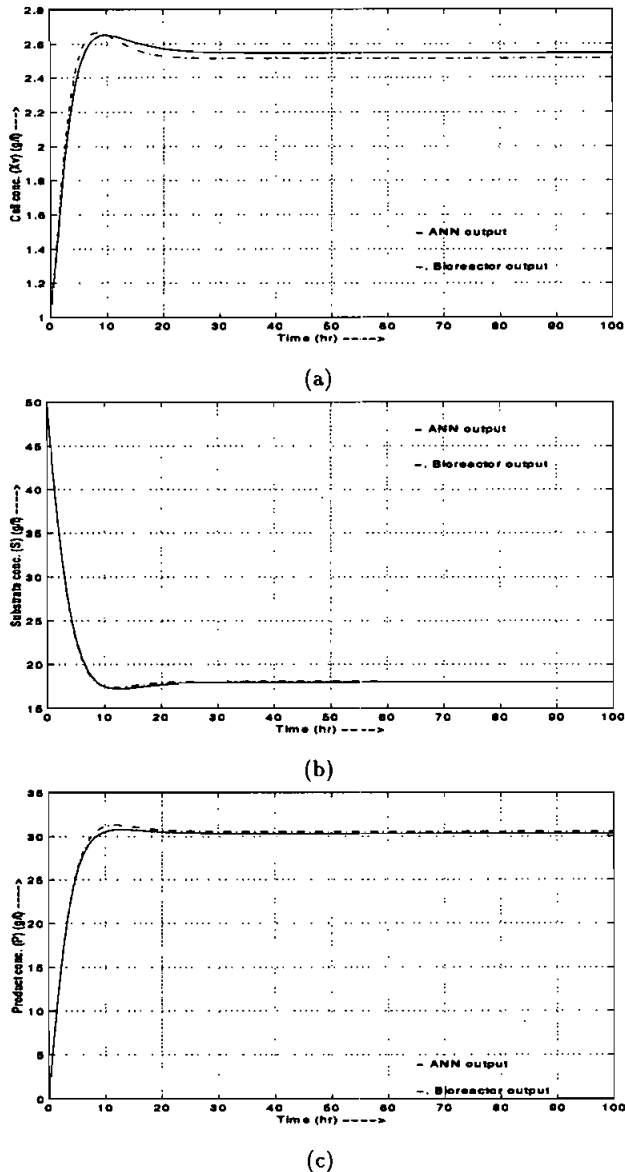


Figure 8. Lactic acid bioreactor identification.

network constructed. Figure 8 compares the response of the recurrent network identifier to the actual system.

4. The RBF network

Over the past few years, radial basis function (RBF) networks using exponentially decaying localized nonlinearities have been studied to construct approximations to the nonlinear input–output mapping. It has been claimed that these networks are capable of learning faster than conventional ones. The sigmoidal nonlinearity employed in the previous section is replaced by a Gaussian nonlinearity in the feedforward network.

The RBF network in its basic form involves two layers. The input to the network is given by source

nodes to a hidden layer of high enough dimension, which has the Gaussian functions as its nonlinearity. The output layer is composed of linear nodes. Each node in the hidden layer initially computes the euclidean norm of the input vector with respect to the centre of that node and passes it through a gaussian function as given in (25). The output u_j of the j th node in the first layer

$$u_j = \exp\left(\frac{-(i - w_j)^T(i - w_j)}{2 * \sigma^2}\right) \quad j = 1, 2, \dots, N \quad (25)$$

where i is the input vector pattern; w_j is the weight vector for the j th node in the first layer, the centre of the gaussian for node j ; σ is the normalization parameter; and N is the number of nodes in the first layer.

The orthogonal least squares learning algorithm (Chen *et al.* 1991) is used to choose the RBF centres so that it systematically adds a node at a time until a proper network is obtained and is tested by cross-validation. The normalization parameter is chosen to be large enough so that the active input region of nodes overlaps enough and several nodes have fairly large outputs at any given moment. With the RBFs, the Bessel differential equation (18) was identified with just 22 iterations in contrast to the 14 000 iterations taken by the sigmoidal network. The resultant network architecture was of the form $N_{4,23,1}^2$. The response of the RBF network is as shown in Fig. 9. It must be emphasized that the responses are for Bessel function orders, $\alpha = 0.5, 2.5, 5.5, 7.5$ none of which are used in the training of the network, thus illustrating the generalization abilities of the RBF network. This procedure was repeated for the identification of the Van der Pol oscillator which required 17 iterations and the network architecture was of the form $N_{2,18,1}^2$. The responses are shown in Fig. 10.

4.1. Training time complexity of RBF

The training task of a neural network (3 node) is shown to be NP-complete (Blum and Rivest 1988). Thus, there is no known optimal algorithm to solve the problem with a computation time that increases polynomially with the size of the problem. The simulation results, with the feedforward network having sigmoidal nonlinearity, support this. The simulation results with the RBF network demonstrate that the algorithm is polynomial time compared with the exponential time complexity of the sigmoid network training algorithm. This difference results from the basic change in the training philosophy. The sigmoid network training consists of loading optimum parameters given an arbitrary choice of the number of parameters. The RBF network on the other hand keeps the complexity of the problem

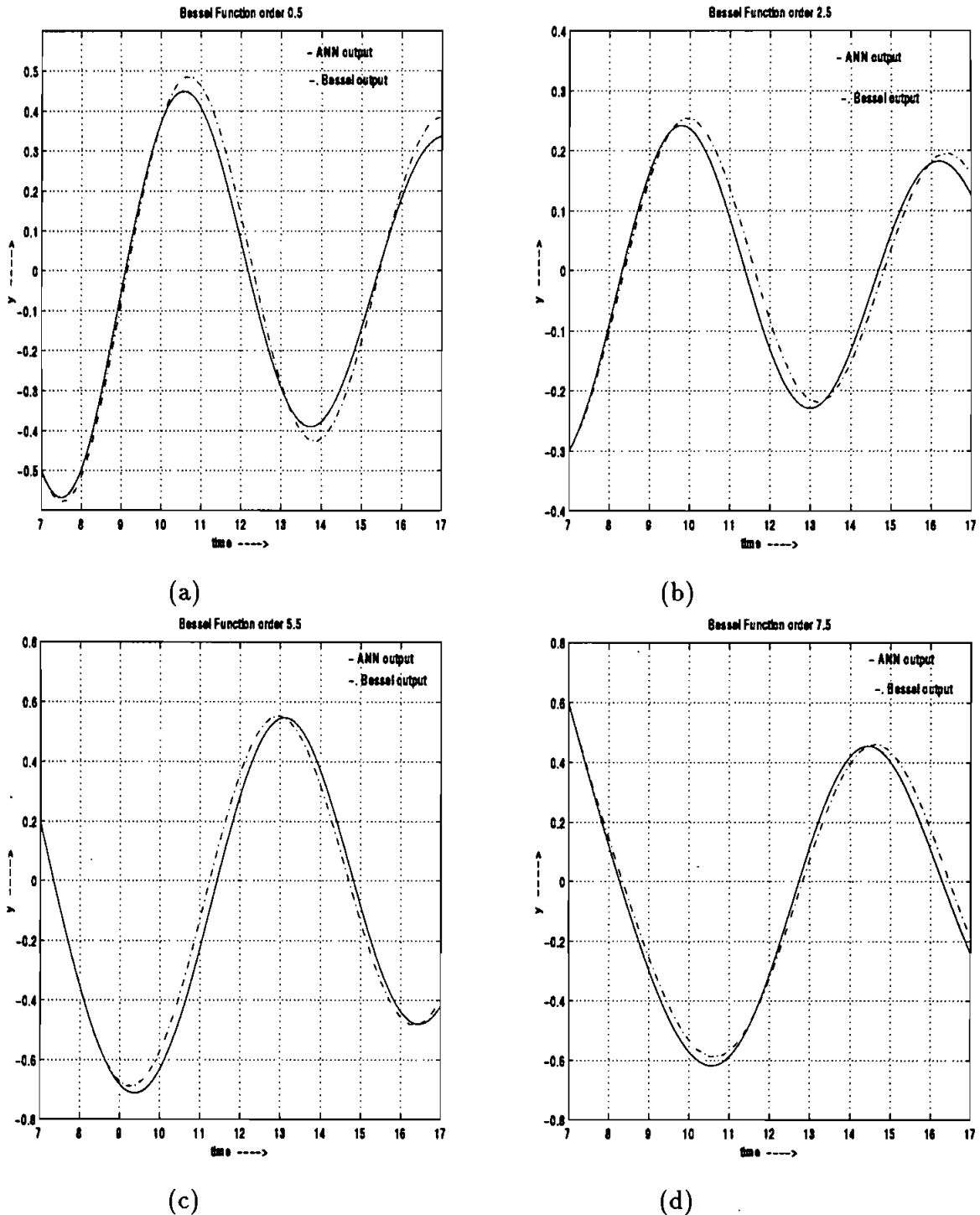
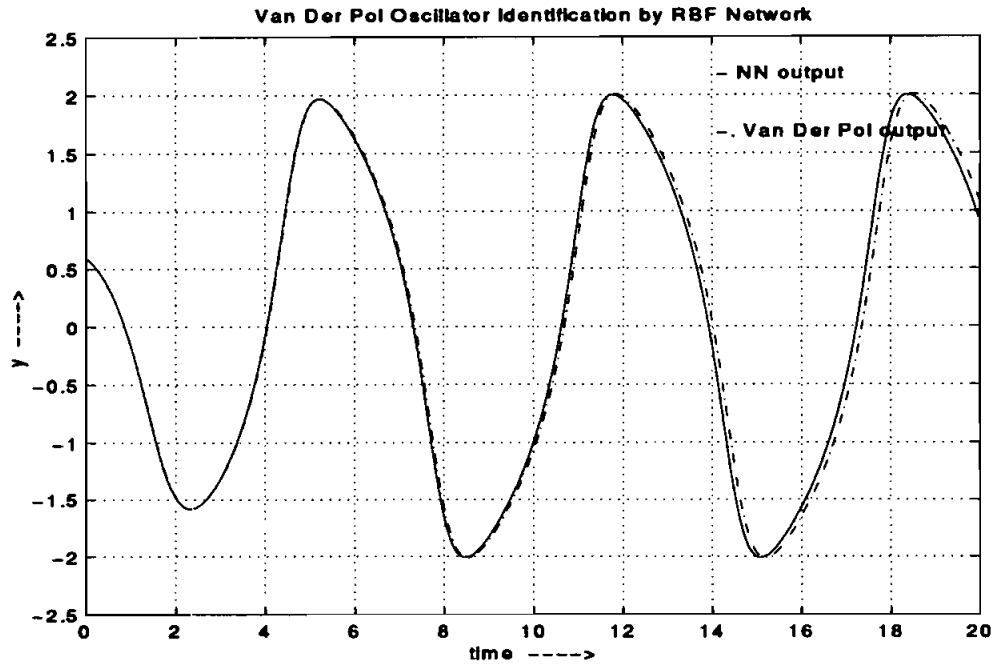


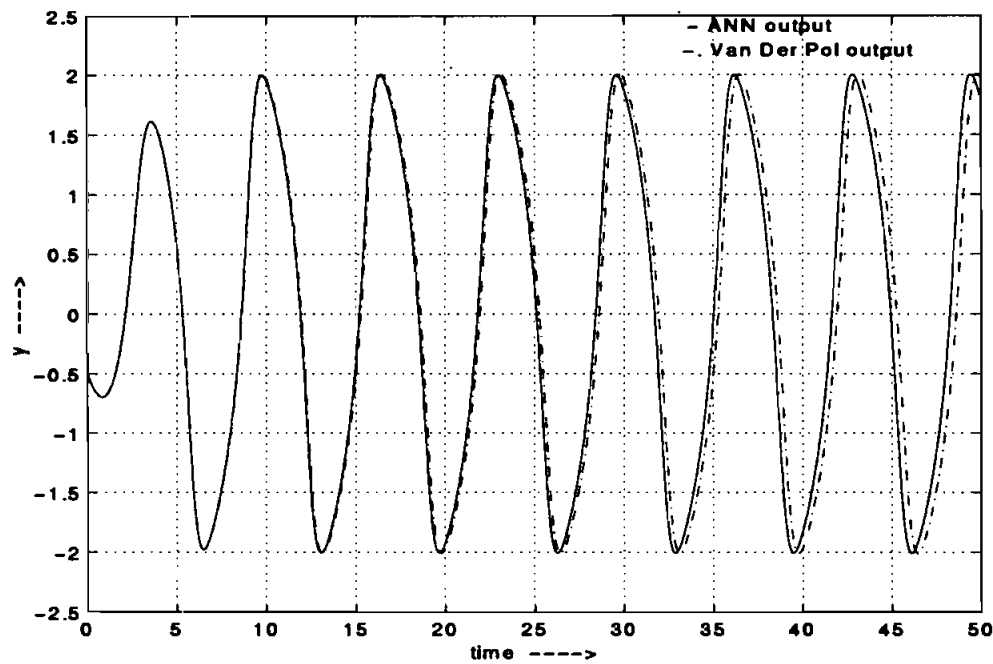
Figure 9. Bessel function identification by RBF networks (a) Bessel function Order 0; (b) Order 2; (c) Order 4; (d) Order 8.

the same at every step as it computes only a single parameter (vector)—the centre of the radial basis function—at a time and, if the error is below the required value, it then computes another centre of the RBF constructing a new node, keeping previously computed parameters at the same value.

The reduction in the training time complexity of the RBF networks is at the cost of an increase in the space complexity of the problem, which is basically the memory required for storing and giving outputs. In an ANN, the space complexity is governed by the number of adjustable parameters (weights + biases) in the net-



(a)



(b)

Figure 10. Van der Pol oscillator identification by RBF networks.

work. The simulation results show that the number of parameters required by the RBF network is greater than that required for the sigmoid network. The comparison between the RBF network and the sigmoid network in terms of the number of training iterations and the parameters of the network is given in Table 1.

The simulation results show that the training time complexity of the RBF networks is at least $O(10^2)$ lower than the training time complexity of the sigmoid networks, while the number of parameters of the RBF network is twice that of sigmoid network. Thus, RBF networks can provide a feasible alternative to sigmoid

Table 1. Comparison in RBF network and sigmoid network

System	RBF network		Sigmoid network	
	Iterations	Parameters	Iterations	Parameters
Bessel function	22	116	14 000	43
Van der Pol oscillator	17	55	9 000	21

networks, particularly where the model must adapt itself to new data continuously. However, for very complex problems like the bioreactor, feasible recurrent RBF networks can have a very large number of parameters. Thus, generalization of the network cannot be guaranteed as it does not satisfy the condition of VCdim (22). Hence, RBF networks could be used for problems where the model could be described with sufficient accuracy by the number of parameters which satisfy the VCdim condition and which require an online adaptation. They can also act as a means for determining the complexity of the problem in a very short time before the actual training task could be started.

5. Conclusions

A new recurrent ANN architecture has been proposed and shown to be stable for constant inputs. The performance of the recurrent network was found to be satisfactory for a variety of dynamic systems even with an additive random noise of amplitude of 10% of the maximum value of the states in the training data set. With radial basis functions, the training time complexity was at least $O(10^2)$ lower than that of networks with sigmoidal nonlinearities. The space complexity of the RBF network was, however, higher than that of the sigmoid network.

References

- AGARWAL, P., LEE, C. C., LIM, H. C., and RAMKRISHNA, D., 1982, Theoretical investigations of dynamic behaviour of isothermal continuously stirred tank biological reactor. *Chemical Engineering Science*, **37**, 453–462.
- BLUM, A., and RIVEST, R. I., 1988, Training a 3-node neural network is NP complete. *Proceedings of the Computational Learning Theory (COLT) Conference* (Morgan Kaufmann), pp. 9–18.
- CHEN, S., COWEN, F. N., and GRANT, P. M., 1991, Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, **2**, 144–150.
- HAYKIN, S., 1994, *Neural Networks—a Comprehensive Foundation* (New York: Macmillan).
- HOPFIELD, J. J., 1982, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Science, US*, **79**, 2554–2558.
- HORNIK, K., STINCHCOMBE, M., and WHITE, H., 1992, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Artificial Neural Network, Approximation and Learning Theory* (Blackwell).
- HUNT, K. J., SBARBARO, D., ZBIKOWSKI, R., and GAWTHROP, P. J., 1992, Neural networks for control systems—a survey. *Automatica*, **28**, 1083–1112.
- HUSH, D. R., and HORNE, B. G., 1993, Progress in supervised neural networks. *IEEE Signal Processing Magazine*, **10**, 8–39.
- NARENDRA, K. S., and PARTHASARATHY, K., 1990, Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, **1**, 4–27.
- NARENDRA, K. S., and PARTHASARATHY, K., 1991, Gradient methods for the optimization of dynamical systems containing neural networks. *IEEE Transactions on Neural Networks*, **2**, 252–262.
- OLUROTIMI, O., 1994, Recurrent neural network training with feedforward complexity. *IEEE Transactions on Neural Networks*, **5**, 185–197.
- VENKATESH, K. V., OKOS, M. R., and WANKAT, P. C., 1993, Kinetic model of growth and lactic acid production from lactose by *Lactobacillus bulgaricus*. *Process Biochemistry*, **28**, 231–241.