Neural Comput & Applic
DOI 10.1007/s00521-016-2695-8

ORIGINAL ARTICLE

Online modeling and adaptive control of robotic manipulators using Gaussian radial basis function networks

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

Received: 3 May 2016 / Accepted: 3 November 2016

Abstract Radial basis function network (RBFN) is used in this paper for predefined trajectory control of both one-link and two-link robotic manipulators. The updating equations for the RBFN parameters were derived using the gradient descent principle. The other advantage of using this principle is that it shows the clustering effect in distributing the radial centres. To increase the complexity, the dynamics of robotic manipulator is assumed to be unknown, and hence, simultaneous control and identification steps were performed using the RBFNs. The performance of the RBFN is compared with the multilayer feed-forward neural network (MLFFNN) in terms of mean square error, tolerance to disturbance and parameter variations in the system. The efficacy of RBFN as a controller and identification tool is verified by performing the simulation study, and the results obtained reveal the superior performance of RBFN over MLFFNN in both identification and control aspects for one-link and two-link robotic manipulators.

Keywords Radial basis function networks · One-link and two-link robotic manipulators · Identification and adaptive control · Multi layer feed-forward neural network · Robustness

✉ Rajesh Kumar
rajeshmahindru23@gmail.com

Smriti Srivastava
smriti.nsit@gmail.com

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

[1] Department of Instrumentation and Control Engineering, Netaji Subhas Institute of Technology, Sector 3, Dwarka, New Delhi 110078, India

1 Introduction

The fact that most systems are inherently nonlinear with partially known or unknown dynamics makes the use of conventional techniques like PID control difficult. This has laid to development and use of intelligent control techniques. There has been a rapid growth seen in the research efforts during the past decades for developing systematic methods for the predefined trajectory control of one-link and two-link robotic manipulators [26]. The robotic manipulators found numerous applications in industries like manufacturing, assembly, space and medical in which they are used for the purpose of pick and place, positioning and path following, etc. Various control methods have been explored as the suitable alternatives to enhance the deficiencies of the traditional PID-type controllers [13, 25]. An adaptive-learning control scheme was proposed by Sun and Mills [25] for improving the trajectory performance, but for this scheme to work, system dynamics details are required. A model-based PID controller was introduced by Li et al. [13] to accomplish the time-varying tracking control of a robot controller. In any case, it is hard to build up a fitting numerical model for the design of a model-based control system. On the other hand, the conventional intelligent control schemes have the capacity to compensate the effects of structured parametric uncertainty and unstructured disturbance by using their powerful learning ability without requiring the prior knowledge about the mathematical model of system (plant) under consideration. In the past decade, considerable attention is given to the use of intelligent control techniques (fuzzy logic-based control or artificial neural-network-based control) for the motion control of robotic manipulators [5, 6, 9, 14]. A stable self-organizing fuzzy

Published online: 17 November 2016

controller was introduced by Huang and Lee [10] for the motion control of robot. This method has the ability to respond to time-varying characteristic of robotic manipulator. But, it suffers from a latent stability problem.

The complexity of problem further increased when the dynamics of the system to be controlled is unknown or partially known. This requires the identification process to be initiated in parallel along with the control [7, 22]. In this paper, both these processes are implemented using RBFNs. RBFNs are being widely used in areas like modeling and identification, pattern recognition, but their potential as a controller is not fully exploited in the control applications. The ability of RBFN to approximate nonlinear mapping directly from the input-output data has made them popular in recent times. RBFNs provide number of advantages over

MLFFNN such as smaller extrapolation errors, simpler structure, higher reliability, fast rate of convergence and have increasingly attracted the interest for various engineering applications [27].

Contributions of this paper are summarized as follows:

1. In the literature, no thorough comparative analysis between RBFN and MLFFNN for identification and control of one-link and two-link robotic manipulators is available. In this paper, performance of both RBFN and MLFFNN as a controller and identification model is investigated.
2. Detailed comparative analysis is done for testing the robustness of RBFN and MLFFNN controllers. For this, both parameter variations and disturbance signals cases are considered.
3. The control and identification scheme utilizing RBFN structure is proposed.
4. Criteria for selecting the inputs to the controller are also proposed.

The remainder of this paper is organized as follows: in Sect. 2, details regarding the mathematical structure of RBFN are given. Section 3 contains the description of various identification modes and their implementation using RBFN. In Sect. 4, the identification algorithm for RBFN identification model is given which includes the derivation for RBFN identification model parameter's update equations. Section 5 contains the description of adaptive control based on RBFN and the necessary update equations derivation of parameters of RBFN controller. Section 6 contains the simulation results involving one-link and two-link robotic manipulators as the plants. Here, a detailed comparative analysis on the performance of RBFN and MLFFNN is done along with the robustness testing. In Sect. 7, achievements of this paper are mentioned, and Sect. 8 concludes the paper.

2 Mathematical structure of RBFN

The structure of RBFN is fixed unlike the MLFFNN which gives it an advantage over MLFFNN in terms of structure complexity. RBFN consists of only 3 layers: input, hidden and output layer [12]. The capability of RBFN to approximate the nonlinear input-output mapping comes from its hidden layer. The hidden layer transforms the n-dimensional input vector (meaning n-inputs are there in one training input sample) into a high-dimensional feature space using Gaussian radial function [2, 8]. Each node in the hidden layer is called as radial centre, and they are different in terms of functionality from the neurons of MLFFNN. The RBFN output layer includes nodes with linear activation [15] and has same functionality as the neurons of MLFFNN [11]. Further, the count of radial centres considered in the hidden layer is lesser in number than the total number of training samples which makes it more efficient than MLFFNN in terms of computational time. The structure of RBFN is shown in Fig. 1.

In the structure of RBFN, the first layer is the input layer $X = (x_1, x_2, \ldots, x_n)$. The radial centres vector $h = (h_1, h_2, \ldots, h_m)$ contains m number of radial centres where m is always less than n and it constitutes the hidden layer. The value of each input weight connecting inputs of input layer to the hidden layer nodes is unity [8, 28]. The output weight vector is (for SISO system) $w = [w_1, w_2, w_3, \ldots, w_m)$ (and the bias). In short, the input is clustered around the radial centres [2] and the output is linear in terms of weights, w

$$y_r(k) = \sum_{i=1}^{m} \phi_i(k) w_i(k) \tag{1}$$

where $y_r(k)$ is the output of RBFN at any kth time instant. The output of the ith radial centre in the hidden layer at kth time instant in RBFN is expressed by the following expression:

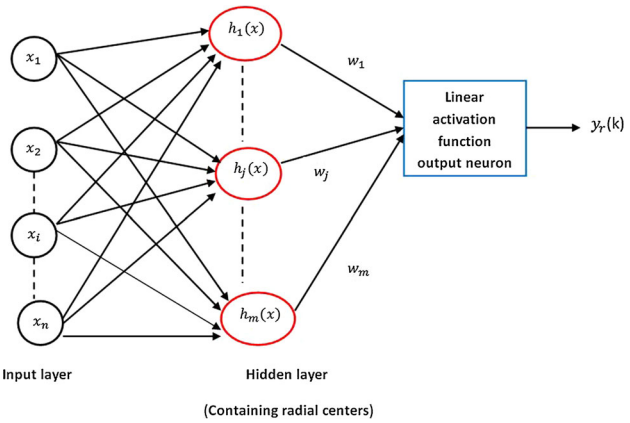

Fig. 1 Structure of RBFN

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

where Euclidean distance between X(k) and $h_i(k)$ is given by $\|X(k) - h_i(k)\|$. Radial basis function, $\phi(.)$, 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 $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ represents the width of each radial centre [2]. During the online training, each radial centre, its corresponding width and the output weight vector are adjusted in order to reduce the cost function (instantaneous mean square error) by using the recursive equations that are obtained by using the gradient descent. However, the contribution of σ to the output of RBFN is not that significant [2].

3 Identification of unknown dynamics using RBFN

The main aim of an identification process is to approximate the unknown dynamics of the given plant. It involves setting up an identification model whose parameters undergo training. After sufficient training, the identification model's mathematical structure can be used in place of the actual plant for computing the output of controller. In order to use identification process, some a-priori information regarding the given plant must be known. This information includes:

1. The value of order n of the plant.
2. The class of model to which the unknown dynamics of plant belongs.

There are several well-established identification models for approximating the nonlinear systems [20, 21]. If the plant output at a given time instant depends upon the input at the same instant, then the plant has a static nature, and hence, static identification model will be used to approximate it. But most plants are dynamic in nature as their present output depends upon their past values as well as on the present and past values of the input. The details on dynamic identification models can be found in [16]. The most general identification model is given by the following dynamic difference equation:

$$\begin{aligned} y_p(k+1) = A[&y_p(k), y_p(k-1)\ldots y_p(k-n+1), \\ &u(k), u(k-1)\ldots u(k-m+1)] \end{aligned} \tag{4}$$

where A is an unknown nonlinear function. Symbol n in above equations denotes the number of state variables of the plant and is called order of the plant and $m \leq n$. If the value of n is unknown, then it is usually taken to be 2 as most plants can be approximated by second-order equation [24]. The same structure identification model based on RBFN is then chosen which will approximate this unknown nonlinear function. The inputs to the RBFN identification model will be the terms present in the argument of nonlinear function as given in Eq. 4. Above model structure is the most general one as it subsumes all the other available identification models and will also be used in those cases where the knowledge about the mathematical structure of the plant is not available.

3.1 Modes of identification

The identification procedure can be carried out in any of the below-mentioned modes:

1. *Parallel identification mode* In this mode, the RBFN will use its own past values along with external input for computing its own next value. For example, the identification model given in Eq. 4 can be written in parallel mode as:

$$\begin{aligned} y_r(k+1) = A[&y_r(k), y_r(k-1)\ldots y_r(k-n+1), \\ &u(k), u(k-1)\ldots u(k-m+1)] \end{aligned} \tag{5}$$

2. *Series-parallel identification mode* In this mode, the plant past outputs are feedback and are used to compute the next value of RBFN identification model. For example, the identification model as given in Eq. 4 can be written in this mode as:

$$\begin{aligned} y_r(k+1) = A[&y_p(k), y_p(k-1)\ldots y_p(k-n+1), \\ &u(k), u(k-1)\ldots u(k-m+1)] \end{aligned} \tag{6}$$

Advantage of using series-parallel mode is that stability is guaranteed (as plant's output are being used to evaluate RBFN next output and plant is assumed to be BIBO stable), whereas this is not guaranteed in case of parallel mode as BIBO stability of RBFN cannot be guaranteed. The general series-parallel identification configuration based on RBFN is shown in Fig. 2.

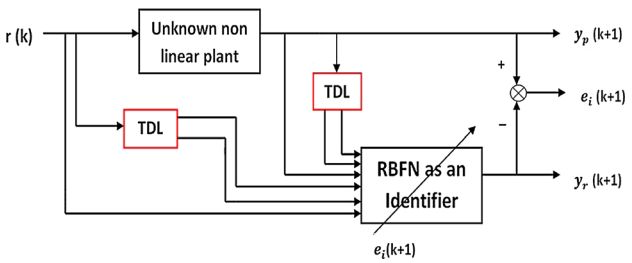

Fig. 2 Identification of dynamics of unknown plant using series-parallel RBFN identification model

4 Algorithm for RBFN identification model

In online training, all the parameters of RBFN will be adjusted at every instant of time using the training sample available at that instant. The training involves the use of recursive update equations for adjusting the parameters of RBFN. These equations are obtained using gradient descent principle. The update equations for RBFN identification model's parameters are derived as:

Let $E(k)$ represent instantaneous cost function value (instantaneous mean square error) at kth instant, and it is defined as:

$$E(k) = \frac{1}{2}(y_p(k) - y_r(k))^2 \tag{7}$$

where $y_p(k)$ is the desired output (plant's output whose dynamics are unknown and need to be identified) and $y_r(k)$ is the output of RBFN at kth time instant. Differentiating $E(k)$ w.r.t. $h_{ij}(k)$ will provide the rate of change of E(k) w.r.t. each element of each radial centre, where $i = 1$ to n, $j = 1$ to m, and this requires the use of chain rule as number of signals are present between $E(k)$ and $h_{ij}(k)$.

$$\frac{\partial E(k)}{\partial h_{ij}(k)} = \left(\frac{\partial E(k)}{\partial y_r(k)} \times \frac{\partial y_r(k)}{\partial \phi_j(k)} \times \frac{\partial \phi_j(k)}{\partial h_{ij}(k)}\right) \tag{8}$$

or

$$\frac{\partial E(k)}{\partial h_{ij}(k)} = \left(\frac{\partial E(k)}{\partial y_r(k)} \times \frac{\partial y_r(k)}{\partial \phi_j(k)} \times \frac{\partial \phi_j(k)}{\partial z_j(k)} \times \frac{\partial z_j(k)}{\partial h_{ij}(k)}\right) \tag{9}$$

where $\frac{\partial E(k)}{\partial y_r(k)} = -e_i(k)$, $\frac{\partial \phi_j(k)}{\partial z_j(k)} = -z_j(k) \times \frac{\phi_j(k)}{\sigma_j^2(k)}$ and $\frac{\partial z_j(k)}{\partial h_{ij}(k)} = \left(\frac{\partial \sum_j ((x_j(k) - h_{ij}(k))^2)^{\frac{1}{2}}}{\partial h_{ij}(k)}\right)$. Simplifying it further, we get

$$\frac{\partial z_j(k)}{\partial h_{ij}(k)} = -\left(\frac{x_j(k) - h_{ij}(k)}{z_j(k)}\right) \tag{10}$$

Thus, the radial centres update equations along with momentum term are given as:

$$h_{ij}(k+1) = h_{ij}(k) + \Delta h_{ij} + \alpha \Delta h_{ij}(k-1) \tag{11}$$

where

$$\Delta h_{ij} = \eta e_i(k) w_j(k) \frac{\phi_j(k)}{\sigma_j^2(k)} \left(x_j(k) - h_{ij}(k)\right) \tag{12}$$

η is known as learning rate with its value chosen anywhere between (0, 1] and $\alpha \Delta h_{ij}(k-1)$ denotes the momentum term and $0 \leq \alpha \leq 1$ and is known as momentum constant. Update rule for adjusting the weights can be similarly evaluated as:

$$\frac{\partial E(k)}{\partial w_j(k)} = \left(\frac{\partial E(k)}{\partial y_r(k)} \times \frac{\partial y_r(k)}{\partial w_j(k)}\right) \tag{13}$$

where $\frac{\partial E(k)}{\partial y_r(k)} = -(y_p(k) - y_r(k)) = -e_i(k)$ and $\frac{\partial y_r(k)}{\partial w_j(k)} = \phi_j(k)$ Thus, every element in $w = [w_1, w_2, w_3, \ldots, w_L]$ is updated as

$$w_j(k+1) = w_j(k) + \Delta w_j + \alpha \Delta w_j(k-1) \tag{14}$$

where $\Delta w_j = \eta e_i(k) \phi_j(k)$

Finally, adjustments in the widths of radial centres can be obtained as:

$$\frac{\partial E(k)}{\partial \sigma_j(k)} = \left(\frac{\partial E(k)}{\partial y_R(k)} \times \frac{\partial y_R(k)}{\partial \phi_j(k)} \times \frac{\partial \phi_j(k)}{\partial \sigma_j(k)}\right) \tag{15}$$

where $\frac{\partial E(k)}{\partial y_R(k)} = -(y_p(k) - y_R(k)) = -e_i(k)$, $\frac{\partial y_R(k)}{\partial \phi_j(k)} = w_j(k)$ 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 $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_L)$ will be updated as:

$$\sigma_j(k+1) = \sigma_j(k) + \Delta \sigma + \alpha \Delta \sigma(k-1) \tag{16}$$

where $\Delta \sigma = \eta e_i(k) w_j(k) \frac{\phi_j(k) z_j^2(k)}{\sigma_j^3(k)}$.

Figure 3 shows the above steps in the form of a flow chart.

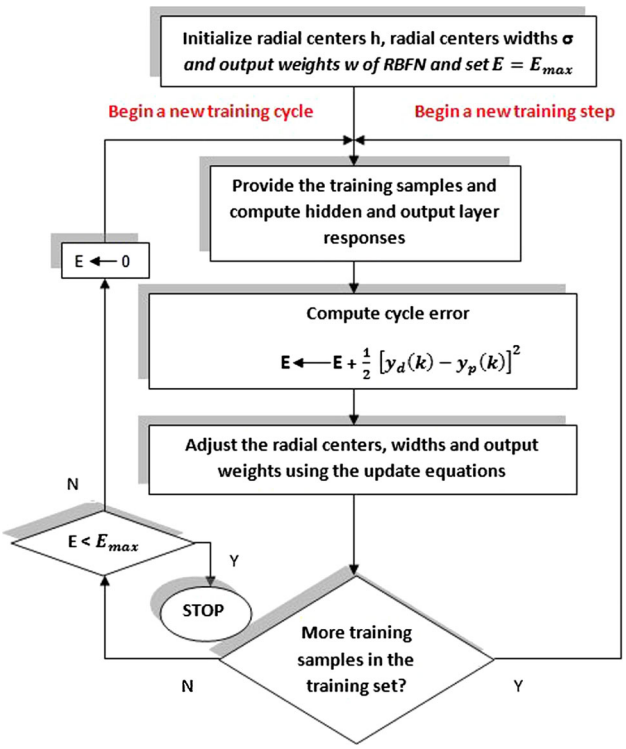

Fig. 3 Flow chart showing the steps of updating algorithm

5 Adaptive control based on RBFN

There are two approaches available for adaptive control of dynamical systems:

1. Direct control
2. Indirect control

In direct control, stable laws exist for linear time-invariant (LTI) systems and the adjustment of controller parameters only requires the value of error which is computed from the plant's and reference model outputs [16, 23]. But such stable laws do not exist for RBFN so indirect control will be used [3, 4]. In indirect adaptive control scheme, mathematical model of plant is required since the control algorithm requires its output value for generating the controller's output. This information of the dynamics of the plant will be given by the RBFN identification model as it will be able to approximate the plant's dynamics during the online control. The other use of RBFN identification model is that it can change its parameters values in response to parameter variations or disturbance signals affecting the plant. Figure 4 shows the RBFN-based configuration for simultaneous online indirect adaptive control and identification of one-link robotic manipulator. The inputs to the RBFN controller in the proposed method are $r(k)$, $(y_p(k), y_p(k-1)\ldots y_p(k-n+1))$ and $uc(k-n+1)$. The idea of using the plant's present and past outputs and controller's past output along with the external input is to make the control algorithm more informative which results in the computation of more accurate controller output. Thus, the reference input $r(k)$, the plant's present output, $y_p(k)$, as well as its past $n-1$ values and $n-1$ past values of $uc(k)$ (i.e. RBFN controller outputs), will constitute the inputs to the controller.

5.1 RBFN controller algorithm

To derive update equations for RBFN controller, similar approach as adopted in deriving the update equations for RBFN identification model is followed. The cost function $E(k)$ at any kth instant is defined using the instantaneous error between desired trajectory and robotic manipulator (plant) output:

$$E(k) = \frac{1}{2}(y_d(k) - y_p(k))^2 \tag{17}$$

where $y_p(k)$ is the output of the plant and $y_d(k)$ is the reference trajectory output at kth instant. Now differentiating $E(k)$ w.r.t. $hij(k)$, we will get:

$$\frac{\partial E(k)}{\partial h_{ij}(k)} = \left(\frac{\partial E(k)}{\partial y_p(k)} \times \frac{\partial y_p(k)}{\partial uc(k)} \times \frac{\partial uc(k)}{\partial \phi_j(k)} \times \frac{\partial \phi_j(k)}{\partial z_j(k)} \times \frac{\partial z_j(k)}{\partial h_{ij}(k)}\right) \tag{18}$$

$$\frac{\partial E(k)}{\partial h_{ij}(k)} = -e_c(k) \times J(k) \times w_j(k) \times \left(\frac{\partial \phi_j(k)}{\partial z_j(k)} \times \frac{\partial z_j(k)}{\partial h_{ij}(k)}\right) \tag{19}$$

where $\frac{\partial y_p(k)}{\partial uc(k)} = J(k)$.

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

The calculation of J(k) is based on the knowledge of the plant's dynamics, and RBFN identification model will approximate its value (as RBFN mathematical structure is known). The rest of the partial derivatives can be evaluated in the same way as done for the case of RBFN identification model.

In Fig. 4, the symbol $e_c(k)$ 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 the 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.

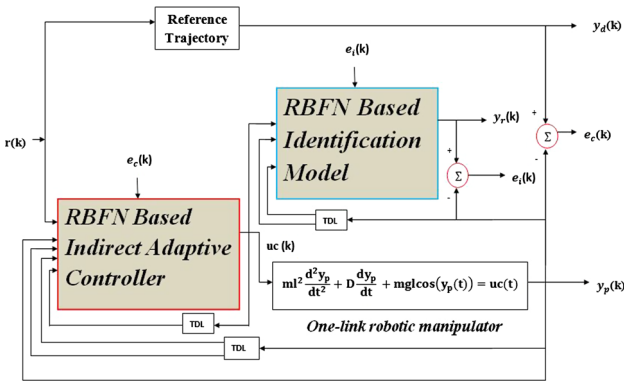

Fig. 4 Indirect adaptive control and identification using RBFN

6 Simulation results

We have performed comparative analysis of performance of RBFN and MLFFNN by taking both one-link and two-link robotic manipulators as the test plants. Let us begin with one-link robotic manipulator.

6.1 Application of RBFN on one-link robotic manipulator

A general dynamical model for representing n serial links robotic manipulator is given by:

$$M(y_p)\ddot{y_p} + C(y_p, \dot{y_p})\dot{y_p} + D\dot{y_p} + G(y_p) = \tau \tag{20}$$

Here y_p, $\dot{y_p}$ and $\ddot{y_p} \in R^l$ represent the position of joint, velocity and the accelerations, respectively. The $l \times l$ real matrices: M, C, D, represent the symmetric positive

definite inertia matrix, centrifugal Coriolis matrix and positive definite diagonal matrix for damping friction coefficients of each joint, respectively. The real $l \times l$ matrix $G(y_p)$ denotes the gravity term, and $\tau \in R^l$ is the input vector torques on joints. In case of one-link robotic manipulator, the value of l is 1. The detailed information regarding the kinematics of robotic manipulator can be found in the literature [17, 19]. In the simulation study carried out in this paper, the performance of RBFN identification model and controller is compared with the MLFFNN identification model and controller, respectively, which is also simulated under same conditions: value of η and α is taken to be same in MLFFNN and RBFN case, the number of hidden layer is taken to be 1 in both, and a number of hidden neurons in MLFFNN are considered to be equal in both the structures. The dynamics of a one-link robotic manipulator can be described by the following second-order differential equation [18]:

$$ml^2\frac{d^2y_p(t)}{dt^2} + D\frac{dy_p(t)}{dt} + mglcos(y_p(t)) = uc(t) \tag{21}$$

where $y_p(t)$ represents the angular position of the robotic manipulator arm, l represents the link length, $D\frac{dy_p(t)}{dt}$ is a viscous friction torque, and g = acceleration due to gravity $= 9.8\,\mathrm{m/s^2}$. For simplicity, the values of $m = l = D = 1$ are used in this paper. The control input torque to the robotic arm is represented by $uc(t)$. If the movement plane of robot is parallel with the horizontal plane, then the gravity term can be ignored [18]. The state-space representation of above dynamical differential equation is given by:

$$\begin{bmatrix} \dot{y_p} \\ \ddot{y_p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \dfrac{-gcos(y_p)}{l} & \dfrac{-D}{ml^2} \end{bmatrix} \begin{bmatrix} y_p \\ \dot{y_p} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{ml^2} \end{bmatrix} [uc] \tag{22}$$

To implement the control algorithm in the programming environment, the differential equation of one-link robotic manipulator is discretized by using a sampling period of $T = 0.45$ s and results in the following difference equation:

$$y_p(n+2) = F[y_p(n+1), y_p(n)] + T^2uc(n) \tag{23}$$

where

$$F[y_p(n+1), y_p(n)] = (2-T)y_p(n+1) + y_p(n)(T-1) - 9.8T^2cos(y_p(n)) \tag{24}$$

is the nonlinear part in the dynamical equation of the one-link robotic manipulator. Since Eq. 21 is of second order, to compute next value $y_p(n+2)$ of y_p, we need its previous two values: $y_p(n+1), y_p(n)$. The differential equation of the reference trajectory is shown below:

$$\frac{d^2y_d(t)}{dt^2} = -2.5y_d(t) - 1.5\frac{dy_d(t)}{dt} + r(t) \tag{25}$$

where $r(t)$ is an externally applied input signal and is considered to be $sin(t)$. The equivalent difference equation of the desired trajectory is obtained by using the same sampling period = 0.45 and is given by:

$$y_d(n+2) = y_d(n+1)(2-1.5T) + y_d(n)(-1-2.5T^2+1.5T) + T^2r(n) \tag{26}$$

The RBFN identification model for the unknown nonlinear dynamics of one-link robotic manipulator will be:

$$y_r(k+2) = RBFN\{y_p(n+1), y_p(n)\} + T^2uc(n) \tag{27}$$

The order n of the plant is 2 so $r(k)$, $y_p(k), y_p(k-1)$ and $uc(k-1)$ will constitute the inputs to the RBFN controller. Figure 5 shows the response of one-link robotic manipulator without the RBFN controller action. It can be seen from the figure that the output of one-link robotic manipulator is not following the desired trajectory, and hence, control action is required. The configuration as shown in Fig. 4 was set up to simultaneously control the one-link robotic manipulator trajectory and identify its dynamics using RBFN controller and RBFN identification model, respectively. The similar configuration containing MLFFNN was also set up separately to perform the same task. The values of parameters used in both MLFFNN and RBFN are: $\eta = 0.039$, $\alpha = 0.23$, and a number of radial centres and neurons in the hidden layer of RBFN and MLFFNN identification model are taken to be 30. Figure 6 shows the response of MLFFN and RBFN identification models during the initial phase of training. It is clear from Fig. 6 that the parameters of RBFN identification model (identifier) are getting trained more quickly than that of MLFFNN identifier as response of RBFN identification model is approaching more closer to one-link robotic manipulator (plant) response in comparison with response of the MLFFNN identifier during the online identification

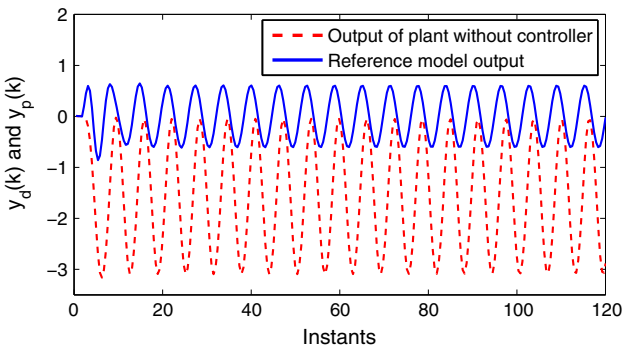

Fig. 5 Response of one-link robotic manipulator without control

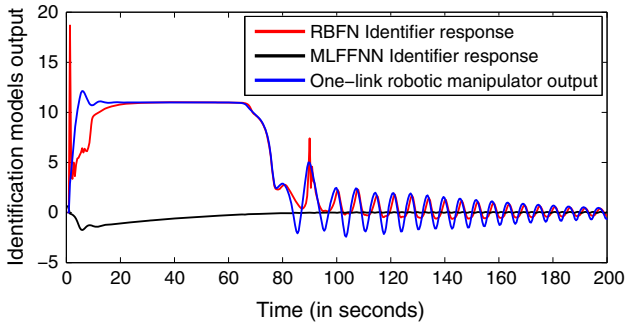

Fig. 6 Initial response of RBFN and MLFFNN identification models during initial stage of online training

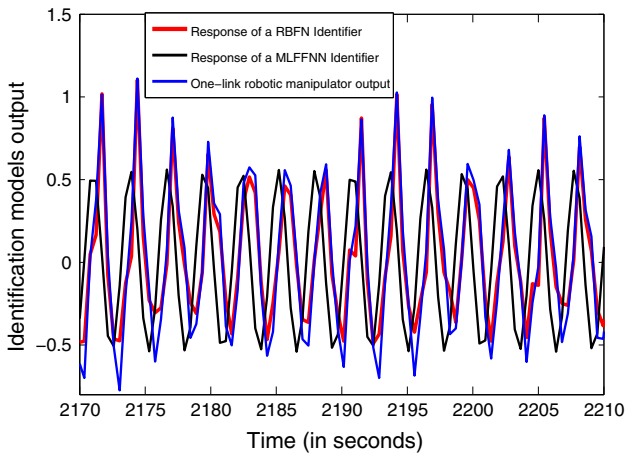

Fig. 7 Response of RBFN and MLFFNN identification models during the online training

and control. From Fig. 7, as the training continued, the response of both RBFN and MLFFNN identifiers improved, but degree of improvement is more in case of RBFN identifier. After sufficient time for which online identification and control underwent, the final response of MLFFNN and RBFN identification models is shown in Fig. 8. From the figure, it can be concluded that RBFN response is closer to plant response as compared to MLFFNN identifier. The indirect adaptive control was also going on along with the identification process, and Figs. 9, 10 and 11 show the initial, midway and final response of one-link robotic manipulator with MLFFNN (continuous black curve) and RBFN controller (continuous red curve), respectively.

Here, also RBFN has provided much better response as compared to response of plant obtained with MLFFNN.

Figure 12 shows the plot of RBFN and MLFFNN controller's output response, and it is clear that response of RBFN controller becomes stable much earlier than that of MLFFNN. The MSE error plot for one-link robotic manipulator under RBFN controller action is shown in

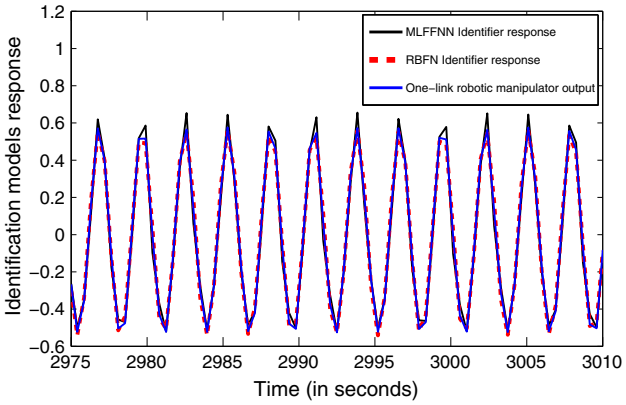

Fig. 8 Final stage of RBFN and MLFFNN identification models response during the online training

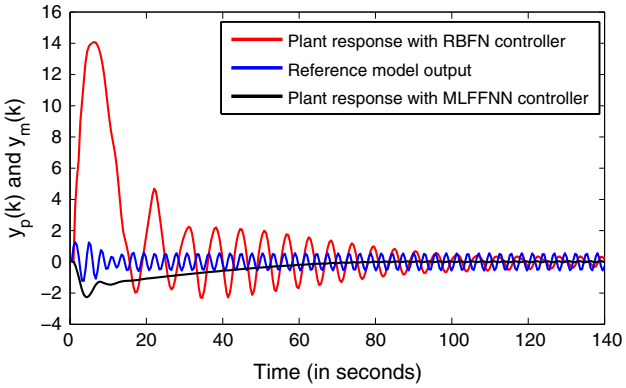

Fig. 9 Initial response of One-link robotic manipulator with RBFN and MLFFNN controller action

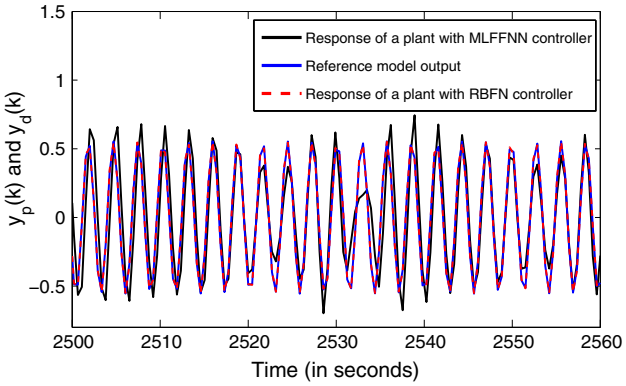

Fig. 10 One-link robotic manipulator response under RBFN and MLFFNN controller action

Fig. 13. The MSE error plot for RBFN and MLFFNN identification model action is shown in Fig. 14. From the plots, it can be seen that the instantaneous error in case of RBFN drops to zero value at a much faster rate as compared to MLFFNN.

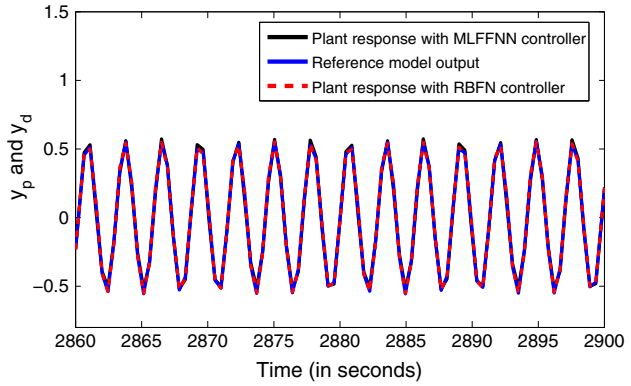

Fig. 11 One-link robotic manipulator response after sufficient training

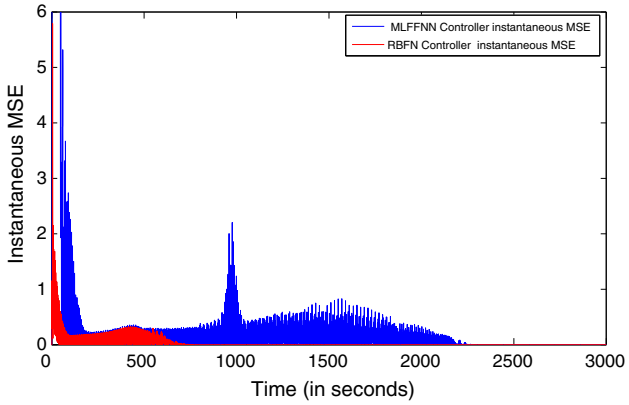

Fig. 13 MSE under RBFN controller action

6.1.1 *Performance of RBFN and MLFFNN with disturbance signal*

Now, the performance of MLFFNN and RBFN controllers and identification models is checked for disturbance signal. The disturbance signal affected the system at $k = 3149$th s which leads to the rise in the MSE of plant. From Figs. 15 and 16, it can be seen that MSE of plant under RBFN action settled very quickly after the impact of disturbance signal as compared to MSE obtained with MLFFNN which actually took more time to reach again to the zero value.

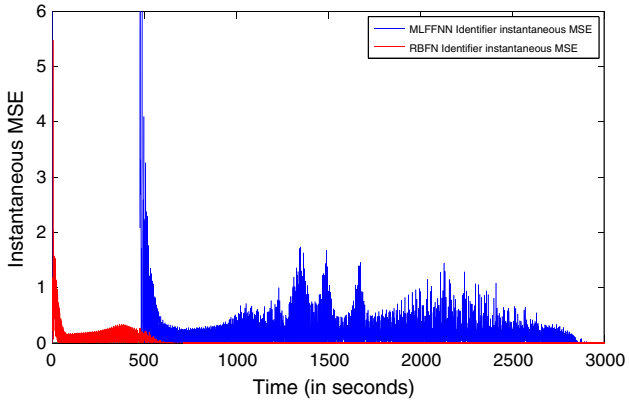

Fig. 14 MSE of RBFN and MLFFNN identification models

6.1.2 *Performance evaluation with different leaning rates*

To test the performance of plant under RBFN controller action, different learning rates, η, were considered, and the corresponding average MSEs were calculated for both RBFN and MLFFNN. From Figs. 17 and 18, it can be seen that average MSEs in case of RBFN controller and identifier are smaller as compared to that obtained from MLFFNN controller and identifier, respectively.

6.1.3 *Performance evaluation of RBFN and MLFFNN with different external input*

For further testing of the RBFN identification and controller, a different external input was applied. It is defined as:

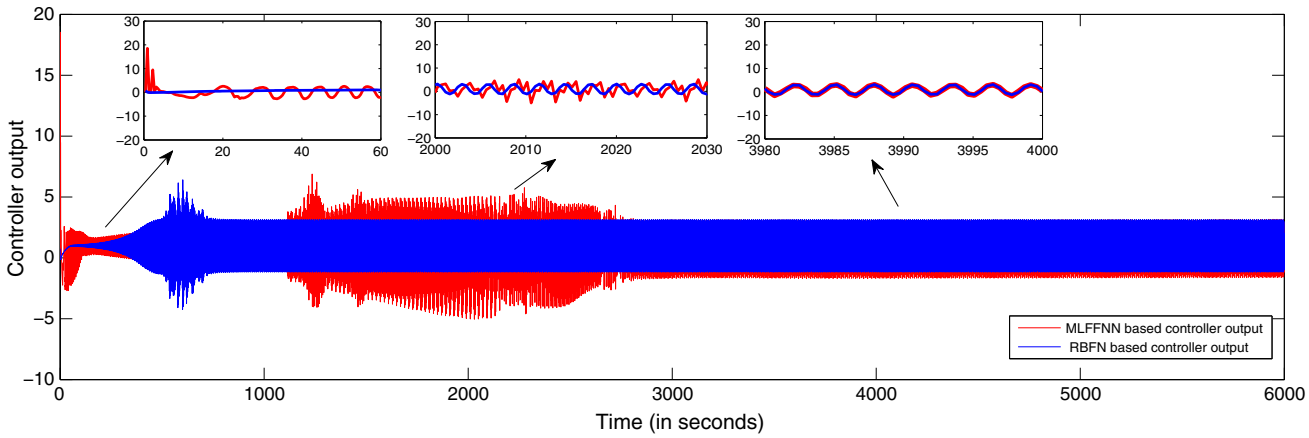

Fig. 12 RBFN and MLFFNN controller's outputs during online training

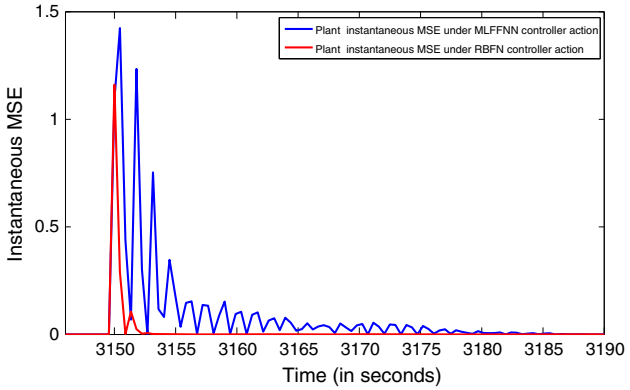

Fig. 15 MSE of RBFN and MLFFNN controllers with disturbance

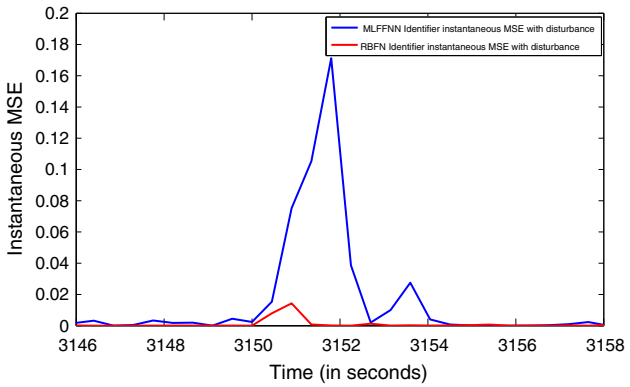

Fig. 16 MSE of RBFN and MLFFNN identification models with disturbance

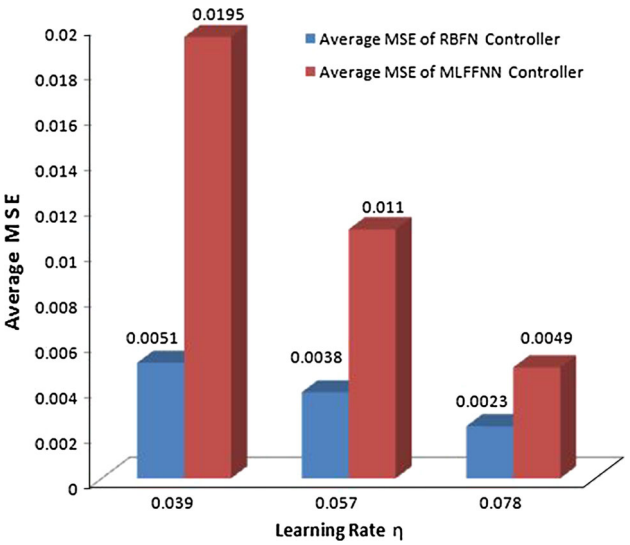

Fig. 17 Average MSE of RBFN and MLFFNN controllers with different learning rates

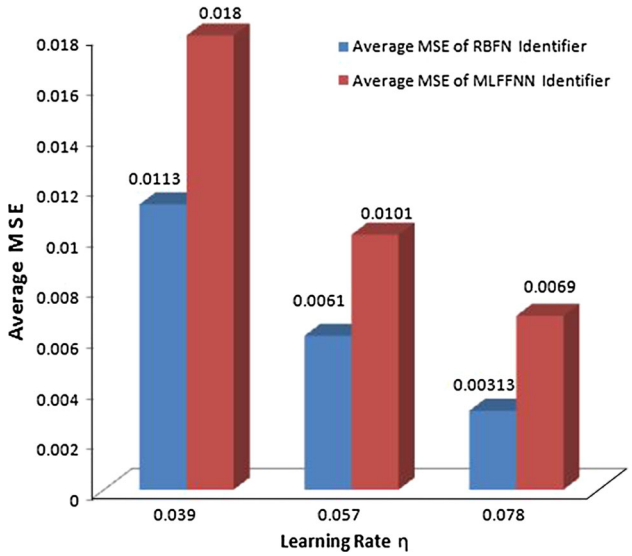

Fig. 18 Average MSE of RBFN and MLFFNN identification models with different learning rates

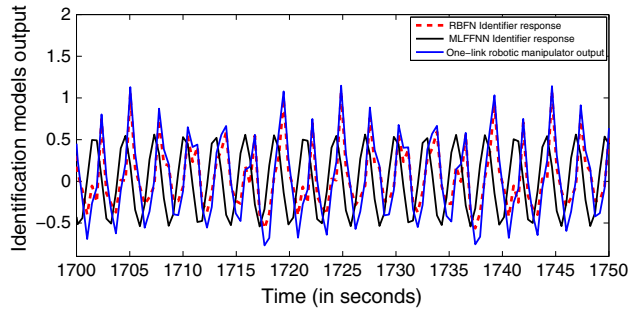

Fig. 19 Response of RBFN and MLFFNN identifiers during initial stage of training

$$r(k) = sin(k) + 0.1 \quad \text{for } k <= 6750 \tag{28}$$

and

$$r(k) = 0.45sin(k) + 0.15cos(k) \quad \text{for } k > 6750 \tag{29}$$

Figures 19 and 20 show the response of RBFN and MLFFNN identification models during the initial and final stages of online learning. Here, also the performance of RBFN identifier in terms of capturing the dynamics of plant was proven superior to that of MLFFNN identifier.

6.1.4 *Testing of performance of RBFN and MLFFNN at the instant of input change*

Now, the external input gets changed at $k = 6750$th instant, and the corresponding variation occurring in the responses of RBFN and MLFFNN identification models is shown in

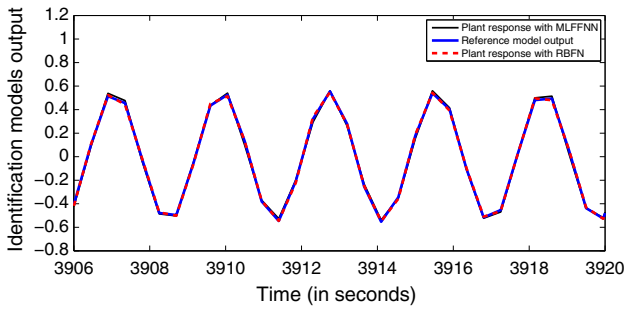

Fig. 20 Response of RBFN and MLFFNN identifiers after sufficient online training

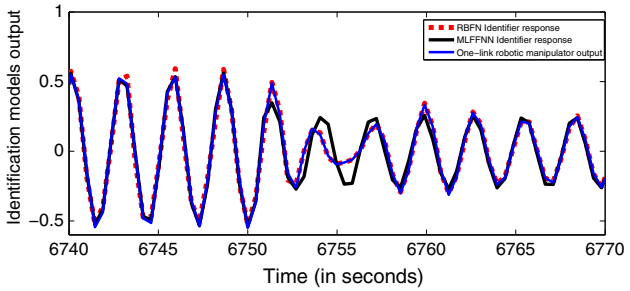

Fig. 21 Response of RBFN and MLFFNN identifiers at the time of input change

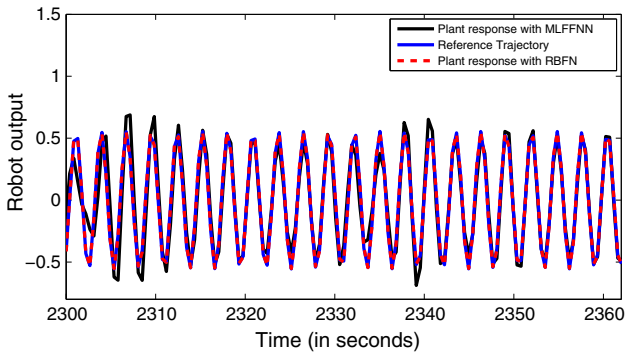

Fig. 22 Response of one-link robotic manipulator under controller action

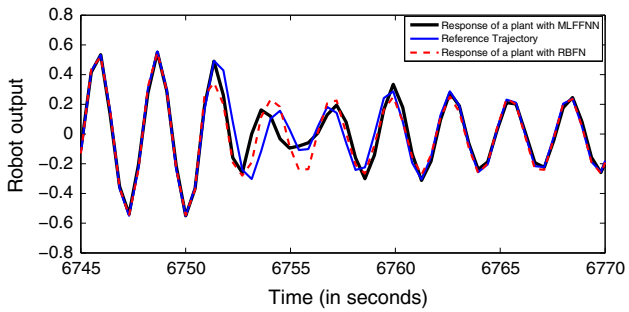

Fig. 24 Response of one-link robotic manipulator under controller action at the time of input change

Fig. 21. Figure 22 shows the one-link robotic manipulator under controller action, Fig. 23 shows the response of RBFN and MLFFNN controller outputs with new external input.

Figure 24 shows the response of plant at the time of input change and Fig. 25 shows the variation occurred in MSE of plant under RBFN and MLFFNN controllers action respectively at the time of input change. From the figures, it can be seen that performance of RBFN-based controller and identification model is much better than that of MLFFNN.

6.1.5 Performance evaluation under parameter variation

Now, the RBFN and MLFFNN controllers and identification models are tested for parameter variation. Figure 26 shows the variation occurred in MSE of RBFN and MLFFNN controllers when system parameter undergoes change at $k = 5400$th instant. From Fig. 27 it can be seen that at the instant of parameter variation there occur more changes in the output values of MLFFNN identifier as compared to the RBFN identifier. Also, Fig. 28 shows the variation occurred in MSE of RBFN and MLFFNN identification models at the same instant.

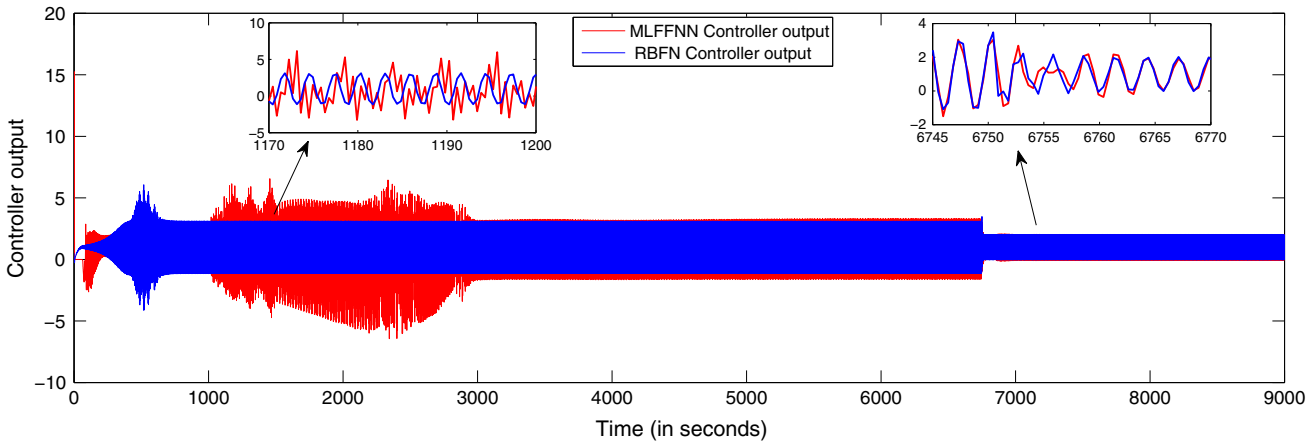

Fig. 23 RBFN and MLFFNN controller's response with new input

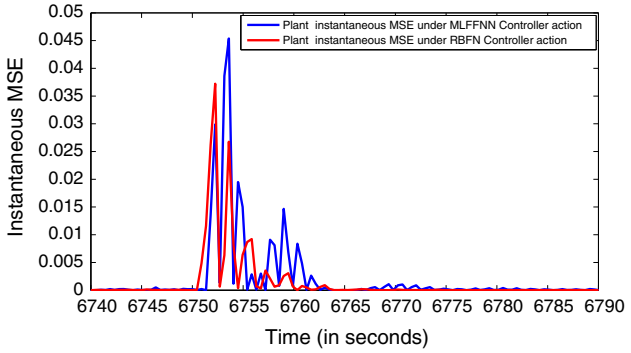

Fig. 25 Variation in MSE of RBFN and MLFFNN controllers at the time of input change

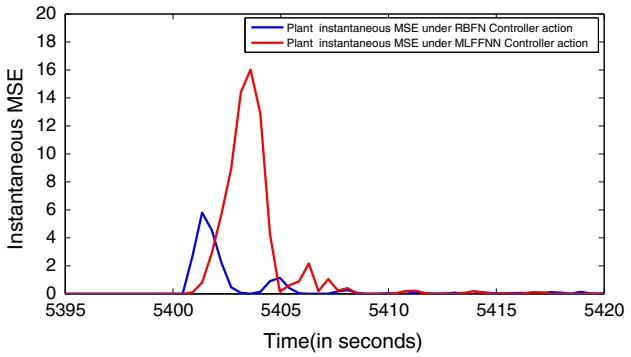

Fig. 26 Variation in MSE of plant under RBFN and MLFFNN controller action at the time of parameter variation

6.2 Application of RBFN controller to two-link robotic manipulator

The general dynamical nonlinear differential equation describing the dynamics of any l-link robotic manipulator is already given in Eq. 20. For the case of two-link (i.e. l= 2) robotic manipulator, the corresponding matrices are defined as [1]

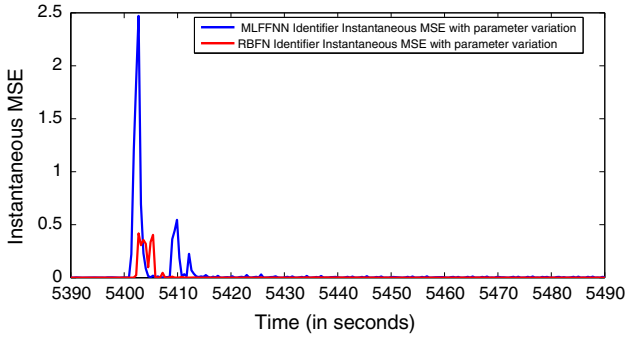

Fig. 28 Variation in MSE of RBFN and MLFFNN identification models at the time of parameter variation

$$M(\theta) = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} \tag{30}$$

where $x_{11} = \left(\frac{1}{4}m_1 + m_2\right)l_1^2 + \frac{1}{4}m_2l_2^2 + m_2l_1l_2cos(\theta_2)$, $x_{12} = \frac{1}{4}m_2l_2^2 + \frac{1}{2}m_2l_1l_2cos(\theta_2)$, $x_{21} = \frac{1}{4}m_2l^2 2 + \frac{1}{2}m_2l_1l_2cos(\theta_2)$

$$G(\theta) = \begin{bmatrix} \left(\frac{1}{2}m_1 + m_2\right)gl_1cos(\theta_1) + \frac{1}{2}m_2l_2gcos(\theta_1 + \theta_2) \\ \frac{1}{2}m_2gl_2cos(\theta_1 + \theta_2) \end{bmatrix} \tag{31}$$

$$C(\dot{\theta}, \ddot{\theta})\dot{\theta} = \begin{bmatrix} -\frac{1}{2}m_2l_1l_2(2\dot{\theta_1}\dot{\theta_2} + \dot{\theta_2^2})sin(\theta_2) \\ \frac{1}{2}m_2l_1l_2\dot{\theta_1^2}sin(\theta_2) \end{bmatrix} \tag{32}$$

where θ_1 and θ_2 represent the angular position of link 1 and link 2, respectively; m_1 and m_2 are mass of link 1 and link 2, respectively, and l_1 and l_2 are lengths of link 1 and link 2, respectively. Thus, there are two outputs of two-link robotic manipulator which are required to be controlled by the RBFN controller such that each output of manipulator follows the desired trajectory. To accomplish this task,

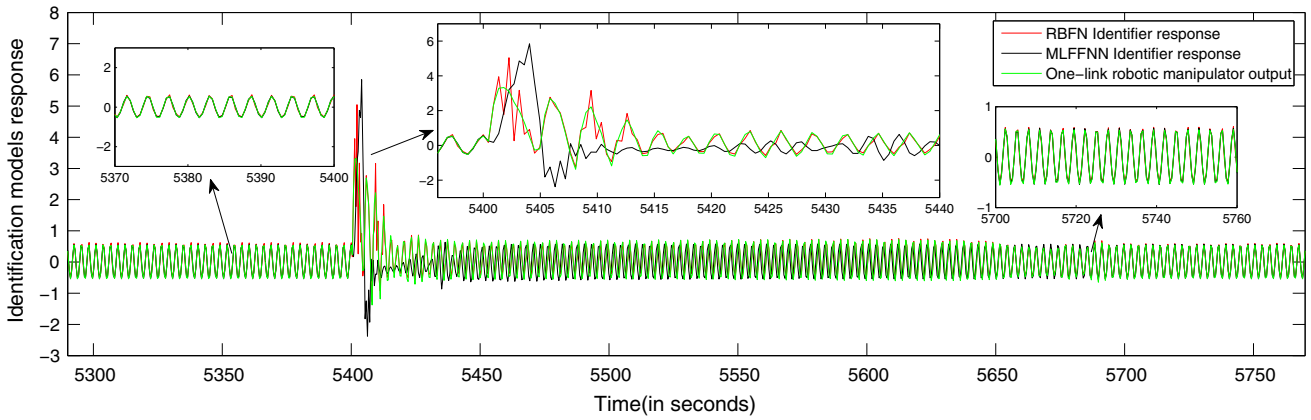

Fig. 27 Response of RBFN and MLFFNN identification models with new input when parameter of system undergoes variation

Table 1 Two-link planar robotic manipulator parameters values

Parameters	Link 1	Link 2
Mass (kg)	0.5	0.5
Link length (m)	0.8	0.4
Acceleration due to gravity (g) (m/s^2)	9.8	9.8

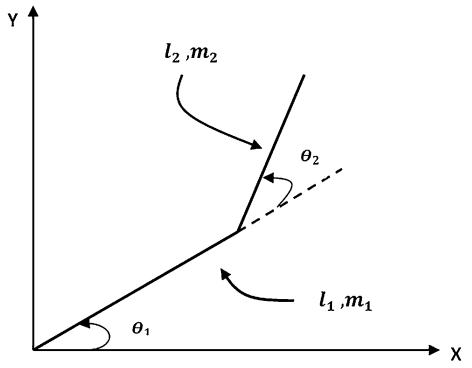

Fig. 29 Two-link planar robotic manipulator

RBFN controller with two outputs (hence two neurons in the output layer of RBFN controller) will be used. These controller outputs or controlling torques ($\tau_1(k)$ and $\tau_2(k)$) at any given instant of time are denoted by $uc_1(k)$ and $uc_2(k)$. The relationship between the RBFN controller outputs and manipulator links angular position is given by

$$uc_1(t) = l_2^2 m_2\left(\frac{d^2\theta_1}{dt^2}\right) + l_1^2(m_1+m_2)\left(\frac{d^2\theta_1}{dt^2}\right) + m_2 g l_2 cos(\theta_1) \times cos(\theta_2) + (m_1+m_2)l_1 g cos(\theta_1) - m_2 l_1 l_2 sin(\theta_2) \times \left(\frac{d\theta_2}{dt}\right)^2 - 2m_2 l_1 l_2 sin(\theta_2)\left(\frac{d\theta_1}{dt}\right)\left(\frac{d\theta_2}{dt}\right) + m_2 l_1 l_2 cos(\theta_2)\left(2\frac{d^2\theta_1}{dt^2} + \frac{d^2\theta_2}{dt^2}\right) \tag{33}$$

$$uc_2(t) = m_2 l_1 l_2 sin(\theta_2)\left(\frac{d\theta_1}{dt}\right)^2 + m_2 l_1 l_2 cos(\theta_2)\left(\frac{d^2\theta_1}{dt^2}\right) + m_2 l_2^2\left(\frac{d^2\theta_1}{dt^2} + \frac{d^2\theta_2}{dt^2}\right) + m_2 l_1 g cos(\theta_1) cos(\theta_2) \tag{34}$$

where $uc_1(t) = \tau_1(t)$ and $uc_2(t) = \tau_2(t)$. By rearrangement of terms in Eqs. 33 and 34, we can find the expressions of $\theta_1(k)$ and $\theta_2(k)$. Table 1 shows the values of various parameters used in the simulation study, and Fig. 29 shows the structure of two-link planar robotic manipulator.

It can be seen that Eqs. 33 and 34 are highly nonlinear and coupled to each other. Both these equations are discretized using sampling time of $T = 0.001$ s in order to obtain their equivalent difference equations. Since the order n of two-link robotic manipulator is 2, total of 4 signals will be used as input signals for the RBFN controller. These 4 inputs of RBFN are $\{e_1(k), e_2(k), \tau_1(k-$

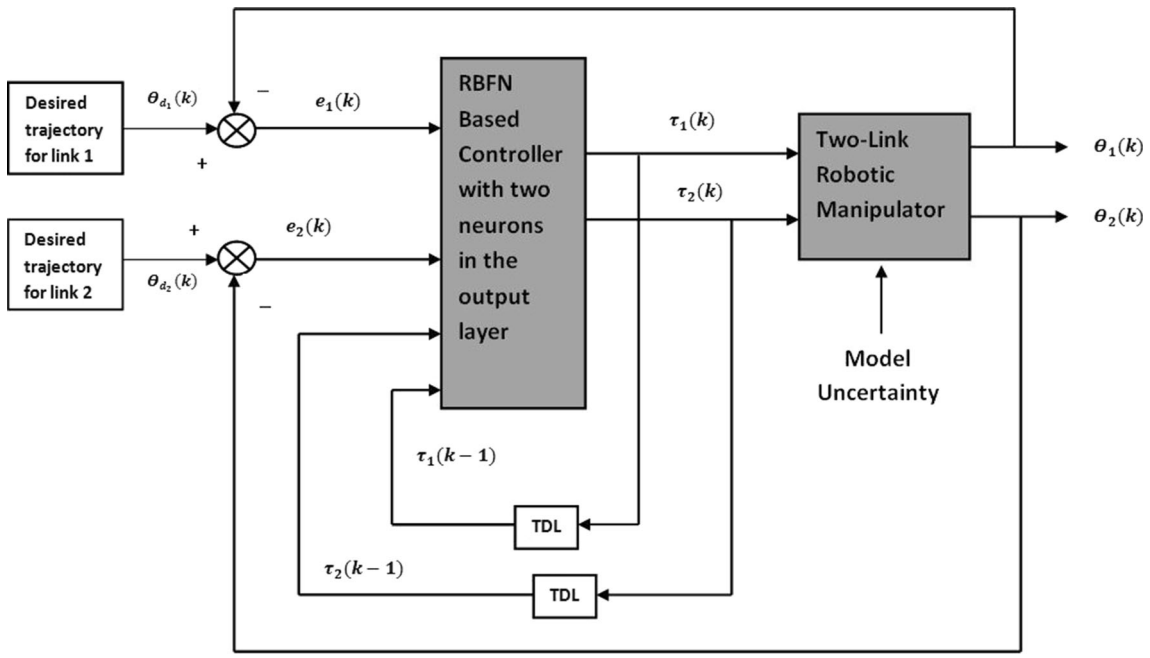

Fig. 30 RBFN-based control configuration for trajectory control of two-link robotic manipulator

$1), \tau_2(k-1)\}$ where $e_1(k) = \theta_{d_1}(k) - \theta_1(k)$ and $e_2(k) = \theta_{d_2}(k) - \theta_2(k)$. The rationale of choosing these 4 signals to be the inputs of RBFN controller is to provide it the information regarding its past output values, past values of plant and external inputs. This will make RBFN controller to generate an appropriate outputs for making the robotic manipulator's outputs to follow the desired trajectory. Also, $\theta_{d_1}(k)$ and $\theta_{d_2}(k)$ are desired angular positions for link 1 and link 2, respectively. The control configuration is shown in Fig. 30.

6.2.1 Properties of robotic manipulator dynamics

Following are the properties which are satisfied by any revolute rigid-link manipulators:

1. Matrix M(θ) is a positive definite symmetric matrix, i.e. $M = M^T$
2. Matrix M(θ) is bounded, i.e. $\mu_1(\theta)I \leq M(\theta) \leq \mu_2(\theta)I$, where $\mu_1(\theta)$ and $\mu_2(\theta)$ are scalar quantities and are constants for the revolute links. I is an identity matrix.
3. Matrix $\dot{M} - 2C$ is a skew symmetric matrix which means given any vector X, $X^T(\dot{M} - 2C)X = 0$.
4. $C(\theta, \dot{\theta})\dot{\theta}$ is quadratic with respect to $\dot{\theta}$ and is bounded as $\left\|C(\theta, \dot{\theta})\dot{\theta}\right\| \leq \mu_3(\theta)\left\|\dot{\theta}\right\|^2$, where $\mu_3(\theta)$ represents a scalar constant for revolute links.
5. Gravity vector G is bounded as $\|G(\theta)\| \leq \mu_4(\theta)$, where $\mu_4(\theta)$ is a constant scalar in case of revolute links and is independent of θ.

6.2.2 Trajectory tracking performance of two-link robotic manipulator

Consider the following desired trajectories to be followed by the two links of robotic manipulator:

$$\theta_{d_1} = sin(2\pi fk) \tag{35}$$

and

$$\theta_{d_1} = 2sin(2\pi fk) \tag{36}$$

where $f = \frac{1}{3}$Hz.

6.2.3 For link 1

Response of Ist link with RBFN and MLFFNN controller is shown in Fig. 31. The sampling time is chosen to be $T = 0.001$ s. Values of learning rate and momentum constant are set to $\eta = 0.0025$ and $\alpha = 0.033$, respectively. From the figure it can be seen that response of Ist link is better when RBFN controller is in the control loop as compared to response obtained when

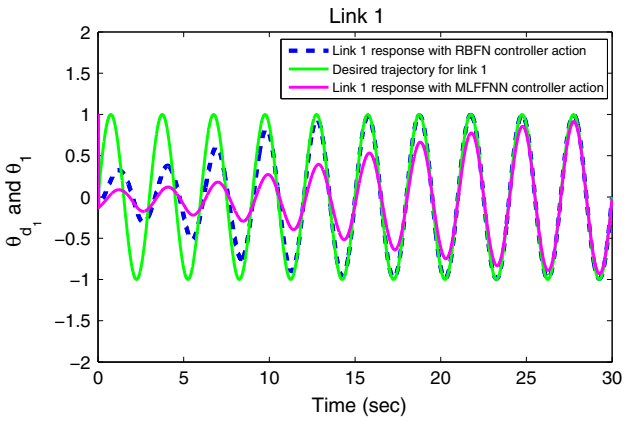

Fig. 31 Response of link 1 under controller action

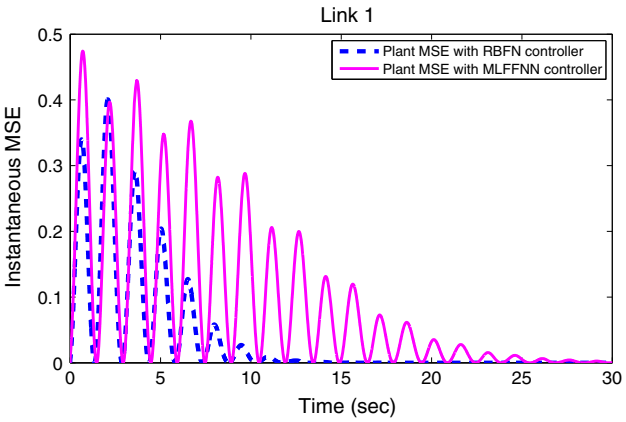

Fig. 32 Instantaneous MSE for link 1 under controller action during the training

MLFFNN is used as a controller. Response of Ist link started following the desired trajectory at around 15th time instant with RBFN controller action, whereas it converged to desired trajectory at around 27th time instant when MLFFNN controller is used. The corresponding MSE plot is shown in Fig. 32. From the MSE plot also it can be seen that RBFN controller got trained much faster as compared to MLFFNN controller as instantaneous MSE with RBFN controller reduces to zero much quickly as compared to MSE obtained with MLFFNN controller.

6.2.4 For link 2

Now, the response of link 2 is shown in Fig. 33 under controller action. From Fig. 33 we can see that link 2 tracking performance with RBFN controller action is better as compared to its performance obtained with the MLFFNN. The corresponding instantaneous MSE plot for link 2 under controller action is shown in Fig. 34.

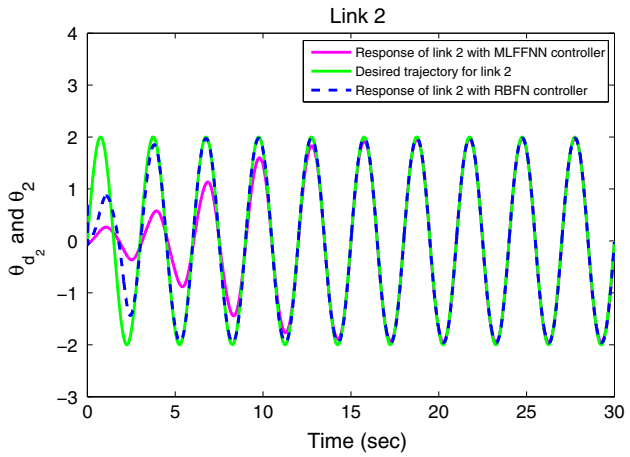

Fig. 33 Response of link 1 under controller action during the training

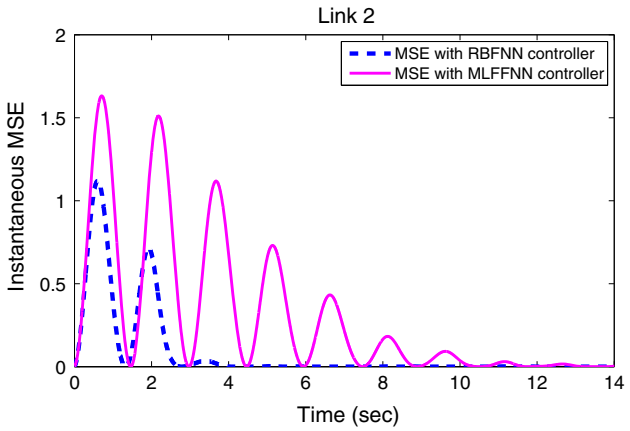

Fig. 34 Instantaneous MSE for link 2 under controller action

Again, MSE with RBFN controller action reduces quickly to zero as compared to MSE obtained with MLFFNN controller. This shows the superior performance of RBFN controller over MLFFNN-based controller.

6.2.5 Testing of performance of RBFN and MLFFNN controller under parameter variation

A mass change of 2.5 kg each in m_1 and m_2 is considered for testing the performance of RBFN and MLFFNN controllers. This variation in link 1 and link 2 mass is introduced at 31^{st} time instant. The corresponding response of link 1 at the time of mass variation is shown in Fig. 35.

From the figure, it can be seen that performance of both RBFN and MLFFNN is similar, but on a closer look, it can be seen that link 1 response recovered slightly quickly under RBFN controller action as compared to response obtained with MLFFNN controller. This conclusion is verified from the MSE plot obtained at the instant of parameter variation. It is shown in Fig. 36. From the plot, it

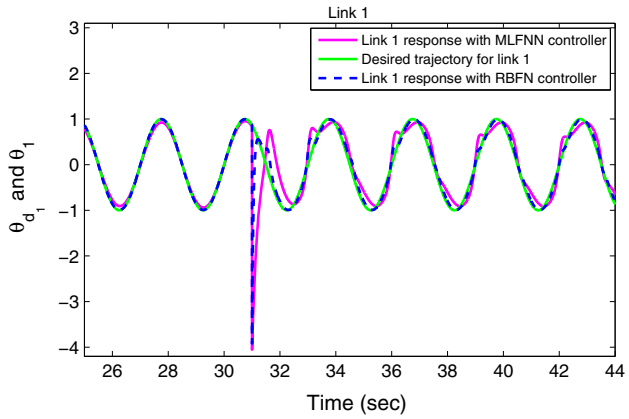

Fig. 35 Response of link 1 at the instant of parameter variation

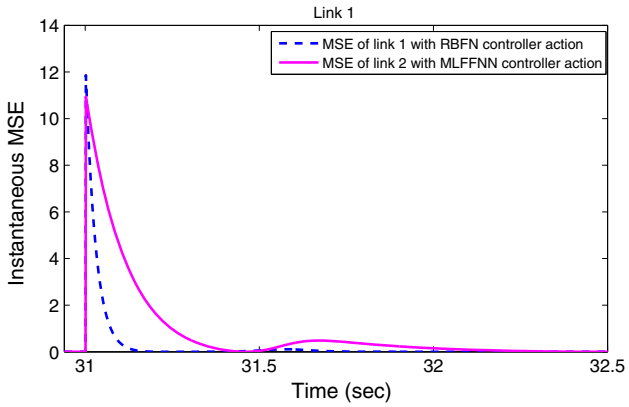

Fig. 36 MSE plot of link 1 with parameter variation

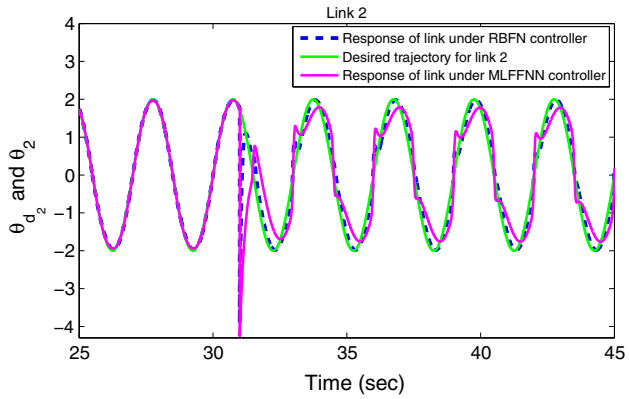

Fig. 37 Response of link 2 at the instant of parameter variation

can be seen that MSE drops very quickly to zero under RBFN controller action after the impact of parameter variation as compared to MSE recovery obtained with MLFFNN.

Similarly, response of link 2 at the instant of parameter variation is shown in Fig. 37. In this case also, we get faster recovery in link 2 response under RBFN controller action

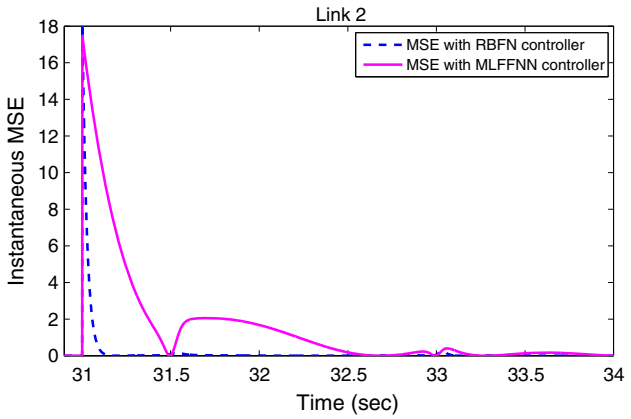

Fig. 38 MSE plot of link 2 with parameter variation

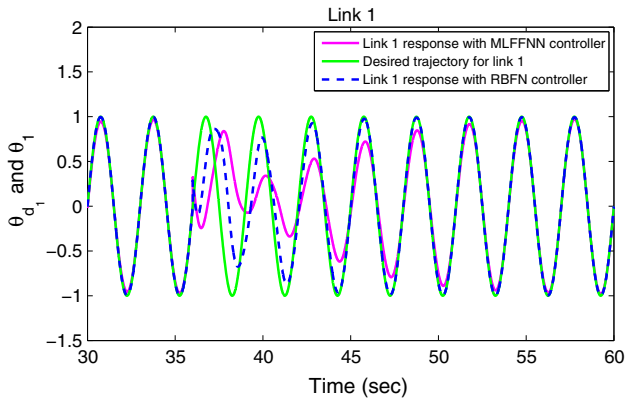

Fig. 39 Response of link 1 at the instant of disturbance signal

as compared to response obtained with MLFFNN controller and corresponding MSE plot is shown in Fig. 38.

From the plot, it can be seen that MSE with RBFN controller action reduces quickly again to zero after the effect of parameter variation as compared to MSE response obtained with MLFFNN controller.

6.2.6 *Testing of performance of RBFN and MLFFNN controllers with disturbance signal*

Now, disturbance signals of magnitude 2.2 N-m are added in both controller outputs at 36th time instant. The corresponding impact on the response of link 1 at the instant of inclusion of disturbance signal is shown in Fig. 39.

From the figure, it can be seen that recovery speed of link 1 response under RBFN controller action is much quicker as compared to recovery speed obtained with MLFFNN controller. This again shows the superior performance of RBFN structure over the MLFFNN structure.

The MSE plot obtained at the instant of disturbance signal is shown in Fig. 40. The MSE plot is also depicting the same conclusion. The rise in MSE due to the disturbance signal reduces again to zero very quickly under

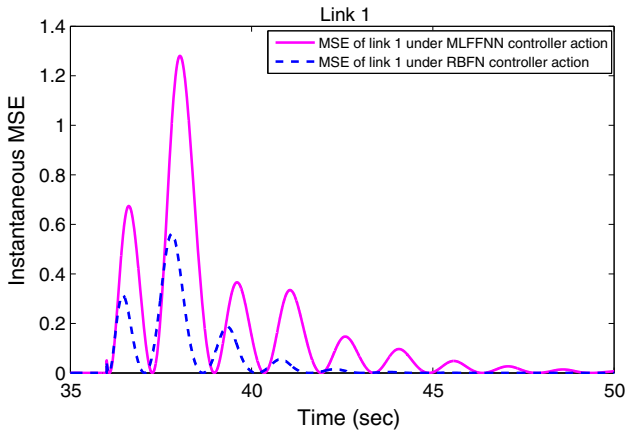

Fig. 40 MSE plot of link 1 with disturbance signal

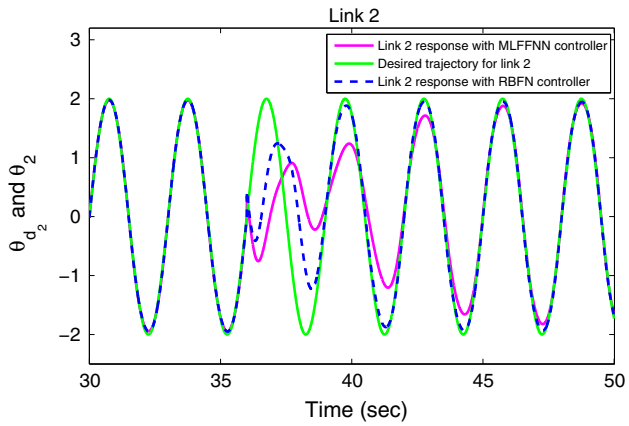

Fig. 41 Response of link 2 at the instant of disturbance signal

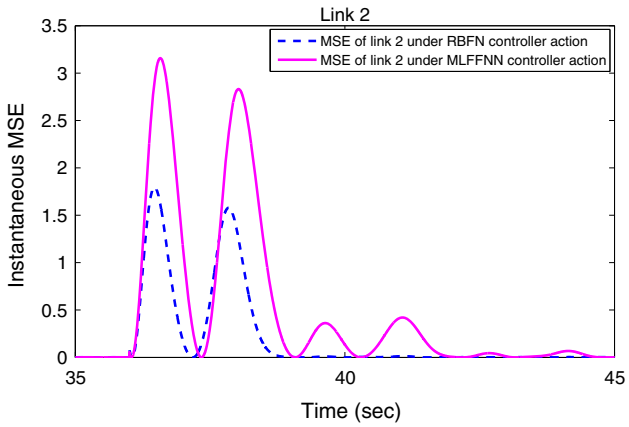

Fig. 42 MSE plot of link 2 with disturbance signal

RBFN controller action as compared to MSE response obtained with MLFFNN controller.

Similarly, link 2 response at the instant of disturbance signal is shown in Fig. 41 and the corresponding MSE plot is shown in Fig. 42. By looking at the link 2 response and its MSE plot under the impact of disturbance signal, we can

see that RBFN controller again performed better than MLFFNN controller.

7 Achievements of the work

1. Simultaneous identification and indirect adaptive control of both one-link and two-link robotic manipulators were implemented successfully by using RBFNs. The effectiveness of its performance as an identification model and as a controller can be seen from the simulation results obtained.
2. The proposed method uses RBFN which has a simpler structure as compared to MLFFNN. The simpler structure of RBFN was able to handle the nonlinearity and complexity of the plant and was able to capture its dynamics.
3. The effectiveness of RBFN in dealing with parameter variation was proved to be better than that of the MLFFNN.
4. The performance of RBFN over the MLFFNN as an indirect adaptive controller is found to be superior in terms of tracking ability and convergence of parameters to the desired values. The response of robotic manipulator under RBFN controller action was found to be better than under MLFFNN controller action.
5. RBFN has successfully compensated the effects of disturbance signals and parameter variations on the one-link and two-link robotic manipulators, and from the corresponding MSE plots obtained, it can be concluded that RBFN has given superior performance as compared to that obtained with the MLFFNN.

8 Conclusion

Online identification and adaptive control of both one-link and two-link robotic manipulators have been successfully implemented using RBFN. Both plants offer a complex dynamics which need to be controlled. Self-adaptive quality of RBFN can be seen when its parameters undergo change in order to compensate the impact of plant uncertainties. Performance of RBFN is also compared with MLFFNN, and after the thorough analysis done in this paper, it can be concluded that RBFN came out to be the winner. The proposed control and identification scheme based on RBFN is suitable for controlling any type of nonlinear system. It allows the plant to be controlled online. Even if the dynamics of given plant is not known, still the scheme will be able to control the plant (this is due to the presence of identification model in parallel to the plant which approximates the unknown dynamics of the plant during the online control).

Funding This study is not funded by any agency.

Compliance with ethical standards

Conflict of interest Rajesh Kumar declares that he has no conflict of interest. Smriti Srivastava declares that she has no conflict of interest. J.R.P. Gupta declares that he has 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. Ayala HVH, dos Santos Coelho L (2012) Tuning of pid controller based on a multiobjective genetic algorithm applied to a robotic manipulator. Expert Syst Appl 39(10):8968–8974
2. Behera L, Kar I (2010) Intelligent systems and control principles and applications. Oxford University Press, Oxford
3. Chen L, Wu R, He Y, Chai Y (2015) Adaptive sliding-mode control for fractional-order uncertain linear systems with nonlinear disturbances. Nonlinear Dyn 80(1–2):51–58
4. Choi YH, Yoo SJ (2016) Single-approximation-based adaptive control of a class of nonlinear time-delay systems. Neural Comput Appl 27(4):1041–1052
5. Corradini ML, Giantomassi A, Ippoliti G, Longhi S, Orlando G (2015) Robust control of robot arms via quasi sliding modes and neural networks. In: Advances and applications in sliding mode control systems. Springer, pp 79–105
6. Gao H, Liu J, Li Y, Hong K, Zhang Y (2015) Dual-layer fuzzy control architecture for the cas rover arm. Int J Control Autom Syst 13(5):1262–1271
7. Gundogdu O, Egrioglu E, Aladag CH, Yolcu U (2016) Multiplicative neuron model artificial neural network based on gaussian activation function. Neural Comput Appl 27(4):927–935
8. Haykin S, Network N (2004) A comprehensive foundation. Neural Netw 2:2004
9. He W, Ge SS, Li Y, Chew E, Ng YS (2015) Neural network control of a rehabilitation robot by state and output feedback. J Intell Robot Syst 80(1):15–31
10. Huang SJ, Lee JS (2000) A stable self-organizing fuzzy controller for robotic motion control. IEEE Trans Ind Electron 47(2):421–428
11. Kondaiah V, Rao JS, Rao VS (2016) Parameter estimation of solid cylindrical electromagnetic actuator using radial basis function neural networks. In: Microelectronics, electromagnetics and telecommunications. Springer, pp 349–363
12. Lazzaretti AE, Tax DM (2015) An adaptive radial basis function kernel for support vector data description. In: Similarity-based pattern recognition. Springer, pp 103–116
13. Li Y, Ho YK, Chua CS (2000) Model-based pid control of constrained robot in a dynamic environment with uncertainty. In: Proceedings of the 2000 IEEE international conference on control applications, 2000. IEEE, pp 74–79
14. Lochan K, Roy B (2015) Control of two-link 2-dof robot manipulator using fuzzy logic techniques: a review. In: Proceedings of fourth international conference on soft computing for problem solving. Springer, pp 499–511
15. Mitra S, Basak J (2001) Frbf: a fuzzy radial basis function network. Neural Comput Appl 10(3):244–252
16. Narendra KS, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Netw 1(1):4–27
17. Paul RP (1981) Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators. Richard Paul

18. Sanxiu W, Shengtao J (2011) Adaptive friction compensation of robot manipulator. In: Electronics and signal processing. Springer, pp 127–134
19. Sciavicco L, Siciliano B (2012) Modelling and control of robot manipulators. Springer, New York
20. Singh M, Srivastava S, Gupta J, Handmandlu M (2007) Identification and control of a nonlinear system using neural networks by extracting the system dynamics. IETE J Res 53(1):43–50
21. Srivastava S, Singh M, Hanmandlu M (2002) Control and identification of non-linear systems affected by noise using wavelet network. In: Computational intelligence and applications. Dynamic Publishers, pp 51–56
22. 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
23. Srivastava S, Singh M, Madasu VK, Hanmandlu M (2008) Choquet fuzzy integral based modeling of nonlinear system. Appl Soft Comput 8(2):839–848
24. Stephanopoulos G (1984) Chemical process control: an introduction to theory and practice
25. Sun D, Mills JK (1999) High-accuracy trajectory tracking of industrial robot manipulator using adaptive-learning scheme. In: American control conference, 1999. Proceedings of the 1999, vol 3, IEEE, pp 1935–1939
26. Tran MD, Kang HJ (2015) Adaptive fuzzy pid sliding mode controller of uncertain robotic manipulator. In: Intelligent computing theories and methodologies. Springer, pp 92–103
27. Várkonyi-Kóczy A, Tusor B, Bukor J (2016) Data classification based on fuzzy-rbf networks. In: Soft computing applications. Springer, pp 829–840
28. Wu XJ, Jiang GC, Wang XJ, Fang N, Zhao L, Ma YM, Luo SJ (2013) Prediction of reservoir sensitivity using rbf neural network with trainable radial basis function. Neural Comput Appl 22(5):947–953