Neural Computing and Applications (2021) 33:1515–1531
https://doi.org/10.1007/s00521-020-05077-1

ORIGINAL ARTICLE

Deep learning controller for nonlinear system based on Lyapunov stability criterion

Ahmad M. Zaki[1] · Ahmad M. El-Nagar[1] · Mohammad El-Bardini[1] · F. A. S. Soliman[2]

Received: 6 December 2019 / Accepted: 3 June 2020 / Published online: 13 June 2020

Abstract
For the current paper, the technique of feed-forward neural network deep learning controller (FFNNDLC) for the nonlinear systems is proposed. The FFNNDLC combines the features of the multilayer feed-forward neural network (FFNN) and restricted Boltzmann machine (RBM). The RBM is a very important part for the deep learning controller, and it is applied in order to initialize a multilayer FFNN by performing unsupervised pretraining, where all the weights are equally zero. The weight laws for the proposed network are developed by Lyapunov stability method. The proposed controller is mainly compared with FFNN controller (FFNNC) and other controllers, where all the weights values for all the designed controllers are equally zero. The proposed FFNNDLC is able to respond the effect of the system uncertainties and external disturbances compared with other existing schemes as shown in simulation results section. To show the ability of the proposed controller to deal with a real system, it is implemented practically using an ARDUNIO DUE kit microcontroller for controlling an electromechanical system. It is proved that the proposed FFNNDLC is faster than other FFNNCs in which the parameters are learned using the backpropagation method. Besides, it is able to deal with the changes in both the disturbance and the system parameters.

Keywords Lyapunov stability · Neural network · Deep learning controller · Nonlinear system · Restricted Boltzmann machine

1 Introduction

Nonlinear systems are the most common in industrial processes where those are defined as their inputs are related to nonlinear to the outputs. These systems have an important area in the research field because the modeling and estimating system nonlinearities are more difficult and contain inherent uncertainties [1–3]. The development of a controller for the nonlinear systems should be skillful to track its output precisely. Noting that, the application of the traditional controllers for the nonlinear systems is inappropriate due to that such nonlinear systems normally suffer from many difficult problems, such as the nonlinear dynamic behavioral, the constraints on the manipulated variables, the uncertain and time-varying parameters, unmeasured and frequent disturbances [4, 5]. Thus, the development of the controller is necessary to control such challenges. Nowadays, many researchers are interested in artificial intelligent (AI) controllers because of their ability to make the nonlinear systems are stable [6–10].

In this concern, artificial neural networks (ANNs) are one of the AI, which are defined as biologically inspired programming paradigm. ANNs enable the controller to learn from observational data [11–14]. The ANNs are considered one of the most successful techniques in nonlinear control applications. In [15], ANN controllers were

✉ Ahmad M. El-Nagar
ahmed.elnagar@el-eng.menofia.edu.eg

Ahmad M. Zaki
ahmed_zaki889@yahoo.com

Mohammad El-Bardini
dralbardini@el-eng.menofia.edu.eg

F. A. S. Soliman
fouad.saad.soliman@gmail.com

1 Department of Industrial Electronics and Control Engineering, Faculty of Electronic Engineering, Menoufia University, Menof 32852, Egypt

2 Department of Electronics and Computers Engineering, Nuclear Materials Authority, 530, El-Maadi, Cairo, Egypt

developed in order to control pressure. Also, it is described in [16] that the diagonal recurrent neural network controller (DRNNC) shows its ability to control the dynamic behavior of the nonlinear plant. The robust analysis of the neural network control was utilized for controlling the speed of a DC motor, which was described in [17]. The robustness was ensured using an internal model controller. In [18], feed-forward neural network controller (FFNNC) with a hybrid method (FFNNC-hybrid) was introduced. The learning of the FFNNC-hybrid was performed based on unsupervised (self-organized leaning) and supervised (gradient descent) method. In [19], the FFNNC and nonlinear autoregressive neural network were used to overcome the delay control for offshore platform system. A direct adaptive inverse control was introduced using FFNNC for controlling the nonlinear system [20]. An adaptive feed-forward neural controller and PID controller were used for controlling the joint-angle position of the SCARA parallel robot [21]. The FFNNC was used to control the angle with position of a nonlinear inverted pendulum system [22].

The controllers design was performed based on FFNNC, which are commonly related to the initialized weights. Noting that, if the process of initializing weights is not appropriate, the NN gets stumbling in local minima, leading the training procedure to unsatisfactory ending or the vanishing gradient issue is happened during the initial layers training and the NN training becomes infeasible [23]. This defect affects the performance of the controller, and it makes the controller unstable sometimes. On the other hand, machine learning (ML) is a part of AI, which is performed based on the techniques that make the computers to discover things from input data. In this concern, deep learning (DL) is a modern topic of ML. DL has learning numerous levels, which supports to deliver sense of data such as images [24–26], sound [27], and text [28]. The most common use of DL is the modeling process of the nonlinear systems.

1.1 Literature review

In [29], the DL framework was proposed for modeling the nonlinear systems. It learns deep reconstruction model, which is performed based on Elman neural network (ENN) and RBM (ENN-RBM) for initializing only the first layer. In [30], the regression and classification issues can be solved using the randomized algorithms and DL techniques. In this concern, these algorithms are used for constructing the statistical features and training the hidden weights. In [31], deep belief network (DBN) with partial least square regression (PLSR) was investigated for the nonlinear system modeling, where the problem of the weights improvement for DBN is performed based on PLSR. In [32], this work was tried for the prediction traffic speeds of multiple road links simultaneously by constructing a DL based on multitask learning model. In [33], the researcher proposed a new method for the optical identification of parts without specific codes, which is performed based on inherent geometrical features with DL. Finally, in [34–37], the researchers cover some DL techniques, which already are applied in radiology and identify radiogenomic associations in breast cancer for image processing.

1.2 Motivation

From the previous studies, it is clear that the application field of DL was limited only for modeling systems and image processing and it does not cover the control area. Besides, the nonlinear systems suffer from uncertainties and external disturbance. So, the main target of the present study is to shed further light on designing a stable feed-forward neural network deep learning controller (FFNNDLC), to be applied for controlling the nonlinear systems to overcome the problems of system uncertainties. The proposed FFNNDLC uses the RBM to initialize the weight values. Lyapunov stability method is used for updating the adaptation parameters laws. The FFNNDLC is learned swift to keep a track of trajectory and to overcome the outside disturbances and the changes in the system parameters. In this concern, FFNNDLC is utilized to the uncertain nonlinear systems in order to guarantee the optimum controlling and decreasing the influence of uncertainties and outside disturbances. Of course, these advantages of FFNNDLC make it sturdier than FFNNC under the same conditions. On the other hand, the proposed controller is implemented practically for controlling a real system.

1.3 Novelties and contributions

The major target of the present paper is summed up as:

- A new controller is proposed in the control field based on DL technique.
- Developing the adaptation law for the proposed controller parameters based on Lyapunov theorem to warranty a stable controller.
- Implementing practically the proposed controller based on an ARDUINO DUE kit for controlling a real system.
- The proposed controller has the ability to reduce the uncertainties influence and outside disturbances compared to other controllers under the same conditions.

The paper organization is as follows: Restricted Boltzmann machine is explained in part 2. The mathematical formulation for restricted Boltzmann machine is introduced in this section. Feed-forward neural network deep learning controller and the Lyapunov stability derivation of FFNNDLC are explained in part 3. The FFNNDL system and controller training steps are introduced in part 4. The simulation results for nonlinear systems are introduced in part 5. The practical results are introduced in part 6. Finally, part 7 presents the conclusion followed by the references.

2 Restricted Boltzmann machine

RBM is an energy-based model, which uses two layers: visible and hidden layers that consists of a group of the visible nodes; V, and a group of the hidden nodes; H. The conventional approach was introduced for training RBM where a linear nature of the neural units was considered the main drawback in this method [29, 38]. Another approach to derive RBM training rules and overcome the main shortcoming of traditional method was introduced in [39, 40]. It takes into computation nonlinear nature of neural nodes and minimizing the mean square error (MSE). The RBM uses three layers as illustrated in Fig. 1.

Based on this approach, contrastive divergence (CD) was proposed by Hinton for learning RBM [41, 42]. Let $V(q-1)$ will be the input data, which shifts to the visible layer at time $(q-1)$. Then, the output of the hidden layer is determined as follows:

$$H_j(q-1) = F\left(\sum_i^N W_{ij}\,V_i(q-1) + b_j\right), \quad i = 1,\ldots,N \quad j = 1,\ldots P \text{ and } q = 1,\ldots,K \tag{1}$$

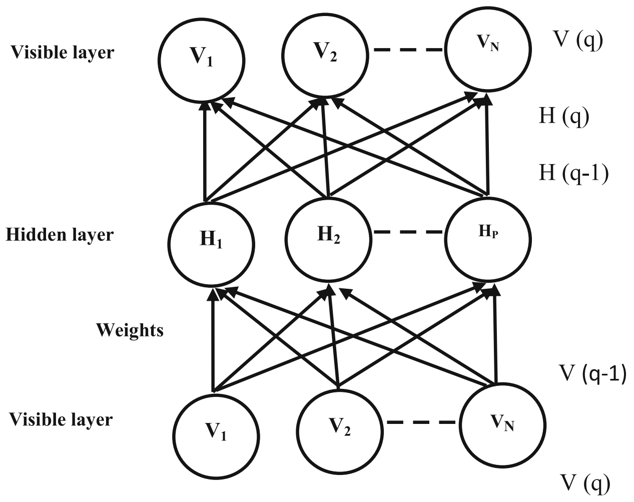


Fig. 1 Structure of RBM

where $\boldsymbol{b} = [b_1 \quad \cdots \quad b_P]^T$ is the biases vector for the hidden nodes, V_i represents the binary state of the visible node, and W_{ij} represents the weight between the visible node i and the hidden node j. N and M are the number of the visible nodes and the hidden nodes, respectively. F represents sigmoid activation function; $F(Q) = 1/(1+\exp(-Q))$.

The inverse layer reconstructs the data from the hidden layer. As a result, $V(q)$ is obtained at time; q as follows:

$$V_i(q) = F\left(\sum_j^P W_{ji}H_j(q-1) + a_i\right) \tag{2}$$

where $\boldsymbol{a} = [a_1 \quad \ldots \quad a_N]^T$ is the biases vector for the visible nodes, H_j represents the binary status of a hidden node, W_{ji} represents the weight between the hidden node j and the visible node i. Subsequently, $V(q)$, which transfers to the visible layer and the hidden layer output, is obtained by the next procedure:

$$H_j(q) = F\left(\sum_i^N W_{ij}V_i(q) + b_j\right) \tag{3}$$

The parameters training rule for the weights and biases of nonlinear RBM in case of CD-K is illustrated as [40]:

$$\begin{aligned} W_{ij}(\hbar+1) &= W_{ij}(\hbar) \\ &+ \alpha\left(\sum_{q=1}^K \left(H_j(q) - H_j(q-1)\right) V_i(q)\, F'\left(S_j(q)\right)\right. \\ &\left.+ (V_i(q) - V_i(q-1))\, H_j(q-1)\, F'(S_i(q))\right) \end{aligned} \tag{4}$$

$$b_j(\hbar+1) = b_j(\hbar) + \alpha\left(\sum_{q=1}^K \left(H_j(q) - H_j(q-1)\right) F'\left(S_j(q)\right)\right) \tag{5}$$

$$a_i(\hbar+1) = a_i(\hbar) + \alpha\left(\sum_{q=1}^K \left(V_i(q) - V_i(q-1)\right) F'\left(S_i(q)\right)\right) \tag{6}$$

where α is the learning rate of RBM and $\hbar$ is the iteration number.

RBM is the very important part for the deep controller, which is used for initializing the FFNN based on performing unsupervised pretraining, where all weights are equal to zero. The stack parameters of RBM also match to the parameters of the multilayer FFNN. Therefore, once the stack of RBM is trained, available parameters can be used to initialize the first layer of FFNN and so on the next layers.

3 Feed-forward neural network deep learning controller

Basically, any ANN with more than two layers is deep. DL is a relatively new advancement in ANN programming and represents a way to train deep neural networks [43]. The DL methods aim to learn feature hierarchies with features from the higher levels of the hierarchy formed by the composition of lower-level features. They include learning methods for a wide array of deep architectures, including ANN with hidden layers [44]. Merge with the features of RBM and multilayer FFNN, FFNNDLC is proposed.

3.1 Feed-forward neural network

The typical four-layer FFNN is shown in Fig. 2 [45]. It contains an input layer, two hidden layers and an output layer.

Input layer: each node is an external input in this concern, and the inputs are denoted as $x_1, x_2, \ldots, x_n$.

Hidden layer (1): each node is determined as the following:

$$net_j^{(1)} = \sum_{i=1}^{n} \psi_{ji} x_i + T_j\,, \quad j = 1, \ldots, J \tag{7}$$

$$y_j^{(1)} = f\left(net_j^{(1)}\right), \quad j = 1, \ldots, J \tag{8}$$

where ψ_{ji} represents the weights between the input layer and the hidden layer (1), T_j represents the threshold value for each node, J is the number of the nodes, $y_j^{(1)}$ is the output of each node, and f is a nonlinear activation function. In this paper, the hyperbolic tangent function is used and it ranges on the interval [− 1, 1], which is defined as:

$$f(W) = \tanh(W) \tag{9}$$

and its derivative can be obtained by $d[f(W)]/dy = 1 - f^2(W)$.

Hidden layer (2): each node is determined as the following:

$$net_m^{(2)} = \sum_{j=1}^{J} \psi_{mj}\, y_j^{(1)} + T_m\,, \quad m = 1, \ldots, M \tag{10}$$

$$y_m^{(2)} = f\left(net_m^{(2)}\right), \quad m = 1, \ldots, M \tag{11}$$

where ψ_{mj} represents the weights between the hidden layer (1) and the hidden layer (2), T_m represents the threshold value for each node in this layer, M is the number of the nodes in this layer, and $y_m^{(2)}$ represents the output of each node.

Output layer: its output is calculated as:

$$u = \sum_{m=1}^{M} \psi_m\, y_m^{(2)} + T \tag{12}$$

where u represents the output of the network, ψ_m represents the weights between the hidden layer (2) and the output layer, and T represents the threshold value.

The NN is trained to minimize the error between the reference input and the measured output [46, 47]. The square of the error is defined as:

$$E_g(\hbar) = \frac{1}{2}\left(y_d(\hbar) - y_a(\hbar)\right)^2 = \frac{1}{2} e_g^2(\hbar) \tag{13}$$

where $y_d(\hbar)$ represents the reference input and $y_a(\hbar)$ represents the actual output.

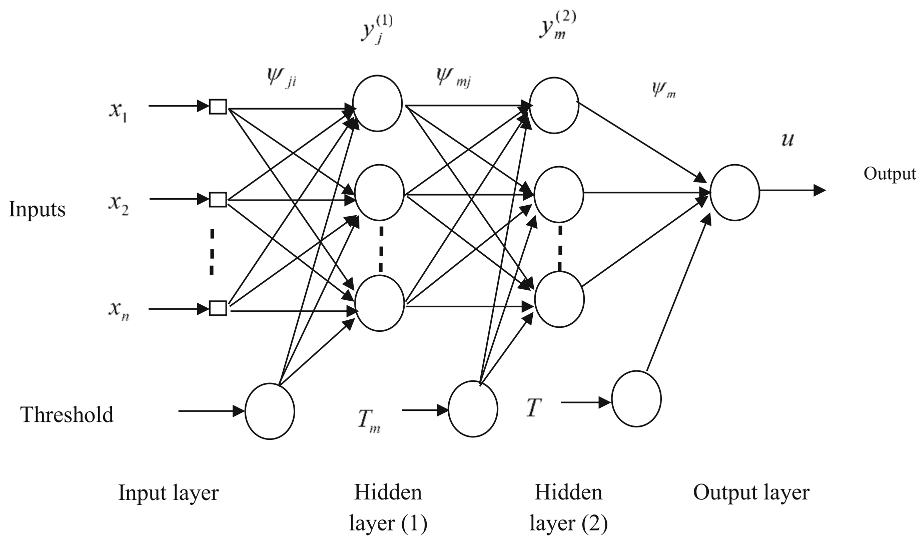


Fig. 2 Feed-forward neural network

3.2 Weights learning based on Lyapunov stability

A scalar function $V_x(\hbar)$ is chosen as a positive definite function for all initial conditions of its arguments [48, 49]. Now, if the two conditions, which are described in Eqs. (14 and 15), are get together, the given system is considered asymptotic stable.

$$V_x(\hbar) > 0 \text{for all } \hbar \text{ except } \hbar = 0 \tag{14}$$

$$\Delta V_x(\hbar) = V_x(\hbar+1) - V_x(\hbar) \le 0 \tag{15}$$

To put the basis of the weight learning algorithm, the weight update equation in a general form can be expressed as:

$$\Psi_g(\hbar+1) = \Psi_g(\hbar) - \eta \Delta\Psi_g(\hbar) \tag{16}$$

where $\Psi_g(\hbar)$ is the generalized weight vector and η is the learning rate. $\Delta\Psi_g(B)$ indicates the desired weights modification.

Theorem 1 *The parameters of the FFNNDLC, which warranty the stability, are obtained based on the following*:

$$\Psi_g(\hbar+1) = \Psi_g(\hbar) + \eta \frac{\beta\left(\frac{\alpha}{\beta} e_g(\hbar) \frac{\partial e_g(\hbar)}{\partial \Psi_g(\hbar)} + \Psi_g(\hbar)\right)}{2\delta} \tag{17}$$

Proof Assume the next Lyapunov function:

$$V_x(\hbar) = V_a(\hbar) + V_b(\hbar) + V_c(\hbar) \tag{18}$$

where $V_a(\hbar) = \frac{\alpha}{2}\left(e_g(\hbar)\right)^2, V_b(\hbar) = \frac{\beta}{2}\left(\Psi_g(\hbar)\right)^2, V_c(\hbar) = \frac{\delta}{2}\left(\Delta\Psi_g(\hbar)\right)^2, \alpha, \beta$ and δ are constants. $\Delta V_a(\hbar)$, $\Delta V_b(\hbar)$ and $\Delta V_c(\hbar)$ are defined as:

$$\begin{aligned}\Delta V_a(\hbar) &= V_a(\hbar+1) - V_a(\hbar)\\ &= \frac{\alpha}{2}\left(e_g(\hbar+1)\right)^2 - \frac{\alpha}{2}\left(e_g(\hbar)\right)^2\end{aligned} \tag{19}$$

$$\begin{aligned}\Delta V_b(\hbar) &= V_b(\hbar+1) - V_b(\hbar)\\ &= \frac{\beta}{2}\left(\Psi_g(\hbar+1)\right)^2 - \frac{\beta}{2}\left(\Psi_g(\hbar)\right)^2\end{aligned} \tag{20}$$

$$\begin{aligned}\Delta V_c(\hbar) &= V_c(\hbar+1) - V_c(\hbar)\\ &= \frac{\delta}{2}\left(\Delta\Psi_g(\hbar+1)\right)^2 - \frac{\delta}{2}\left(\Delta\Psi_g(\hbar)\right)^2\end{aligned} \tag{21}$$

The term $\frac{\alpha}{2}\left(e_g(\hbar+1)\right)^2$ can be formulated based on the Taylor series as [2, 6]:

$$\begin{aligned}\frac{\alpha}{2}\left(e_g(\hbar+1)\right)^2 &= \frac{\alpha}{2}\left(e_g(\hbar)\right)^2 + \frac{\partial\left(\frac{\alpha}{2}\left(e_g(\hbar)\right)^2\right)}{\partial\Psi_g(\hbar)}\Delta\Psi_g(\hbar)\\ &\quad + \text{higher order terms } (HOT)\end{aligned} \tag{22}$$

where HOT can be ignored. Therefore, Eq. (22) can be reformulated as:

$$\frac{\alpha}{2}\left(e_g(\hbar+1)\right)^2 - \frac{\alpha}{2}\left(e_g(\hbar)\right)^2 = \alpha e_g(\hbar)\frac{\partial\left(e_g(\hbar)\right)}{\partial\Psi_g(\hbar)}\Delta\Psi_g(\hbar) \tag{23}$$

Similarity,

$$e_g(\hbar+1) = e_g(\hbar) + \frac{\partial e_g(\hbar)}{\partial\Psi_g(\hbar)}\Delta\Psi_g(\hbar) \tag{24}$$

Equation (24) can be reformulated as:

$$e_g(\hbar+1) - e_g(\hbar) = \Delta e_g(\hbar) = \frac{\partial e_g(\hbar)}{\partial\Psi_g(\hbar)}\Delta\Psi_g(\hbar) \tag{25}$$

Then, by replacing $\frac{\partial e_g(\hbar)}{\partial\Psi_g(\hbar)}\Delta\Psi_g(\hbar)$ in Eq. (23), we obtain

$$\Delta V_a(\hbar) = \frac{\alpha}{2}\left(e_g(\hbar+1)\right)^2 - \frac{\alpha}{2}\left(e_g(\hbar)\right)^2 = \alpha\, e_g(\hbar)\,\Delta e_g(\hbar) \tag{26}$$

Similarity, $\Delta V_b(\hbar) = \beta\,\Psi_g(\hbar)\Delta\Psi_g(\hbar)$ and $\Delta V_c = \delta\left(\Delta\Psi_g(\hbar)\right)^2$

The second stability condition is obtained as:

$$\begin{aligned}\Delta V_x(\hbar) &= \alpha\, e_g(\hbar)\,\Delta e_g(\hbar) + \beta\,\Psi_g(\hbar)\,\Delta\Psi_g(\hbar)\\ &\quad + \delta\left(\Delta\Psi_g(\hbar)\right)^2 \le 0\end{aligned} \tag{27}$$

Equation (27) can be reformulated as:

$$\begin{aligned}\Delta V_x(\hbar) &= \alpha\, e_g(\hbar)\Delta e_g(\hbar) + \beta\,\Psi_g(\hbar)\,\Delta\Psi_g(\hbar)\\ &\quad + \delta\left(\Delta\Psi_g(\hbar)\right)^2\\ &= -Z\end{aligned} \tag{28}$$

where $Z \ge 0$ so as to achieve the condition, $\Delta V_x(\hbar) \le 0$

So

$$\begin{aligned}&\delta\left(\Delta\Psi_g(\hbar)\right)^2 + \beta\,\Delta\Psi_g(\hbar)\left(\frac{\alpha}{\beta}\, e_g(\hbar)\frac{\Delta e_g(\hbar)}{\Delta\Psi_g(\hbar)} + \Psi_g(\hbar)\right) + Z\\ &= 0\end{aligned} \tag{29}$$

In this concern, suppose a general quadratic equation, which is defined as:

$$a\,X^2 + b\,X + c = 0 \tag{30}$$

The roots of Eq. (30) are calculated as:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{31}$$

From Eqs. (29) and (30), obviously, $\Delta\Psi_g(\hbar)$ acts as X in Eq. (30) and the values of a, b and c in Eq. (29) are obtained as:

$$a = \delta, \quad b = \beta\left(\frac{\alpha}{\beta}e_g(\hbar)\frac{\Delta e_g(\hbar)}{\Delta\Psi_g(\hbar)} + \Psi_g(\hbar)\right) \quad \text{and} \quad c = Z \tag{32}$$

Equation (30) has a single unique solution, if $\sqrt{b^2 - 4ac} = 0$. So,

$$\beta^2\left(\frac{\alpha}{\beta}e_g(\hbar)\frac{\Delta e_g(\hbar)}{\Delta\Psi_g(\hbar)} + \Psi_g(\hbar)\right)^2 - 4\delta Z = 0 \tag{33}$$

and therefore, Z is determined as:

$$Z = \frac{\beta^2\left(\frac{\alpha}{\beta}e_g(\hbar)\frac{\Delta e_g(\hbar)}{\Delta\Psi_g(\hbar)} + \Psi_g(\hbar)\right)^2}{4\delta} \tag{34}$$

Since $Z \geq 0$ which means

$$\frac{\beta^2\left(\frac{\alpha}{\beta}e_g(\hbar)\frac{\Delta e_g(\hbar)}{\Delta\Psi_g(\hbar)} + \Psi_g(\hbar)\right)^2}{4\delta} \geq 0 \tag{35}$$

So, the unique root of Eq. (29) is $x_{1,2} = \frac{-b}{2a}$; similarly,

$$\Delta\Psi_g(\hbar) = -\frac{\beta\left(\frac{\alpha}{\beta}e_g(\hbar)\frac{\Delta e_g(\hbar)}{\Delta\Psi_g(\hbar)} + \Psi_g(\hbar)\right)}{2\delta} \tag{36}$$

Equation (36) can be reformulated as:

$$\Delta\Psi_g(\hbar) = -\frac{\beta\left(\frac{\alpha}{\beta}e_g(\hbar)\frac{\partial e_g(\hbar)}{\partial\Psi_g(\hbar)} + \Psi_g(\hbar)\right)}{2\delta} \tag{37}$$

So, by replacing $\Delta\Psi_g(\hbar)$ in Eq. (16), the updating equation is obtained as illustrated in Eq. (17).

4 FFNNDL controller training steps

In this section, the block diagram of the proposed FFNNDLC for nonlinear system is shown in Fig. 3. The output of the nonlinear system, $y_a(\hbar)$, is measured, and then the error signal, $e_g(\hbar)$, between the reference input, $y_d(\hbar)$, and the measured output is calculated. The proposed FFNNDLC is received the error signal, and it calculates the control signal, $u(\hbar)$, which feeds to the nonlinear system.

Figure 4 shows the structure of the proposed FFNNDLC block in details. The first layer of the controller network consists of three nodes; the first node is the measured error signal, $e_1(\hbar) = e_g(\hbar)$. The second node is the change of the error signal, $e_2(\hbar) = e_g(\hbar) - e_g(\hbar - 1)$, and the third node is the change of the change of error signal, $e_3(\hbar) = e_g(\hbar) - 2e_g(\hbar - 1) + e_g(\hbar - 2)$, and the output layer consists of one node, which is the control signal, $u(\hbar)$. The proposed network consists of two parts: the first part is the feed-forward neural network and the second part is the RBM, which is used to perform the initial values for the weights of the feed-forward neural network as shown in Fig. 4. The overall procedures of the proposed FFNNDLC are illustrated as:

Step 1 Enter the reference input for the nonlinear system and calculate the controller inputs, $e_1(\hbar)$, $e_2(\hbar)$ and $e_3(\hbar)$. In this paper, we set, $n = 3$, $J = 10$ and $M = 10$.
Step 2 Choose the number of RBM weights, which are exactly like the weights between the input layer and the hidden layer (1). In this paper, we set $N = 3$, $P = 10$ and $q = 1$.
Step 3 Choose the appropriate value for the learning rate of RBM, α. Initialize the weights parameters of the RBM with zero. The number of the RBM inputs equals the number of the controller inputs.
Step 4 According to Eqs. (4), (5) and (6), update the RBMs weights parameters.
Step 5 Set the initial values of the FFNN weights matrix; ψ_{ji} (between input layer and hidden layer (1)) as the values of the RBM weights matrix.
Step 6 Calculate the outputs of the hidden layer (1); $y_j^{(1)}$.
Step 7 Call RBM again where the number of the RBM weights is exactly like the weights between the hidden layer (1) and the hidden layer (2). In this paper, we set $N = 10$ and $P = 10$. The number of the RBM inputs equals the number of the outputs of the hidden layer (1); $y_j^{(1)}$.
Step 8 Update the RBMs weights parameters based on Eqs. (4), (5) and (6).
Step 9 Set the initial values of the FFNN weights matrix; ψ_{mj} (between the hidden layer (1) and the hidden layer (2)) as the values of the RBM weights matrix.
Step 10 Calculate the outputs of the hidden layer (2); $y_m^{(2)}$.
Step 11 Call the RBM again where the number of the RBM weights is exactly like the weights between the hidden layer (2) and the output layer. In this paper, we

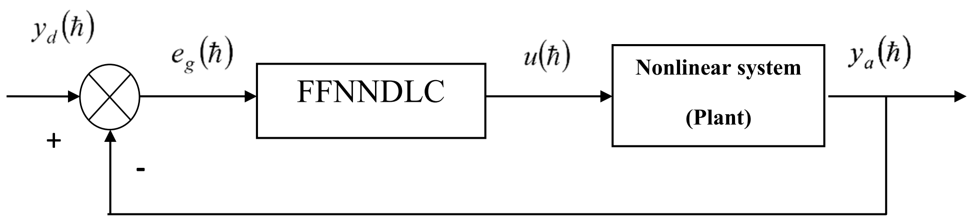
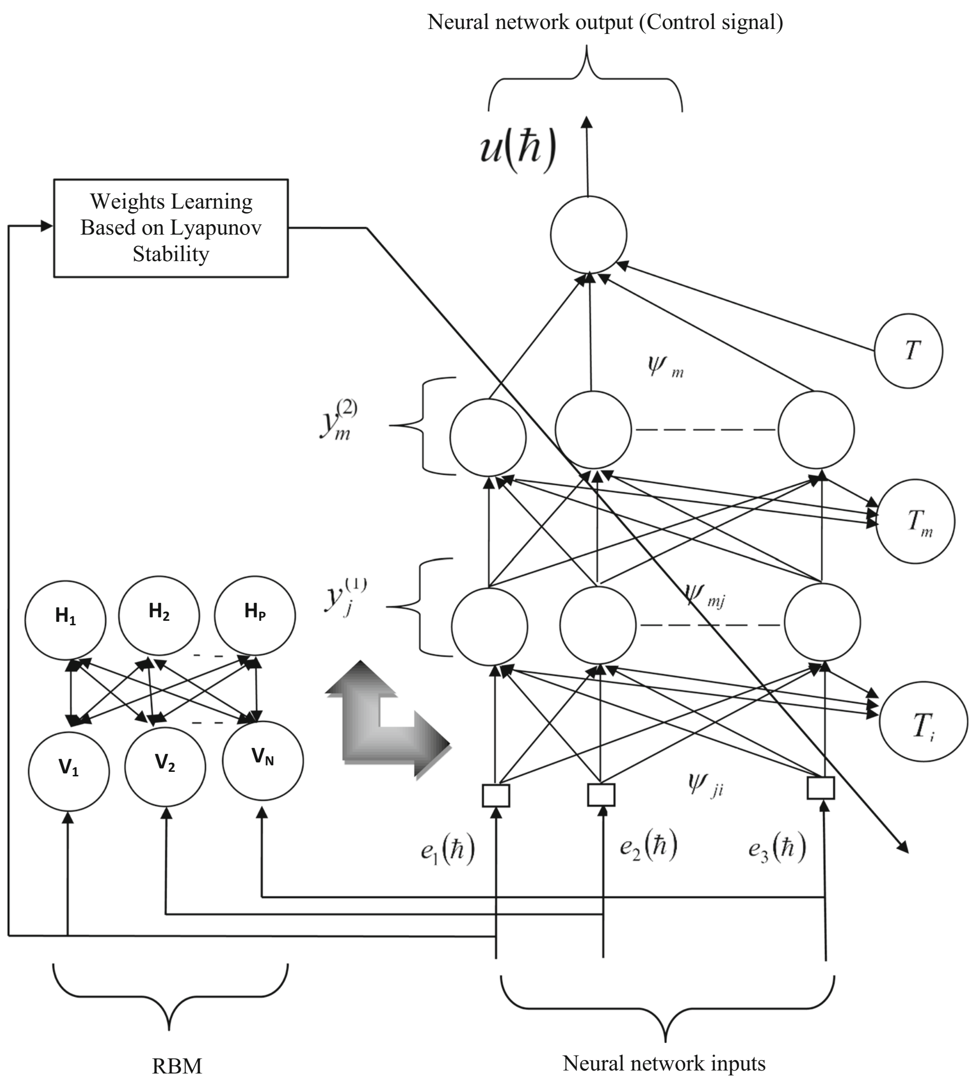


Fig. 3 Block diagram of the FFNNDLC with nonlinear system

Fig. 4 Structure of the FFNNDLC block



set $N = 10$ and $P = 1$. The number of the RBM inputs equals the number of the hidden layer (2); $y_m^{(2)}$.
Step 12 Update the RBMs weights parameters based on Eqs. (4), (5) and (6).
Step 13 Set the initial values for the FFNN weights matrix; ψ_m (between the hidden layer (2) and the output layer) as the values of the RBM weights matrix.
Step 14 Calculate the control signal; $u(\hbar)$.
Step 15 Update the FFNN weights parameters according to the Lyapunov Stability formula, Eq. (17). After then, set the values for the RBM weights matrix as the values of the FFNN weights matrix.
Step 16 Calculate the plant output;$y_a(\hbar)$.
Step 17 Go to step 1.

5 Simulation Results

The simulation results for the proposed FFNNDLC are compared with the results of the FFNNC with the same conditions and initial values to show the robustness of the proposed controller. Mean absolute error (MAE) and root-mean-square error (RMSE) are used to evaluate the performance of the proposed controller. RMSE and MAE are calculated as [50, 51]:

$$MAE = \frac{1}{k_N}\sum_{\hbar=1}^{k_N}\left|e_g(\hbar)\right| \tag{38}$$

$$RMSE = \sqrt{\left(\frac{1}{k_N}\sum_{\hbar=1}^{k_N}\left(e_g(\hbar)\right)^2\right)} \tag{39}$$

where k_N represents iterations number.

5.1 Example 1: Suppose the following mathematical system that is presented as [52]

$$y_a(\hbar) = \frac{y_a(\hbar-1)y_a(\hbar-2)(y_a(\hbar-1)+a_4)}{a_1 + a_2 y_a^2(\hbar-1) + a_3 y_a^2(\hbar-2)} + u(\hbar) \quad (40)$$

where the parameters are set as $a_1 = a_2 = a_3 = 1$ and $a_4 = 0.05$.

5.2 Test 1: Tracking the reference signal trajectory

Figures (5 and 6) show the system response and its control signal for tracking the reference signal trajectory, respectively, for both FFNNC and the proposed FFNNDLC. In this concern, the reference trajectory signal is illustrated as:

$$y_d(\hbar) = 0.5\sin(0.1\hbar T) \quad (41)$$

where T is sampling period.

The FFNNC and the proposed FFNNDLC weights are considered have zero values. It is clear at the beginning that there is a difference in tracking the signal in terms of the FFNNC. On the other hand, the proposed FFNNDLC can overcome the signal tracking as a result of learning from RBM, better than FFNNC.

5.3 Test 2: Uncertainty due to disturbance

During the present test, the robustness of the proposed FFNNDLC is tested after the system output has been reached the trajectory. This carried out by adding disturbance value equals 50% of its desired value to the system output at $\hbar$ = 3000 instant. Figures 7 and 8 show the system output and the control signal, respectively, for both the FFNNC and the proposed FFNNDLC with 50% disturbances. Also, another test is carried out, at 80% disturbances, to ensure the robustness of FFNNDLC (Figs. 9 and 10). From which, it is shown that the response of the proposed FFNNDLC is recovered very quickly to the desired value. So, the proposed FFNNDLC is able to respond the external disturbance compared with the FFNNC.

5.4 Test 3: Uncertainty in the system parameter

To show the robustness of the proposed FFNNDLC, the parameters (a_1, a_2, a_3 and a_4) of the given plant are changed from their actual values (1, 1, 1 and 0.05) to (-3, 2.5, 1.5, 0.5) at $\hbar$ = 3000th instant. The system output for the proposed FFNNDLC is recovered very quickly rather than the FFNNC as shown in Figs. (11 and 12). So, the proposed FFNNDLC is able to reduce the effect of parameters uncertainty compared with FFNNC.

5.5 Test 4: Uncertainty due to disturbance and parameter variation

Figures (13 and 14) show the effect of uncertainty due to disturbance and parameter variation to show the robustness of the proposed controller. At $\hbar$ = 3000th instant, the disturbance with 50% from the reference input and the parameters variation are added to the system output. The parameters (a_1, a_2, a_3 and a_4) are varied to values (-3, 2.5, 1.5, 0.5). The FFNNDLC response is able to respond the effect of the uncertainty due to the disturbance and system parameters variation.

The MAE and RMSE values for the proposed FFNNDLC, the FFNNC and other published schemes such as DRNNC [16], FFNNC with a hybrid method (FFNNC-hybrid) [18] and ENN-RBM [29] are shown in Tables 1 and 2. It is obvious that MAE and RMSE values for the proposed FFNNDLC are smaller than that are obtained for other controllers. Therefore, the proposed FFNNDLC, which uses the RBM network to initialize the values of the FFNN weights, is able to reduce the effect of external disturbance and system uncertainties compared with other schemes.

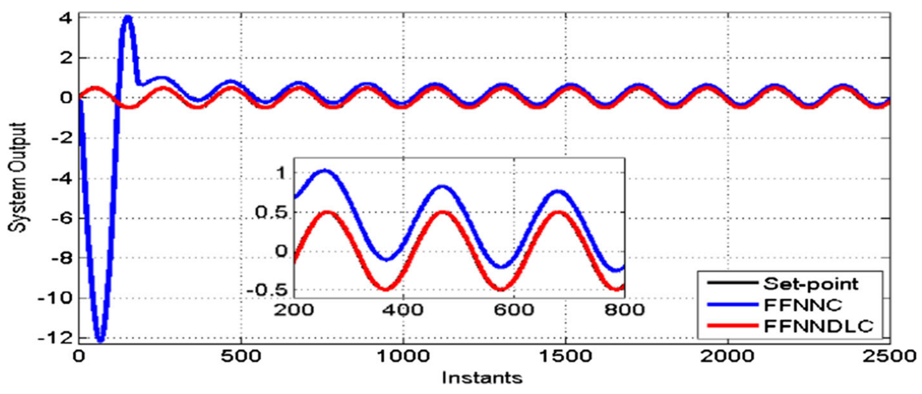


Fig. 5 System output for Test 1

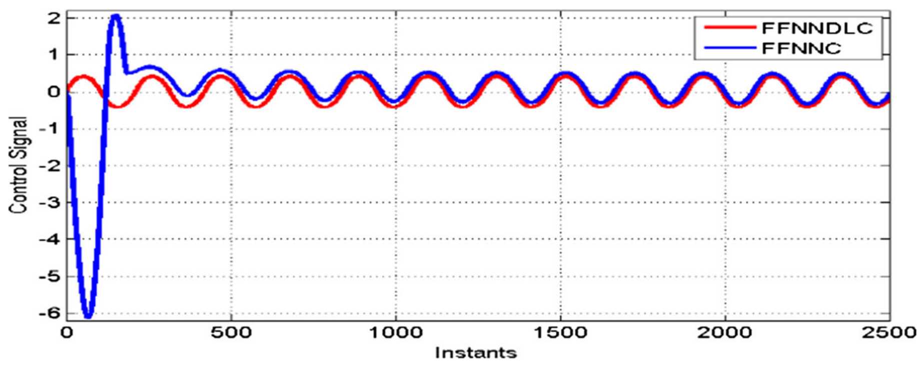


Fig. 6 Control signal for both controllers (Test 1)

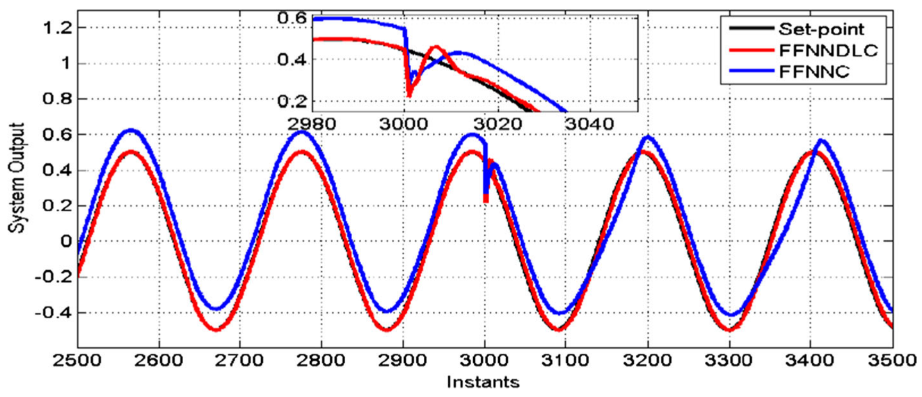


Fig. 7 System output under 50% disturbances

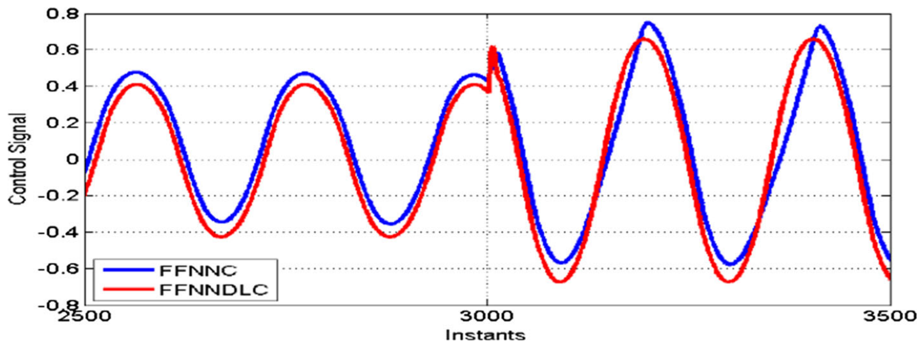


Fig. 8 Control signal for response of the system under 50% disturbances (Test 2)

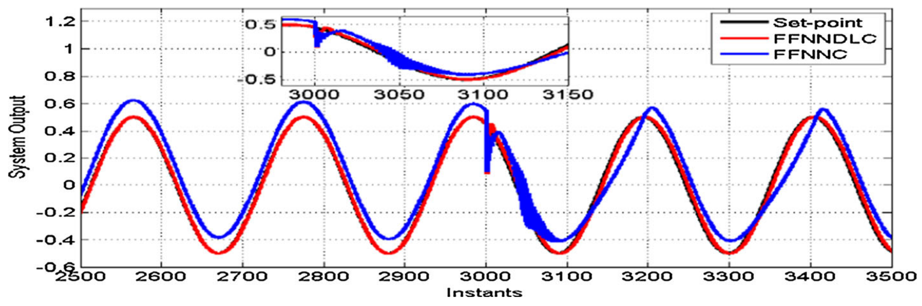


Fig. 9 System output under 80% disturbances (Test 2)

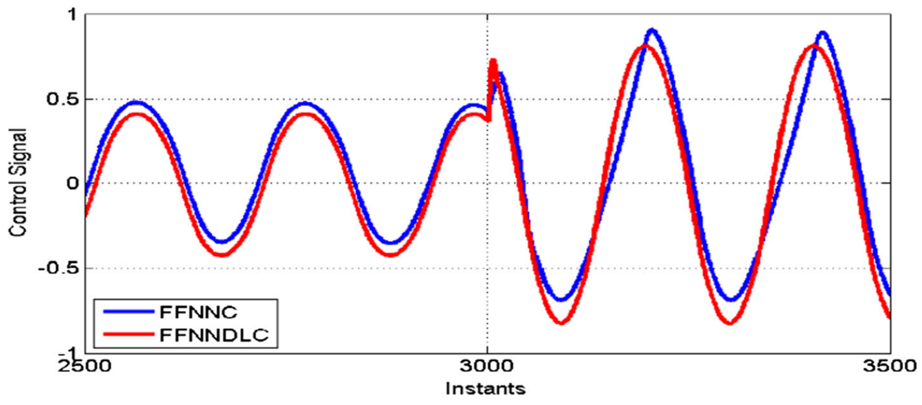


Fig. 10 Control signal for the system response under 80% disturbances (Test 2)

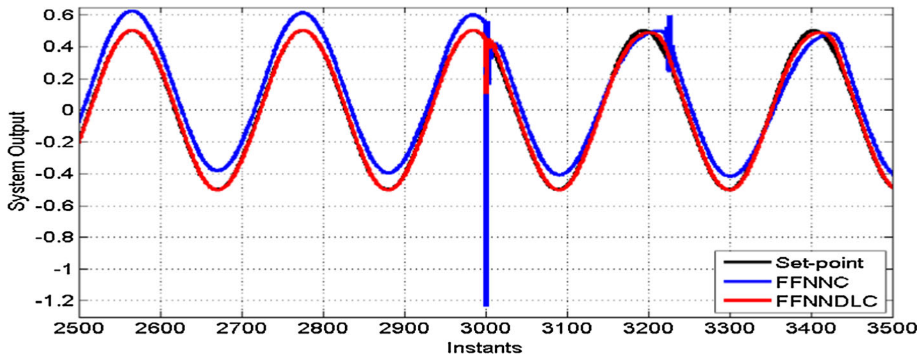


Fig. 11 System output under uncertainty of the system parameters (Test 3)

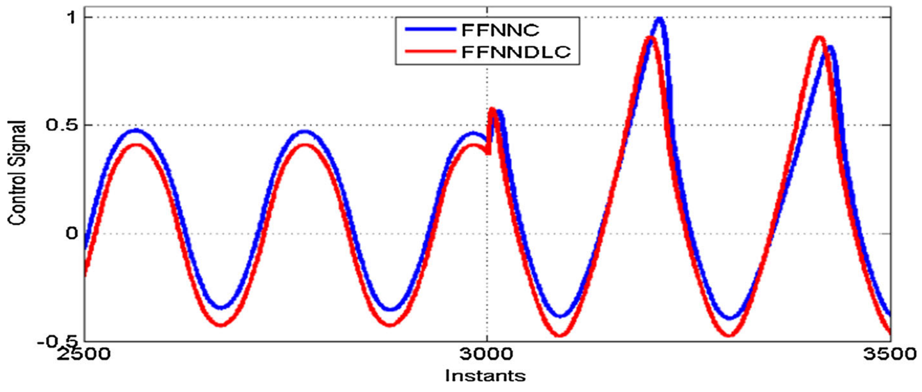


Fig. 12 Control signal for the system response (Test 3)

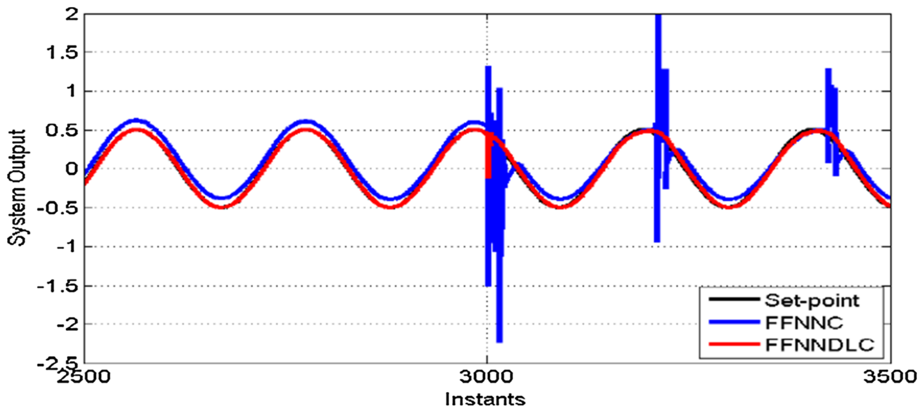


Fig. 13 System output under uncertainty of the system parameters and 50% disturbances (Test 4)

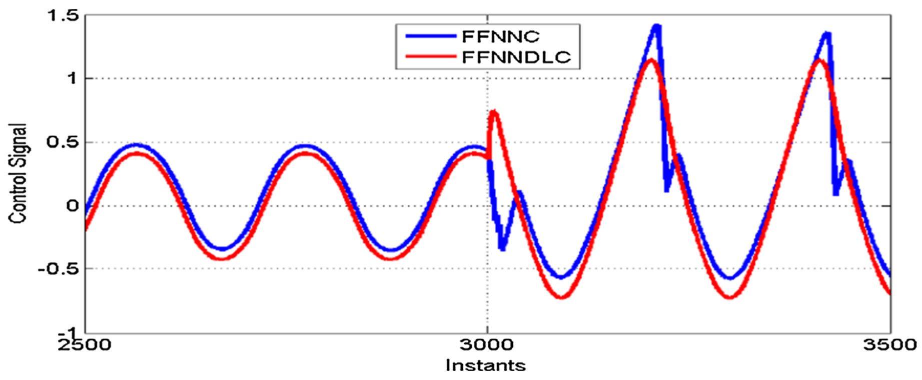


Fig. 14 Control signal for the system response under uncertainty of the system parameters and 50% disturbances (Test 4)

Table 1 MAE values (Example 1)

	Test 1	Test 2		Test 3	Test 4
		50% Disturbance	80% Disturbance		
FFNNC	0.5034	0.5062	0.5070	0.5033	0.5103
DRNNC [16]	0.5227	0.5258	0.5274	0.5226	0.5271
FFNNC-hybrid [18]	0.1762	0.1827	0.1859	0.1865	0.1995
ENN-RBM [29]	0.3592	0.3615	0.3629	0.3611	0.4050
FFNNDLC	0.0150	0.0167	0.0175	0.0166	0.0180

Table 2 RMSE values (Example 1)

	Test 1	Test 2		Test 3	Test 4
		50% Disturbance	80% Disturbance		
FFNNC	1.7156	1.7158	1.7159	1.7158	1.7185
DRNNC [16]	1.7385	1.7189	1.7391	1.7387	1.7408
FFNNC-hybrid [18]	0.6306	0.6321	0.6330	0.6362	0.6574
ENN-RBM [29]	1.3329	1.3330	1.3331	1.3330	1.4175
FFNNDLC	0.0176	0.0202	0.0220	0.0203	0.0236

5.6 Example 2: In this test, the following mathematical system, which is presented by the three sub-systems, is given as [53]

The first sub-system: for $0 < \hbar < 3000$

$$y_a(\hbar) = 0.1 y_a(\hbar - 1) + \frac{u(\hbar)(0.5 + u^2(\hbar))}{1 + 2u^2(\hbar)} \tag{42}$$

The second sub-system: for $3000 \le \hbar < 6000$

$$y_a(\hbar) = \frac{[y_a(\hbar-1)y_a(\hbar-2)y_a(\hbar-3)u(\hbar-1)][y_a(\hbar-3) - b(\hbar)] + c(\hbar)u(\hbar)}{a(\hbar) + y_a^2(\hbar-2) + y_a^2(\hbar-3)} \tag{43}$$

The time-varying parameters $a(\hbar)$, $b(\hbar)$ and $c(\hbar)$ in the previous equation are defined as: $a(\hbar) = 1.2 - 0.2\cos(0.1\hbar T)$, $b(\hbar) = 1 - 0.4\sin(0.1\hbar T)$ and $c(\hbar) = 1 + 0.4\sin(0.1\hbar T)$.

The third sub-system: for $6000 \le \hbar < 9000$

$$\begin{aligned} y_a(\hbar) &= 0.2\, y_a^2(\hbar-1) + 0.2\, y_a(\hbar-2) \\ &+ 0.4\ \sin\ [0.5\, y_a(\hbar-1) + 0.5\, y_a(\hbar-2)] \\ &.\ \cos\ [0.5\, y_a(\hbar-1) + 0.5\, y_a(\hbar-2)] + 1.2\, u(\hbar) \end{aligned} \tag{44}$$

The system output and the control signal for the proposed FFNNDLC and the FFNNC for the three sub-systems are shown in Figs. (15, 16, 17, 18). It is obvious that the response of FFNNDLC is reached to the desired output faster than the FFNNC during the change from sub-system to another. The MAE and RMSE values for the proposed controller and other controllers are shown in Table 3.

The main advantages of the proposed FFNNDLC are summarized as follows: 1) It is a fast-learning controller because it uses RBM network, which initializes the values of the FFNN. 2) It has the ability to handle system uncertainties and external disturbance due to the online learning for the proposed scheme based in the Lyapunov stability theorem. In the next section, the proposed scheme is applied practically to control a real plant.

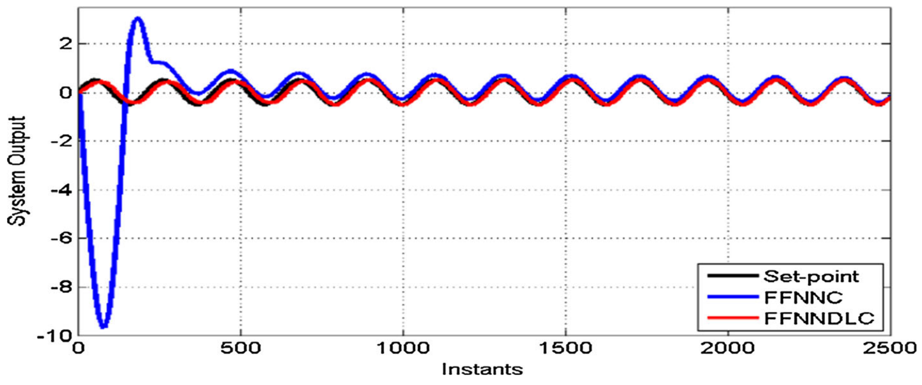


Fig. 15 First sub-system output

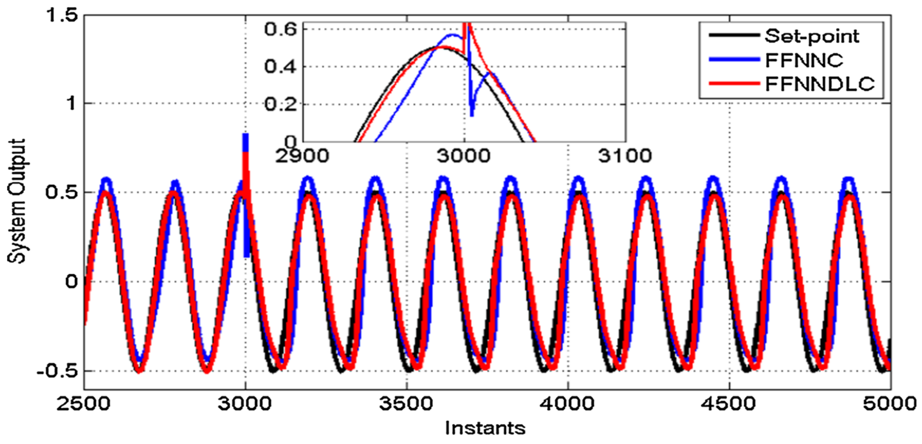


Fig. 16 Second sub-system output

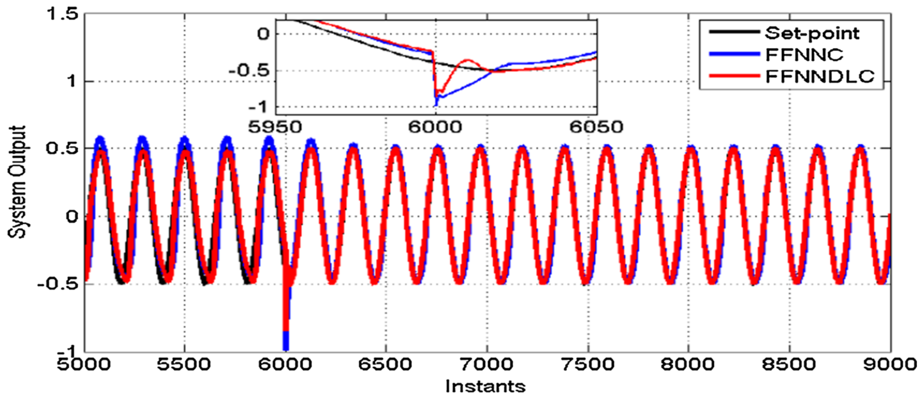


Fig. 17 Third sub-system output

6 Practical results

In this section, the proposed FFNNDLC is implemented practically to show the ability of the proposed controller to deal with a real system. Figure 19 shows the practical system. It is the DC machine, which consists of a DC motor that is coupled with a DC generator to test the load conditions. The feedback signal, which is the measuring speed, is done based on an optical encoder. The proposed FFNNDLC algorithm is implemented in ARDUINO DUE kit, which is a microcontroller board based on Atmel SAM3X8E ARM Cortex-M3 CPU. To show the ability of the proposed FFNNDLC in real system, different tests are performed.

6.1 Test 1: Tracking the different reference signal trajectory

In this test, different set points are considered. Figure 20 shows the system response and its control signal for tracking the different reference signal trajectory for the proposed FFNNDLC and the FFNNC when all the initial conditions for the weights equal zero for both controllers. It is clear that the response of the proposed FFNNDLC has

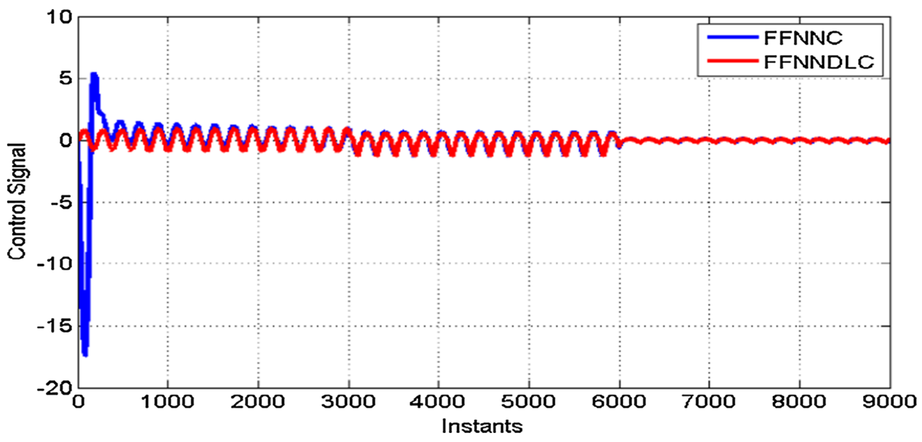


Fig. 18 Control signal for the three sub-systems

Table 3 MAE and RMSE values for both controllers

	MAE	RMSE
FFNNC	0.2397	0.9397
DRNNC [16]	0.2526	0.9411
FFNNC-hybrid [18]	0.1275	0.3163
ENN-RBM [29]	0.17824	0.80335
FFNNDLC	0.0626	0.0918

fast tracking for the reference signal compared with FFNNC due to the advantage of the RBM, which is used to initialize the proposed network. To show the visual indications and robustness of the control performance, an objective measure of an error performance is done using MAE as shown in Fig. 21. It is clear that the MAE for proposed FFNNDLC is lower than that obtained for the FFNNC.

6.2 Test 2: Uncertainty due to load

In this test, a load value equals 35% of its desired value is applied to the system output at $time = 18\ \text{sec}$. Figure 22 shows the system output and its control signal, for both the FFNNC and the proposed FFNNDLC due to the load. It is clear that the output response for the proposed FFNNDLC has smaller settling time than that is obtained for the FFNNC. Figure 23 shows the MAE for both controllers where the MAE for the proposed controller is lower than that is obtained for another controller. So the proposed FFNNDLC is better than the FFNNC, which has fast learned to overcome quickly the effect of uncertainty due to the external load.

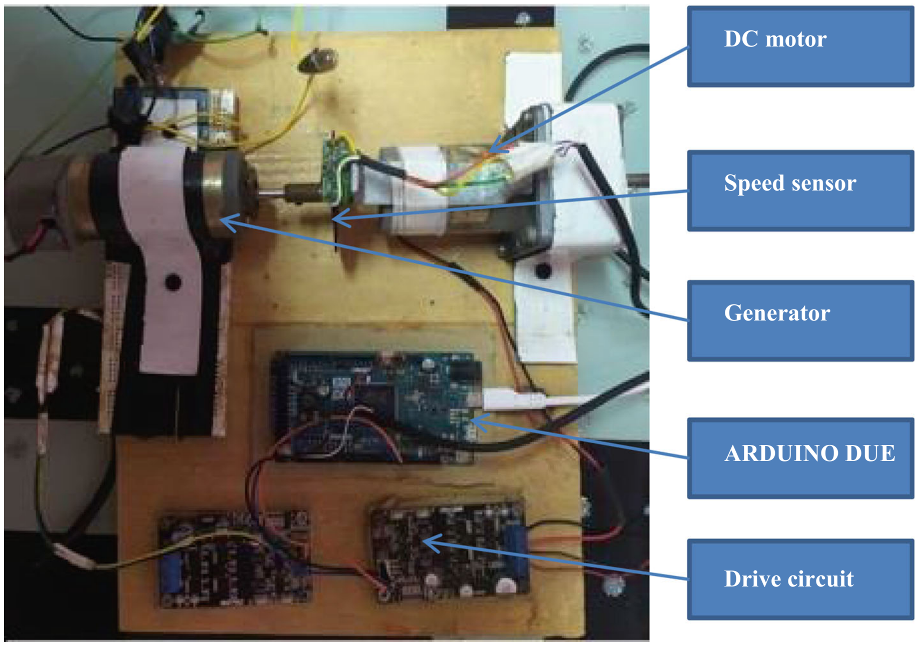


Fig. 19 Practical system

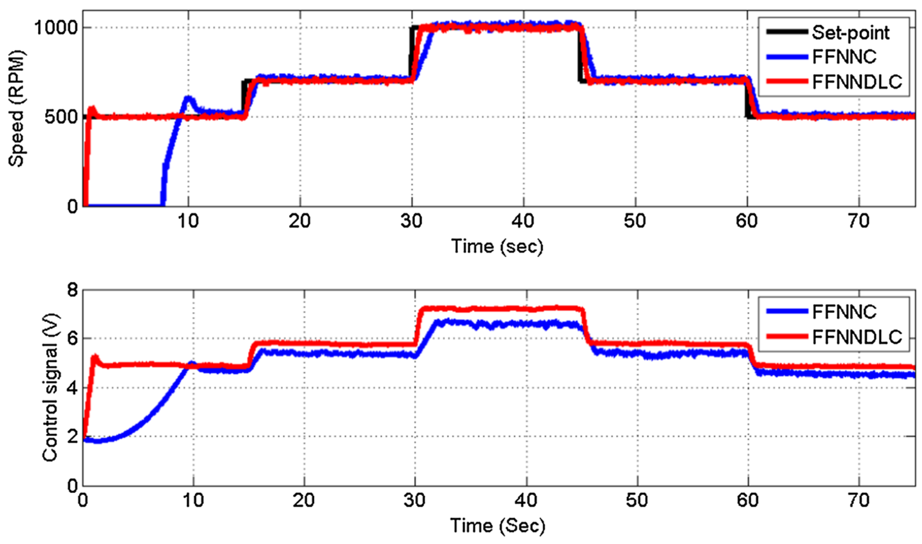


Fig. 20 System response and its control signal for tracking the different reference signal

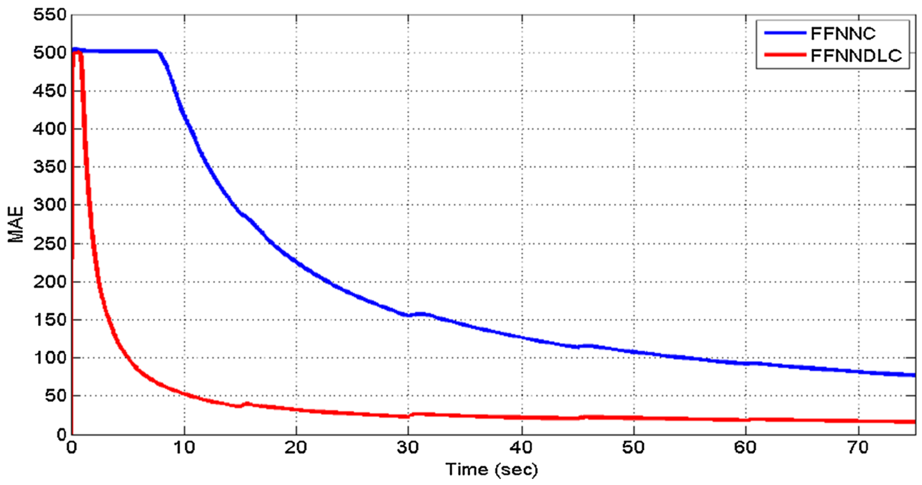


Fig. 21 MAE for tracking the different reference signal

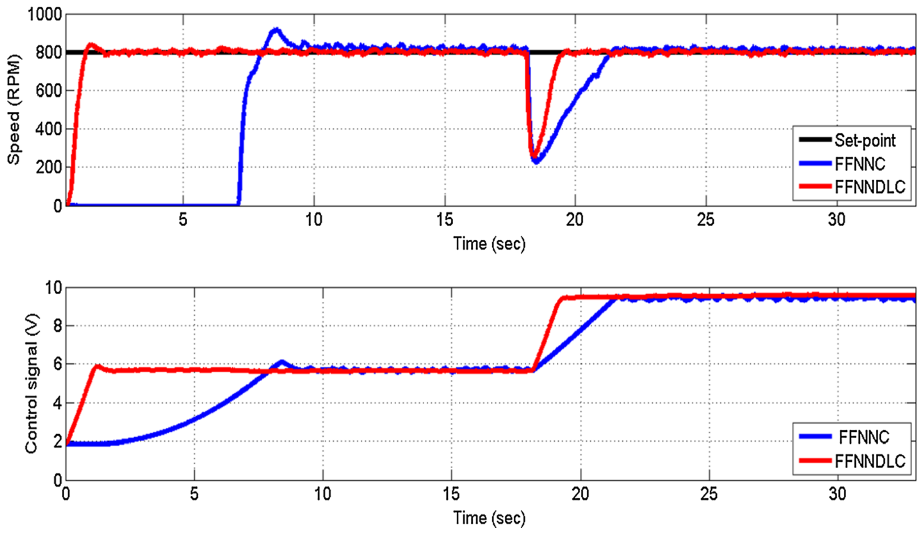


Fig. 22 System response and its control signal due to 35% load

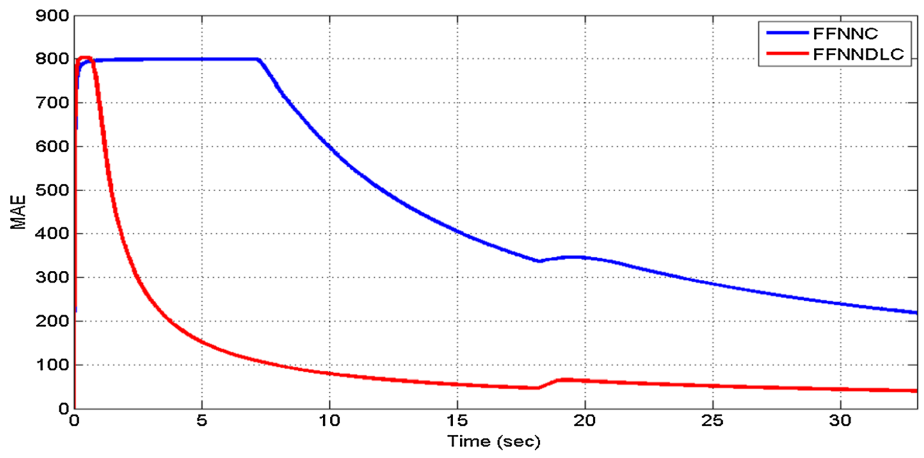


Fig. 23 MAE for controlled system due to 35% load

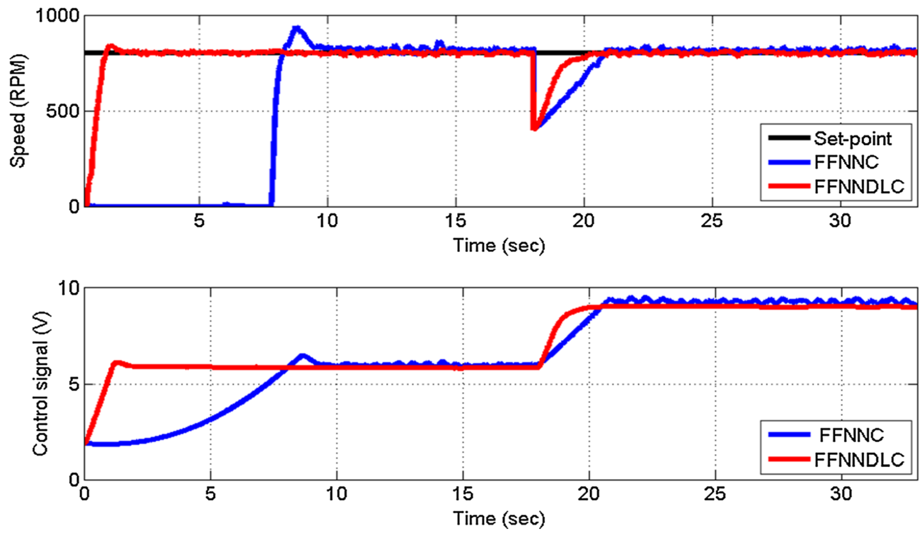


Fig. 24 System response and its control signal due to 50% uncertainty in measurement error

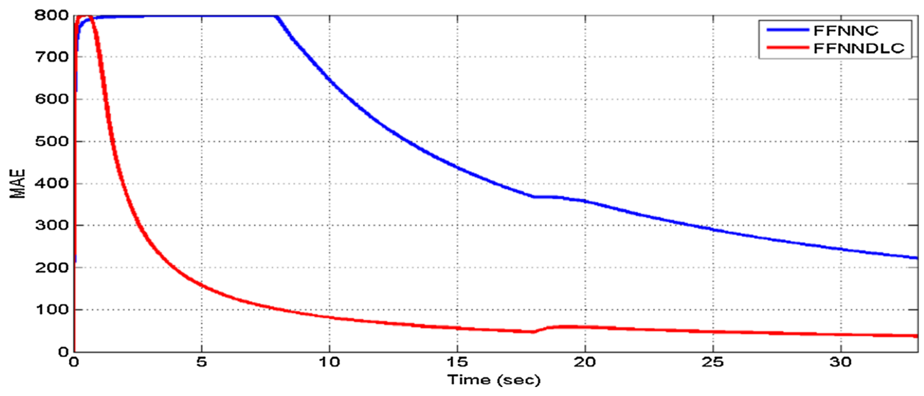


Fig. 25 MAE for controlled system due to 50% uncertainty in measurement error

6.3 Test 3: Uncertainty due to the measurement error

During this test, the sensor reading error value is considered with 50% of its value to the system output at $time = 18$ sec. Figure 24 shows the system output and its control signal, for both the FFNNC and the proposed FFNNDLC due to uncertainty in the measurement error. Figure 25 shows that the MAE for the proposed FFNNDLC is less than that is obtained for another controller. So, the proposed FFNNDLC is better than the FFNNC, which has fast learned to overcome quickly the effect of uncertainty due to the measurement error.

7 Conclusions

In this paper, the FFNNDLC is proposed, which consists of two parts. The first part is the FFNN which is the main network, and the other part is the RBM network, which is used to initialize the values of the FFNN weights. The weights of the proposed FFNNDLC are learned online based on the developed algorithm, which is proved based on the Lyapunov stability theorem to guarantee the controller stability. To show the robustness of the proposed network, it is applied to two nonlinear mathematical models. The performance of the proposed controller is tested when there is uncertainty in the controlled system due to the parameters variations and external disturbances. To show the advantages of the proposed controller, it is compared with other published schemes by measuring some performance indices such as MAE and RMSE. The simulation results show that the performance of the proposed FFNNDLC has fast-learning algorithm compared with other schemes.

In order to show the ability of the proposed controller to deal with a real system, it is implemented practically using an ARDUINO DUE kit for controlling a DC motor-generator system. The practical results show that the proposed controller is able to respond the effect of system uncertainty due to the measurement error and external disturbances. Thus, the methodology proposed in this study can be used to realize a robust, practically realizable, FFNNDLC capable of controlling a real system with system uncertainties and environmental disturbances.

Finally, the proposed FFNNDLC has fast-learning controller because it uses RBM network, which initializes the values of the FFNN. Furthermore, it has the ability to handle system uncertainties and external disturbance due to the online learning for the proposed scheme based on the Lyapunov stability theorem.

Compliance with ethical standards

Conflict of interest There is no conflict of interest between the authors to publish this manuscript.

References

1. Khater AA, El-Bardini M, El-Rabaie NM (2015) Embedded adaptive fuzzy controller based on reinforcement learning for dc motor with flexible shaft. Arab J Sci Eng 40:2389–2406
2. Kumar R, Srivastava S, Gupta JRP, Mohindru A (2018) Diagonal recurrent neural network based identification of nonlinear dynamical systems with Lyapunov stability based adaptive learning rates. Neurocomputing 287:102–117
3. Guclu R, Gulez K (2008) Neural network control of seat vibrations of a non-linear full vehicle model using PMSM. Math Comput Modell 47:1356–1371
4. Zaki AM, El-Bardini M, Soliman FAS, Sharaf MM (2018) Embedded two level direct adaptive fuzzy controller for DC motor speed control. Ain Shams Eng J 9:65–75
5. Chang WD, Shih SP (2010) PID controller design of nonlinear systems using an improved particle swarm optimization approach. Commun Nonlinear Sci Numerl Simul 15:3632–3639
6. Khater AA, El-Nagar AM, El-Bardini M, El-Rabaie NM (2018) Adaptive T-S fuzzy controller using reinforcement learning based on Lyapunov stability. J Frankl Inst 355:6390–6415
7. Shang C, Yang F, Huang D, Lyu W (2014) Data-driven soft sensor development based on deep learning technique. J Process Control 24:223–233
8. Zuo R, Xiong Y, Wang J, Carranza EJM (2019) Deep learning and its application in geochemical mapping. Earth-Sci Rev 192:1–14
9. Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE (2017) A survey of deep neural network architectures and their applications. Neurocomputing 234:11–26
10. Qiao J, Wang G, Li X, Li W (2018) A self-organizing deep belief network for nonlinear system modeling. Appl Soft Comput 65:170–183
11. Sutar MK, Pattnaik S, Rana J (2015) Neural based controller for smart detection of crack in cracked cantilever beam. Mater Today Proc 2:2648–2653
12. Medjber A, Guessoum A, Belmili H, Mellit A (2016) New neural network and fuzzy logic controllers to monitor maximum power for wind energy conversion system. Energy 106:137–146. https://doi.org/10.1016/j.energy.2016.03.026
13. Rajan S, Sahadev S (2016) Performance improvement of fuzzy logic controller using neural network. Procedia Technol 24:704–714
14. Farahani M, Ganjefar S (2015) An online trained fuzzy neural network controller to improve stability of power systems. Neurocomputing 162:245–255
15. da Silva Ribeiro VDJ, de Moraes Oliveira GF, Cristian M, Martins AL, Fernandes LD, Vega MP (2019) Neural network based controllers for the oil well drilling process. J Pet Sci Eng 176:573–583
16. Kumar R, Srivastava S, Gupta JRP (2017) Diagonal recurrent neural network based adaptive control of nonlinear dynamical systems using lyapunov stability criterion. ISA Trans 67:407–427
17. Zaineb BM, Aicha A, Mouna BH, Lassaad S (2017) Speed control of DC motor based on an adaptive feed forward neural IMC controller. In: 2017 International conference on green energy conversion systems (GECS), pp 1–7
18. Nasr MB, Chtourou M (2014) Neural network control of nonlinear dynamic systems using hybrid algorithm. Appl Soft Comput 24:423–431
19. Cai Z, Zhang B, Yu X (2017) Neural network delayed control of an idealized offshore steel jacket platform. In: 2017 Eighth international conference on intelligent control and information processing (ICICIP). IEEE, pp 282–286
20. Shafiq MA (2016). Direct adaptive inverse control of nonlinear plants using neural networks. In: 2016 Future Technologies Conference (FTC). IEEE, pp 827–830
21. Son NN, Van Kien C, Anh HPH (2017) A novel adaptive feed-forward-PID controller of a SCARA parallel robot using pneumatic artificial muscle actuator based on neural network and modified differential evolution algorithm. Robot Auton Syst 96:65–80
22. Upadhyay D, Tarun N, Nayak T (2013) ANN based intelligent controller for inverted pendulum system. In: 2013 students conference on engineering and systems (SCES). IEEE, pp 1–6

23. Chen J, Huang TC (2004) Applying neural networks to on-line updated PID controllers for nonlinear process control. J Process Control 14:211–230
24. Litjens G, Kooi T, Bejnordi BE et al (2017) A survey on deep learning in medical image analysis. Med Image Anal 42:60–88
25. Yuan J, Hou X, Xiao Y, Cao D, Guan W, Nie L (2019) Multi-criteria active deep learning for image classification. Knowledge-Based Syst 172:86–94
26. Mohanty SP, Hughes DP, Salathé M (2016) Using deep learning for image- based plant disease detection. Front Plant Sci 7:14–19
27. Altan G, Kutlu Y, Pekmezci AÖ, Nural S (2018) Deep learning with 3D-second order difference plot on respiratory sounds. Biomed Signal Process Control 45:58–69
28. Chatterjee A, Gupta U, Chinnakotla MK, Srikanth R, Galley M, Agrawal P (2019) Understanding emotions in text using deep learning and big data. Comput Human Behav 93:309–317
29. Jin X, Shao J, Zhang X, An W, Malekian R (2016) Modeling of nonlinear system based on deep learning framework. Nonlinear Dyn 84:1327–1340
30. De la Rosa E, Yu W (2016) Randomized algorithms for nonlinear system identification with deep learning modification. Inf Sci 364:197–212
31. Qiao J, Wang G, Li W, Li X (2018) A deep belief network with PLSR for nonlinear system modeling. Neural Netw 104:68–79
32. Zhang K, Zheng L, Liu Z, Jia N (2019) A deep learning based multitask model for network-wide traffic speed predication. Neurocomputing 396:438–450
33. Krüger J, Lehr J, Schlüter M, Bischoff N (2019) Deep learning for part identification based on inherent features. CIRP Ann 68:9–12
34. Saba L, Biswas M, Kuppili V et al (2019) The present and future of deep learning in radiology. Eur J Radiol 114:14–24
35. Lyu Y, Chen J, Song Z (2019) Image-based process monitoring using deep learning framework. Chemom Intell Lab Syst 189:8–17
36. McBee MP, Awan OA, Colucci AT et al (2018) Deep learning in radiology. Acad Radiol 25:1472–1480
37. Zhu Z, Albadawy E, Saha A, Zhang J, Harowicz MR, Mazurowski MA (2019) Deep Learning for identifying radiogenomic associations in breast cancer. Comput Biol Med 109:85–90
38. Hinton GE (2012) A practical guide to training restricted Boltzmann machines. Neural networks: tricks of the trade. Springer, Berlin, pp 599–619
39. Golovko V, Kroshchanka A, Turchenko V, Jankowski S, Treadwell D (2015) A new technique for restricted Boltzmann machine learning. In: 2015 IEEE 8th international conference on intelligent data acquisition and advanced computing systems: technology and applications (IDAACS) (Vol. 1, pp. 182–186)
40. Golovko V, Kroshchanka A, Treadwell D (2016) The nature of unsupervised learning in deep neural networks: a new understanding and novel approach. Opt Memory Neural Netw 25:127–141
41. Hinton GE (2002) Training products of experts by minimizing contrastive divergence. Neural Comput 14:1771–1800
42. Tieleman T (2008) Training restricted Boltzmann machines using approximations to the likelihood gradient. In: Proceedings of the 25th international conference on Machine learning (pp. 1064–1071)
43. Heaton, J (2015) Artificial Intelligence for Humans, Volume 3: Neural Networks and Deep Learning. Heaton Research
44. Erhan D, Bengio Y, Courville A et al (2010) Why does unsupervised pre-training help deep learning? J Mach Learn Res 11:625–660
45. Ahn IS, Lan JH (1995) Implementation of a neural network controller and estimator using a digital signal processing chip. Math Comput Model 21:133–141
46. Marini F, Magrì AL, Bucci R (2007) Multilayer feed-forward artificial neural networks for class modeling. Chemom Intell Lab Syst 88:118–124
47. Fourati F, Chtourou M (2007) A greenhouse control with feed-forward and recurrent neural networks. Simul Modell Pract Theory 15:1016–1028
48. Aftab MS, Shafiq M, Yousef H (2015) Lyapunov stability criterion based neural inverse tracking for unknown dynamic plants. In: 2015 IEEE international conference on industrial technology (ICIT) (pp. 321–325)
49. Behera L, Kumar S, Patnaik A (2006) On adaptive learning rate that guarantees convergence in feed forward networks. IEEE Trans Neural Netw 17:1116–1125
50. Khater AA, El-Nagar AM, El-Bardini M, El-Rabaie NM (2019) Online learning based on adaptive learning rate for a class of recurrent fuzzy neural network. Neural Comput Appl. https://doi.org/10.1007/s00521-019-04372-w
51. Khater AA., El-Nagar AM, El-Bardini M El-Rabaie N (2019) A novel structure of actor-critic learning based on an interval Type-2 TSK fuzzy neural network. IEEE Trans Fuzzy Syst
52. Narendra KS, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Netw 1:4–27
53. Kayacan E, Kayacan E, Khanesar MA (2014) Identification of nonlinear dynamic systems using type-2 fuzzy neural networks—A novel learning algorithm and a comparative study. IEEE Trans Indus Electr 62:1716–1724