Direct Adaptive Control of Unknown Nonlinear Systems Using Radial Basis Function Networks with Gradient Descent and K-means

Tamer A. Alzohairy

Abstract—**In this paper we propose a direct adaptive neural network control strategy for a class of unknown nonlinear systems. The adaptive controller is based on Radial Basis Function neural network. Training the RBF network as a neural model of the system using Gradient descent method initialized with unsupervised K-means clustering algorithm, control signals are directly obtained by minimizing the instant difference between a set point and the output of the Radial Basis function neural network model using the well established gradient descent rule. Since the training algorithm guarantees that the output of the RBF neural network model approaches that of the actual system, it is shown that the control signals obtained can also make the real system output close to a set point. Simulation results for both SISO and MIMO type nonlinear systems have been presented toward the end of the paper to show the validity and performance of the proposed method1.**

Index Terms—**Direct adaptive control, discrete-time systems, k-means clustering algorithm, MIMO systems, Radial Basis Function (RBF), SISO systems.**

I. Introduction

Dynamic and complex system control has been an important research area mainly due to the difficulties in modeling and estimating system nonlinearities. Controllers developed for such systems should be able to track the dynamic nonlinear plant output precisely because often the linear controllers fail to perform effectively. Thus a changing dynamic controller is necessary to control such a plant.

In recent years Artificial Neural Network (ANN) has been effectively utilized as a tool to generate dynamic control schemes. Lot of research has been done using these computational intensive algorithms in parallel with a mathematical formulation or by itself in developing such controllers. Some of the relevant research work involving ANN as part of the control scheme is illustrated next.

Identification and control of dynamic systems using a backpropagation neural network is shown in [1]-[6]. A robust adaptive control of uncertain nonlinear systems using neural network is proposed in [7]. It was shown that the explicit identification of the nonlinear system is possible if the persistency of excitation condition is fulfilled.

Utilization of RBF neural network as an intelligent controller is often been emphasized in many of these and other researches. RBF neural networks differ from multi layered neural networks in that it has simple structure and fast learning algorithms and they perform local representations. Moreover, the basis function in the RBF neural network structure ensures that only the weights located in the vicinity of the input need adjustment during training [8]. This feature makes RBF neural networks attractive as an on-line controller where there is often no control of the order of presentation of the data samples used during training.

In this paper, we propose a direct adaptive control scheme to achieve output tracking of a class of nonlinear systems using RBF neural network. The main advantage of direct adaptive control scheme over an indirect adaptive control scheme is that in a direct adaptive control scheme there is no need for explicit system identification. In indirect adaptive control scheme, the system is generally identified off-line from its input-output data and the controller is designed based on the identified system model. The identified model should be accurate enough for better performance of the controller. Moreover stability is a critical issue in indirect adaptive control. But in direct adaptive control, the controller is designed in such a way that the closed loop stability is maintained while the tracking error converges to 0 with time.

In this paper, RBF neural network is used on line to estimate the system using gradient descent technique [9] initialized with unsupervised k-means clustering method [10] to improve the success of RBF network in identification of the unknown nonlinear system. Taking the resulting RBF neural network estimation as a known nonlinear dynamic model for the system, control signals can be directly obtained using the well-established gradient descent rule. As we will see, satisfactory solutions to the tracking problem are obtained by applying radial basis function (RBF) neural network.

This paper is organized as follows: In section II the unknown nonlinear system and the problem under consideration are stated. Section III gives details of the RBF neural network and algorithms used to adjust its parameters for identification purpose. The structures for direct control in case of SISO and MIMO systems are given in Section IV. In section V the algorithm used for direct control in case of SISO and MIMO systems are given. Section VI presents the simulation results for both SISO and MIMO type nonlinear systems. A comparison study of the direct control scheme in two cases is shown in Section VII. Finally, some conclusions are remarked in Section VIII.

Manuscript received May 12, 2011; revised August 2, 2011.

T. A. Alzohairy is with the Computer Science Department, Arriyadh Community College, King Saud University, Malaz, P. O. Box 28095, Riyadh 11437, Saudi Arabia. On leave from Mathematics Department, Faculty of Science, Al-Azhar University, Nasr City (11884), Cairo, Egypt (e-mail: tt_tamer@yahoo.com).

II. Statement of The Problem

This section covers the details regarding the SISO and MIMO systems under study.

A. Single Input Single Output Systems

Consider a single input, single output system:

$$y_p(k+1) = f[y_p(k), y_p(k-1), \dots, y_p(k-L+1)] + \sum_{j=0}^{L-1} b_j u(k-j) \tag{1}$$

where $y_p(k)$ is the plant output, $u(k)$ is the plant input, $b_0, b_1, \dots, b_{L-1}$ are system parameters where $b_0 \neq 0$, and $f[.]$ is a nonlinear function.

The control problem is to determine the bounded input $u(k)$ so that the plant output $y_p(k)$ follows the model output $y_m(k)$ asymptotically, that is, $\lim_{k\to\infty} \| y_p(k) - y_m(k) \| = 0$.

We shall consider the case when the function $f[.]$ and the parameters $b_0, b_1, \dots, b_{L-1}$ are unknown. In this case one RBF neural network is used to approximate the SISO system.

B. Multi Input Multi Output Systems

Consider a discrete time nonlinear multivariable system described by the difference equations:

$$\begin{aligned} y_{p1}(k+1) &= f_1[y_{p1}(k), \dots, y_{p1}(k-L+1), \dots, y_{pn}(k), \dots, y_{pn}(k-L+1)] + \sum_{j=0}^{L-1} b_{1j} u_1(k-j) \\ y_{p2}(k+1) &= f_2[y_{p1}(k), \dots, y_{p1}(k-L+1), \dots, y_{pn}(k), \dots, y_{pn}(k-L+1)] + \sum_{j=0}^{L-1} b_{2j} u_2(k-j) \\ &\dots \qquad \dots \qquad \dots \\ y_{pn}(k+1) &= f_n[y_{p1}(k), \dots, y_{p1}(k-L+1), \dots, y_{pn}(k), \dots, y_{pn}(k-L+1)] + \sum_{j=0}^{L-1} b_{nj} u_n(k-j) \end{aligned} \tag{2}$$

where, b_{ij} for $(i:1\dots n)$, $(j:0\dots L-1)$, $b_{i0} \neq 0$ are the system parameters, $u(k) = [u_1(k), u_2(k), \dots, u_n(k)]^T$ and $y_p(k) = [y_{p1}(k), y_{p2}(k), \dots, y_{pn}(k)]^T$ are the input and output vectors respectively, $f_1, f_2, \dots, f_n$ are nonlinear functions, and $L \leq n$.

The control problem is to determine the bounded input vector $u(k) = [u_1(k), u_2(k), \dots, u_n(k)]^T$ so that the plant output vector $y_p(k) = [y_{p1}(k), y_{p2}(k), \dots, y_{pn}(k)]^T$ follows the model output vector $y_m(k) = [y_{m1}, y_{m2}, \dots, y_{mn}]^T$ asymptotically, that is, $\lim_{k\to\infty} \| y_{pi}(k) - y_{mi}(k) \| = 0$ for $(i:1\dots n)$.

We shall consider the case when the functions $f_1, f_2, \dots, f_n$ and the parameters $b_{10}, b_{11}, \dots, b_{1(L-1)}, \dots, b_{n0}, b_{n1}, \dots, b_{n(L-1)}$ are unknown. In this case n RBF neural networks are used to approximate the multivariable system.

III. RBF Neural Network

In this section the structure of the RBF neural network and the algorithms used to adjust its parameters for identification purpose are introduced.

A. Structure of RBF Neural Network

A RBF network is a three-layer feed-forward neural network. The mapping from input to output is nonlinear, but from hidden layer to output layer is linear. Learning rate is quickened greatly and the problem of local minimum is avoided. A typical RBF network configuration is shown in Fig. 1.

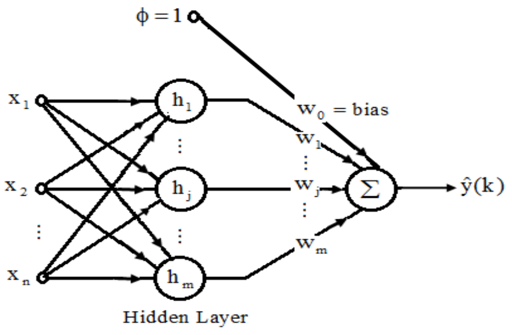


Fig. 1. RBF neural network configuration.

In the structure of RBF neural network, the first layer $X = [x_1, x_2, \dots, x_n]^T$ is the input vector of the network. Neurons of the second layer (hidden layer) are activated by radial-basis function. Suppose the radial vector of RBF network is $H = [h_1, h_2, \dots, h_m]^T$, where h_j is multivariate Gaussian function.

$$h_j = \exp\left(-\frac{X - C_j}{2\sigma_j^2} \right) \tag{3}$$

The center vector of the j^{th} network node is $C_j = [c_{j1}, c_{j2}, \dots, c_{ji}, \dots, c_{jn}]^T$; $j = 1, 2, \dots, n$ and the radial width vector is $\sigma = [\sigma_1, \sigma_2, \dots, \sigma_j, \dots, \sigma_m]^T$, where σ_j is the radial parameter and $\sigma_j > 0$.

The weight vector of the network is W, and $W = [w_1, w_2, \dots, w_j, \dots, w_m]^T$.

The network output $\hat{y}$ is found by a linearly weighted sum of the number of basis functions in the hidden layer.

$$\hat{y}(k+1) = w_0 + w_1 h_1 + w_2 h_2 + \dots + w_m h_m = w_0 + \sum_{j=1}^{m} w_j h_j \tag{4}$$

B. Training Radial Basis Function Network

The objective of the training process is to adjust parameters of the RBF network to minimize the error between the RBF network output $\hat{y}(k+1)$ and that of the real plant $y_p(k+1)$. The learning phase in the radial basis function neural network can be divided into three steps and two phases. The three steps are 1) find the centers, 2) find the widths, and 3) weight training on the outer layer. The two phases proposed in this paper are:

1) Unsupervised learning to give initial values for the location of centers and widths (using K-means clustering algorithm and P-nearest neighbor method respectively).
2) Supervised learning, initialized with unsupervised learning given by (1), to improve the location of centers and widths, and estimating output layer weights (using gradient descent method).

The following subsections give a detailed discussion for the two phases mentioned above.

B.1. Phase (1): Initializing Parameters for Radial Basis Functions using unsupervised methods

Radial-basis functions, as in (3), each have two parameters, C_j and σ_j for the j^{th} basis function h_j. There are numerous algorithms that can be used to find the centers and widths of the hidden layer. In this paper, we propose an adaptive K-means clustering algorithm and P-nearest neighbor method to give initial values for the radial-basis functions parameters. The K-means clustering algorithm and P-nearest neighbor method are shown as follows.

K-means clustering algorithm:

Given a set of observations $(X(1), X(2), \ldots, X(\mathrm{k}), \ldots, X(\mathrm{z}))$, where each observation is an n-dimensional real vector, K-means clustering aims to partition the z observations into $K = m$ sets $(K < z)$ $S = \{S_1, S_2, \ldots, S_{\mathrm{K}}\}$ so as to minimize mean squared distance from each observation to its closest center [11]. The recursive K-means algorithm is given as follows:

1) Choose a set of centers $\{ C_1, C_2, \ldots, C_{\mathrm{j}}, \ldots, C_m \}$ arbitrarily and give the initial learning rate $\gamma(0) = 1$.
2) Compute the minimum Euclidean distance

$$\begin{aligned} &L_j(k) = \left\| X(k) - C_j(k-1) \right\| \qquad \mathrm{j}:1\ldots m \\ &\mathrm{r} = \arg\left|\min \mathrm{L}_{\mathrm{j}}(k)\right| \end{aligned} \tag{5}$$

3) Adjust the position of this centers as follows:

$$\begin{aligned} C_j(k) &= C_j(k-1) + \gamma(k)(\underline{x}(k) - C_j(k-1)) \quad (\mathrm{j} = \mathrm{r}) \\ &= C_j(k-1) \quad (\mathrm{j} \neq \mathrm{r}) \end{aligned} \tag{6}$$

4) $k = k+1$, $\gamma(k) = 0.998\gamma(k-1)$ and go to 2.

P-nearest neighbor algorithm:

After the RBF centers have been found, the width is calculated. The width represents a measure of the spread of data associated with each node. Calculation of the width is usually done using the P-nearest neighbor algorithm. A number P is chosen and for each center, the P nearest centers is found. The root-mean squared distance between the current cluster and its P nearest neighbors is calculated, and this is the value chosen for σ. So, if the current cluster center is C_j, the value of width is given by:

$$\sigma_j = \sqrt{\frac{1}{P}\sum_{i=1}^{P}(C_j - C_i)^2} \tag{7}$$

A typical value of P is 2, in which case σ is set to be the average distance from the two nearest neighboring cluster centers.

B.2. Phase (2): Optimize the location of centers and widths obtained from phase (1) of Radial Basis Functions and Estimating Output layer weights using supervised gradient descent method

Drawback of purely unsupervised method, such as K-means clustering, is that clustering may not be relevant for the target function. Then improving parameters using gradient descent is imported for the control process. Train the network like in backpropagation using gradient descent method; one can then determine improved values of the basis function centers and widths together with the output unit weights.

Our error function is defined as

$$\mathrm{E} = \frac{1}{2}e^2(k+1) = \frac{1}{2}(y_p(k+1) - \hat{y}(k+1))^2. \tag{8}$$

in order to minimize the error between the identification model output $\hat{y}(k+1)$ and that of the real plant $y_p(k+1)$, gradient descent method is adopted here to modify weights of the output layer, node center and node width parameters. The corresponding modifier formulas are as follows:

$$w_j(k+1) = w_j(k) + \eta e(k+1)\frac{\partial \hat{y}(k+1)}{\partial w_j(k)}. \tag{9}$$

$$c_{ji}(k+1) = c_{ji}(k) + \eta e(k+1)\frac{\partial \hat{y}(k+1)}{\partial c_{ji}(k)}. \tag{10}$$

$$\sigma_j(k+1) = \sigma_j(k) + \eta e(k+1)\frac{\partial \hat{y}(k+1)}{\partial \sigma_j(k)}. \tag{11}$$

Differentiating (4) with respect to the corresponding parameter using equation (3) we may write equations (9)-(11) as:

$$w_j(k+1) = w_j(k) + \eta \; e(k+1)\, h_j \tag{12}$$

$$c_{ji}(k+1) = c_{ij}(k) + \eta e(k+1) w_j h_j \frac{(x_i - c_{ji})}{\sigma_j^{\,2}} \tag{13}$$

$$\sigma_j(k+1) = \sigma_j(k) + \eta e(k+1) w_j h_j \frac{\left\| X - C_j \right\|^2}{\sigma_j^3(k)} \tag{14}$$

where η is learning rate.

IV. Structure for Direct Control

In this section the closed loop control structures for both SISO and MIMO systems are given

A. Structure of The Closed Loop Control for Siso Systems

The control process in case of SISO system, given in Section II.A., is to find the control signal $u(k)$ such that the output of the system $y_p(k)$ is made as close as possible to model output signal $y_m(k)$. Fig. 2 shows the structure of the closed loop control system used for this purpose which consists of 1) the system (1); 2) a RBF neural network to give estimate $\hat{y}(k+1)$ of the unknown system output $y_p(k)$; 3) a controller obtained using gradient descent method.

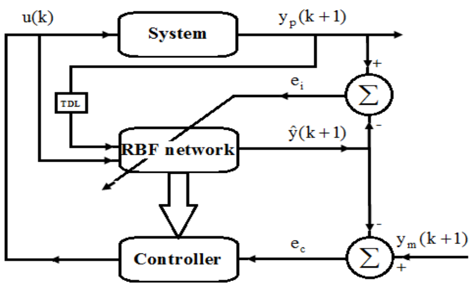


Fig. 2. RBF neural network Controller for SISO system.

B. Structure of the Closed Loop Control for Mimo Systems

The control process in case of MIMO system, given in Section II.B., is to find the bounded control vector $\mathrm{u(k)}=[u_1(k),u_2(k),\ldots,u_n(k)]^{\mathrm{T}}$ such that the output of the system $y_p(k)=[y_{p1}(k),\mathrm{y}_{p2}(k),\ldots,\mathrm{y}_{p\mathrm{n}}(k)]^T$ is made as close as possible to model output signal $y_m(k)=[y_{m1},y_{m2},\ldots,y_{m\mathrm{n}}]^T$. Fig. 3 shows the structure of the closed loop control system used for this purpose, with $n=2$, which consists of 1) the system (2); 2) two RBF neural networks to give estimate $\hat{y}_1(k+1),\hat{\mathrm{y}}_2(k+1),\ldots,\hat{\mathrm{y}}_\mathrm{n}(k+1)$ of the unknown system outputs $y_p(k)=[y_{p1}(k),\mathrm{y}_{p2}(k),\ldots,\mathrm{y}_{p\mathrm{n}}(k)]^T$; 3) two controllers obtained using gradient descent method.

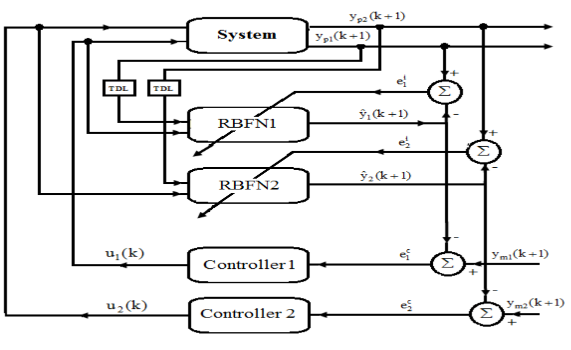


Fig. 3. RBF neural network Controller for MIMO system of order 2.

V. Algorithm Description for Direct Control

This section covers the details of the algorithms used for direct control of unknown SISO and MIMO systems given by (1) and (2). The approach used for direct control is as follows: RBF neural network is used on-line to estimate the system. K-means clustering algorithm and P-nearest neighbor method are used to give initial values for the location of centers and widths. Gradient method is applied to improve the centers and widths of the radial basis functions and to find weights of the output layer. Taking the resulting RBF neural network estimation as a known nonlinear dynamic model for the system, control signals can be directly obtained using well-established gradient descent rule. The following subsections give a detailed discussion for the two algorithms mentioned above.

A. Direct Controller Algorithm for SISO Systems

The structure of Fig. 2 is used for direct controller of the unknown nonlinear SISO system given by (1). The following steps describe the control process:

1) The unknown nonlinear system (1) is expressed by the following model equation

$$\hat{y}(k+1)=\hat{f}(y_p(k),y_p(k\text{-}1),\ldots,y_p(k\text{-}\mathrm{L}\text{-}1),u(k),\\ u(k\text{-}1),\ldots,u(k\text{-}\mathrm{L}+1)) \tag{15}$$

using only one RBF neural network, where $\hat{\mathrm{y}}(\mathrm{k}+1)$ is the output of the RBF neural network given by (4) and $\hat{\mathrm{f}}$ is the estimate of $f[y_p(k),y_p(k-1),\ldots,y_p(k-L+1)]+\sum_{j=0}^{L-1} b_j u(k-j)\cdot$

Training the RBF network must guarantees that

$$[y_p(k+1)-\hat{y}(k+1)]^2=\min. \tag{16}$$

Training the radial basis function neural network can be done with two separate phases: *First,* the initial values for the location of centers and widths are obtained using unsupervised K-means clustering and P-nearest neighbor methods given in Subsection III.B.1. *Second,* gradient descent method, given in Subsection III.B.2, is used to improve values of the basis function centers, widths and to find output layer weights.

2) As a result obtained during the training RBF network for identification using gradient descent method, the control signal can be selected such that $\hat{y}(k+1)$ is made as close as possible to the model output $y_m(k+1)$. For this purpose, we define an objective function J_c as follows:

$$J_c=\frac{1}{2}e_c^2(k+1) \tag{17}$$

where

$$e_c(k+1)=y_m(k+1)-\hat{y}(k+1) \tag{18}$$

The control signal $u(k)$ should therefore be selected to minimize J_c. Using the RBF neural network structure, (15) can be rewritten to give

$$\hat{y}(k+1) = w_0 + \sum_{j=1}^{m} w_j \exp\left(-\frac{X - C_j}{2\sigma_j^2}\right) \tag{19}$$

where

$$X = (y(k), y(k\text{-}1), \dots, y(k\text{-}\mathrm{L}+1), u(k), \\ u(k\text{-}1), \dots, u(k\text{-}\mathrm{L}+1))^{\mathrm{T}} \tag{20}$$

of dimension $2L$.

To minimize J_c, the $u(k)$ is calculated using gradient descent rule

$$u(k+1) = u(k) - \eta_c \frac{\partial J_c}{\partial u(k)} \tag{21}$$

where $\eta_c > 0$ is a learning rate. It can be seen that the controller relies on the approximation made by the RBF neural network. Therefore it is necessary that $\hat{y}(k+1)$ approaches the real system output $y_p(k+1)$ asymptotically. This can be achieved by keeping the RBF neural network training online. Differentiating (17) with respect to $u(k)$, it can be obtained that

$$u(k+1) = u(k) + \eta_c e_c(k+1) \frac{\partial \hat{y}(k+1)}{\partial u(k)} \tag{22}$$

The gradient can then be analytically evaluated by using the known RBF neural network structure (19) as follows:

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{j=1}^{m} w_j \,.\exp\left(-\frac{X - C_j}{2\sigma_j^2}\right).(-\frac{u(\mathrm{k}) - c_{\mathrm{j(L+1)}}}{\sigma_{\mathrm{j}}^2}) \tag{23}$$

Finally, (22) becomes

$$u(k+1) = u(k) + \eta_c \,.\, e_c(k+1)\,. \\ \sum_{j=1}^{m} w_j \,.\exp\left(-\frac{X - C_j}{2\sigma_j^2}\right).(-\frac{u(\mathrm{k}) - c_{\mathrm{j(L+1)}}}{\sigma_{\mathrm{j}}^2}) \tag{24}$$

The summary of the SISO algorithm is as follows:

1) Find the initial values of the centers and widths of the radial basis functions using K-means clustering algorithm and P-nearest neighbor method given in Section III.B.1.
2) Give an initial value for the control input $\mathrm{u(k)}$ equal to zero.
3) Produce $\hat{y}(k+1)$ using (19);
4) Find $e_c(k+1)$ using (18);
5) Update the values of centers and widths of the hidden layer basis functions and output layer weights from (12), (13) and (14);
6) Compute new control signal from (24);
7) Feed $u(t+1)$ to the system;
8) Go to step 3).

B. Direct Controller Algorithm for MIMO System

The structure given by Fig. 3, is used for direct controller of the unknown nonlinear MIMO system given by (2) in case of $n = 2$. The following steps describe the control process:

1) The unknown nonlinear system (2) can be expressed by the following model equation

$$\hat{\mathrm{y}}_1(\mathrm{k}+1) = \hat{f}_1(y_{\mathrm{p1}}(k), \dots, y_{\mathrm{p1}}(k\text{-}L+1), \dots, y_{\mathrm{pn}}(k), \dots, y_{\mathrm{pn}}(k\text{-}L+1), \\ u_1(k), \dots \mathrm{u}_1(k\text{-}\mathrm{L}+1))$$
$$\hat{\mathrm{y}}_2(\mathrm{k}+1) = \hat{f}_2(y_{\mathrm{p1}}(k), \dots, y_{\mathrm{p1}}(k\text{-}L+1), \dots, y_{\mathrm{pn}}(k), \dots, y_{\mathrm{pn}}(k\text{-}L+1), \\ u_2(k), \dots \mathrm{u}_2(k\text{-}\mathrm{L}+1))$$
$$\dots \qquad \dots \qquad \dots$$
$$\hat{\mathrm{y}}_{\mathrm{n}}(\mathrm{k}+1) = \hat{f}_n(y_{\mathrm{p1}}(k), \dots, y_{\mathrm{p1}}(k\text{-}L+1), \dots, y_{\mathrm{pn}}(k), \dots, y_{\mathrm{pn}}(k\text{-}L+1), \\ u_n(k), \dots \mathrm{u}_{\mathrm{n}}(k\text{-}\mathrm{L}+1)) \tag{25}$$

using n RBF neural networks, where $\hat{y}_1(k+1), \hat{y}_2(k+1), \dots, \hat{y}_{\mathrm{n}}(k+1)$ are the outputs of the n RBF neural networks given by

$$\hat{y}_1(k+1) = w_0^{(1)} + \sum_{j=1}^{m} w_j^{(1)} h_j^{(1)}$$
$$\hat{y}_2(k+1) = w_0^{(2)} + \sum_{j=1}^{m} w_j^{(2)} h_j^{(2)}$$
$$\dots \qquad \dots$$
$$\hat{y}_n(k+1) = w_0^{(n)} + \sum_{j=1}^{m} w_j^{(n)} h_j^{(n)} \tag{26}$$

and $\hat{f}_1, \hat{f}_2, \dots, \hat{f}_{\mathrm{n}}$ are the estimate of

$$f_1[y_{p1}(k), \dots, y_{p1}(k\text{-}L+1), \dots, y_{p\mathrm{n}}(k), \dots, y_{p\mathrm{n}}(k\text{-}L+1)] + \sum_{j=0}^{L-1} b_{1j} u_1(k\text{-}j),$$
$$f_2[y_{p1}(k), \dots, y_{p1}(k\text{-}L+1), \dots, y_{p\mathrm{n}}(k), \dots, y_{p\mathrm{n}}(k\text{-}L+1)] + \sum_{j=0}^{L-1} b_{2j} u_2(k\text{-}j),$$
$$\dots, f_n[y_{p1}(k), \dots, y_{p1}(k\text{-}L+1), \dots, y_{p\mathrm{n}}(k), \dots, y_{p\mathrm{n}}(k\text{-}L+1)] + \sum_{j=0}^{L-1} b_{nj} u_n(k\text{-}j).$$

Training the n RBF networks must guarantee that

$$[y_{p1}(k+1) - \hat{y}_1(k+1)]^2 = \min,$$
$$[y_{p2}(k+1) - \hat{y}_2(k+1)]^2 = \min,$$
$$\dots \quad \dots \quad \dots$$
$$[y_{pn}(k+1) - \hat{y}_n(k+1)]^2 = \min. \tag{27}$$

Training each of the n RBF neural networks can be done with two separate phases: *First,* the initial values for the location of centers and widths are obtained using unsupervised K-means clustering and P-nearest neighbor methods given in Subsection III.B.1. *Second,* gradient descent method, given in Subsection III.B.2, is used to improve values of the basis function centers, widths and to find the output layer weights.

2) As a result obtained during the training n RBF networks, the control signals can be selected such that

$\hat{y}_1(k+1), \hat{y}_2(k+1), \ldots, \hat{y}_{\mathrm{n}}(k+1)$ are made as close as possible to the model outputs $y_{m1}(k+1), y_{m2}(k+1), \ldots, y_{m\mathrm{n}}(k+1)$. For this purpose, we define n objective functions $J_{c1}, J_{c2}, \ldots, J_{\mathrm{cn}}$ as follows:

$$\begin{aligned} J_{c1} &= \frac{1}{2} e_{c1}^2(k+1) \\ J_{c2} &= \frac{1}{2} e_{c2}^2(k+1) \\ &\cdots \\ J_{cn} &= \frac{1}{2} e_{cn}^2(k+1). \end{aligned} \tag{28}$$

where

$$\begin{aligned} e_{c1}(k+1) &= y_{m1}(k+1) - \hat{y}_1(k+1) \\ e_{c2}(k+1) &= y_{m2}(k+1) - \hat{y}_2(k+1) \\ \cdots \quad & \quad \cdots \\ e_{cn}(k+1) &= y_{mn}(k+1) - \hat{y}_n(k+1) \end{aligned} \tag{29}$$

The control signals $u_1(k), u_2(k), \ldots, u_{\mathrm{n}}(k)$ should therefore be selected to minimize $J_{c1}, J_{c2}, \ldots, J_{\mathrm{cn}}$. Using n RBF neural networks structure, (25) can be rewritten to give

$$\begin{aligned} \hat{y}_1(k+1) &= w_0^{(1)} + \sum_{j=1}^{m} w_j^{(1)} \exp\left(-\frac{X^{(1)} - C_j^{(1)}}{2\sigma_{(1)j}^2} \right) \\ \hat{y}_2(k+1) &= w_0^{(2)} + \sum_{j=1}^{m} w_j^{(2)} \exp\left(-\frac{X^{(2)} - C_j^{(2)}}{2\sigma_{(1)j}^2} \right) \\ \cdots \quad & \quad \cdots \quad \cdots \\ \hat{y}_n(k+1) &= w_0^{(n)} + \sum_{j=1}^{m} w_j^{(n)} \exp\left(-\frac{X^{(n)} - C_j^{(n)}}{2\sigma_{(n)j}^2} \right) \end{aligned} \tag{30}$$

where

$$\begin{aligned} X^{(1)} &= (y_1(k), \ldots, y_1(k-L+1), \ldots, y_{\mathrm{n}}(k), \ldots, y_{\mathrm{n}}(k-L+1), \\ &\quad u_1(k), \ldots u_1(\mathrm{k-L+1}))^{\mathrm{T}} \\ X^{(2)} &= (y_1(k), \ldots, y_1(k-L+1), \ldots, y_{\mathrm{n}}(k), \ldots, y_{\mathrm{n}}(k-L+1), \\ &\quad u_2(k), \ldots u_2(\mathrm{k-L+1}))^{\mathrm{T}} \\ &\cdots \quad \cdots \quad \cdots \\ X^{(n)} &= (y_1(k), \ldots, y_1(k-L+1), \ldots, y_{\mathrm{n}}(k), \ldots, y_{\mathrm{n}}(k-L+1), \\ &\quad u_n(k), \ldots u_{\mathrm{n}}(\mathrm{k-L+1}))^{\mathrm{T}} \end{aligned} \tag{31}$$

each of dimension $(n+1)L$.

To minimize $J_{c1}, J_{c2}, \ldots, J_{\mathrm{cn}}$, the $u_1(k), u_2(k), \ldots, u_{\mathrm{n}}(k)$ are calculated using gradient descent rule

$$\begin{aligned} u_1(k+1) &= u_1(k) - \eta_1 \frac{\partial J_{c1}}{\partial u_1(k)} \\ u_2(k+1) &= u_2(k) - \eta_2 \frac{\partial J_{c2}}{\partial u_2(k)} \\ \cdots \quad & \quad \cdots \\ u_n(k+1) &= u_n(k) - \eta_n \frac{\partial J_{cn}}{\partial u_n(k)} \end{aligned} \tag{32}$$

where $\eta_1, \eta_2, \ldots, \eta_n > 0$ are the learning rates. It can be seen that the controllers depends on the approximation made by the n RBF neural networks. Therefore it is necessary that $\hat{y}_1(k+1), \hat{y}_2(k+1), \ldots, \hat{y}_{\mathrm{n}}(k+1)$ approach the real system outputs $y_{p1}(k+1), y_{p2}(k+1), \ldots, y_{p\mathrm{n}}(k+1)$ asymptotically. This can be achieved by keeping the n RBF neural networks training online. Differentiating $J_{c1}, J_{c2}, \ldots, J_{\mathrm{cn}}$ in (28) with respect to $u_1(k), u_2(k), \ldots, u_{\mathrm{n}}(k)$ respectively, it can be obtained that

$$\begin{aligned} u_1(k+1) &= u_1(k) + \eta_1 e_{c1}(k+1) \frac{\partial \hat{y}_1(k+1)}{\partial u_1(k)} \\ u_2(k+1) &= u_2(k) + \eta_2 e_{c2}(k+1) \frac{\partial \hat{y}_2(k+1)}{\partial u_2(k)} \\ \cdots \quad & \quad \cdots \quad \cdots \\ u_n(k+1) &= u_n(k) + \eta_n e_{cn}(k+1) \frac{\partial \hat{y}_n(k+1)}{\partial u_n(k)} \end{aligned} \tag{33}$$

The gradients can then be analytically evaluated by using the known *n* RBF neural networks structure (30) as follows:

$$\begin{aligned} \frac{\partial \hat{y}_1(k+1)}{\partial u_1(k)} &= \sum_{j=1}^{m} w_j^{(1)} . \exp\left(-\frac{X^{(1)} - C_j^{(1)}}{2\sigma_{(1)j}^2} \right) . (-\frac{u_1(k) - c_{\mathrm{j(nL+1)}}^{(1)}}{\sigma_{(1)\mathrm{j}}^2}) \\ \frac{\partial \hat{y}_2(k+1)}{\partial u_2(k)} &= \sum_{j=1}^{m} w_j^{(2)} . \exp\left(-\frac{X^{(2)} - C_j^{(2)}}{2\sigma_{(2)j}^2} \right) . (-\frac{u_2(k) - c_{\mathrm{j(nL+1)}}^{(2)}}{\sigma_{(2)\mathrm{j}}^2}) \\ \cdots \quad & \quad \cdots \quad \cdots \\ \frac{\partial \hat{y}_n(k+1)}{\partial u_n(k)} &= \sum_{j=1}^{m} w_j^{(n)} . \exp\left(-\frac{X^{(n)} - C_j^{(n)}}{2\sigma_{(n)j}^2} \right) . (-\frac{u_{\mathrm{n}}(k) - c_{\mathrm{j(nL+1)}}^{(\mathrm{n})}}{\sigma_{(\mathrm{n})\mathrm{j}}^2}) \end{aligned} \tag{34}$$

Finally, (33) becomes

$$\begin{aligned} \mathrm{u}_1(\mathrm{k}+1) &= \mathrm{u}_1(\mathrm{k}) + \eta_1 . \mathrm{e}_{\mathrm{c}1}(\mathrm{k}+1) . \\ &\sum_{\mathrm{j}=1}^{\mathrm{m}} \mathrm{w}_{\mathrm{j}}^{(1)} . \exp\left(-\frac{\mathrm{X}^{(1)} - \mathrm{C}_{\mathrm{j}}^{(1)}}{2\sigma_{(1)\mathrm{j}}^2} \right) . (-\frac{\mathrm{u}_1(\mathrm{k}) - \mathrm{c}_{\mathrm{j(nL+1)}}^{(1)}}{\sigma_{(1)\mathrm{j}}^2}) \\ \mathrm{u}_2(\mathrm{k}+1) &= \mathrm{u}_2(\mathrm{k}) + \eta_2 . \mathrm{e}_{\mathrm{c}2}(\mathrm{k}+1) . \\ &\sum_{\mathrm{j}=1}^{\mathrm{m}} \mathrm{w}_{\mathrm{j}}^{(2)} . \exp\left(-\frac{\mathrm{X}^{(2)} - \mathrm{C}_{\mathrm{j}}^{(2)}}{2\sigma_{(2)\mathrm{j}}^2} \right) . (-\frac{\mathrm{u}_2(\mathrm{k}) - \mathrm{c}_{\mathrm{j(nL+1)}}^{(2)}}{\sigma_{(2)\mathrm{j}}^2}) \\ \cdots \quad & \quad \cdots \quad \cdots \\ \mathrm{u}_{\mathrm{n}}(\mathrm{k}+1) &= \mathrm{u}_{\mathrm{n}}(\mathrm{k}) + \eta_{\mathrm{n}} . \mathrm{e}_{\mathrm{cn}}(\mathrm{k}+1) . \\ &\sum_{\mathrm{j}=1}^{\mathrm{m}} \mathrm{w}_{\mathrm{j}}^{(\mathrm{n})} . \exp\left(-\frac{\mathrm{X}^{(\mathrm{n})} - \mathrm{C}_{\mathrm{j}}^{(\mathrm{n})}}{2\sigma_{(\mathrm{n})\mathrm{j}}^2} \right) . (-\frac{\mathrm{u}_{\mathrm{n}}(\mathrm{k}) - \mathrm{c}_{\mathrm{j(nL+1)}}^{(\mathrm{n})}}{\sigma_{(\mathrm{n})\mathrm{j}}^2}) \end{aligned} \tag{35}$$

The summary of the MIMO algorithm is as follows:

1) Find the initial values of the centers and widths of the radial basis functions of the n RBF networks using K-means clustering algorithm and P-nearest neighbor method given in Section III.B.1.
2) Give initial values for the control inputs $u_1(k), u_2(k), \ldots, u_n(k)$ equal to zero.
3) Generate $\hat{y}_1(k+1), \hat{y}_2(k+1), \ldots, \hat{y}_{\mathrm{n}}(k+1)$ using (25);

4) Find $e_{c1}(k+1), e_{c2}(k+1), \ldots, e_{cn}(k+1)$ using (29);
5) Update the values of centers and widths of the hidden layer basis functions and output layer weights from (12), (13) and (14);
6) Compute new control signal from (35);
7) Feed $u_1(k+1), u_2(k+1), \ldots, u_n(k+1)$ to the system;
8) Go to step 3).

VI. Simulation Results

The performance of the proposed controller is demonstrated through simulation results. Two nonlinear systems have been taken for this purpose. The first example considered here is a SISO system while the second example is a MIMO one.

A. Example 1

The first example taken for the simulation is same as that of [4]. The dynamics of the unknown nonlinear SISO system to be considered is given by the following equation:

$$y_p(k+1) = \frac{5y_p(k)y_p(k-1)}{1+y_p^2(k)+y_p^2(k-1)+y_p^2(k-2)} + u(k) + 0.8u(k-1) \tag{36}$$

and the control problem is to find a direct control signal, $u(k)$, such that the output of the system (36), $y_p(k+1)$, is made as close as possible to the stable reference model, $y_m(k+1)$, described by

$$y_m(k+1) = 0.32y_m(k) + 0.64y_m(k-1) - 0.5y_m(k-2) + r(k) \tag{37}$$

where r(k) is the uniform bounded reference input given by

$$r(k) = \sin\left(\frac{2\pi k}{25}\right) \tag{38}$$

To solve direct control problem we follow the steps given in subsection V.A., using the architecture given in fig. 2, where the unknown nonlinear system (36) is expressed by the model equation

$$\hat{y}(k+1) = \hat{f}(y_p(k), y_p(k-1), y_p(k-2), u(k), u(k-1)) \tag{39}$$

using only one RBF neural network with input vector $(y(k), y(k-1), y(k-2), u(k), u(k-1))$, 25 hidden neurons and one output neuron. Training the RBF network can be obtained in two phases for control purpose:

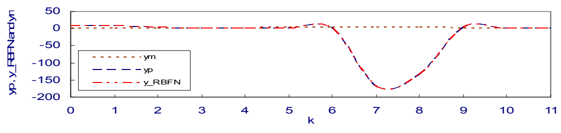
First, the initial values for the location of centers and widths are obtained using unsupervised K-means clustering and P-nearest neighbor, given in Subsection III.B.1., using an input which is random and distributed uniformly over the interval $[-2, 2]$. *Second,* gradient descent method, given in Subsection III.B.2, is used to find output layer weights initialized to very small values and to improve values of the centers and widths of the basis functions. The corresponding modifier formulas for output layer weights, centers and widths in our example are that given by (12), (13) and (14) with a learning rate value $\eta = 0.005$. During the training RBF network using gradient descent method to make $\hat{y}(k+1)$ approaches the real system output $y_p(k+1)$ of (36), the control signal can be selected such that $\hat{y}(k+1)$ is made as close as possible to the model output $y_m(k+1)$ of (37). The corresponding modifier formulas for the control signal is that given by (24) with a learning rate value $\eta_c = 0.002$.

The performance of the direct controller algorithm at the first eleven steps is shown in Fig. 4(a) which is unsatisfactory because the system tries to adjust its parameters for direct control. Fig. 4(b) shows the performance from step twelve to ninety-nine which gives satisfactory output tracking. Fig. 5 (a) and (b) give the identification error and control error respectively which show that a large error at the first eleven steps and from step eleven to ninety-nine tends to zero. Taking the results from step eleven to ninety-nine, RMS error for the controller is 0.005 and that for identification is 2.2×10^{-6}. Fig. 6 gives the required control signal from step eleven to ninety-nine.

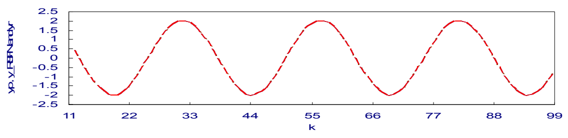
To demonstrate the effectiveness of the direct controller algorithm we take another case with the system (36). We will change the reference input given in (38) to be

$$r(k) = \sin\left(\frac{2\pi k}{25}\right) + \sin\left(\frac{2\pi k}{10}\right) \tag{40}$$

Tracking results in this case are given in fig. 7. Identification and control errors are shown in fig. 8. Taking the calculations from step eleven to ninety-nine, the RMS error for the controller is 0.02 and that for identification is 2.5×10^{-5}. The control input in this case is given in fig. 9.

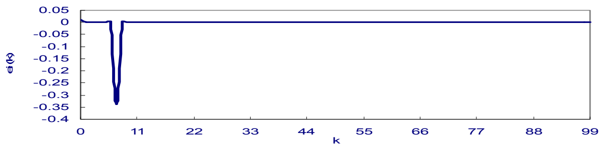


(a)

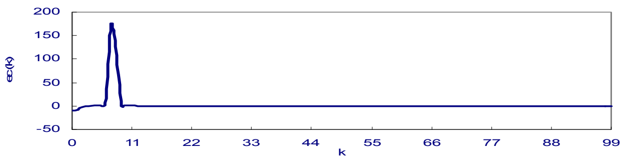


(b)

Fig. 4. Performance of the RBFN Direct controller algorithm for the SISO system given in example 1 using the structure of Fig. 2 with $r(k) = \sin(2\pi k/25)$. (a) gives the performance at the first eleven steps. (b) gives the performance from twelve to ninety-nine.



(a)



(b)

Fig. 5. (a) gives the identification error between system output, and RBFN output. (b) gives the control error between output of the RBFN and that of the model, $y_m(k+1)$.

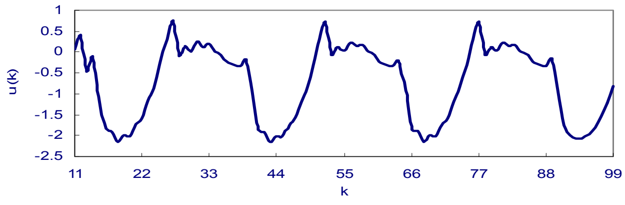
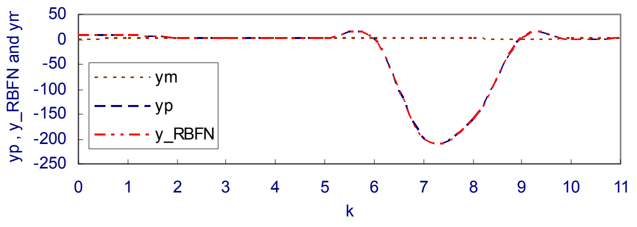
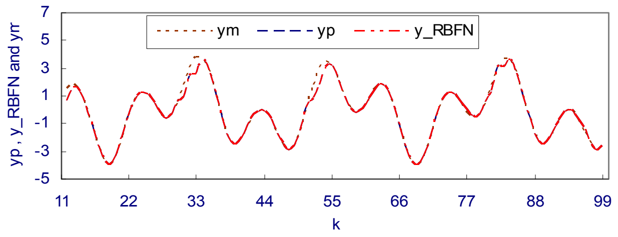


Fig. 6. Control signal for the system (36) in case of reference model (37) with $r(k)=\sin(2\pi k/25)$.

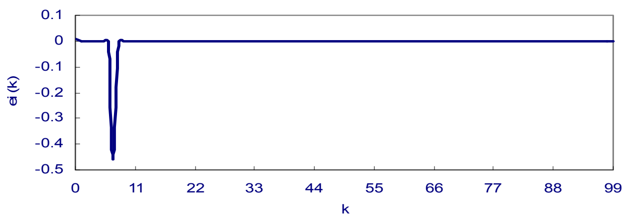


(a)

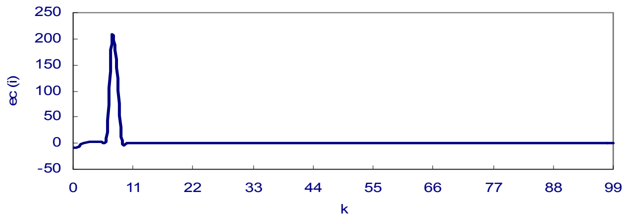


(b)

Fig. 7. Performance of the RBFN Direct controller algorithm for the SISO system (36) using the structure of Fig. 2 with $r(k)=\sin(2\pi k/25)+\sin(2\pi k/10)$. (a) gives the performance at the first eleven steps. (b) gives the performance from twelve to ninety-nine.



(a)



(b)

Fig. 8. (a) gives the identification error between system output, and RBFN output. (b) gives the control error between output of the RBFN and that of the model, $y_m(k+1)$.

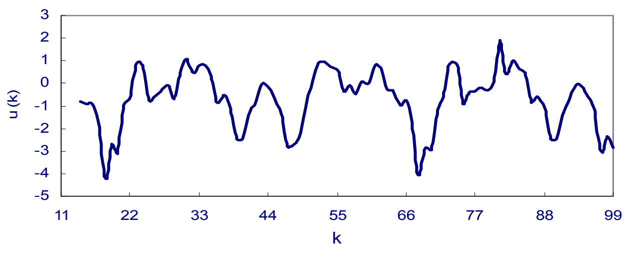


Fig. 9. Control signal for the system (36) in case of reference model (37) with $r(k)=\sin(2\pi k/25)+\sin(2\pi k/10)$.

B. Example 2

The second example taken for the simulation is same as that of [4]. The dynamics of the unknown nonlinear MIMO system to be considered is given by the following equation:

$$\begin{aligned} y_{p1}(k+1) &= \frac{y_{p1}(k)}{1+y_{p2}^2(k)}+u_1(k) \\ y_{p2}(k+1) &= \frac{y_{p1}(k)y_{p2}(k)}{1+y_{p2}^2(k)}+u_2(k) \end{aligned} \tag{41}$$

and the control problem is to find a direct control signals, $u_1(k),u_2(k)$, such that the outputs of the system (41), $y_{p1}(k+1),y_{p2}(k+1)$, are made as close as possible to the stable reference models, $y_{m1}(k+1),\mathrm{y}_{m2}(k+1)$, described by

$$\begin{aligned} y_{m1}(k+1) &= 0.6y_{m1}(k)+0.2y_{m2}(k)+r_1(k) \\ y_{m2}(k+1) &= 0.1y_{m1}(k)-0.8y_{m2}(k)+r_2(k) \end{aligned} \tag{42}$$

where $r_1(k),r_2(k)$ are the uniform bounded reference inputs given by

$$\begin{aligned} r_1(k) &= \sin\left(\frac{2\pi k}{25}\right) \\ r_2(k) &= \cos\left(\frac{2\pi k}{25}\right) \end{aligned} \tag{43}$$

To solve direct control problem we follow the steps given in subsection V.B., using the architecture given in fig.3, where the unknown nonlinear system (41) is expressed by the model equation

$$\begin{aligned} &\hat{y}_1(k \quad 1) \quad \hat{f}_1(y_{p1}(k),y_{p2}(k),\mathrm{u}_1(k)) \\ &\hat{y}_2(k+1)=\hat{f}_2(y_{p1}(k),y_{p2}(k),\mathrm{u}_2(k)) \end{aligned} \tag{44}$$

using two RBF neural networks with input vectors $(y_{p1}(k),y_{p2}(\mathrm{k}),u_1(\mathrm{k}))$ for the first RBF network and $(y_{p1}(k),y_{p2}(\mathrm{k}),u_2(\mathrm{k}))$ for the second RBF network, 35 hidden neurons for each one and one output neuron for each one. Training the two RBF networks can be obtained in two phases for control purpose:

First, the initial values for the location of centers and widths for the two RBF networks are obtained using unsupervised K-means clustering and P-nearest neighbor, given in Subsection III.B.1., with random inputs $u_1(k)$ and $u_2(k)$ uniformly distributed over the interval $[-1,1]$. *Second,* gradient descent method, given in subsection III.B.2, is used to find values of output layer weights and to improve values of centers and widths of the basis functions of the two RBF networks. The corresponding modifier formulas for output layer weights, centers and widths in our example are that given by (12), (13) and (14) with a learning rate value $\eta=0.005$ for both RBF networks. During the training RBF networks using gradient descent method to make $\hat{y}_1(k+1)$ and $\hat{y}_2(k+1)$ approaches the real system outputs $y_{p1}(k+1)$ and $y_{p2}(k+1)$ of (41), the control signals $u_1(k)$ and $u_2(k)$ can be selected such that $\hat{y}_1(k+1)$ and $\hat{y}_2(k+1)$ are made as close as possible to the model outputs $y_{m1}(k+1)$ and $y_{m1}(k+1)$ of (41). The corresponding modifier formulas for the control signals are that given by (35) with a learning rate values $\eta_1, \eta_2 = 0.05$.

The performance of the direct controller algorithm at the first 30 steps is shown in Fig. 10(a) and (c) which is unsatisfactory because the system tries to adjust its parameters for control. Fig. 10(b) and (d) shows the performance from step 30 to 150 which gives satisfactory output tracking. Fig. 11 (a) and (b) give the identification error and Fig. 11 (c) and (d) give the control error. Taking the results from step 30 to 150, RMS error for ec1 is 0.021 and 0.024 for ec2 . The identification error is 2.2×10^{-15} for $ei1$ and 5.1×10^{-15} for $ei2$. Fig. 12 gives the required control signals from step 30 to 150.

VII. Comparison Study

In this section, we will compare the direct control algorithm for two cases:

1) when the RBFN training is initialized by unsupervised clustering methods (K-means clustering algorithm and P-nearest neighbor method) and improved with gradient descent method.
2) when the RBFN is trained using gradient descent method without using unsupervised clustering methods (K-means clustering algorithm and P-nearest neighbor method).

When RBFN is trained using gradient descent method without using unsupervised clustering methods we get the following results:

3) In case of example 1 with $r(k)=\sin(2\pi k/25)$, the RMS error for the controller is 0.114 and that for identification is 0.0003.
4) In case of example 1 with $r(k)=\sin(2\pi k/25)+\sin(2\pi k/10)$, the RMS error for the controller is 0.8 and that for identification is 0.009.
5) In case of example 2 with RMS error for $ec1$ is 0.6 and 0.05 for $ec2$. The identification error is 0.005 for $ei1$ and 0.0004 for $ei2$. Fig. 12 gives the required control signals from step 30 to 150.

In comparison of the two cases mentioned above, it was found that direct control algorithm with RBFN trained using gradient descent method with unsupervised clustering methods improves the learning process than when unsupervised clustering methods are not used.

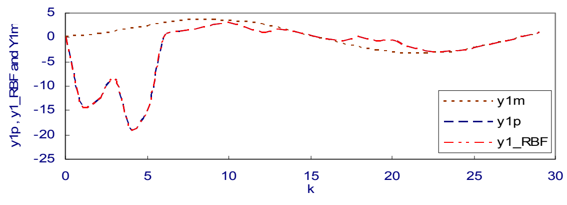
VIII. Conclusion

The RBF networks can be used as a base to implement nonlinear model-based direct controllers. The approach given in this paper can identify and control nonlinear plants in real time.

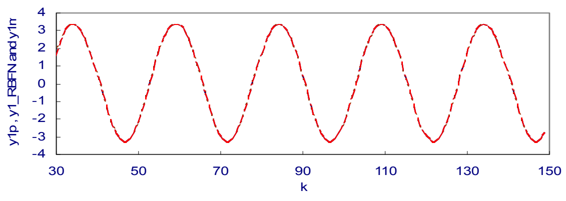
Two nonlinear systems have been taken for simulation study. The first one is a SISO system which is same as that of [4]. The second example represents a MIMO system which is the same as that of [4] also. Simulation results have shown that the direct controller method can drive the system outputs to the desired reference with a satisfactory performance.

The comparison of the direct control algorithm in two cases, *first* when the RBFN is trained initialized by unsupervised clustering methods (K-means clustering algorithm and P-nearest neighbor method) and improved with gradient descent method *second* when the RBFN is trained using gradient descent method without using unsupervised clustering methods (K-means clustering algorithm and P-nearest neighbor method), have showed that direct control algorithm with RBFN trained using gradient descent method with unsupervised clustering methods improve the learning process than that when unsupervised clustering methods are not used.

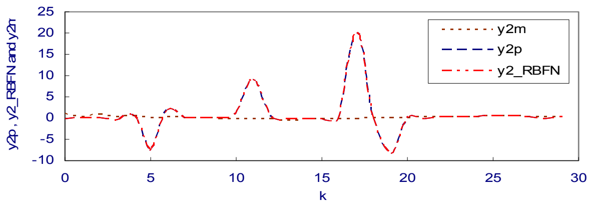
Finally, RBF neural network can provide a good solution for a wide range of adaptive real-time control problems.



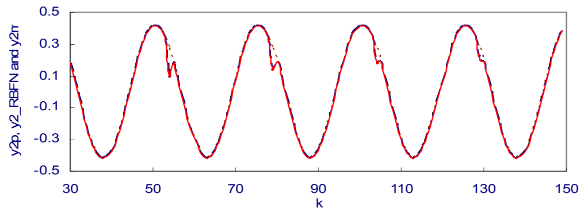
(a)



(b)

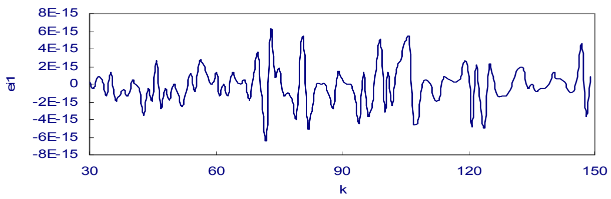


(c)

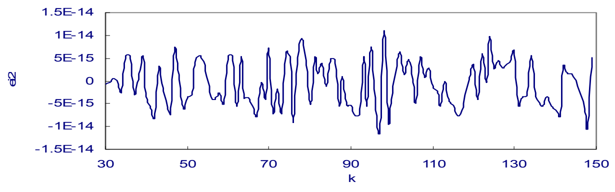


(d)

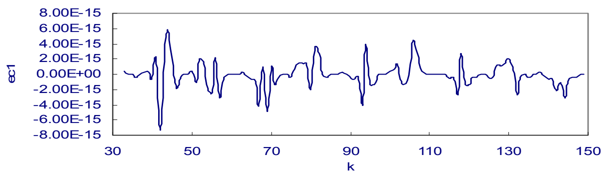
Fig. 10. Performance of the RBFN Direct controller algorithm for the MIMO system given in example 2 using the structure of Fig. 3 with $r_1(k)=\sin(2\pi k/25)$ and $r_2(k)=\cos(2\pi k/25)$. (a) gives the performance at the first 30 steps. (b) gives the performance from 30 to 150.



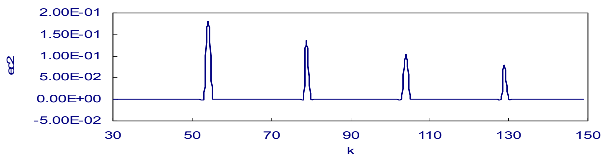
(a)



(b)

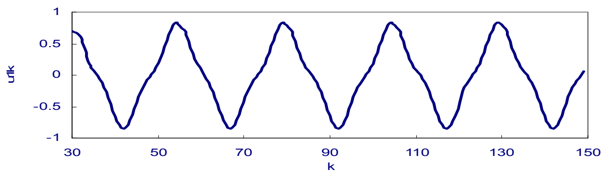


(c)

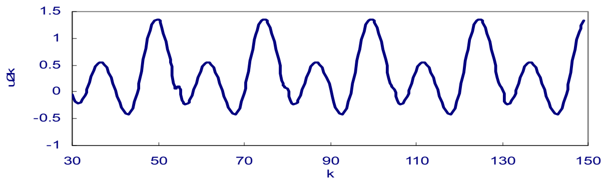


(d)

Fig. 11. (a) and (b) give the identification errors between real system outputs $y_{p1}(k+1)$ and $y_{p2}(k+1)$ and that of the two RBFN outputs $\hat{y}_1(k+1)$ and $\hat{y}_2(k+1)$. (c) and (d) give the control errors between two RBFNs outputs and that of the model outputs $y_{m1}(k+1)$ and $y_{m1}(k+1)$.



(a)



(b)

Fig. 12. (a) and (b) give control signals for the system (41) in case of reference model (42) with $r_1(k) = \sin(2\pi k/25)$ and $r_2(k) = \cos(2\pi k/25)$.

References

[1] A. U. Liven and K. S. Narendra, "Control of Nonlinear Dynamical Systems Using Neural Networks− Part II: Observability, Identification and control," *IEEE Trans. Neural Networks*, vol. 7, No. 1, pp. 30-42, Jan. 1996.

[2] H. Li and H. Deng, "An Approximate internal Model-based neural control for Unknown Nonlinear Discrete Processes," *IEEE Trans. Neural Networks*, vol. 17, No. 3, pp. 659-669, May. 2006.

[3] I. Douratsos and J. B. Gomm, "Neural Network Based Model Reference Adaptive Control For Processes With Time Delay," *Int. Journal of Information and system Sciences*, vol. 3, No. 1, pp. 161-179, 2007.

[4] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, No. 1, pp. 4-27, Mar. 1990.

[5] K. S. Narendra and S. Mukhopadhyay, "Adaptive control of Nonlinear Multivariable Using neural networks," *IEEE Trans. Neural Networks*, vol. 7, No. 5, pp. 737-752, 1994.

[6] K. S. Narendra and S. Mukhopadhyay, "Adaptive control using neural networks and Approximate Models," *IEEE Trans. Neural Networks*, vol. 8, No. 3, pp. 475-485, May. 1997.

[7] B. M. Michael and J. C. Anthony, "Robust Adaptive Control of Uncertain Nonlinear Systems Using Neural Networks," *American Control Conference, Albuquerque, New Mexico*, June 1997.

[8] S. G. Fabri et. al, "Functional Adaptive Control – An Intelligent Systems Approach," *Springer-verlag*, 2001.

[9] R. M. Murray et. al, "Future directions in control in an information-rich world," *IEEE Control Syst. Mage*, vol. 32, No. 2, 2003.

[10] T. Hachino, Hasuka, and H. Takata, "On-line Identification of Continuous-time Nonlinear Systems via RBF Network Model," *SICE Annual Conference in Kyusy*u, Japan, 2001.

[11] T. Kanungo et. Al., "The Analysis of a Simple k-means Clustering Algorithm," *ACM symposium on Computational Geometry, Hong Kong*, 2000.

Tamer A. Alzohairy received the B. Sc. Degree in computer science from faculty of Science, Ain Shams University, Cairo, Egypt in 1989. M.Sc degree in Computer science from Math. Dept., Faculty of science, Al-Menoufia university, Egypt, in 1997 and Ph.D in computer science from math. Dept. Faculty of science, Suez Canal University, Egypt in 2003.

He is working currently in Computer science Dept.,Community college, King Saud University. Saudi Arabia on leave from Math. Dept., faculty of science, Al-Azhar university, Cairo, Egypt..