Soft Comput (2017) 21:4447–4463
DOI 10.1007/s00500-016-2447-9

METHODOLOGIES AND APPLICATION

Modeling and adaptive control of nonlinear dynamical systems using radial basis function network

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

Published online: 29 November 2016

Abstract In this paper, the use of radial basis function network (RBFN) for simultaneous online identification and indirect adaptive control of nonlinear dynamical systems is demonstrated. The motivation of using RBFN comes from the simplicity of its structure and simpler mathematical formulation, which gives it an advantage over multi-layer feed-forward neural network (MLFFNN). Since most processes are nonlinear, the use of conventional proportional-integral-derivative controller is not useful. Most of the time plant's dynamics information is not available. This creates another limitation on the use of conventional control techniques, which works only if plant's dynamics information is available. The proposed controller is tested for parameter variations and disturbance effects. Simulation results showed that RBFN is able to capture the unknown dynamics as well as simultaneously able to adaptively control the plant. It is also found to compensate the effects of parameter variations and disturbances. The comparative analysis is also done with MLFFNN in each simulation example, and it is found that performance of RBFN is better than that of MLFFNN.

Keywords Radial basis function network · Nonlinear system identification and control · Gradient descent principle · Multi-layer feed-forward neural network · Robustness

Communicated by V. Loia.

✉ Rajesh Kumar
rajeshmahindru23@gmail.com

Smriti Srivastava
smriti.nsit@gmail.com

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

[1] Netaji Subhas Institute of Technology, Sector 3, Dwarka, New Delhi, India

1 Introduction

A lot of progress in recent years has witnessed the identification and modeling of nonlinear dynamical systems due to great demands for controller design (Qiao and Han 2012). Due to the lack of knowledge of some parameters of given system, the formulation of mathematical model of the system is very difficult. RBFNs are known to have various advantages like strong immunity to input noise, simpler topology and good generalizations (Fu and Chai 2007; Rossomando et al. 2011). This makes them very suitable for identification and control applications. They are being widely used in various areas like modeling and identification (Attaran et al. 2016), pattern recognition (Jankowski and Kadirkamanathan 1997), but their potential as a controller is not fully exploited in control applications. Multi-layer feed-forward neural network (MLFFNN) has also shown good results (Kayacan et al. 2015) as they have the ability to approximate the nonlinear dynamics of the plant (Bishop 1995; Wang et al. 2016). However, MLFFNN has poor process interpretability and other problems like slow learning and sometimes stucking at local minima (Srivastava et al. 2005). The ability of RBFN to approximate nonlinear mapping directly from the input–output data has made them popular in recent times. RBFNs offer several advantages over multi-layer feed-forward network like smaller extrapolation errors, higher reliability, simpler structure and faster convergence and have increasingly attracted the interest for various engineering applications. MLFFNNs work globally, since the output of output layer neuron(s) is/are decided by all the neurons in the network (Haykin and Network 2004), but this is not the case in RBFNs as they are local approximation networks (Yu et al. 2011), which means the network output in them is determined by specified radial centres in a certain local receptive fields. At present, radial RBFN provides an alternate approach (Seng et al. 1998) to conventional MLFFNN. They have diverse applications and are probably the main rival to the multi-layer feed-forward neural network (Moody and Darken 1989; Poggio and Girosi 1990). RBFN has been used to identify the optimal controller gain for fractional-order (Perng et al. 2016) proportional-integral-derivative (FOPID) controller of time-delay systems. In Sutrisno et al. (2015), a novel self-organizing quasi-linear ARX radial basis function network model (SOQARX–RBFN) is proposed to improve the prediction accuracy of the QARX–RBFN model. This improved version is then applied to the identification and control of nonlinear dynamical systems. In Hsu et al. (2013), a supervisory adaptive dynamic RBF-based neural fuzzy controller is proposed for the nonlinear dynamical systems. Dai et al. (2012) used RBFN-based controller to solve the learning control problem for ocean surface ship in uncertain dynamical environments. In Almaadeed et al. (2015), RBFN has been applied to speaker identification problem.

In the paper (Qiu et al. 2016b), a Takagi–Sugeno (T–S) fuzzy-based modeling of nonlinear systems has been done. The issues like loss of data due to failure of links (known as data packet dropouts) or sensors, presence of time delays are addressed. The interesting thing is that whether the controller will receive the output of plant or not or plant receives the controller output or not are also modelled. The common feature which fuzzy-based system and RBFN share is that both involve parameters (in case of RBFN, they are radial centres, output weights, width of each radial centres. For fuzzy systems, they are membership function parameters). These parameters are allowed to adjust (and change continuously) at each instant based on the error value received at the same instant during the training. The main purpose of modeling of any system is to have an equivalent mathematical representation of its dynamics. If RBFN is chosen, then after the sufficient training the dynamics of given system will be captured in the form of parameter values of RBFN. On the other hand, if fuzzy-based system is used, then parameters of membership function will have the information about the dynamics of the plant. So, after a sufficient training of RBFN or fuzzy-based system, we can expect to have their outputs at any instant to be equal to that of given system's output at the same instant. In Qiu et al. (2013a, b, 2016a), the nonlinear systems have been modelled using Takagi–Sugeno fuzzy models with parameter uncertainties. In the paper (Qiu et al. 2015), input–output approach has been adopted to analyse the time-delay-dependent stability and design of controller for class of continuous-time Markovian jump linear systems.

In our present paper, RBFN has been used for implementing the identification model as well as the controller. RBFNs are represented by a network-like structure, similar to that of a perceptron. RBFN can approximate dynamics of any nonlinear system, a feature which is utilized in system identification (Chen et al. 1992). Their potential to act as a controller as well as an identification tool is exploited in this paper. RBFN is very popular among the researchers, and during last two decades, many researchers have been working towards developing more efficient (Gomm and Yu 2000; Sarimveis et al. 2003) training algorithms and potential applications.

1.1 Novelties and contributions of the paper

1. In this paper, a thorough testing of RBFN for identification and control purpose is done by considering plant's of different complexity. Initial and final plant's responses under the action of RBFN identification model and RBFN controller are shown, which clearly depicts the learning of RBFN parameters.
2. The online control and identification configuration based on RBFN are proposed. The advantage of this scheme over other schemes is that there is no need to have knowledge of plant's dynamics. Moreover, the effects of plant's uncertainties can be compensated very quickly, thus maintaining the desired response from the output of the plant.
3. The criteria for selecting the inputs to the RBFN controller is proposed which is, to the best of our knowledge, is not available in the literature.
4. Thorough analysis of robustness of both RBFN identification model and controller is performed. Both parameter variation and disturbance signals effects are considered. Such a detailed analysis is not available in the literature.
5. Detailed computation of RBFN parameter's update equations is given.

The remaining paper is organized as follows: In Sect. 2, RBFN structure and its defining mathematical equations are given. The superiority of RBFN over MLFFNN in terms of mathematical computation is also explained. In Sect. 3, practical considerations for using RBFN are addressed. Section 4 discusses the computation complexities for RBFN. In Sect. 5, four different models/classes to which any nonlinear plant can belong are given, and the two different approaches of identification procedures are given. Section 6 starts with the derivation of update equations for updating parameters of RBFN using gradient descent, and then, simulation examples are given to demonstrate the identification process. In Sect. 7, the use of RBFN as a controller is proposed. Here, both identification and control are implemented simultaneously by using two RBFNs in which one act as a controller and other as an identifier. Necessary equations for updating parameters of RBFN controller are given, and with the help of three simulation examples containing plants of different complexities (in terms of dynamics), simultaneous online identification and control using RBFNs are illustrated. In each example, the comparison of performances of RBFN with MLFFNN is made. After that the robustness of proposed RBFN controller in each of the examples is checked in terms of parameter variations and noise/disturbance signals. In Sect. 8, the achievements of this paper are given, and Sect. 9 includes future scope and conclusion.

2 Structure of radial basis function network

Radial basis functions were first used in the design of neural networks by Broomhead and Lowe (1988). It is like a MLFFNN but contains only a single hidden layer, and the functionality of nodes present in the hidden layer of RBFN is different from the neurons present in the MLFFNN. This makes RBFN different from the MLFFNN. The structure of RBFN is fixed unlike MLFFNN, which gives it an advantage over the MLFFNN in terms of structure complexity (Behera and Kar 2010; Elanayar et al. 1994). Further, the computation time in case of RBFN is less as compared to MLFFNN because there is no back propagation of error unlike the supervised learning in a MLFFNN. For single-input single-output (SISO) plants, there is only a single neuron in the output layer of RBFN. The hidden layer transforms the p-dimensional input vector (meaning p-inputs in one training input sample) into a high-dimensional feature space using Gaussian radial function. The output of output layer in RBFN is simply a weighted sum of the inputs received from hidden layer. The nodes in the hidden layer of RBFN are called as radial centres/hidden unit (Lowe 2015). Each radial centre is equal to one of the input training sample. It has its own receptive field in the input space. Thus, each radial centre represents all those input samples which are presented in its vicinity, and this is termed as 'local representation of inputs'. So each radial centre is also a p-dimensional vector. If given input vector lies in the vicinity of some radial centre, then this will activate that radial centre (which means large output given by that radial centre) and by proper choice of weights in output layer, the desired output from RBFN can be obtained. If the given input vector lies between the receptive fields of two radial centres, then the output of both of these radial centres will be appreciably high. The number of radial centres considered in hidden layer is usually less than the total number of training samples. The data for training of RBFN are not always present beforehand because in majority of the cases the data are coming online, and hence, to do online training, principle of gradient descent is used. The structure of RBFN is shown in Fig. 1. In the structure of RBFN which is shown in Fig. 1, the first layer is the input layer $X = (x_1, x_2, \ldots, x_p)$, which is a $p \times 1$ vector, that is, p number of inputs are there in each training

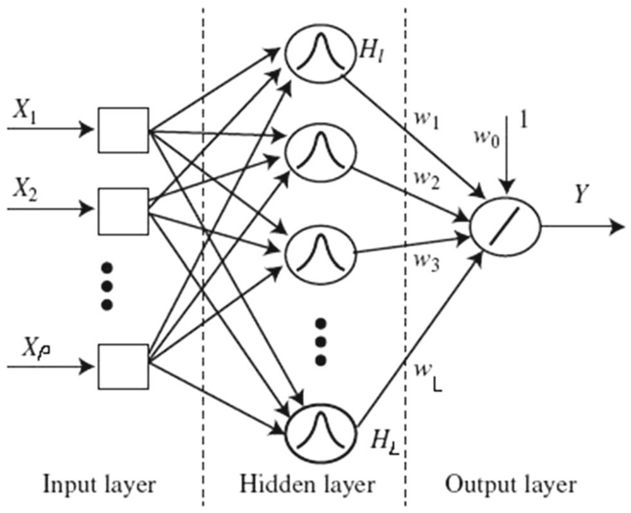

Fig. 1 RBFN structure

sample. The radial centre vector $H = (H_1, H_2, \ldots, H_L)$ consists of L number of radial centres where L is always $\leq p$ and constitutes the hidden layer. The values of each weight connecting input layer to the hidden layer are unity. The output weight vector is (for single-input single-output system) $w = [w_1, w_2, w_3, \ldots, w_L)$ (and the bias). The activation function of output neuron is taken to be linear. In short, the input is clustered around the radial centres, and the output is linear in terms of weights, w

$$y_R(k) = \sum_{i=1}^{L} \phi_i(k) w_i(k) \tag{1}$$

where $y_R(k)$ is RBFN output at kth time instant. The response of the ith radial centre at kth time instant in RBFN is usually expressed by the following expression

$$\phi_i(k) = \phi \, \|X(k) - H_i(k)\| \tag{2}$$

where $\|X(k) - H_i(k)\|$ is the Euclidean distance between X(k) and $H_i(k)$ at kth time instant. The $\phi\,(\cdot)$ is a radial basis function and is generally taken to be Gaussian

$$\phi\,(z) = \exp\left(\frac{-z^2}{2\sigma^2}\right) \tag{3}$$

where $z = \|X(k) - H_i(k)\|$ and $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_L)$ represents the width of each radial centre. In case of Gaussian radial function, each node produces an identical output for input within a fixed radial distance from the centre, i.e. they are radially symmetric. That is why they are called as radial basis function (Behera and Kar 2010). The training of RBFN involves the adjustment of radial centres in hidden layers, adjustment of widths of each radial centre in the hidden layer and optimizing the output layer weight vector such that the cost function (instantaneous mean square error) gets minimized. However, the performance of RBFN (Behera and Kar 2010) is not much sensitive to the accurate and precise value of σ. If the plant's input–output data are available beforehand (which is not always the case), then offline training can be used. In offline training, radial centres and widths are fixed and adjustment of only output weight vector is done by the application pseudo-inverse technique (Paul et al. 2015). The work done in this paper is concerned with online training (Soudry et al. 2015) of RBFN, and method of gradient descent is used to get its parameters learned.

3 Issues to be addressed while using RBFN

There are certain issues which need to be addressed for using RBFN. These include

1. Which radial function ϕ should be used.
2. How many radial centres must be chosen so that desired results can be obtained.
3. What should be the initial locations for radial centres.

Usually Gaussian radial function is used. Other choices include quadratic, inverse quadratic and thin plate of spline (Schultz 1973). Among these radial functions, the Gaussian function is very popular because it is more intuitive in a sense that if the input lies in the neighbourhood of radial centre, then the corresponding radial basis function output will be large. However, thin plate of spline given by

$$\phi(z) = z^2 \ln z \tag{4}$$

can also work well in practice and can be recommended. On the other hand, direct links are provided by Gaussian form to fuzzy logic systems (Torshizi et al. 2015) and cluster analysis, and hence, each of the basis function centre positions has significance and has a physical meaning attached to them. Each radial basis function has its own certain properties, and for a given problem, some radial functions ϕ are more suited than the others. To get best results, optimum number of radial centres must be chosen as too little or too many will give undesirable results. This choice depends upon the amount of training data, and this dependency is of proportional type. The best initial locations of radial centres would be those places in input space where more training data are present (Ayala and Santos Coelho 2016). These issues need to be taken care of for using the RBFN. To position the radial centres in a self-organizing fashion, techniques such as k-means clustering algorithm or the mean-tracking cluster algorithm can be used.

4 Remark on computation complexities for RBFN

The number of computational steps required in case of RBFN is as follows:

1. Calculation of Euclidian distance between the input vectors and radial centres.
2. Computation of each radial centre output using Gaussian radial basis function.
3. Multiplication of each radial centre output with a weight (of output weight vector).
4. Calculation of induced field of output neuron (weighted sum of inputs for output neuron). Since activation function for output neuron is linear, output of output neuron is equal to its own induced field.

But in case of MLFFNN there is one additional computation need to be performed which is the multiplication of external input vector with the input weights (which connects external inputs to the neurons present in the hidden layer).

So, if the training continued for 2000 iterations, then it means 2000 extra computations will be performed in MLFFNN as compared to computations required by RBFN. The other added benefit of using RBFN is that its structure RBFN is fixed containing only one single hidden layer, whereas in case of MLFFNN it is not always like that. So, increasing the number of hidden layer will proportionally increase the number of computational steps required by MLFFNN and hence will make the system complex.

5 Identification of dynamics of nonlinear dynamical system using RBFN

Many control schemes work only if good approximate mathematical model of a given system to be controlled is available. For any such scheme to perform satisfactorily, the quality of system model obtained through identification procedure must be of a certain standard. So, system identification procedure must be carefully chosen. The process of identification involves choosing the model structure to which the given system/plant belongs and approximation to its order and then setting up the same structure for RBFN model to approximate the unknown dynamics of given plant/system. The nonlinear dynamical plant/system may belong to any of the four models introduced here by the following nonlinear difference equations (Narendra and Parthasarathy 1992):

$$\text{Model 1: } y_p(k+1) = \left[\sum_{i=0}^{n-1} a_i y_p(k-i)\right] + G\left[u(k), u(k-1) \cdots u(k-m+1)\right] \tag{5}$$

$$\text{Model 2: } y_p(k+1) = F\left[y_p(k), y_p(k-1)\cdots y_p(k-n+1)\right] + \left[\sum_{i=0}^{m-1} b_i u(k-i)\right] \quad (6)$$

$$\text{Model 3: } y_p(k+1) = F\left[y_p(k), y_p(k-1)\cdots y_p(k-n+1)\right] + G\left[u(k), u(k-1)\cdots u(k-m+1)\right] \quad (7)$$

$$\text{Model 4: } y_p(k+1) = F\left[y_p(k), y_p(k-1)\cdots y_p(k-n+1), u(k), u(k-1)\cdots u(k-m+1)\right] \quad (8)$$

The models shown above are used for representing the discrete time systems. Continuous analogs of these can be described by using the differential equations. Symbol n represents the order of the system/plant and $m \leq n$. F and G represent nonlinear functions. All the four models represent the dynamical system in which plant's output at $(k+1)$th instant depends upon present value of plant's output and external input as well as their past values (Narendra and Parthasarathy 1990). The identification process requires the knowledge of two things:

1. The class/model to which the given plant belongs.
2. The estimate of plant's order (value of n).

The RBFN will be used to approximate the nonlinearity present in the plant. The problem of identification is to set up a suitably parameterized identification model (RBFN) and then adjusting its associated parameters based on the instantaneous mean square error between the plant's output and identification model output. The structure of identification model chosen is same as the mathematical structure of plant. But certain things, as shown in what follows, must be kept in mind to ensure that process of identification leads to the convergence (Wei and Liu 2015) of identification model's parameters to their desired value.

5.1 Identification in parallel mode

Figure 2 shows a plant which belongs to model 1 with $n = 2$ and $m = 1$ and depends nonlinearly on external input u. This nonlinear dependence is denoted by F. For identification purpose, the identification model is also given in Fig. 2. It will approximate the nonlinear dependence of plant's output on external input u. The RBFN identification model is described by equation

$$\hat{y}_R(k+1) = \hat{a}_0\hat{y}_R(k) + \hat{a}_1\hat{y}_R(k-1) + \text{RBFN}(u(k)) \quad (9)$$

where $\hat{y}_R(k+1)$ is RBFN output at $(k+1)$th instant and $\hat{a}_0$, $\hat{a}_1$ are parameters of plant which are to be estimated using the gradient descent.

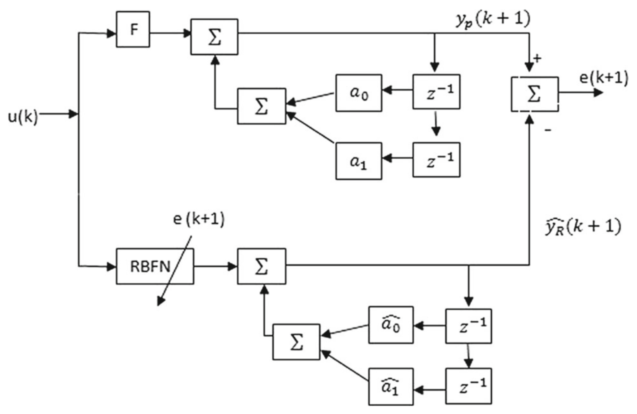

Fig. 2 Parallel identification model

Figure 2 represents the parallel mode in which RBFN uses its own present and past values along with external input $u(k)$ to compute the $\hat{y}_R(k+1)$ value. Identification thus involves the estimation of parameters $\hat{a}_0$, $\hat{a}_1$ and adjustment of RBFN parameters. All the plants used in this paper are assumed to be bounded-input bounded-output (BIBO) stable in the presence of an input (of known class), which means all the signals in the plants will also be BIBO stable. But stability of RBFN identification model is not guaranteed (Cao and Liang 2004) and so using its own output values may lead to instability and hence identification process will not be complete. Hence, for plant representations using models 1–4, series–parallel mode will be used.

It is to be clear that the mathematical structure assumed for RBFN identification model is same as that of the plant. In the present discussion, we considered plant to belong to the model 1. In this model, the known part in the mathematical equation includes plant's present and past output value, and the unknown part is its dependency on the external input u. So, the overall output of RBFN identification model would also be the sum of known part plus the unknown part approximated by RBFN. Thus, in parallel identification mode, RBFN present and past output values plus the F approximated by it (represented as RBFN $[u(k)]$) would constitute the overall output of RBFN identification model. In case of series–parallel mode, for model 1 case, present and past output values of plant plus the F approximated by RBFN would constitute the overall output of the RBFN identification model.

5.2 Identification in series–parallel mode

In this mode, plant's output is fed back to identification model for computing the identification model (RBFN) output. Figure 3 represents the series–parallel identification model. The advantage of using series–parallel mode (Narendra and Parthasarathy 1990) is that since the given plant is

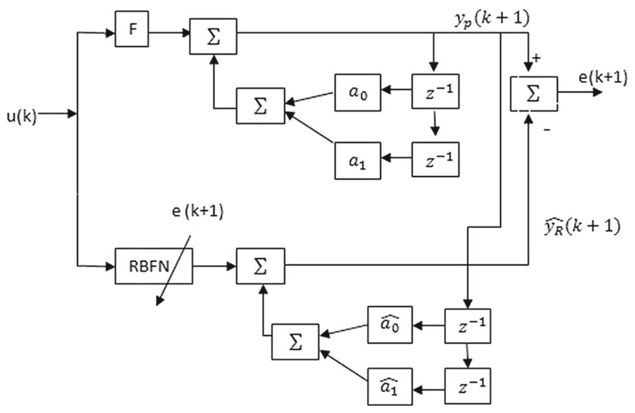

Fig. 3 Series–parallel identification model

assumed to be BIBO stable, all the signals used in the identification process (inputs of the RBFN) will also be bounded. This leads to the following form of the identification model:

$$\hat{y}_R(k+1) = \hat{a}_0 y_p(k) + \hat{a}_1 y_p(k-1) + \text{RBFN}(u(k)) \quad (10)$$

6 Identification algorithm using RBFN

One of the powerful tools to train the parameters of RBFN is the gradient descent algorithm. The same principle is also applied for updating the weights of MLFFNN-based identification model and controller. This is done in order to have a fair comparison of performance evaluation of RBFN and MLFFNN. Here, the algorithm is used in incremental mode (that is sample by sample) because many times the training data are not available in advance but is coming online and hence incremental type (Huang et al. 2006) of training needs to be done. Let $E(k)$ (instantaneous mean square error) denotes instantaneous cost function value at kth instant and is given by

$$E(k) = \frac{1}{2}\left(y_p(k) - y_R(k)\right)^2 \quad (11)$$

where $y_p(k)$ is the desired output (plant's output whose dynamics needs to be identified) and $y_R(k)$ is the output of RBFN at kth time instant. Differentiating $E(k)$ with respect to $H_{ij}(k)$ will provide the rate at which $E(k)$ undergoes change with respect to every element present in each radial centre, where $i = 1$ to P, $j = 1$ to L, and this requires the use of chain rule as number of signals are between $E(k)$ and $H_{ij}(k)$.

$$\frac{\partial E(k)}{\partial H_{ij}(k)} = \left(\frac{\partial E(k)}{\partial y_R(k)} \times \frac{\partial y_R(k)}{\partial \phi_j(k)} \times \frac{\partial \phi_j(k)}{\partial H_{ij}(k)}\right) \quad (12)$$

where $\frac{\partial E(k)}{\partial y_R(k)} = -\text{e}(k)$, $\frac{\partial \phi_j(k)}{\partial z_j(k)} = -z_j(k) \times \frac{\phi_j(k)}{\sigma_j^2(k)}$ and $\frac{\partial z_j(k)}{\partial H_{ij}(k)} = \left(\frac{\partial \sum_j ((x_j(k) - H_{ij}(k))^2)^{\frac{1}{2}}}{\partial H_{ij}(k)}\right)$

On simplification, we will get

$$\frac{\partial z_j(k)}{\partial H_{ij}(k)} = -\left(\frac{x_j(k) - H_{ij}(k)}{z_j(k)}\right) \quad (13)$$

Thus, update equation for radial centres is

$$H_{ij}(k+1) = H_{ij}(k) + \Delta H_{ij} \quad (14)$$

where

$$\Delta H_{ij} = \eta_1 e(k) w_j(k) \frac{\phi_j(k)}{\sigma_j^2(k)} \left(x_j(k) - H_{ij}(k)\right) \quad (15)$$

Update rule for weights can be evaluated in a same fashion as

$$\frac{\partial E(k)}{\partial w_j(k)} = \left(\frac{\partial E(k)}{\partial y_R(k)} \times \frac{\partial y_R(k)}{\partial w_j(k)}\right) \quad (16)$$

where $\frac{\partial E(k)}{\partial y_R(k)} = -(y_p(k) - y_R(k)) = -\text{e}(k)$ and $\frac{\partial y_R(k)}{\partial w_j(k)} = \phi_j(k)$. Thus, each element in $w = [w_1, w_2, w_3, \ldots, w_L)$ is updated as

$$w_j(k+1) = w_j(k) + \Delta w_j \quad (17)$$

where $\Delta w_j = \eta_2 e(k) \phi_j(k)$ and η_2 is the learning rate and is set between 0 and 1. Similarly, update equations for radial centres width's are obtained by applying the chain rule

$$\frac{\partial E(k)}{\partial \sigma_j(k)} = \left(\frac{\partial E(k)}{\partial y_R(k)} \times \frac{\partial y_R(k)}{\partial \phi_j(k)} \times \frac{\partial \phi_j(k)}{\partial \sigma_j(k)}\right) \quad (18)$$

where $\frac{\partial E(k)}{\partial y_R(k)} = -(y_p(k) - y_R(k)) = -\text{e}(k)$, $\frac{\partial y_R(k)}{\partial \phi_j(k)} = w_j(k)$ and $\frac{\partial \phi_j(k)}{\partial \sigma_j(k)} = \frac{\phi_j(k) z_j^2(k)}{\sigma_j^3(k)}$. So, each element in $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_L)$ is updated as

$$\sigma_j(k+1) = \sigma_j(k) + \Delta\sigma \quad (19)$$

where

$$\Delta\sigma = \eta_3 e(k) w_j(k) \frac{\phi_j(k) z_j^2(k)}{\sigma_j^3(k)} \quad (20)$$

where η_3 is the learning rate and its value lies between 0 and 1. Before proceeding further, two simulation-based identification examples using RBFN are given in the next section.

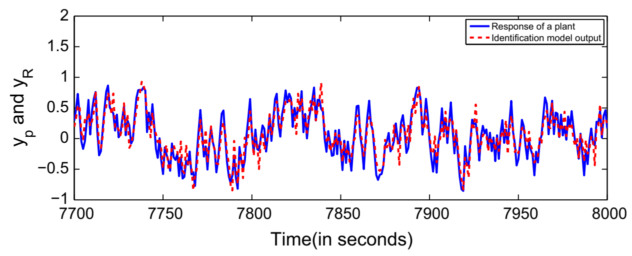

Fig. 4 Identification by RBFN during the training

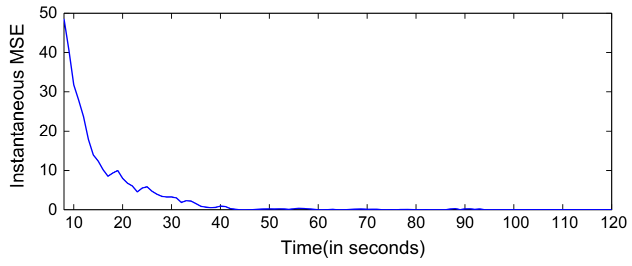

Fig. 5 Instantaneous MSE during the training process

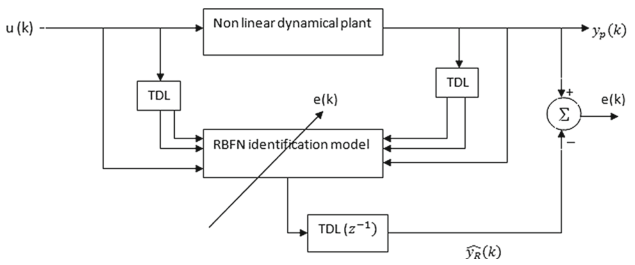

Fig. 6 Identification of dynamics of nonlinear plant using RBFN

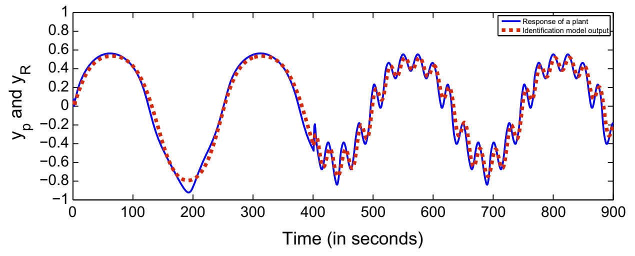

Fig. 7 Validation of RBFN identification model after identification process is over

6.1 Identification Example 1

The difference equation describing the dynamics of plant (Narendra and Parthasarathy 1990) to be identified is given by

$$y_p(k+1) = \frac{y_p(k)y_p(k-1)y_p(k-2)u(k-1)}{1 + y_p^2(k-2) + y_p^2(k-1)} \tag{21}$$

The given plant belongs to model 4. The learning rates η_1, η_2 and η_3 were all set to a value of 0.0045. Number of radial centres taken is 20. The training was continued up to 50,000 time steps using a random input signal with amplitude uniformly distributed in the interval $[-1, 1]$. Red curve in Fig. 4 shows the identification model response, and blue curve shows the response of plant. It can be seen from the figure that RBFN identification model is able to capture the dynamics of the plant. Figure 5 shows the instantaneous mean square error during the training. As the training progresses, the MSE decreases showing the learning ability of RBFN. The series–parallel RBFN identification model corresponding to a nonlinear dynamical plant represented by model 4 is shown in Fig. 6. TDL in Fig. 6 denotes tapped delay lines whose output vector constitutes the delayed values of its input signal. During the identification, series–parallel mode was selected but after the identification process is over the performance of RBFN identification model is tested using parallel mode. The reason of testing in parallel mode is to check whether identification model is stable or not. Figure 7 shows the validation of RBFN identification model (after the identification process is over), and for validating/testing the identification model, the input chosen is given by

$$u(k) = \sin\left(\frac{2\pi k}{25}\right) \quad \text{for} \quad k \le 500 \tag{22}$$

and

$$u(k) = 0.75\sin\left(\frac{2\pi k}{250}\right) + 0.24\cos\left(\frac{2\pi k}{25}\right) \quad \text{for} \quad k > 500 \tag{23}$$

6.2 Identification Example 2

In this example, the nonlinear dynamical plant belongs to model 2 and has the following form

$$y_p(k+1) = F + u(k) + 0.8u(k-1) \tag{24}$$

where

$$F = \frac{5y_p(k)y_p(k-1)}{1 + y_p^2(k) + y_p^2(k-1) + y_p^2(k-2)} \tag{25}$$

The identification process was continued for 30,000 time steps using a random input whose amplitude is uniformly distributed in the interval $[-1, 1]$. Number of radial centres are 20, and learning rate was chosen to be 0.05. Figure 8 shows the identification model response during the training process, and Fig. 9 shows the instantaneous mean square error. Figure 10 shows the validation of RBFN identification model after the training process is over. The input taken for testing/validation purpose is shown below:

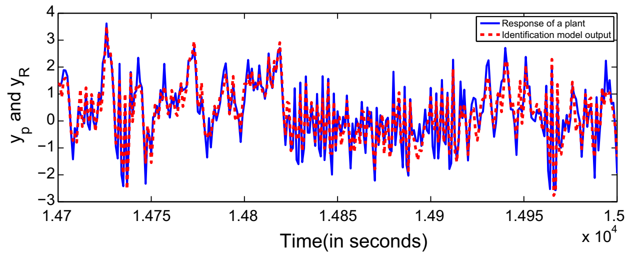

Fig. 8 Identification by RBFN during the training for random input

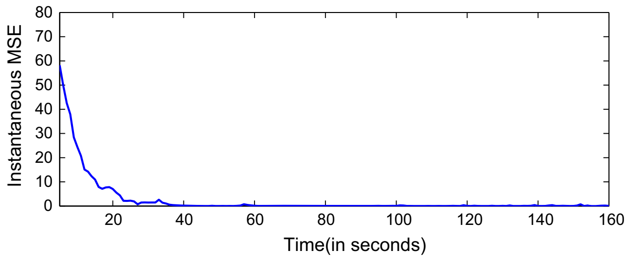

Fig. 9 Instantaneous mean square error during the training process

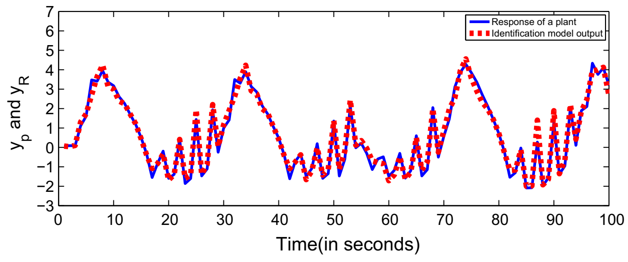

Fig. 10 Validation of RBFN identification model after the training process is over

$$u(k) = \sin\left(\frac{2\pi k}{25}\right) \quad \text{for} \quad k \le 50 \tag{26}$$

and

$$u(k) = 1.2\cos\left(\frac{2\pi k}{24}\right) \quad \text{for} \quad k > 50 \tag{27}$$

From the figures, it can be concluded that RBFN has the ability of capturing the dynamics of unknown plant in the form of its own tuned parameters (which moves towards their desired values as the online training progressed).

7 Simultaneous online identification and control using RBFN

There are two different approaches to adaptively control the plant. These are

1. Direct control.
2. Indirect control.

For direct control of linear time invariant system, stable laws exist (Narendra and Parthasarathy 1991) for updating the parameters of controller and that only requires the knowledge of the error between plant output and reference model output. But such stable laws do not exist (Wang 1993) for RBFN, and hence, indirect adaptive control will be used.

For applying indirect control method, knowledge regarding the plant's dynamics must be known since this information is required for Jacobian value computation. The value of Jacobian is needed during the training for updating the RBFN parameters (Khalil and Dombre 2004). In this paper, plant's dynamics are assumed to be unknown. So, for implementing the control action we have to incorporate identification model in parallel to the plant which will learn the dynamics of the plant as the training progress. In online training, both RBFN-based identification model and controller will operate simultaneously (which also ensures the stability of the system). In this section, first the control algorithm based on RBFN will be given. After that numerical examples will be given for demonstrating simultaneous identification and control using RBFN for nonlinear dynamical systems. This will also include the test for robustness of proposed RBFN controller in terms of system's uncertainties.

7.1 Control algorithm using RBFN

Again, the gradient descent is used for obtaining controller's parameters update equations in an incremental mode. Let $E(k)$ denote the instantaneous error between the output of plant and reference model at kth instant and is given as:

$$E(k) = \frac{1}{2}(y_m(k) - y_p(k))^2 \tag{28}$$

where $y_p(k)$ is plant output and $y_m(k)$ is the reference model output at kth instant. Differentiating $E(k)$ with respect to $Hij(k)$ will provide the rate at which $E(k)$ undergoes change with respect to every element present in each radial centre, where $i = 1$ to P, $j = 1$ to L and this requires the use of a chain rule as number of signals are between $E(k)$ and $H_{ij}(k)$.

$$\frac{\partial E(k)}{\partial H_{ij}(k)} = \frac{\partial E(k)}{\partial y_p(k)} \times \frac{\partial y_p(k)}{\partial uc(k)} \times \frac{\partial uc(k)}{\partial \phi_j(k)} \times \frac{\partial \phi_j(k)}{\partial z_j(k)} \times \frac{\partial z_j(k)}{\partial H_{ij}(k)} \tag{29}$$

$$\frac{\partial E(k)}{\partial H_{ij}(k)} = -\mathrm{e}(k) \times J(k) \times w_j(k) \times \left(\frac{\partial \phi_j(k)}{\partial z_j(k)} \times \frac{\partial z_j(k)}{\partial H_{ij}}\right) \tag{30}$$

where

$$\frac{\partial y_p(k)}{\partial uc(k)} = J(k) \tag{31}$$

$J(k)$ is called as Jacobian of plant and $uc(k)$ = RBFN controller output at kth instant.

The computation of $J(k)$ requires the knowledge of the plant's dynamics, and if they are unknown, then a separate RBFN will be used as an identification model (identifier), and it will capture the dynamics of the plant in the form its own parameters during the online training. The RBFN identification model thus will be used to calculate the Jacobian value of plant as RBFN mathematical structure is known. Now

$$\frac{\partial \phi_j(k)}{\partial z_j(k)} = -z_j(k) \times \frac{\phi_j(k)}{\sigma_j^2(k)} \tag{32}$$

and

$$\frac{\partial z_j(k)}{\partial H_{ij}(k)} = \left(\frac{\partial \sum_j \left((x_j(k) - H_{ij}(k))^2\right)^{\frac{1}{2}}}{\partial H_{ij}(k)} \right) \tag{33}$$

On simplification

$$\frac{\partial z_j(k)}{\partial H_{ij}(k)} = -\left(\frac{x_j(k) - H_{ij}(k)}{z_j(k)} \right) \tag{34}$$

Thus, update equation for radial centres is

$$H_{ij}\,(k+1) = H_{ij}\,(k) + \Delta H_{ij} \tag{35}$$

where

$$\Delta H_{ij} = \eta_1 J(k) e(k) w_j(k) \frac{\phi_j(k)}{\sigma_j^2(k)} \left(x_j(k) - H_{ij}(k)\right) \tag{36}$$

where η_1 is a learning rate and its value lies between 0 and 1. Update rule for weights is

$$\frac{\partial E(k)}{\partial w_j(k)} = \left(\frac{\partial E(k)}{\partial y_{\mathrm{p}}(k)} \times \frac{\partial y_{\mathrm{p}}(k)}{\partial uc(k)} \times \frac{\partial uc(k)}{\partial w_j(k)} \right) \tag{37}$$

where $\frac{\partial E(k)}{\partial y_{\mathrm{p}}(k)} = -(y_m(k) - y_{\mathrm{p}}(k)) = -\mathrm{e}(k)$, $\frac{\partial y_{\mathrm{p}}(k)}{\partial uc(k)} = J(k)$ and $\frac{\partial uc(k)}{\partial w_j(k)} = \phi_j(k)$. Thus, each element in $w = (w_1, w_2, w_3, \ldots, w_{\mathrm{L}})$ is updated as

$$w_j(k+1) = w_j(k) + \eta_2 e(k) J(k) \phi_j(k) \tag{38}$$

where η_2 is the learning rate, and its value is anywhere between 0 and 1. Similarly, update equation for widths is obtained as:

$$\frac{\partial E(k)}{\partial \sigma_j(k)} = \left(\frac{\partial E(k)}{\partial y_{\mathrm{p}}(k)} \times \frac{\partial y_{\mathrm{p}}(k)}{\partial uc(k)} \times \frac{\partial uc(k)}{\partial \phi_j(k)} \times \frac{\partial \phi_j(k)}{\partial \sigma_j(k)} \right) \tag{39}$$

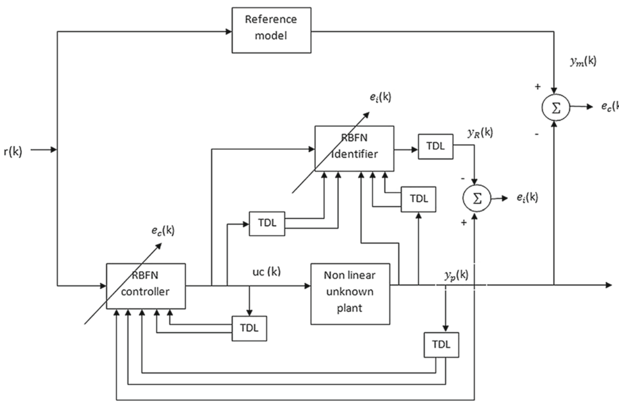

Fig. 11 Identification and indirect adaptive control using RBFN

where $\frac{\partial E(k)}{\partial y_{\mathrm{p}}(k)} = -(y_m(k) - y_{\mathrm{p}}(k)) = -\mathrm{e}(k)$, $\frac{\partial y_{\mathrm{p}}(k)}{\partial uc(k)} = J(k)$ and $\frac{\partial uc(k)}{\partial \phi_j(k)} = w_j(k)$ and

$$\frac{\partial \phi_j(k)}{\partial \sigma_j(k)} = \frac{\phi_j(k) z_j^2(k)}{\sigma_j^3(k)} \tag{40}$$

So, each element in $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{\mathrm{L}})$ is updated as

$$\sigma_j(k+1) = \sigma_j(k) + \eta_3 e(k) J(k) w_j(k) \frac{\phi_j(k) z_j^2(k)}{\sigma_j^3(k)} \tag{41}$$

where η_3 is the learning rate and η_3 lies between 0 and 1. Figure 11 shows the configuration for simultaneous online identification and control based on RBFN. The symbol $e_{\mathrm{c}}(k)$ denotes the instantaneous error between plant's output and reference model output and is used to update the parameters of RBFN controller at kth instant. The $e_i(k)$ denotes error between RBFN identification model (identifier) and actual plant, and this error is used to update the parameters of RBFN identification model at the same kth instant. Three simulation examples are given consisting of plants of different complexity, and their dynamics are assumed to be unknown. These examples demonstrate the process of simultaneous identification and control using RBFNs and also show the robustness of proposed RBFN controller.

The flowchart describing the various computational steps to be done for both RBFN identification model and controller is shown in Fig. 12

7.2 Example 1

Consider a nonlinear plant belonging to model 2 and (Narendra and Parthasarathy 1990) is described by the following difference equation

$$y_{\mathrm{p}}(k+1) = \frac{y_{\mathrm{p}}(k) y_{\mathrm{p}}(k-1) \left[y_{\mathrm{p}}(k) + 2.5\right]}{1 + y_{\mathrm{p}}^2(k) + y_{\mathrm{p}}^2(k-1)} + u(k) \tag{42}$$

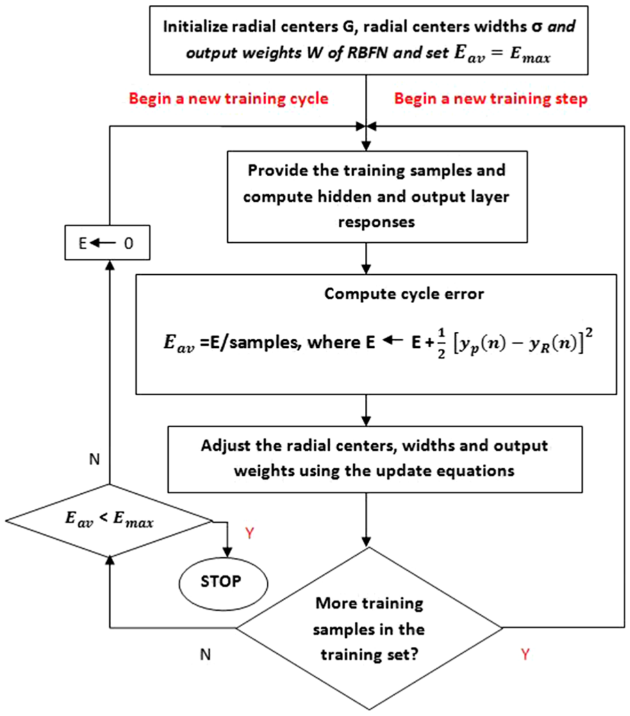
Initialize radial centers G, radial centers widths σ and output weights W of RBFN and set $E_{av} = E_{max}$
Begin a new training cycle
Begin a new training step
Provide the training samples and compute hidden and output layer responses
E ← 0
Compute cycle error
E_{av} =E/samples, where E ← E + $\frac{1}{2}[y_p(n) - y_R(n)]^2$
Adjust the radial centers, widths and output weights using the update equations
N
$E_{av} < E_{max}$
Y
STOP
More training samples in the training set?
N
Y

Fig. 12 Flowchart describing the various steps involved in the training

where the function

$$F\left[y_{\mathrm{p}}(k), y_{\mathrm{p}}(k-1)\right] = \frac{y_{\mathrm{p}}(k)y_{\mathrm{p}}(k-1)\left[y_{\mathrm{p}}(k)+2.5\right]}{1+y_{\mathrm{p}}^2(k)+y_{\mathrm{p}}^2(k-1)} \quad (43)$$

is assumed to be unknown and $u(k)$ denotes input to the plant.

The external input is BIBO stable and is given by

$$r(k) = \sin\left(\frac{2\pi k}{25}\right) \quad (44)$$

The reference model is described by the following second-order difference equation

$$y_m(k+1) = 0.6y_m(k) + 0.3y_m(k-1) + r(k) \quad (45)$$

The objective of control is to make $e_{\mathrm{c}}(k+1) = y_m(k+1) - y_{\mathrm{p}}(k+1)$ equal to or closer to zero by generating an appropriate control signal $uc(k)$ at each instant. If the function F is known, it follows directly that at kth instant, $uc(k)$ can be computed from the knowledge of $y_{\mathrm{p}}(k)$ and its past values as

$$uc(k) = -F\left[y_{\mathrm{p}}(k), y_{\mathrm{p}}(k-1)\right] + 0.6y_{\mathrm{p}}(k) + 0.3y_{\mathrm{p}}(k-1) + r(k) \quad (46)$$

If F is unknown, then it is first identified offline using RBFN whose inputs will be $y_{\mathrm{p}}(k)$ and $y_{\mathrm{p}}(k-1)$ using the series parallel mode. But this is not always the case as plant output at $(k+1)$th instant may depend nonlinearly on its input, and hence, above procedure will not be applicable. So to handle linear as well as nonlinear cases in terms of dependency of plant's output on its input, the general procedure is given and it involves a RBFN controller whose inputs are $r(k)$, $(y_{\mathrm{p}}(k), y_{\mathrm{p}}(k-1), \ldots, y_{\mathrm{p}}(k-n+1))$ and $uc(k-n+1)$ where n denotes the order of plant (which is assumed to be known). Thus, the reference input $r(k)$, the plant's present output, $y_{\mathrm{p}}(k)$, as well as its past $n-1$ values and $n-1$ past values of $uc(k)$ (i.e. RBFN controller output) will constitute the inputs to the controller. Then, by using the update equations the parameters of RBFN controller (using $e_{\mathrm{c}}(k+1)$) as well as of RBFN identification model (using $e_i(k+1)$) will be adjusted simultaneously during the online control and identification process.

In the given example, $n = 2$; thus, the number of inputs to RBFN controller will be 4, i.e.

$$uc(k) = -\mathrm{RBFN}\left[y_{\mathrm{p}}(k), r(k), y_{\mathrm{p}}(k-1), uc(k-1)\right] + 0.6y_{\mathrm{p}}(k) + 0.3y_{\mathrm{p}}(k-1) + r(k) \quad (47)$$

where uc(k) is RBFN controller output, and hence, $u(k)$ in Eq. 42 is equal to $uc(k)$.

Figure 13 shows the response of plant (dotted lines) and reference model (solid line) without control action. It can be seen from the figure that plant output is not following the reference model output. Now, using the configuration as shown in Fig. 11, both RBFN controller and RBFN identifier are set up and their parameters will be updated simultaneously online. The learning rate was set to 0.025 value. The number of radial centres was chosen to be 25 in both RBFN identifier and controller. Figure 14 shows the response of RBFN identification model (growing curve in amplitude) and plant during the initial phase of learning. As the training progresses, the output of identification model started following the plant's output, and Fig. 15 shows the response of plant (growing curve in amplitude) and reference model during the initial stage of RBFN controller training. Here also the response of plant with RBFN controller action improved (become closer to reference model response) and become similar to that of

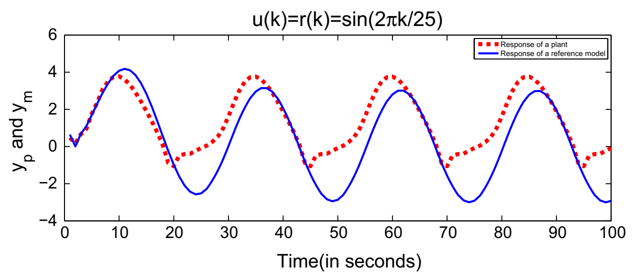

Fig. 13 Response of plant without control

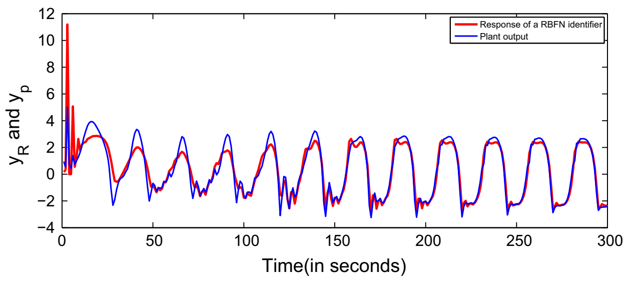

Fig. 14 Response of RBFN identifier and plant during initial phase of training

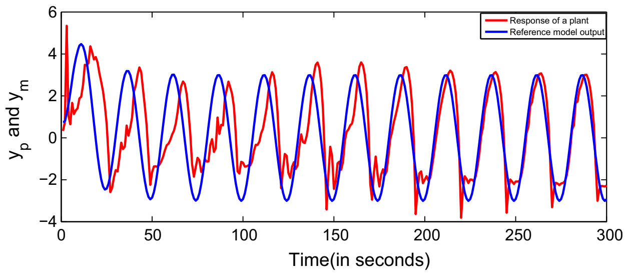

Fig. 15 Response of plant and reference model during initial stage of RBFN controller training

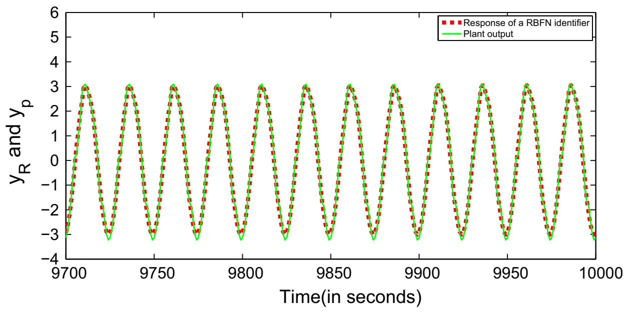

Fig. 16 RBFN identification model and plant response after sufficient online training

reference model. The updating process continued for 10,000 time steps after which it was terminated. It can be seen from Fig. 16 that RBFN identifier is able to capture the dynamics of the plant. Figure 17 shows the mean square error plot for RBFN identification model. When the RBFN parameters were updating, then at the same time RBFN controller parameters were also updating. Figure 18 shows the response of plant with RBFN controller in action after the sufficient online training of RBFN controller is done. It is clear from Fig. 18 that RBFN parameters attained their optimal values and generate the desired control output such that plant output is following the reference model output at each instant. Figure 19 shows the mean square error plot of RBFN controller, and it is clear from Fig. 19 that as the training progresses mean square error keeps on decreasing because as the training progresses the parameters of RBFN move towards their

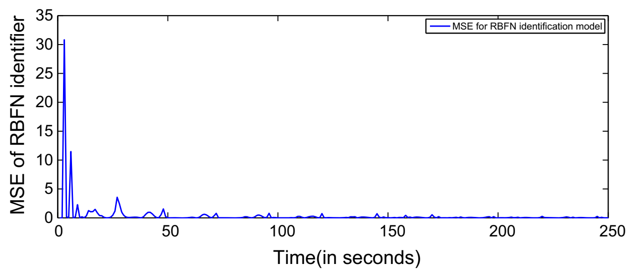

Fig. 17 Instantaneous mean square error of RBFN identification model

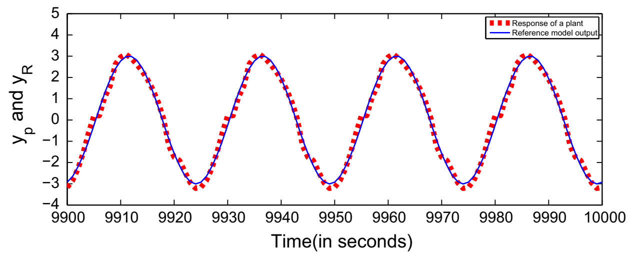

Fig. 18 Plant response with RBFN controller after sufficient online training

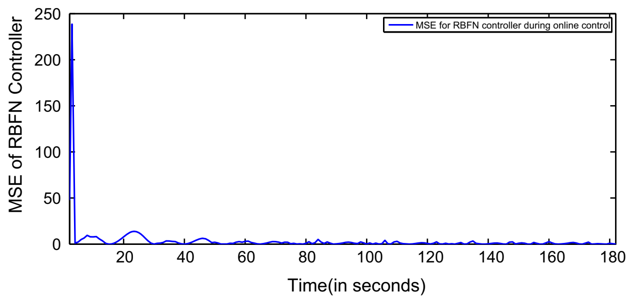

Fig. 19 Instantaneous mean square error of RBFN controller model

Table 1 Comparison of instantaneous mean square error (MSE) of RBFN and MLFFNN identifier during training

Network type	Average MSE
Average MSE of RBFN identifier	0.0170
Average MSE of MLFFNN identifier	0.0335

optimum values and hence instantaneous error at subsequent instants keeps on decreasing. The same example was simulated using the MLFFNN with single hidden layer and with number of hidden neurons equals to number of radial centres in RBFN, i.e. 25. The learning rate was also chosen same, i.e. $\eta = 0.025$, and the training continued for same number of time instants, i.e. 10,000. Table 1 shows the comparative results showing average mean square error (MSE) during online training of RBFN and MLFFNN identification model. Table 2 shows the average mean square error of RBFN and MLFFNN controller. From both the tables, it is

Table 2 Comparison of instantaneous mean square error (MSE) of RBFN and MLFFNN controller during training

Network type	Average MSE
Average MSE of RBFN controller	0.0231
Average MSE of MLFFNN controller	0.0318

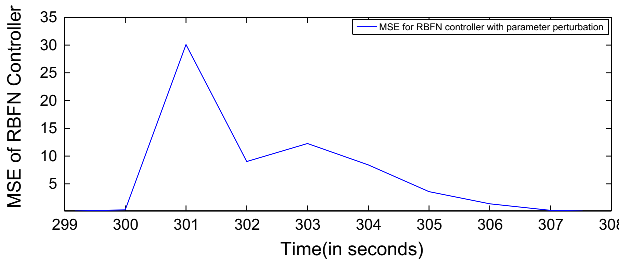

Fig. 20 Instantaneous MSE of RBFN controller during parameter perturbation

clear that RBFN has given better results than the MLFFNN. Now the proposed controller is checked for robustness. It is one of the important and useful properties of feedback. A system having robustness is very little sensitive or immune to parameter variations (component's values variations) in a plant/system. Since the given plant's dynamics are unknown, they are approximated by the RBFN identification model. So, in essence the unknown parameters values of plant are stored in the form of parameters values of the RBFN identification model. Thus, in order to check the robustness of RBFN controller, any element of output weight vector of RBFN identification model can be perturbed. The consequence of this will be seen as an increase in the value of instantaneous mean square error in MSE plot of RBFN controller at the instant at which perturbation occurred. If the MSE again goes back to zero, then the system has robustness otherwise not. Figure 20 shows the MSE plot of RBFN controller when first perturbation in parameter occurred at $K = 300$th s instant. From Fig. 20, notice that when perturbation occurred MSE suddenly increases and then quickly drops towards zero but again a perturbation in parameter occurred at $k = 302$th instant which again made the MSE to increase but as training progresses MSE again started to decrease and becomes zero at $k = 307$th time instant which shows the robust nature of RBFN controller. Now controller robustness is checked against the disturbance signals in the system.

Disturbance also leads to spikes in the MSE plot of RBFN controller, and from Fig. 21, it is clear that at two noise signals (of step type) entered the system at $k = 3000$ and $k = 3002$th s instants, respectively, and there occurs an increase in MSE value but as the training progresses it quickly recovered and drops down to zero within few seconds proving the robust characteristic of RBFN controller. This shows the effectiveness of the control configuration as

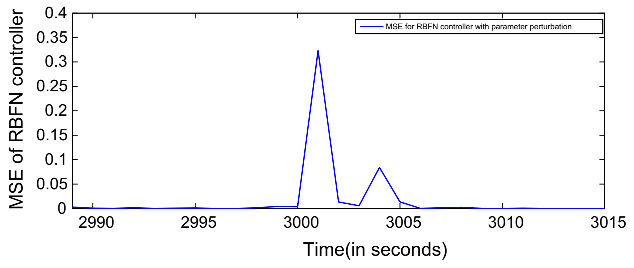

Fig. 21 Instantaneous MSE of RBFN controller during noise

shown in Fig. 11. The identification model is able to capture the changed dynamics of plant which gets modified due to the disturbance signals of parameter variations, and this allows the controller to modify its own parameters accordingly so that a correction can be made in the control signal value. This makes plant's to provide again the desired output values.

7.3 Example 2

In this example, the unknown dynamics of the nonlinear plant (Narendra and Parthasarathy 1990) is described by the following difference equation

$$y_p(k) = \frac{5y_p(k)y_p(k-1)}{1+y_p^2(k)+y_p^2(k-1)+y_p^2(k-2)} + u(k) + 0.8u(k-1) \tag{48}$$

The external input $r(k)$ is given by $r(k) = \sin(2\pi k/25)$. The order of the given plant is 3 so number of inputs to the RBFN controller will be 6. The present external input $r(k)$, present plant output $y_p(k)$ and its past values $y_p(k-1)$, $y_p(k-2)$ along with past two values of controller outputs: $uc(k-1)$ and $uc(k-2)$ will form the six inputs to the RBFN controller. For RBFN identifier, $y_p(k)$, $y_p(k-1)$ and $y_p(k-2)$ form the inputs. Thus, RBFN identification model is described by the following difference equation in the series–parallel mode as

$$y_R(k+1) = \text{RBFN}\left[y_p(k), y_p(k-1), y_p(k-2)\right] + u(k) + 0.8u(k-1) \tag{49}$$

The reference model is described by the following second-order difference equation

$$y_m(k+1) = 0.72y_m(k)+0.64y_m(k-1)-0.5y_m(k-2)+r(k) \tag{50}$$

The learning rate, η, was chosen to be 0.0015. Figure 22 shows the response of plant (non sinusoidal curve with smaller amplitude) and reference model (sinusoidal curve) when no control action is initiated. It is clear that plant output is not following the reference model output, and hence,

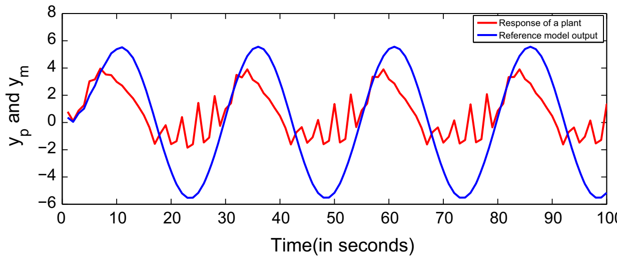

Fig. 22 Plant output and reference model output without control

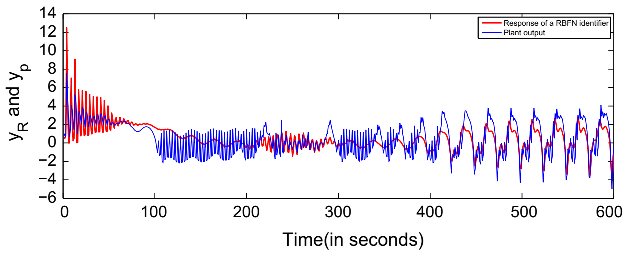

Fig. 23 RBFN identification model output and plant output during the initial phase identification

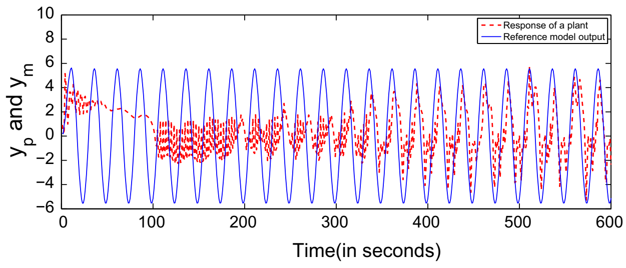

Fig. 24 Output of plant with controller action and reference model output during the initial phase of controller training

control action is needed. Figure 23 shows the output of RBFN identification model (curve with higher amplitude at the beginning) and plant output (curve with smaller amplitude at the beginning) when training is in its initial stage, and Fig. 24 shows the plant output (non sinusoidal curve) and reference model output (sinusoidal curve) when training of RBFN controller is in initial phase. From both the figures, it can be seen that as the training progresses parameters of both RBFNs are approaching towards their desired optimal values.

After a sufficient training (for 25,000 time steps), the RBFN identifier and controller parameters reached to their desired values. Figure 25 shows the response of identification model and plant. It is clear from the figure that the output of RBFN identifier and plant is indistinguishable. Figure 26 shows the response of plant (dotted curve) with trained RBFN controller, and it can be easily seen from the figure that plant output is following the reference model output (solid line curve) under RBFN controller action.

Figure 27 shows the instantaneous mean square error plot of RBFN identification model during the training. It is clear

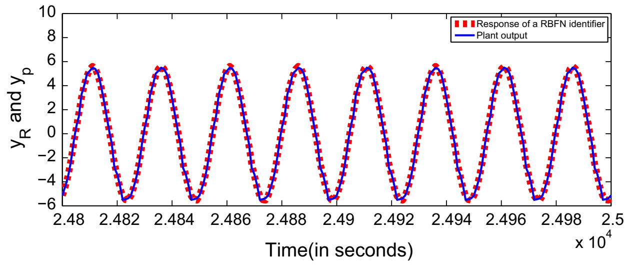

Fig. 25 RBFN identification model output and plant output at end stage of the identification process

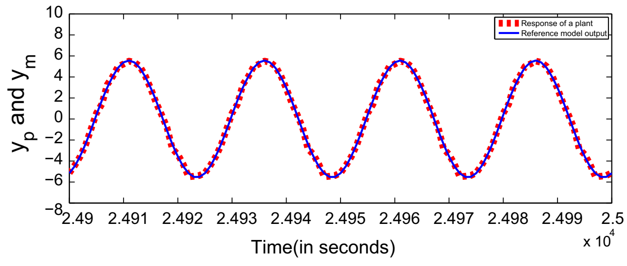

Fig. 26 Plant output and reference model output at end stage of the training process

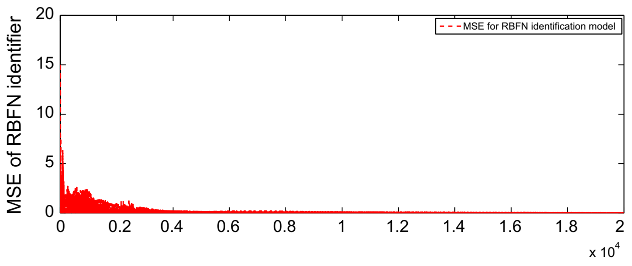

Fig. 27 Instantaneous mean square error of RBFN identification model during the online training

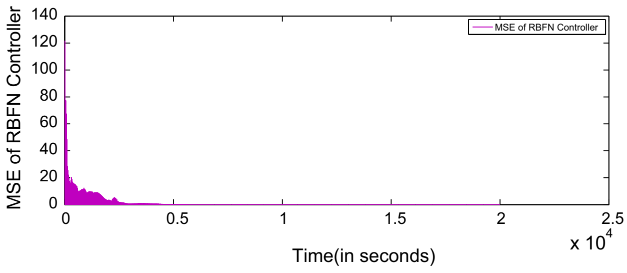

Fig. 28 Instantaneous mean square error of RBFN controller during the online training

from the figure that in the earlier stages of training the error was appreciable but as the training progresses the error keeps on decreasing and finally become closer to zero. Similar is the case with the instantaneous mean square error plot of RBFN controller which is shown in Fig. 28. Again in this example the comparative analysis is done by simulating the same example with MLFFNN under same conditions, and

Table 3 Comparison of mean square error (MSE) of RBFN and MLFFNN identifier during training

Network type	Average MSE
Average MSE of RBFN identifier	0.0356
Average MSE of MLFFNN identifier	0.2042

Table 4 Comparison of mean square error (MSE) of RBFN and MLFFNN controller during training

Network type	Average MSE
Average MSE of RBFN controller	0.1232
Average MSE of MLFFNN controller	0.1872

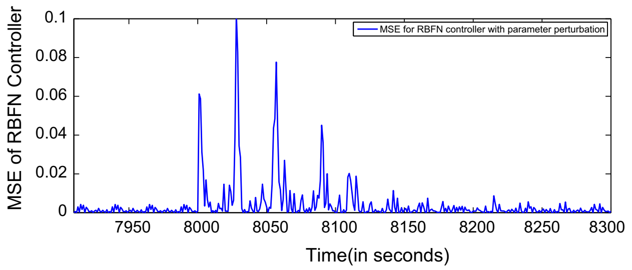

Fig. 29 Instantaneous mean square error of RBFN controller with noise in the system

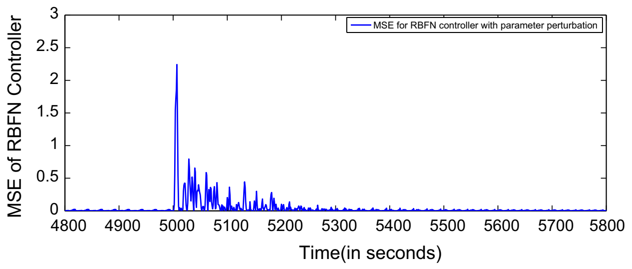

Fig. 30 Mean square error of RBFN controller during parameter perturbation

the results in terms of average instantaneous MSE are given in Tables 3 and 4. The RBFN controller in this example is also checked for robustness. The system undergone parameter perturbation at $k = 8000$th time instant, and from Fig. 29 it can be seen that the MSE at that instant rapidly increased and then in a very short time goes to zero. Figure 30 shows the effect on the MSE when noise entered the system at $k = 5000$th instant.

The RBFN has again given improved response which proves it as a strong option for implementing as a controller and identifier. The MSE value increases in response to the noise, but parameters of RBFN recovered as the training progresses and MSE again decrease and drops down to zero value. This example again shows that the RBFN has the robustness characteristic.

7.4 Example 3

In this last example a nonlinear plant belonging to Narendra and Parthasarathy (1990), a class of model 4 is considered, and it has the unknown dynamics which are to be indentified parallel along with the control action:

$$y_p(k+1) = \frac{y_p(k)y_p(k-1)y_p(k-2)u(k-1)\left[y_p(k-2)-1\right]+u(k)}{1+y_p^2(k-1)+y_p^2(k-2)} \tag{51}$$

The external input $r(k)$ is given by

$$r(k) = \sin(2\pi k/25) \tag{52}$$

and reference model dynamics is given by

$$y_m(k+1) = 0.08y_m(k) + 0.13y_m(k-1) + r(k) \tag{53}$$

The dependency of plant output at $(k+1)$th instant on its input is not linear. The order of the plant is $n = 3$ so RBFN controller will have six inputs. The learning rate, η, was chosen to be 0.0038. The external reference input $r(k)$, the plant's output $y_p(k)$ as well as its past $n-1$ values, namely $y_p(k-1)$, $y_p(k-2)$ and $n-1$ past values of RBFN controller outputs: $uc(k-1)$ and $uc(k-1)$. Since the given dynamics of the plant are assumed to be unknown, they will be identified simultaneously along with RBFN controller's parameter tuning using RBFN identifier having the same plant's mathematical structure shown in below equation:

$$y_R(k+1) = \text{RBFN}\left[y_p(k), y_p(k-1), y_p(k-2), u(k), u(k-1)\right] \tag{54}$$

Both RBFN controller and RBFN identification model will operate and undergo online training simultaneously. The response of the plant to external input, $r(k)$, without RBFN controller is shown in Fig. 31. The identification model and plant's response during early stages of training are shown in Fig. 32. It can be seen from the figure that as the training progresses (along with RBFN control) the RBFN identification model output started following the plant's output. Figure 33 shows the plant's output (growing in amplitude) and reference model output (sinusoidal curve with constant amplitude) during the initial stages of RBFN controller training.

After sufficient training (for 30,000 time steps), the final stage response (during the training process) of RBFN identification model and plant's output under RBFN controller action are shown in Figs. 34 and 35, respectively. From both the figures, a conclusive evidence of successful online

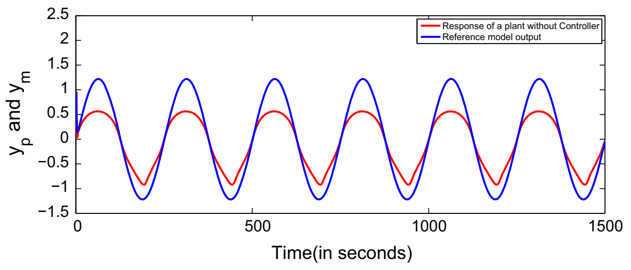

Fig. 31 Plant output and reference model output without control action

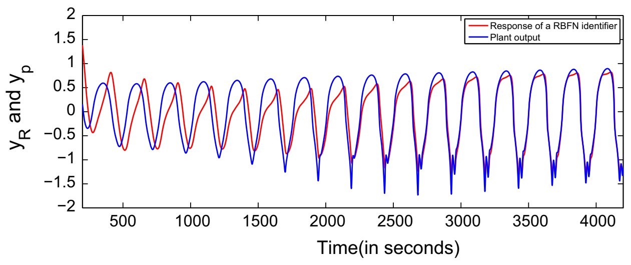

Fig. 32 RBFN identification model output and plant's output during the initial phase identification

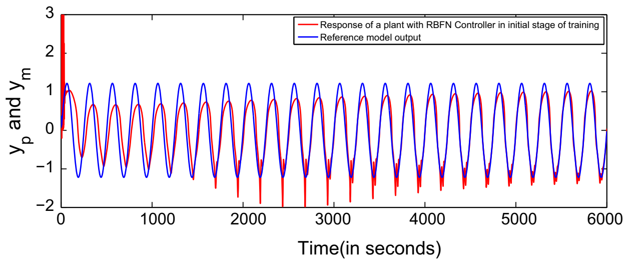

Fig. 33 RBFN controlled plant output and reference model output during the initial phase of controller training

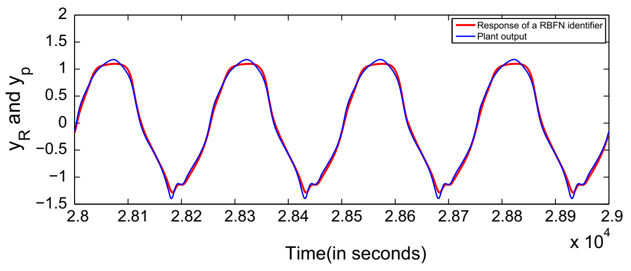

Fig. 34 RBFN identification model output and plant output at end stage of the identification process

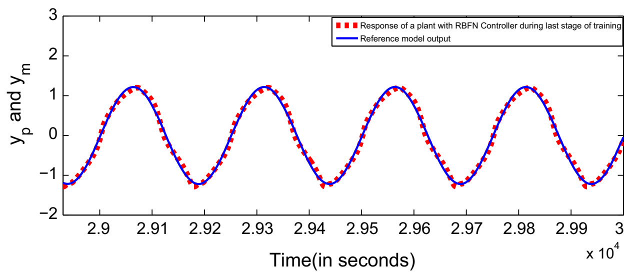

Fig. 35 Plant output and reference model output at end stage of the training process

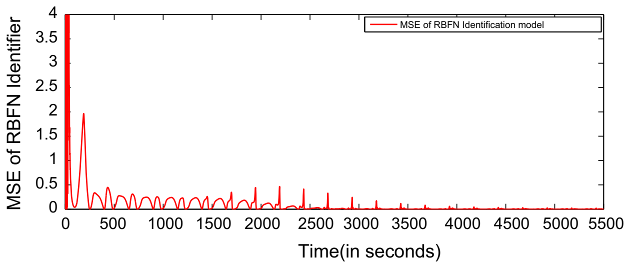

Fig. 36 Instantaneous mean square error of RBFN identification model during online training

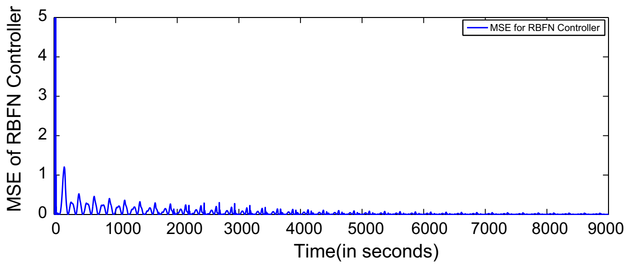

Fig. 37 Instantaneous mean square error of RBFN controller during online training

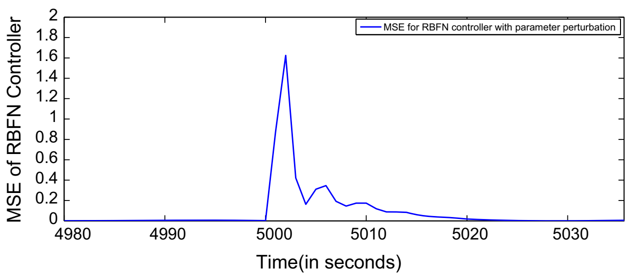

Fig. 38 Instantaneous MSE of RBFN controller during parameter perturbation

training of both RBFN controller and identifier is obtained. Figures 36 and 37 show the mean-square-error (MSE) plots of RBFN controller and RBFN identification model, respectively, during the simultaneous online training. Now, the developed controller is checked for robustness. For this the performance of controller is checked against parameter variation and noise in the system. The parameter variation (one of the RBFN weight undergoes step like change) occurred at $k = 5000$th instant, and from Fig. 38 it can be seen that the MSE responded to that in the form of increase in the value but as the time progresses MSE quickly drops down towards zero which shows the robust nature of RBFN controller. After this, RBFN is tested against the effects of noise which entered the system at $k = 6000$th instant. In this case also, the rise in MSE due to noise was soon taken care of, and from Fig. 39,

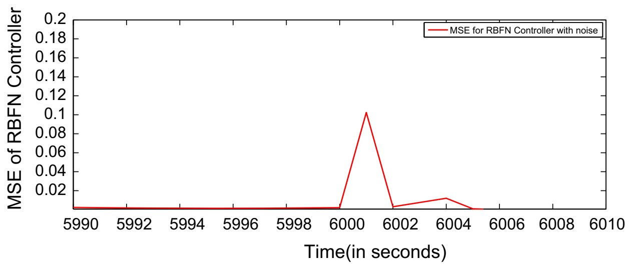

Fig. 39 Instantaneous MSE of RBFN controller when noise enters in the system

Table 5 Comparison of mean square error (MSE) of RBFN and MLFFNN identifier during training

Network type	Average MSE
Average MSE of RBFN identifier	0.0153
Average MSE of MLFFNN identifier	0.0245

Table 6 Comparison of mean square error (MSE) of RBFN and MLFFNN controller during training

Network type	Average MSE
Average MSE of RBFN controller	0.0236
Average MSE of MLFFNN controller	0.0562

one can see that the MSE reduced to zero within a very short time proving the robustness of RBFN controller against the effects of noise. Again in this example the comparative analysis is done by simulating the same example with MLFFNN under same conditions, and the results in terms of MSE are given in Tables 5 and 6. From the tables, it can be said that RBFN has given better results over MLFFNN.

8 Achievements of the paper

The achievements of this paper are summarized below:

1. The application of RBFN as a robust controller is demonstrated in this paper. Its performance was checked by considering nonlinear plant's of different complexities. Simulation results showed that RBFN is able to efficiently control the plants.
2. In this paper, the RBFN performance is compared with MLFFNN, and results showed the superior performance of RBFN over MLFFNN.
3. The robustness of proposed RBFN controller is checked against parameter variations and disturbances. The plots of MSE showed that RBFN parameters recovered quickly and MSE again dropped down to zero.
4. The dynamics of the plants considered in the simulation examples were assumed to be unknown; hence, both identification and control were implemented simultaneously which again shows the capability of RBFN as an identifier and as a controller.

9 Conclusion and future scope

In this paper, radial basis function network has been used for online identification and control of nonlinear plants of different complexities. The results reveal that RBFN is able to efficiently control the plant without requiring the knowledge of the plant's parameters. The response of RBFN controller and identification model depends on the learning of their parameters, and results showed that as the training progressed their parameters reached to their optimum values, thus delivering the right value of RBFNs output. It has also shown good performance under the impact of noise and parameter variations as the parameters of RBFN are able to quickly adjust them in order to cope with the changes occurred in the system. The RBFN performance in comparison with MLFFNN is also checked, and results showed that RBFN performed better than the MLFFNN. The topological structure of RBFN is much simpler than MLFFNN which gives it an edge over it in terms of computation time required during the online training phase. This makes RBFN as a perfect candidate for control and identification of plants with nonlinearity, and from the simulation results, it can be proposed that RBFN can be used as a robust controller and as an identifier. The application of RBFN-based identification and control can be extended to multi-input multi-output (MIMO) systems. New learning algorithms can be explored, and their stability analysis can be done for updating the parameters of RBFN. RBFN can be combined with fuzzy-based systems (like neuro-fuzzy systems) in order to use the power of both structures. The issues like packet dropouts and time delays also create problems while controlling the plant and application of RBFN for dealing with such cases is itself a new problem, which can be addressed in future works.

Compliance with ethical standards

Conflict of interest All 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

Almaadeed N, Aggoun A, Amira A (2015) Speaker identification using multimodal neural networks and wavelet analysis. IET Biom 4(1):18–28

Attaran SM, Yusof R, Selamat H (2016) A novel optimization algorithm based on epsilon constraint-RBF neural network for tuning PID controller in decoupled HVAC system. Appl Therm Eng 99:613–624

Ayala HVH, dos Santos Coelho L (2016) Cascaded evolutionary algorithm for nonlinear system identification based on correlation functions and radial basis functions neural networks. Mech Syst Signal Process 68:378–393

Behera L, Kar I (2010) Intelligent systems and control principles and applications. Oxford University Press Inc., Oxford

Bishop CM (1995) Neural networks for pattern recognition. Oxford University Press, Oxford

Broomhead DS, Lowe D (1988) Radial basis functions, multi-variable functional interpolation and adaptive networks. Tech. rep., DTIC document

Cao J, Liang J (2004) Boundedness and stability for Cohen–Grossberg neural network with time-varying delays. J Math Anal Appl 296(2):665–685

Chen S, Billings S, Grant P (1992) Recursive hybrid algorithm for nonlinear system identification using radial basis function networks. Int J Control 55(5):1051–1070

Dai SL, Wang C, Luo F (2012) Identification and learning control of ocean surface ship using neural networks. IEEE Trans Ind Inf 8(4):801–810

Elanayar V, Shin YC et al (1994) Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems. IEEE Trans Neural Netw 5(4):594–603

Fu Y, Chai T (2007) Nonlinear multivariable adaptive control using multiple models and neural networks. Automatica 43(6):1101–1110

Gomm JB, Yu DL (2000) Selecting radial basis function network centers with recursive orthogonal least squares training. IEEE Trans Neural Netw 11(2):306–314

Haykin S, Network N (2004) A comprehensive foundation. Neural Netw 2(2004)

Hsu CF, Lin CM, Yeh RG (2013) Supervisory adaptive dynamic RBF-based neural-fuzzy control system design for unknown nonlinear systems. Appl Soft Comput 13(4):1620–1626

Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892

Jankowski N, Kadirkamanathan V (1997) Statistical control of RBF-like networks for classification. In: Artificial neural networks ICANN'97. Springer, Berlin Heidelberg, pp 385–390

Kayacan E, Kayacan E, Khanesar MA (2015) Identification of nonlinear dynamic systems using type-2 fuzzy neural networks—a novel learning algorithm and a comparative study. IEEE Trans Ind Electron 62(3):1716–1724

Khalil W, Dombre E (2004) Modeling, identification and control of robots. Butterworth-Heinemann, London

Lowe D (2015) Radial basis function networks-revisited. Math Today 51(3):124–126

Moody J, Darken CJ (1989) Fast learning in networks of locally-tuned processing units. Neural Comput 1(2):281–294

Narendra KS, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Netw 1(1):4–27

Narendra KS, Parthasarathy K (1991) Gradient methods for the optimization of dynamical systems containing neural networks. IEEE Trans Neural Netw 2(2):252–262

Narendra KS, Parthasarathy K (1992) Neural networks and dynamical systems. Int J Approx Reason 6(2):109–131

Paul A, Bhattacharya P, Maity SP (2015) Comparative analysis of radial basis functions with SAR images in artificial neural network. In: Advances in intelligent informatics. Springer, Switzerland, pp 125–131

Perng JW, Chen GY, Hsu YW (2016) FOPID controller optimization based on SIWPSO–RBFNN algorithm for fractional-order time delay systems. Soft Comput 1–14

Poggio T, Girosi F (1990) Networks for approximation and learning. Proc IEEE 78(9):1481–1497

Qiao JF, Han HG (2012) Identification and modeling of nonlinear dynamical systems using a novel self-organizing RBF-based approach. Automatica 48(8):1729–1734

Qiu J, Feng G, Gao H (2013a) Static-output-feedback control of continuous-time T–S fuzzy affine systems via piecewise Lyapunov functions. IEEE Trans Fuzzy Syst 21(2):245–261

Qiu J, Tian H, Lu Q, Gao H (2013b) Nonsynchronized robust filtering design for continuous-time T–S fuzzy affine dynamic systems based on piecewise Lyapunov functions. IEEE Trans Cybern 43(6):1755–1766

Qiu J, Wei Y, Karimi HR (2015) New approach to delay-dependent H∞ control for continuous-time Markovian jump systems with time-varying delay and deficient transition descriptions. J Frankl Inst 352(1):189–215

Qiu J, Ding SX, Gao H, Yin S (2016a) Fuzzy-model-based reliable static output feedback control of nonlinear hyperbolic pde systems. IEEE Trans Fuzzy Syst 24(2):388–400

Qiu J, Gao H, Ding SX (2016b) Recent advances on fuzzy-model-based nonlinear networked control systems: a survey. IEEE Trans Ind Electron 63(2):1207–1217

Rossomando FG, Soria C, Carelli R (2011) Autonomous mobile robots navigation using RBF neural compensator. Control Eng Pract 19(3):215–222

Sarimveis H, Alexandridis A, Bafas G (2003) A fast training algorithm for RBF networks based on subtractive clustering. Neurocomputing 51:501–505

Schultz MH (1973) Spline analysis In: Prentice-Hall series in automatic computation. Prentice-Hall, London

Seng TL, Khalid M, Yusof R, Omatu S (1998) Adaptive neuro-fuzzy control system by RBF and GRNN neural networks. J Intell Robot Syst 23(2–4):267–289

Soudry D, Di Castro D, Gal A, Kolodny A, Kvatinsky S (2015) Memristor-based multilayer neural networks with online gradient descent training. IEEE Trans Neural Netw Learn Syst 26(10):2408–2421

Srivastava S, Singh M, Hanmandlu M, Jha AN (2005) New fuzzy wavelet neural networks for system identification and control. Appl Soft Comput 6(1):1–17

Sutrisno I, Jami'in MA, Hu J, Marhaban MH (2015) Self-organizing quasi-linear ARX RBFN modeling for identification and control of nonlinear systems. In: 2015 54th annual conference of the society of instrument and control engineers of Japan (SICE). IEEE, pp 642–647

Torshizi AD, Petzold L, Cohen M (2015) Direct higher order fuzzy rule-based classification system: application in mortality prediction. In: 2015 IEEE international conference on bioinformatics and biomedicine (BIBM). IEEE, pp 846–852

Wang LX (1993) Stable adaptive fuzzy control of nonlinear systems. IEEE Trans Fuzzy Syst 1(2):146–155

Wang T, Gao H, Qiu J (2016) A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control. IEEE Trans Neural Networks Learn Syst 27(2):416–425

Wei Q, Liu D (2015) Neural-network-based adaptive optimal tracking control scheme for discrete-time nonlinear systems with approximation errors. Neurocomputing 149:106–115

Yu H, Xie T, Paszczynski S, Wilamowski BM (2011) Advantages of radial basis function networks for dynamic system design. IEEE Trans Ind Electron 58(12):5438–5450