Neurocomputing 287 (2018) 102–117

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Diagonal recurrent neural network based identification of nonlinear dynamical systems with Lyapunov stability based adaptive learning rates

Rajesh Kumar[a,*], Smriti Srivastava[b], J.R.P. Gupta[b], Amit Mohindru[c]

[a] Department of Instrumentation and Control Engineering, Bharati Vidyapeeth's College of Engineering, A-4, Paschim Vihar, New Delhi 110063, India
[b] Division of Instrumentation and Control Engineering, Netaji Subhas Institute of Technology, Sector 3, Dwarka, New Delhi 110078, India
[c] Department of Electronics and Communication Engineering, Indraprastha Institute of Information Technology, New Delhi 110020, India

A R T I C L E I N F O

Article history:
Received 15 September 2017
Revised 29 November 2017
Accepted 21 January 2018
Available online 6 February 2018

Communicated by H.R. Karimi

Keywords:
Diagonal recurrent neural network
Nonlinear system identification
Radial basis function network
Multi-layer feed forward neural network
Robustness
Lyapunov stability based dynamic learning rate

A B S T R A C T

This paper proposes a diagonal recurrent neural network (DRNN) based identification model for approximating the unknown dynamics of the nonlinear plants. The proposed model offers deeper memory and a simpler structure. Thereafter, we have developed a dynamic back-propagation learning algorithm for tuning the parameters of DRNN. Further, to guarantee the faster convergence and stability of the overall system, dynamic (adaptive) learning rates are developed in the sense of Lyapunov stability method. The proposed scheme is also compared with multi-layer feed forward neural network (MLFFNN) and radial basis function network (RBFN) based identification models. Numerical experiments reveal that DRNN has performed much better in approximating the dynamics of the plant and have also shown more robustness toward system uncertainties.

1. Introduction

Science plays a pivotal role in understanding and predicting the behavior of the universe and the systems within it. Much of this owes to the human tendency to ask questions and seek the answers. This has led to the development of various mathematical models which actually agrees with the observations. No one could deny the importance of nonlinear system identification since most control strategies possess the linearity property only over a limited range of operation [1]. It can be argued that linear identification techniques are simple and straightforward but their applicability in real world system's (which are nonlinear) identification is very limited. The process of identification involves the selection of some suitable class of model for approximating the input–output behavior of the system in a best possible manner. The nonlinear identification methods can operate in a wider region and are not limited by certain limit bounds as compared to linear methods which often work in a restricted range. For nonlinear affine systems in which the non-linearity in the system are known, various methods have been developed based on the feedback linearization [2]. These methods, however, work only if some or complete knowledge regarding the system is available. There is an increasing need to estimate the behavior of the system with partially known dynamics. In fact, in the areas of control and pattern recognition, the system under consideration needs to be understood to some extent. In literature we can find number of parameterized models like Winner–Hammerstein model [3], Voltera series [4] and polynomial identification methods [5]. These methods provide accuracy to some extent but at the increasing cost of complexities. Soft computing is an alternative practical approach to solve such mathematically intractable and computationally complex problems. The two main popular components of the soft computing methodologies are fuzzy systems and neural networks. Neural networks emulate the working of the human brain whereas fuzzy logic systems are based on the operator's qualitative knowledge about the system. These tools when equip with the learning algorithms can approximate the unknown dynamics of any given system [6] and then can be used in the design of a nonlinear controller [7]. Fuzzy based identification models have also been successfully applied in system identification and control problems. For example, in [8], the authors have used Takagi–Sugeno fuzzy affine model with parameter uncertainties to

* Corresponding author. Ph.:+919971252729.
E-mail addresses: rajeshmahindru23@gmail.com, rajesh.kumar@bharatividyapeeth.edu (R. Kumar).

https://doi.org/10.1016/j.neucom.2018.01.073
0925-2312/© 2018 Elsevier B.V. All rights reserved.

represent the nonlinear system. Each of these techniques is proven to be effective when used on their own. However, neural networks offer some advantages over these fuzzy systems. These are:

1. Neural Networks, when used for identification purpose, requires only the input–output data set for approximating the unknown dynamics of the given system. They are self learning tools. On the other hand fuzzy logic based identification systems lack the capability for self-learning, and must be modified by an external agent. This is an approach where precision is not paramount and hence, may give vague results.
2. Another disadvantage of fuzzy based system is that they involve more number of arithmetical operations for calculating the output as compared to neural network based models. Hence, convergence speed of fuzzy based system is comparatively slow.
3. Fuzzy based models require finer tuning and simulation before they are put to operation as compared to neural network based models.
4. Real time operation: Neural network computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
5. Neural networks have been trained to perform complex functions in various fields of applications including pattern recognition, identification, classification, speech, and vision and control systems. They can be trained even if the data set is small. For fuzzy based systems, many researchers have proposed different rules/methods of combining the conjunctive or disjunctive clauses: for example, instead of taking the minimum or the maximum of the membership functions, they take the arithmetic or the geometric mean. These rules are arbitrary, and there are lots of them. It is possible, if we have enough training data, i.e. conditions and class assignments by the experts, to train our system so that it chooses the best rule that fits the way of reasoning of the expert that did the classification.

1.1. Feed forward and recurrent neural networks

A neural network typically consists of many simple computational elements or nodes (known as neurons) arranged in the layers and are operating in parallel. The weights, which define the strength of connection between the nodes, are adapted during the learning procedure to yield a good performance. Neural networks are used as black-box models (since the parameters associated with them have no physical meaning) for identifying the unknown mathematical relationship existing between the input and output of the system. Structurally, neural networks are of two types: feed-forward neural network (FNN) and recurrent neural network (RNN). In FNN the input feeds forward through the network layers to the output and hence, only the forward connections are present between the neurons while in the case of RNN both feed-forward and feedback connections are present which makes them the nonlinear dynamic feedback systems [9]. The two variants of FNN are MLFFNN and RBFN. MLFFNN approximation capabilities have been well investigated by many researchers [10], [11]. Any continuous or discrete function $F : D_F \subset \Re^{n_I} \to \Re^m$ can be approximated uniformly to any degree of accuracy by $\hat{F}$ on D_F, where D_F is a compact subset of $\Re^{n_I}$, provided there are a sufficient number of hidden nodes present in the network and activation function used must be continuous, bounded and non-constant [12]. This result is valid for MLFFNN having only single layer. The alternative to MLFFNN is the RBFN. Earlier, RBFN was used only for strict multi-variable function interpolation and requires as many neurons as the available data points. But this is not practically feasible since data points are generally large in number. In [13], the authors have relaxed this strict interpolation restriction and proposed a neural network architecture in which number of neurons required in RBFN are far less than the data points. Due to so many similarities between RBFN and MLFFNN it is expected that former also shows the same approximation ability as the latter. This has, in fact, been proved [14]. Also, RBFN is structurally more compact (as it contains only 3 layers) and takes very less time to train as compared to MLFFNN. This makes it more computationally efficient than MLFFNN. In the case of nonlinear dynamic systems, the output of the physical system is a function of the past history of inputs and outputs and the problem of approximation gets more complicated when FNN are used for this purpose. This is because they do not contain any memory feature in their structure and hence, are useful for approximating only static type of systems (memory-less transformation). But in spite of these limitations, FNN are being used in identification and control applications with the help of the tapped delay lines (TDLs). If the order (or an upper bound on the order) of the system being modeled is known then all the necessary past inputs and output signals can be explicitly fed to the FNN model with the help of these TDLs. Then, this static neural network becomes the dynamic neural network. The network, then, can learn the memory-less transformation that captures the dependence of the output of the system on the specified past inputs and outputs. But from the practical point of view leaning memory-less transformations in this manner may not lead to versatile dynamical models [15]. Another difficulty is in deducing the order of the plant since most of the plant's dynamics are very complex and are not fully understood. So, now the question arises whether it is possible to include the dynamics directly into the structure so that it can learn the dynamics of the system without requiring any a-priori knowledge regarding the system. This is possible, if the neural network contains the memory property in the form of feedback loops and the learning algorithm which can train these feedback weights. These neural networks with self-feedback loops are called as recurrent neural networks (RNNs). They have been used successfully in speech recognition applications [16]. Further, the recurrent nature of neurons in RNN gives them the ability to model the spatial as well as temporal dependencies present between the input and output sequences [17].

More general forms of recurrent neural networks have been developed using the FNN in the cascade and feedback configurations in a globally recurrent manner (without local feedback). This model is referred to as a nonlinear auto-regressive moving average model with exogenous inputs (NARMAX). When NARMAX is used for approximating the dynamics of the plant, the information regarding the number of delayed input(s) and output(s) should be known in advance. This actually makes the problem difficult as the number of delayed signals required actually depends upon the order of the plant. Further, the network size, computation cost and the complexity of the network increases with an increase in the number of tapped delay lines [9]. Locally recurrent neural networks (LRNNs), owing to their simpler structure, do not suffer from these shortcomings. Generally, LRNN can be classified into two categories: Fully connected recurrent neural networks (FCRNNs) and partially connected recurrent neural networks (PCRNNs). In FCRNN, all layers except the input layer are connected to each other with the trainable weights. This leads to a complex structure which makes the training procedure very difficult. On the other hand, in PCRNNs there is only local feedback connection instead of the global one. This leads to a simpler network structure with a memory feature and hence, the training process will be less complex [18]. The diagonal recurrent neural network structure (DRNN) belongs to the PCRNN category. In our present paper, we have proposed a DRNN based identification model for modeling the nonlinear dynamic systems. The DRNN has only one hidden layer of recurrent nodes in which each node is feeding back its own output into itself and not to any other nodes in the same layer. Since there is no interconnection among the nodes, the DRNN has fewer weights as compared to FCRNN and hence, we can expect less training time.

2. Related work—progress in the field of identification

Hornik et al. [19] and Funahashi [20] have emphasized that as long as there are sufficient number of neurons present in the hidden layer any complex function can be approximated (realized) by the neural networks. After that Narendra and Parthasarathy [11] have demonstrated how the neural networks can be used in identification and control of nonlinear systems. They have used back propagation algorithm [21] for the training. Liu et al. [22] performed online identification of nonlinear dynamical systems using Volterra polynomial basis function neural networks. Efe and Kaynak [23] have performed a comparative analysis between different neural network structures in respect of their identification capabilities. In [24], the authors have used fuzzy based recurrent neural network for identifying the nonlinear system and then use it to provide an indirect adaptive control. Gabrijel and Dobnik [25] used recurrent neural networks for identification of unknown dynamics of system. In [26], the authors have used Kalman filter based method to train the recurrent neural network. But Kalman filter based learning algorithm is quite complex and is very sensitive to the nature of noises which may enter in the system. In [27], the authors have used genetic algorithm (GA) to optimize the structure of recurrent neural network having multiple hidden layers. Further, they have applied back-propagation algorithm to optimize the network parameters. In [28], the author have used recurrent neural network for system identification. The neurons are arranged in the three layers for obtaining the temporal relations. In [29], the authors have developed a continuous time recurrent neural network for predicting the dynamic behavior of nonlinear processes. For training the network parameters they have used automatic differentiation techniques. In [30], the authors have used recurrent neural network for identifying the nonlinear system and used extended Kalman filtering and particle filtering approaches for tuning the network parameters. In [31], the authors have used the recurrent neural network as an identifier in which activation functions for hidden neurons is the combination of sigmoid and wavelet functions and then used it for forecasting the dynamics of the plant. In [32], the authors have used a special type of neural network known as Hopfield neural network and used it to estimate the time varying parameters of the dynamical system. In [33], the authors have combined internal and external type recurrent neural networks for identifying the dynamics of a highly maneuverable fighter aircraft.

2.1. Importance of the work

1. The developed DRNN based identification model can be used in designing the nonlinear controller for the plant.
2. Most systems are nonlinear with unknown complex dynamics. The DRNN based identification model can be used for approximating these dynamics in both online and epoch mode.
3. The proposed DRNN based identification model offers a simpler structure and requires lesser number of parameters to be tuned.
4. The performance comparison of DRNN with the state of art methods like MLFFNN and RBFN has been shown in detail.
5. The responses of the identification models during all stages of the training are shown in detail. These responses show how these neural network based models are performing.
6. Novel adaptive learning rate is developed using Lyapunov stability method. Its advantages are two-folds. First, it ensures the overall stability of the system and second, it improves the performance of back-propagation algorithm.

2.2. Objectives/contributions of the paper

1. To develop a recurrent neural network based identification model which requires less number of inputs and hidden neurons for learning the dynamics of the unknown nonlinear system.
2. To test and compare the performance of DRNN based identification model with that of MLFFNN and RBFN identification models.
3. To compare the structural complexity of DRNN, RBFN and MLFFNN based identification models.
4. To describe in detail the computational complexities involved in the identification models.
5. To show in detail, through simulations, how the identification models learns the unknown dynamics of the nonlinear system.
6. To develop and show the effectiveness of the adaptive learning rate on the learning of the identification models.
7. To test and compare the robustness of the identification models.

The main outline of the paper is as follows: After Section 1 we come to Section 3 which involves the mathematical formulation of DRNN, RBFN and MLFFNN models. In Section 4 the structural complexity of these models is compared. In Section 5 the computational complexity of these models is compared. Section 6 involves the brief overview of the nonlinear system identification and development of DRNN based identification model. In Section 7 the necessary weight adjustment equations are developed using dynamic back-propagation algorithm. Section 8 involves the development of novel adaptive learning rate based on discrete Lyapunov stability method. In Section 9 simulation experiments are performed by considering 4 numerical examples. The performances of all three identifiers are tested and compared. Further, robustness analysis is also done in these examples. Section 9 highlights some of the limitations of the proposed method. In Section 10 conclusion and remark on future scope is given.

3. Theoretical foundations of DRNN, MLFFNN and RBFN models

This section involves the brief overview of these intelligent models along with their detailed mathematical formulation.

3.1. DRNN model

The structure of DRNN is shown in Fig. 1. The red arrows, shown in the figure of DRNN, denote the diagonal weights which transmits the unit delay output of the recurrent neuron as an input to the same recurrent neuron. These diagonal weights are represented by diagonal weight vector, $W_D = \left\{w_D^1, w_D^2 ... w_D^q\right\}$ and W_I denotes the input weight vector having adjustable weights. Further, zeroing of vector W^D reduces DRNN to MLFFNN. The connection between recurrent and output layer neurons is given by output weight vector, $W_O = \left\{w_O^1, w_O^2 .... w_O^q\right\}$. Also, the p-input vector denotes externally applied signal and is denoted by vector X= $\{x_1(k), x_2(k) .... x_p(k)\}$. Further, the output of any qth recurrent neuron at any kth time instant is given by:

$$T_q(k) = f[Z_q(k)] \tag{1}$$

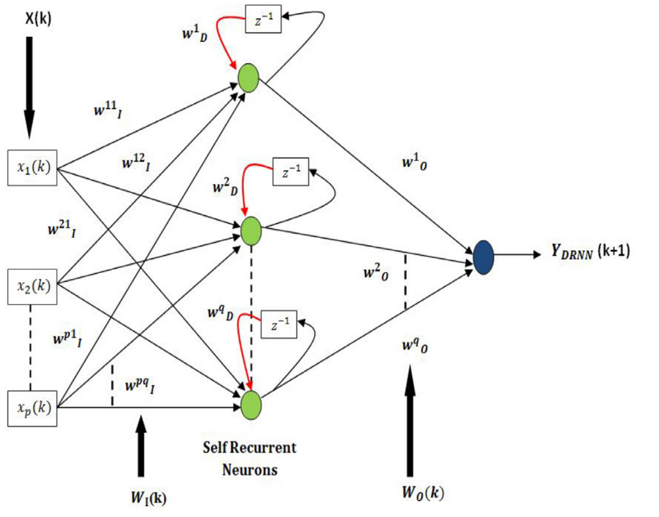

Fig. 1. DRNN structure. (For interpretation of the references to colour in the text, the reader is referred to the web version of this article.)

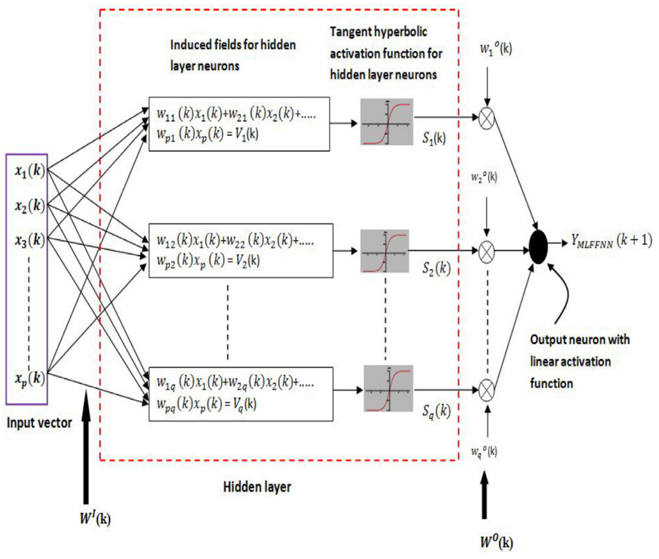
Induced fields for hidden layer neurons
Tangent hyperbolic activation function for hidden layer neurons
$w_{11}(k)x_1(k)+w_{21}(k)x_2(k)+.....w_{p1}(k)x_p(k) = V_1(k)$
$w_{12}(k)x_1(k)+w_{22}(k)x_2(k)+.....w_{p2}(k)x_p(k) = V_2(k)$
$w_{1q}(k)x_1(k)+w_{2q}(k)x_2(k)+.....w_{pq}(k)x_p(k) = V_q(k)$
$S_1(k)$
$S_2(k)$
$S_q(k)$
$w_1{}^o(k)$
$w_2{}^o(k)$
$w_q{}^o(k)$
$Y_{MLFFNN}(k+1)$
Output neuron with linear activation function
$x_1(k)$
$x_2(k)$
$x_3(k)$
$x_p(k)$
Input vector
Hidden layer
W^I(k)
W^O(k)

Fig. 2. MLFFNN structure.

Note that f is a nonlinear tangent hyperbolic function. The induced field of any qth recurrent neuron is computed as follows:

$$Z_q(k) = W_D^q(k)T_q(k-1) + \sum_p W_I^{pq}(k)x_p(k) \tag{2}$$

Next, the induced field of output layer neuron is given by:

$$V(k) = \sum_q W_O^q T_q(k) \tag{3}$$

The output of the output layer neuron is equal to its own induced field since linear function is considered as its activation function. So, we can write:

$$Y_{DRNN}(k) = V(k) = \sum_q W_O^q T_q(k) \tag{4}$$

3.2. MLFFNN model

In Fig. 2, the dashed line shows an expanded view of the hidden layer neurons in MLFFNN. Further, the q dimensional vectors $V(k) = \{V_1(k), V_2(k), ...V_q(k)\}$ and $S(k) = \{S_1(k), S_2(k), ...S_q(k)\}$ denotes an induced fields and outputs of hidden layer neurons. The

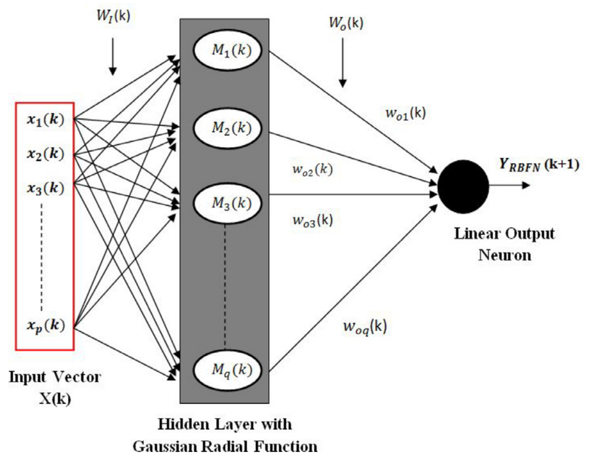

Fig. 3. RBFN structure.

input and output weight vectors are denoted by $W^I(k)$ and $W^O(k)$. The mathematical model of MLFFNN is given by

$$\begin{aligned} Y_{MLFFNN}(k+1) &= \sum_q W_r^O f[V_q(k)] \\ &= \sum_q W_q^O f\left[\sum_p W_{pq}^I(k)x_p(k)\right] \end{aligned} \tag{5}$$

where $I^p = \{x_1(k), x_2(k)...x_p(k)\}$ denotes an externally applied p-input signal vector.

3.3. RBFN model

The structure of RBFN model is shown in Fig. 3. Its input vector at any kth time instant is denoted by $X(k) = \{x_1(k), x_2(k), ...x_p(k)\}$. The input weight vector, having unity value weights (which remain fixed), is denoted by $W_I(k)$. The neurons present in the hidden layer of RBFN are called as radial centers. Each radial center consists of a coordinate having same dimension as that of the input vector, which means p number of elements are there in every radial center. Further, there are q number of radial centers that are denoted by $M_1(k)$, $M_2(k)$....$M_q(k)$. The adjustable output weight vector, $W_o(k)$, consists of q number of adjustable weights, $w_{o1}(k)$, $w_{o2}(k)$, ...$w_{oq}(k)$. The computation of RBFN output is done as:

$$Y_{RBFN}(k) = \sum_{i=1}^{q} \phi_i(k)W_{oi}(k) \tag{6}$$

For any ith radial center, its output at any kth time instant can be found as:

$$\phi_i(k) = \phi\|X(k) - M_i(k)\| \tag{7}$$

where $\|X(k) - M_i(k)\|$ is equal to the Euclidean distance between $X(k)$ and $M_i(k)$. Further, $\phi(.)$ denotes a Gaussian radial basis function having the following definition [34]:

$$\phi(\nu) = \exp\left(\frac{-\nu^2(k)}{2\sigma^2(k)}\right) \tag{8}$$

where $\nu(k) = \|X(k) - M_i(k)\|$ and the widths associated with each radial center are represented by $\sigma(k) = \{\sigma_1(k), \sigma_2(k).....\sigma_q(k)\}$. Further, there exists number of choices for radial basis functions like Gaussian, quadratic, inverse quadratic and thin plate of spline. However, Gaussian function is more intuitive than other available functions since it produces a significant output if input lies in the neighborhood of the radial center and vice versa.

Table 1
Computational complexity of identifier models.

Inputs=2 Hidden neurons= 5	Input to hidden layer operations	Induced fields calculations at hidden neurons	Euclidean distance calculation	Feedback of unit delay output of hidden neurons	Calculation of hidden layer neurons output	Hidden to output layer operations	Induced field calculation at output neuron	Total number of operation in one iteration
DRNN	5+5=10	5	0	5	5	5	1	31
RBFN	0	0	5	0	5	5	1	16
MLFFNN	5+5=10	5	0	0	5	5	1	26

4. Structural comparison of DRNN, MLFFNN and RBFN models

In this section we have formulated the parameter count associated with DRNN, MLFFNN and RBFN models.

Definition 1. Any neural network can be denoted by an ordered tuple $\{N\}^T$ where $\{N\}^T = \{X^p, H^q, O^m\}$ and X,H,O denotes input, hidden layer and output layer respectively. Also, m denotes number of output neurons present in the output layer of MLFFNN, DRNN and RBFN.

Definition 2. Let $\{N\}^{MLFFNN}$, $\{N\}^{DRNN}$, $\{N\}^{RBFN}$ denotes total number of parameters to be tuned present in MLFFNN, DRNN and RBFN (including the external bias signals applied to hidden neurons (or nodes) and output layer neuron(s)) respectively.

Lemma 1. *The total count of parameters to be tuned in all these three identifiers can be find out as:*

$$\{N\}^{MLFFNN} = (p+m+1)q+m \tag{9}$$

$$\{N\}^{DRNN} = (p+m+2)q+m \tag{10}$$

$$\{N\}^{RBFN} = (m+3)q+m \tag{11}$$

In order to understand how Eqs. (9)–(11)are developed we consider one example with the following data: $p = 2, q = 3$ and $m = 2$. So, total number of weights present in each input weight vectors of MLFFNN and DRNN will be $pq = 6$. The total count of weights which connects the hidden layer neurons/nodes to the output layer neurons will be $qm = 6$. Further, the total number of bias signals which are applied to hidden layer neurons and output layer neurons will be $q + m = 5$. In the case of DRNN, the hidden nodes also contain self-feedback loops so the total count of these diagonal (feedback) weights will be q. So, total weight count in MLFFNN model will be $pq + qm + q + m = (p+m+1)q+m = 17$. In DRNN case this count will be $pq + qm + q + m + q = (p+m+2)q+m = 20$. For RBFN model the total number of parameters to be tuned is = number of weights which connects the hidden neurons to the output neurons $(= q * m)$ + number of radial centers $(= q)$ + number of widths $(= q)$ + number of bias signals applied to the hidden neurons $(= q)$ + the number of bias signals applied to output neurons $(= m)$. The total comes out to be $qm + q + q + q + m = (m+3)q+m = 17$.

5. Computational complexity of identifiers

For comparing the computational complexity of DRNN, RBFN and MLFFNN models, let us consider that two input signals $x_1(k)$ and $x_2(k)$ are applied to these models. Let there are 5 neurons in the single hidden layer and one neuron in the output layer in all these identifiers. The various arithmetic operations involved in calculating the identifier's output are shown in Table 1. From the table it can be seen that RBFN involves least number of mathematical operations followed by MLFFNN and DRNN. But as we know DRNN involves memory feature in the form of self-feedback loops so we can expect it to approximate the unknown function by utilizing the lesser number of neurons and external inputs. This will substantially reduce the number of arithmetic operations required in DRNN and hence, make it computationally more efficient.

6. Nonlinear system identification

The basic aspect of any control scheme is to provide the desired control. For any such scheme to work properly the knowledge regarding the dynamics of plant (to be controlled) should be known. But many of times these plant's dynamics are not fully understood. Neural networks are known to be good approximators. They are inherently nonlinear and with the help of learning algorithms they can learn the unknown dynamics of the given plant. The most general mathematical model for representing any dynamical system is given by

$$Y_p(k+1) = F[Y_p(k), Y_p(k-1)...Y_p(k-n+1), r(k), r(k-1)...r(k-m+1)] \tag{12}$$

Here, $Y_p(k+1)$ denotes the plant output which depends upon present and past values of input, $r(k-m+1)$, as well as on its own present and past values, $Y_p(k-n+1)$. The term n represents the order of the plant and m is generally taken to be less than n. F is an unknown nonlinear function which is to be identified. The MLFNN and RBFN based identifiers have similar mathematical structures and are given by

$$Y_{MLFFNN}(k+1) = \hat{F}[Y_p(k), Y_p(k-1)...Y_p(k-n+1), r(k), r(k-1)...r(k-m+1)] \tag{13}$$

and

$$Y_{RBFN}(k+1) = \hat{F}[Y_p(k), Y_p(k-1)...Y_p(k-n+1), r(k), r(k-1)...r(k-m+1)] \tag{14}$$

The objective is to have $\hat{F} \approx F$. Now, from Eqs. (13) and (14) it can be seen that the next output of MLFFNN and RBFN identification models depends upon the present and past history of input and output. Such identification structure is known as series-parallel type and is shown in Fig. 4. For DRNN case we have proposed an identification structure as shown in Fig. 5. From the figure it can be seen that DRNN only requires plant's present value, $Y_p(k)$ and one past value, $Y_p(k-1)$ along with present value of externally applied input, $r(k)$ in order to compute its next value, $Y_{DRNN}(k+1)$. This input choice of input signals reduces the number of parameters to be tuned in case of DRNN identifier. The proposed DRNN identifier structure is also of series-parallel type and has the following dynamic equation:

$$Y_{DRNN}(k+1) = \hat{F}[Y_p(k), Y_p(k-1), r(k)] \tag{15}$$

7. Learning algorithm for DRNN identification model

To provide an incremental type of learning, we have used dynamic back-propagation method which is based on gradient descent principle. The objective is to develop a DRNN based identification model which on getting trained able to track the plant's output. So, the purpose is to reduce the mean square error (MSE) which is defined as:

$$E_i(k) = \frac{1}{2}[Y_p(k) - Y_{DRNN}(k)]^2 = \frac{1}{2}e_i^2(k) \tag{16}$$

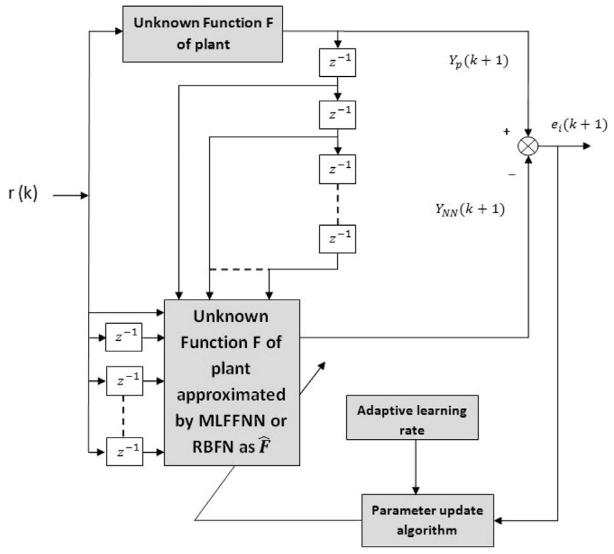
Unknown Function F of plant
z^{-1}
$Y_p(k+1)$
$e_i(k+1)$
+
−
r (k)
$Y_{NN}(k+1)$
Unknown Function F of plant approximated by MLFFNN or RBFN as $\widehat{F}$
Adaptive learning rate
Parameter update algorithm

Fig. 4. Series-parallel identification model structure for MLFFNN and RBFN identifiers.

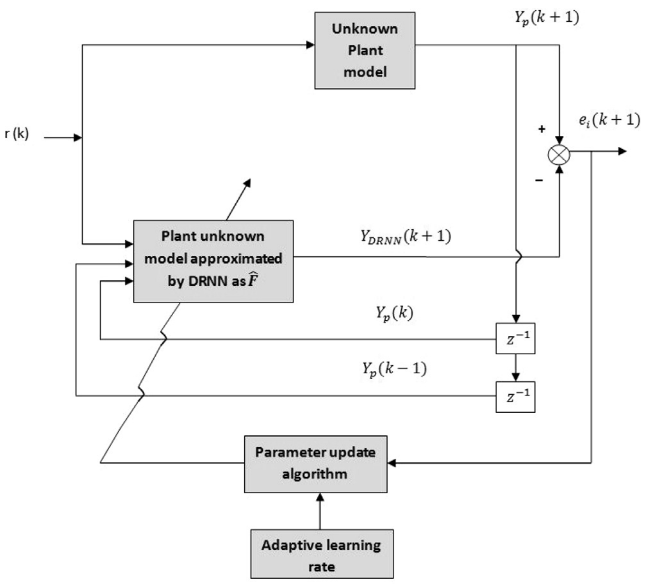

Fig. 5. Series-parallel identification model structure for DRNN (Proposed).

where $E_i(k)$ denotes the performance function for DRNN identification model and $e_i(k)$ denotes an identification error. In DRNN based identification model, self-recurrent feedback weights (diagonal weights) and output layer weights are to be tuned. Their recursive update equations are derived in the following subsections.

7.1. Adjustment of output weight vector

The gradient of identification error with respect to any jth output layer weight is given by

$$\frac{\partial E_i(k)}{\partial W_O^j(k)} = \left(\frac{\partial E_i(k)}{\partial Y_{DRNN}(k)} \times \frac{\partial Y_{DRNN}(k)}{\partial W_O^j(k)} \right) \tag{17}$$

where $\frac{\partial Y_{DRNN}(k)}{\partial W_O^j(k)} = T_j(k)$. Thus, each element in $W_O(k) = \{w_O^1(k), w_O^2(k), \ldots. w_O^q(k)\}$ will be updated as

$$W_O^j(k+1) = W_O^j(k) - \Delta W_O^j(k) \tag{18}$$

where $\Delta W_O^j(k) = -\eta_i e_i(k) T_j(k)$.

7.1.1. Adjustment of diagonal (feedback) weight vector

For any jth diagonal weight vector its update equation is obtained as follows:

$$\frac{\partial E_i(k)}{\partial W_D^j(k)} = \left(\frac{\partial E_i(k)}{\partial Y_{DRNN}(k)} \times \frac{\partial Y_{DRNN}(k)}{\partial W_D^j(k)} \right) \tag{19}$$

where $\frac{\partial Y_{DRNN}(k)}{\partial W_D^j(k)} = \frac{\partial Y_{DRNN}(k)}{\partial V(k)} \frac{\partial V(k)}{\partial W_D^j(k)}$.

Now $\frac{\partial Y_{DRNN}(k)}{\partial V(k)} = 1$. Evaluation of $\frac{\partial V(k)}{\partial W_D^j(k)}$ is as follows:

$$\frac{\partial V(k)}{\partial W_D^j(k)} = \frac{\partial V(k)}{\partial T_j(k)} \frac{\partial T_j(k)}{\partial W_D^j(k)} \tag{20}$$

where

$$\frac{\partial V(k)}{\partial T_j(k)} = W_O^j(k) \tag{21}$$

and

$$\frac{\partial T_j(k)}{\partial W_D^j(k)} = \left(1 - T_j^2(k)\right)\left[T_j(k-1) + W_D^j(k) L_j(k-1)\right] \tag{22}$$

where $L_j(k) = \frac{\partial T_j(k)}{\partial W_D^j(k)}$ and $L_j(0) = 0$. So, each element of diagonal weight vector will be updated as:

$$W_D^j(k+1) = W_D^j(k) - \Delta W_D^j(k) \tag{23}$$

where

$$\Delta W_D^j(k) = -\eta_i e_i(k) W_O^j(k)\left[1 - T_j^2(k)\right] \times \left[T_j(k-1) + W_D^j(k) L_j(k-1)\right]. \tag{24}$$

7.1.2. Adjustment of input weight vector

Following the same approach, the adjustment equation for any jth element of input weight vector will be:

$$W_I^j(k+1) = W_I^j(k) - \Delta W_I^j(k) \tag{25}$$

where

$$\Delta W_I^j(k) = -\eta_i e_i(k) W_O^j(k)\left[1 - T_j^2(k)\right] \times \left[x_j(k) + W_D^j Q_j(k-1)\right]. \tag{26}$$

where $Q_j(k) = \frac{\partial T_j(k)}{\partial W_I^j(k)}$ and $Q_j(0) = 0$

8. Derivation of adaptive learning rates using Lyapunov stability method

The learning rate, η_i, in the update equations of DRNN parameters plays a crucial role in determining the convergence and the stability of the system. If its value is set closer to zero then the convergence is guaranteed but the training time will be more. On the other hand if its value is set closer to 1 then the learning speed will be fast but system may become unstable. So, this requires a judicious selection of the learning rate value. In this section, Lyapunov stability method is used to develop an adaptive learning rate. At each iteration the learning rate value will be adjusted so as to optimize the speed of learning. The procedure is as follows:

Let the positive definite Lyapunov function is chosen as:

$$V_l(k) = \frac{1}{2} e_i^2(k) \tag{27}$$

Now, $V_l(k)$ is non zero and positive as long as $e_i(k)$ is non zero. For the discrete time system, stability is guaranteed if

$$\Delta V_l(k) = V_l(k+1) - V_l(k) \le 0 \tag{28}$$

Eq. (28) can be written as

$$\Delta V_l(k) = V_l(k+1) - V_l(k) = \frac{1}{2}\left\{e_i^2(k+1) - e_i^2(k)\right\} \tag{29}$$

or

$$\Delta V_I(k) = \frac{1}{2}[e_i(k+1) + e_i(k)][e_i(k+1) - e_i(k)] \tag{30}$$

Now, $\Delta e_i(k) = e_i(k+1) - e_i(k)$ so Eq. (30) can be written as

$$\Delta V_I(k) = \frac{1}{2}[\Delta e_i(k) + 2e_i(k)][\Delta e_i(k)] \tag{31}$$

or

$$\Delta V_I(k) = \Delta e_i(k)\left[\frac{1}{2}[\Delta e_i(k)] + e_i(k)\right] \tag{32}$$

In order to obtain an optimal learning rate, we can write $e_i(k+1)$ in the linear form using the Taylor series expansion as

$$e_i(k+1) = e_i(k) + \frac{\partial e_i(k)}{\partial P(k)}\Delta P(k) + hot \tag{33}$$

Here $P(k)$ denotes any parameter in DRNN identifier to be adjusted and hot denotes higher order terms which can be neglected. Let us consider any weight vector $W(k)$ as parameter $P(k)$. Then Eq. (33) can be written as

$$e_i(k+1) - e_i(k) = \Delta e_i(k) = \frac{\partial e_i(k)}{\partial W(k)}\Delta W(k) \tag{34}$$

Now we know that for DRNN based identifier, $\Delta W(k)$ is given by

$$\Delta W(k) = -\eta_i e_i(k)\frac{\partial e_i(k)}{\partial W(k)} \tag{35}$$

where $e_i(k) = Y_p(k) - Y_{DRNN}(k)$ so Eq. (35) can be written as

$$\Delta W(k) = \eta_i e_i(k)\frac{\partial Y_{DRNN}(k)}{\partial W(k)} \tag{36}$$

Putting Eq. (34) in Eq. (32) we get

$$\Delta V_I(k) = \left[\frac{\partial e_i(k)}{\partial W(k)}\Delta W(k)\right]\left[\frac{1}{2}\left[\frac{\partial e_i(k)}{\partial W(k)}\Delta W(k)\right] + e_i(k)\right] \tag{37}$$

Now putting $\frac{\partial e_i(k)}{\partial W(k)} = \frac{\partial (Y_p(k) - Y_{DRNN}(k))}{\partial W(k))} = -\frac{\partial Y_{DRNN}(k)}{\partial W(k)}$ and Eq. (36) in Eq. (37) we get

$$\Delta V_I(k) = \frac{1}{2}\left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^4 \eta_i^2 e_i^2(k) - \left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^2 \eta_i e_i^2(k) \tag{38}$$

or

$$\Delta V_I(k) = -\eta_i e_i^2(k)\left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^2 \times\left\{1 - \frac{\eta_i}{2}\left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^2\right\} \tag{39}$$

Now for stability $\Delta V \le 0$ so

$$-\eta_i e_i^2(k)\left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^2\left\{1 - \frac{\eta_i}{2}\left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^2\right\} \le 0 \tag{40}$$

Multiplying both sides by minus sign we get

$$\eta_i e_i^2(k)\left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^2\left\{1 - \frac{\eta_i}{2}\left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^2\right\} \ge 0 \tag{41}$$

Now for Eq. (41) to hold true we must analyze the conditions on various terms in Eq. (41) which must be satisfied. In the term $\eta_i e_i^2(k)[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}]^2$, both $e_i^2(k)$ and $[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}]^2$ will either be positive or zero so it means η_i must have to be ≥ 0 for Eq. (41) to hold true. So, we have:

$$\eta_i \ge 0 \tag{42}$$

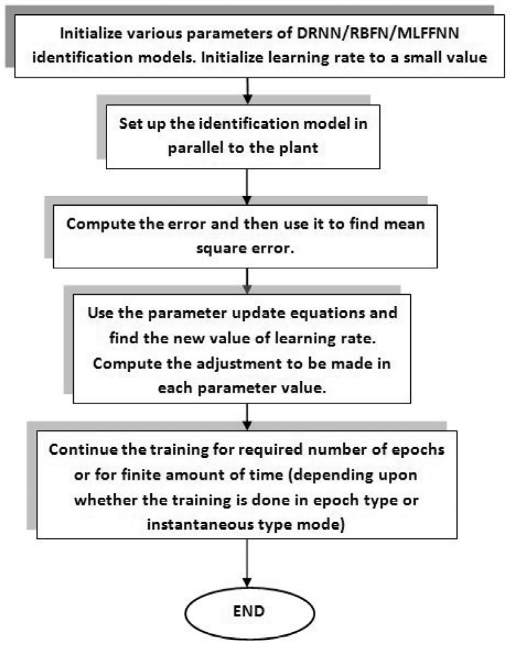
Initialize various parameters of DRNN/RBFN/MLFFNN identification models. Initialize learning rate to a small value
Set up the identification model in parallel to the plant
Compute the error and then use it to find mean square error.
Use the parameter update equations and find the new value of learning rate. Compute the adjustment to be made in each parameter value.
Continue the training for required number of epochs or for finite amount of time (depending upon whether the training is done in epoch type or instantaneous type mode)
END

Fig. 6. Steps of neural network based identifier training.

Since η_i cannot be set to zero as in that case there will not be any learning so the first condition on the value of η_i is:

$$\eta_i > 0 \tag{43}$$

If Eq. (43) hold true then $\Delta V(k) \le 0$ only if the remaining term present in Eq. (41) also satisfies below condition which is given by:

$$\left\{1 - \frac{\eta_i}{2}\left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^2\right\} \ge 0 \tag{44}$$

or

$$\left\{2 - \eta_i\left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^2\right\} \ge 0 \tag{45}$$

Now $[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}]^2$ will always be positive or zero. Eq. (45) will be equal to zero if η_i is set equal to $\frac{2}{[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}]^2}$. So, Eq. (41) holds true if

$$\eta_i \le \frac{2}{\left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^2} \tag{46}$$

Thus, from the Eqs. (43) and (46) we have the lower and upper limit setting on the value of η_i which leads to the following condition on η_i that must be satisfied to ensure stability of the system

$$0 < \eta_i \le \frac{2}{\left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^2} \tag{47}$$

Now by using Eq. (47) we can make the learning rate adaptive as:

$$\eta_i(k) = \frac{2}{\left[\frac{\partial Y_{DRNN}(k)}{\partial W(k)}\right]^2} \tag{48}$$

Following the similar approach we can derive conditions on the $\eta_i(k)$ values corresponding to other parameters associated with identifiers. Fig. 6 shows the flowchart which highlights the various steps involved in the identification process.

9. Simulation study

Now the performance of our proposed DRNN based identification model is tested and compared with RBFN and MLFFNN based identification models. For carrying out the thorough analysis we have considered 4 dynamic plants of different complexity. In one of the example we have considered both epoch and online mode of training whereas in the rest of the three examples only online mode of training is performed. In all the 4 examples, we have considered only single hidden layer in all the identifiers. Further, we have considered only 10 recurrent neurons in our proposed DRNN based identification model whereas in RBFN and MLFFNN we have considered 20 neurons each in their hidden layers. Besides learning the unknown dynamics of the plant these identifiers were also put to test against plants uncertainties like parameter variations and disturbance signals.

9.1. Example 1 Online mode of identification

Consider a nonlinear dynamical system whose dynamics are given as in [11]

$$Y_p(k+1) = F[x_1, x_2, x_3, x_4, x_5] \tag{49}$$

where $x_1 = Y_p(k), x_2 = Y_p(k-1), x_3 = Y_p(k-2), x_4 = r(k), x_5 = r(k-1)$ and the unknown function F is

$$F[x_1, x_2, x_3, x_4, x_5] = \frac{\alpha x_1 x_2}{1 + x_1^2 + x_2^2 + x_3^2} + x_4 + 0.8x_5 \tag{50}$$

where value of parameter $\alpha = 5$. The identification structure of MLFFNN, RBFN and DRNN is given by Eqs. (51) and (52) and respectively.

$$Y_{MLFFNN}(k+1) = \widehat{F}[x_1, x_2, x_3, x_4, x_5] \tag{51}$$

$$Y_{RBFN}(k+1) = \widehat{F}[x_1, x_2, x_3, x_4, x_5] \tag{52}$$

$$Y_{DRNN}(k+1) = \widehat{F}[x_1, x_2, x_4] \tag{53}$$

Online mode of training is adopted to update the parameters of the identifiers in a recursive manner (sample by sample). Initial value of adaptive learning rate, at the start of the training, is set equal to $\eta_i(0) = 0.004$. This same value is also considered for the fixed learning rate. The externally applied input $r(k)$ is a bounded signal having its amplitude lying in the interval of $[-1, 1]$ and is equal to $r(k) = sin(\frac{2\pi k}{25})$.

9.1.1. Discussion on the performance of identifiers from the obtained simulation results [Example 1]

The response of DRNN, RBFN and MLFFNN identification models with fixed as well as with dynamic learning rate during the various stages of online training are shown from Figs. 7–15. It can be seen from the figures that performances of these identifiers in learning the unknown dynamics of the plant are better with adaptive learning rate as compared to fixed one. Further, DRNN can be seen performing better followed by RBFN and MLFFNN identifiers. The time axis in these figures reveals that MLFFNN took maximum time to learn the dynamics of the plant as compared to DRNN and MLFFNN. The various details associated with this experiment are shown in Table 2. From the table it can be seen that DRNN offers least number of parameters to be tuned as compared to RBFN and MLFFNN. Also, the average MSE value obtained with DRNN identifier is least among the other identifiers. This shows the superior performance of DRNN over RBFN and MLFFNN identifiers.

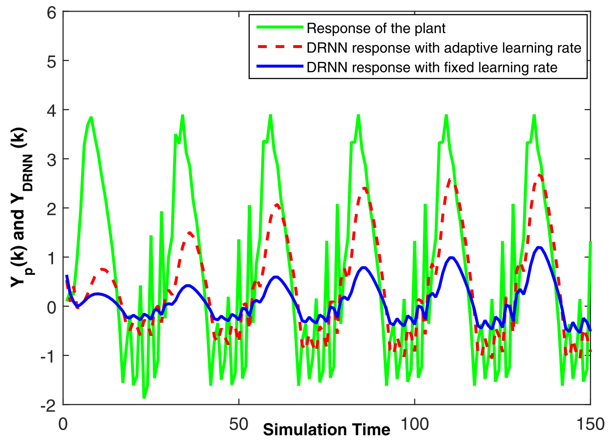

Fig. 7. DRNN identifier response at the initial stage of training [Example 1].

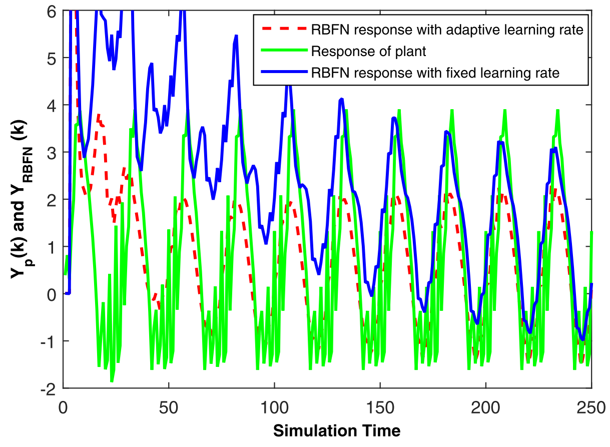

Fig. 8. RBFN identifier response at the initial stage of training [Example 1].

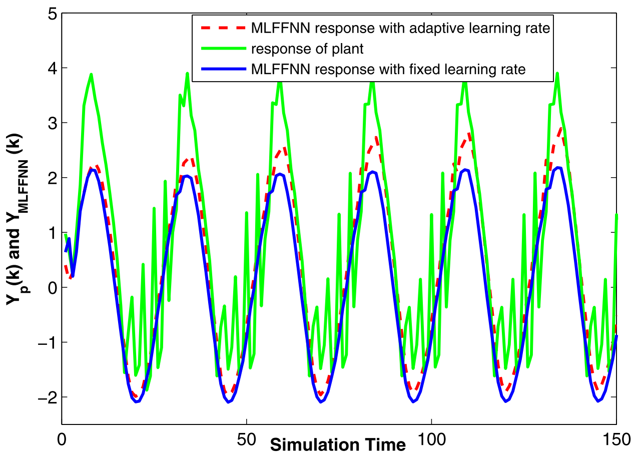

Fig. 9. MLFFNN identifier response at the initial stage of training [Example 1].

9.1.2. Comparison of performance of proposed identification model with the existing literature [Example 1]

Further, the neural network used in [11] for identification purpose consists of 2 hidden layers with 20 neurons each and took 50,000 time instants to learn the dynamics of the plant whereas in our paper the DRNN based identification model with adaptive learning rate learns the dynamics of the plant in a shorter period of time (around 11,000 sampling instants, see dashed red curve in Fig. 10–12 and the corresponding time axis) and consists of only one hidden layer with only 10 neurons.

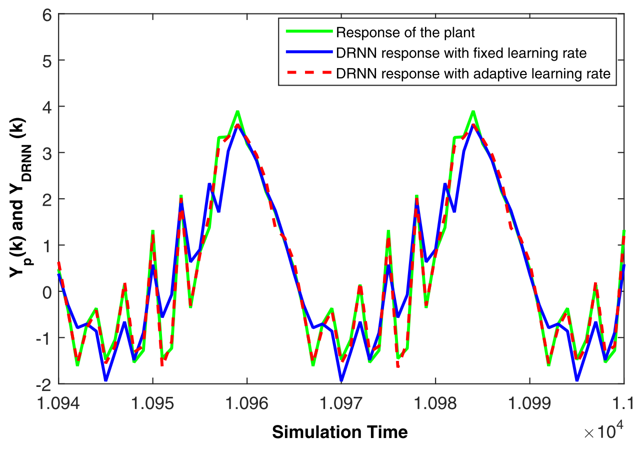

Fig. 10. DRNN identifier response after certain amount of training [Example 1]. (For interpretation of the references to colour in the text, the reader is referred to the web version of this article.)

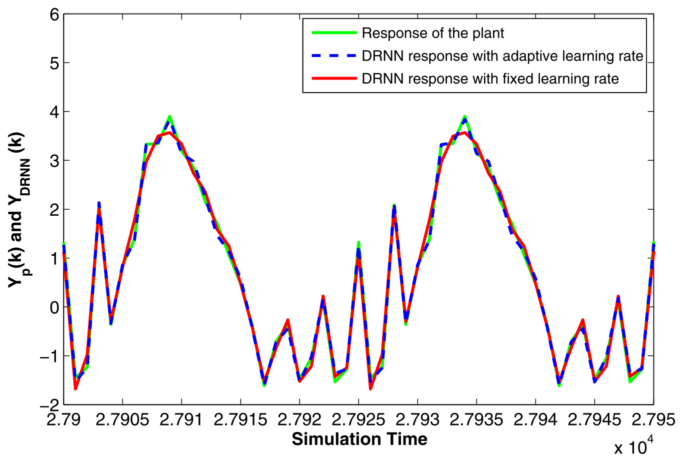

Fig. 13. DRNN identifier response at the final stages of training [Example 1].

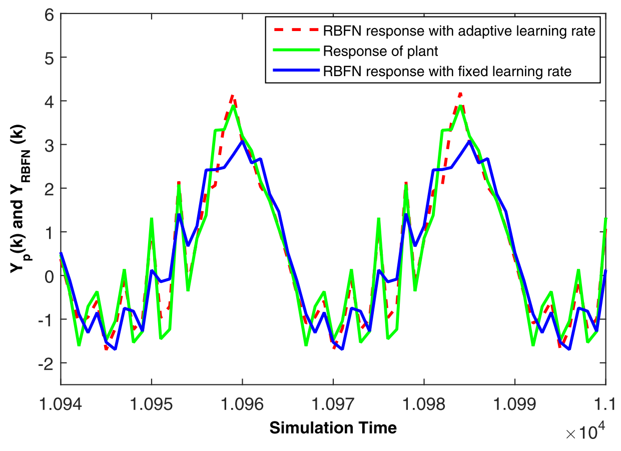

Fig. 11. RBFN identifier response after certain amount of training [Example 1].

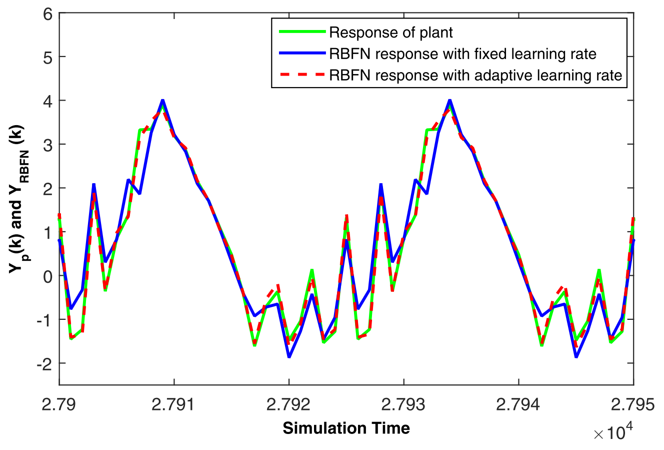

Fig. 14. RBFN identifier response at the final stages of training [Example 1].

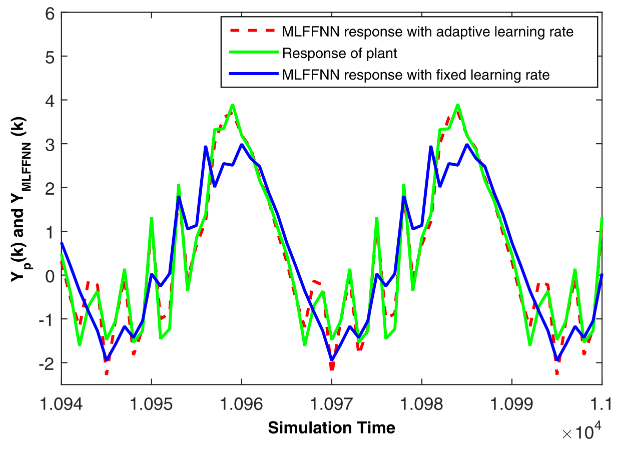

Fig. 12. MLFFNN identifier response after certain amount of training [Example 1].

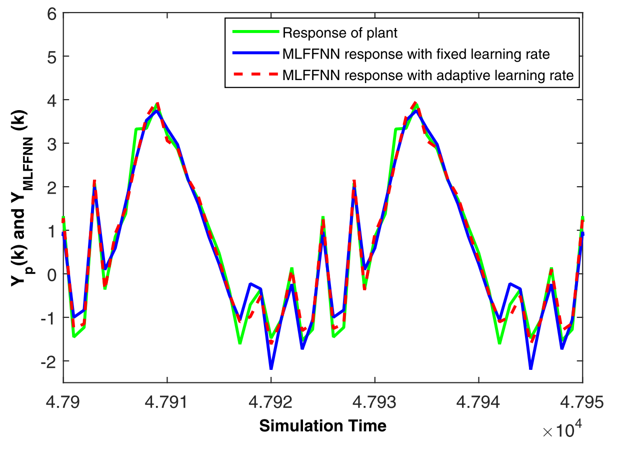

Fig. 15. MLFFNN identifier response at the final stages of training [Example 1].

Table 2
Comparison of identifiers in terms of various parameters [Example 1].

	DRNN based identifier	RBFN identifier	MLFFNN identifier
Total number of inputs	03	05	05
Number of nodes in hidden layer	10	20	20
Total count of parameters to be tuned	61	81	141
Average MSE (with adaptive learning rate)	0.0091	0.0328	0.0513
Average MSE (with fixed learning rate)	0.0618	0.1047	0.1871
Simulation time	28000	28000	48000

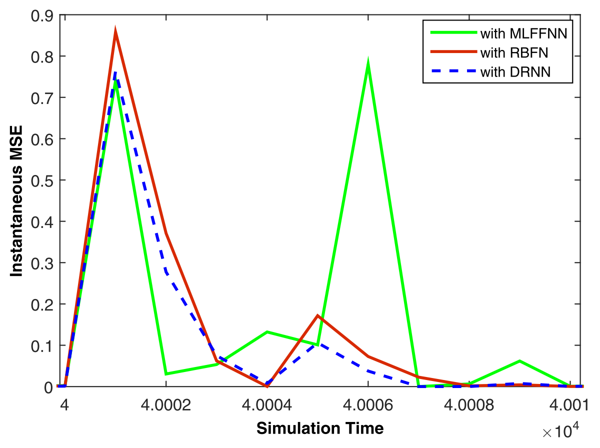

Fig. 16. Instantaneous MSE obtained when parameter variation occurred [Example 1].

9.1.3. *Robustness testing [Example 1]*

One role of identifier is to approximate the unknown dynamics of the plant. The other requirement Figs. 13 which identifier must fulfill is that it adapts its parameters so as to learn the modified Figs. 14 dynamics of the plant which may get changed due to parameter variation. In the following section we have tested and compared the robustness of identifiers to the effects of parameter variations.

9.1.4. *Parameter variation*

Now the identifiers were put to test when parameter α of the given plant was deliberately changed from value 5 units to 10 units at $k = 40,000$th time instant. This caused a rise in the MSE value. The expectation from the identifiers is that they should quickly learn the changed value of parameter and again start following the plant's response. The MSE's obtained from the identifiers at the instant of parameter variation are shown in Fig. 16. It can be easily seen from the MSE plot that DRNN able to capture the new dynamics of plant more rapidly (in less time) followed by RBFN and then MLFFNN. This suggests that DRNN based identifier has more robustness toward parameter variations as compared to MLFFNN and RBFN.

9.2. *Example 2 Epoch mode of identification*

Consider a nonlinear dynamical system whose dynamics are assumed to be unknown and are given as in [11]

$$Y_p(k+1) = F[x_1, x_2, x_3, x_4, x_5] \tag{54}$$

where $x_1 = Y_p(k)$, $x_2 = Y_p(k-1)$, $x_3 = Y_p(k-2)$, $x_4 = r(k)$, $x_5 = r(k-1)$ and unknown function has the following form

$$F[x_1, x_2, x_3, x_4, x_5] = \frac{x_1 x_2 x_3 x_5 [x_3 - 1] + x_4}{\left[1 + x_3^2 + x_2^2\right]} \tag{55}$$

The identification structure of MLFFNN, RBFN and DRNN are given by Eqs. (56)–(58) respectively.

$$Y_{MLFFNN}(k+1) = \widehat{F}[x_1, x_2, x_3, x_4, x_5] \tag{56}$$

$$Y_{RBFN}(k+1) = \widehat{F}[x_1, x_2, x_3, x_4, x_5] \tag{57}$$

$$Y_{DRNN}(k+1) = \widehat{D}[x_1, x_2, x_4] \tag{58}$$

In this example we have trained the identifiers using the epoch-mode of training. A total of 300 input–output training samples were generated from the plant. The training was continued for

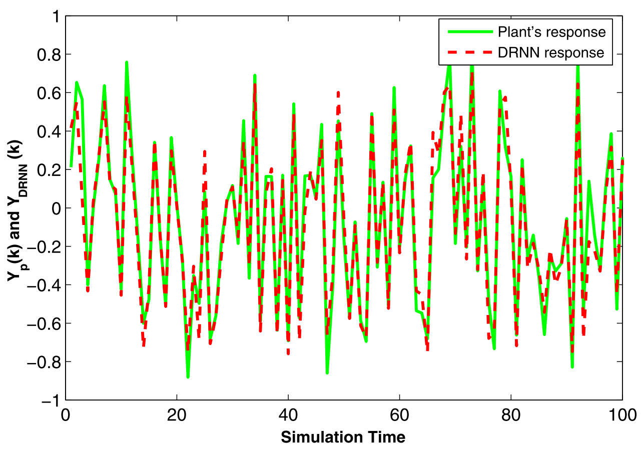

Fig. 17. DRNN identifier response at the end of training [Example 2].

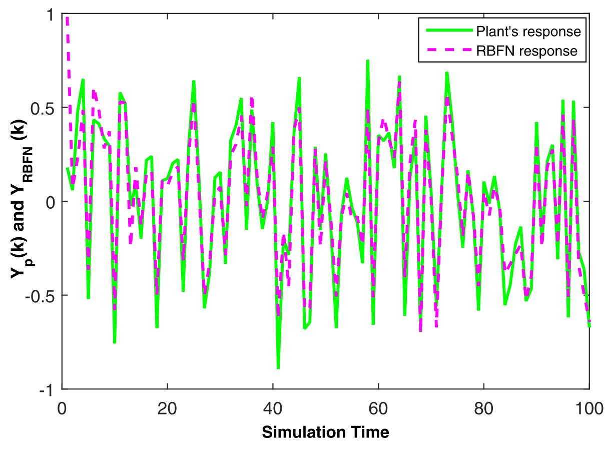

Fig. 18. RBFN identifier response at the end of training [Example 2].

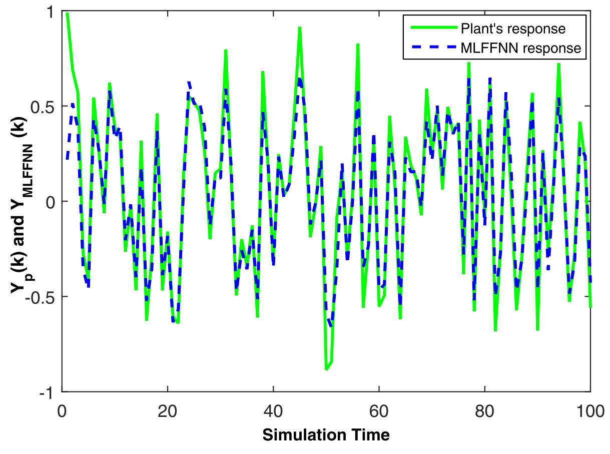

Fig. 19. MLFFNN identifier response at the end of training [Example 2].

20 epochs. The initial value of $\eta_i(0)$ for each parameter of identifier was set to 0.0052. During the training, the external input $r(k)$ is considered to be random having value distributed in the interval $[-1, 1]$.

9.2.1. *Discussion on the obtained simulation results [Example 2]*

The responses of identifiers at the end of the 20th epoch of the training are shown in Figs. 17–19. It is clear from the figures that DRNN is able to capture the dynamics of the plant much better than the RBFN and MLFFNN models. This can also be seen from

Table 3
Comparison of identifiers in terms of various parameters [Example 2].

	DRNN based identifier	RBFN identifier	MLFFNN identifier
Total number of inputs	03	05	05
Number of nodes in hidden layer	10	20	20
Total count of parameters to be tuned	61	81	141
Samples in each epoch	300	300	300
Average MSE (of all 20 epochs)	0.0045	0.031	0.065

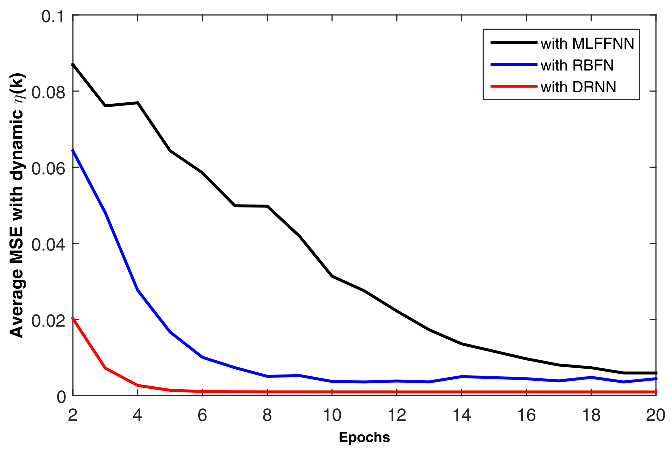

Fig. 20. Average MSE obtained during the training [Example 2].

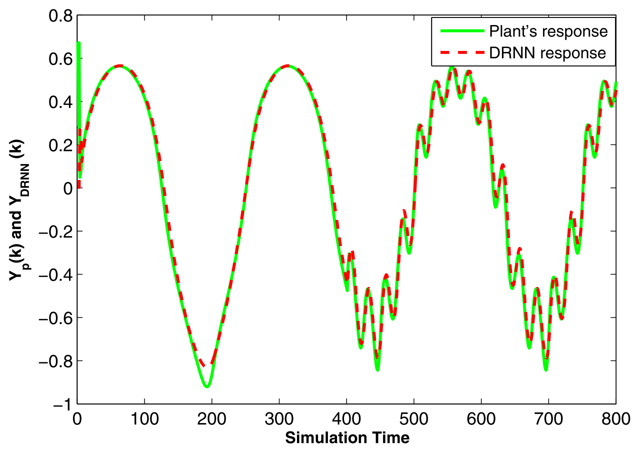

Fig. 21. Validation of DRNN identifier after the training [Example 2].

the MSE plots obtained during the training. It is shown in Fig. 20. It can be easily seen that average MSE obtained in case of DRNN decreases at a much faster rate as compared to RBFN and MLFFNN. This suggests better approximation capability of DRNN. Further, various details associated with this example are shown in Table 3. It can be seen from the table that minimum average MSE value is obtained with DRNN model. Also, DRNN required lesser number of parameters to be tuned. This makes it more computationally efficient than RBFN and MLFFNN.

9.2.2. Validation stage

After the training, the next step is to test the performance of identifiers by using a new external input (whose values lie in a same range as that of input which was used during the training). This step is called as validation. This new external input is given by:

$$r(k) = \begin{cases} sin\left(\frac{2\pi k}{250}\right) & \text{if } \; k \leq 400 \\ 0.8sin\left(\frac{2\pi k}{250}\right) + 0.2sin\left(\frac{2\pi k}{25}\right) & \text{if } \; 400 < x \leq 800 \end{cases} \tag{59}$$

The corresponding responses of the DRNN, RBFN and MLFFNN based identifiers are shown in Figs. 21–23 respectively. It can be seen that DRNN response is much closer to the plant's response as compared to responses obtained with RBFN and MLFFNN identifiers. This shows that DRNN have much better ability of approximating the unknown dynamics as compared to MLFFNN and RBFN.

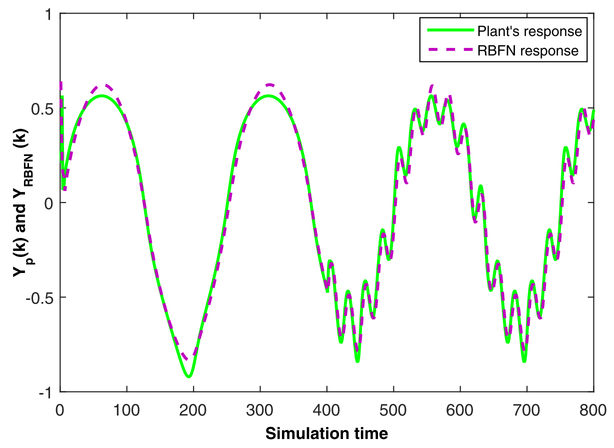

Fig. 22. Validation of RBFN identifier after the training [Example 2].

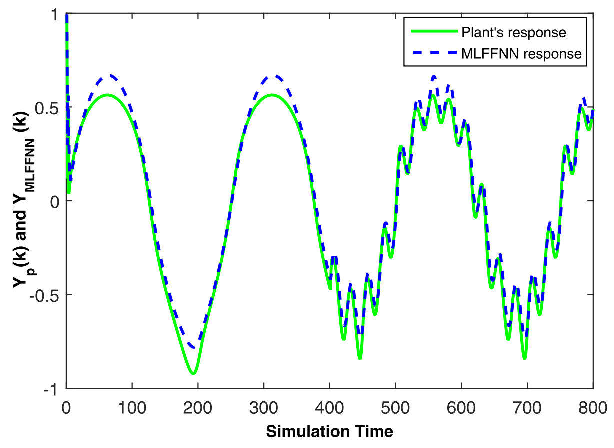

Fig. 23. Validation of MLFFNN identifier after the training [Example 2].

9.3. Example 3 Robotic manipulator dynamics identification

Any n serial link robotic manipulator can be described by the following dynamical equation:

$$M(Y_p)\ddot{Y}_p + C(Y_p, \dot{Y}_p)\dot{Y}_p + D\dot{Y}_p + G(Y_p) = \tau \tag{60}$$

Here, Y_p, $\dot{Y_p}$ and $\ddot{Y_p} \in R^l$ denote the position of joint, velocity and the accelerations respectively. The $l \times l$ real matrices: M, C, D denote the symmetric positive definite inertia matrix, centrifugal Coriolis matrix and positive definite diagonal matrix for damping friction coefficients of each joint respectively. The $l \times l$ real matrices: C, D, M denote the centrifugal Coriolis matrix, positive definite diagonal matrix and symmetric positive definite inertia matrix and they denote coefficients of damping friction for each joint respectively. In case of one link robotic manipulator the value of l is 1. The gravity term is denoted by real $l \times l$ matrix $G(Y_p)$ and joint's input vector torques are represented by $\tau \in R^l$. Further, l is 1 in the case of one-link robotic manipulator. The dynamics of a one-link robotic manipulator can be described by the following second order differential equation [35]:

$$ml^2 \frac{d^2Y_p(t)}{dt^2} + D\frac{dY_p(t)}{dt} + mglcos(Y_p(t)) = r(t) \tag{61}$$

where $Y_p(t)$ represents robotic manipulator arm's angular position, l denotes link length, $D\frac{dY_p(t)}{dt}$ represents viscous friction torque and $g =$ is acceleration due to gravity $= 9.8$ m/s^2. For simplicity, $m = l = D = 1$ are used in this example. The external input torque to the robotic arm is denoted by $r(t)$. Further, if robot movement is in parallel to that of horizontal plane then the effect of gravity can be excluded from the analysis [35]. The corresponding state space representation of above dynamical differential equation is obtained as:

$$\begin{bmatrix} \dot{Y_p} \\ \ddot{Y_p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-gcos(Y_p}{l} & \frac{-D}{ml^2} \end{bmatrix} \begin{bmatrix} Y_p \\ \dot{Y_p} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} [r] \tag{62}$$

The equivalent difference equation of Eq. (61) is given by

$$Y_p(k+1) = F[x_1, x_2, x_3] \tag{63}$$

where $x_1 = Y_P(k), x_2 = Y_P(k-1)$ and $x_3 = r(k)$. The unknown function F is given by

$$\begin{aligned} F[x_1, x_2, x_3] = &\ (2 - T)x_1 + x_2(T - 1) \\ &- 9.8T^2cos(x_2) + T^2x_3 \end{aligned} \tag{64}$$

The sampling period T is 0.45 s. The corresponding NN based identification models are given by:

$$Y_{MLFFNN}(k+1) = \widehat{F}[x_1, x_2, x_3] \tag{65}$$

$$Y_{RBFN}(k+1) = \widehat{F}[x_1, x_2, x_3] \tag{66}$$

$$Y_{DRNN}(k+1) = \widehat{F}[x_1, x_2, x_3] \tag{67}$$

9.3.1. Discussion on the obtained simulation results [Example 3]

The initial and final stage responses of identifiers during the online training with dynamic learning rate (whose initial value was set 0.005) are shown in Figs. 24 and 25 respectively. From the initial response of identifiers, it can be easily seen that DRNN response is getting improved at a much faster rate as compared to RBFN and MLFFNN response and within 300 time instants the DRNN output started following the output of the plant. The learning time in case of RBFN and MLFFNN based identifiers was more as compared to DRNN. The various details related to this example are listed in Table 4. It can be seen that DRNN provided minimum MSE value and requires minimum number of parameters to be tuned as compared to RBFN and MLFFNN.

9.3.2. Comparison of performance of proposed identification model with the existing literature [Example 3]

If we compare the performance of our proposed identification model with that of neural network based identification model used in [36] it can be seen that our proposed identification model, which consists of only three inputs and 10 neurons in the single hidden layer, is able to learn the dynamics of the robotic manipulator in around 1200 time instants as compared to time (which is equal to around 3000 time instants) taken by identifier used in [36] having 4 inputs and 30 neurons in the single hidden layer.

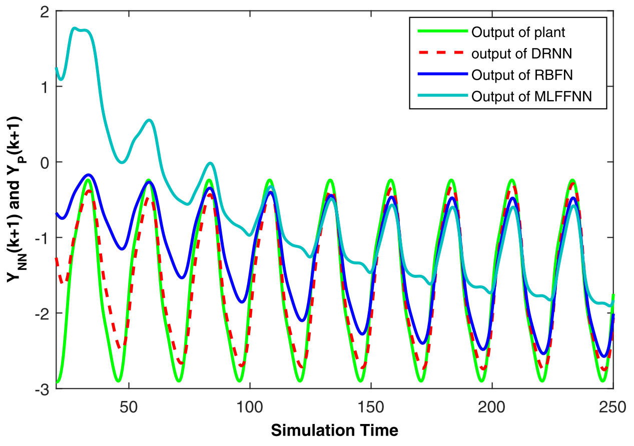

Fig. 24. Responses of identifiers at the earlier stages of training [Example 3].

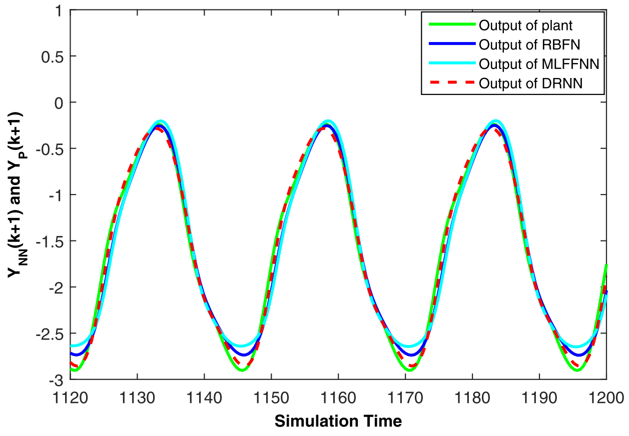

Fig. 25. Responses of identifiers at final stage of training [Example 3].

9.3.3. Robustness testing [Example 3]

In this section the robustness of identifiers is tested and compared to the effects of disturbance/noise signals. The source of these signals is usually unknown. The identifier must compensate the effects of these unwanted signals. The performance of identifiers is tested and compared under the effects of disturbance signals in the following subsection.

9.3.4. Disturbance signal

At 5400th time instant, a disturbance signal of 0.7 value is added in the output of identifiers. This has caused a rise in the MSE value. The performance of identifiers is evaluated in terms of the time taken by them to compensate the effect of disturbance. Fig. 26 shows the MSE plot when disturbance signal is added. From the MSE plot we can see that MSE obtained with DRNN do not change much and also took less time to drop to zero value as compared to time it took with RBFN and MLFFNN identifiers. This shows that DRNN is more robust than RBFN and MLFFNN.

Table 4
Performance comparison of identifiers [Example 3].

	DRNN identifier	RBFN identifier	MLFFNN identifier
Total number of inputs	03	03	03
Number of nodes in hidden layer	10	20	20
Total count of parameters to be tuned	61	81	101
Simulation time	1200	1200	1200
Average MSE	0.0060	0.015	0.091

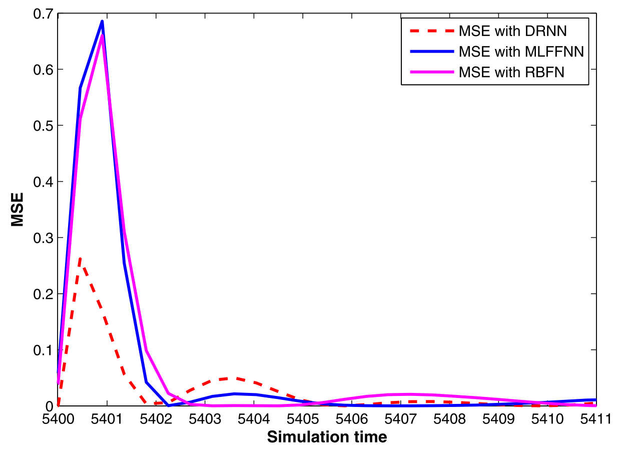

Fig. 26. MSE response of identifiers when disturbance signal is added in the system [Example 3].

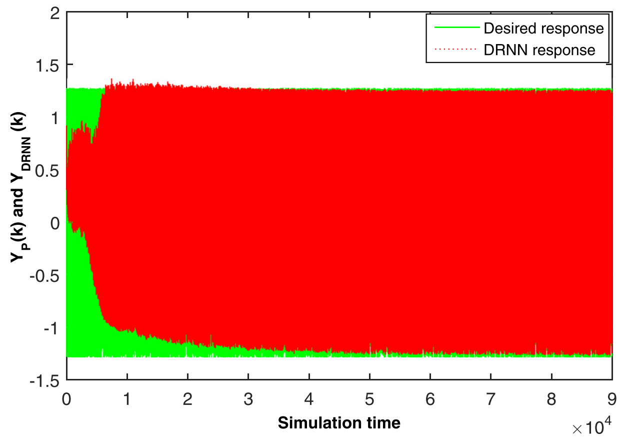

Fig. 27. Response of DRNN identifier during the online training [Example 4].

9.4. Example 4 Chaotic series prediction

The Henon chaotic sequence with one time delay and having two sensitive parameters is given as in [37]:

$$Y_P(k+1) = F[x_1, x_2, x_3] \tag{68}$$

where $x_1 = Y_P(k), x_2 = Y_P(k-1)$ and $x_3 = r(k)$. The externally applied input $r(k)$ is considered to be a unit step input. The unknown function in Eq. (68) is $F = -Mx_1^2 + Qx_2 + x_3$. Further, $x_1 = 0.4$ and $x_2 = 0.4$ are the initial conditions assumed in this example. The values of parameters M and Q are 1.4 and 0.3 respectively. The initial value of dynamic learning rate $\eta_i(0)$ is set equal to 0.0065. The corresponding identification model's dynamic equations are:

$$Y_{MLFFNN}(k+1) = \widehat{F}[x_1, x_2, x_3] \tag{69}$$

$$Y_{RBFN}(k+1) = \widehat{F}[x_1, x_2, x_3] \tag{70}$$

$$Y_{DRNN}(k+1) = \widehat{F}[x_1, x_2, x_3] \tag{71}$$

The instantaneous online training was continued for 96, 000 time steps and Figs. 27–29 show the total response of DRNN, RBFN and MLFFNN identifiers respectively.

9.4.1. Discussion on the obtained simulation results [Example 4]

It can be seen from the figures that DRNN identifier is performing better as compared to that of RBFN and MLFFNN since its output started to follow plant's output in a much short period of time as compared to time taken by RBFN and MLFFNN. The instantaneous MSE's obtained from the identifiers during the online training are shown in Figs. 30–32. Again the same conclusion can be drawn that DRNN identifier is learning the unknown dynamics at a much faster pace as compared to RBFN and MLFFNN as its MSE can be seen to decrease at a much faster rate. Further, Figs. 33–35 show the magnified view of the responses of DRNN, RBFN and MLFFNN identifiers respectively around the 18, 980 time

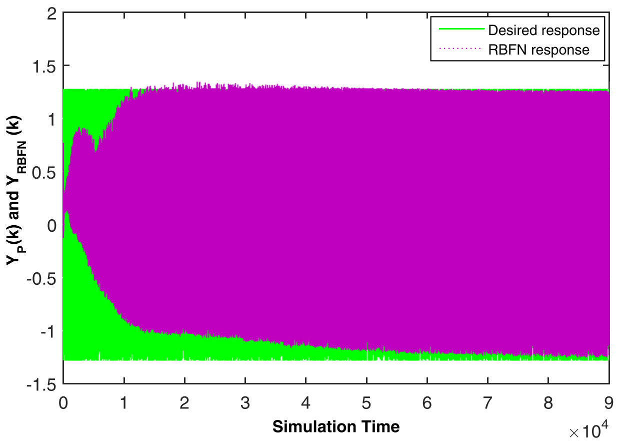

Fig. 28. Response of RBFN identifier during the online training [Example 4].

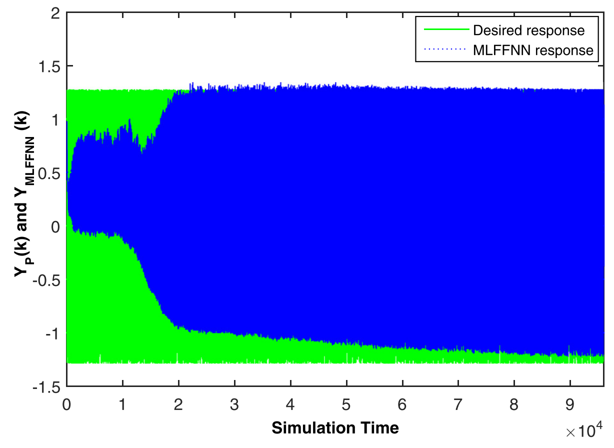

Fig. 29. Response of MLFFNN identifier during the online training [Example 4].

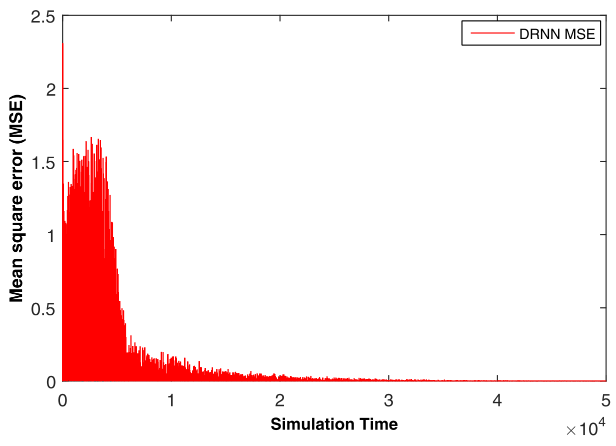

Fig. 30. MSE obtained with DRNN identifier during the online training [Example 4].

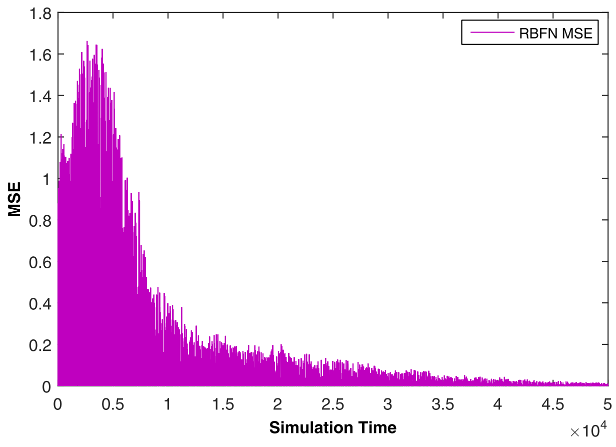

Fig. 31. MSE obtained with RBFN identifier during the online training [Example 4].

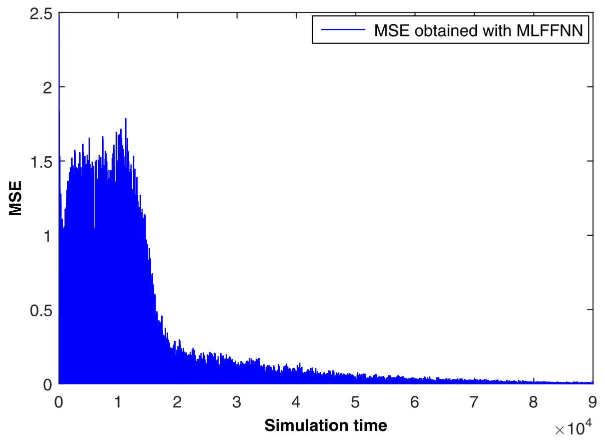

Fig. 32. MSE obtained with MLFFNN identifier during the online training [Example 4].

instant. Again, we can see that DRNN is performing better than RBFN and MLFFNN identifiers. The rest of the details associated with this example is given in Table 5. It can be easily that DRNN is giving minimum MSE value and requires minimum number of parameters to be tuned.

9.4.2. Performance comparison of proposed identification model with the existing literature [Example 4]

Further, the performance of our proposed DRNN based identifier is found to be better than that of the neuro-fuzzy based

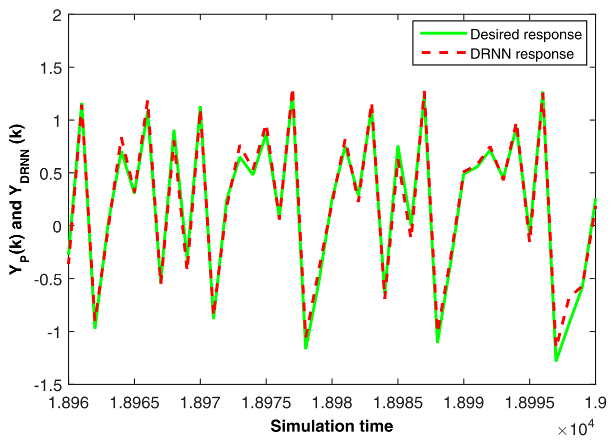

Fig. 33. DRNN identifier response during the online training around 18980 instant [Example 4].

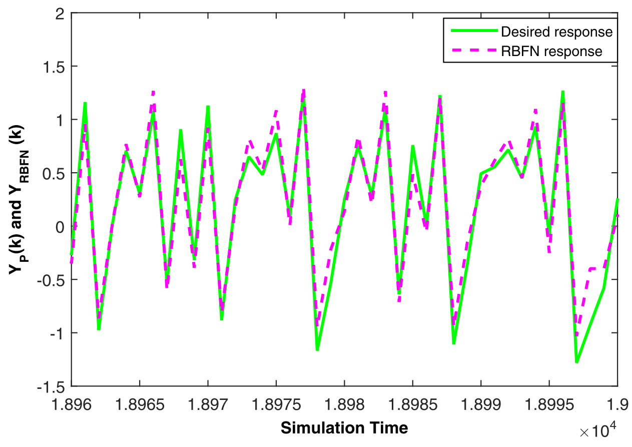

Fig. 34. RBFN identifier response during the online training around 18, 980 instant [Example 4].

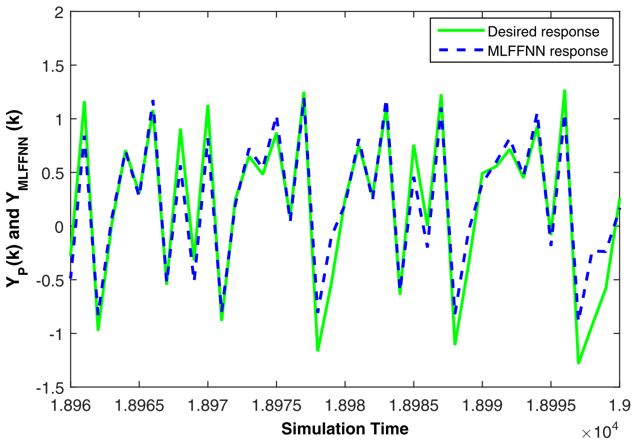

Fig. 35. MLFFNN identifier response during the online training around 18, 980 instant [Example 4].

identification model used in [37]. The comparison is as follows: In [37], two cases were considered. In first case the neuro-fuzzy model used for identification consists of one hidden layer with 18 hidden nodes and 6 inputs (present as well as past history of inputs and outputs) were applied. Total 700 samples were used during the learning. At the end of the training the average MSE value found was 0.014. In second case, 9 inputs were used and 27 hidden nodes were considered in the single hidden layer. The aver-

Table 5
Performance comparison of identifiers [Example 4].

	DRNN identifier	RBFN identifier	MLFFNN identifier
Total number of inputs	03	03	03
Number of nodes in hidden layer	10	20	20
Total count of parameters to be tuned	61	81	101
Average MSE	0.0023	0.015	0.0522
Simulation time	96000	96000	96000

age MSE value found in this case is 0.015. In our proposed DRNN based identification model, we have considered only 3 input signals, one hidden layer containing only 10 recurrent neurons. The average MSE value found at the end of the training was 0.0023 which is better than the average MSE value obtained in [37].

10. Some limitations/challenges of the proposed method

1. Most neural network architectures are black box models. They cannot be checked whether their solution is plausible, i.e. their final state cannot be interpreted in terms of rules.
2. The solution obtained from the learning process usually cannot be interpreted. Thus, we cannot relate the values of tuned weights with the actual values of the unknown parameters of the plant.
3. A neural network usually cannot be initialized with prior knowledge if it is available, and thus the network must learn from scratch.
4. The selection of optimal number of neurons in the hidden layer and the number of hidden layers is a very difficult task.
5. The back-propagation algorithm is generally slow and can trap in the local minima. To avoid this, we have developed an adaptive learning rate in our paper.
6. The initialization of weights values has an impact on the training time. But it is very difficult to choose these initial values.

11. Conclusion and future works

11.1. Conclusion

In this article, a novel DRNN based identification model is proposed for the problem of online and offline identification of nonlinear systems. The proposed DRNN structure comprises of the self-feedback loops and requires fewer inputs for learning the dynamics of the systems. An adaptive learning algorithm is developed which is driven based on the conventional backpropagation method using gradient descent algorithm. In order to improve the performance of the algorithm, novel adaptive learning rates are developed in the sense of discrete Lyapunov stability method. To show the capabilities of proposed DRNN identification model, performance comparison is made with the state of art techniques like MLFFNN and RBFN. The DRNN identification model is tested on both theoretical and experimental examples which contain complex nonlinear systems. Simulation results demonstrate that in almost all the cases examined, the DRNN has shown superior modeling accuracy and has also shown more robustness as compared to the other 2 identification models. Thus, DRNN can be regarded as a general identification network that can be applied to the identification of a wide class of nonlinear dynamic systems.

11.2. Future works

There still exist a number of open studies in the follow up the research. In regarding the neural network enhancement to approximate complex systems, the potential studies will cover to investigate new structure of RNNs and new adaptive laws to adjust the weights of these RNNs. As the structure changes with the introduction of more feedback connections, the new DRNN will require lesser number of inputs and hidden neurons for learning the dynamics of the system. The new learning algorithm will involve the application of evolutionary optimization techniques like PSO, differential evolution, water drop method etc. combined with the Lyapunov stability method. This will result in development of more effective and efficient learning algorithm. Another future work will be to identify the MIMO type of plants which offers more complexity than the SISO type.

Acknowledgement

The authors would like to sincerely thanks the Editor in Chief (Prof. Zidong Wang), the handling Editor (Prof. H.R. Karimi) and the anonymous reviewers for their timely and constructive reviews. Their comments have helped the authors in improving the quality of the manuscript.

References

[1] J. Deng, Dynamic neural networks with hybrid structures for nonlinear system identification, Eng. Applic. Artif. Intell. 26 (1) (2013) 281–292.
[2] A. Isidori, Nonlinear Control Systems, Springer Science & Business Media, 2013.
[3] F. Ding, T. Chen, Identification of Hammerstein nonlinear ARMAX systems, Automatica 41 (9) (2005) 1479–1489.
[4] M. Schetzen, The Volterra and Wiener Theories of Nonlinear Systems, 1980.
[5] T.-T. Lee, J.-T. Jeng, The Chebyshev-polynomials-based unified model neural networks for function approximation, IEEE Trans. Syst. Man Cybern. Part B (Cybern.) 28 (6) (1998) 925–935.
[6] O. Nelles, Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models, Springer Science & Business Media, 2013.
[7] Y. Wei, J.H. Park, H.R. Karimi, Y.-C. Tian, H. Jung, Improved stability and stabilization results for stochastic synchronization of continuous-time semi-Markovian jump neural networks with time-varying delay, IEEE Trans. Neural Networks Learn. Syst. (2017) 1–14, doi:10.1109/TNNLS.2017.2696582.
[8] Y. Wei, J. Qiu, H.R. Karimi, Reliable output feedback control of discrete-time fuzzy affine systems with actuator faults, IEEE Trans. Circ. Syst. I: Regul. Pap. 64 (1) (2017) 170–181.
[9] R. Coban, A context layered locally recurrent neural network for dynamic system identification, Eng. Applic. Artif. Intell. 26 (1) (2013) 241–250.
[10] S.A. Billings, S.I. Chen, Identification of non-linear rational systems using a prediction-error estimation algorithm, Int. J. Syst. Sci. 20 (3) (1989) 467–494.
[11] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Networks 1 (1) (1990) 4–27.
[12] S. Chen, S.A. Billings, Neural networks for nonlinear dynamic system modelling and identification, Int. J. Control 56 (2) (1992) 319–346.
[13] D.S. Broomhead, D. Lowe, Radial basis functions, multi-variable functional interpolation and adaptive networks, Technical Report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.
[14] J. Park, I.W. Sandberg, Universal approximation using radial-basis-function networks, Neural Comput. 3 (2) (1991) 246–257.
[15] P.S. Sastry, G. Santharam, K.P. Unnikrishnan, Memory neuron networks for identification and control of dynamical systems, IEEE Trans. Neural Networks 5 (2) (1994) 306–319.
[16] K.P. Unnikrishnan, J.J. Hopfield, D.W. Tank, Connected-digit speaker-dependent speech recognition using a neural network with time-delayed connections, IEEE Trans. Signal Process. 39 (3) (1991) 698–713.
[17] H.-G. Han, Y.-N. Guo, J.-F. Qiao, Nonlinear system modeling using a self-organizing recurrent radial basis function neural network, Appl. Soft Comput. (2017), doi:10.1016/j.asoc.2017.10.030.
[18] D. Xu, Z. Li, W. Wu, Convergence of gradient method for a fully recurrent neural network, Soft Comput. 14 (3) (2010) 245.
[19] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.
[20] K.-I. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural Networks 2 (3) (1989) 183–192.

[21] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, Technical Report, DTIC Document, 1985.
[22] G.P. Liu, V. Kadirkamanathan, S.A. Billings, On-line identification of nonlinear systems using Volterra polynomial basis function neural networks, Neural Networks 11 (9) (1998) 1645–1657.
[23] M.O. Efe, O. Kaynak, A comparative study of neural network structures in identification of nonlinear systems, Mechatronics 9 (3) (1999) 287–300.
[24] I.S. Baruch, J.M. Flores, R. Garrido, A fuzzy neural recurrent multi-model for systems identification and control, in: 2001 European Control Conference (ECC), IEEE, 2001, pp. 3540–3545.
[25] I. Gabrijel, A. Dobnikar, On-line identification and reconstruction of finite automata with generalized recurrent neural networks, Neural Networks 16 (1) (2003) 101–120.
[26] J. de Jesús Rubio, W. Yu, Dead-zone Kalman filter algorithm for recurrent neural networks, in: 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05, IEEE, 2005, pp. 2562–2567.
[27] Y.-q. Zhu, W.-f. Xie, J. Yao, Nonlinear system identification using genetic algorithm based recurrent neural networks, in: Canadian Conference on Electrical and Computer Engineering, 2006. CCECE'06, IEEE, 2006, pp. 571–575.
[28] A. Savran, Multifeedback-layer neural network, IEEE Trans. Neural Networks 18 (2) (2007) 373–384.
[29] R.K. Al Seyab, Y. Cao, Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation, J. Process Control 18 (6) (2008) 568–581.
[30] M.V. Rajesh, R. Archana, A. Unnikrishnan, R. Gopikakaumari, Particle filter based neural network modeling of nonlinear systems for state space estimation, in: Chinese Control and Decision Conference, 2009. CCDC'09, IEEE, 2009, pp. 1477–1482.
[31] A. Banakar, M.F. Azeem, Local recurrent sigmoidal–wavelet neurons in feed–forward neural network for forecasting of dynamic systems: theory, Appl. Soft Comput. 12 (3) (2012) 1187–1200.
[32] M. Atencia, G. Joya, F. Sandoval, Identification of noisy dynamical systems with parameter estimation based on Hopfield neural networks, Neurocomputing 121 (2013) 14–24.
[33] A. Roudbari, F. Saghafi, Intelligent modeling and identification of aircraft nonlinear flight, Chin. J. Aeronaut. 27 (4) (2014) 759–771.
[34] L. Behera, I. Kar, Intelligent Systems and Control Principles and Applications, Oxford University Press, Inc., 2010.
[35] W. Sanxiu, J. Shengtao, Adaptive friction compensation of robot manipulator, Electronics and Signal Processing, 2011, pp. 127–134.
[36] R. Kumar, S. Srivastava, J.R.P. Gupta, Online modeling and adaptive control of robotic manipulators using Gaussian radial basis function networks, Neural Computing and Applications, 2016, pp. 1–17.
[37] L.P. Maguire, B. Roche, T.M. McGinnity, L.J. McDaid, Predicting a chaotic time series using a fuzzy neural network, Inf. Sci. 112 (1–4) (1998) 125–136.

Rajesh Kumar received the B.Tech. degree in Applied Electronics and Instrumentation Engineering in 2011, and M.Tech. degree in Electrical Engineering (Control Systems) in 2013 from National Institute of Technology, Kurukshetra, India. He is currently pursuing his Ph.D. in the area of control and identification of nonlinear systems using intelligent tools from Netaji Subhas Institute of Technology, New Delhi (University of Delhi), India. His research interests include: Neural networks, intelligent control, system identification, stability analysis, Fuzzy systems.

Smriti Srivastava received the B.E. degree in Electrical Engineering and the M.Tech. degree in Heavy Electrical Equipment from Maulana Azad College of Technology [now Maulana Azad National Institute of Technology (MANIT)], Bhopal, India, in 1987 and 1991, respectively, and the Ph.D. degree in intelligent control from the Indian Institute of Technology, New Delhi, India, in 2005. She is the author of a number of publications in transactions, journals and conferences in the areas of neural networks, fuzzy logic, and control systems. She has given a number of invited talks and tutorials mostly in the area of fuzzy logic, process control, and neural networks. Her current research interests include neural networks, fuzzy logic, and hybrid methods in modeling, identification and control of nonlinear systems.

J.R.P. Gupta received the B.Sc. (Engg.) degree in electrical engineering from Muzaffarpur Institute of Technology (MIT), Muzaffarpur, India, in 1972 and the Ph.D. degree from the University of Bihar, Muzaffarpur, India, in 1983. He is the author of more than 100 publications in reputed journals and conferences. He was a recipient of the Institution of Electrical and Telecommunication Engineers India Best Paper Award. He is a life member of ISTE and a Senior Member of IEEE. His research interests include power electronics, control systems, and biomedical instrumentation.

Amit Mohindru received the B.Tech. degree in Electronics and Communication Engineering from the LNM Institute of Information Technology, Jaipur, India in 2013 and M.Tech. degree in Electronics and Communication from Indraprastha Institute of Information Technology, New Delhi, India in 2016. His areas of interests: Control systems, signals and systems, communications systems, intelligent techniques.