NEURAL NETWORKS FOR NON-LINEAR DYNAMIC SYSTEM MODELLING AND IDENTIFICATION

S. Chen† and S.A. Billings‡

†Department of Electrical Engineering
University of Edinburgh
Kings' Buildings
Edinburgh EH9 3JL
Scotland

‡Department of Automatic Control and Systems Engineering
University of Sheffield
Mappin street
Sheffield S1 4DU
England

Research Report No. 436

September 1991

Abstract

Many real-world systems exhibit complex non-linear characteristics and cannot be treated satisfactorily using linear systems theory. A neural network which has the ability to learn sophisticated non-linear relationships provides an ideal means of modelling complicated non-linear systems. This paper addresses the issues related to the identification of non-linear discrete-time dynamic systems using neural networks. Three network architectures, namely the multi-layer perceptron, the radial basis function network and the functional-link network, are presented and several learning or identification algorithms are derived. Advantages and disadvantages of these structures are discussed and illustrated using simulated and real data. Particular attention is given to the connections between existing techniques for non-linear systems identification and some aspects of neural network methodology, and this demonstrates that certain techniques employed in the neural network context have long been developed by the control engineering community.

1. Introduction

In the past decade there has been a strong resurgence in the field of artificial neural networks involving researchers from many diverse disciplines. This renewed interest in neural networks is fuelled by new network topologies, improved learning algorithms, and by emerging analogue VLSI implementation techniques. An introduction to various neural network models and learning strategies can be found, for example, in the article by Lippmann (1987) and in the books by Pao (1989), Beale and Jackson (1990), and Hecht-Nielsen (1990). Mead (1989) gives an up-to-date overview of analogue VLSI techniques for implementing neural networks.

A neural network typically consists of many simple computational elements or nodes arranged in layers and operating in parallel. The weights which define the strength of connection between the nodes are adapted during use to yield good performance. Neural networks are therefore specified by the network architectures, node characteristics and learning rules. A class of neural networks, where the input feeds forward through the network layers to the output, is referred to as the feedforward network. This kind of network is known to be capable of learning complex input-output mappings. That is, given a set of inputs and desired outputs or targets, an adequately chosen neural network can emulate the mechanism which produced the data set through learning. This scenario can obviously be included in the framework of system identification and signal prediction and, therefore, it is not surprising that many applications of neural networks to non-linear system identification and prediction have been reported (e.g. Lapedes and Farber 1988, Chen *et al.* 1990a, 1990c, Narendra and Parthasarathy 1990, Weigend *et al.* 1990).

The present study is concerned with the modelling and identification of non-linear discrete-time systems based on the neural network approach. Three different network architectures are considered, namely the multi-layer perceptron (MLP), the radial basis function network (RBF) and the functional-link network (FLN). The modelling capabilities of these network structures are analyzed, and both block-data and recursive identification algorithms are derived for each of these neural network models. Convergence properties of the learning algorithms are compared, and the advantages and disadvantages are discussed. The results are illustrated using data generated from both simulated and real systems.

Feedforward neural networks can of course be viewed as non-linear input-output models. Researchers working in the area of system modelling and identification have being actively investigating various non-linear models (e.g. Billings 1980, Billings and Leontaritis 1981, Billings *et al.* 1984, Billings and Chen 1989b, Chen and Billings 1989b, 1989c, Leontaritis and Billings 1985, Ozaki 1985, Priestley 1980). Many techniques employed in the field of neural networks have naturally also been developed by the control community and others. The famous back propagation algorithm (Rumelhart *et al.* 1986), for instance, is a simple version of the smoothed stochastic gradient algorithm widely used in system identification and adaptive control. The so-called FLN is identical to the concept of extended model set (Billings and Chen 1989b). These similarities between the neural network and more conventional methods are emphasised in the current study. Hopefully this will help to remove some of the myths of neural networks and to encourage a wider application of the neural network approach.

2. Problem formulation

Consider dynamic systems which are governed by the following non-linear relationship:

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

where

$$y(t)=[y_1(t)\cdots y_m(t)]^T, \tag{2}$$
$$u(t)=[u_1(t)\cdots u_r(t)]^T, \tag{3}$$
$$e(t)=[e_1(t)\cdots e_m(t)]^T, \tag{4}$$

are the system output, input and noise vectors respectively; n_y and n_u are the corresponding lags in the output and input; and $f()$ is some vector-valued non-linear function. The aim is to realize or to approximate the underlying dynamics $f()$ using neural networks. Introduce the network input vector

$$\mathbf{x}(t)=[y^T(t-1)\cdots y^T(t-n_y)u^T(t-1)\cdots u^T(t-n_u)]^T, \tag{5}$$

with a dimension

$$n_I=m\times n_y+r\times n_u. \tag{6}$$

The modelling and identification task can then be formulated as one of using the network input-output response

$$\hat{y}(t)=\hat{f}(\mathbf{x}(t)), \tag{7}$$

as the one-step-ahead predictor for $y(t)$.

The system representation (1) is a simplified case of the general non-linear system known as the NARMAX model (Billings and Leontaritis 1981, Leontaritis and Billings 1985, Chen and Billings 1989b):

$$y(t)=f(y(t-1),\dots,y(t-n_y),u(t-1),\dots,u(t-n_u),e(t-1),\dots,e(t-n_e))+e(t). \tag{8}$$

Although analysis for (8) in terms of stability and convergence is generally more difficult than that for (1), the techniques developed based on the simpler system (1) can readily be extended to the general case (8). Since the aim is to present a coherent structure of the neural network approach to system modelling and identification and to highlight the basic principles and concepts, all the discussions will be based on the system representation (1).

3. The multi-layer perceptron

The MLP is a layered network and each layer of the network consists of computing nodes. All the nodes in a layer are fully connected to the nodes in adjacent layers but there is no connection between the nodes within the same layer and no bridging layer connections. The topology of the MLP is shown in Fig.1.

Inputs to the network are passed to each node in the first layer. The outputs of the first layer nodes then become inputs to the second layer and so on. The last layer therefore acts as the network output layer and all the other layers below it are called hidden layers. The architecture of a MLP can conveniently be summarized as $n_0 - n_1 - \cdots - n_l$, where n_0 is the network input dimension and n_i, $1 \le i \le l$, are the numbers of nodes in the respective layers. The input-output relationship of a generic node, the ith node in the kth layer, is depicted in Fig.2. From Fig.2,

$$z_i^{(k)} = \sum_{j=1}^{n_{k-1}} \eta_{ij}^{(k)} x_j^{(k-1)} + \mu_i^{(k)}, \tag{9}$$

$$x_i^{(k)} = a(z_i^{(k)}), \tag{10}$$

where $\eta_{ij}^{(k)}$ and $\mu_i^{(k)}$ are the node connection weights and the threshold respectively; and $a()$ is called the node activation function.

In applications to modelling non-linear dynamic system (1), the network input is given by $\mathbf{x}(t)$ defined in (5) with $n_0 = n_I$, and the number of output nodes is $n_l = m$. The activation function of hidden nodes is typically chosen as

$$a(z) = \frac{1}{1+\exp(-z)}, \tag{11}$$

or

$$a(z) = \tanh(z) = \frac{1-\exp(-2z)}{1+\exp(-2z)}. \tag{12}$$

Other choices of $a()$ can also be employed. The output nodes usually do not contain a threshold parameter and the associated activation functions are linear, that is,

$$\hat{y}_i = x_i^{(l)} = \sum_{j=1}^{n_{l-1}} \eta_{ij}^{(l)} x_j^{(l-1)}, \quad 1 \le i \le m. \tag{13}$$

The overall input-output mapping of the network is $\hat{f} : R^{n_I} \to R^m$.

3.1. Approximation capabilities of MLP

The approximation capabilities of MLP have been investigated by many authors (e.g. Cybenko 1989, Funahashi 1989, Hecht-Nielsen 1990, Hornik 1991). Basically, any continuous function $f : D_f \subset R^{n_I} \to R^m$ can be uniformly approximated to within an arbitrary accuracy by an $\hat{f}$ on D_f, where D_f is a compact subset of R^{n_I}, provided that there are a sufficient number of hidden nodes in the network. This result is valid even for networks with only one hidden layer.

Typical assumptions on the activation function for hidden nodes are that $a()$ is continuous, bounded and nonconstant. These are very mild requirements, and the sigmoid function (11) is just one example of many possible choices of activation function. This theoretical result provides a sound foundation for modelling non-linear systems using MLP.

These mathematical results prove that the MLP is a general function approximator and guarantee that a one-hidden-layer network will always be sufficient to represent any arbitrary continuous function. But this does not say how many hidden nodes will be necessary to achieve the task. For some practical problems, networks with two or more hidden layers may be more efficient in terms of the total hidden nodes required. It is also worth emphasising that the proof of the approximation capabilities of MLP assumes that the weights have been correctly assigned. Whether or not these appropriate weights can be determined using any existing learning algorithm is an open question.

3.2. Prediction error learning algorithms

Estimation of the parameters in non-linear models is generally based on non-linear optimisation techniques (Goodwin and Payne 1977, Ljung and Söderström 1983, Billings and Chen 1989a, Chen and Billings 1989a). A class of learning algorithms, known as the prediction error algorithms, can be derived for MLP by adopting ideas from non-linear system identification (Chen *et al.* 1990a, 1990b). Both the batch and recursive versions of the prediction error algorithm are briefly summarized below.

Assume that all the weights and thresholds of the MLP have been arranged into an n_Θ-dimensional parameter vector

$$\Theta=[\theta_1\cdots\theta_{n_\Theta}]^T, \tag{14}$$

where

$$n_\Theta=\sum_{i=0}^{l-2}(n_i+1)n_{i+1}+n_{l-1}n_l;\ \ n_0=n_I,\ n_l=m. \tag{15}$$

The input-output equation of the n_I-input m-output MLP is defined by

$$\hat{y}(t,\Theta)=\hat{f}(\mathbf{x}(t);\Theta). \tag{16}$$

In the terminology of system identification, the discrepancy between the system output $y(t)$ and $\hat{y}(t,\Theta)$

$$\epsilon(t,\Theta)=y(t)-\hat{y}(t,\Theta) \tag{17}$$

is called the prediction error. The gradient of $\hat{y}(t,\Theta)$ with respect to Θ is an $n_\Theta \times m$ matrix

$$\Psi(t,\Theta)=G(\mathbf{x}(t);\Theta)=\left[\frac{d\hat{y}(t,\Theta)}{d\Theta}\right]^T, \tag{18}$$

and this plays a vital role during learning. The combination of (16) and (18)

$$\begin{bmatrix}\hat{y}(t,\Theta)\\ \Psi(t,\Theta)\end{bmatrix}=\begin{bmatrix}\hat{f}(\mathbf{x}(t);\Theta)\\ G(\mathbf{x}(t);\Theta)\end{bmatrix} \tag{19}$$

will be referred to as the extended network model. The gradient of $\hat{y}(t,\Theta)$ with respect to θ_i is an $1\times m$ row vector and is denoted as $\psi_i(t,\Theta)$, where $1\le i\le n_\Theta$. $\Psi(t,\Theta)$ can therefore be written as

$$\Psi(t,\Theta)=\begin{bmatrix}\psi_1(t,\Theta)\\ \cdot\\ \cdot\\ \cdot\\ \psi_{n_\theta}(t,\Theta)\end{bmatrix}. \tag{20}$$

Assume that the network has q nodes and the weights and threshold of the ith node are arranged in an n_{Θ_i}-dimensional vector Θ_i, $1\le i\le q$. Θ and $\Psi(t,\Theta)$ can therefore be represented as

$$\Theta=\begin{bmatrix}\Theta_1\\ \cdot\\ \cdot\\ \cdot\\ \Theta_q\end{bmatrix}, \quad \Psi(t,\Theta)=\begin{bmatrix}\Psi_1(t,\Theta)\\ \cdot\\ \cdot\\ \cdot\\ \Psi_q(t,\Theta)\end{bmatrix}, \tag{21}$$

where $\Psi_i(t,\Theta)$, an $n_{\Theta_i}\times m$ matrix, is the gradient of $\hat{y}(t,\Theta)$ with respect to Θ_i.

3.2.1. Batch identification algorithms

The batch prediction error learning algorithms for training the networks are a class of optimisation algorithms (Chen *et al.* 1990a, 1990b). The optimisation criterion or loss function is typically chosen as

$$J_N(\Theta)=\frac{1}{2N}\sum_{t=1}^{N}\epsilon^T(t,\Theta)\epsilon(t,\Theta), \tag{22}$$

where N is the length of the training data. Learning is achieved by minimizing

(22) subject to Θ and is usually done iteratively according to

$$\Theta(k)=\Theta(k-1)+\alpha\Xi(\Theta(k-1)), \tag{23}$$

where k denotes the iteration step in the minimisation procedure,

$$\Xi(\Theta(k-1))=[\xi_1(\Theta(k-1))\cdots\xi_{n_\Theta}(\Theta(k-1))]^T \tag{24}$$

is a search direction based on information about $J_N(\Theta)$ acquired at a previous iteration, and α is a positive constant which is appropriately chosen to guarantee convergence of the iterative procedure. The most commonly used search direction is the modified negative gradient direction defined by

$$\Xi(\Theta)=M(\Theta)(-\nabla J_N(\Theta)) \tag{25}$$

where $M(\Theta)$ is a positive definite $n_\Theta \times n_\Theta$ matrix and

$$\nabla J_N(\Theta)=-\frac{1}{N}\sum_{t=1}^{N}\Psi(t,\Theta)\epsilon(t,\Theta) \tag{26}$$

is the gradient of $J_N(\Theta)$ with respect to Θ.

The Gauss-Newton search direction is obtained by choosing $M(\Theta)$ as the inverse of the approximate Hessian of (22), that is,

$$M^{-1}(\Theta)=H(\Theta)=\frac{1}{N}\sum_{t=1}^{N}\Psi(t,\Theta)\Psi^T(t,\Theta). \tag{27}$$

This gives rise to the full prediction error (also called Gauss-Newton) algorithm.

The Hessian matrix $H(\Theta)$ can be partitioned into $q\times q$ sub-matrices

$$H_{ij}(\Theta)=\frac{1}{N}\sum_{t=1}^{N}\Psi_i(t,\Theta)\Psi_j^T(t,\Theta),\quad 1\le i,j\le q. \tag{28}$$

If $M^{-1}(\Theta)$ is chosen as the following near-diagonal matrix

$$M^{-1}(\Theta)=\begin{bmatrix} H_{11}(\Theta) & & 0 \\ & \ddots & \\ 0 & & H_{qq}(\Theta) \end{bmatrix}, \tag{29}$$

a parallel prediction error algorithm is obtained which consists of q sub-algorithms

$$\Theta_i(k)=\Theta_i(k-1)+\alpha\Xi_i(\Theta(k-1)),\quad 1\le i\le q, \tag{30}$$

with

$$\Xi_i(\Theta)=H_{ii}^{-1}(\Theta)\frac{1}{N}\sum_{t=1}^{N}\Psi_i(t,\Theta)\epsilon(t,\Theta). \tag{31}$$

Each of these sub-algorithms corresponds to a node in the network.

The well-known steepest-descent algorithm can be derived by choosing the search direction as the negative gradient of (22) and the algorithm can be decomposed into n_Θ scalar equations

$$\theta_i(k)=\theta_i(k-1)+\alpha\xi_i(\Theta(k-1)), \quad 1\le i\le n_\Theta, \tag{32}$$

with

$$\xi_i(\Theta)=\frac{1}{N}\sum_{t=1}^{N}\psi_i(t,\Theta)\epsilon(t,\Theta). \tag{33}$$

This algorithm is obtained by setting $M(\Theta)$ to be the identity matrix I.

3.2.2. Recursive identification algorithms

The batch prediction error algorithms have recursive counterparts (Chen and Billings 1989a, Chen *et al.* 1990b). Define a time-varying version of the extended network model (19)

$$\begin{bmatrix}\hat{y}(t)\\ \Psi(t)\end{bmatrix}=\begin{bmatrix}\hat{f}(\mathbf{x}(t);\hat{\Theta}(t-1))\\ G(\mathbf{x}(t);\hat{\Theta}(t-1))\end{bmatrix}, \tag{34}$$

where $\hat{\Theta}(t)$ denotes the estimate of Θ at t, and an approximate prediction error

$$\epsilon(t)=y(t)-\hat{y}(t). \tag{35}$$

The full recursive prediction error algorithm is given as follows

$$\Delta(t)=\alpha_m\Delta(t-1)+\alpha_g\Psi(t)\epsilon(t), \tag{36}$$

$$P(t)=[P(t-1)-P(t-1)\Psi(t)(\lambda I+\Psi^T(t)P(t-1)\Psi(t))^{-1}\Psi^T(t)P(t-1)]/\lambda, \tag{37}$$

$$\hat{\Theta}(t)=\hat{\Theta}(t-1)+P(t)\Delta(t), \tag{38}$$

where α_g and α_m are the adaptive gain and momentum respectively, and λ is the forgetting factor.

Similarly, the recursive version of the parallel prediction error algorithm can be written as

$$\left.\begin{aligned}\Delta_i(t)&=\alpha_m\Delta_i(t-1)+\alpha_g\Psi_i(t)\epsilon(t)\\ \hat{\Theta}_i(t)&=\hat{\Theta}_i(t-1)+P_i(t)\Delta_i(t)\end{aligned}\right\}\quad 1\le i\le q, \tag{39}$$

and the formula for updating each $P_i(t)$ is identical to that used for $P(t)$.

Finally, the recursive implementation of the steepest descent algorithm gives rise to the following smoothed stochastic gradient algorithm

$$\left.\begin{aligned}\sigma_i(t)&=\alpha_m\sigma_i(t-1)+\alpha_g\psi_i(t)\epsilon(t)\\ \hat{\theta}_i(t)&=\hat{\theta}_i(t-1)+\sigma_i(t)\end{aligned}\right\}\quad 1\le i\le n_\Theta. \tag{40}$$

The quantity $\psi_i(t)\epsilon(t)$ corresponds to the negative gradient of $\epsilon^T(t)\epsilon(t)/2$ with respect to θ_i and is noisy or stochastic in nature. $\sigma_i(t)$ is therefore a smoothed stochastic gradient.

The back propagation algorithm for training MLP is in fact identical to the smoothed stochastic gradient algorithm in (40). Using the generic equations (9) and (10), it is a relatively simple task to formulate (40) as

$$\left.\begin{aligned} \eta_{ij}^{(k)}(t) &= \eta_{ij}^{(k)}(t-1) + \Delta\eta_{ij}^{(k)}(t) \\ \mu_i^{(k)}(t) &= \mu_i^{(k)}(t-1) + \Delta\mu_i^{(k)}(t) \end{aligned}\right\} \tag{41}$$

with

$$\left.\begin{aligned} \Delta\eta_{ij}^{(k)}(t) &= \alpha_m \Delta\eta_{ij}^{(k)}(t-1) + \alpha_g \delta_i^{(k)}(t) x_j^{(k-1)}(t) \\ \Delta\mu_i^{(k)}(t) &= \alpha_m \Delta\mu_i^{(k)}(t-1) + \alpha_g \delta_i^{(k)}(t) \end{aligned}\right\} \tag{42}$$

and

$$\delta_i^{(l)}(t) = a'(z_i^{(l)}(t))(y_i(t) - \hat{y}_i(t)), \tag{43}$$

$$\delta_i^{(k)}(t) = a'(z_i^{(k)}(t)) \sum_s \delta_s^{(k+1)}(t) \eta_{si}^{(k+1)}(t-1), \quad k = l-1, \ldots, 2, 1, \tag{44}$$

where $a'(z)$ is the derivative of $a(z)$. This is the usual form for the back propagation algorithm (Rumelhart *et al.* 1986).

Many numerically robust versions of recursive prediction error algorithms can be derived using modifications to the basic form (37) of computing $P(t)$ or $P_i(t)$ (e.g. Sripada and Fisher 1987, Salgado *et al.* 1988). For example a constant trace version is given as

$$\left.\begin{aligned} \bar{P}(t) &= P(t-1) - P(t-1)\Psi(t)(\lambda I + \Psi^T(t)P(t-1)\Psi(t))^{-1}\Psi^T(t)P(t-1) \\ P(t) &= \frac{K_0}{trace\,[\bar{P}(t)]}\bar{P}(t), \quad K_0 > 0 \end{aligned}\right\}. \tag{45}$$

3.3. Convergence properties

The batch prediction error algorithms are gradient descent optimisation algorithms and are guaranteed to converge to a local minimum that contains the initial parameter vector in its attractive basin. It can be shown that the recursive prediction error algorithms have the same convergence properties as their batch counterparts (Ljung and Söderström 1983, Chen *et al.* 1990a). One of these properties is that $\hat{\Theta}(t)$ converges with probability one to a local minimum of

$$J_{\infty}(\Theta)=\lim_{N\to\infty}\frac{1}{2N}\sum_{t=1}^{N}E\,[\epsilon^{T}(t,\Theta)\epsilon(t,\Theta)], \tag{46}$$

where $E\,[]$ is the expectation operator.

The convergence of the back propagation algorithm can be very slow. The advantages of this algorithm are the computational simplicity and parallel structure. Because learning is distributed to each weight, the algorithm is coherent with the massively parallel nature of the network. The full recursive prediction error algorithm on the other hand has much better convergence performance at the expense of increased complexity. It also requires a centralized learning mechanism and thus violates the principle of distributed computing. The parallel recursive prediction error algorithm represents a good compromise between the two algorithms. It is computationally much simpler than the the full recursive prediction error algorithm and, like the back propagation algorithm, learning is distributed to each individual node. Although the parallel recursive prediction error algorithm is more complex than the back propagation algorithm, the former is generally much more efficient in terms of convergence than the latter. This can readily be demonstrated using a real-system identification application.

The data was generated from a liquid level system. The system consists of a DC water pump feeding a conical flask which in turn feeds a square tank. The system input is the voltage to the pump motor and the system output is the water level in the conical flask. This is a single-input single-output ($r=m=1$) process. A one-hidden-layer network was employed to model this real process. The network structure was specified by $n_I=n_y+n_u=3+5$, $n_1=5$ and $n_2=1$, giving rise to $n_{\Theta}=50$. The hidden node activation function was chosen as (11). Initial weights and thresholds were set randomly between -0.3 to 0.3. After several trial runs, it was found that $\alpha_g=0.01$ and $\alpha_m=0.8$ were appropriate for the back propagation algorithm. For the parallel recursive prediction error algorithm, the constant-trace technique (45) was used to update $P_i(t)$, and the parameters of the algorithm were given by $\alpha_g=1.0$, $\alpha_m=0.0$, $\lambda=0.99$, $K_0=60.0$ and $P_i(0)=1000.0I$. The evolutions of the mean square error in dB's obtained by the back propagation algorithm (bpa) and the parallel recursive prediction error algorithm (prpe) are depicted in Fig.3.

This clearly demonstrates the improved convergence performance of the parallel recursive prediction error algorithm compared to the back propagation algorithm.

Because the MLP is highly non-linear in the parameters, the mean square error error surface (46) is very complicated. It typically has a large number of global minima which may lie at infinity for some problems. Thus the error surfaces are often highly degenerate and have numerous troughs. The error surfaces also generally contain many local minima and may have flat areas where the gradients almost vanish. When the weights fall into these flat regions, learning becomes extremely slow and it can take a long time for the algorithm to escape. The following simple example provides some illustration of the complexity of the error surfaces. The example is a simple classifier, the input of which is a scalar x uniformly distributed in the interval $[-1, 1]$. The desired output is defined as

$$y = f(x) = \begin{cases} 1, & x \in [0, 0.5], \\ -1, & \text{otherwise}. \end{cases}$$

The classifier

$$\hat{y} = \hat{f}(x) = \tanh(\eta_1 x + \eta_2 x^2)$$

consists of a single node with two inputs x and x^2, and no threshold. The error surface

$$E[\epsilon^2(\eta_1, \eta_2)] = \frac{1}{2} \int_{-1}^{1} (\tanh(\eta_1 x + \eta_2 x^2) - f(x))^2 dx$$

is the three-dimensional surface depicted in Fig.4. Notice that there is a trough along the direction of $\eta_1 > 0$ and $\eta_2 = -2\eta_1$, and the global minimum is achieved when $\eta_1 \rightarrow +\infty$ along this direction. There clearly exists a flat region where the gradients are almost zero. Despite the possible pitfalls of gradient learning methods, the MLP proves to be a popular network architecture and has been widely used in many different disiplines as a universal mapping approximator.

4. The radial basis function network

An alternation network architecture to the MLP is the RBF network (Broomhead and Lowe 1988, Moody and Darken 1989, Chen *et al.* 1990c, 1991a, 1991c). The RBF network is a two-layer processing structure. The hidden layer consists of an array of nodes. Each node contains a parameter vector called a centre. The node calculates the Euclidean distance between the centre and the network input vector, and passes the result through a non-linear function. The output layer is essentially a set of linear combiners. The architecture of the RBF network is shown in Fig.5. The overall input-output response of the RBF network is a mapping $\hat{f}: R^{n_I} \to R^m$, that is,

$$\hat{f}_i(\mathbf{x}) = \sum_{j=1}^{n_h} \eta_{ji} \phi_j = \sum_{j=1}^{n_h} \eta_{ji} \phi(\|\mathbf{x} - \mathbf{c}_j\|, \rho_j), \quad 1 \le i \le m, \tag{47}$$

where η_{ji} are the weights of the linear combiners, $\|\cdot\|$ denotes the Euclidean norm, ρ_j are some positive scalars called widths, $\phi(\cdot,\rho)$ is a function from $R^+ \to R$, $\mathbf{c}_j$ are known as the RBF centres, and n_h is the number of nodes in the hidden layer. Some typical choices of $\phi()$ are the thin-plate-spline function

$$\phi(z,1) = z^2 \log(z), \tag{48}$$

the multiquadric function

$$\phi(z,\rho) = \sqrt{z^2 + \rho^2}, \tag{49}$$

the inverse multiquadric function

$$\phi(z,\rho) = \frac{1}{\sqrt{z^2 + \rho^2}}, \tag{50}$$

and the Gaussian function

$$\phi(z,\rho) = \exp(-z^2/\rho^2). \tag{51}$$

The topology of the RBF network is obviously similar to that of the two-layer perceptron, and the differences lie in the characteristics of the hidden nodes.

4.1. Approximation capabilities of RBF

Because of the similarities between the RBF network and the two-layer perceptron, it is expected that the former has the same approximation ability as the latter. This has in fact been proved (Cybenko 1989, Park and Sandberg 1991). Thus, under very mild assumptions on the non-linearity $\phi()$, any continuous function $f: D_f \subset R^{n_I} \to R^m$ can be uniformly approximated to within an arbitrary accuracy by a RBF network $\hat{f}$ on D_f, provided that there are a sufficient number of hidden nodes.

In the proofs of the universal approximation ability of the RBF network, it is often assumed that $\phi()$ is continuous and bounded (Park and Sandberg 1991). This is of course a very mild assumption, and the non-linear functions (50) and (51) satisfy this requirement. The non-linearities (48) and (49) do not belong to this class of functions because $\phi(z,\rho) \to \infty$ as $z \to \infty$. According to Powell (1987), however, this kind of RBF network also has good approximation capabilities. In fact, the success of approximation is easier to achieve if $\phi(z,\rho) \to \infty$ as $z \to \infty$ than if $\phi(z,\rho) \to 0$ as $z \to \infty$. These theoretical results suggest that the selection of the non-linearity $\phi()$ is not crucial for performance. Although the width in each node can have a different value, the same width for every node is sufficient for universal approximation (Park and Sandberg 1991). This means that all the widths ρ_j can be fixed to a value ρ to provide a simpler training strategy. In practice, the value of ρ may have some effects on the numerical properties of the learning algorithms but not on the general approximation ability of the RBF network. Some choices of the non-linearity such as the non-linear function (48) do not require the specification of a width parameter. Finally, as in the case of the MLP, although the theory guarantees the ability of a RBF network with correct weights η_{ji} and centres $\mathbf{c}_j$ to accurately represent an arbitrary continuous function, it does not provide information on whether or not these parameters can actually be learned using any existing learning law.

4.2. The orthogonal least squares algorithm

In the case of batch identification, a block of data is available. The parameters of a RBF network can then be estimated using the prediction error estimation method. The prediction error method however results in a non-linear learning rule just as in the case of the MLP and advantages can be gained by exploiting the structure of the RBF network and developing a linear learning rule. A common strategy is to select some data points as the RBF centres. Once this has been done, the weights can be learned using the least squares (LS) method. This learning strategy has its roots in the strict interpolation of data in multidimensional space (Broomhead and Lowe 1988). The problem can also be formulated as one of selecting subset models (Chen *et al.* 1990c), and the existing subset selection methods developed for non-linear NARMAX models (Billings *et al.* 1989, Chen *et al.* 1989, Leontaritis and Billings 1987) can readily be applied. This is the approach adopted in the present study and an orthogonal least squares (OLS) learning algorithm for RBF networks (Chen *et al.* 1990c, 1991a, 1991c) is described. The main advantage of this approach is that an appropriate set of RBF centres can be identified from the data set and estimates of the corresponding weights can be simultaneously determined in a very efficient manner.

Assume that a non-linearity $\phi()$ is chosen, a fixed width ρ is given, and the centres are formed from the data points $\mathbf{x}(t)$, $1 \le t \le N$. The RBF network is a special case of the following linear regression model:

$$y_i(t) = \sum_{j=1}^{M} \phi_j(t)\theta_{ji} + e_i(t), \ 1 \le i \le m, \tag{52}$$

where $\phi_j(t)$ are known as the regressors which are some fixed functions of the input $\mathbf{x}(t)$, θ_{ji} are the parameters to be estimated, and $y_i(t)$ and $e_i(t)$ are the ith desired output and the ith error signal respectively. This model can now be interpreted as a subset NARMAX model (Billings and Leontaritis 1981, Chen and Billings 1989b). Define

$$\mathbf{y}_i = [y_i(1) \cdots y_i(N)]^T, \ 1 \le i \le m, \tag{53}$$
$$\mathbf{e}_i = [e_i(1) \cdots e_i(N)]^T, \ 1 \le i \le m, \tag{54}$$
$$\Phi_j = [\phi_j(1) \cdots \phi_j(N)]^T, \ 1 \le j \le M. \tag{55}$$

Then, for $1 \le t \le N$, (52) can be collectively arranged as

$$[\mathbf{y}_1 \cdots \mathbf{y}_m] = [\Phi_1 \cdots \Phi_M] \begin{bmatrix} \theta_{11} & \cdots & \theta_{1m} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \theta_{M1} & \cdots & \theta_{Mm} \end{bmatrix} + [\mathbf{e}_1 \cdots \mathbf{e}_m] \tag{56}$$

or more concisely in the following matrix form

$$\mathbf{Y} = \Phi\Theta + \mathbf{E}. \tag{57}$$

The well-known LS method can be used to obtain the parameter estimate $\hat{\Theta}$. The LS solution has a clear geometric interpretation. The regressor vectors Φ_j form a set of basis vectors, and the solution $\hat{\Theta}$ satisfies the condition that $\Phi\hat{\Theta}$ is the projection of $\mathbf{Y}$ onto the space spanned by these basis vectors. In other words, the trace of the square of the projection $\Phi\hat{\Theta}$ is part of the desired output energy that can be counted by the regressors. Because different regressors are generally correlated, it is not clear how each individual regressor contributes to this desired output energy.

The OLS method involves the transformation of the set of Φ_j into a set of orthogonal basis vectors, and thus makes it possible to calculate the individual contribution to the desired output energy from each basis vector. The regression matrix Φ can be decomposed into

$$\Phi = \mathbf{W}\,\mathbf{B}, \tag{58}$$

where

$$\mathbf{W} = [\mathbf{w}_1 \cdots \mathbf{w}_M] \tag{59}$$

with orthogonal columns $\mathbf{w}_j$, $1 \le j \le M$, that satisfy

$$\mathbf{w}_i^T \mathbf{w}_j = 0, \ \text{if} \ i \ne j, \tag{60}$$

and

$$\mathbf{B} = \begin{bmatrix} 1 & \beta_{12} & \cdot & \cdot & \cdot & \beta_{1M} \\ 0 & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot & \cdot & \cdot \\ \cdot & & & \cdot & 1 & \beta_{M-1M} \\ 0 & \cdot & & \cdot & 0 & 1 \end{bmatrix}. \tag{61}$$

The space spanned by the set of $\mathbf{w}_j$ is the same space spanned by the set of Φ_j, and (57) can be rewritten as

$$\mathbf{Y} = \mathbf{W}\,\Gamma + \mathbf{E}. \tag{62}$$

The OLS solution

$$\hat{\Gamma} = \begin{bmatrix} \hat{\gamma}_{11} & \cdots & \hat{\gamma}_{1m} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \hat{\gamma}_{M1} & \cdots & \hat{\gamma}_{Mm} \end{bmatrix} \tag{63}$$

is given by

$$\hat{\Gamma} = (\mathbf{W}^T \mathbf{W})^{-1} \mathbf{W}^T \mathbf{Y}, \tag{64}$$

or

$$\hat{\gamma}_{ji} = \frac{\mathbf{w}_j^T \mathbf{y}_i}{(\mathbf{w}_j^T \mathbf{w}_j)}, \quad 1 \le j \le M, \ 1 \le i \le m. \tag{65}$$

The OLS solution $\hat{\Gamma}$ and the ordinary LS solution $\hat{\Theta}$ satisfy the triangular system

$$\mathbf{B}\,\hat{\Theta} = \hat{\Gamma}. \tag{66}$$

The classical and modified Gram-Schmidt methods (Björck 1967) can be employed to derive B and $\hat{\Gamma}$ and thus to solve for $\hat{\Theta}$ from (66). The Householder transformation method (Golub 1965) can alternatively be used to obtain a similar orthogonal decomposition. As an illustration, the well-known classical Gram-

Schmidt method computes one column of $\mathbf{B}$ at a time and orthogonalizes Φ as follows: at the kth stage make the kth column orthogonal to each of the $k-1$ previously orthogonalized columns and repeat the operation for $k=2,\ldots,M$. The computational procedure can be represented as:

$$\mathbf{w}_1=\Phi_1, \tag{67}$$

$$\left.\begin{aligned} &\beta_{ik}=\mathbf{w}_i^T\Phi_k/(\mathbf{w}_i^T\mathbf{w}_i),\ 1\le i<k \\ &\mathbf{w}_k=\Phi_k-\sum_{i=1}^{k-1}\beta_{ik}\mathbf{w}_i \end{aligned}\right\}\ k=2,\ldots,M. \tag{68}$$

The OLS method has superior numerical properties compared with the ordinary LS method. It can also be utilized for subset selection (Chen *et al.* 1989, Billings *et al.* 1989) which is very important in non-linear system identification.

In the case of RBF networks, the number of data points $\mathbf{x}(t)$ is often very large and centres are to be chosen as a subset of the data set. In general the number of all the candidate regressors, M, can also be very large but an adequate model may only require $M_s(<<M)$ significant regressors. These significant regressors can be selected using the OLS algorithm operating in a forward regression manner (Chen *et al.* 1989). Because the error matrix $\mathbf{E}$ is orthogonal to $\mathbf{W}$, after some simple calculations, it can be shown that the trace of the covariance of $y(t)$ is given by

$$\mathrm{trace}\left(\frac{1}{N}\mathbf{Y}^T\mathbf{Y}\right)=\frac{1}{N}\sum_{j=1}^{M}\left(\sum_{i=1}^{m}\gamma_{ji}^2\right)\mathbf{w}_j^T\mathbf{w}_j+\mathrm{trace}\left(\frac{1}{N}\mathbf{E}^T\mathbf{E}\right). \tag{69}$$

The first term in the right-hand side of (69) is the part of the trace of the desired output covariance which can be explained by the regressors and the second term is the unexplained trace of the desired output covariance. Thus

$$\frac{1}{N}\left(\sum_{i=1}^{m}\gamma_{ji}^2\right)\mathbf{w}_j^T\mathbf{w}_j \tag{70}$$

is the increment to the explained trace introduced by $\mathbf{w}_j$, and the error reduction ratio due to $\mathbf{w}_j$ can be defined as (Chen *et al.* 1991c)

$$[err]_j=\frac{\left(\sum_{i=1}^{m}\gamma_{ji}^2\right)\mathbf{w}_j^T\mathbf{w}_j}{\mathrm{trace}\,(\mathbf{Y}^T\mathbf{Y})},\quad 1\le j\le M. \tag{71}$$

Based on this ratio, a simple and effective procedure can be derived for selecting a subset of significant regressors in a forward-regression manner. The classical

Gram-Schmidt scheme will again be used as an example, and with the regressor selection procedure attached the algorithm is summarized as follows:

□ At the 1*st* step, for $1 \le i \le M$, compute

$$\left.\begin{array}{l} \mathbf{w}_1^{(i)} = \Phi_i \\ \gamma_{1q}^{(i)} = (\mathbf{w}_1^{(i)})^T \mathbf{y}_q / ((\mathbf{w}_1^{(i)})^T \mathbf{w}_1^{(i)}), \quad 1 \le q \le m \\ [err]_1^{(i)} = (\sum_{q=1}^{m} (\gamma_{1q}^{(i)})^2)(\mathbf{w}_1^{(i)})^T \mathbf{w}_1^{(i)} / \mathrm{trace}(\mathbf{Y}^T \mathbf{Y}) \end{array}\right\}.$$

Find

$$[err]_1 = [err]_1^{(i_1)} = \max\{[err]_1^{(i)}, 1 \le i \le M\}$$

and select

$$\mathbf{w}_1 = \mathbf{w}_1^{(i_1)} = \Phi_{i_1}.$$

□ At the kth step where $k \ge 2$, for $1 \le i \le M$, $i \ne i_1, \ldots, i \ne i_{k-1}$, compute

$$\left.\begin{array}{l} \beta_{jk}^{(i)} = \mathbf{w}_j^T \Phi_i / (\mathbf{w}_j^T \mathbf{w}_j), \quad 1 \le j < k \\ \mathbf{w}_k^{(i)} = \Phi_i - \sum_{j=1}^{k-1} \beta_{jk}^{(i)} \mathbf{w}_j \\ \gamma_{kq}^{(i)} = (\mathbf{w}_k^{(i)})^T \mathbf{y}_q / ((\mathbf{w}_k^{(i)})^T \mathbf{w}_k^{(i)}), \quad 1 \le q \le m \\ [err]_k^{(i)} = (\sum_{q=1}^{m} (\gamma_{kq}^{(i)})^2)(\mathbf{w}_k^{(i)})^T \mathbf{w}_k^{(i)} / \mathrm{trace}(\mathbf{Y}^T \mathbf{Y}) \end{array}\right\}.$$

Find

$$[err]_k = [err]_k^{(i_k)} = \max\{[err]_k^{(i)}, 1 \le i \le M, i \ne i_1, \ldots, i \ne i_{k-1}\}$$

and select

$$\mathbf{w}_k = \mathbf{w}_k^{(i_k)} = \Phi_{i_k} - \sum_{j=1}^{k-1} \beta_{jk} \mathbf{w}_j,$$

where $\beta_{jk} = \beta_{jk}^{(i_k)}$, $1 \le j < k$.

□ The procedure is terminated at the M_sth step when

$$1 - \sum_{j=1}^{M_s} [err]_j < \zeta, \tag{72}$$

where $0 < \zeta < 1$ is a chosen tolerance. This gives rise to a subset model containing M_s significant regressors.

In the above procedure each selected centre (regressor) maximizes the increment to the explained trace of the desired output covariance. The selection of centres is therefore directly linked to the reduction of the error covariance trace. This is clearly superior to a random selection of centres proposed originally by Broomhead and Lowe (1988). Similar selection procedures can be derived using the modified Gram-Schmidt method and Householder transformation method (Chen *et al.* 1989, Billings and Chen 1989b).

The tolerance ζ is an important instrument in balancing the accuracy and the complexity of the final network. It is apparent from (69) that ζ should ideally be slightly larger than the ratio trace$(\mathbf{E}^T\mathbf{E})$/trace$(\mathbf{Y}^T\mathbf{Y})$. The desired value for ζ can actually be learnt during the selection procedure (Billings and Chen 1989b). Trace$(\mathbf{Y}^T\mathbf{Y})$ is known from the measured or desired outputs and, during the selection procedure, an estimate of trace$(\mathbf{E}^T\mathbf{E})$ can be computed. After a few trials, an appropriate ζ can usually be found. The terminating criterion (72) emphasises only the performance of the network. Because a more accurate performance is often achieved at the expense of using a larger network, a trade-off between the performance and complexity of the network is often desired. An alternative terminating criterion can be employed based on an Akaike information criterion and this is discussed in (Chen *et al.* 1989, Billings and Chen 1989b).

A simulated non-linear time series process will be used to illustrate the OLS algorithm. A total of 1000 observation samples were generated using the model

$$\begin{aligned} y(t) = {} & (0.8-0.5\exp(-y^2(t-1)))y(t-1)-(0.3+0.9\exp(-y^2(t-1)))y(t-2) \\ & +0.1\sin(3.1415926y(t-1))+e(t), \end{aligned} \tag{73}$$

where the noise $e(t)$ was a Gaussian white sequence with mean zero and variance 0.01. The inputs to the RBF network were given as $\mathbf{x}(t)=[y(t-1)\ y(t-2)]^T$, and there were about 1000 candidate centres when the centres were chosen from the data set. The non-linearity $\phi()$ was selected as (48). The OLS algorithm identified a RBF model with 30 centres. The distribution of the noisy observations and the selected centres are depicted in Fig.6. How accurate the identified RBF network represents the system can best be illustrated by examining the autonomous system response and the iterative network response generated from

$$\hat{y}(t)=\hat{f}(\hat{\mathbf{x}}(t)),\ \ \hat{\mathbf{x}}(t)=[\hat{y}(t-1)\ \hat{y}(t-2)]^T. \tag{74}$$

It can easily be verified that without the noise $e(t)$ the simulated system (73) generates a stable limit cycle as shown in Fig.7. The iterative network response produces a similar limit cycle as can be seen from Fig.8. Waveforms of the autonomous system output and the iterative network output are illustrated in Fig.9. It is clear that, even though the RBF network was identified using the noisy system observations, the iterative network response closely matches the response of the autonomous system. This demonstrates that the identified RBF model does capture the underlying dynamics of the simulated non-linear process.

4.3. The hybrid clustering and least squares algorithm

In general the network input data can only exist in some regions of the input space R^{n_I}. It is reasonable to allocate the RBF centres in these regions and to reflect the data patterns by the positions of the centres. Moody and Darken (1989) suggested using a κ-means clustering technique to adjust the centres. The κ-means clustering algorithm partitions the input data set into κ clusters and yields κ cluster centres by minimizing the total squared error incurred in representing the data set by the κ cluster centres (Duda and Hart 1973). The κ-means clustering is an unsupervised procedure using only the network input data and it closely resembles the Kohonen self-organising algorithm (Kohonen 1987). Because the response of the RBF network is linear with respect to the network weights, it is natural to consider the LS method for adjusting the weights. The LS algorithm is a supervised learning procedure requiring the desired output response. Thus learning can be achieved in a hybrid manner, combining an unsupervised clustering sub-algorithm for adjusting the RBF centres and a supervised LS sub-algorithm for updating the RBF weights. This hybrid algorithm can be implemented in batch form. However the advantage of this hybrid approach is that it can naturally be implemented in recursive form (Moody and Darken 1989, Chen *et al.* 1991b). The recursive version of the hybrid algorithm is described here.

Given initial centres $\mathbf{c}_j(0)$, $1 \le j \le n_h$, and an initial learning rate for the centres $\alpha_c(0)$, at each sample t the recursive clustering algorithm consists of the following computational steps:

□ Compute distances and find a minimum distance

$$d_j(t) = \|\mathbf{x}(t) - \mathbf{c}_j(t-1)\|, \quad 1 \le j \le n_h,$$
$$k = arg\,[\min\{d_j(t),\ 1 \le j \le n_h\}].$$

□ Update centres and re-compute kth distance

$$\mathbf{c}_j(t)=\mathbf{c}_j(t-1),\ 1\le j\le n_h \text{ and } j\ne k,$$
$$\mathbf{c}_k(t)=\mathbf{c}_k(t-1)+\alpha_c(t)(\mathbf{x}(t)-\mathbf{c}_k(t-1)),$$
$$d_k(t)=\|\mathbf{x}(t)-\mathbf{c}_k(t)\|.$$

The initial centres are often chosen randomly. The learning rate should be $\alpha_c(t)<1$ and should slowly decrease to zero. One rule is

$$\alpha_c(t)=\frac{\alpha_c(t-1)}{\sqrt{1+\text{int}[t/n_h]}}, \tag{75}$$

where int[] denotes the integer part of the argument. The problem of finding a minimum is equivalent to one of finding a maximum, and determining which centre is closest to the input data vector $\mathbf{x}(t)$ can be implemented in a sub-network called MAXNET (Lippmann 1987).

At the output layer, each node or linear combiner has its own recursive LS (RLS) estimator. Define the hidden layer output vector at t as

$$\hat{\phi}(t)=[\phi_1(t)\cdots\phi_{n_h}(t)]^T=[\phi(d_1(t),\rho)\cdots\phi(d_{n_h}(t),\rho)]^T, \tag{76}$$

and the weight vector of the ith output node after adaptation at t as

$$\Theta_i(t)=[\eta_{1i}(t)\cdots\eta_{n_h i}(t)]^T,\ 1\le i\le m. \tag{77}$$

The ith RLS estimator can be written as

$$\left.\begin{array}{l} \epsilon_i(t)=y_i(t)-\hat{\phi}^T(t)\Theta_i(t-1), \\ P(t)=\dfrac{1}{\lambda(t)}[P(t-1)-\dfrac{P(t-1)\hat{\phi}(t)\hat{\phi}^T(t)P(t-1)}{\lambda(t)+\hat{\phi}^T(t)P(t-1)\hat{\phi}(t)}], \\ \Theta_i(t)=\Theta_i(t-1)+P(t)\hat{\phi}(t)\epsilon_i(t), \end{array}\right\}\ 1\le i\le m. \tag{78}$$

The forgetting factor $\lambda(t)$ is usually computed according to the rule

$$\lambda(t)=\lambda_0\lambda(t-1)+1-\lambda_0. \tag{79}$$

Notice that a universal $P(t)$ is used in all the estimators. Numerically robust versions of RLS can be used instead of (78), for example, a Givens LS algorithm is described in (Chen *et al.* 1991b).

For fixed centres at the hidden layer, the mean square error surface is quadratic and the RLS sub-algorithm is therefore guaranteed to converge to the single global minimum rapidly. The κ-means cluster algorithm minimizes the total squared error incurred in representing the input data set by the κ cluster centres. In general only a local minimum of the total squared error is found in this way.

This is however sufficient because the value of the total squared error is unimportant to system modelling. The ultimate performance criterion is the (output) mean square error. The important thing is that, like the LS method, the clustering algorithm is based on a linear learning rule which has a fast convergence rate. The overall learning procedure is therefore guaranteed to converge rapidly. This is often an advantage of RBF networks compared to the MLP in recursive identification applications. The liquid level system described above will be used to illustrate this aspect.

The structure of the RBF network employed to identify the liquid level system was defined by $n_I = n_y + n_u = 3+5$ and $n_h = 40$. The non-linearity $\phi()$ was chosen as the thin-plate-spline function (48). The learning rate for the centres and the forgetting factor were computed using (75) and (79) respectively. The Givens version (Chen *et al.* 1991b) was used for weight updating. The parameters for the hybrid identification algorithm were chosen to be $P(0) = 1000.0I$, $\lambda_0 = 0.99$, $\lambda(0) = 0.95$ and $\alpha_c(0) = 0.6$. The initial centres were randomly set. The evolution of the mean square error obtained by the recursive hybrid clustering and LS algorithm (hyb.) is depicted in Fig.3, where it is seen that the reduction in the mean square error was faster compared with that of the two-layer perceptron trained by the recursive prediction error method.

The same version of the hybrid algorithm with $P(0) = 1000.0I$, $\lambda_0 = 0.99$, $\lambda(0) = 0.95$ and $\alpha_c(0) = 0.9$ was also used to identify the non-linear time series process (73). A RBF network with a structure defined by $n_I = n_y = 2$ and $n_h = 30$ was used. Random initial centres and the non-linearity (48) were chosen. A total of 1500 observation samples were used in the recursive identification, and the distribution of the noisy observations and the final network centres are plotted in Fig.10. The identified RBF network was employed to produce iteratively the network output in the way indicated by (74). The iterative network response generated the limit cycle shown in Fig.11. A comparison of the waveform of the iterative network response with that of the autonomous system response is given in Fig.12.

5. The functional-link network

In the MLP, inputs to a node are first linearly weighted before the sum is passed through some non-linear activation function. The network may have many layers but it is this non-linear activation function of nodes that ultimately gives the network its non-linear approximation ability. The same non-linearity however creates problems in learning the network weights. Non-linear learning rules must be used, the learning rate is often unacceptably slow and local minima may cause problems. One way of avoiding non-linear learning is to initially perform some non-linear functional transform or expansion of the network inputs and then to combine the resulting terms linearly. The structure would have a good non-linear approximation ability and yet learning of the weights is a linear problem. Pao (1989) referred to such a structure as the functional-link network (FLN).

The general architecture of the FLN is shown in Fig.13. The hidden layer enhances the network inputs by performing a certain functional expansion on the inputs. This maps the input space R^{n_I} onto a new space of increased dimension n, that is,

$$\mathbf{x} \in R^{n_I} \rightarrow [h_1(\mathbf{x}) \cdots h_n(\mathbf{x})]^T \in R^n, \tag{80}$$

where $n_I < n$. The set of $h_i(\mathbf{x})$ can be viewed as a new basis set. The output layer consists of m nodes, each node is in fact a linear combiner. The overall input-output mapping of the FLN $\hat{f}: R^{n_I} \rightarrow R^m$ is defined as

$$\hat{f}_i(\mathbf{x}) = \sum_{j=1}^{n} \eta_{ji} h_j(\mathbf{x}), \quad 1 \le i \le m. \tag{81}$$

Generally the value of a new basis function $h_j(\mathbf{x})$ depends only on the input $\mathbf{x}$ and a given functional expansion contains no other free parameters. In this sense the RBF network is different because centres and widths are free parameters. Only when the centres and widths are all fixed can a RBF network be regarded as a FLN.

The FLN is mostly relevant to a conventional view on non-linear system modelling. In fact the concept of FLN is exactly the same as the extended model set introduced by Billings and Chen (1989b). There are numerous ways of forming the set of model bases $\{h_i(\mathbf{x}), 1 \le i \le n\}$. The simplest and most well-known one is perhaps the polynomial expansion. In this representation, the set of model bases is the set of monomials of $\mathbf{x}$. A order-3 polynomial expansion, for example, is defined as

$$\{x_i,\ 1 \le i \le n_I\} \rightarrow \{x_i,\ x_i x_j,\ x_i x_j x_k,\ 1 \le i \le j \le k \le n_I\}. \tag{82}$$

Many other functions, such as the absolute value, exponential, logarithmic, hyperbolic, trigonometric, Coulomb friction and saturation functions, can certainly be employed to create an extended model set. For example, $\{h_i(\mathbf{x})\}$ may contain model bases like $|x_i - x_j|$, $\sin\pi x_i$, $\cos 2\pi x_i$, $\exp(-x_i^2)$, $\tanh(x_i)$, $\exp(-x_i^2)x_j$ and so on (Billings and Chen 1989b). The list of potential choices is endless. Such an extended model set can give a very rich description to non-linear systems and can provide an effective modelling framework. In practice, physical knowledge of the system to be identified can often be used to help in the selection of the model bases.

5.1. Approximation capabilities of FLN

The approximation ability of an FLN obviously relies on the model bases. Provided that the set of model bases is sufficiently rich or contains sufficient "higher-order" terms, it can be confidently said that any continuous function $f : D_f \subset R^{n_I} \rightarrow R^m$ can be uniformly approximated to within an arbitrary accuracy by a FLN $\hat{f}$ on D_f. Consider the simplest case where the extended model set contains only monomials of the network inputs. The use of a polynomial function to approximate a continuous function is an old but effective technique. Based on the Stone-Weierstrass theorem (e.g. Simmons 1963), it can be shown that any continuous function can be approximated to within an arbitrary accuracy by a polynomial function with a sufficient order. In any case an extended model set can contain components richer than monomials, and a more effective approximation can be achieved.

The FLN is truly linear in the parameters. Approximation theory not only says that a sufficient FLN with the correct weights can accurately implement an arbitrary continuous function but also guarantees that these parameters can always be learnt at least in the least squares sense. This second property is an advantage of using the extended model set concept or the FLN to model non-linear systems.

5.2. Identification algorithms

The FLN is a very flexible and rich structure. Too flexible some may argue. How to specify an extended model set can become a problem since potential choices are so numerous and a dimensional explosion can easily occur. Consider modelling a non-linear system with a scalar output and 20 inputs for instance.

Using the order-3 polynomial expansion alone produces an extended model set of dimension $n=1770$. Adding other numerous choices of model bases, it soon becomes impossible to construct such a large FLN. Subset model selection is therefore crucial for the FLN and this was solved by Billings and Chen (1989b) based on the extended model set idea. The OLS algorithm described above provides an efficient solution for subset model selection. A full extended model set is first specified. In order to cover the non-linear system to be identified well, the full model set may have a huge dimension, containing several thousands of terms for example. The OLS algorithm is used to identify a subset that achieves the desired accuracy. For many practical systems, the selected subset may only have 20 or so significant terms. Once the structure of the FLN or the set of model bases is given, the RLS algorithm provides an efficient means for real-time adaptation of the network weights. The convergence properties of the RLS algorithm are well established (Ljung and Söderström 1983).

To demonstrate the idea of FLN, a polynomial expansion was used to model the non-linear time series (73). The input dimension was set to $n_I=n_y=2$ and the polynomial order was chosen to be 7. This gave rise to a full extended model set of 35 monomials, a polynomial NARMAX model in fact (Leontaritis and Billings 1985). The OLS algorithm selected a subset of 9 monomials based on 1000 samples of noisy observation and the final model was

$$\begin{aligned}\hat{f}(y(t-1),y(t-2)) = &-1.17059y(t-2)+0.606861y(t-1)+0.679190y^2(t-1)y(t-2)\\ &-0.136235y^4(t-1)y(t-2)+0.165646y^3(t-1)y(t-2)-0.00711966y^6(t-2)\\ &-0.114719y^5(t-1)y(t-2)-0.0314354y(t-1)y(t-2)+0.0134682y^3(t-1).\end{aligned}$$

Notice that the highest-order of the selected model bases was 6. This subset model was then used to generate iteratively the model response

$$\hat{y}(t)=\hat{f}(\hat{y}(t-1),\hat{y}(t-2)),$$

which produced a limit cycle depicted in Fig.14. Excellent agreement was obtained bewteen the iterative model output and the autonomous system output as can be seen from Fig.15. Alternative extended model set terms could also be used to provide different functional model forms.

6. Conclusions

The artificial neural network approach has been shown to be a general scheme for non-linear dynamic system modelling and identification. Several feedforward network architectures have been considered as non-linear models and it has been demonstrated that they all have good capabilities for representing complex non-linear systems. Identification or learning algorithms for these network structures have been presented and the advantages and disadvantages of these network models have been discussed.

The multi-layer perceptron proves to be a universal model for non-linear systems and the prediction error method developed for system identification offers a class of batch and recursive algorithms for multi-layer perceptron models. The back propagation algorithm for example is just a special case of this class of learning algorithms. Unfortunately the error surface of a multi-layer perceptron model is often highly complex. This is a severe disadvantage and potential pitfalls may exist in the identification procedure.

The radial basis function network provides an alternative two-layer network structure. Each node in the hidden layer has a radially symmetric response around the node centre and linear learning laws can be derived. Two identification approaches, the orthogonal least squares method and the hybrid clustering and least squares method, have been described. The former is a batch learning algorithm capable of identifying adequate network structures and the latter can naturally be implemented as a recursive learning algorithm.

Many conventional non-linear models can be interpreted as some kind of networks that can be dubbed as a neural network. It has been shown that the functional-link network is just a class of non-linear models that was previously referred to as the extended model set representation. This class of networks is linear in the parameters and this offers a significant advantage in learning. For practical system identification, efficient subset selecting techniques are required to overcome excessive model dimension.

7. References

[1] Beale, R., and Jackson, T., 1990, *Neural Computing: An Introduction,* (Bristol: Adam Hilger).

[2] Billings, S.A., 1980, Identification of nonlinear systems - a survey. *Proc. IEE, Part D,* **127**, 272-285.

[3] Billings, S.A., and Chen, S., 1989a, Identification of non-linear rational systems using a prediction-error estimation algorithm. *Int. J. Systems Sci.*, **20**, 467-494.

[4] Billings, S.A., and Chen, S., 1989b, Extended model set, global data and threshold model identification of severely non-linear systems. *Int. J. Control,* **50**, 1897-1923.

[5] Billings, S.A., Chen, S., and Korenberg, M.J., 1989, Identification of MIMO non-linear systems using a forward-regression orthogonal estimator. *Int. J. Control,* **49**, 2157-2189.

[6] Billings, S.A., Gray, J.O., and Owens, D.H., (eds.), 1984, *Nonlinear System Design,* (London: Peter Peregrinus).

[7] Billings, S.A., and Leontaritis I.J., 1981, Identification of nonlinear systems using parameter estimation techniques. *Proc. IEE Conf. Control and Its Applications,* Warwick, U.K., pp.183-187.

[8] Björck, A., 1967, Solving linear least squares problems by Gram-Schmidt orthogonalization, *Nordisk Tidskr. Informations-Behandling,* **7**, 1-21.

[9] Broomhead, D.S., and Lowe, D., 1988, Multivariable functional interpolation and adaptive networks. *Complex Systems,* **2**, 321-355.

[10] Chen, S., and Billings, S.A., 1989a, Recursive prediction error estimator for non-linear models. *Int. J. Control,* **49**, 569-594.

[11] Chen, S., and Billings, S.A., 1989b, Representation of non-linear systems: the NARMAX model. *Int. J. Control,* **49**, 1013-1032.

[12] Chen, S., and Billings, S.A., 1989c, Modelling and analysis of non-linear time series. *Int. J. Control,* **50**, 2151-2171.

[13] Chen, S., Billings, S.A., and Luo, W., 1989, Orthogonal least squares methods and their application to non-linear system identification. *Int. J.*

Control, **50**, 1873-1896.

[14] Chen, S., Billings, S.A., and Grant, P.M., 1990a, Non-linear systems identification using neural networks. *Int. J. Control*, **51**, 1191-1214.

[15] Chen, S., Cowan, C.F.N., Billings, S.A., and Grant, P.M., 1990b, Parallel recursive prediction error algorithm for training layered neural networks. *Int. J. Control*, **51**, 1215-1228.

[16] Chen, S., Billings, S.A., Cowan, C.F.N., and Grant, P.M., 1990c, Practical identification of NARMAX models using radial basis functions. *Int. J. Control*, **52** 1327-1350.

[17] Chen, S., Cowan, C.F.N., and Grant, P.M., 1991a, Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans. Neural Networks*, **2**, 302-309.

[18] Chen, S., Billings, S.A., and Grant, P.M., 1991b, A recursive hybrid algorithm for non-linear system identification using radial basis function networks. *Int. J. Control*, to appear.

[19] Chen, S., Grant, P.M., and Cowan, C.F.N., 1991c, Orthogonal least squares algorithm for training multi-output radial basis function networks. To be presented at *IEE 2nd Int. Conf. Artificial Neural Networks*, Bournemouth Int. Centre, U.K., Nov.18-20, 1991.

[20] Cybenko, G., 1989, Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, **2**, 303-314.

[21] Duda, R.O., and Hart, P.E., 1973, *Pattern Classification and Scene Analysis*, (New York: John Wiley and Sons).

[22] Funahashi, K., 1989, On the approximate realization of continuous mappings by neural networks. *Neural Networks*, **2**, 183-192.

[23] Golub, G., 1965, Numerical methods for solving linear least squares problems. *Numerische Mathematik*, **7**, 206-216.

[24] Goodwin, G.C., and Payne, R.L., 1977, *Dynamic System Identification: Experiment Design and Data Analysis*, (New York: Academic Press).

[25] Hecht-Nielsen, R., 1990, *Neurocomputing*, (Reading: Addison-Wesley).

[26] Hornik, K., 1991, Approximation capabilities of multilayer feedforward

networks. *Neural Networks,* **4,** 251-257.

[27] Kohonen, T., 1987, *Self-Organization and Associative Memory,* (Berlin: Springer-Verlag).

[28] Lapedes, A., and Farber, R., 1988, How neural nets works. *Neural Information Processing Systems,* edited by D.Z. Anderson (New York: American Institute of Physics), pp.442-456.

[29] Leontaritis, I.J., and Billings, S.A., 1985, Input-output parametric models for non-linear systems - Part 1: deterministic non-linear systems; Part 2: stochastic non-linear systems. *Int. J. Control,* **41**, 303-344.

[30] Leontaritis, I.J., and Billings, S.A., 1987, Model selection and validation methods for non-linear systems. *Int. J. Control,* **45**, 311-341.

[31] Lippmann, R.P., 1987, An introduction to computing with neural nets. *IEEE ASSP Magazine,* **4**.

[32] Ljung, L., and Söderström, T., 1983, *Theory and Practice of Recursive Identification,* (Cambridge: MIT Press).

[33] Mead, C., 1989, *Analog VLSI and Neural Systems,* (Reading: Addison-Wesley).

[34] Moody, J., and Darken, C., 1989, Fast-learning in networks of locally-tuned processing units. *Neural Computation,* **1**, 281-294.

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

[36] Ozaki, T., 1985, Non-linear time series models and dynamical systems. *Handbook of Statistics 5: Time Series in the Time Domain,* edited by E.J. Hannan, P.R. Krishnaiah and M.M. Rao (Amsterdam: North-Holland), pp.25-83.

[37] Pao, Yoh-Han, 1989, *Adaptive Pattern Recognition and Neural Networks,* (Reading: Addison-Wesley).

[38] Park, J., and Sandberg, I.W., 1991, Universal approximation using radial-basis-function networks. *Neural Computation,* **3**, 246-257.

[39] Powell, M.J.D., 1987, Radial basis function approximations to polynomials.

Proc. 12th Biennial Numerical Analysiş Conference, Dundee, pp.223-241.

[40] Priestley, M.B., 1980, State-dependent models: a general approach to non-linear time series analysis. *J. Time Series Analysis,* **1**, 47-71.

[41] Rumelhart, D.E., Hinton, G.E., and Williams, R.J., 1986, Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* edited by D.E. Rumelhart and J.L. McClelland (Cambridge: MIT Press), pp.318-362.

[42] Salgado, M.E., Goodwin, G.C., and Middleton, R.H., 1988, Modified least squares algorithm incorporating exponential resetting and forgetting. *Int. J. Control,* **47**, 477-491.

[43] Simmons, G.F., 1963, *Introduction to Topology and Modern Analysis,* (New York: McGraw-Hill).

[44] Sripada, N.R., and Fisher, D.G., 1987, Improved least squares identification. *Int. J. Control,* **46**, 1889-1913.

[45] Weigend, A.S., Huberman, B.A., and Rumelhart, D.E., 1990, Predicting the future: a connectionist approach. *Int. J. Neural Systems,* **1**, 193-209.

List of Figures

Fig. 1. Topology of Multi-Layer Perceptron.

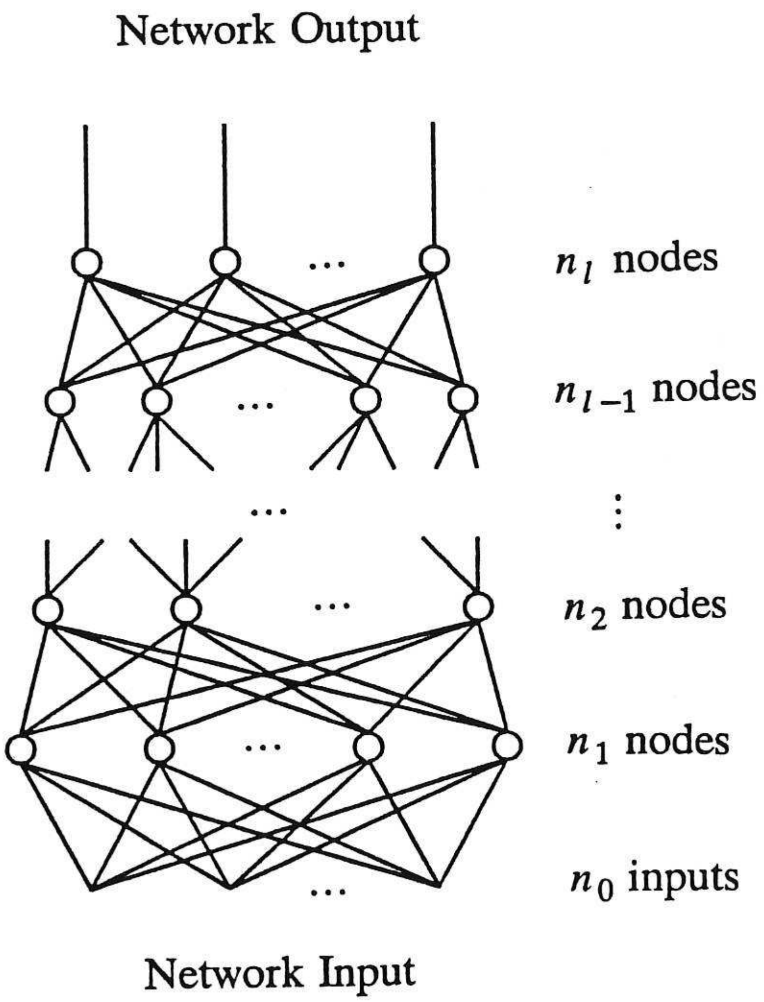
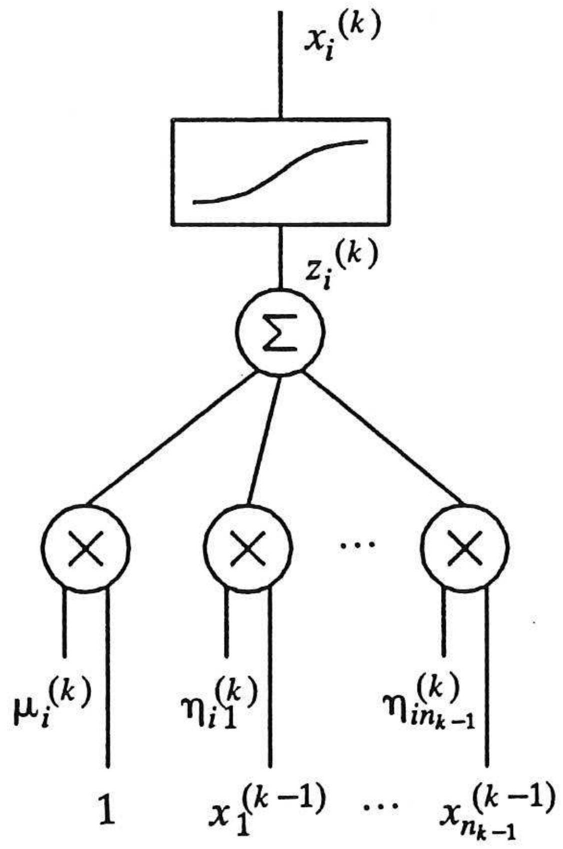


Fig.2. Structure of a Node.



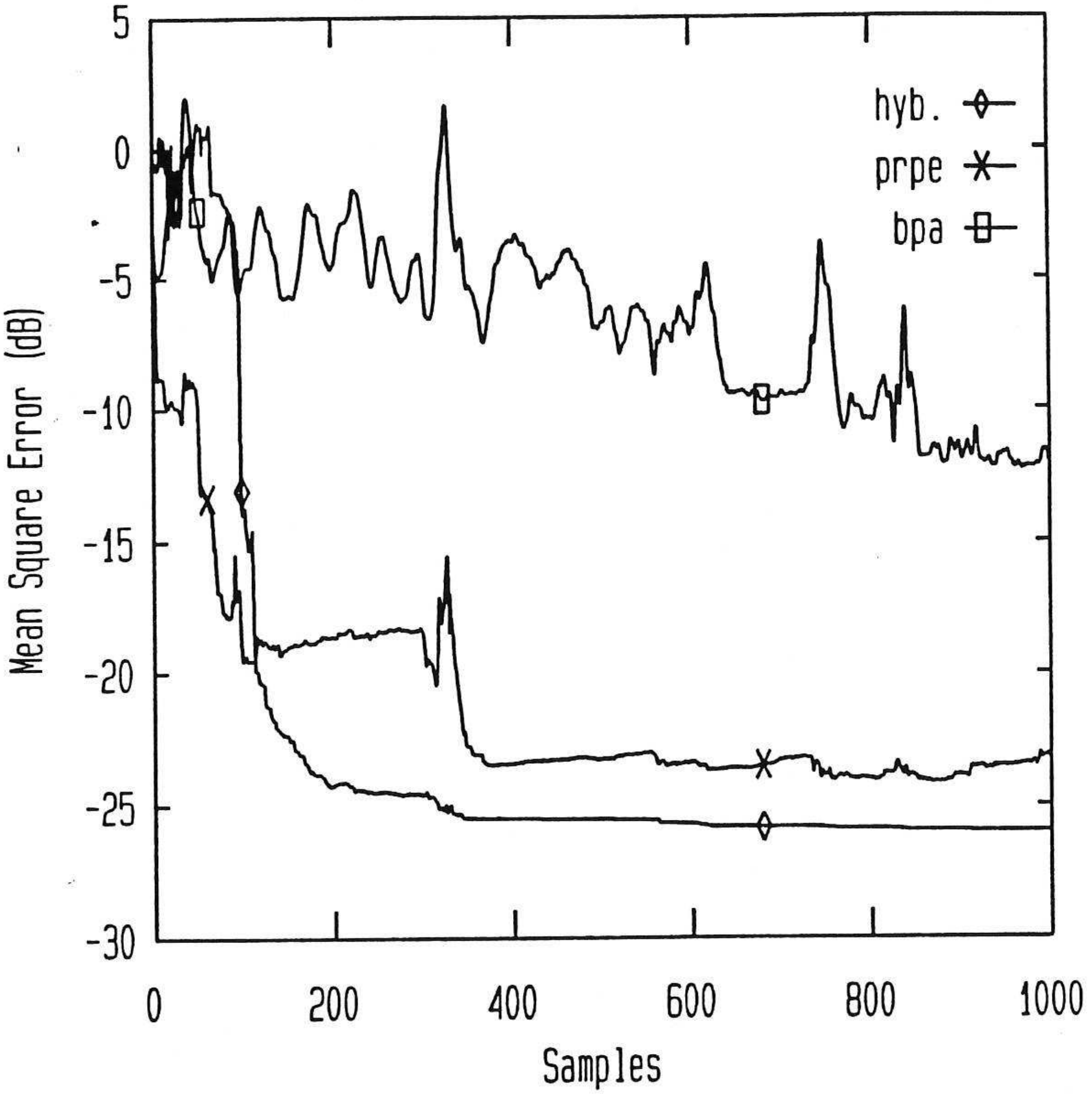


Fig.3. Evolution of Mean Square Error. Liquid level system.

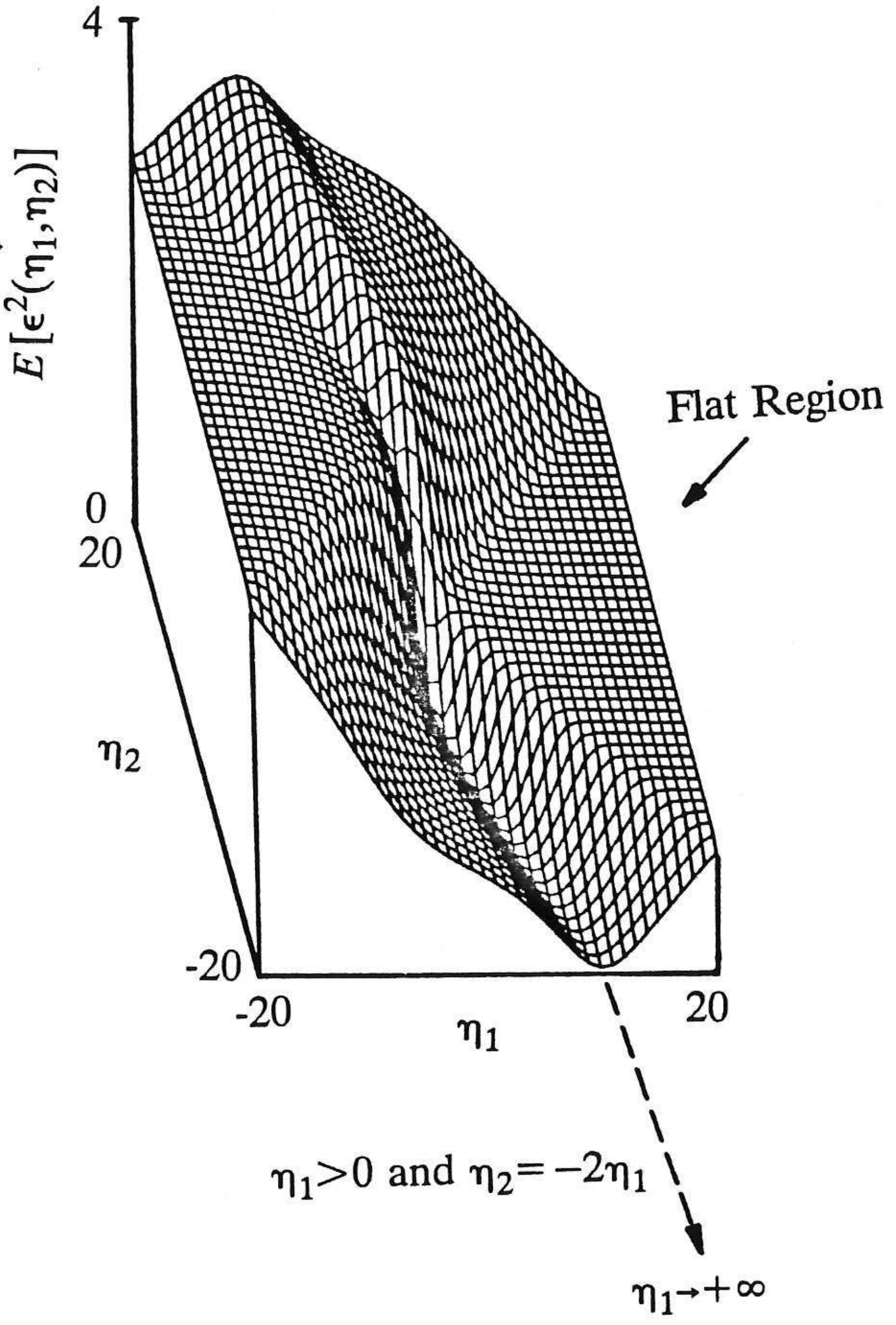
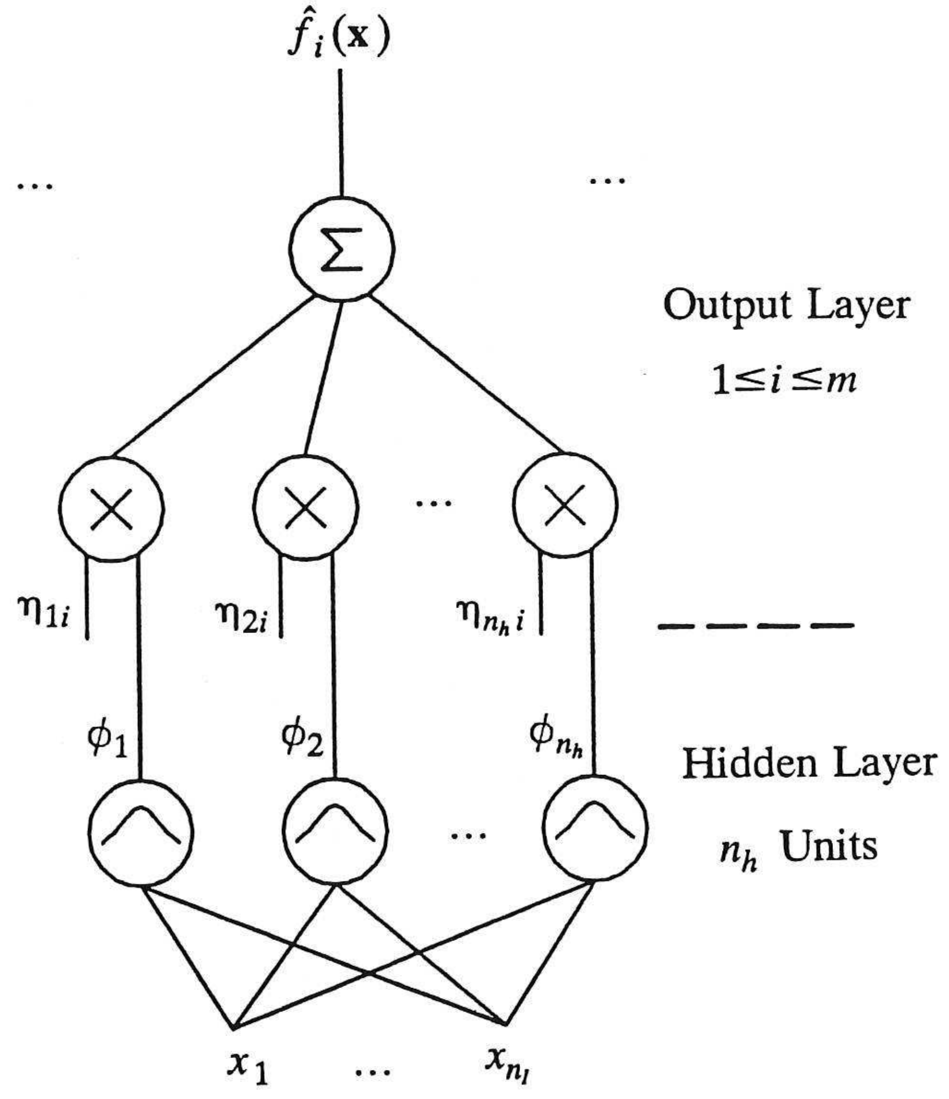


Fig.4. Error Surface of a Simple Classifier.

Fig.5. Schematic of Radial Basis Function Network.



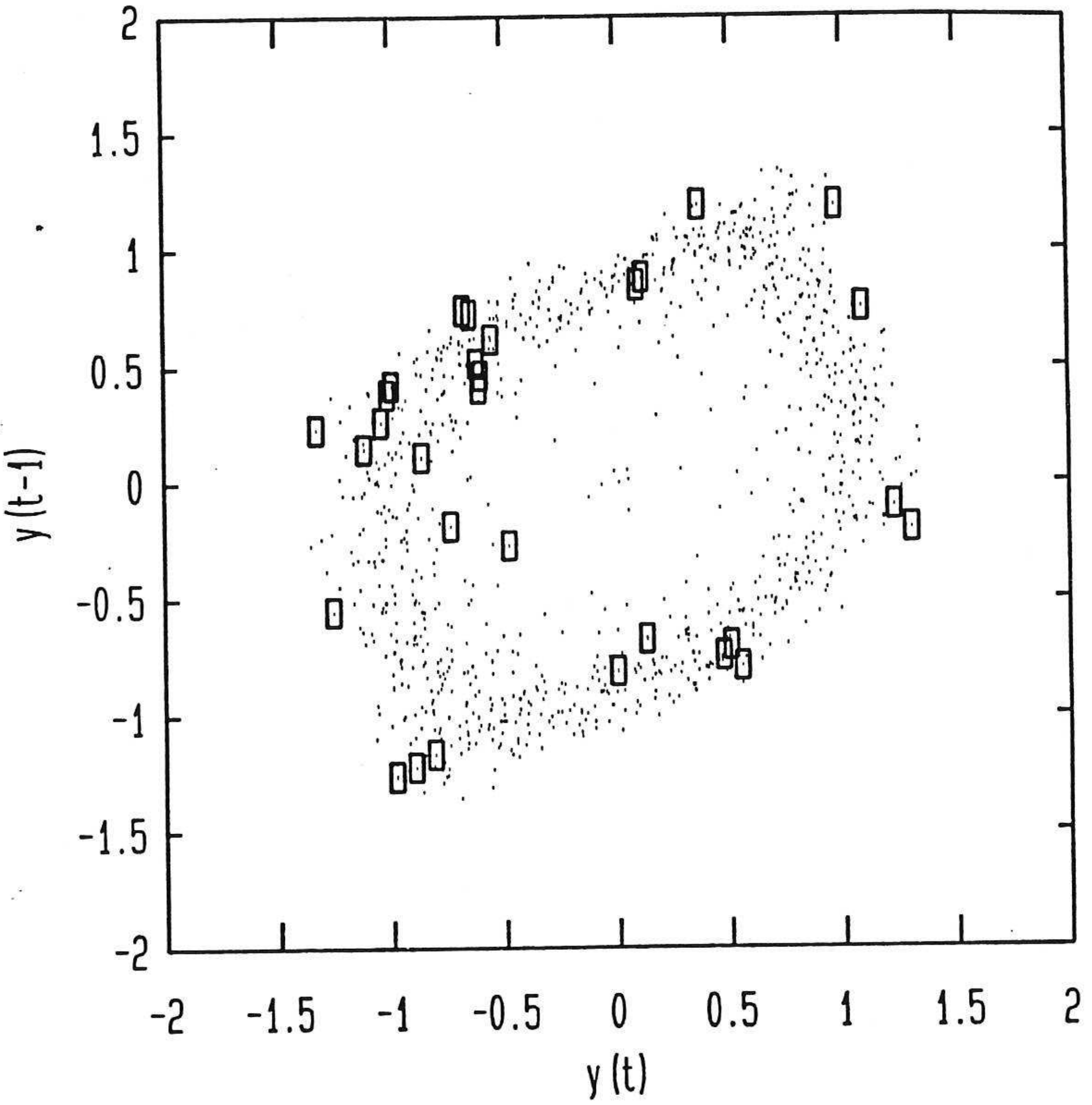


Fig.6. Distribution of System Observations and RBF Centres Obtained Using the OLS Algorithm. 1000 samples, □: position of RBF centre.

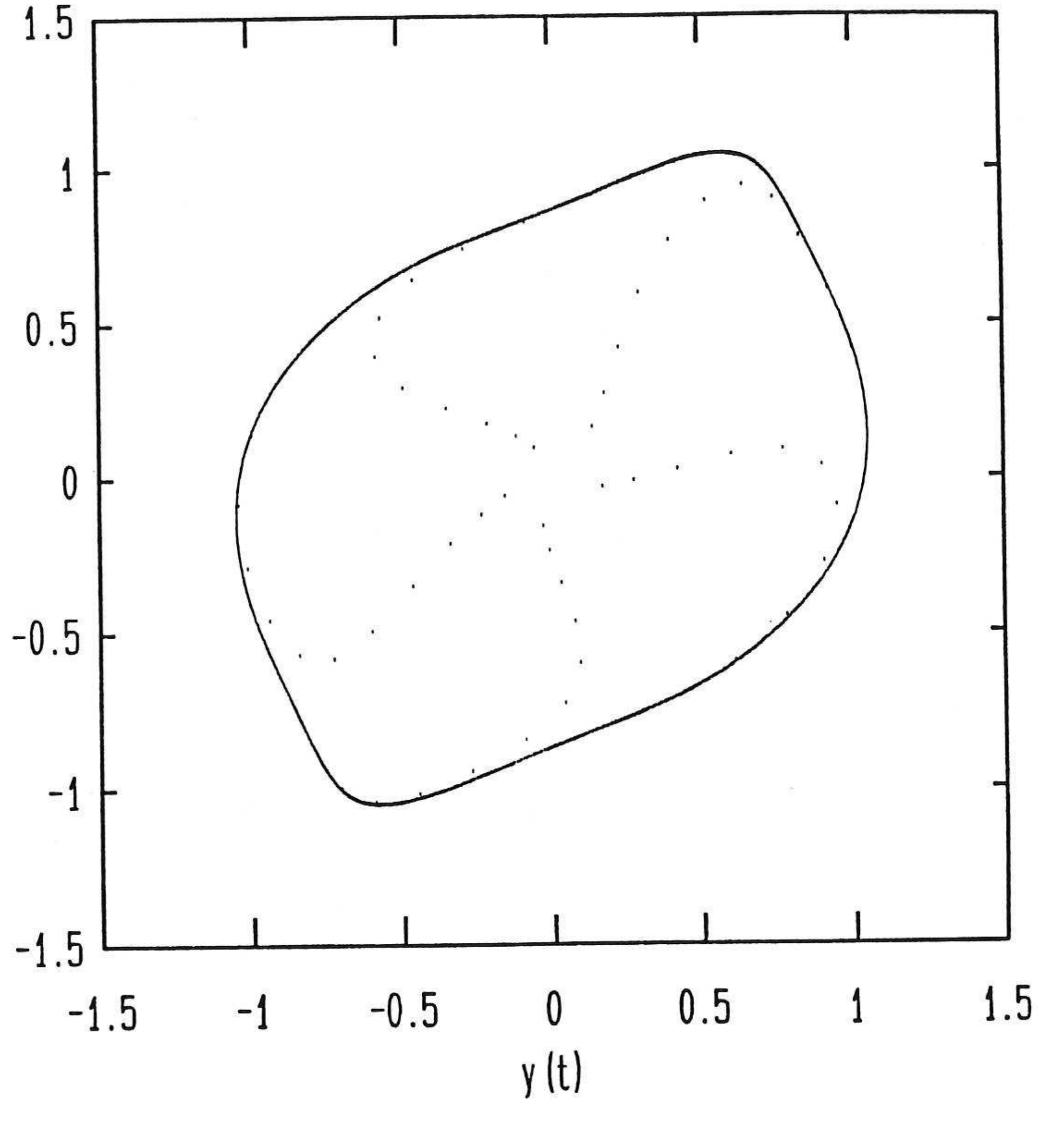


Fig.7. Limit Cycle Generated by Autonomous System Response. 1500 samples, initial condition: $y(0)=y(-1)=0.1$.

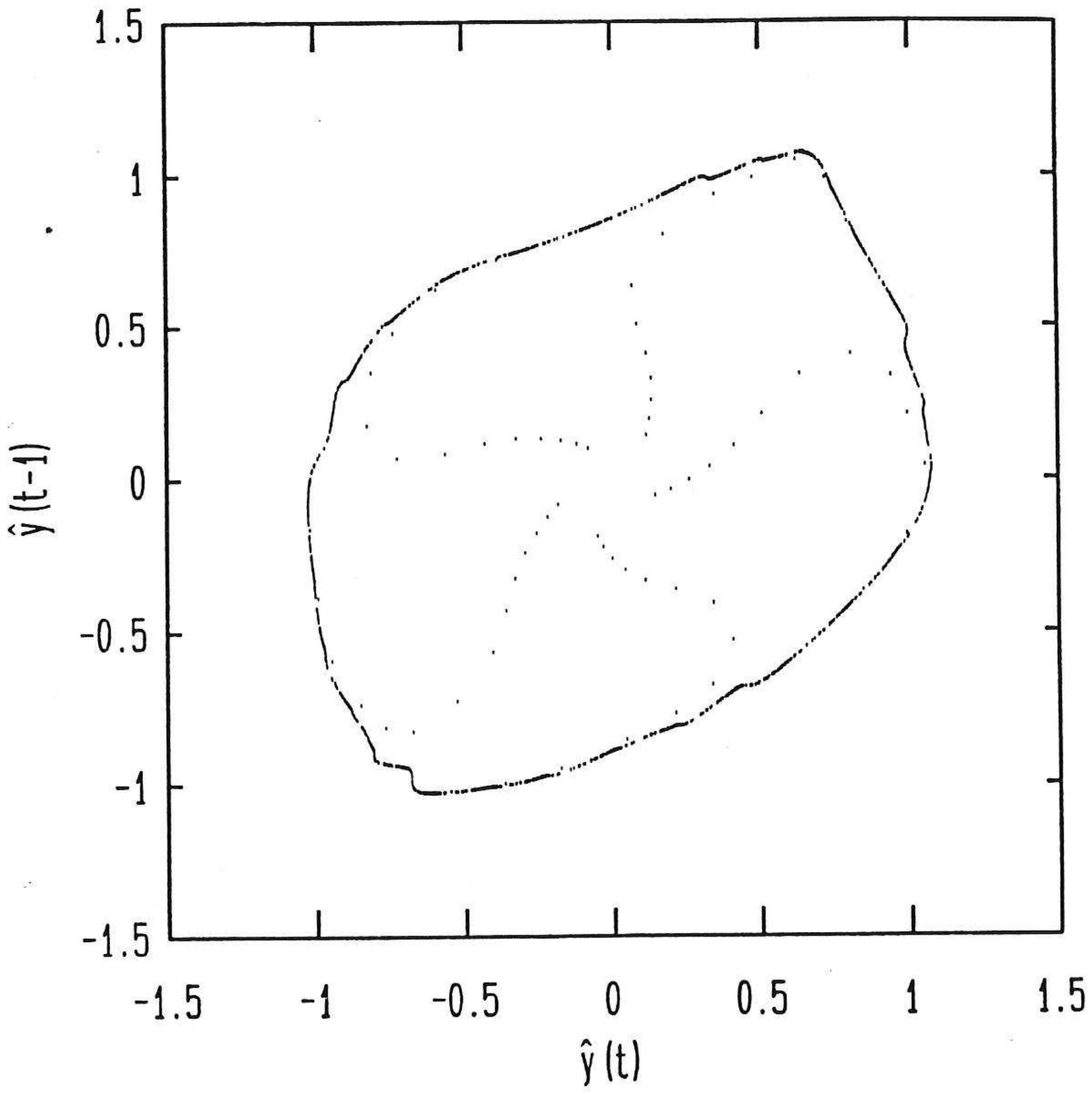


Fig.8. Limit Cycle Generated by Iterative Network Response. 1500 samples, initial condition: $\hat{y}(0)=\hat{y}(-1)=0.1$, the RBF network was obtained using the OLS algorithm.

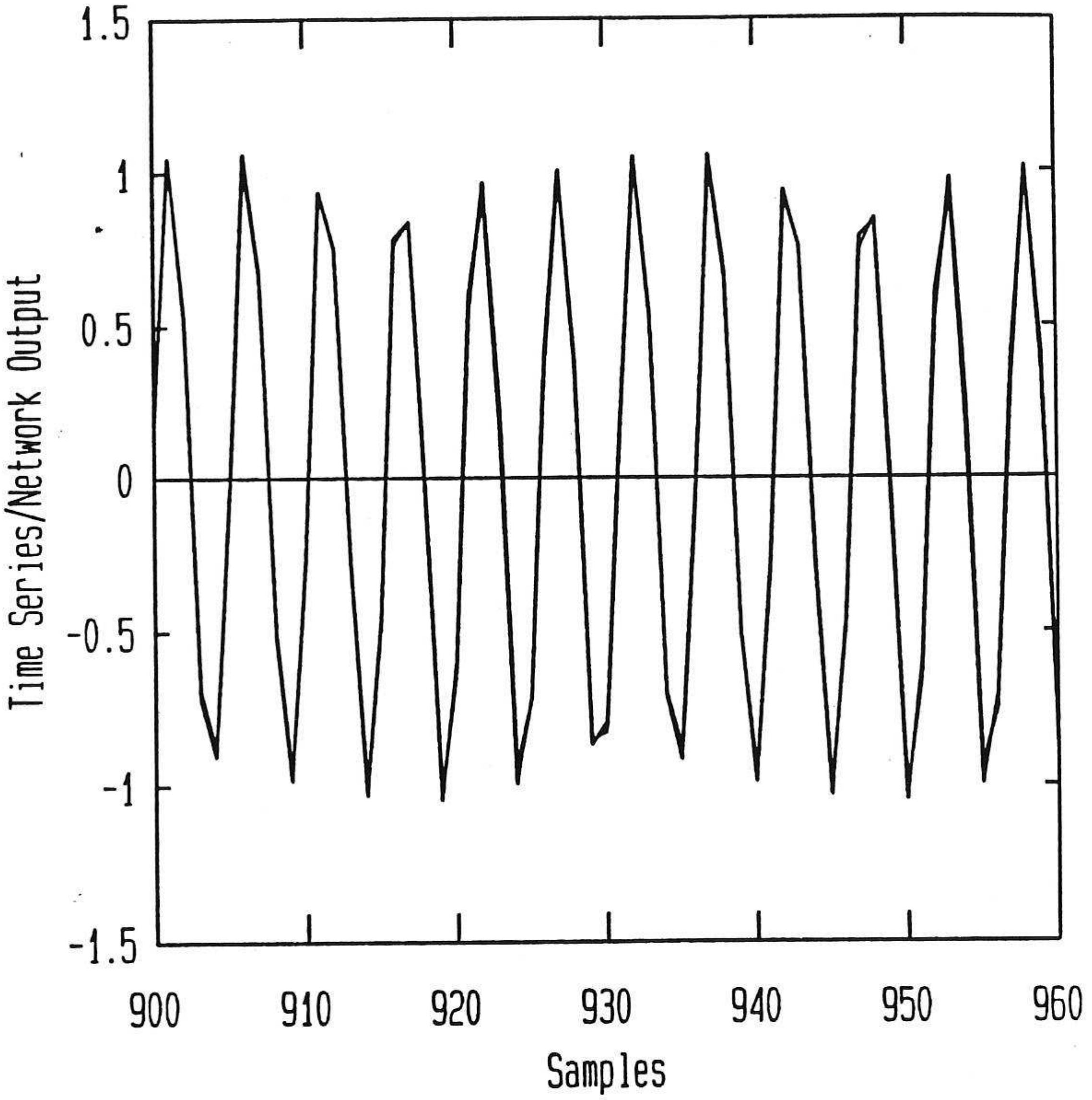


Fig.9. Waveforms of Autonomous System Response and Iterative Network Response. The RBF network was obtained using the OLS algorithm.

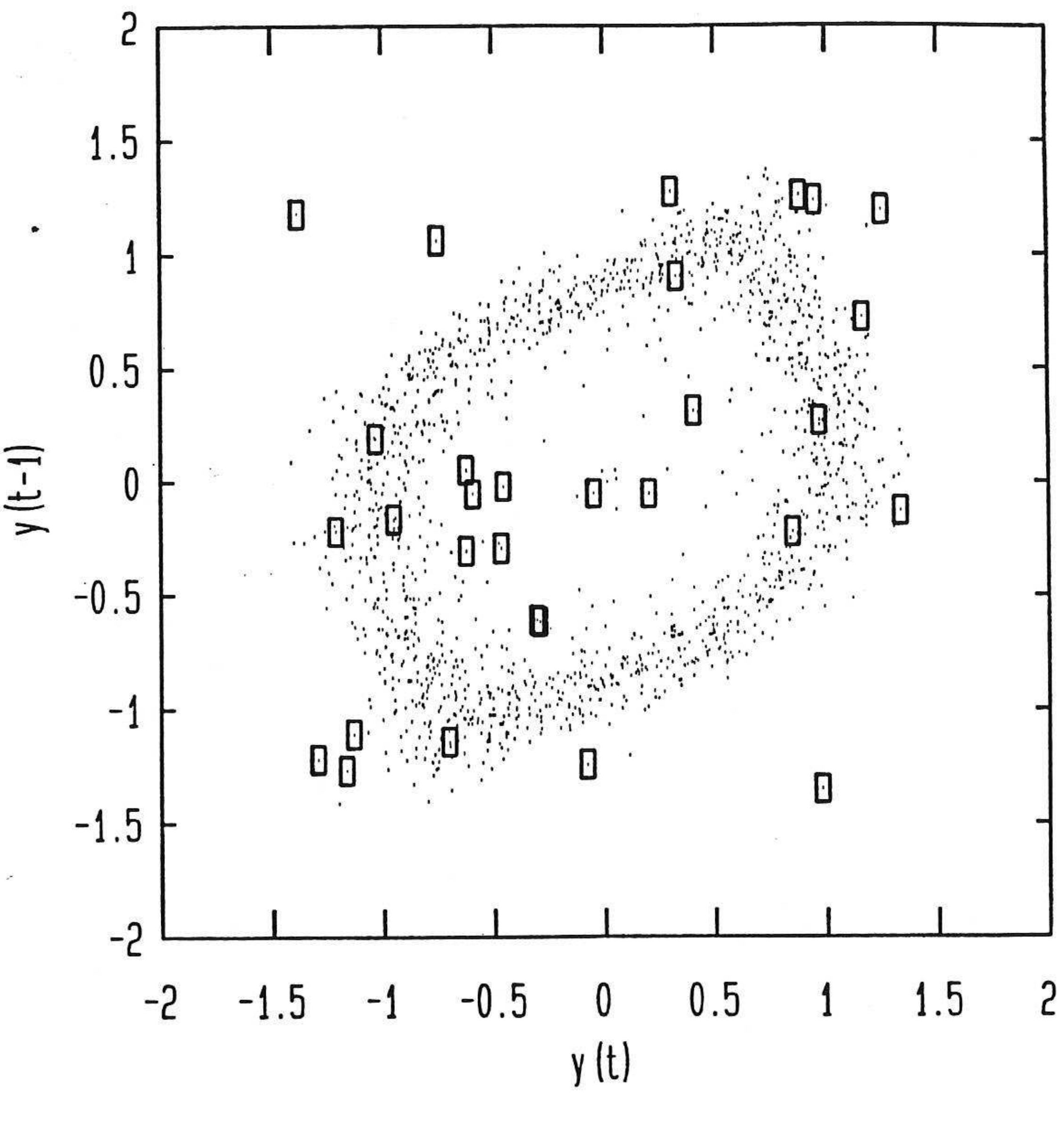


Fig.10. Distribution of System Observations and RBF Centres Obtained Using the Hybrid Algorithm. 1500 samples, □: position of RBF centre.

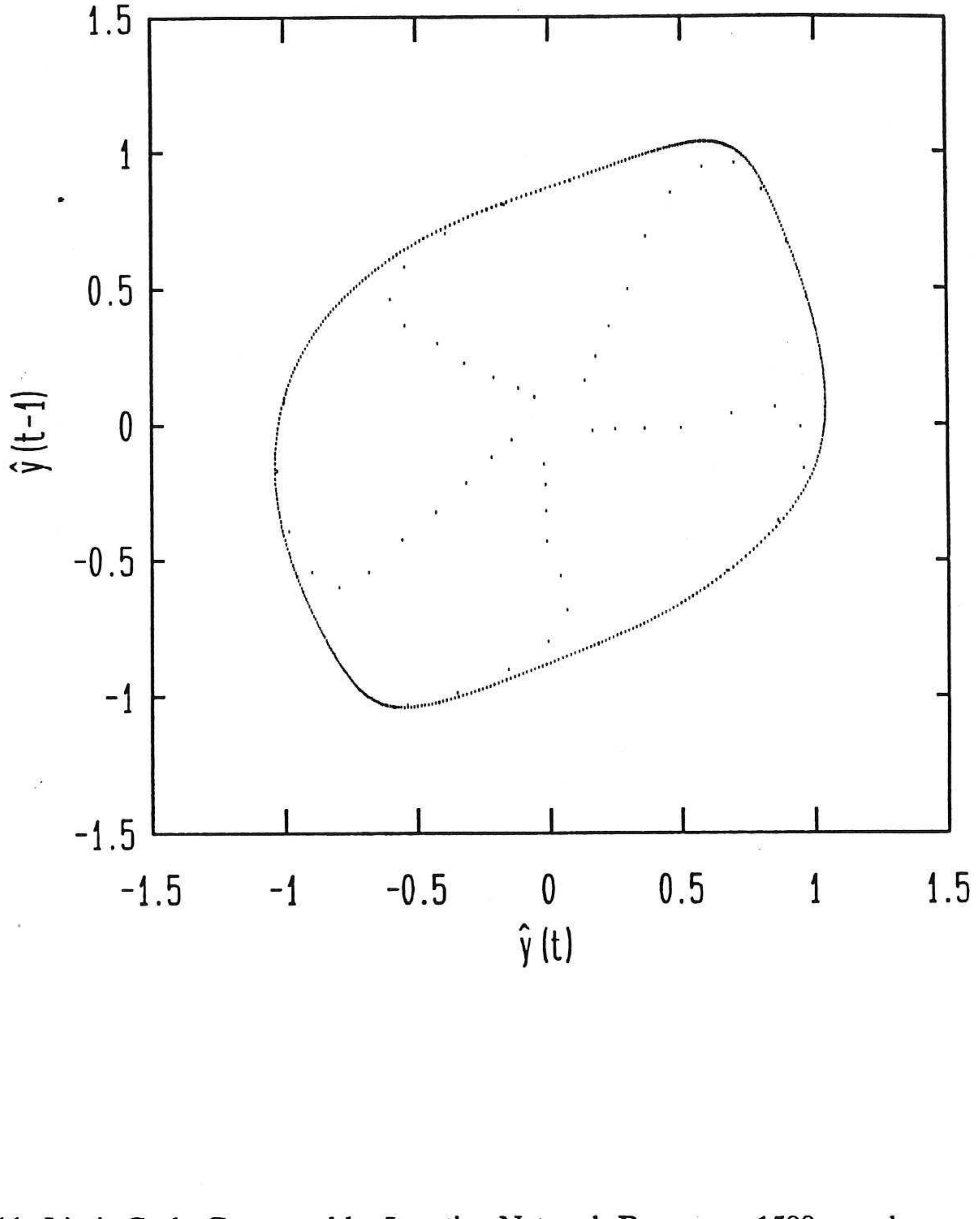


Fig.11. Limit Cycle Generated by Iterative Network Response. 1500 samples, initial condition: $\hat{y}(0)=\hat{y}(-1)=0.1$, the RBF network was obtained using the hybrid algorithm.

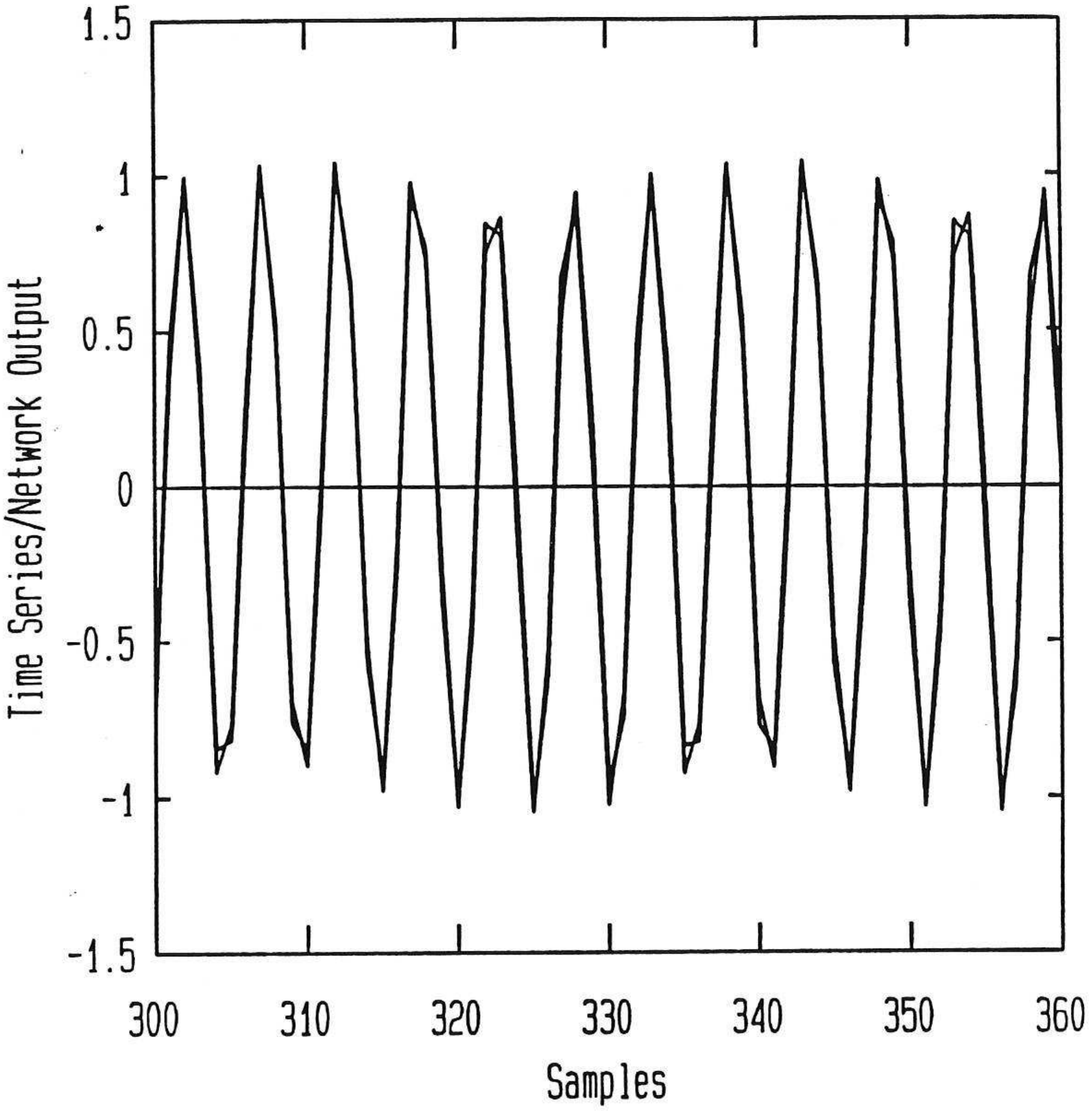


Fig.12. Waveforms of Autonomous System Response and Iterative Network Response. The RBF network was obtained using the hybrid algorithm.

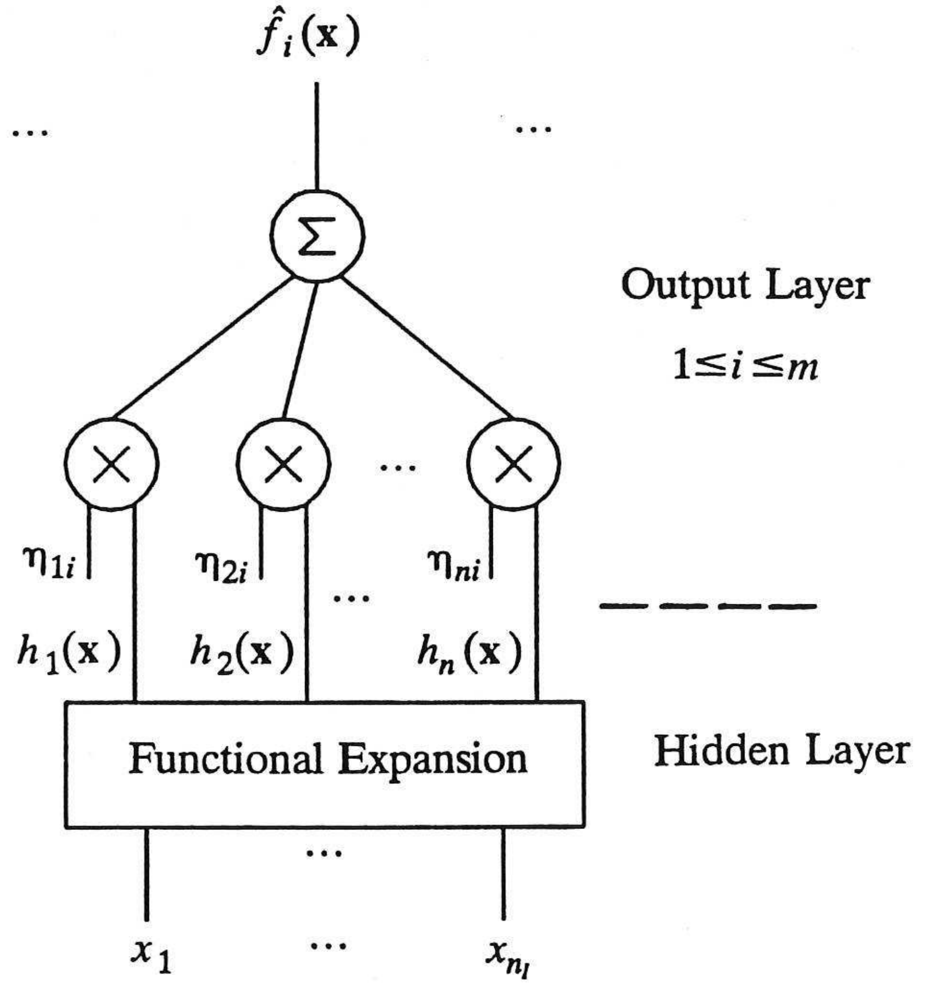


Fig.13. Schematic of Functional-Link Network.

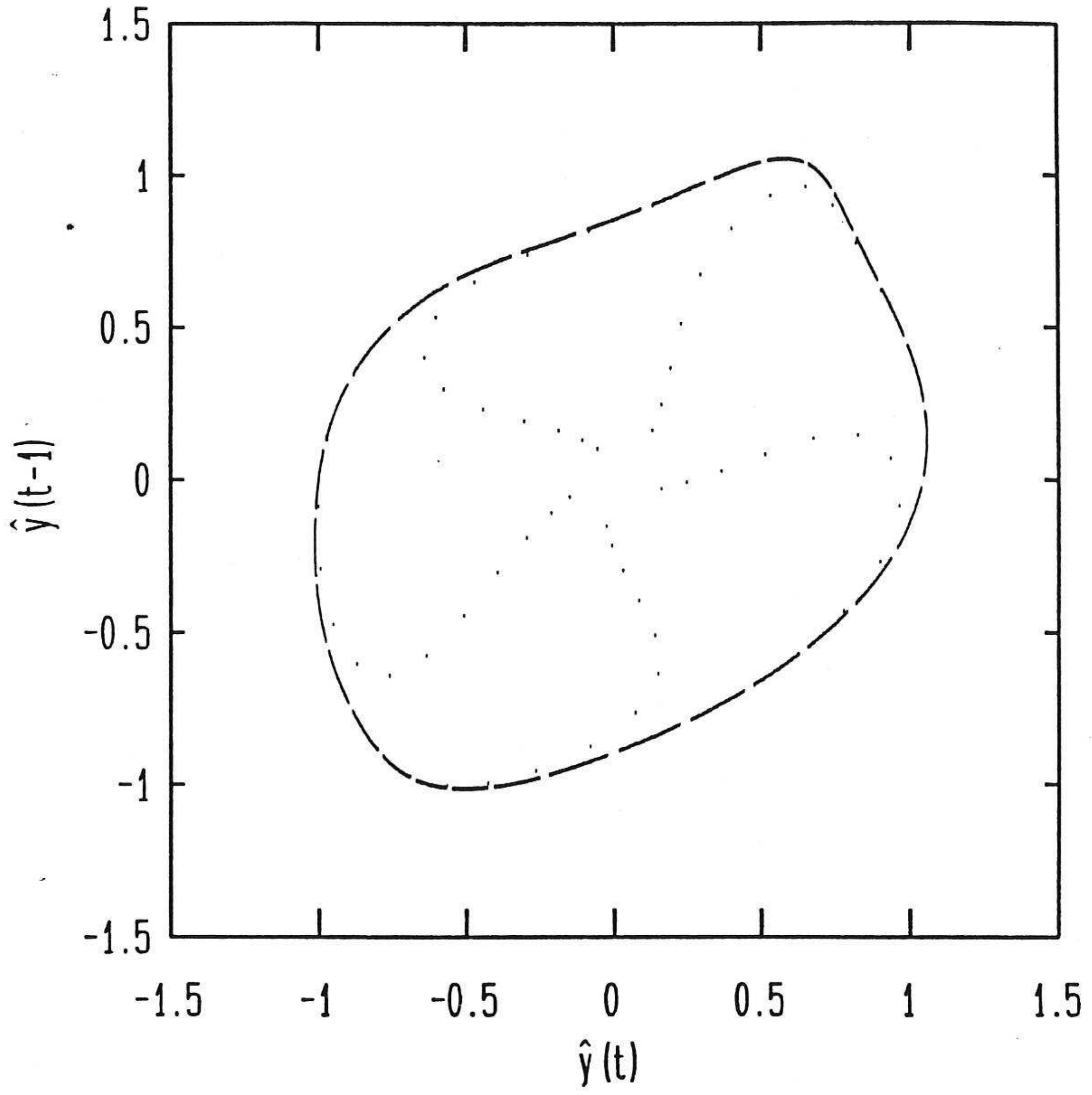


Fig.14. Limit Cycle Generated by Iterative Subset Polynomial Model. 1500 samples, initial condition: $\hat{y}(0)=\hat{y}(-1)=0.1$.