Arabian Journal for Science and Engineering (2018) 43:2971–2993
https://doi.org/10.1007/s13369-017-3034-9

RESEARCH ARTICLE - ELECTRICAL ENGINEERING

# Comparative Study of Neural Networks for Control of Nonlinear Dynamical Systems with Lyapunov Stability-Based Adaptive Learning Rates

Rajesh Kumar[1] · Smriti Srivastava[1] · J. R. P. Gupta[1]

Received: 31 May 2017 / Accepted: 21 December 2017 / Published online: 9 January 2018

**Abstract**
This paper performs the comparative study of two feed-forward neural networks: radial basis function network (RBFN), multilayer feed-forward neural network (MLFFNN) and a recurrent neural network: nonlinear auto-regressive with exogenous inputs (NARX) neural network for their ability to provide an adaptive control of nonlinear systems. Dynamic back-propagation algorithm is used to derive parameter update equations. To ensure stability and faster convergence, an adaptive learning rate is developed in the sense of discrete Lyapunov stability method. Both parameter variation and disturbance signal cases are considered for checking and comparing the robustness of controller. Three simulation examples are considered for carrying out this study. The results so obtained reveal that RBFN-based controller is performing better than that of NARX- and MLFFNN-based controllers.

**Keywords** Radial basis function network · Nonlinear adaptive control · NARX · Multilayer feed-forward neural network · Robustness · Lyapunov stability-based adaptive learning rate

## 1 Introduction

### 1.1 Background

The advancement in the mathematical theory has led to the development of some powerful techniques to handle the dynamical systems. It uses well-established techniques, which are based on linear algebra, complex variable theory, and the theory of ordinary linear differential equations for analysing the dynamical systems. The procedure available to design the controller is closely related to the stability properties of the system. For LTI systems, necessary and sufficient conditions for stability are available and hence designing a controller for them is not a difficult task. In contrast to this, the stability of the nonlinear system can be established on a system-by-system basis, and hence, it is not surprising that the general design procedure which fulfils the requirement of desired dynamical response and stability is not available for the large class of nonlinear systems [1]. In reality, most of the systems are inherently nonlinear, and hence, designing a suitable controller for them is a challenging task. Further, the linearization of these systems around an equilibrium point may yield the linearly tractable mathematical models. However, it has been found that the conventionally designed linear controller may not able to achieve some stringent specifications. Also, it may not be able to provide the required performance over the variety of operating regimes, especially in the case of highly nonlinear system [2–4]. Another limitation of the conventional control methods is their dependency on the availability of systems model in designing the controller. Also, the conventional methods are not useful in deriving the control laws for the nonlinear systems since a large amount of approximations, assumptions and simplifications have to be performed which ultimately affect the desired accuracy [5,6]. Various advanced methods have been developed to control the nonlinear systems. These include sliding mode control [7,8], back-stepping control [9,10], decentralized control [11,12].

✉ Rajesh Kumar
rajeshmahindru23@gmail.com

Smriti Srivastava
smriti.nsit@gmail.com

J. R. P. Gupta
jairamprasadgupta@gmail.com

1 Division of Instrumentation and Control Engineering, Netaji Subhas Institute of Technology, Sector 3, Dwarka, New Delhi, India

But it is now well known that nonlinear systems have the tendency to show extremely complex dynamic behaviour, and hence, these advanced methods may not give the desired performance [13]. All these factors among others have contributed in the development of more sophisticated nonlinear intelligent techniques which can handle nonlinear complex processes [14]. Artificial neural network (ANN) is one such tool. ANN is the distributed processing systems, which are inspired by the biological nerve system. They contain neurons which approximates the behaviour of nerve neurons. These neurons in ANN are connected to each other with the weights. It is one of the popular computational intelligent approach which is capable of approximating any unknown nonlinear function to any degree of accuracy [13], and it can provide accurate control without requiring any a priori information regarding the dynamics of the nonlinear system [6]. There are number of books written on ANN which details its applications in identification and control of nonlinear systems. Haykin [15] presented a nice and solid foundation in ANN. Norgaard [16] discussed various approaches by which ANN can be used to identify and control the dynamical systems. Liu [17] discussed various structures of ANN and provided various applications of ANN in nonlinear system identification and control. Zilouchin with Jamshidi [18] in their book included number of articles related to applications of intelligent controllers. In structure, ANN can be classified into two types: multilayer feed-forward neural network (MLFFNN) and recurrent neural network (RNN). Both these structures have common characteristics like parallelism in their operation and nonlinear transformation which makes them an effective tool for control of nonlinear systems [3,19]. To use ANN as a controller, its parameters need to be tuned by using some parameter adjustment methods. There exists the number of methods which are available in the literature that can be used for this purpose. The most basic one is based on back-propagation method which is based on gradient descent method. Although gradient descent-based learning algorithms have been employed successfully in various control applications, they usually suffer from the problem of slow convergence [13,20]. Another approach to derive the parameter adjustment equations is by using the Lyapunov stability method [14]. The advantage of using this method is that it guarantees the overall stability of the system. In this paper, our attempt is to combine both these popular approaches so as to utilize the merits of each of them. The parameter update equations are derived using the back-propagation algorithm. Further, to speed up the convergence rate and to ensure the stability of the system, a novel adaptive learning rate is developed based on the Lyapunov stability method. In the simulation section, the performance of the controller with fixed as well as with the adaptive learning rate is shown and compared.

The main issues in the field of ANN-based control are the choice of neural network to be used as a controller, and which parameter learning algorithm is to be employed for tuning its parameters. In our paper, an attempt has been made to compare three types of neural networks: two feed-forward (MLFFNN and RBFN) and one recurrent type of neural network (NARX) as a controller. The control system configuration employed in our paper consists of ANN-based controller in cascade with the plant and the training is performed online.

## 2 Related Work

In [1], the authors have used ANN as an identification and control tool. They have derived the weight updating algorithm based on back-propagation with constant learning rate. In [21], the authors have used neural networks for the control of nonlinear systems. They have provided the method that ensures the stability of the plant around an equilibrium point. In [22], the authors have used recurrent neural network for the control of nonlinear systems. They have used back-propagation method to derive the weight update equations with fixed learning rate. In [23], the plant is described as NARMA model, and neural network-based control is implemented in order to accomplish the set-point control. In [24], recurrent neural networks have been used for providing the speed control to the nonlinear motor-drive system using a model-following control scheme. In [25], the main focus of the authors is to have the stabilized control of dynamical systems based on neural networks. The weight adjustment algorithm is derived using back-propagation algorithm. In [26], neural network-based adaptive dynamic programming scheme is used to achieve optimal control of nonlinear discrete-time affine systems. In [27], the objective is to provide adaptive control to nonlinear switched type nonlinear system using multilayer neural network. The weight update equations are derived using the Lyapunov stability method. In [28], the authors have presented a scheme based on recurrent neural network for nonlinear control of pH. In [29], the authors have proposed a fast sliding mode method, and they have used ANN to provide control to dual-motor servo system. They have also shown that their method works well in the presence of external disturbance. In [30], the authors have used ANN to control the conical tank. The weight update equations are derived using back-propagation method. In [31], the author has derived the sufficient conditions for the existence of the solution for impulsive neutral high-order Hopfield neural networks with mixed time-varying delays and leakage delays. He has also discussed about the exponential stability of the derived solution. An example is also presented to show the effectiveness of the derived results. In [32], the authors have established some sufficient conditions for the existence and global exponential stability of the pseudo almost automorphic solutions for the class of recurrent type of neural networks with time-varying coefficients and continuously distributed delays. For deriving these conditions, the authors have used the fixed point theorem and differential inequality method. In [33], the authors have considered a class of impulsive high-order neural networks with mixed delays and derived a criteria by using the fixed point theorem, Lyapunov functional method and differential inequality techniques on the existence and global exponential stability of piecewise differentiable pseudo almost periodic solution. In [34], the author has discussed in detail the special class of neural networks known as neutral impulsive shunting inhibitory cellular neural networks with time-varying coefficients and leakage delays. A detailed description about the existence, uniqueness and exponential stability of piecewise differentiable pseudo almost periodic solutions is given. The results are derived using the fixed point theorem, Lyapunov functional method and differential inequality techniques. An example is also given to show the effectiveness of the proposed method. In [35], the authors have used generalized predictive control scheme based on recurrent type of wavelet neural network and used it for identification and control of mobile robots. Further, the authors have used gradient descent method to derive the parameter update equations. In [36], the authors have used self-recurrent wavelet neural network for the identification and control of nonlinear systems and compared its performance with that of the wavelet neural network. Gradient descent and Lyapunov stability methods are used to derive the parameter update equations. In [37], the authors have proposed a control scheme for trajectory tracking control of 3-degree-of-freedom four-rotor hover vehicle. They have used RBFN as a controller. Further, Lyapunov stability method is used to derive the update equations. In [38], the authors have used the back-stepping technique with the neural networks approximation ability for tracking control for a class of high-order nonlinear systems with completely unknown nonlinearities. They have used RBFN for this purpose. In [39], the authors have used model reference adaptive controller (MRAC) scheme based on RBFN for controlling the fixed-wing unmanned aerial vehicle (UAV). In [40], the authors have used ANN for controlling the magnetic levitation system. They have used Lyapunov method to prove the stability of the overall system. In [41], the authors have used recurrent neural network for identification, modelling and control of nonlinear systems. They have used Lyapunov stability method to prove the convergence of their proposed algorithm. In [42], the authors have proposed a neural network-based adaptive dynamic programming scheme for online identification and control of nonlinear systems. They have used Lyapunov stability method to derive the weight update equations. In [43], the authors have used recurrent neural network coupled with Kalman filter for identifying the dynamic terms of the robotic manipulator. The parameter update equations are derived using gradient descent and Lyapunov stability method. In [44], the authors have used the neural network to obtain an approximate solution to the value function of the Hamilton–Jacobi–Bellman (HJB) equation. For deriving the update equations, they have used steepest descent method. In [45], the authors have proposed a model-free discrete-time neural network control scheme for doing the trajectory tracking control of nonlinear processes. The Lyapunov method is used to ensure the stability of the system. In [46], the authors have proposed a Lyapunov function-based neural network tracking strategy for single-input, single-output (SISO) discrete-time nonlinear dynamic systems. The estimator error convergence and closed-loop system stability analysis are performed by using the Lyapunov stability method.

1. *Remark 1* Most of the papers have focused on designing an ANN-based controller but have not compared its performance by using different neural networks. The other missing part is the robustness analysis. Many papers have not tested the performance of their controllers under various system uncertainties.
2. *Remark 2* Many papers have shown the final simulation results which are obtained after the training. What could have been better if the output of the plant is shown during all the phases of the training. In such case, the detailed analysis of the performance of the controller can be done.

## 2.1 Motivation

The main motivation of writing this paper is to show in detail the performance comparison of feed-forward and recurrent type of neural networks. Most of the papers in the literature have used only one type of neural network, and the comparative analysis part is missing. Further, to the best of our knowledge there exists no paper in the literature which has done in detail the robustness comparison of feed-forward and recurrent type of neural networks. Also, to the best of our knowledge no paper has shown responses of the plant during all the phases of training. This has been done in our paper. One can get an idea from these responses how the various controllers are performing. Further, many papers have employed new powerful algorithms for designing their controllers or for obtaining the parameters update equations, but many of these algorithms are quite complex which makes the analysis quite difficult and are also quite difficult to implement. In our paper, dynamic back-propagation algorithm is used for obtaining the parameter update equations, and to make these update equations more effective they are equipped with adaptive learning rates that has been derived using the Lyapunov stability method.

### 2.2 Contributions of the Paper

1. To thoroughly analyse and compare the performances of the feed-forward and recurrent type neural network controllers with fixed as well as with adaptive learning rate.
2. To test the performances of these controllers under the system's uncertainties like parameter variation and disturbance signals.
3. To discuss and compare the computational complexity of these controllers.
4. A novel adaptive learning rate is developed using the Lyapunov stability method. The advantage of using adaptive learning rate is twofold. Firstly, faster convergence of parameters is ensured. Secondly, stability of the overall system is guaranteed.

The main outline of the paper is as follows: in Sect. 3, problem statement is given. Section 4 includes the brief introduction about RBFN, MLFFNN and NARX models and their mathematical formulation. In Sect. 5, these ANNs are compared in terms of their structural complexities and the computational time. Section 6 contains the derivation of parameter adjustment equations. In Sect. 7, Lyapunov stability method is used to develop the adaptive learning rate. Section 8 includes the discussion on the adaptive control of nonlinear systems and also includes the discussion on the selection of controller's inputs. Section 9 contains the simulation study. Three dynamical systems of different complexities are considered for evaluation of the performances of the controllers. The controllers are also tested against the effects of parameter variations and disturbance signals. Section 10 includes the discussion section. In Sect. 11, conclusion of the paper is given.

## 3 Problem Statement

Let the given plant is described by following dynamical difference equation

$$\begin{aligned} Y_p(k+1) = F\left[Y_p(k), Y_p(k-1)\ldots Y_p(k-n+1)\right] \\ + G\left[u_c(k), u_c(k-1), \ldots u_c(k-m+1)\right] \end{aligned} \tag{1}$$

where $Y_p(k+1)$ is the one step ahead output of the plant and $u_c(k)$ represents the control input to the plant. Further, $F \rightarrow \Re$, $G \rightarrow \Re$ are linear or nonlinear functions and are assumed to be differentiable. The term $n$ represents the order of the plant. Let the stable reference model dynamics is described by the following difference equation

$$\begin{aligned} Y_d(k+1) = M\left[Y_d(k), Y_d(k-1)\ldots Y_d(k-a+1)\right] \\ + N\left[r(k), r(k-1), \ldots r(k-b+1)\right] \end{aligned} \tag{2}$$

where $Y_d(k+1)$ denotes the one step ahead output signal of the reference model, $r(k)$ denotes the externally applied reference input signal and $a > 0, b > 0$. Further, $M \rightarrow \Re$, $N \rightarrow \Re$ are known linear or nonlinear differentiable functions.

The objective of the adaptive control can be stated as follows: it is desired to obtain controller's parameter update equations so that output of the controller $u_c(k)$ forces the plant output to track the reference model output. Mathematically, it can be stated as: the plant with an input–output pair $Y_p(k), u_c(k)$ is given as in Eq. 1, and the stable reference model specified with its input–output pair $Y_d(k), r(k)$ is given as in Eq. 2 with reference input signal of the system $r(k)$. Then, the requirement is to update the parameters of the ANN-based controller so that it generates a control signal sequence $u_c(k)$ such that

$$\lim_{k\rightarrow\infty} |Y_d(k+1) - Y_p(k+1)| \leq \epsilon \tag{3}$$

where $\epsilon \rightarrow 0$. The error between plant output and reference model output will be used in the update equations for updating the parameters of the ANN-based controller. The mode of training will be performed online. Further, the value of learning rate will be governed by the equation that is obtained using the Lyapunov stability method. The other requirement from the control system is to compensate the effects of parameter variation and disturbance signal. The control scheme used in this paper is able to adjust the parameters of controllers in such a way so as to compensate the effects of system's uncertainties.

## 4 Preliminaries, Basic Concepts and Mathematical Formulation of ANNs

Since a lot of literature exists on NARX, RBFN and MLFFNN so in this section a brief introduction regarding their mathematical formulation is given. The structures of RBFN, MLFFNN and NARX models are shown in Figs. 1, 2 and 3, respectively.

### 4.1 RBFN Model

The input vector of RBFN at any $k$th time instant is denoted by $X(k) = \left\{x_1(k), x_2(k), \ldots x_p(k)\right\}$. Thus, there are $p$ number of inputs. The input weight vector is denoted by $W_I(k)$, and its each element is of unity value. The adjustable output weight vector, $W_o(k)$, consists of $q$ number of adjustable weights, $w_{o1}(k)$, $w_{o2}(k)$, $\ldots$ $w_{oq}(k)$, which connects the output of hidden neurons to the output neuron(s). The activation

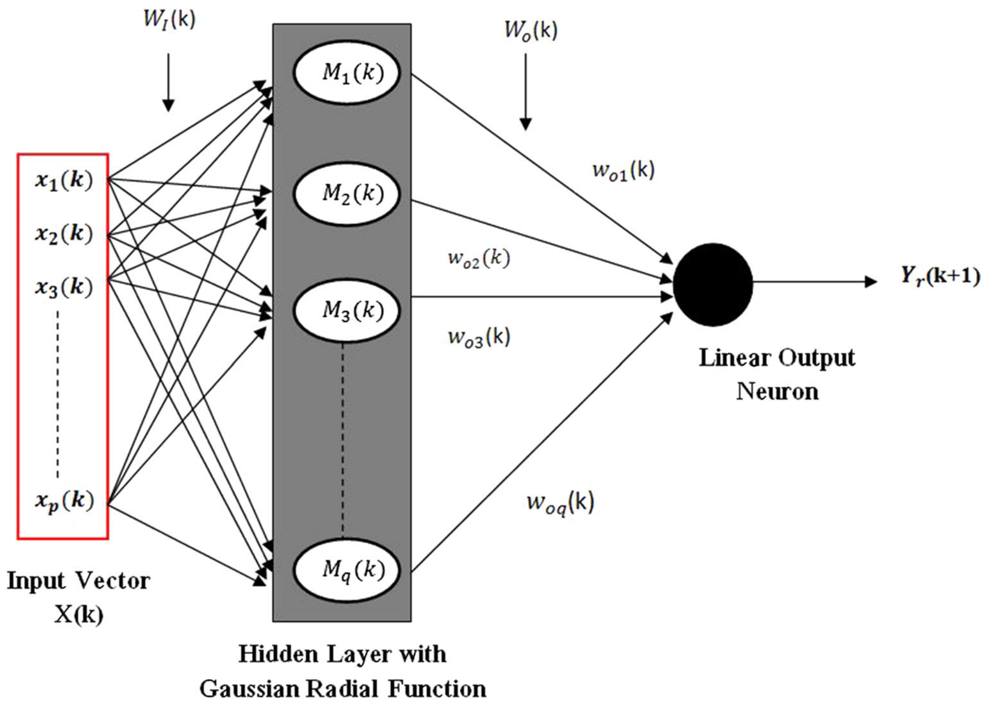
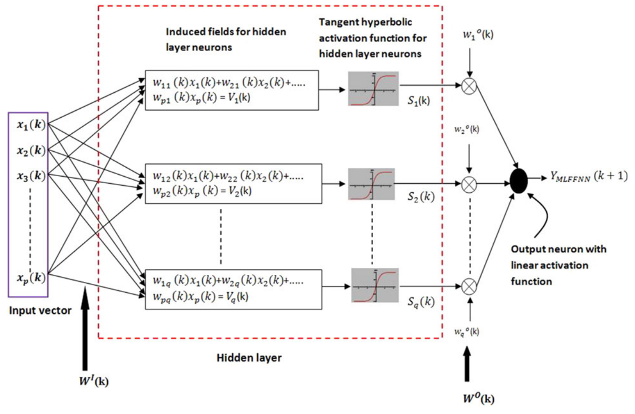


**Fig. 1** RBFN structure



Induced fields for hidden layer neurons
Tangent hyperbolic activation function for hidden layer neurons
$w_{11}(k)x_1(k)+w_{21}(k)x_2(k)+\ldots w_{p1}(k)x_p(k)=V_1(k)$
$w_{12}(k)x_1(k)+w_{22}(k)x_2(k)+\ldots w_{p2}(k)x_p(k)=V_2(k)$
$w_{1q}(k)x_1(k)+w_{2q}(k)x_2(k)+\ldots w_{pq}(k)x_p(k)=V_q(k)$
$S_1(k)$
$S_2(k)$
$S_q(k)$
$Y_{MLFFNN}(k+1)$
Output neuron with linear activation function
Input vector
Hidden layer
$W^I$(k)
$W^O$(k)

**Fig. 2** MLFFNN structure

function for output neuron(s) is/are taken to be linear so as not to restrict the output range of RBFN values. The hidden neurons of RBFN are also known as radial centres, and their functioning is quite different from that of the normal neuron (like neuron of NARX and MLFFNN). Each radial centre is assigned a coordinate having the same dimension as that of the input vector, thus, $p$ number of elements would be there in each radial centre. There are $q$ number of radial centres which are denoted by $M_1(k), M_2(k)\ldots M_q(k)$. The mathematical formulation of RBFN is as follows [47]:

$$Y_r(k)=\sum_{i=1}^{q}\phi_i(k)W_{oi}(k) \tag{4}$$

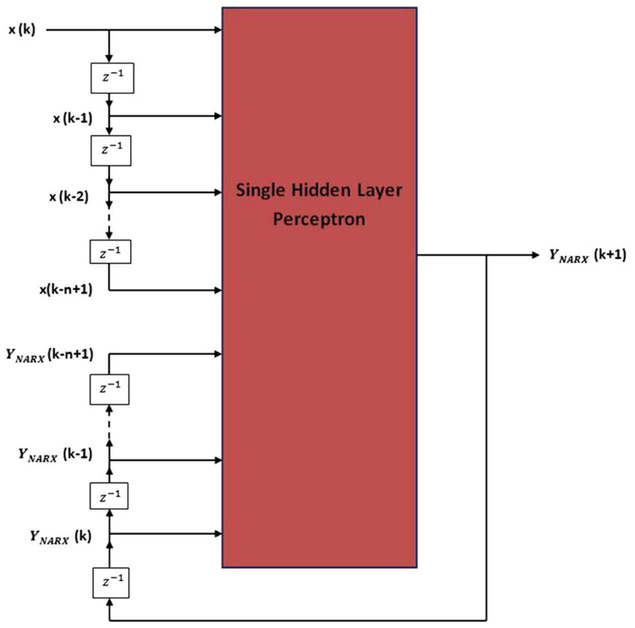


**Fig. 3** NARX structure

Here, $\phi_i(k)$ represents the Gaussian radial basis function. It can be seen from Eq. 4 that linear activation function of unity gain is used for the output neuron. Here, $Y_r(k)$ is the RBFN output at any $k$th time instant. Also, for any $i$th radial centre, its output can be found as:

$$\phi_i\left(v_i(k)\right) = \exp\left(\frac{-v_i^2(k)}{2\sigma_i^2(k)}\right) \tag{5}$$

where $v_i(k) = \|X(k) - M_i(k)\|$ is the Euclidean distance present between $X(k)$ and $M_i(k)$. The widths are represented by $\sigma(k) = \left\{\sigma_1(k), \sigma_2(k) \ldots \sigma_q(k)\right\}$.

#### 4.1.1 Choice of Radial Basis Function

There exists the number of choices for the radial basis functions such as quadratic, inverse quadratic and thin plate of spline. However, Gaussian radial basis function is more intuitive than others as it produces significant output when the training data happen to be in the neighbourhood of the radial centre and vice versa [47]. This property of Gaussian radial basis function gives it an edge over other available choices of radial functions.

### 4.2 MLFFNN Model

Dashed red line square box in Fig. 2 shows the expanded view of hidden layer of MLFFNN. The induced field vector (sum of weighted inputs) of hidden neurons in MLFFNN is denoted by the vector $V(k)$ where $V(k) = \left\{V_1(k), V_2(k), \ldots V_q(k)\right\}$. The output of hidden neurons is denoted by a vector, $S(k) = \left\{S_1(k), S_2(k), \ldots S_q(k)\right\}$. Further, tangent hyperbolic function is used as an activation function for hidden neurons (as it can have both the positive and the negative values as its output) and linear activation function (so that output of output neuron can have any range of the values) for output neuron. The input weight vector (which connects external applied inputs to the neurons of hidden layer) is denoted by vector $W^I(k)$, and the output weight vector (which connects hidden neurons outputs to the neuron present in the output layer) in MLFFNN structure is denoted by $W^O(k) = \left\{w_1^O(k), w_2^O(k), \ldots w_q^O(k)\right\}$. The mathematical model of MLFFNN is given by [1]

$$\begin{aligned} Y_{\text{MLFFNN}}(k+1) &= \sum_q W_r^O f\left[V_q(k)\right] \\ &= \sum_q W_q^O f\left[\sum_p W_{pq}^I(k) x_p(k)\right] \end{aligned} \tag{6}$$

where $Y_{\text{MLFFNN}}(k+1)$ represents the MLFFNN output. Further, the externally applied $p$-input signal vector is denoted by $I^p = \left\{x_1(k), x_2(k) \ldots x_p(k)\right\}$.

### 4.3 NARX Model

The NARX model represents a generic recurrent neural network having its one step ahead output, $Y_{\text{NARX}}(k+1)$, depends upon the present input value along with its past values (called exogenous inputs) which are represented by: $\{x(k), x(k-1), \ldots x(k-n+1)\}$ as well as on its own present as well as the past values, that is, $\{Y_{\text{NARX}}(k), Y_{\text{NARX}}(k-1) \ldots Y_{\text{NARX}}(k-n+1)\}$ [15].

The term $n$ is decided by the operator, and in this paper it is taken equal to the order of the given plant. The induced field of any $r$th hidden neuron of NARX model is given by [15]

$$\begin{aligned} V_r(k) &= \sum_{j=1}^{n} W_{j1}^I(k) x(k-j+1) \\ &\quad + \sum_{j=1}^{n} W_{(j+2)1}^I(k) Y_{\text{NARX}}(k-j+1) \end{aligned} \tag{7}$$

The output of any $r$th hidden neuron at any $k$th instant is

$$S_r(k) = f\left[V_r(k)\right] \tag{8}$$

where $f$ is a hyperbolic tangent activation function. The output of the NARX model is then given by

$$Y_{\text{NARX}}(k) = \sum_q W_q^O S_q(k). \tag{9}$$

## 5 Structural Comparison of NARX, MLFFNN and RBFN

In this section, these ANNs are compared in terms of their structural characteristics.

**Definition 1** [48] Any neural network can be denoted by an ordered tuple $\{N\}^T$ which is given by $\{N\}^T = \{X^p, H^q, O^m\}$ with $p, q$ and $m$ denoting the number of externally applied inputs in the input vector $\{X^p\}$, number of hidden neurons (or nodes) present in the hidden layer $\{H^q\}$ and number of output neurons present in the output layer $\{O^m\}$ in MLFFNN, NARX and RBFN.

**Definition 2** [48] Let $\{N\}^{\text{MLFFNN}}, \{N\}^{\text{NARX}}, \{N\}^{\text{RBFN}}$ denote the total number of parameters, to be tuned present, in the MLFFNN, NARX and RBFN including the weights connecting the external bias signals which are applied to hidden neurons (or nodes) and output-layer neuron(s), respectively.

**Lemma** *The total count of parameters to be tuned in all these three NNs can be calculated using the below simple mathematical expressions* [48]:

$$\{N\}^{\text{MLFFNN}} = (p+m+1)q + m \tag{10}$$

$$\{N\}^{\text{NARX}} = (2n+m+1)q + m \tag{11}$$

$$\{N\}^{\text{RBFN}} = (m+3)q + m. \tag{12}$$

**Table 1** Comparative analysis of computational complexity of ANNs

| No. of external inputs = 2 no. of hidden neurons = 4 | No. of calculations for input multiplied with input weights | No. of operations for calculation of induced fields of hidden neurons | No. of Euclidean distance calculation | No. of output calculation of hidden neurons | No. of calculations of weighted output of hidden- layer neurons | Induced field calculation at output neuron | Total number of operation in one iteration |
|---|---|---|---|---|---|---|---|
| RBFN | 0 | 0 | 4 | 4 | 4 | 1 | 13 |
| MLFFNN | 8 | 4 | 0 | 4 | 4 | 1 | 21 |
| NARX | 8 | 4 | 0 | 4 | 4 | 1 | 21 |

The derivations of Eqs. 10–12 are as follows [48]: the total number of weights which connects hidden neurons present in the MLFFNN, NARX to these inputs will be $p \times q = 6$ (where in the case of NARX, $p = 2n$). Further, the total number of weights connecting hidden neurons/nodes to output-layer neurons will be $q \times m$. Also, the total number of weights connecting the bias signals applied to the hidden neurons (equals to $q$) and output neurons (equals to $m$) will be equal to $q + m$. So, in the case of MLFFNN and NARX models, the count of total number of weights will be $p \times q + q \times m + q + m = (p + m + 1)\, q + m$. In the case of RBFN, since weights connecting external inputs to hidden neurons are of unity value (and are not updated during the learning process) so they will not be considered in total weight count. So, the total number of parameters to be tuned in RBFN would be = number of weights connecting the hidden neurons to the output neurons ($= q \times m$) + number of radial centres ($= q$) + number of widths ($= q$) + number of weights connecting the bias signals applied to the hidden neurons ($= q$) + the number of weights connecting the bias signals applied to the output neurons ($= m$) which leads to a total count of $q \times m + q + q + q + m = (m + 3)q + m$.

### 5.1 Computational Complexity of ANN's Structures

Now these ANNs are compared in terms of the number of mathematical operations which are required to be performed for calculating the outputs. Consider $x_1(k)$ and $x_2(k)$ are the two inputs which are applied to the ANNs. Let hidden layer and output layer contain 4 and 1 neurons, respectively. The various mathematical operations which are performed in order to calculate the ANN's output are listed in Table 1. It can be seen from the table that in each iteration RBFN requires 8 less mathematical operations as compared to MLFFNN and NARX models. So, for instance, if training is continued for 1000 iterations, then RBFN will require $8 \times 1000$ less mathematical operations. This makes RBFN more computationally efficient than MLFFNN and NARX models.

## 6 Parameters Adjustment Rules for ANN-Based Controller

To provide an incremental type of learning, dynamic back-propagation algorithm based on gradient descent principle [1, 49,50] is used for tuning the various parameters of ANN-based controller. For doing this, mean square error (MSE) is used as the performance function. It is defined as follows:

$$E_c(k) = \frac{1}{2}\left[Y_d(k) - Y_p(k)\right]^2 \tag{13}$$

where $Y_d(k)$ and $Y_p(k)$ denote desired and actual plant's output and $Y_d(k) - Y_p(k) = -e_c(k)$ is the control error. The aim is to reduce the MSE value so that output of plant starts following the desired output. This will be achieved when all the parameters of ANN-based controller are adjusted using the update equations which are derived in the subsequent sections.

### 6.1 Case I: RBFN Controller

In the case of RBFN, radial centres, widths and the output weight vector are the parameters which are required to be tuned [47].

#### 6.1.1 Update Equations for Radial Centres

The update equation for radial centres is obtained by taking the partial derivative of $E_c(k)$ with respect to radial centre $M_{ij}(k)$ where $i = 1, \ldots, p$ and $j = 1, \ldots, q$. Thus,

$$\frac{\partial E_c(k)}{\partial M_{ij}(k)} = \frac{\partial E(k)}{\partial Y_p(k)} \times \frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial \phi_j(k)} \times \frac{\partial \phi_j(k)}{\partial M_{ij}(k)}. \tag{14}$$

The term $\frac{\partial Y_p(k)}{\partial u_c(k)}$ is called as sensitivity/Jacobian of the plant, and its value can be calculated if the mathematical expression (model) of $Y_p(k)$ is known. The model of plant is usually unknown or partially known. In such cases, we use ANN-based identifier in parallel to the plant and during the online training its parameters along with the parameters of controller will be adjusted. As the training progresses, value of $\frac{\partial Y_p(k)}{\partial u_c(k)} \approx \frac{\partial Y_{\mathrm{NN}}(k)}{\partial u_c(k)}$ (where $Y_{\mathrm{NN}}(k)$ represents ANN-based identifier output). Thus, mathematical model of identification model (with tuned parameters) can approximate the dynamics of the plant [51]. The other method is the perturbation method [52]. In this method, a slight change in the controller output leads to a change in the plant output. The ratio of change occurred in plant output to the small change occurred in controller output signal approximates the Jacobian value.

Further,

$$\frac{\partial \phi_j(k)}{\partial v_j(k)} = -v_j(k) \times \frac{\phi_j(k)}{\sigma_j^2(k)} \tag{15}$$

and

$$\frac{\partial v_j(k)}{\partial M_{ij}(k)} = \left( \frac{\partial \sum_j ((x_j(k) - M_{ij}(k))^2)^{\frac{1}{2}}}{\partial M_{ij}(k)} \right). \tag{16}$$

The following expression is obtained after doing some simplification:

$$\frac{\partial v_j(k)}{\partial M_{ij}(k)} = -\left( \frac{x_j(k) - M_{ij}(k)}{v_j(k)} \right) \tag{17}$$

Thus, the update equation for radial centres is

$$M_{ij}\,(k+1) = M_{ij}\,(k) + \Delta M_{ij}(k) \tag{18}$$

where

$$\Delta M_{ij} = \eta_c e_c(k) \frac{\partial Y_p(k)}{\partial u_c(k)} W_j(k) \frac{\phi_j(k)}{\sigma_j^2(k)} \left( x_j(k) - M_{ij}(k) \right) \tag{19}$$

and $\eta_c$ is a learning rate with its value set between 0 and 1.

### 6.1.2 Update Equations for Output Weight Vector

Now the adjustment of the output weight vector of RBFN is done as follows:

$$\frac{\partial E_c(k)}{\partial W_{oj}(k)} = \frac{\partial E_c(k)}{\partial Y_p(k)} \times \frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W_{oj}(k)} \tag{20}$$

where $\frac{\partial u_c(k)}{\partial W_{oj}(k)} = \phi_j(k)$. Thus, each element in $W_o(k) = [w_{o1}(k), w_{o2}(k), \ldots w_{oq}(k)]$ is updated as

$$W_{oj}\,(k+1) = W_{oj}\,(k) + \Delta W_{oj}(k) \tag{21}$$

where $\Delta W_{oj}(k) = \eta_c e_c(k) \frac{\partial Y_p(k)}{\partial u_c(k)} \phi_j(k)$.

### 6.1.3 Update Equations for the Radial Centre's Width

$$\frac{\partial E_c(k)}{\partial \sigma_j(k)} = \frac{\partial E_c(k)}{\partial Y_p(k)} \times \frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial \phi_j(k)} \times \frac{\partial \phi_j(k)}{\partial \sigma_j(k)} \tag{22}$$

where $\frac{\partial u_c(k)}{\partial \phi_j(k)} = W_{oj}(k)$ and $\frac{\partial \phi_j(k)}{\partial \sigma_j(k)} = \frac{\phi_j(k)(v_j(k))^2}{\sigma_j^3(k)}$. So, each element in $\sigma(k) = (\sigma_1(k), \sigma_2(k) \ldots \sigma_q(k))$ is updated as

$$\sigma_j(k+1) = \sigma_j(k) + \Delta\sigma_j(k) \tag{23}$$

where

$$\Delta\sigma(k) = \eta_c e_c(k) \frac{\partial Y_p(k)}{\partial u_c(k)} W_{oj}(k) \frac{\phi_j(k)(v_j(k))^2}{\sigma_j^3(k)} \tag{24}$$

It is to be noted that effect of $\sigma(k)$ values is not much on the overall output of the RBFN [47].

## 6.2 Case II: MLFFNN Controller Model

The input and output weight vectors are required to be tuned in case of MLFFNN-based controller [1]. They are updated as follows:

### 6.2.1 Adjustment of Output Weight Vector

The procedure of updating the weights of the output weight vector is as follows:

$$\frac{\partial E_c(k)}{\partial W_j^o(k)} = \frac{\partial E_c(k)}{\partial Y_p(k)} \times \frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W_j^o(k)} \tag{25}$$

where $\frac{\partial u_c(k)}{\partial W_j^o(k)} = S_j(k)$. Thus, each element in $W^O(k) = \left\{ w_1^O(k), w_2^O(k), \ldots w_q^O(k) \right\}$ will be updated as

$$W_j^o\,(k+1) = W_j^o\,(k) + \Delta W_j^o(k) \tag{26}$$

where $\Delta W_j^o(k) = \eta_c e_c(k) \frac{\partial Y_p(k)}{\partial u_c(k)} S_j(k)$.

### 6.2.2 Adjustment of Input Weight Vector

For updating the input weight vector, the update equation is obtained as:

$$\frac{\partial E_c(k)}{\partial W_j^I(k)} = \frac{\partial E_c(k)}{\partial Y_p(k)} \times \frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial S_j(k)} \times \frac{\partial S_j(k)}{\partial V_j(k)} \times \frac{\partial V_j(k)}{\partial W_j^I(k)} \tag{27}$$

where $\frac{\partial u_c(k)}{\partial S_j(k)} = W_j^o(k)$ since linear activation function of unity gain is used for output- layer neuron in MLFFNN and

NARX models, $\frac{\partial S_j(k)}{\partial V_j(k)} = 1 - S_j^2(k)$ as tangent hyperbolic function is used for hidden neurons in MLFFNN and NARX models, $\frac{\partial V_j(k)}{\partial W_j^I(k)} = x_j(k)$. For NARX controller model, the procedure of obtaining its update equations is same as that of the MLFFNN. The only difference is in the selection of input vector for NARX model [15].

## 7 Lyapunov Stability-Based Adaptive Learning Rate

The learning rate, $\eta_c$, in the update equations of ANN controllers plays a crucial role in the convergence speed and stability of the system. If its value is chosen to be very small, then convergence is guaranteed but the tuning of parameters will take more time. On the other hand, if its value is chosen to be large, then speed by which parameters undergo updating will be fast but there can be a chance of instability. So, this requires a judicious selection of the learning rate value. In this section, discrete Lyapunov stability method is used which will dictate how the learning rate value is to be chosen. It eventually leads to the development of an adaptive learning rate.

**Theorem** *In order to ensure the stability and convergence, the following condition on the learning rate, $\eta_c$, must be satisfied*

$$0 < \eta_c \leq \frac{2}{\left[\frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W(k)}\right]^2}. \tag{28}$$

***Proof*** Let the positive definite discrete Lyapunov function is given by:

$$V_l(k) = \frac{1}{2} e_c^2(k) \tag{29}$$

Now, $V_l(k)$ is nonzero and positive as long as $e_c(k)$ is nonzero. For the discrete-time system, stability is guaranteed if

$$\Delta V_l(k) = V_l(k+1) - V_l(k) < 0, \quad \text{in } D - \{0\} \tag{30}$$

where $D(\text{domain}) \rightarrow \Re$. Further, Eq. 30 can be written as

$$\Delta V_l(k) = V_l(k+1) - V_l(k) = \frac{1}{2}\left\{e_c^2(k+1) - e_c^2(k)\right\} \tag{31}$$

or

$$\Delta V_l(k) = \frac{1}{2}\left[e_c(k+1) + e_c(k)\right]\left[e_c(k+1) - e_c(k)\right]. \tag{32}$$

Now, $\Delta e_c(k) = e_c(k+1) - e_c(k)$ so Eq. 32 can be written as

$$\Delta V_l(k) = \frac{1}{2}\left[\Delta e_c(k) + 2e_c(k)\right]\left[\Delta e_c(k)\right] \tag{33}$$

or

$$\Delta V_l(k) = \Delta e_c(k)\left[\frac{1}{2}[\Delta e_c(k)] + e_c(k)\right] \tag{34}$$

Now in order to introduce any parameter of ANN controller into consideration, the fact that change in output error, $e_c(k)$, depends upon the amount of change in the parameter occurred. Let weight parameter $W$ is selected for derivation purpose. The procedure will remain same if any other parameter is considered. So, $\Delta e_c(k)$ can be written as

$$\Delta e_c(k) = \frac{\partial e_c(k)}{\partial W(k)} \Delta W(k) \tag{35}$$

where $\Delta W(k) \rightarrow 0$. Now on putting Eq. 35 in Eq. 34 results in

$$\Delta V_l(k) = \left[\frac{\partial e_c(k)}{\partial W(k)} \Delta W(k)\right]\left[\frac{1}{2}\left[\frac{\partial e_c(k)}{\partial W(k)} \Delta W(k)\right] + e_c(k)\right] \tag{36}$$

Now, from the update equations derived earlier, the change in $W$ is given by

$$\Delta W(k) = -\eta_c e_c(k) \frac{\partial e_c(k)}{\partial W(k)} \tag{37}$$

and

$$\frac{\partial e_c(k)}{\partial W(k)} = -\frac{\partial Y_p(k)}{\partial u_c(k)} \frac{\partial u_c(k)}{\partial W(k)} \tag{38}$$

Now substituting Eqs. 37 and 38 into Eq. 36 results in

$$\begin{aligned}\Delta V_l(k) = &\frac{1}{2}\left[\frac{\partial Y_p(k)}{\partial u_c(k)}\right]^4\left[\frac{\partial u_c(k)}{\partial W(k)}\right]^4 \eta_c^2 e_c^2(k) \\ &- \left[\frac{\partial Y_p(k)}{\partial u_c(k)}\right]^2\left[\frac{\partial u_c(k)}{\partial W(k)}\right]^2 \eta_c e_c^2(k)\end{aligned} \tag{39}$$

or

$$\begin{aligned}\Delta V_l(k) = &-\eta_c e_c^2(k)\left[\frac{\partial Y_p(k)}{\partial u_c(k)}\right]^2\left[\frac{\partial u_c(k)}{\partial W(k)}\right]^2 \\ &\times \left\{1 - \frac{\eta_c}{2}\left[\frac{\partial Y_p(k)}{\partial u_c(k)}\right]^2\left[\frac{\partial u_c(k)}{\partial W(k)}\right]^2\right\}.\end{aligned} \tag{40}$$

For implementation purpose, Eq. 30 can be written as $\Delta V \leq 0$ so

$$- \eta_c e_c^2(k) \left[\frac{\partial Y_p(k)}{\partial u_c(k)}\right]^2 \left[\frac{\partial u_c(k)}{\partial W(k)}\right]^2 \times \left\{1 - \frac{\eta_c}{2}\left[\frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W(k)}\right]^2\right\} \le 0. \tag{41}$$

Multiplying both sides by negative sign results in

$$\eta_c e_c^2(k) \left[\frac{\partial Y_p(k)}{\partial u_c(k)}\right]^2 \left[\frac{\partial u_c(k)}{\partial W(k)}\right]^2 \times \left\{1 - \frac{\eta_c}{2}\left[\frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W(k)}\right]^2\right\} \ge 0 \tag{42}$$

Now Eq. 42 is true if certain conditions imposed on various terms present in it gets satisfied. In the term $\eta_c e_c^2(k)\left[\frac{\partial Y_p(k)}{\partial u_c(k)}\right]^2\left[\frac{\partial u_c(k)}{\partial W(k)}\right]^2$, terms $e_c^2(k)$, $\left[\frac{\partial Y_p(k)}{\partial u_c(k)}\right]^2$ and $\left[\frac{\partial u_c(k)}{\partial W(k)}\right]^2$ can be either positive or zero so $\eta_c$ must have to be $\ge 0$ for Eq. 42 to hold true. Thus,

$$\eta_c \ge 0 \tag{43}$$

Since $\eta_c$ cannot be set to equal to zero (as there will not be any learning in such a case), the first condition on the value of $\eta_c$ can be written as:

$$\eta_c > 0. \tag{44}$$

If Eq. 44 holds true, then $\Delta V(k) \le 0$ hold true only if the remaining terms present in Eq. 42 also satisfy the below condition

$$\left\{1 - \frac{\eta_c}{2}\left[\frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W(k)}\right]^2\right\} \ge 0 \tag{45}$$

or

$$\left\{2 - \eta_c\left[\frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W(k)}\right]^2\right\} \ge 0 \tag{46}$$

Now $\left[\frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W(k)}\right]^2$ will be either positive or zero. So, Eq. 46 equals to zero if

$$\eta_c = \frac{2}{\left[\frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W(k)}\right]^2} \tag{47}$$

Hence, Eq. 46 holds true if

$$\eta_c \le \frac{2}{\left[\frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W(k)}\right]^2} \tag{48}$$

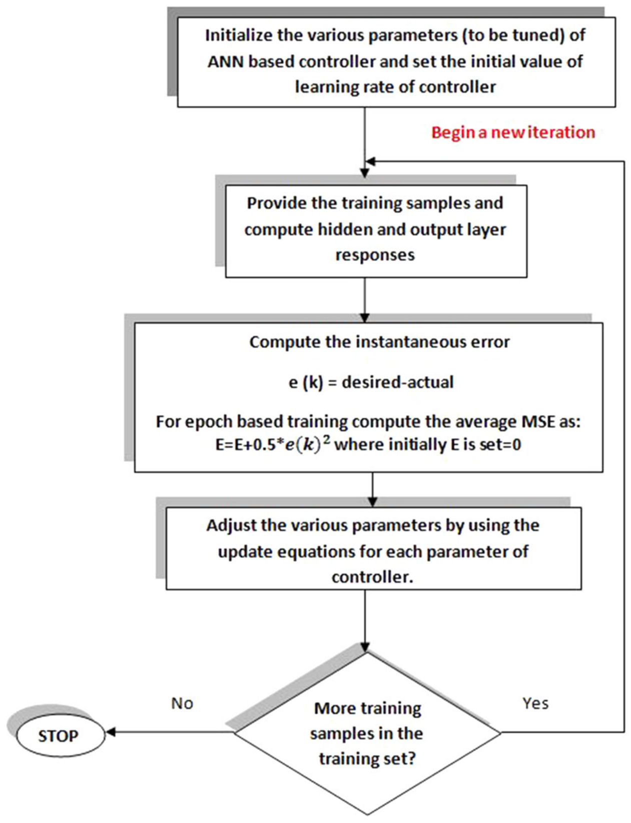

Initialize the various parameters (to be tuned) of ANN based controller and set the initial value of learning rate of controller
Begin a new iteration
Provide the training samples and compute hidden and output layer responses
Compute the instantaneous error
e (k) = desired-actual
For epoch based training compute the average MSE as: E=E+0.5*e(k)^2 where initially E is set=0
Adjust the various parameters by using the update equations for each parameter of controller.
More training samples in the training set?
No
Yes
STOP

**Fig. 4** General sequence of steps in the parameter iteration algorithm

Thus, from Eqs. 44 and 48 we have the lower and upper bounds for the $\eta_c$ which leads to the following condition on $\eta_c$ that must hold true for the given system to be stable.

$$0 < \eta_c \le \frac{2}{\left[\frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W(k)}\right]^2} \tag{49}$$

Hence, Eq. 28 is proved. Now by using Eq. 49 the learning rate can be made adaptive as:

$$\eta_c(k) = \frac{2}{\left[\frac{\partial Y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W(k)}\right]^2} \tag{50}$$

In the same manner, adaptive equations for the learning rates associated with remaining parameters of the ANN can be obtained. The basic steps involved in the online training of ANN controller are shown in Fig. 4 in the form of a flowchart. □

## 8 Adaptive Control of Nonlinear Systems

The main aim of control is to have the desired output values from the plant. The plant could be required either to follow the output of reference model (this is called as model reference

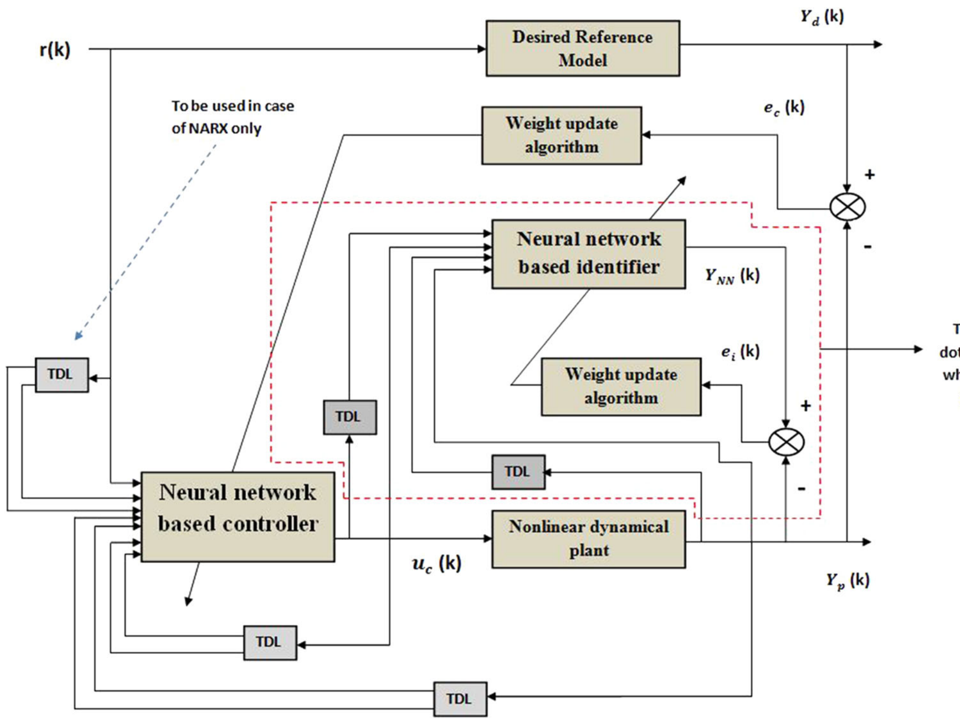

r(k)
Desired Reference Model
$Y_d$ (k)
Weight update algorithm
$e_c$ (k)
To be used in case of NARX only
+
-
Neural network based identifier
$Y_{NN}$ (k)
This portion (enclosed in dotted line box) is to be used when dynamics of the plant is unknown or partially known
$e_i$ (k)
Weight update algorithm
+
-
TDL
TDL
TDL
Neural network based controller
Nonlinear dynamical plant
$u_c$ (k)
$Y_p$ (k)
TDL
TDL

Fig. 5 Indirect adaptive control scheme based on artificial neural networks

adaptive control) or to follow the externally applied input signal (this is known as reference input tracking control). If the parameters of plant undergo change and/or disturbance signal affects the system and even then if plant is able to give the desired values under the controller action, then such type of control is called as adaptive control. There exist two types of adaptive control

1. Direct control
2. Indirect control

In direct control, parameters of the controller are tuned (or adjusted) at every time instant using the error between the plant output and desired output. This approach has been successfully applied to the linear case. For nonlinear plants, direct methods are not developed, and hence, indirect adaptive control is required. In indirect control, parameter adjustment algorithm requires the knowledge regarding the mathematical model of the plant along with the value of error at each instant. This indirect adaptive control scheme is shown in Fig. 5.

From the figure, it can be seen that inputs of ANN-based controller include externally applied input along with its own delayed values as well as the delayed values of the plant's output. How many delayed values will be used is explained latter. Also, for the case of tracking type of control, the reference model dynamics would be equal to unity and in such case $Y_d(k) = r(k)$. The TDL block denotes tapped delay line and is modelled as $z^{-1}$; thus, its output is one unit time delay of its input. For example, if the input of TDL is $Y_p(k)$, then its output will be $Y_p(k-1)$.

### 8.1 Selection of Controller Inputs

To provide the sufficient information to the MLFFNN-, NARX- and RBFN-based controllers, the following approach is used: the total count of inputs presented to the RBFN- and MLFFNN-based controllers is considered equal to $2n$ (where $n$ is equal to the order of the plant) and it includes: $r(k)$, $Y_p(k), \ldots Y_p(k-n+1)$ and past $n-1$ values of $u_c(k)$, whereas in the NARX-based controller, the control inputs are [15]: $Y_p(k), \ldots Y_p(k-n+1)$ and $r(k), \ldots r(k-n+1)$.

## 9 Simulation Study

Three simulation examples are used for comparing the performances of RBFN-, MLFFNN- and NARX-based controllers. In each controller, only one hidden layer containing 20 neurons/nodes is considered.

### 9.1 Example 1 Dynamical System of Relative Degree 3

Let the dynamics of the plant is given by the difference equation as given in [1]:

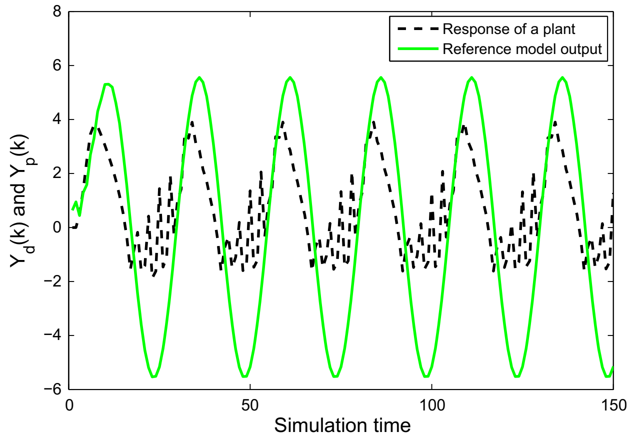


**Fig. 6** Open-loop response of plant without controller action (Example 1)

$$Y_p(k+1) = F\left[Y_p(k), Y_p(k-1), Y_p(k-2)\right] + u_c(k) + 0.8u_c(k-1) \quad (51)$$

The nonlinear function $F$ is given by:

$$F\left[x_1, x_2, x_3\right] = \frac{5x_1x_2}{1 + x_1^2 + x_2^2 + x_3^2} \quad (52)$$

where $x_1 = Y_p(k), x_2 = Y_p(k-1), x_3 = Y_p(k-2)$. The desired reference model difference equation is given by

$$Y_d(k+1) = 0.72Y_d(k) + 0.64Y_d(k-1) - 0.5Y_d(k-2) + r(k) \quad (53)$$

The externally applied input signal in this example is taken to be $r(k) = \sin\left(\frac{2\pi k}{25}\right)$. From Eq. 52, it can be seen that the order $n$ of the given plant is 3 so total 6 inputs will be given to the controllers. The open-loop response (without controller action) of the plant and that of the reference model with respect to externally applied input $r(k)$ signal are shown in Fig. 6.

From the figure it can be seen that response given by the plant (dashed black curve) is not following the reference model output (continuous green curve); hence, controller action is needed. For this, control scheme as shown in Fig. 5 is set up. The response of the plant is recorded by considering both fixed ($\eta_c = 0.005$) and adaptive learning rate (with initial value set to $\eta_c(0) = 0.005$). During the initial stage of training, output of the controller will not be equal to as desired but it will keep on getting improved since its parameters are getting tuned as the training progresses, its output will serve as an input to the plant and hence forces the plant to give the desired output. The plant response during the initial and final stages of training under the MLFFNN-, NARX- and RBFN-based controller actions is shown in Figs. 7, 8, 9, 10, 11 and

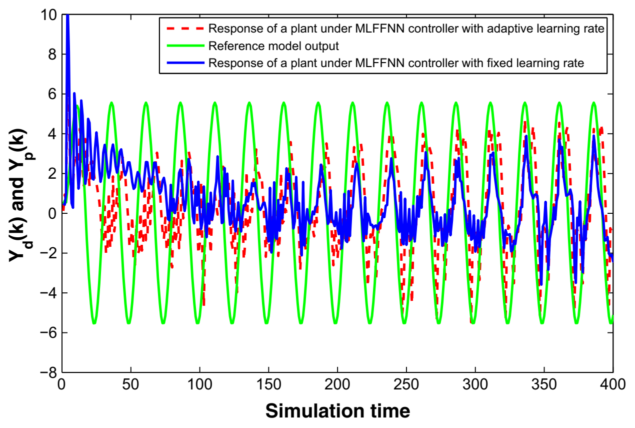


**Fig. 7** Response of plant under MLFFNN controller during the initial stage of training (Example 1)

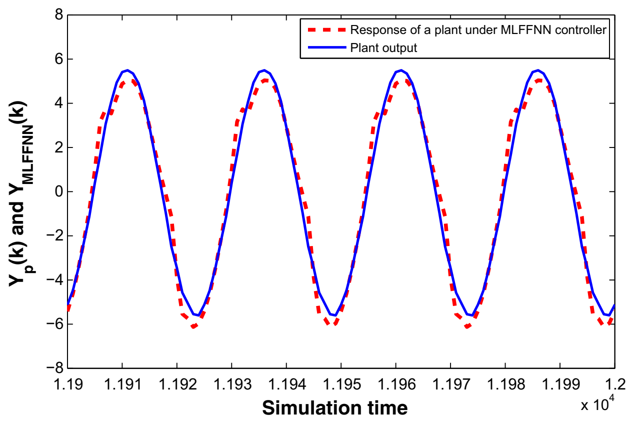


**Fig. 8** Response of plant under MLFFNN controller after sufficient amount of training (Example 1)

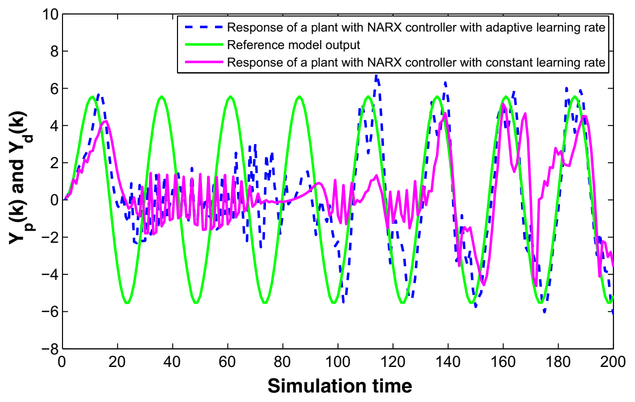


**Fig. 9** Response of plant under NARX controller during the initial stages of training (Example 1)

12, respectively. Further, the MSE plots obtained during the online training are shown in Figs. 13 and 14.

The terms $\eta^O_{c\mathrm{RBFN}}$, $\eta^W_{c\mathrm{RBFN}}$ and $\eta^C_{c\mathrm{RBFN}}$ denote adaptive learning rates associated with output-layer weight vector, widths and radial centres of RBFN controller, respectively, and at the

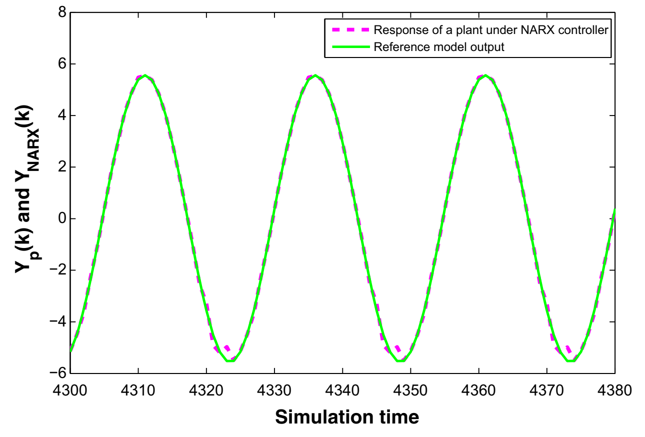


**Fig. 10** Response of plant under NARX controller after sufficient amount of training (Example 1)

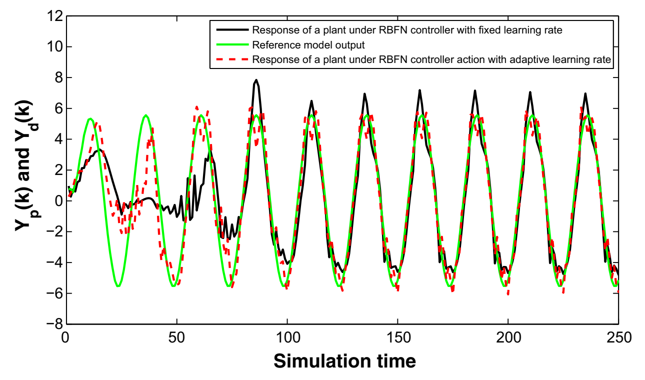


**Fig. 11** Response of plant under RBFN controller during the initial stages of training (Example 1)

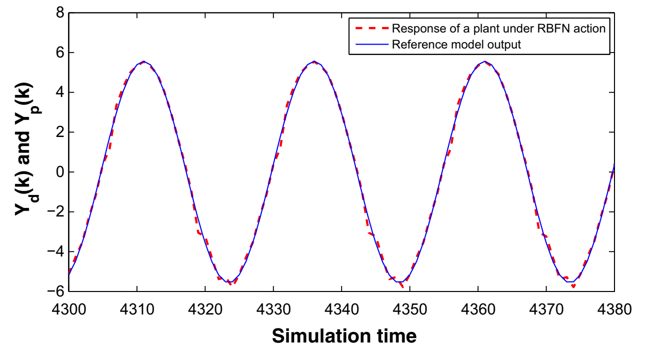


**Fig. 12** Response of plant under RBFN controller during the final stages of training (Example 1)

end of the online training they converged to the following values: 0.0000101, 0.000251 and 0.0000374, respectively. During the initial stage of training, when the error was large and the parameters of controllers are still in the early stage of tuning, the learning rate values $\eta^{O}_{c\text{RBFN}}$, $\eta^{W}_{c\text{RBFN}}$ and $\eta^{C}_{c\text{RBFN}}$ shoot to a maximum value of 2.12, 2.35 and 1.61, respectively. As the training continued they finally settled down to a small value as mentioned above. Similarly in the case of

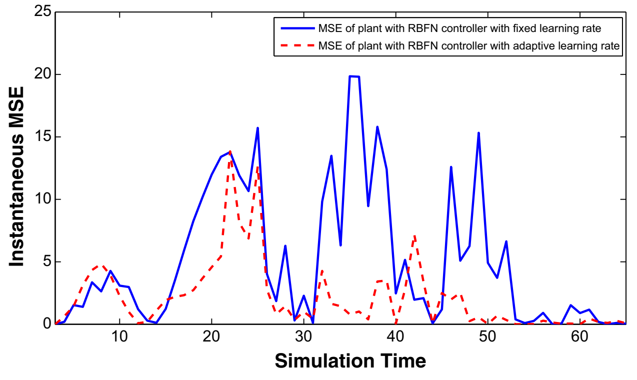


**Fig. 13** MSE of plant under RBFN controller during the online training (Example 1)

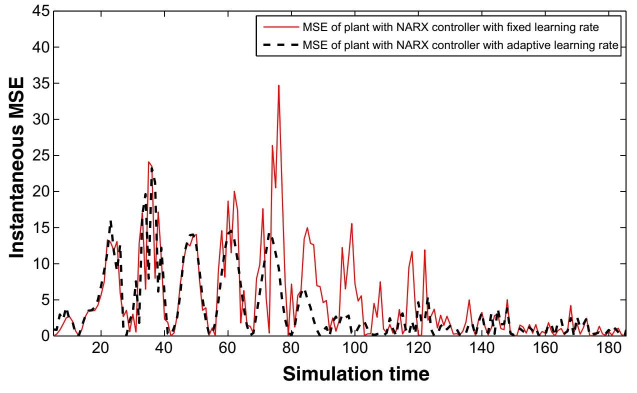


**Fig. 14** MSE of plant under NARX controller during the online training (Example 1)

NARX controller, the final values obtained of adaptive learning rates associated with the output and input weight vectors are $\eta^{O}_{c\text{NARX}} = 0.000041$ and $\eta^{I}_{c\text{NARX}} = 0.0000727$. The upper bounds found in the learning rates associated with the output- and input-layer weight vectors of NARX-based controller during the online training are 2.44 and 2.56, respectively. Lastly, the adaptive rate values associated with the output- and input-layer weight vectors of MLFFNN controller converged to 0.000213 and 0.0004 at the end of the training with upper bounds found equal to 3.1 and 2.55, respectively, during the training.

### 9.1.1 Robustness Testing

A system is said to be robust if it is able to compensate the effects of system's uncertainties like disturbance signal and parameter variations. In this section, robustness of all these three controllers is tested and compared. The learning rate remains adaptive in the analysis.

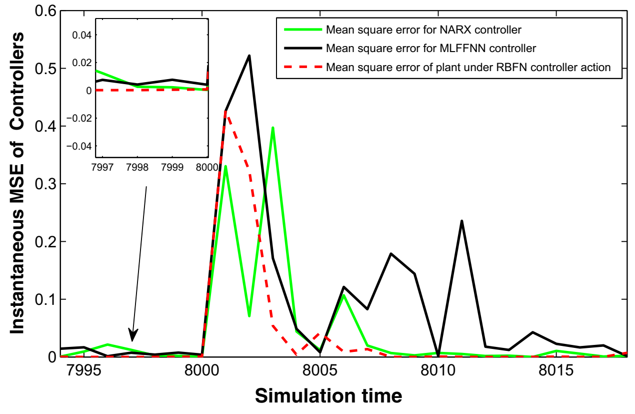


Fig. 15 Instantaneous MSE when disturbance signal affects the system (Example 1)

## 9.1.2 Case I: Disturbance Signal

Let the disturbance signal of magnitude 0.9 units be added in the output of the controller at $k = 8000$th time instant. Naturally, the error would increase at this instant, but if the controller is robust, then this error must be compensated and reduced down to zero in a short period of time. This instantaneous MSE plots corresponding to RBFN-, MLFFNN- and NARX-based controller are shown in Fig. 15.

From the figure it can be seen that MSE drops again very quickly to zero when RBFN controller is used followed by NARX- and MLFFNN-based controllers. This shows that RBFN is much better in compensating the effects of external disturbance signal as compared to NARX and MLFFNN.

## 9.1.3 Case II: Parameter Variation

Now at $k = 12000$th time instant, the output weight vector of all the three controllers was perturbed by deliberately adding a signal of 0.8 units. This should cause a rise in MSE which should reduce again to zero if the parameter update equations are correct. The instantaneous MSE of plant under RBFN, MLFFNN and NARX controller actions in the presence of parameter variation is shown in Fig. 16.

From the figure it can be easily seen that the rise in the MSE is very small at the time of parameter variation when RBFN is used as a controller (shown by red curve) as compared to when NARX (shown by green curve) and MLFFNN (shown by black curve) are used as the controllers. Also, this rise in the MSE is compensated and reduced much quickly to zero in the RBFN case as compared to error compensation obtained with NARX- and MLFFNN-based controllers. The corresponding changes which occurred in the RBFN, MLFFNN and NARX controller outputs are shown in Figs. 17, 18 and 19, respectively.

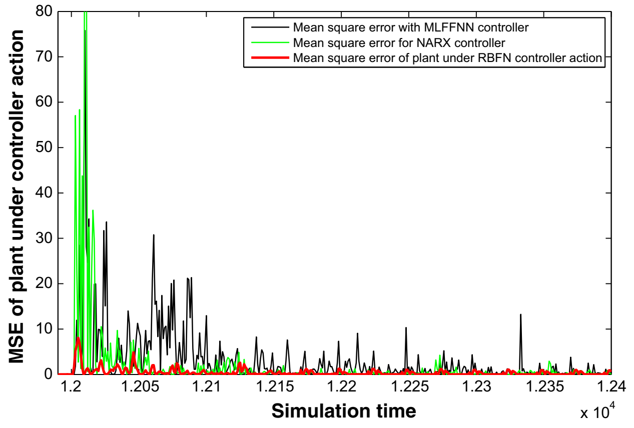


Fig. 16 Instantaneous MSE when parameter variation affects the system (Example 1)

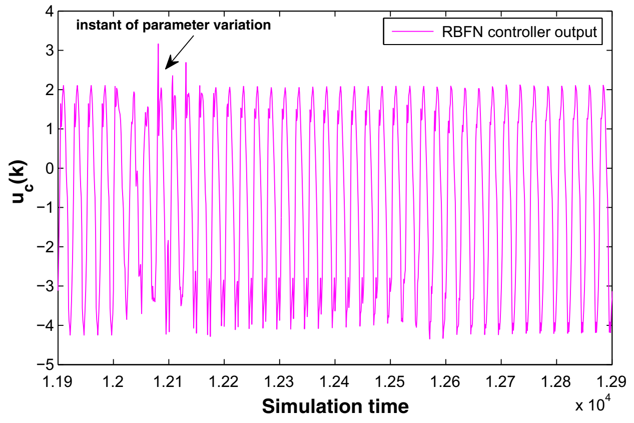


Fig. 17 RBFN controller output when parameter variation affects the system (Example 1)

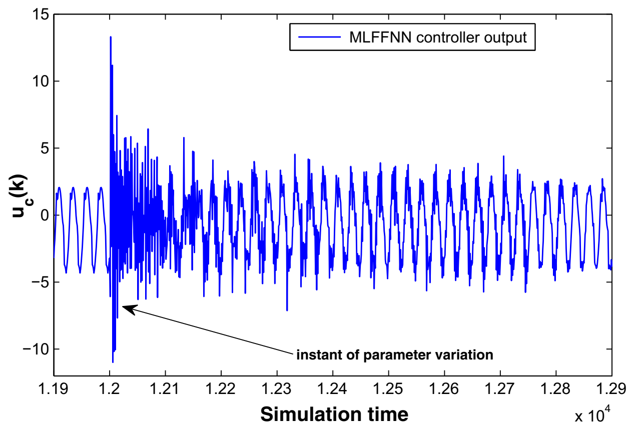


Fig. 18 MLFFNN controller output when parameter variation affects the system (Example 1)

From the figures it can be seen that the RBFN controller output gets least affected and recovered at a much faster rate as compared to NARX and MLFFNN controller outputs. This

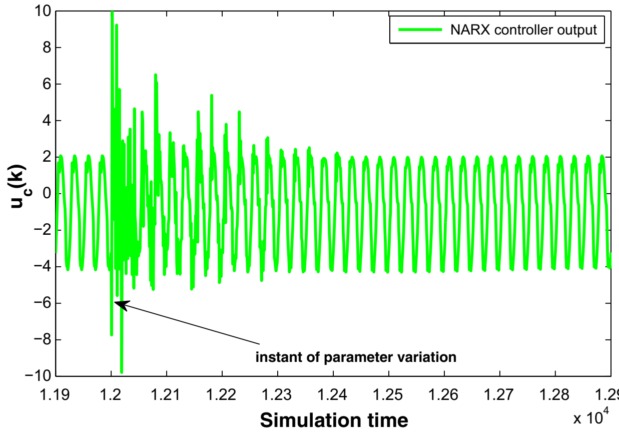


**Fig. 19** NARX controller output when parameter variation affects the system (Example 1)

**Table 2** Time elapsed for the ANN's control program during the training (Example 1)

| Type of neural network | Simulation time (s) |
|---|---|
| RBFN | 0.533788 |
| MLFFNN | 8.929530 |
| NARX | 9.328220 |

again shows that the RBFN is more robust than NARX- and MLFFNN-based controllers.

### 9.1.4 Analysis of the Results: Example 1

Now from the initial responses of the plant, it can be seen that when adaptive learning rate is used the output of the plant is seen to improve at a faster rate as compared to the output obtained with the fixed learning rate. Now, considering the performances of the individual controllers, it can be easily seen that initial response of the plant got improved at a much faster rate when RBFN is used as a controller, whereas with NARX- or MLFFNN-based controllers, the improvement in the response is comparatively less. Further, the instantaneous MSE plots are obtained with fixed as well as with the adaptive learning rate. These are shown in Figs. 13 and 14. From the plots, it can be concluded that the error tends to decrease at a much faster rate when adaptive learning rate is used as compared to when fixed learning rate is used. Also, the time elapsed for which the control program run in this example for each RBFN-, MLFFNN- and NARX-based control configuration is shown in Table 2 [using MATLAB version 7.12.0.635(R2011a), 4 GB RAM, Windows 8, Intel core 2 duo processor].

It can be seen from Table 2 that the control program took very less time to execute when RBFN is used as a controller, whereas the program took more time to finish when NARX and MLFFNN are used in the loop as the controller. This makes RBFN an efficient tool for doing the control operation. Further, the average MSEs obtained with different controllers (when the control program is run for 2000 iterations) are listed in Table 3.

**Table 3** Average MSE obtained during the online control (Example 1)

| Type of neural network | Average MSE (after 20,000 iterations) |
|---|---|
| RBFN | 0.012 |
| MLFFNN | 0.17 |
| NARX | 0.08 |

From the table it can be seen that average MSE obtained with RBFN controller is minimum as compared to the MSE obtained with NARX- or MLFFNN-based controllers. Further, the RBFN-based controller is found to be more robust in comparison with NARX and MLFFNN controllers in compensating the effects of parameter variations and disturbance signals.

## 9.2 Example 2 Dynamical System of Relative Degree 4

Consider a nonlinear system whose dynamics are given as in [53]

$$\begin{aligned} Y_p(k+1) &= 0.8Y_p(k) + 0.15Y_p(k-1) \\ &\quad + 0.15\sin\left[Y_p(k-2) - e^{-Y_p(k-3)}\right] \\ &\quad + 0.2u_c(k) + 0.16u_c(k-1) \end{aligned} \tag{54}$$

The order of the plant is $n = 4$, and the inputs for RBFN and MLFFNN controllers are: $r(k)$, $Y_p(k)$, $Y_p(k-1)$, $Y_p(k-2)$, $Y_p(k-3)$, $u_c(k-1)$, $u_c(k-2)$ and $u_c(k-3)$ and inputs for NARX-based controller are: $Y_p(k)$, $Y_p(k-1)$, $Y_p(k-2)$, $Y_p(k-3)$, $r(k)$, $r(k-1)$, $r(k-2)$, $r(k-3)$. The desired reference model difference equation is given as:

$$Y_d(k) = 0.6Y_d(k) + 0.2Y_d(k-1) + r(k) \tag{55}$$

The externally applied input signal is given by $r(k) = \sin(\frac{2\pi k}{250}) + \cos(\frac{2k}{123})$. The initial value of fixed and adaptive $\eta$ is set to 0.0052. The response of plant and desired reference model when no controller is used is shown in Fig. 20.

It is clear from the figure that the output of the plant is not following the desired response. Now the control configuration as shown in Fig. 5 is set up and the control action is initiated. The parameters of the controller undergo change (as guided by the learning algorithm) and eventually reach to their corresponding desired values. The response of plant

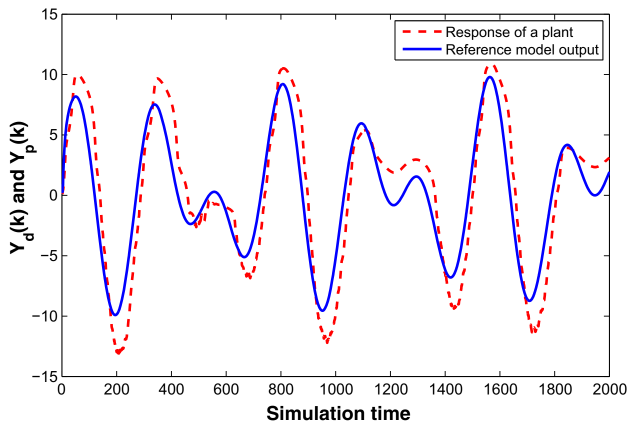


**Fig. 20** Response of plant without controller action (Example 2)

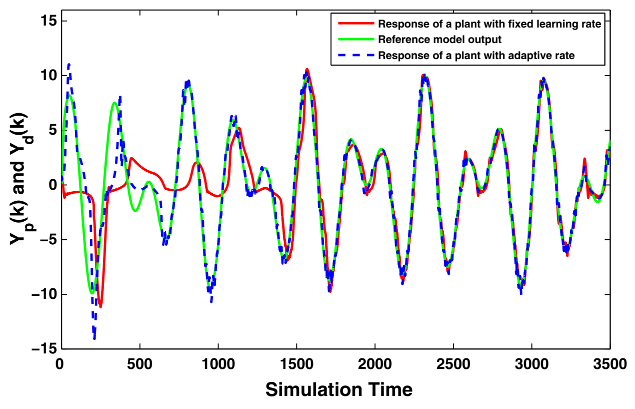


**Fig. 21** Response of plant under RBFN controller action (Example 2)

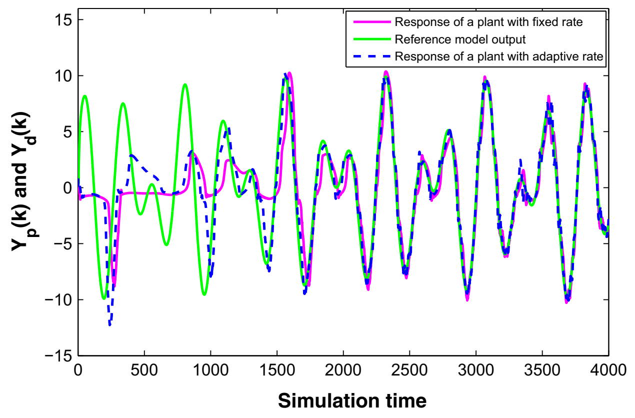


**Fig. 22** Response of plant under MLFFNN controller action (Example 2)

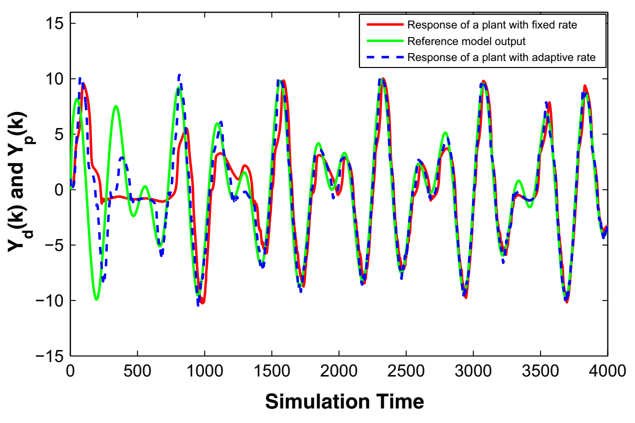


**Fig. 23** Response of plant under NARX controller action (Example 2)

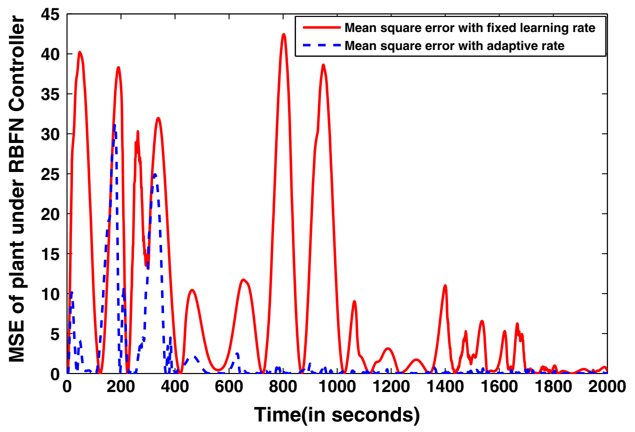


**Fig. 24** MSE of plant obtained under RBFN controller action (Example 2)

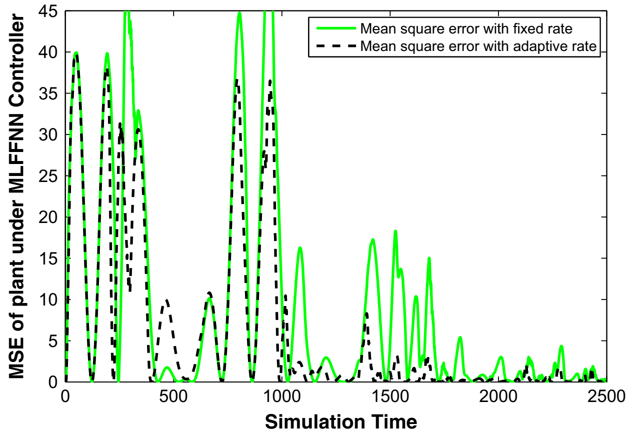


**Fig. 25** MSE of plant under MLFFNN controller action (Example 2)

obtained under RBFN-, MLFFNN- and NARX-based controllers is shown in Figs. 21, 22 and 23, respectively.

The corresponding instantaneous MSE obtained is shown in Figs. 24, 25 and 26, respectively.

From the figures it can be seen that response of plant is reaching the desired response quickly when adaptive learning rate is employed. Among the controllers, RBFN is seen to be performing better than NARX- and MLFFNN-based controllers (with both fixed and adaptive learning rate). This is also evident from the MSE obtained during the online control. Further, the average MSE obtained and the amount of time taken by control algorithm to run are listed in Table 4.

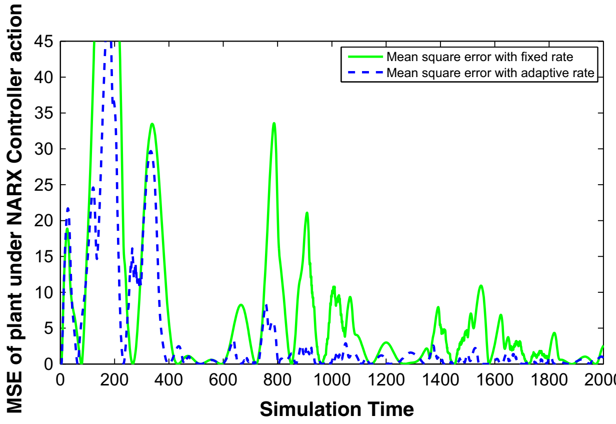


**Fig. 26** MSE of plant under NARX controller action (Example 2)

**Table 4** Average MSE and run-time comparison (Example 2)

| Type of controller | Average MSE (10,000 iterations) | Simulation time (s) |
|---|---|---|
| MLFFNN | 0.0913 | 4.722920 |
| RBFN | 0.0228 | 0.335487 |
| NARX | 0.0491 | 4.738064 |

It is clear that the RBFN-based control configuration took least time to run and produced least average MSE as compared to the values obtained with NARX- and MLFFNN-based controllers. This shows that RBFN is more computationally efficient as well as more capable of controlling the nonlinear systems. Further, the adaptive learning rate values associated with the output- and input-layer weight vectors of MLFFNN controller converge to 0.00050 and 0.00061; in the case of NARX-based controller they converged to 0.000088 and 0.000059; and in the RBFN case the final learning rate values associated with the output-layer weight vector, widths and radial centres converge to 0.000033, 0.00031 and 0.000019, respectively. The upper bounds of adaptive learning rate found in case of MLFFNN controller are 2.81 and 2.41 for output- and input-layer weights, respectively. In case of NARX-based controller, the upper bound is equal to 2.1 and 1.96. Finally, for the RBFN-based controller the upper bounds found for output-layer weight vector, widths and radial centres are 1.7, 2.34 and 1.41, respectively.

### 9.2.1 Robustness Testing

### 9.2.2 Case I: Disturbance Signal

Let the disturbance signal of magnitude 2.9 units be added in the controller output at the 7000th time instant. The recovery ability of controllers against this disturbance signal is shown in Fig. 27.

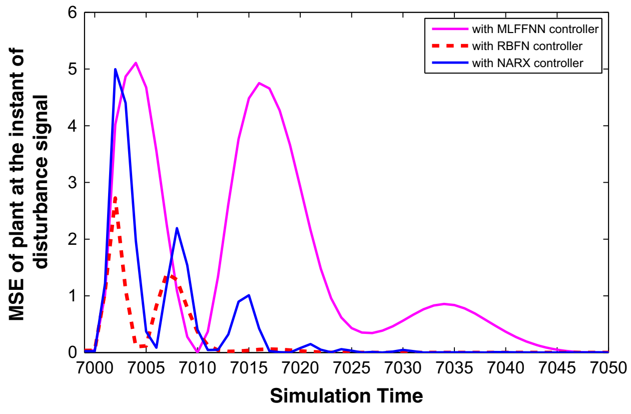


**Fig. 27** MSE of plant at the instant of disturbance signal (Example 2)

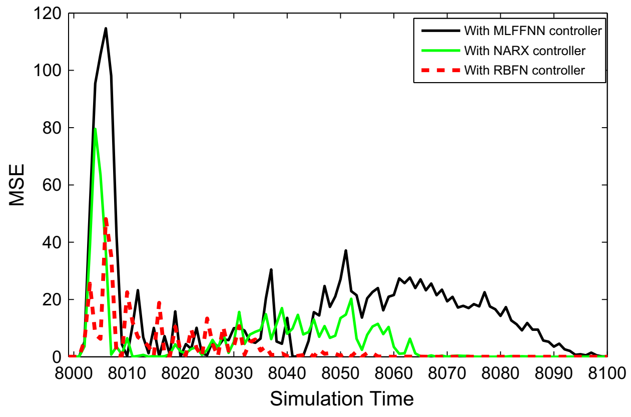


**Fig. 28** MSE of plant at the instant of parameter variation (Example 2)

It can be seen that rise in MSE caused by the disturbance signal is much quickly reduced down to zero when RBFN controller is used as compared to other two controllers. This again shows the effectiveness of RBFN controller over NARX- and MLFFNN-based controllers.

### 9.2.3 Case II: Parameter Variation

In order to test the robustness of controllers against the effects of parameter variations, a signal of 1.8 unit magnitude is added in the output-layer weights of these controllers. This causes an increase in the MSE value. Figure 28 shows the recovery ability of the controllers against the effects of parameter variations. It can be seen that RBFN is performing better than NARX- and MLFFNN-based controllers.

### 9.2.4 Analysis of the Results: Example 2

From the simulation results, it can be easily seen that the performance of RBFN-based controller is much better than that of NARX- and MLFFNN-based controller with fixed as well as with adaptive learning rate. The other important observation that can be deduced from the simulation results is that the RBFN-based controller has shown more robustness as compared to NARX- and MLFFNN-based controllers towards parameter variations and disturbance signals. This makes RBFN an ideal control tool for nonlinear systems.

### 9.3 Example 3 Robotic Manipulator Control

The mathematical model of an $n$ serial links robotic manipulator is given by:

$$M(Y_p)\ddot{Y}_p + C(Y_p, \dot{Y}_p)\dot{Y}_p + D\dot{Y}_p + G(Y_p) = \tau \tag{56}$$

In the above equation, $Y_p$, $\dot{Y}_p$ and $\ddot{Y}_p \in R^n$ denote the position of the joint, velocity and the accelerations, respectively. The $n \times n$ real matrices: $M, C, D$ are the symmetric positive definite inertia matrix, centrifugal Coriolis matrix and positive definite diagonal matrix for damping friction coefficients of each joint, respectively. The gravity term is denoted by $n \times n$ matrix $G(y_p)$. Also, $\tau \in R^n$ represents the input vector of torques exerted on the joints. In this paper, the control of one-link robotic manipulator is performed. For more details regarding the kinematics of robotic manipulator, readers are encouraged to read these reference materials [54–56]. The second-order differential equation which describes the dynamics of one-link robotic manipulator is given as in [57]:

$$ml^2\frac{\mathrm{d}^2Y_p(t)}{\mathrm{d}t^2} + D\frac{\mathrm{d}Y_p(t)}{\mathrm{d}t} + mgl\cos(Y_p(t)) = u_c(t) \tag{57}$$

Here the angular position of the robotic manipulator arm is represented by $Y_p(t)$. The other parameters include: link length which is denoted by $l$. Also, the term $D\frac{\mathrm{d}Y_p(t)}{\mathrm{d}t}$ quantifies the viscous friction torque and $g =$ is the acceleration due to gravity, $= 9.8\,\mathrm{m/s}^2$. For simplicity, the values of $m = l = D = 1$ are used in this paper. Further, $u_c(t)$ is the control torque exerted on the robotic arm. The corresponding state space representation of above differential equation is given by:

$$\begin{bmatrix}\dot{Y}_p \\ \ddot{Y}_p\end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-g\cos(Y_p)}{l} & \frac{-D}{ml^2}\end{bmatrix}\begin{bmatrix}Y_p \\ \dot{Y}_p\end{bmatrix} + \begin{bmatrix}0 \\ \frac{1}{ml^2}\end{bmatrix}[u_c]. \tag{58}$$

The difference equation of one-link robotic manipulator with sampling period, $T = 0.45\,\mathrm{s}$ is given by

$$Y_p(k+1) = F[Y_p(k), Y_p(k-1)] + T^2u_c(k) \tag{59}$$

where the nonlinear function $F$ is

$$F[Y_p(k), Y_p(k-1)] = (2-T)Y_p(k) + Y_p(k-1)(T-1) - 9.8T^2\cos(Y_p(k-1)). \tag{60}$$

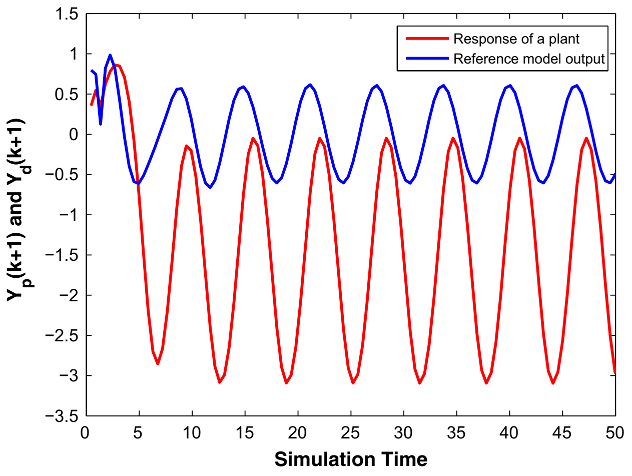


**Fig. 29** Open-loop response of plant without controller in the loop (Example 3)

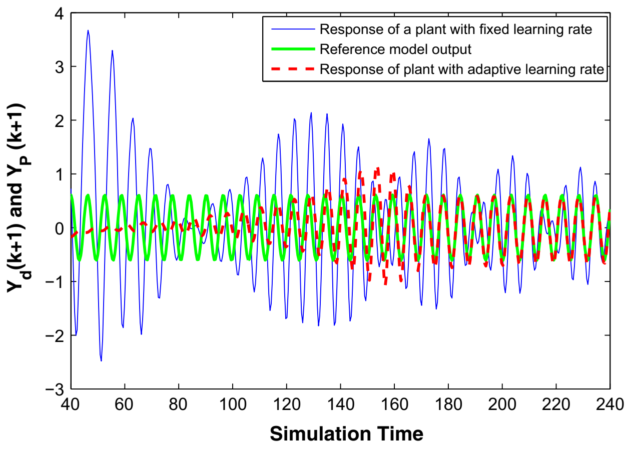


**Fig. 30** Response of plant under RBFN controller action at the initial stage of training (Example 3)

Note that the order of Eq. 59 is $n = 2$, so total number of inputs applied to the controllers are equal to 4. The desired reference model dynamics are given by the following difference equation

$$\begin{aligned} Y_d(k+1) &= Y_d(k)(2-1.5T) + Y_d(k-1) \\ &\quad \times (-1 - 2.5T^2 + 1.5T) + T^2r(k-1) \end{aligned} \tag{61}$$

where $r(k) = \sin(t)$ is the externally applied input. The response of plant (without control) and reference model is shown in Fig. 29.

It can be seen from the figure that output of plant is not following the reference model output. Hence, the controller is required in the loop. The value of fixed learning rate, $\eta_c$, as well as initial value of adaptive learning rate, $\eta_c(0)$, is set equal to 0.027. The responses of plant obtained at various stages of training under RBFN controller action are shown in Figs. 30, 31 and 32.

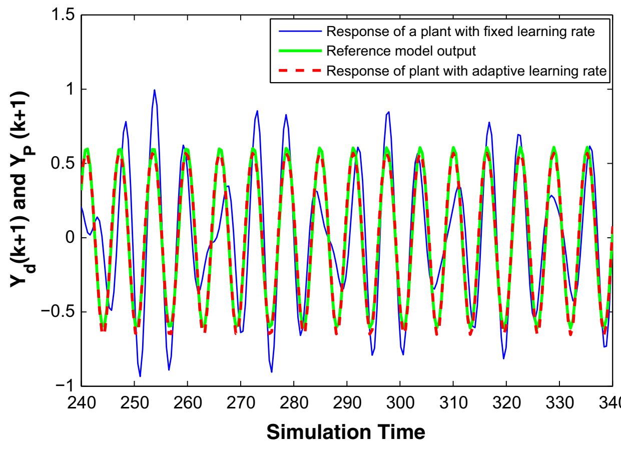


**Fig. 31** Response of plant under RBFN controller action as the training progresses (Example 3)

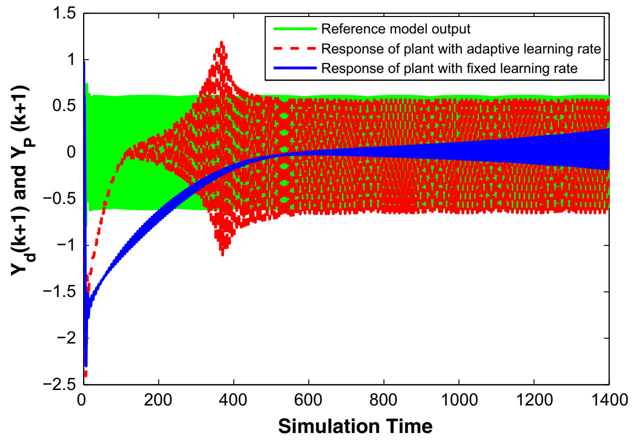


**Fig. 34** Response of plant under MLFFNN controller action as the training progresses (Example 3)

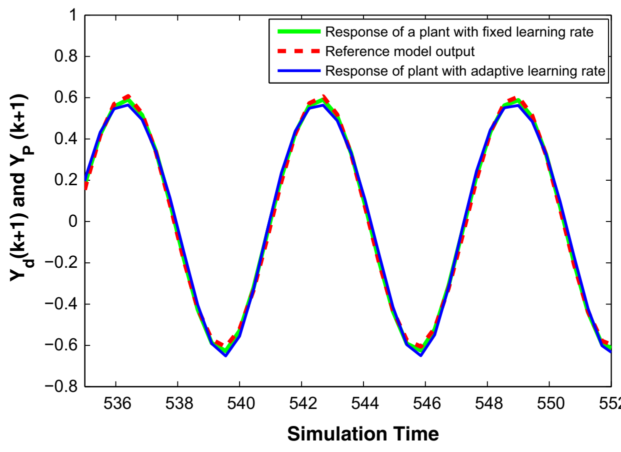


**Fig. 32** Response of plant under RBFN controller action after sufficient amount of training (Example 3)

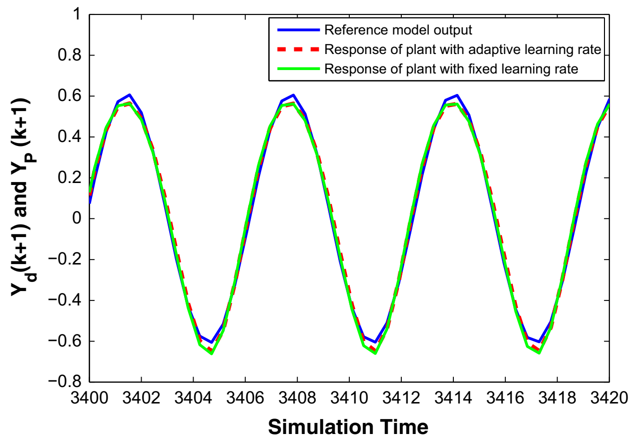


**Fig. 35** Response of plant under MLFFNN controller action after sufficient amount of training (Example 3)

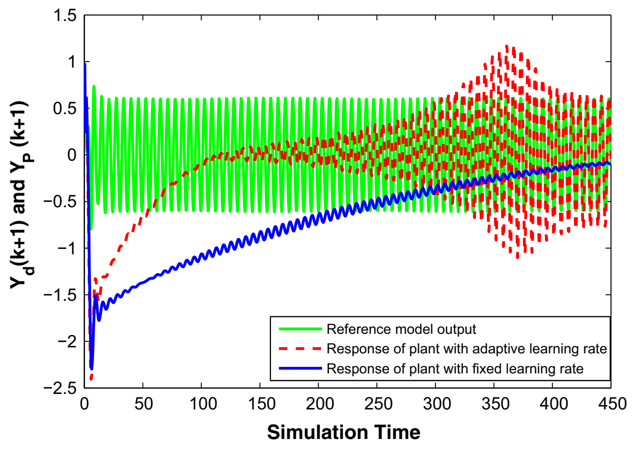


**Fig. 33** Response of plant under MLFFNN controller action at the initial stage of training (Example 3)

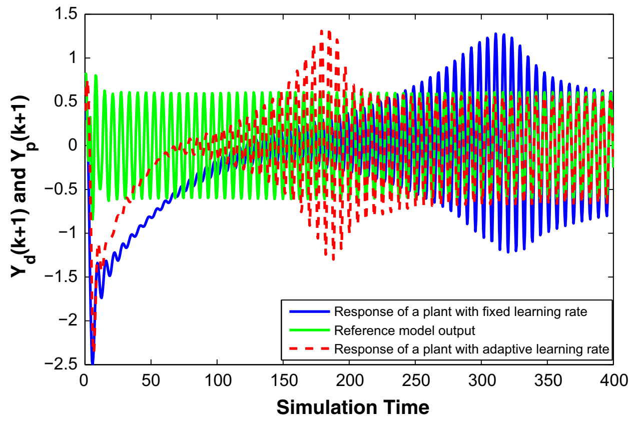


**Fig. 36** Response of plant under NARX controller action at the initial stage of training (Example 3)

Similarly, the outputs of the plant when NARX- and MLFFNN-based controllers are used in the loop are shown in Figs. 33, 34, 35, 36 and 37.

It can be seen from the figures that response of the plant improved at a much better rate when adaptive learning rate is used as compared to improvement obtained with fixed

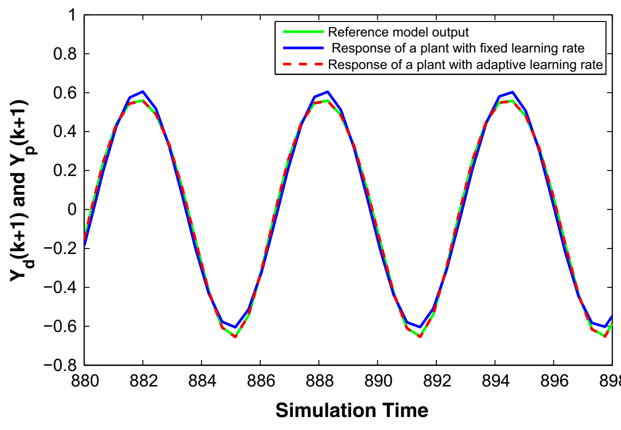


**Fig. 37** Response of plant under NARX controller action after certain amount of training (Example 3)

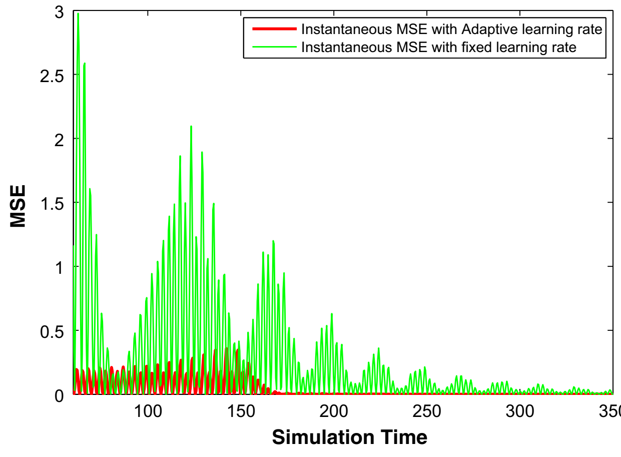


**Fig. 38** MSE of plant obtained under RBFN controller action (Example 3)

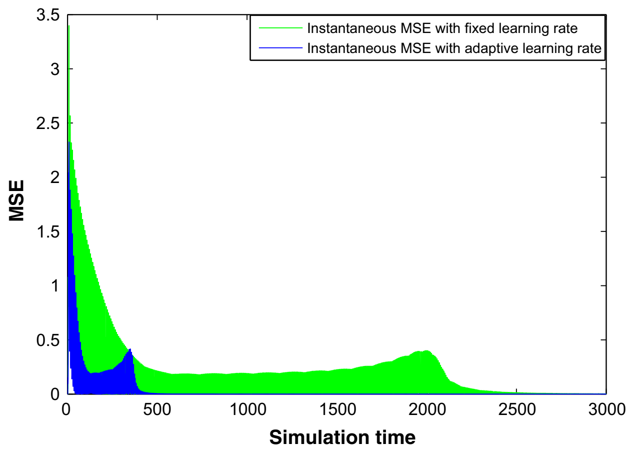


**Fig. 39** MSE of plant obtained under MLFFNN controller action (Example 3)

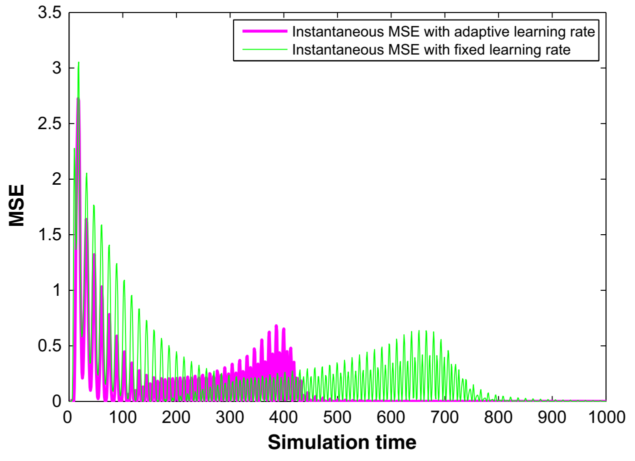


**Fig. 40** MSE of plant obtained under NARX controller action (Example 3)

learning rate. Also, it can be seen that response of the plant improved at a much better rate when RBFN is used as a controller. This can be inferred from the time axis. This conclusion can be further verified from the plots of instantaneous mean square error (MSE) which are shown in Figs. 38, 39 and 40, respectively.

It can be seen that MSE decreases down to zero very quickly when the adaptive learning rate is used as compared to MSE plot obtained with fixed learning rate. Among the controllers, RBFN has performed better than NARX- and MLFFNN-based controllers as its MSE took very less time to reduce to zero (with both fixed and with adaptive learning rate). Further, the time taken by control algorithm to run is listed in Table 5.

It can be seen that RBFN-based control program took least time in executing, whereas NARX and MLFFNN took more time as compared to RBFN. This makes RBFN more efficient than NARX- and MLFFNN-based controllers. The adaptive learning rate values associated with the output- and input-layer weight vectors of MLFFNN controller converges to 0.000850 and 0.0048. In NARX-based controller, they converged to 0.00049 and 0.00073 and in the RBFN case the final learning rate values associated with the output-layer weight vector, widths and radial centres converges to 0.000133, 0.00037 and 0.0031, respectively. The upper bounds of adaptive learning rate found in case of MLFFNN controller are 2.78 and 2.22 for output- and input-layer weights, respectively. In case of NARX-based controller, the upper bound is equal to 2.28 and 1.49. Finally, for the RBFN-based con-

**Table 5** Elapsed time for control program to run

| Type of controller | Time taken by program to run (in seconds) |
|---|---|
| RBFN | 1.24 |
| NARX | 13.62 |
| MLFFNN | 13.40 |

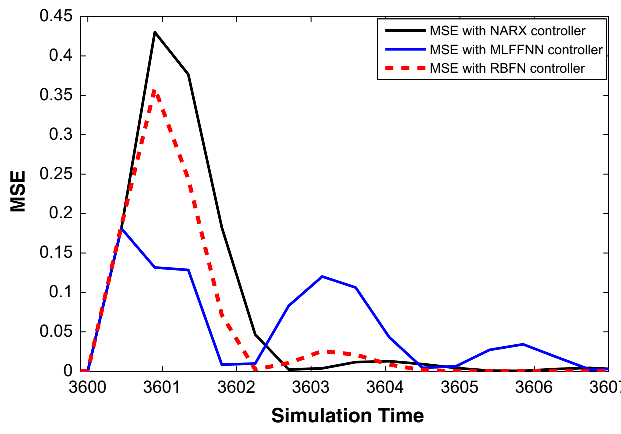


**Fig. 41** MSE of plant obtained at the instant of disturbance signal (Example 3)

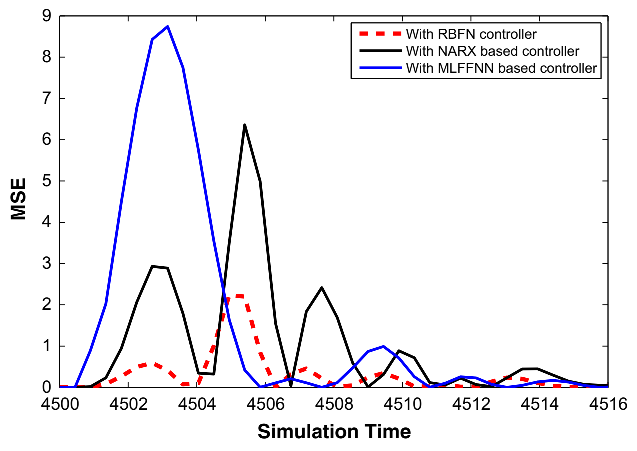


**Fig. 42** MSE of plant obtained at the instant of parameter variation (Example 3)

troller the upper bounds found for output-layer weight vector, widths and radial centres are 1.33, 2.16 and 1.61, respectively.

### 9.3.1 Robustness Testing

### 9.3.2 Disturbance Signal

A signal of 2.9 units is added in the output of the controller at $k = 3600$th time instant. This causes an increment in the MSE. The corresponding MSE plots obtained with different controllers with adaptive learning rate are shown in Fig. 41.

From the figure notice that MSE obtained with RBFN controller action took least time to reduce down to zero. This makes RBFN more robust than NARX- and MLFFNN-based controllers.

### 9.3.3 Parameter Variation

Now the values of weights in the output weight vectors of all the controllers are varied by an amount of 0.75 value. This leads to an increase in the MSE value. The corresponding MSE plots obtained with different controllers are shown in Fig. 42.

The performance of RBFN-based controller is again found superior as the MSE drops quickly down to zero after the impact of parameter variations.

### 9.3.4 Analysis of the Results: Example 3

In this example also, the performance of RBFN-based controller is found to be superior that of NARX- and MLFFNN-based controller. The plant's response improved quickly with RBFN-based controller and also recovered quickly from the effects of parameter variation and disturbance signal under RBFN controller action. The MSE obtained of plant under RBFN controller action also reduces quickly to zero during the online training with fixed as well as with the adaptive learning rate.

## 10 Discussion

After doing the simulation study, it can be easily concluded that RBFN-based controller has performed much better than that of the MLFFNN- and NARX-based controllers. The performances of controllers got improved when the adaptive learning rates are used in place of the fixed one. Also, the stability of the system remains intact during the online training. As far as the structural complexity is concerned, RBFN-based controller requires less number of parameters to be trained. Also, the computational time of RBFN is found to be least as compared to MLFFNN- and NARX-based controllers. This makes RBFN an efficient controller. Another reason of now choosing RBFN over MLFFNN- and NARX-based controllers is the robustness shown by it to the parameter variations and disturbance signals.

## 11 Conclusion

This paper involves a comparative study of feed-forward and recurrent type neural networks as adaptive controllers. The various parameters present in the neural networks were trained online using the dynamic back-propagation method. To make the method more powerful, an adaptive learning rate was developed using the Lyapunov stability method. It ensures the faster convergence of parameters and avoids the possibility of instability. The control scheme proposed in this paper is suitable even when the dynamics of the system are not known. Three simulation examples are considered in the simulation study, and the performances of these controllers (for controlling the complex nonlinear systems and to compensate the effects of system uncertainties) were tested and compared. The RBFN- based controller was found to be more computationally efficient, better in controlling the nonlinear systems and more robust in handling the system uncertainties as compared to NARX- and MLFFNN- based controllers.

**Funding** This study is not funded by any agency.

## Compliance with Ethical Standards

**Conflict of interest** The authors declare that they have no conflict of interest.

**Ethical Approval** This article does not contain any studies with human participants or animals performed by any of the authors.

## References

1. Narendra, K.S.: Identification and control of dynamical systems using neural networks. IEEE Trans. Neural Netw. **1**(1), 4–27 (1990)
2. Khalil, H.K.: Noninear systems. Prentice-Hall, New Jersey **2**(5), 5-1 (1996)
3. Wang, T.; Gao, H.; Qiu, J.: A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control. IEEE Trans. Neural Netw. Learn. Syst. **27**(2), 416–425 (2016)
4. Wei, Q.L.; Liu, D.R.: A novel policy iteration based deterministic q-learning for discrete-time nonlinear systems. Sci. China Inf. Sci. **58**(12), 1–15 (2015)
5. Ku, C.-C.; Lee, K.Y.: Diagonal recurrent neural network based control using adaptive learning rates. In: Decision and Control, 1992, Proceedings of the 31st IEEE Conference on, pp. 3485–3490. IEEE (1992)
6. Ge, H.-W.; Wen-Li, D.; Qian, F.; Liang, Y.-C.: Identification and control of nonlinear systems by a time-delay recurrent neural network. Neurocomputing **72**(13), 2857–2864 (2009)
7. Elmali, H.; Olgac, N.: Robust output tracking control of nonlinear MIMO systems via sliding mode technique. Automatica **28**(1), 145–151 (1992)
8. Sadati, N.; Ghadami, R.: Adaptive multi-model sliding mode control of robotic manipulators using soft computing. Neurocomputing **71**(13), 2702–2710 (2008)
9. Kanellakopoulos, I.; Kokotovic, P.V.; Morse, A.S.: Systematic design of adaptive controllers for feedback linearizable systems. IEEE Trans. Autom. Control **36**(11), 1241–1253 (1991)
10. Kokotovic, P.V.: The joy of feedback: nonlinear and adaptive. IEEE Control Syst. **12**(3), 7–17 (1992)
11. Liu, M.: Decentralized control of robot manipulators: nonlinear and adaptive approaches. IEEE Trans. Autom. Control **44**(2), 357–363 (1999)
12. Guo, Y.; Hill, D.J.; Wang, Y.: Nonlinear decentralized control of large-scale power systems. Automatica **36**(9), 1275–1289 (2000)
13. Nasr, M.B.; Chtourou, M.: Neural network control of nonlinear dynamic systems using hybrid algorithm. Appl. Soft Comput. **24**, 423–431 (2014)
14. Kumar, R.; Srivastava, S.; Gupta, J.R.P.: Diagonal recurrent neural network based adaptive control of nonlinear dynamical systems using Lyapunov stability criterion. ISA Trans. **67**, 407–427 (2017)
15. Haykin, S.; Neural Network: A comprehensive foundation. Neural Netw. **2**(2004), 41 (2004)
16. Nørgaard, P.M.: System identification and control with neural networks. Department of Automation, Technical University of Denmark (1996)
17. Liu, G.P.: Nonlinear Identification and Control: A Neural Network Approach. Springer, Berlin (2012)
18. Zilouchian, A.; Jamshidi, M.: Intelligent Control Systems Using Soft Computing Methodologies. CRC Press, Inc., Boca Raton (2000)
19. He, W.; Chen, Y.; Yin, Z.: Adaptive neural network control of an uncertain robot with full-state constraints. IEEE Trans. Cybern. **46**(3), 620–629 (2016)
20. Nelles, O.: Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models. Springer, Berlin (2013)
21. Levin, A.U.; Narendra, K.S.: Control of nonlinear dynamical systems using neural networks: controllability and stabilization. IEEE Trans. Neural Netw. **4**(2), 192–206 (1993)
22. Sastry, P.S.; Santharam, G.; Unnikrishnan, K.P.: Memory neuron networks for identification and control of dynamical systems. IEEE Trans. Neural Netw. **5**(2), 306–319 (1994)
23. Noriega, J.R.; Wang, H.: A direct adaptive neural-network control for unknown nonlinear systems and its application. IEEE Trans. Neural Netw. **9**(1), 27–34 (1998)
24. Nouri, K.; Dhaouadi, R.; Braiek, N.B.: Adaptive control of a nonlinear dc motor drive using recurrent neural networks. Appl. Soft Comput. **8**(1), 371–382 (2008)
25. Fourati, F.; Chtourou, M.; Kamoun, M.: Stabilization of unknown nonlinear systems using neural networks. Appl. Soft Comput. **8**(2), 1121–1130 (2008)
26. Zhang, H.; Luo, Y.; Liu, D.: Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints. IEEE Trans. Neural Netw. **20**(9), 1490–1503 (2009)
27. Yu, L.; Fei, S.; Long, F.; Zhang, M.; Yu, J.: Multilayer neural networks-based direct adaptive control for switched nonlinear systems. Neurocomputing **74**(1), 481–486 (2010)
28. Wu, Y.; Shen, L.; Zhang, L.: Study on nonlinear pH control strategy based on external recurrent neural network. Proc. Eng. **15**, 866–871 (2011)
29. Zhao, W.; Ren, X.: Neural network-based sliding mode control for dual-motor servo systems. IFAC Proc. **46**(20), 382–386 (2013)
30. Rajesh, RJ.; Preethi, R.; Mehata, P.; Pandian, B.J.: Artificial neural network based inverse model control of a nonlinear process. In: Computer, Communication and Control (IC4), 2015 International Conference on, pp. 1–6. IEEE (2015)
31. Aouiti, C.: Oscillation of impulsive neutral delay generalized high-order hopfield neural networks. Neural Comput. Appl., 1–19 (2016). https://doi.org/10.1007/s00521-016-2558-3
32. Aouiti, C.; Mhamdi, M.S.; Touati, A.: Pseudo almost automorphic solutions of recurrent neural networks with time-varying coefficients and mixed delays. Neural Process. Lett. **45**(1), 121–140 (2017)
33. Aouiti, C.; Mhamdi, M.S.; Cao, J.; Alsaedi, A.: Piecewise pseudo almost periodic solution for impulsive generalised high-order hopfield neural networks with leakage delays. Neural Process. Lett. **45**(2), 615–648 (2017)
34. Aouiti, C.: Neutral impulsive shunting inhibitory cellular neural networks with time-varying coefficients and leakage delays. Cogn. Neurodyn. **10**(6), 573–591 (2016)
35. Yoo, S.J.; Choi, Y.H.; Park, J.B.: Generalized predictive control based on self-recurrent wavelet neural network for stable path tracking of mobile robots: adaptive learning rates approach. IEEE Trans. Circuits Syst. I Regul. Pap. **53**(6), 1381–1394 (2006)
36. Yoo, S.J.; Park, J.B.; Choi, Y.H.: Indirect adaptive control of nonlinear dynamic systems using self recurrent wavelet neural networks via adaptive learning rates. Inf. Sci. **177**(15), 3074–3098 (2007)
37. ul Amin, R.; Aijun, L.; Khan, M.U.; Shamshirband, S.; Kamsin, A.: An adaptive trajectory tracking control of four rotor hover vehicle

using extended normalized radial basis function network. Mech. Syst. Signal Process. **83**, 53–74 (2017)
38. Zhao, X.; Shi, P.; Zheng, X.; Zhang, J.: Intelligent tracking control for a class of uncertain high-order nonlinear systems. IEEE Trans. Neural Netw. Learn. Syst. **27**(9), 1976–1982 (2016)
39. Noble, D.; Bhandari, S.: Neural network based nonlinear model reference adaptive controller for an unmanned aerial vehicle. In: Unmanned Aircraft Systems (ICUAS), 2017 International Conference on, pp. 94–103. IEEE (2017)
40. de Jesús Rubio, J.; Zhang, L.; Lughofer, E.; Cruz, P.; Alsaedi, A.; Hayat, T.: Modeling and control with neural networks for a magnetic levitation system. Neurocomputing **227**, 113–121 (2017)
41. Gonzlez-Olvera; Marcos, A.; Tang, Y.: Modeling, identification and control based on recurrent neural networks of a class of nonlinear systems. IFAC Proc. Vol. **42**(10), 1511–1516. 15th IFAC Symposium on System Identification (2009)
42. Kumar, R.; Srivastava, S.; Gupta, J.R.P.: Lyapunov stability-based control and identification of nonlinear dynamical systems using adaptive dynamic programming. Soft Comput. **21**(15), 4465–4480 (2017)
43. Agand, P.; Shoorehdeli, M.A.; Khaki-Sedigh, A.: Adaptive recurrent neural network with Lyapunov stability learning rules for robot dynamic terms identification. Eng. Appl. Artif. Intell. **65**, 1–11 (2017)
44. Bakefayat, A.S.; Tabrizi, M.M.: Lyapunov stabilization of the nonlinear control systems via the neural networks. Appl. Soft Comput. **42**, 459–471 (2016)
45. de Jesus, R.J.: Discrete time control based in neural networks for pendulums. Appl. Soft Comput. (2017). https://doi.org/10.1016/j.asoc.2017.04.056
46. Aftab, M.S.; Shafiq, M.: Neural networks for tracking of unknown SISO discrete-time nonlinear dynamic systems. ISA Trans. **59**, 363–374 (2015)
47. Behera, L.; Kar, I.: Intelligent Systems and Control Principles and Applications. Oxford University Press, Inc., Oxford (2010)
48. Ku, C.-C.; Lee, K.Y.: Diagonal recurrent neural networks for dynamic systems control. IEEE Trans. Neural Netw. **6**(1), 144–156 (1995)
49. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J.: Learning Internal Representations by Error Propagation, Technical report. California Univ San Diego La Jolla Inst for Cognitive Science (1985)
50. Phansalkar, V.V.; Sastry, P.S.: Analysis of the back-propagation algorithm with momentum. IEEE Trans. Neural Netw. **5**(3), 505–506 (1994)
51. Polycarpou, M.M.; Ioannou, P.A.: Identification and Control of Nonlinear Systems Using Neural Network Models: Design and Stability Analysis. University of Southern California, Los Angeles (1991)
52. Zurada, J.M.: Introduction to Artificial Neural Systems, vol. 8. West, St Paul (1992)
53. Liao, T.L.; Horng, J.H.: Adaptive control of a class of nonlinear discrete-time systems using hybrid neural networks. In: Control Conference (ECC), 1997 European, pp. 506–511. IEEE (1997)
54. Yang, G.-B., Donath, M.: Dynamic model of a one-link robot manipulator with both structural and joint flexibility. In: Robotics and Automation, 1988. Proceedings, 1988 IEEE International Conference on, pp. 476–481. IEEE (1988)
55. Paul, R.: Robot Manipulators: Mathematics, Programming, and Control: The Computer Control of Robot Manipulators. MIT Press, Cambridge, MA (1981)
56. Sciavicco, L.; Siciliano, B.: Modelling and Control of Robot Manipulators. Springer, Berlin (2012)
57. Sanxiu, W.; Shengtao, J.: Adaptive friction compensation of robot manipulator. In: Hu, W. (ed.) Electronics and Signal Processing, pp. 127–134. Springer, Berlin (2011)