Neural Computing and Applications (2020) 32:8691–8710
https://doi.org/10.1007/s00521-019-04372-w

ORIGINAL ARTICLE

Online learning based on adaptive learning rate for a class of recurrent fuzzy neural network

A. Aziz Khater[1] · Ahmad M. El-Nagar[1] · Mohammad El-Bardini[1] · Nabila M. El-Rabaie[1]

Received: 17 January 2019 / Accepted: 19 July 2019 / Published online: 29 July 2019

Abstract
This paper proposes a novel structure of a recurrent interval type-2 TSK fuzzy neural network (RIT2-TSK-FNN) controller based on a reinforcement learning scheme for improving the performance of nonlinear systems using a less number of rules. The parameters of the proposed RIT2-TSK-FNN controller are leaned online using the reinforcement actor–critic method. The controller performance is improved over the time as a result of the online learning algorithm. The controller learns from its own mistakes and faults through the reward and punishment signal from the external environment and seeks to reinforce the RIT2-TSK-FNN controller parameters to converge. In order to obtain less number of rules, the structure learning is performed and thus the RIT2-TSK-FNN rules are obtained online based on the type-2 fuzzy clustering. The online adaptation of the proposed RIT2-TSK-FNN controller parameters is developed using the Levenberg–Marquardt method with adaptive learning rates. The stability analysis is discussed using the Lyapunov theorem. The obtained results show that the proposed RIT2-TSK-FNN controller using the reinforcement actor–critic technique is more preferable than the RIT2-TSK-FNN controller without the actor–critic method under the same conditions. The proposed controller is applied to a nonlinear mathematical system and an industrial process such as a heat exchanger to clarify the robustness of the proposed structure.

Keywords Reinforcement learning · Recurrent interval type-2 fuzzy neural networks · LM method · Lyapunov function · Adaptive learning rate

1 Introduction

The control of the industrial systems that have inherent uncertainties in terms of precision of the sensors, a noise produced by the sensors, nonlinear characteristic of the actuator and the system structure has been a serious challenge [1, 2]. The conventional control methodologies are found to be inadequate for meeting these requirements, especially when it is needed for controlling the nonlinear dynamical systems in real time [3]. Moreover, it is hard to apply the conventional control approaches when the system model is anonymous or slightly known. Accordingly, the development of more advanced techniques becomes more important, in particular such complex nonlinear dynamical systems [4]. Recently, the stable controllers for the nonlinear dynamical systems are presented based on the learning approaches such as genetic, sliding mode, backstepping and particle swarm optimization [5–8].

On the other hand, there is an active area of model-free approaches within the framework of machine learning and computational intelligence such as reinforcement learning (RL) [9–11]. The RL paradigm, which is called a goal-oriented system, is mainly depending on the concept of learning from the experience with the principle of the reward and the punishment. The RL algorithm obtains its experience and evolves strategies to take an optimal control

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

A. Aziz Khater
Abdelazizali2020@yahoo.com

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

Nabila M. El-Rabaie
Nabila2100@gmail.com

[1] Department of Industrial Electronics and Control Engineering, Faculty of Electronic Engineering, Menoufia University, Menouf 32852, Egypt

policy for achieving a good control quality from unknown nonlinear dynamical system [12]. Q-learning, actor–critic learning, Sarsa learning and adaptive dynamic programming (ADP) are different algorithms normally used in the RL [13]. The Q-learning and Sarsa learning algorithms need a range of actions and the number of states to give the best response [14, 15]. Dynamic programming (DP) is a widely used method for generating the optimal control for nonlinear systems. This method employs Bellman's principle of optimality, but it has a well-recognized problem, namely curse of dimensionality [16]. Because of the backward direction of research and the particular number of analytic conditions, the Hamilton–Jacobi–Bellman (HJB) equation makes the DP method prohibit the wide use in real-time control. The approximated solution for the Bellman equation was obtained by developing an element that is known as a critic [17]. The DP and the multilayer perceptron (MLP) neural network are incorporated together to obtain the ADP [18]. The adaptive critic designs (ACDs), which are considered the universal functional approaching structures, are formed from the artificial neural networks (ANNs) to adapt the controller parameters when the controlled plant is affected by disturbances, uncertainties and load changes [19, 20]. The asymptotic stability analysis issue for the ANNs is investigated in [21–26]. The reinforcement actor–critic learning methods have a separate memory structure to explicitly represent the policy independent of the value function, where the actor carries out the approximation of the control policy function by applying an action to the system and then the critic recognizes the approximation of the value function, i.e., assessing the current control policy.

In [27], the authors implemented the actor and the critic elements using the ANN, but their algorithm has not been tested under parameter variation uncertainty and disturbance effects. In [28–30], the critic and the actor elements were implemented using the feed-forward ANN (FFANN) where their structures utilize the states and the output of the actor as the inputs to the critic. In [31], the authors presented one network that represents the incorporation of the critic and the actor elements, the so-called consolidated actor–critic model (CACM), in which the prior values of the hidden layer nodes beside the states and the action are considered the inputs of the network. The researchers added a third network, which is used as a reference [32]. Accordingly, the ADP structure combines an actor, a critic and a reference network (ADPACRN). This structure facilities the learning by building an internal reinforcement signal. The third network may be considered a goal network as described in [33]. This structure is known as a goal representation heuristic dynamic programming (GRHDP). The inserted third network contributes to improving the performance of the controller; however, it increases the computational time as a result of incrementing the number of weights [34]. Moreover, the actor was represented by the Takagi–Sugeno (T–S) fuzzy system in which the Lyapunov theory was used for deriving the parameter adaptation law, namely adaptive T–S fuzzy using reinforcement learning based on Lyapunov stability (ATSFRL-LS) [35]. The structure of the ATSFRL-LS depends on the parallel distributed control (PDC), which is used for stabilizing the system. The PDC uses a fuzzy system for the system model and another fuzzy system.

The FNN merges the capability of the fuzzy inference technique and the capability of the ANNs for online learning from the plant [36]. There are two types of the FNN: the feed-forward FNN (FFNN) and the recurrent FNN (RFNN). The topologies of the RFNN involve feedback loops that can memorize the past data. In contrast to the FFNN architectures, which exhibit static input–output behavior, the RFNN is capable of storing the data from the past (e.g., previous plant states) and overcoming the network size problem. Therefore, it is more convenient for the nonlinear dynamic systems analysis [37, 38]. The FFNN and the RFNN cannot minimize the influence of system uncertainties in the real plant due to their dependence on the type-1 fuzzy sets (T1-FSs) [39]. On the other hand, the type-2 fuzzy logic systems (T2-FLSs), which use the type-2 fuzzy sets (T2-FSs), are capable of minimizing the effect of the system uncertainties compared with the type-1 fuzzy logic systems (T1-FLSs) counterpart. The T2-FLSs have been successfully applied in various applications [40–45]. In recent years, the interval type-2 FNNs (IT2-FNNs) have been used for handling the uncertainties and the interval type-2 TSK-FNNs (IT2-TSK-FNNs) are used for nonlinear systems identification and control. The learning accuracy and the network performance for the TSK-type IT2-FNNs are better than for the Mamdani-type IT2-FNNs [46–48].

All the previous works that were described in [27–35] have a less performance with the influence of the system uncertainties, external disturbances and measurement noise. For solving this problem, a novel structure of a recurrent interval type-2 TSK fuzzy neural network (RIT2-TSK-FNN) is proposed. The proposed RIT2-TSK-FNN is learned online using the RL scheme. The proposed RL scheme consists of two parts: The first part is the critic that is represented based on the ANN and the other is the actor that is implemented based on the proposed RIT2-TSK-FNN. The number of the IF-THEN rules for the proposed RIT2-TSK-FNN is obtained online based on the type-2 fuzzy clustering. The online learning method for the critic and the actor is developed based on the LM method, which gives a perfect exchange between the stability of the gradient steepest descent method and the speed of the Newton algorithm. To speed up the learning algorithm, the adaptive learning parameter is developed by utilizing a fuzzy logic system. The learning rates conditions are derived to guarantee the controlled system stability by using the Lyapunov stability theory. To show the robustness of the proposed structure to respond the system uncertainties, the proposed controller is applied to an uncertain mathematical nonlinear system and the heat exchanger process. The main advantages of the proposed controller over existing controllers are summarized as: (1) The proposed controller has a robustness performance with the influence of the system uncertainties and external disturbances compared to other learning techniques due to using the IT2-TSK-FNN. (2) The proposed controller has structure learning and has a good performance with small number of rules due to the recurrent in the input and rule layers. (3) The learning rates are changed online according to fuzzy logic system during the implementation of the algorithm to assure the stability and speed up the convergence.

The main contributions of this paper are summarized as: (1) Proposing a new structure of the RIT2-TSK-FNN. (2) Proposing a new online learning for the developed RIT2-TSK-FNN based on a reinforcement actor–critic scheme. (3) Developing the parameters of the actor–critic based on the LM algorithm. (4) Obtaining the optimal values for the learning rates using the fuzzy logic system and the Lyapunov function.

The rest of the paper is organized as follows: The structure of the RIT2-TSK-FNN is described in Sect. 2. Section 3 describes the proposed RL scheme. Section 4 presents the online learning of the proposed RL scheme. Section 5 describes the simulation results for nonlinear mathematical system and the steam-water heat exchanger process. This would be followed by the conclusion and the relevant references.

2 Proposed RIT2-TSK-FNN structure

This section presents the structure of the RIT2-TSK-FNN. Figure 1 shows the structure of the proposed network, which is composed of six layers. The antecedent parts of the RIT2-TSK-FNN are represented using the IT2-FSs. The consequent part for each fuzzy rule is characterized by the TSK type. Each type-2 fuzzy rule for the RIT2-TSK-FNN is defined as:

$$\text{Rule } i: \text{IF } x_1(n) \text{ is } \tilde{B}_1^i \text{ AND} \ldots x_k(n) \text{ is } \tilde{B}_k^i \text{ THEN } h^i(n) \text{ is } \tilde{b}_0^i + \sum_{j=1}^{k} \tilde{b}_j^i x_j(n), \quad i = 1, 2, \ldots, M, \tag{1}$$

where $x_1(n), \ldots, x_k(n)$ are the recurrent incoming inputs, $\tilde{B}_j^i$ are the IT2-FSs, $\tilde{b}_0^i$, $\tilde{b}_j^i$ are interval sets where $\tilde{b}_0^i = [c_0^i - s_0^i,\ c_0^i + s_0^i]$ and $\tilde{b}_j^i = [c_j^i - s_j^i, c_j^i + s_j^i]$, and k and M are the network inputs and the number of the rules, respectively. For each layer, $O_j^{(p)}$ is symbolizing the output of layer p.

Layer 1: This layer is called an input layer, which is indicated in Fig. 1, in which the internal feedback connection can temporarily store the dynamic data and cope with temporal input noise efficiently. The mathematical expression for this layer is expressed as:

$$O_j^{(1)}(n) = x_j(n) + \alpha_j(n)\, O_j^{(1)}(n-1), \quad j = 1, \ldots, k, \tag{2}$$

where $x_j(n)$ represents the input variable and $\alpha_j(n)$ is the input recurrent weight.

Layer 2: This layer is called a membership layer, which is described in Fig. 1, in which each node is represented by a membership function (MF) where the fuzzification operation is performed. The output from the first layer $O_j^{(1)}(n)$ is fuzzified using the Gaussian IT2-FS with a certain mean $\left[m_{j1}^i, m_{j2}^i\right]$ and a fixed width σ as shown in Fig. 2, where i represents the fuzzy set. Each layer node can be symbolized as an upper MF, $\bar{O}_{ij}^{(2)}(n)$, and a lower MF, $\underline{O}_{ij}^{(2)}(n)$, which are defined as:

$$\bar{O}_{ij}^{(2)}(n) = \begin{cases} \exp\left(-\frac{1}{2}\left(\frac{O_j^{(1)}(n) - m_{j1}^i(n)}{\sigma_j^i(n)}\right)^2\right), & O_j^{(1)}(n) < m_{j1}^i(n) \\ 1, & m_{j1}^i(n) \le O_j^{(1)}(n) \le m_{j2}^i(n), \\ \exp\left(-\frac{1}{2}\left(\frac{O_j^{(1)}(n) - m_{j2}^i(n)}{\sigma_j^i(n)}\right)^2\right), & O_j^{(1)}(n) > m_{j2}^i(n) \end{cases} \tag{3}$$

$$\underline{O}_{ij}^{(2)}(n) = \begin{cases} \exp\left(-\frac{1}{2}\left(\frac{O_j^{(1)}(n) - m_{j2}^i(n)}{\sigma_j^i(n)}\right)^2\right), & O_j^{(1)}(n) \le \frac{m_{j1}^i(n) + m_{j2}^i(n)}{2} \\ \exp\left(-\frac{1}{2}\left(\frac{O_j^{(1)}(n) - m_{j1}^i(n)}{\sigma_j^i(n)}\right)^2\right), & O_j^{(1)}(n) > \frac{m_{j1}^i(n) + m_{j2}^i(n)}{2}. \end{cases} \tag{4}$$

Layer 3: This layer is called a recurrent layer, which is indicated in Fig. 1, where the number of the nodes equals the number of the recurrent rules, which corresponds to the number of the fuzzy sets in each input. Each recurrent rule node contains an internal feedback loop, which handles the uncertainties. The output of each node performs a temporal firing strength whose value does rely not only on the current spatial firing strength $F^i(n)$ but also on the previous temporal firing strength $F^i(n-1)$. The output of each recurrent node is computed by a fuzzy AND operator. The firing strength can be described by a crisp interval as follows:

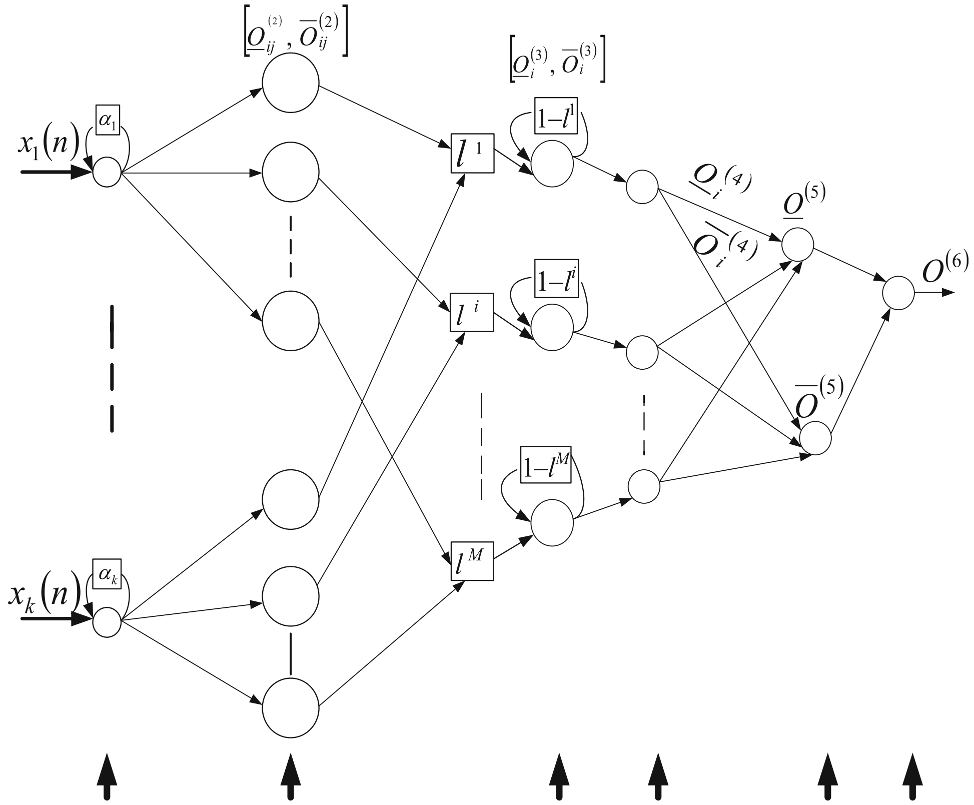


Fig. 1 Structure of the proposed RIT2-TSK-FNN

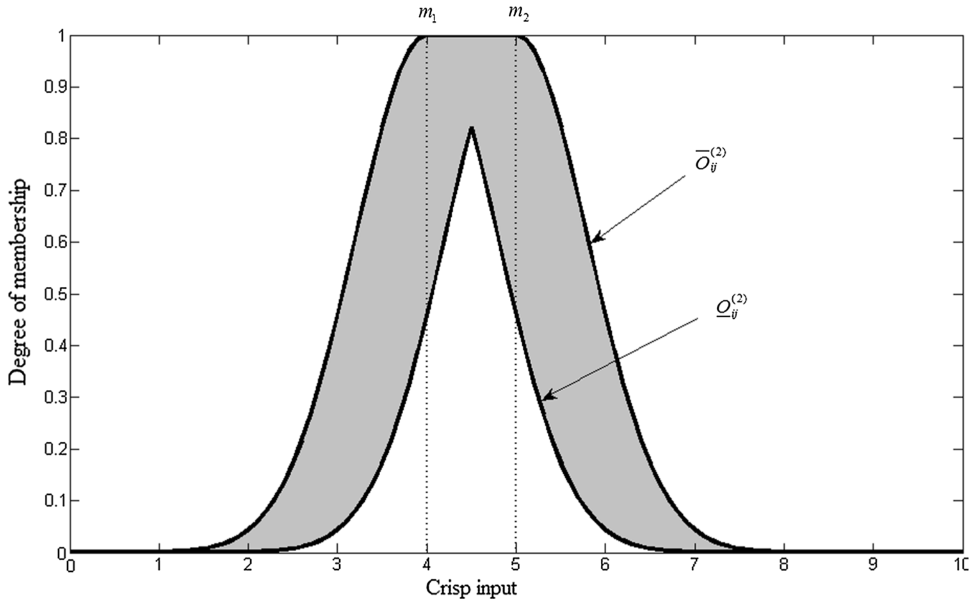


Fig. 2 Uncertain mean Gaussian IT2-FS

$$F^i(n) = \left[\bar{\Omega}^i(n), \underline{\Omega}^i(n)\right], \quad i = 1, \ldots, M, \tag{5}$$

$$\underline{\Omega}^i(n) = \prod_{j=1}^{k} \underline{O}_{ij}^{(2)}(n), \bar{\Omega}^i(n) = \prod_{j=1}^{k} \bar{O}_{ij}^{(2)}(n). \tag{6}$$

The crisp interval of the temporal firing strength $\left[\bar{O}_i^{(3)}(n), \underline{O}_i^{(3)}(n)\right]$ is a linear aggregation of the previous temporal firing strength $O_i^{(3)}(n-1)$ and the spatial firing strength $F^i(n)$, which can be given as:

$$O_i^{(3)}(n) = l^i F^i(n) + \left(1 - l^i\right) O_i^{(3)}(n-1), \tag{7}$$

where l^i is an internal feedback weight, $(0 \le l^i \le 1)$. The value of l^i defines the achievement ratio between the immediate and previous inputs that effects on the network output. Thus, (7) can be written as:

$$\left[\bar{O}_i^{(3)}(n), \underline{O}_i^{(3)}(n)\right] = l^i \left[\bar{\Omega}^i(n), \underline{\Omega}^i(n)\right] + \left(1 - l^i\right)\left[\bar{O}_i^{(3)}(n-1), \underline{O}_i^{(3)}(n-1)\right], \tag{8}$$

where

$$\bar{O}_i^{(3)}(n) = l^i \bar{\Omega}^i(n) + \left(1 - l^i\right) \bar{O}_i^{(3)}(n-1) \tag{9}$$

and

$$\underline{O}_i^{(3)}(n) = l^i \underline{\Omega}^i(n) + \left(1 - l^i\right) \underline{O}_i^{(3)}(n-1). \tag{10}$$

Layer 4: This layer is called a consequent layer, which is described in Fig. 1, in which each node output is implemented depending on the TSK type. The consequent parameters for each rule are T1-FS. The number of the layer nodes is corresponding to the number of the obtained rules. The output of this layer $\left[\underline{O}_i^{(4)}(n), \bar{O}_i^{(4)}(n)\right]$ is expressed as:

$$\left[\underline{O}_i^{(4)}(n), \bar{O}_i^{(4)}(n)\right] = \left[c_0^i - s_0^i, c_0^i + s_0^i\right] + \left[\left(\sum_{j=1}^{k} c_j^i O_j^{(1)}(n) - \sum_{j=1}^{k} s_j^i \left|O_j^{(1)}(n)\right|\right), \times \left(\sum_{j=1}^{k} c_j^i O_j^{(1)}(n) + \sum_{j=1}^{k} s_j^i \left|O_j^{(1)}(n)\right|\right)\right], \tag{11}$$

where

$$\underline{O}_i^{(4)}(n) = \sum_{j=0}^{k} c_j^i O_j^{(1)}(n) - \sum_{j=0}^{k} s_j^i \left|O_j^{(1)}(n)\right| \tag{12}$$

and

$$\bar{O}_i^{(4)}(n) = \sum_{j=0}^{k} c_j^i O_j^{(1)}(n) + \sum_{j=0}^{k} s_j^i \left|O_j^{(1)}(n)\right|, \tag{13}$$

where $O_0^{(1)}(n) = 1$.

Layer 5: This layer is called a type reduction layer, which is indicated in Fig. 1, in which the H factors are used to reduce the computation operations of the IT2-FLS [49]. The design factors $[H_l, H_r]$ weight the sharing of lower and upper firing levels of each fired rule. The output of type reduction is represented by two nodes, which can be calculated as:

$$\underline{O}^{(5)}(n) = \frac{(1-H_l)\sum_{i=1}^{M} \bar{O}_i^{(3)}(n)\underline{O}_i^{(4)}(n) + H_l \sum_{i=1}^{M} \underline{O}_i^{(3)}(n)\underline{O}_i^{(4)}(n)}{\sum_{i=1}^{M} \underline{O}_i^{(3)}(n) + \bar{O}_i^{(3)}(n)}, \tag{14}$$

$$\bar{O}^{(5)}(n) = \frac{(1-H_r)\sum_{i=1}^{M} \underline{O}_i^{(3)}(n)\bar{O}_i^{(4)}(n) + H_r \sum_{i=1}^{M} \bar{O}_i^{(3)}(n)\bar{O}_i^{(4)}(n)}{\sum_{i=1}^{M} \underline{O}_i^{(3)}(n) + \bar{O}_i^{(3)}(n)}. \tag{15}$$

Layer 6: This is the final layer, which represents the output layer. The defuzzified output is computed as:

$$O^{(6)}(n) = \underline{O}^{(5)}(n) + \bar{O}^{(5)}(n). \tag{16}$$

3 Proposed RL scheme

The block diagram of the proposed RL structure is shown in Fig. 3, where it consists of two parts: the critic and the actor. The proposed RL structure has one input variable, $e(n)$, which is the error signal between the reference trajectory and the system output, and one output variable, $u(n)$, which is the control signal that is applied to the system. The actor part has two inputs: $e(n)$ and change in error signal $\Delta e(n)$. The critic part has three inputs: $e(n)$, $e(n-1)$ and $u(n)$. The reward signal $r(n)$ is defined as:

$$r(n) = \Theta_{c_1}(n) + \Theta_{c_2}(n), \tag{17}$$

where

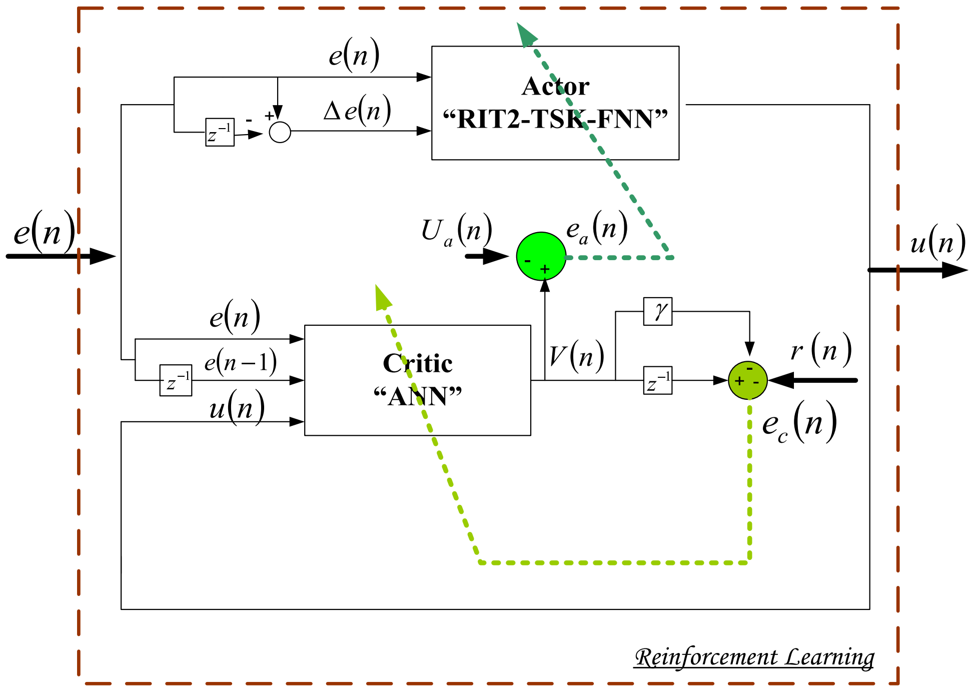
$$\Theta_{c_1}(n) = \begin{cases} 0 & |e(n)| \le \delta \\ -1 & \text{otherwise} \end{cases} \qquad \Theta_{c_2}(n) = \begin{cases} 0 & |e(n)| \le |e(n-1)| \\ -1 & \text{otherwise} \end{cases}, \tag{18}$$

where the symbol δ symbolizes a small constant value, i.e., $\delta = 0.001$. Simply the reinforcement reward signal gives the effect of applying the control signal from the actor to the system that can be represented by either a "zero" or "negative value" corresponding to "adequate" or "insufficiency," respectively. The parameter $e_c(n)$ represents the error of the critic network, $e_a(n)$ represents the error of the actor network, $V(n)$ represents the output of the critic and $U_a(n)$ represents the required conclusive goal.

3.1 The critic network

The critic part shown in Fig. 3 is implemented using ANN, and the structure of this ANN is shown in Fig. 4. The inputs of the critic network are $z_1(n) = e(n-1)$, $z_2(n) =$

Fig. 3 Block diagram of the proposed RL structure



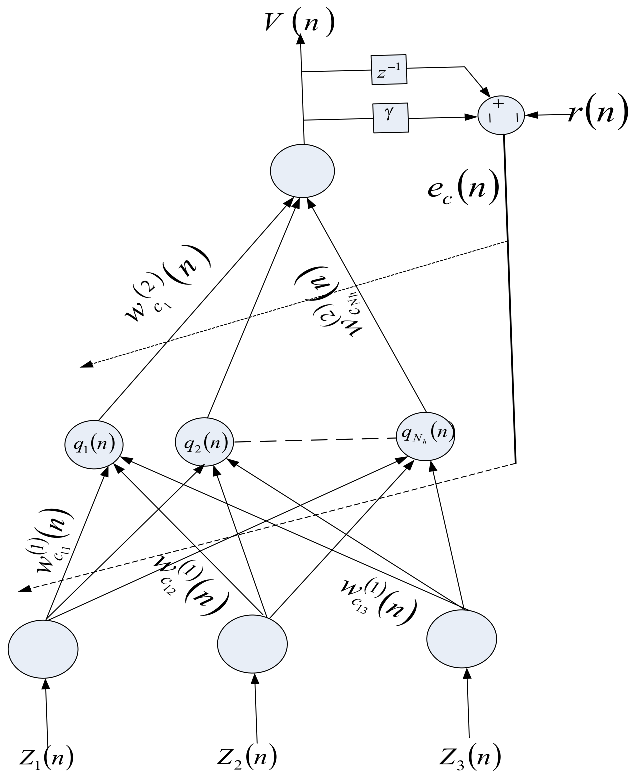


Fig. 4 Critic neural network

$e(n)$, $z_3(n) = u(n)$, and the critic output is $V(n)$. The critic output, $V(n)$, is calculated as follows:

$$\text{net}_m(n) = \sum_{b=1}^{3} w_{c_{mb}}^{(1)}(n) z_b(n), \quad m = 1, \ldots, N_h, \tag{19}$$

$$q_m(n) = \frac{1 - e^{-\text{net}_m(n)}}{1 + e^{-\text{net}_m(n)}}, \quad m = 1, \ldots, N_h, \tag{20}$$

$$V(n) = \sum_{m=1}^{N_h} w_{c_m}^{(2)}(n) q_m(n), \tag{21}$$

where net_m is the mth hidden node input of the critic network, q_m is the hidden node output, $w_{c_{mb}}^{(1)}(n)$ is the weight between bth input node and mth hidden node, $w_{c_m}^{(2)}(n)$ is the weight between mth hidden node and output node and N_h is the total number of hidden nodes.

The critic network parameters, $\theta_c(n)$, are updated dependent on the prediction error of the critic network, which can be described as:

$$e_c(n) = V(n-1) - r(n) - \gamma V(n). \tag{22}$$

3.2 The actor network

The actor part shown in Fig. 3 is implemented using the RIT2-TSK-FNN, which is explained in detail in Sect. 2. The inputs of the actor RIT2-TSK-FNN are $x_1(n) = e(n)$, $x_2(n) = \Delta e(n)$ and the output is $O^{(6)}(n) = u(n)$. The actor network parameters, $\theta_a(n)$, are updated based on the error between the required conclusive goal, which is denoted by $U_a(n)$ and the approximate $V(n)$ function from the critic network given in (23). In the design model, U_a is limited to "0," which means the success of the reinforcement signal.

$$e_a(n) = V(n) - U_a(n). \tag{23}$$

4 Online learning for the proposed RL scheme

In this section, the online learning for the proposed RL scheme is composed of two parts which are described. The first part is the structure learning that aims to obtain the number of rules for the actor (RIT2-TSK-FNN). The other part is the parameter learning that updates the proposed RL scheme parameters based on the LM algorithm with an adaptive learning rate.

4.1 Structure learning

At each instant, the online structure learning creates the fuzzy rules based on the input data. Here, the type-2 fuzzy clustering is used to perform the structure learning with respect to the rule firing strength [50, 51]. The incoming data, $x_j(n)$, are utilized for creating the first type-2 fuzzy rule. The first Gaussian IT2-FS parameters related to the first rule are designed as:

$$\left[m_{j1}^{1}, m_{j2}^{1}\right] = \left[O_j^{(1)}(n) - \Delta O, O_j^{(1)}(n) + \Delta O\right], \quad \sigma = \sigma_{\text{const}}, \quad j = 1, \ldots, k, \tag{24}$$

where the parameter ΔO is a constant value that represents the range of the initial IT2-FS. The parameter σ_{const} is a predefined value, which is set as $\sigma_{\text{const}} = 0.3$. At each instant n, the firing strengths are calculated as given in (7). The mean value of the firing strength O_f^i can be computed for each instant as follows:

$$O_f^i = \frac{\underline{O}_i^{(3)}(n) + \bar{O}_i^{(3)}(n)}{2}. \tag{25}$$

For subsequent input data $x_j(n)$, the calculation of the number of the rules is obtained using the following formula:

$$\Psi = \underset{1 \le i \le M(n)}{\arg\max}\; O_f^i, \tag{26}$$

where $M(n)$ denotes the number of the current rules at instant n. A new rule is generated at $M(n+1) = M(n) + 1$ if $O_f^{\Psi} \le O_{\text{th}}$ (O_{th} is a pre-specified threshold). The parameters of the Gaussian IT2-FSs for a new rule are defined as:

$$\left[m_{j1}^{M(n)+1}, m_{j2}^{M(n)+1}\right] = \left[O_j^{(1)}(n) - \Delta O, O_j^{(1)}(n) + \Delta O\right], \quad j = 1, \ldots, k, \tag{27}$$

and the spread is computed as:

$$\sigma^{M(n+1)} = \vartheta \cdot \left| O_j(n) - \left(\frac{m_{j1}^{\Psi} + m_{j2}^{\Psi}}{2} \right) \right|. \tag{28}$$

When the overlapping parameter, ϑ, is defined as $\vartheta = 0.5$, this indicates that the new Gaussian IT2-FS spread is chosen as the half of the Euclidean space from the best matching center. The initial consequent parameters for each new rule are as follows:

$$c_0^1 = c_0^{M(n+1)} = r_1, \quad c_j^1 = c_j^{M(n+1)} = r_2,$$
$$s_0^1 = s_0^{M(n+1)} = r_3, \quad s_j^1 = s_j^{M(n+1)} = r_4 \quad j = 1, \ldots, k, \tag{29}$$

where r_1, r_2, r_3 and r_4 are small random values. These parameters are learned online according to the proposed method.

4.2 Parameter learning

The Gauss–Newton (GN) strategy has good convergence characteristics. These characteristics, however, are based on the initial parameters' values. If these values are not chosen properly, this method may easily diverge. On the other hand, the Gradient descent (GD) algorithm demonstrates an excellent behavior in the vicinity of a minimum point, but this algorithm is restricted by a slow speed of the convergence. The performance of the GD method is not adversely affected by the initial values selection [52]. Accordingly, the parameters of the critic and the actor are updated using the LM algorithm, which contributes a nice compromise between guaranteed convergence of the GD algorithm and the speed of the GN strategy. Therefore, the LM algorithm behaves as the GD algorithm when the immediate solution is quite from the correct one and behaves GN strategy when the immediate solution is close to the correct solution [53, 54].

The update rule for the critic weights and the actor parameters according to the GN strategy has the following formula:

$$\Delta\theta_g = -\left[\nabla^2 E_g(\theta_g)\right]^{-1} \nabla E_g(\theta_g), \tag{30}$$

which depends on the performance function, which is defined as:

$$E_g(\theta_g) = \frac{1}{2} e_g^{\mathrm{T}}(\theta_g) e_g(\theta_g), \tag{31}$$

where θ_g is the weights/parameters vector, the suffix g denotes a general for the actor and the critic and $\nabla^2 E_g(\theta_g)$ represents the Hessian matrix. The Hessian term can be expressed as:

$$\nabla^2 E_g(\theta_g) = B^{\mathrm{T}}(\theta_g) B(\theta_g) + S(\theta_g), \tag{32}$$

where the Jacobin matrix $B(\theta_g)$ contains the first derivatives of the critic and actor errors with respect to their weights and parameters. The term $S(\theta_g)$ contains the second derivatives of the critic and the actor errors with respect to their weights and parameters, and it is assumed that $S(\theta_g)$ has a small value comparing with the product of the Jacobin. Therefore, (32) can be approximated as:

$$\nabla^2 E_g(\theta_g) \approx B^{\mathrm{T}}(\theta_g) B(\theta_g). \tag{33}$$

Thus, the GN algorithm can be rewritten as

$$\Delta\theta_g = -\left[B^{\mathrm{T}}(\theta_g) B(\theta_g)\right]^{-1} B^{\mathrm{T}}(\theta_g) e_g(\theta_g). \tag{34}$$

One limitation of this algorithm is that the simplified Hessian matrix might be invertible. To overcome this problem, a modified Hessian matrix can be used as:

$$\nabla^2 E_g(\theta_g) \approx B^{\mathrm{T}}(\theta_g) B(\theta_g) + \lambda_g I, \tag{35}$$

where I denotes the identity matrix. The parameter λ_g should be chosen such that $\nabla^2 E_g(\theta_g)$ is positive definite and thus can be invertible. This modification in the Hessian matrix is corresponding to the LM algorithm. The LM modification to the GN strategy can be expressed as:

$$\Delta\theta_g = -\left[B^{\mathrm{T}}(\theta_g) B(\theta_g) + \lambda_g I\right]^{-1} B^{\mathrm{T}}(\theta_g) e_g(\theta_g). \tag{36}$$

Here, the parameter λ_g guides the algorithm. If it is chosen as a large value, then (36) approximates the GD method, and if it is chosen as a small value, then (36) approximates the GN strategy. If the parameter λ_g is adaptively chosen, the LM algorithm can manage between its two extremes: the GD and GN algorithms. Consequently, the LM algorithm can merge the features of the GD and the GN algorithms, while bypassing their limitations.

The parameter λ_g is now written as $\lambda_g(n)$, which is updated online during the implementation of the algorithm to guarantee the stability (confirming that the Hessian matrix is inverted) and makes the LM algorithm have a fast convergence.

The updating equation for the critic weights according to the LM method is given as:

$$\Delta\theta_c = -\left[B^{\mathrm{T}}(\theta_c) B(\theta_c) + \lambda_c(n) I\right]^{-1} B^{\mathrm{T}}(\theta_c) e_c(\theta_c) \tag{37}$$

where $\lambda_c(n)$ represents a generalized weights vector, i.e., $(w_{c_{mb}}^{(1)}(n), w_{c_m}^{(2)}(n))$, and the derivatives in the Jacobin matrix $B(\theta_c)$ are derived as:

$$\frac{\partial e_c(n)}{\partial w_{c_m}^{(2)}(n)} = \frac{\partial e_c(n)}{\partial V(n)} \frac{\partial V(n)}{\partial w_{c_m}^{(2)}(n)} = -\gamma q_m(n), \tag{38}$$

$$\begin{aligned} \frac{\partial e_c(n)}{\partial w_{c_{mb}}^{(1)}(n)} &= \frac{\partial e_c(n)}{\partial V(n)} \frac{\partial V(n)}{\partial q_m(n)} \frac{\partial q_m(n)}{\partial \mathrm{net}_m(n)} \frac{\partial \mathrm{net}_m(n)}{\partial w_{c_{mb}}^{(1)}(n)}, \\ &= -\gamma w_{c_m}^{(2)}(n) \left[\frac{1}{2}\left(1 - q_m^2(n)\right)\right] z_b(n). \end{aligned} \tag{39}$$

The updating equation for the actor parameters according to the LM method is given as:

$$\Delta\theta_a = -\left[B^{\mathrm{T}}(\theta_a) B(\theta_a) + \lambda_a(n) I\right]^{-1} B^{\mathrm{T}}(\theta_a) e_a(\theta_a), \tag{40}$$

where θ_a represents a generalized actor parameters vector, i.e., $(c_j^i, s_j^i, m_{j1}^i, m_{j2}^i, \sigma_j^i, l^i, \alpha_j, H_l$ and $H_r)$, and the derivatives in the Jacobin matrix $J(\theta_a)$ are derived as:

$$\frac{\partial e_a(n)}{\partial \theta_a(n)} = \frac{\partial e_a(n)}{\partial O^{(6)}(n)} \frac{\partial O^{(6)}(n)}{\partial \theta_a(n)}. \tag{41}$$

The term $\frac{\partial e_a(n)}{\partial O^{(6)}(n)}$ is calculated from the critic neural network as:

$$\begin{aligned} \frac{\partial e_a(n)}{\partial O^{(6)}(n)} &= \frac{\partial e_a(n)}{\partial V(n)} \frac{\partial V(n)}{\partial O^{(6)}(n)} \\ &= \sum_{m=1}^{N_h} w_{c_m}^{(2)}(n) \left[\frac{1}{2}\left(1 - q_m^2(n)\right)\right] w_{c_{m3}}^{(1)}(n). \end{aligned} \tag{42}$$

The term $\frac{\partial O^{(6)}(n)}{\partial \theta_a(n)}$ is obtained as described in "Appendix A."

4.3 Learning rate adaptation

The learning parameter λ_g in the LM method usually takes a constant value. This learning coefficient determines the dynamics of the analyzed RIT2-TSK-FNN controller and determines the system stability. Therefore, the learning rate λ_g (i.e., λ_c for the critic and λ_a for the actor) makes adaptively during the execution of the algorithm. The adaptation of the learning rate should be depending on the system output. When the error is a small value, the learning rate should take a relatively big value. When the error is a big value, the learning rate should take a smaller value.

The change in the learning rate λ_g is associated with the system operation conditions. Hence, the learning rate can be recognized using the fuzzy logic system in which the error value and the change in the error with the scaling factors, J_e and $J_{\Delta e}$, respectively, are the input signals to the fuzzy logic system, while the $\lambda_c(n)$ and $\lambda_a(n)$ are the output

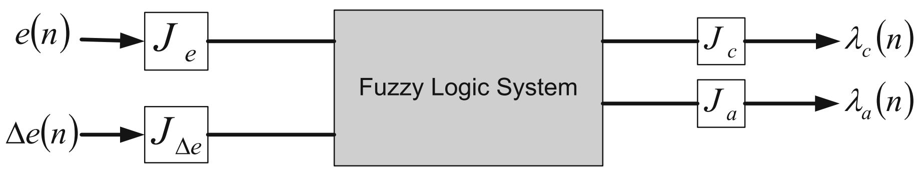


Fig. 5 Block diagram for updating the learning rates using fuzzy logic

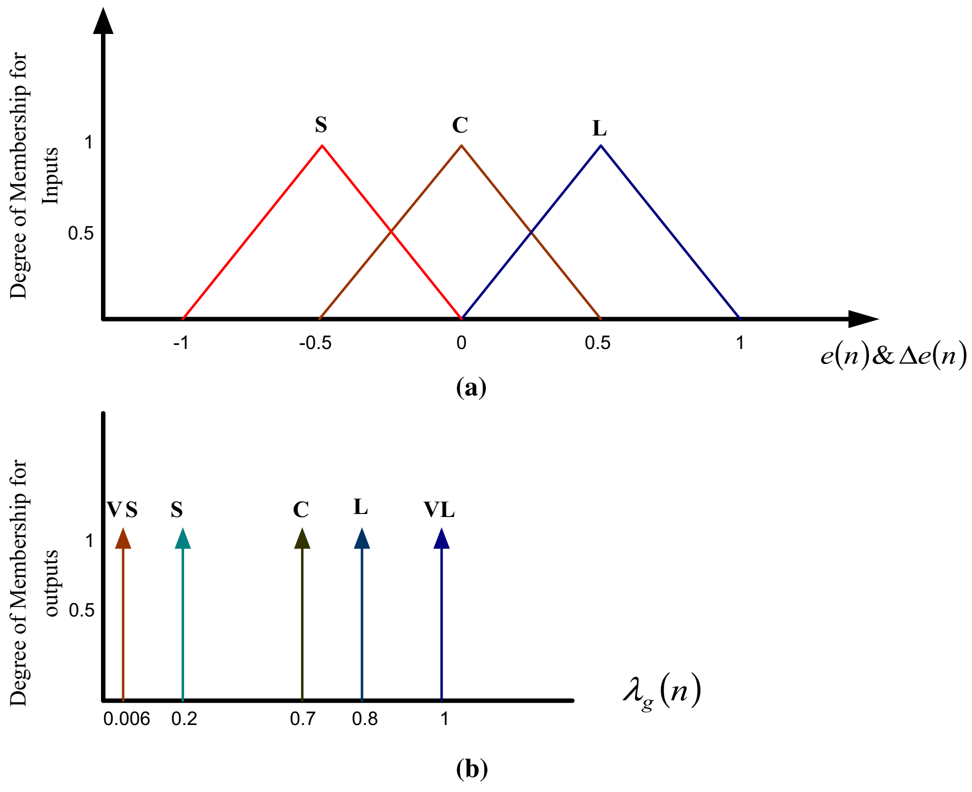


Fig. 6 **a** Input MFs, **b** output MFs

Table 1 Rule base

Change in error signal	Error signal		
	S	C	L
S	VL	L	C
C	L	C	S
L	C	S	VS

with their scaling factors, J_c and J_a, respectively, as shown in Fig. 5. The symmetrical triangular input MFs can be used, and for the simplicity of the defuzzification, the singleton method is performed as shown in Fig. 6. Fuzzy rules are described in Table 1, which is characterized the behavior of the learning rates.

For the online learning procedure of the RL scheme, the learning rate $\lambda_g(n)$ should be chosen to assure the stability of the online updating of the weights/parameters for the critic and the actor, respectively. So, the technique for choosing properly $\lambda_g(n)$ is developed.

Theorem *The learning rates $\lambda_c(n)$ for the critic neural network and $\lambda_a(n)$ for the actor RIT2-TSK-FNN, which are shown in* (37) and (40), *respectively, have the following constraints to guarantee the stability*:

$$\lambda_c(n) \geq \frac{1}{2}\left\|\frac{\partial e_c(n)}{\partial \theta_c(n)}\right\|^2, \quad \lambda_a(n) \geq \frac{1}{2}\left\|\frac{\partial e_a(n)}{\partial \theta_a(n)}\right\|^2. \tag{43}$$

Proof The theorem proof is given in "Appendix B." □

Remark 1 The derivatives in (43) are calculated previously, where the derivatives of the critic error with respect to its parameters are discussed in (38) and (39) and the derivatives of the actor error with respect to its parameters are discussed in (41) and (42).

Remark 2 The proposed RIT2-TSK-FNN controller using RL (RIT2-TSK-FNN-RL) is designed for controlling two nonlinear systems to handle the effect of system uncertainties due to the external disturbance and environmental noise.

5 Simulation results

In order to show the improvements in the proposed RIT2-TSK-FNN-RL controller, the simulation results for the RIT2-TSK-FNN controller in which the parameters are derived based on the fuzzy clustering and the LM learning without the RL are implemented for comparison purposes. Two performance indices are used to measure the performance of the proposed RIT2-TSK-FNN-RL controller, which are the integral absolute error (IAE) and the mean absolute error (MAE). These indices are defined as:

$$\mathrm{IAE} = T\sum_{k=1}^{K_N} |e(n)|, \tag{44}$$

$$\mathrm{MAE} = \frac{1}{k_N}\sum_{k=1}^{k_N} |e(n)|, \tag{45}$$

where k_N is the number of iterations and T is the sampling period.

Once the proposed RIT2-TSK-FNN-RL controller is initialized using one rule for the actor element, the initialized parameters of the rule consequence are described as (29) and the Gaussian membership function initialized as (24), it will be plugged into the system and works in the following procedure:

Step 1: The actor element receives the measured system states; $x_1(n) = e(n)$, $x_2(n) = \Delta e(n)$ and uses it to generate a new rule if the condition $O_f^{\Psi} \le O_{\mathrm{th}}$ is satisfied and initiate the parameters of the new rule as (27)–(29). The actor output is the control signal that is implemented according to (16).

Step 2: The critic element receives the measured system states: $z_1(n) = e(n-1), z_2(n) = e(n), \quad z_3(n) = u(n)$, and uses it to calculate the output of the critic as $V(n)$ as (21).

Step 3: The critic element will update its parameters according to (37)–(39) and the learning parameter $\lambda_c(n)$ according to fuzzy logic given in Table 1.

Step 4: The actor element will update its parameters according to (40)–(42) and the learning parameter $\lambda_a(n)$ according to fuzzy logic given in Table 1.

Step 5: Steps (1) to (4) are repeated in each sampling time step until the end of the simulation.

5.1 Case study 1

Consider a nonaffine nonlinear system defined as [56]:

$$x_1(n+1) = m_1(n)\,x_2(n) + m_2(n)\ \sin(x_1(n)), \tag{46}$$

$$\begin{aligned} x_2(n+1) = m_3(n)\cos(x_2(n))\ \sin(x_1(n)) + m_4(n)\,u(n) \\ + m_5(n)\ \tanh(u(n)) + d_m(n), \end{aligned} \tag{47}$$

$$y(n+1) = x_1(n+1), \tag{48}$$

where the parameters are set as $m_1(n) = 0.5$, $m_2(n) = -0.3$, $m_3(n) = -1$, $m_4(n) = 2$, $m_5(n) = -2$ and $d_m(n) = 0$.

5.1.1 Task 1—effect due to variation of desired output

Figure 7 shows the system response when the desired output changes, which is indicated by black line. The proposed RIT2-TSK-FNN-RL controller has a smaller time for tracking the reference trajectory than the RIT2-TSK-FNN controller due to the critic network and the adaptation based on the LM algorithm with adaptive learning rate that can enforce the actor parameters to converge quickly. The number of the generated rules for both controllers is $M = 1$. This number of the rules is small due to the recurrent in the input and the rule layers.

5.1.2 Task 2—variation of the system parameters

This task is carried out after the system output tracks the reference trajectory. The actual values of the system parameters are changed to $m_1(n) = 0.7$, $m_2(n) = -0.5$, $m_3(n) = 1$, $m_4(n) = 3$ and $m_5(n) = -2.5$ at $n = 500$th instant. Then, it changed again to $m_1(n) = 0.2$, $m_2(n) = -0.7$, $m_3(n) = -1.5$, $m_4(n) = 2.2$ and $m_5(n) = -3$ at $n =$ 1000th instant. These changes in system parameters are used to show the robustness of the controllers. Figure 8 shows that the response of the proposed RIT2-TSK-FNN-RL controller has a smaller settling time than that of the RIT2-TSK-FNN controller at the two changes. The number of the generated rules for both controllers is $M = 1$.

5.1.3 Task 3—disturbance uncertainty

The performance evaluation of the proposed RIT2-TSK-FNN-RL controller is tested by adding the disturbance value $d_m(n)$ to (47) at $n = 750$th instant, which is given as:

$$d_m(n) = -\left(0.3x_1^3(n) + 0.05x_2^4(n)\right)\cos(10nT), \tag{49}$$

where T is the sampling period that equals 0.001.

Figure 9 shows that the proposed RIT2-TSK-FNN-RL controller tracks the trajectory after adding the disturbance but the RIT2-TSK-FNN has a large oscillation about the set point. The number of the generated rules for both controllers is $M = 1$.

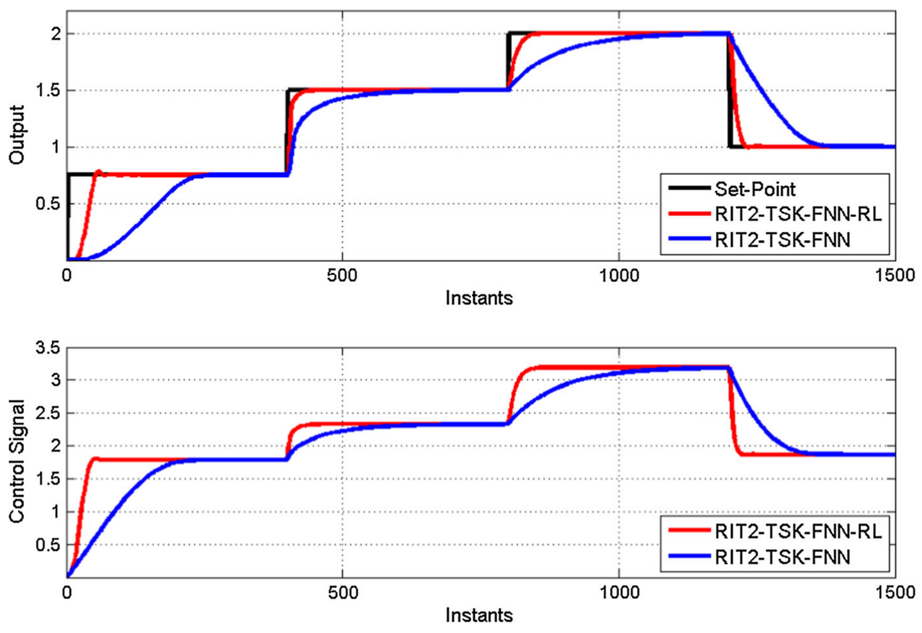


Fig. 7 Response of the tracking reference signal

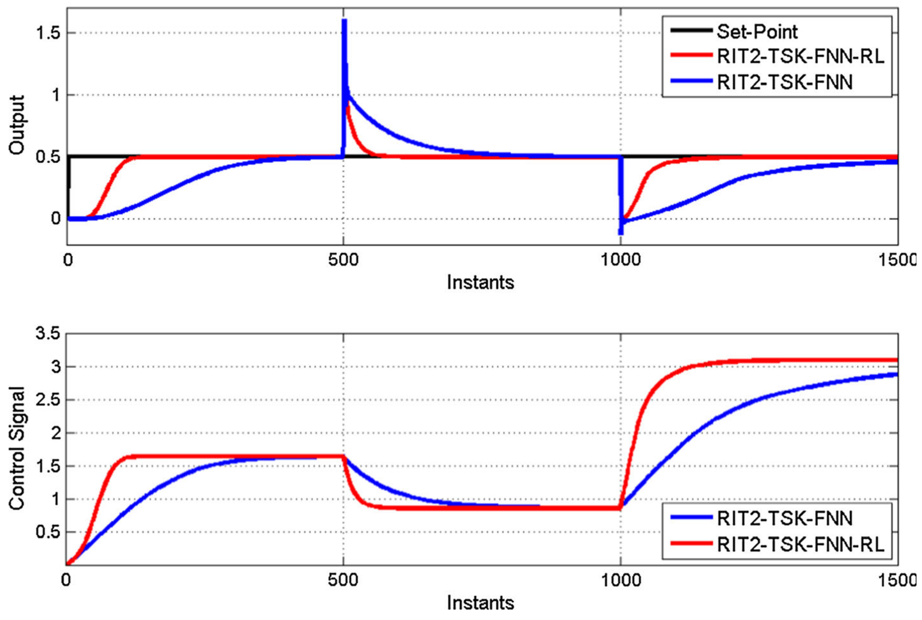
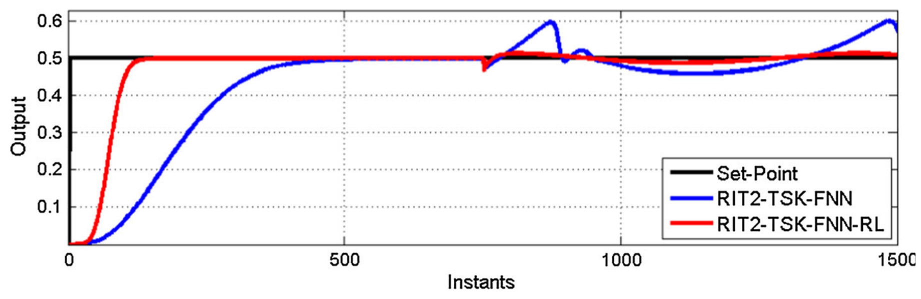


Fig. 8 Response of the system for uncertainty in the system parameters

5.1.4 Task 4—actuator noise

In this task, the performance of the proposed RIT2-TSK-FNN-RL controller is evaluated by adding the noise to the control signal at $n = 750$th instant. Figure 10 shows that the proposed RIT2-TSK-FNN-RL controller tracks the trajectory after adding the noise, but the RIT2-TSK-FNN has a large oscillation about the set point. The number of the obtained rules for both controllers is $M = 1$.

Tables 2 and 3 show the MAE and the IAE, respectively, for the proposed RIT2-TSK-FNN-RL controller, the RIT2-TSK-FNN controller and other controllers, which are published previously such as the FFANN [28–30], the ADPACRN [32], the GRHDP [33] and the ATSFRL-LS [35]. The results of the above simulation tasks are repeated using the average of 15 experiments. It is clear that the values of the MAE and IAE for the proposed RIT2-TSK-FNN-RL controller are smaller than those obtained for the RIT2-TSK-FNN, which are not used in the critic network. On the other hand, the values of the performance indices for the proposed controller, which depend on the RFNN and actor–critic learning scheme, are lower than those obtained for other controllers such as FFANN [28–30], ADPACRN [32] and GRHDP [33] which depend on the ANN and gradient descent method for the adaptation of the parameters. Also, the proposed controller is better than the



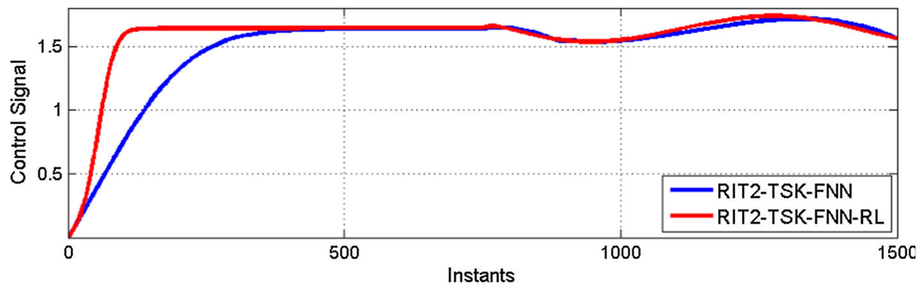
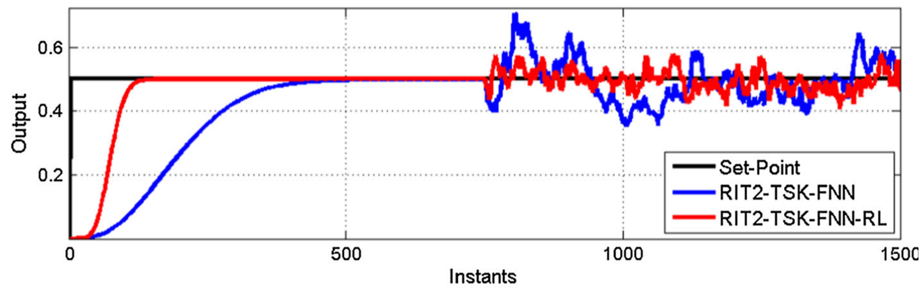


Fig. 9 Response of the system due to disturbance



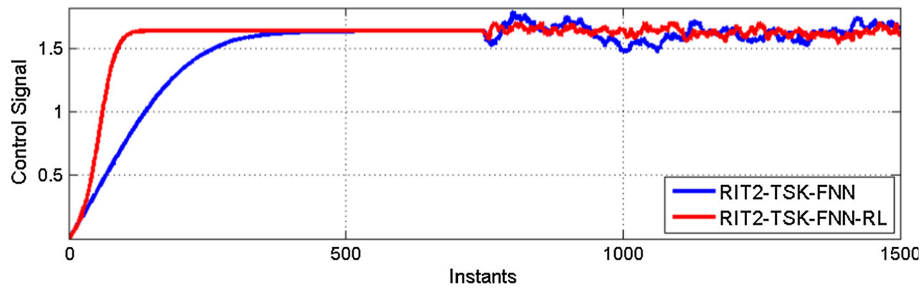


Fig. 10 Response of the system due to noise

Table 2 Average MAE values of 15 experiments

	Task 1	Task 2	Task 3	Task 4
RIT2-TSK-FNN-RL	0.0321	0.0460	0.0288	0.0388
RIT2-TSK-FNN	0.1578	0.1687	0.0806	0.0936
FFANN [28–30]	0.1605	0.1815	0.0647	0.0866
ADPACRN [32]	0.1568	0.1754	0.0659	0.0890
GRHDP [33]	0.1498	0.1729	0.0620	0.0870
ATSFRL-LS [35]	0.0973	0.1527	0.1126	0.1116

Table 3 Average IAE values of 15 experiments

	Task 1	Task 2	Task 3	Task 4
RIT2-TSK-FNN-RL	0.0481	0.0689	0.0431	0.0581
RIT2-TSK-FNN	0.2365	0.2529	0.1209	0.1404
FFANN [28–30]	0.2460	0.2721	0.0971	0.1299
ADPACRN [32]	0.2351	0.2630	0.0987	0.1335
GRHDP [33]	0.2245	0.2592	0.0929	0.1305
ATSFRL-LS [35]	0.1459	0.2289	0.1688	0.1673

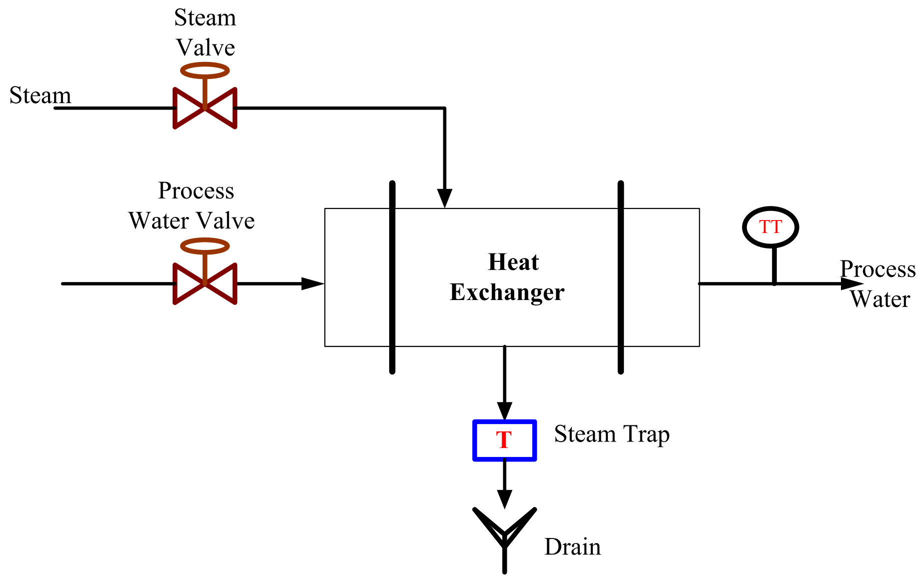


Fig. 11 Heat exchanger

ATSFRL-LS [35], which was implemented by fuzzy logic system and Lyapunov criteria for deriving the law of parameter adaptation.

5.2 Case study 2

One of the universal elements in the process and chemical industry is the steam-water heat exchanger in which the temperature control is very important task, especially when the process is opened over a broad scale [57, 58]. The steam-water heat exchanger is described in Fig. 11. The two inputs are the input process water flow and the steam flow rate that can be controlled by two pneumatic control values. The steam condenses in the two-pass shell and tube heat exchanger, hence raising the process water temperature. The exchanger has a nonlinear behavior when using a fix steam flow rate [59]. Accordingly, the process input is the input flow rate while the output is the temperature of the process output fluid, which is measured by thermocouple sensor and the steam flow is considered constant. The complex behavior of the steam-water heat exchanger can be represented by

$$y(n) = a_1(n)y(n-1) + a_2(n)y(n-2) + a_3(n)\xi(n-1) + a_4(n)\xi(n-2) + 0.005r \text{ and } (1), \tag{50}$$

$$\xi(n) = u(n) + a_5(n)u^2(n) + a_6(n)u^3(n) + a_7(n)u^4(n), \tag{51}$$

where the parameters are set as $a_1(n) = 1.608$, $a_2(n) = -0.6385$, $a_3(n) = -6.5306$, $a_4(n) = 5.5652$, $a_5(n) = -1.3228$, $a_6(n) = 0.7671$ and $a_7(n) = -2.1755$.

The models shown in (50) and (51) are derived using real data from the practical system as described in [57–59]. So, these equations are used in this paper, which simulate this system. Furthermore, the control simulation is based on measurement noise.

5.2.1 Task 1—effect due to variation of desired output

Figure 12 shows the heat exchanger process response when the desired trajectory signal is described as a black line. The proposed RIT2-TSK-FNN-RL controller has acceptable set-point tracking, which is realized with a rise time less than that obtained for the RIT2-TSK-FNN controller. The number of the obtained rules for both controllers is $M = 2$. The recurrent in the rule layers has a major impact for decreasing the number of rules that contribute to the nonlinearities besides the recurrent in the input, which deals with the measurement noise.

5.2.2 Task 2—process parameters uncertainty

Here, the robustness of the performance using the proposed RIT2-TSK-FNN-RL controller is evaluated under process parameters variations. The system response is shown in Fig. 13, in which the process parameters are changed at instant $n = 500$th to $a_1(n) = 1.64$, $a_2(n) = -0.8$,$a_3(k) = -7.4$, $a_4(n) = 6.4$, $a_5(n) = -1.8$, $a_6(n) = 0.47$ and $a_7(n) = -1.8$. It is clear that the proposed controller has a smaller settling time than RIT2-TSK-FNN controller. The RIT2-TSK-FNN controller has a large oscillation after applying this uncertainty. The number of the obtained rules for the proposed controller and the RIT2-TSK-FNN controller is $M = 2$ and $M = 5$, respectively.

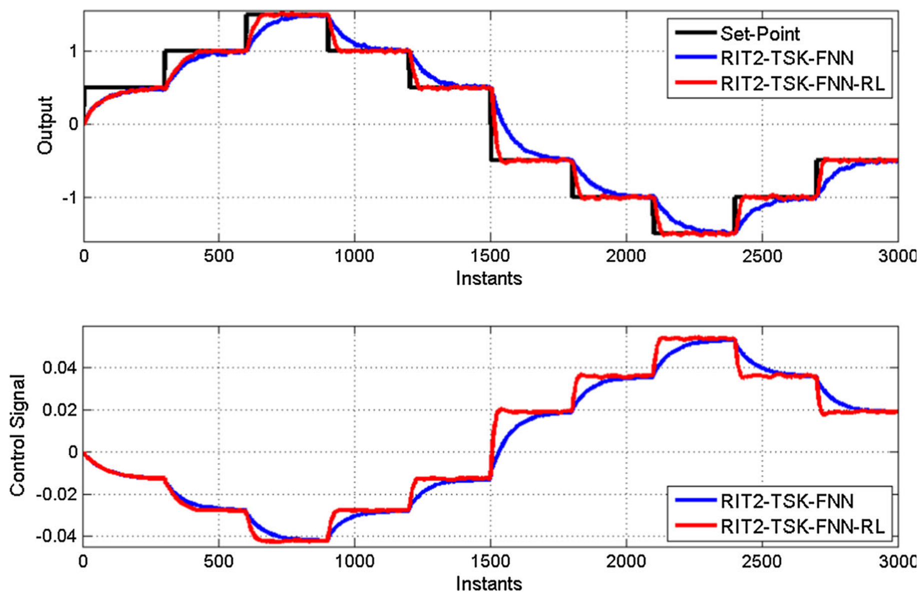


Fig. 12 Response of the tracking reference signal for heat exchanger process

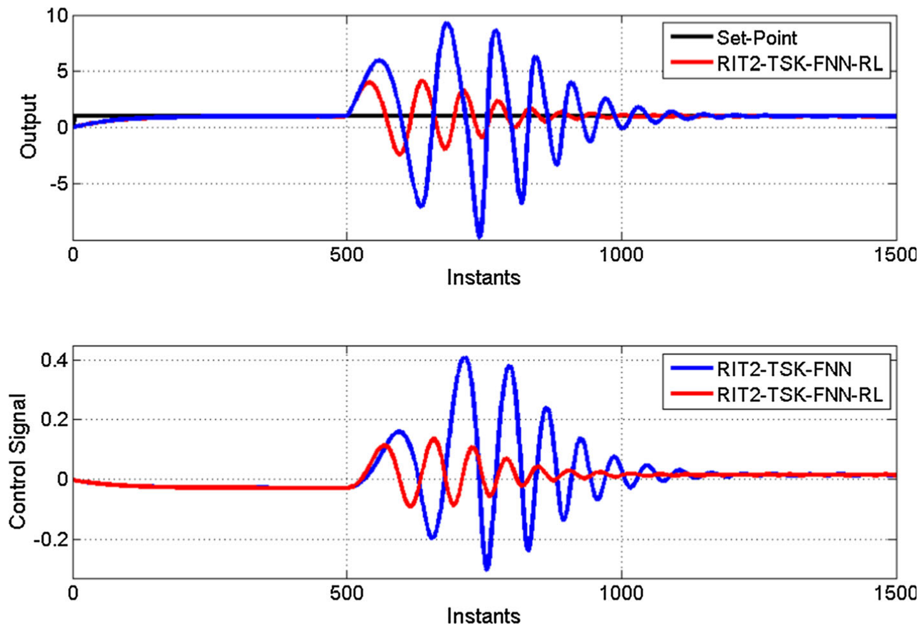


Fig. 13 Response of the process under parameters uncertainty

5.2.3 Task 3—sensor measurement uncertainty

In this task, the performance evaluation of the proposed RIT2-TSK-FNN-RL controller is tested by adding the sensor measurement uncertainty value $ds(n)$ to (50) at $n =$ 500th instant, which is given as:

$$ds(n) = -0.1\sin(y(n) - 0.1y(n-1)). \tag{52}$$

Figure 14 shows that the proposed RIT2-TSK-FNN-RL controller tracks the trajectory after adding the sensor measurement uncertainty and it has a stronger anti-interference ability than the RIT2-TSK-FNN controller. The number of the generated rules for both controllers is $M = 2$.

5.2.4 Task 4—actuator failure due to noise

The performance of the proposed RIT2-TSK-FNN-RL controller is evaluated by adding the noise to the control signal at $n =$ 500th instant. Figure 15 shows that the proposed RIT2-TSK-FNN-RL controller has an actuator noise rejection, which tracks the trajectory after adding the noise. However, the RIT2-TSK-FNN has a large error about the set point after adding the actuator noise. The number of the obtained rules for both controllers is $M = 2$.

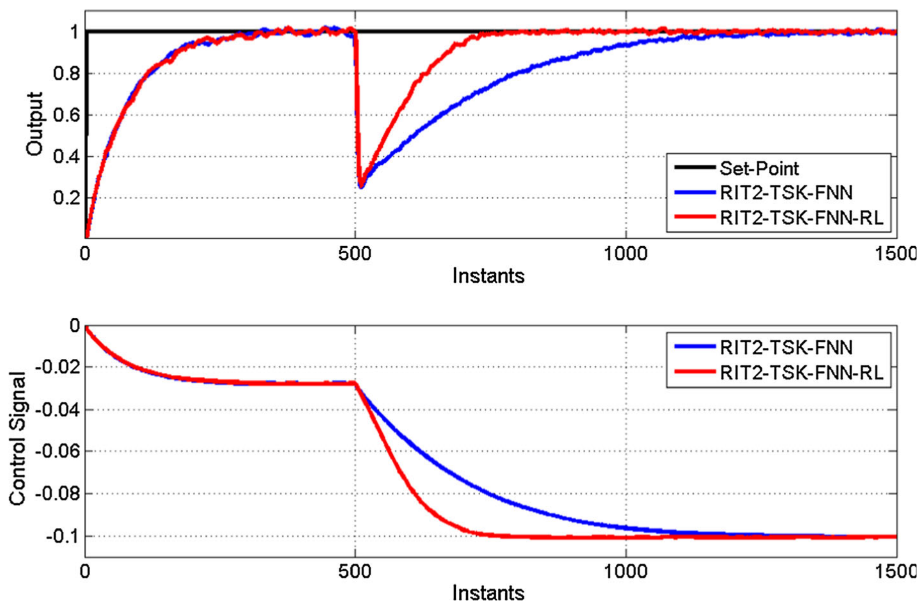


Fig. 14 Response of the process under sensor measurement uncertainty

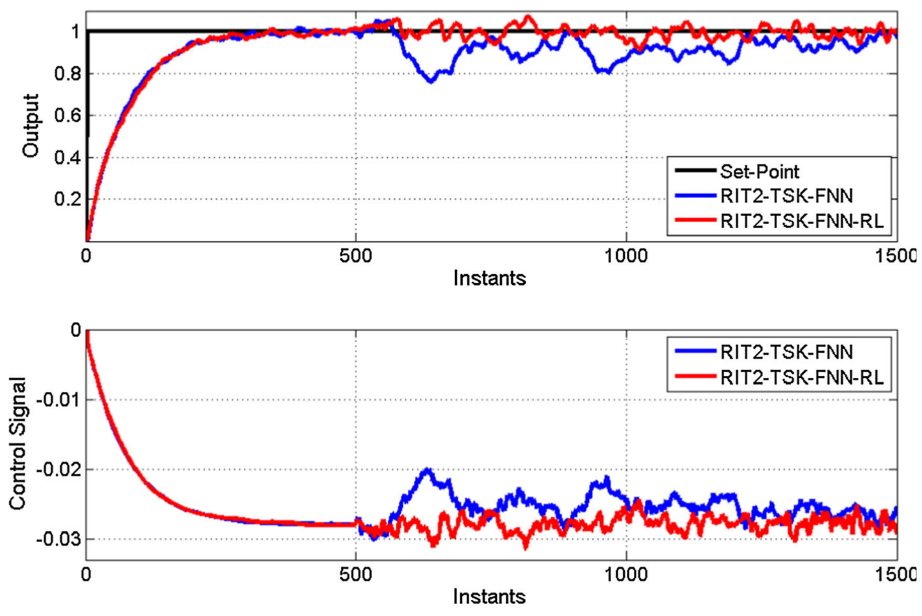


Fig. 15 Response of the process under actuator noise uncertainty

Table 4 Average MAE values of 15 experiments

	Task 1	Task 2	Task 3	Task 4
RIT2-TSK-FNN-RL	0.0469	0.4846	0.1025	0.0677
RIT2-TSK-FNN	0.1259	1.3129	0.1582	0.0835
FFANN [28–30]	0.1491	0.9058	0.16080	0.0843
ADPACRN [32]	0.1199	0.8770	1503	0.0768
GRHDP [33]	0.1234	0.8718	0.1466	0.0766
ATSFRL-LS [35]	0.1142	0.7825	0.1472	0.1165

Table 5 Average IAE values of 15 experiments

	Task 1	Task 2	Task 3	Task 4
RIT2-TSK-FNN-RL	0.1406	0.7264	0.1536	0.1015
RIT2-TSK-FNN	0.3774	1.9681	0.2372	0.1253
FFANN [28–30]	0.4471	1.3588	0.2411	0.1265
ADPACRN [32]	0.3597	1.3158	0.2253	0.1152
GRHDP [33]	0.3701	1.3082	0.2198	0.1149
ATSFRL-LS [35]	0.3427	1.2431	0.2208	0.1747

Table 6 Computation time of all algorithms

	Computation time (ms)
RIT2-TSK-FNN-RL	0.5248
RIT2-TSK-FNN	0.3481
FFANN [28–30]	0.1178
ADPACRN [32]	0.1897
GRHDP [33]	0.1938
ATSFRL-LS [35]	0.3276

Tables 4 and 5 show the MAE and the IAE for the proposed RIT2-TSK-FNN-RL controller, the RIT2-TSK-FNN and other controllers that are published previously such as the FFANN [28–30], the ADPACRN [32], the GRHDP [33] and the ATSFRL-LS [35], respectively. The results of the above simulation tasks are repeated using the average of 15 experiments. It is clear that the values of the MAE and the IAE for the proposed RIT2-TSK-FNN-RL controller are smaller than those obtained for the RIT2-TSK-FNN, which are not used in the critic network. On the other hand, the values of the performance indices for the proposed controller are lower than those obtained for other controllers such as FFANN [28–30], ADPACRN [32], GRHDP [33] and ATSFRL-LS [35].

The proposed controller and other controllers are performed on a PC, which has a processor Intel(R), Core(TM) i5-250 M with CPU @ 2.5GHZ, RAM 4.0 GB, 64-bit operating system and Windows 10. The computation time for all controllers is indicated in Table 6.

Remark 3 Although the proposed AC-IT2-TSK-FNN has a larger computation time than the other controllers, it has a better performance when the controlled system has uncertainties such as environmental noise, external disturbance and parameter uncertainties.

6 Conclusion

In this paper, the online learning for a novel structure of the RIT2-TSK-FNN based on RL scheme is proposed for controlling nonlinear systems. The LM algorithm with adaptive learning rate is developed for updating the parameters of the proposed controller. The stability conditions for the learning rates are achieved using the Lyapunov function. The output of the proposed RL controller forced the system to follow the reference input with one rule, which means that the proposed scheme has small parameters. The rule reduction was due to using the recurrent in the input layer and the firing layer. To evaluate the performance of the proposed controller, it is compared with the results of the RIT2-TSK-FNN controller and other published controllers. The proposed controller is tested using the mathematical nonlinear simulation system and the heat exchanger process with noisy measurement data. The results showed the superiority of the proposed controller to respond to the system uncertainties rather than the other controllers. The main advantages of proposed controller can be summarized as follows: (1) It has fast learning due to using reinforcement actor–critic method. (2) It has ability to handle the system uncertainties and noisy measurement data. (3) The number of the generated rules is small due to the recurrent in the input and the rule layers. (4) The learning rates are changed online according to fuzzy logic during the implementation of the algorithm to assure the stability (assuring that the Hessian matrix can be inverted) and the speed of the convergence. (5) The stability conditions are discussed using Lyapunov criteria. In future work, the authors will use a hierarchical deep RL framework.

Compliance with ethical standards

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

Appendix A

The term $\frac{\partial O^{(6)}(n)}{\partial \theta_a(n)}$ for the actor network is described by the following equations:

$$\frac{\partial O^{(6)}(n)}{\partial c_j^i(n)} = \frac{\partial O^{(6)}(n)}{\partial \underline{O}^{(5)}(n)} \frac{\partial \underline{O}^{(5)}(n)}{\partial \underline{O}_i^{(4)}(n)} \frac{\partial \underline{O}_i^{(4)}(n)}{\partial c_j^i(n)} + \frac{\partial O^{(6)}(n)}{\partial \bar{O}^{(5)}(n)} \frac{\partial \bar{O}^{(5)}(n)}{\partial \bar{O}^{(4)}(n)} \frac{\partial \bar{O}^{(4)}(n)}{\partial c_j^i(n)},$$
$$= \frac{(1 - H_l + H_r) \cdot \bar{O}_i^{(3)}(n) + (1 - H_r + H_l)\underline{O}_i^{(3)}(n)}{\sum_{i=1}^{M} \underline{O}_i^{(3)}(n) + \bar{O}_i^{(3)}(n)} \cdot O_j^{(1)}(n), \tag{53}$$

$$\frac{\partial O^{(6)}(n)}{\partial s_j^i(n)} = \frac{\partial O^{(6)}(n)}{\partial \underline{O}^{(5)}(n)} \frac{\partial \underline{O}^{(5)}(n)}{\partial \underline{O}_i^{(4)}(n)} \frac{\partial \underline{O}_i^{(4)}(n)}{\partial s_j^i(n)} + \frac{\partial O^{(6)}(n)}{\partial \bar{O}^{(5)}(n)} \frac{\partial \bar{O}^{(5)}(n)}{\partial \bar{O}^{(4)}(n)} \frac{\partial \bar{O}^{(4)}(n)}{\partial s_j^i(n)},$$
$$= \frac{(1 - H_r - H_l) \cdot \underline{O}_i^{(3)}(n) - (1 - H_l - H_r) \cdot \bar{O}_i^{(3)}(n)}{\sum_{i=1}^{M} \underline{O}_i^{(3)}(n) + \bar{O}_i^{(3)}(n)} \cdot \left| O_j^{(1)}(n) \right|, \tag{54}$$

$$\frac{\partial O^{(6)}(n)}{\partial m_{j1}^i(n)} = \left(\frac{\partial O^{(6)}(n)}{\partial \underline{O}^{(5)}(n)} \frac{\partial \underline{O}^{(5)}(n)}{\partial \underline{O}_i^{(3)}(n)} \right) \frac{\partial \underline{O}_i^{(3)}(n)}{\partial m_{j1}^i(n)} + \left(\frac{\partial O^{(6)}(n)}{\partial \bar{O}^{(5)}(n)} \frac{\partial \bar{O}^{(5)}(n)}{\partial \bar{O}^{(3)}(n)} \right) \frac{\partial \bar{O}^{(3)}(n)}{\partial m_{j1}^i(n)}, \tag{55}$$

$$\frac{\partial O^{(6)}(n)}{\partial m^i_{j2}(n)}=\left(\frac{\partial O^{(6)}(n)}{\partial \underline{O}^{(5)}(n)}\frac{\partial \underline{O}^{(5)}(n)}{\partial \underline{O}^{(3)}_i(n)}\right)\frac{\partial \underline{O}^{(3)}_i(n)}{\partial m^i_{j2}(n)}+\left(\frac{\partial O^{(6)}(n)}{\partial \bar{O}^{(5)}(n)}\frac{\partial \bar{O}^{(5)}(n)}{\partial \bar{O}^{(3)}(n)}\right)\frac{\partial \bar{O}^{(3)}(n)}{\partial m^i_{j2}(n)},\tag{56}$$

$$\frac{\partial O^{(6)}(n)}{\partial \sigma^i_j(n)}=\left(\frac{\partial O^{(6)}(n)}{\partial \underline{O}^{(5)}(n)}\frac{\partial \underline{O}^{(5)}(n)}{\partial \underline{O}^{(3)}_i(n)}\right)\frac{\partial \bar{O}^{(3)}_i(n)}{\partial \sigma^i_j(n)}+\left(\frac{\partial O^{(6)}(n)}{\partial \bar{O}^{(5)}(n)}\frac{\partial \bar{O}^{(5)}(n)}{\partial \bar{O}^{(3)}(n)}\right)\frac{\partial \bar{O}^{(3)}(n)}{\partial \sigma^i_j(n)}\tag{57}$$

$$\frac{\partial O^{(6)}(n)}{\partial l^i(n)}=\left(\frac{\partial O^{(6)}(n)}{\partial \underline{O}^{(5)}(n)}\frac{\partial \underline{O}^{(5)}(n)}{\partial \underline{O}^{(3)}_i(n)}+\frac{\partial O^{(6)}(n)}{\partial \bar{O}^{(5)}(n)}\frac{\partial \bar{O}^{(5)}(n)}{\partial \underline{O}^{(3)}_i(n)}\right)\frac{\partial \underline{O}^{(3)}_i(n)}{\partial l^i(n)}+\left(\frac{\partial O^{(6)}(n)}{\partial \bar{O}^{(5)}(n)}\frac{\partial \bar{O}^{(5)}(n)}{\partial \bar{O}^{(3)}(n)}+\frac{\partial O^{(6)}(n)}{\partial \underline{O}^{(5)}(n)}\frac{\partial \underline{O}^{(5)}(n)}{\partial \bar{O}^{(3)}(n)}\right)\frac{\partial \bar{O}^{(3)}(n)}{\partial l^i(n)},\tag{58}$$

$$\frac{\partial O^{(6)}(n)}{\partial \alpha_j(n)}=\left(\frac{\partial O^{(6)}(n)}{\partial \underline{O}^{(5)}(n)}\frac{\partial \underline{O}^{(5)}(n)}{\partial \underline{O}^{(3)}_i(n)}+\frac{\partial O^{(6)}(n)}{\partial \bar{O}^{(5)}(n)}\frac{\partial \bar{O}^{(5)}(n)}{\partial \underline{O}^{(3)}_i(n)}\right)\frac{\partial \underline{O}^{(3)}_i(n)}{\partial \alpha_j(n)}+\left(\frac{\partial O^{(6)}(n)}{\partial \bar{O}^{(5)}(n)}\frac{\partial \bar{O}^{(5)}(n)}{\partial \bar{O}^{(3)}(n)}+\frac{\partial O^{(6)}(n)}{\partial \underline{O}^{(5)}(n)}\frac{\partial \underline{O}^{(5)}(n)}{\partial \bar{O}^{(3)}(n)}\right)\frac{\partial \bar{O}^{(3)}(n)}{\partial \alpha_j(n)},\tag{59}$$

$$\frac{\partial O^{(6)}(n)}{\partial H_l(n)}=\frac{\partial O^{(6)}(n)}{\partial \underline{O}^{(5)}(n)}\frac{\partial \underline{O}^{(5)}(n)}{\partial H_l(n)}=\frac{\sum_{i=1}^{M}\underline{O}^{(4)}_i(n)\left(\underline{O}^{(3)}_i(n)-\bar{O}^{(3)}_i(n)\right)}{\sum_{i=1}^{M}\underline{O}^{(3)}_i(n)+\bar{O}^{(3)}_i(n)},\tag{60}$$

$$\frac{\partial O^{(6)}(n)}{\partial H_r(n)}=\frac{\partial O^{(6)}(n)}{\partial \bar{O}^{(5)}(n)}\frac{\partial \bar{O}^{(5)}(n)}{\partial H_r(n)}=\frac{\sum_{i=1}^{M}\bar{O}^{(4)}_i(n)\left(\underline{O}^{(3)}_i(n)+\bar{O}^{(3)}_i(n)\right)}{\sum_{i=1}^{M}\underline{O}^{(3)}_i(n)+\bar{O}^{(3)}_i(n)}.\tag{61}$$

The derivatives in (55)–(61) are defined as:

$$\frac{\partial \underline{O}^{(5)}(n)}{\partial \underline{O}^{(3)}_i(n)}=\frac{H_l\underline{O}^{(4)}_i(n)-\underline{O}^{(5)}(n)}{\sum_{i=1}^{M}\underline{O}^{(3)}_i(n)+\bar{O}^{(3)}_i(n)},\tag{62}$$

$$\frac{\partial \underline{O}^{(5)}(n)}{\partial \bar{O}^{(3)}_i(n)}=\frac{(1-H_l)\underline{O}^{(4)}_i(n)-\underline{O}^{(5)}(n)}{\sum_{i=1}^{M}\underline{O}^{(3)}_i(n)+\bar{O}^{(3)}_i(n)},\tag{63}$$

$$\frac{\partial \bar{O}^{(5)}(n)}{\partial \underline{O}^{(3)}_i(n)}=\frac{(1-H_r)\bar{O}^{(4)}_i(n)-\bar{O}^{(5)}(n)}{\sum_{i=1}^{M}\underline{O}^{(3)}_i(n)+\bar{O}^{(3)}_i(n)},\tag{64}$$

$$\frac{\partial \bar{O}^{(5)}(n)}{\partial \bar{O}^{(3)}_i(n)}=\frac{H_r\bar{O}^{(4)}_i(n)-\bar{O}^{(5)}(n)}{\sum_{i=1}^{M}\underline{O}^{(3)}_i(n)+\bar{O}^{(3)}_i(n)},\tag{65}$$

$$\frac{\partial \bar{O}^{(3)}(n)}{\partial m^i_{j1}(n)}=\frac{\partial \bar{O}^{(3)}(n)}{\partial \bar{\Omega}^i(n)}\frac{\partial \bar{\Omega}^i(n)}{\partial \bar{O}^{(2)}_{ij}(n)}\frac{\partial \bar{O}^{(2)}_{ij}(n)}{\partial m^i_{j1}(n)}=\begin{cases} l^i\bar{\Omega}^i(n)\dfrac{O^1_j(n)-m^i_{j1}(n)}{\left(\sigma^i_j(n)\right)^2}, & O^1_j(n)<m^i_{j1}(n)\\ 0, & \text{otherwise}\end{cases},\tag{66}$$

$$\frac{\partial \underline{O}^{(3)}(n)}{\partial m^i_{j1}(n)}=\frac{\partial \underline{O}^{(3)}(n)}{\partial \underline{\Omega}^i(n)}\frac{\partial \underline{\Omega}^i(n)}{\partial \underline{O}^{(2)}_{ij}(n)}\frac{\partial \underline{O}^{(2)}_{ij}(n)}{\partial m^i_{j1}(n)}=\begin{cases} l^i\underline{\Omega}^i(n)\dfrac{O^1_j(n)-m^i_{j1}(n)}{\left(\sigma^i_j(n)\right)^2}, & O^1_j(n)>\dfrac{m^i_{j1}(n)+m^i_{j2}(n)}{2}\\ 0, & \text{otherwise}\end{cases},\tag{67}$$

$$\frac{\partial \bar{O}^{(3)}(n)}{\partial m^i_{j2}(n)}=\frac{\partial \bar{O}^{(3)}(n)}{\partial \bar{\Omega}^i(n)}\frac{\partial \bar{\Omega}^i(n)}{\partial \bar{O}^{(2)}_{ij}(n)}\frac{\partial \bar{O}^{(2)}_{ij}(n)}{\partial m^i_{j2}(n)}=\begin{cases} l^i\bar{\Omega}^i(n)\dfrac{O^1_j(n)-m^i_{j2}(n)}{\left(\sigma^i_j(n)\right)^2}, & O^1_j(n)>m^i_{j2}(n)\\ 0, & \text{otherwise}\end{cases},\tag{68}$$

$$\frac{\partial \underline{O}^{(3)}(n)}{\partial m^i_{j2}(n)}=\frac{\partial \underline{O}^{(3)}(n)}{\partial \underline{\Omega}^i(n)}\frac{\partial \underline{\Omega}^i(n)}{\partial \underline{O}^{(2)}_{ij}(n)}\frac{\partial \underline{O}^{(2)}_{ij}(n)}{\partial m^i_{j2}(n)}=\begin{cases} l^i\underline{\Omega}^i(n)\dfrac{O^1_j(n)-m^i_{j2}(n)}{\left(\sigma^i_j(n)\right)^2}, & O^1_j(n)\le\dfrac{m^i_{j1}(n)+m^i_{j2}(n)}{2}\\ 0, & \text{otherwise}\end{cases},\tag{69}$$

$$\frac{\partial \bar{O}^{(3)}(n)}{\partial \sigma^i_j(n)}=\frac{\partial \bar{O}^{(3)}(n)}{\partial \bar{\Omega}^i(n)}\frac{\partial \bar{\Omega}^i(n)}{\partial \bar{O}^{(2)}_{ij}(n)}\frac{\partial \bar{O}^{(2)}_{ij}(n)}{\partial \sigma^i_j(n)}=\begin{cases} l^i\bar{\Omega}^i(n)\dfrac{\left(O^1_j(n)-m^i_{j1}(n)\right)^2}{\left(\sigma^i_j(n)\right)^3}, & O^1_j(n)<m^i_{j1}(n)\\ l^i\bar{\Omega}^i(n)\dfrac{\left(O^1_j(n)-m^i_{j2}(n)\right)^2}{\left(\sigma^i_j(n)\right)^3}, & O^1_j(n)<m^i_{j2}(n)\\ 0, & \text{otherwise}\end{cases},\tag{70}$$

$$\frac{\partial \underline{O}^{(3)}(n)}{\partial \sigma_j^i(n)} = \frac{\partial \underline{O}^{(3)}(n)}{\partial \underline{\Omega}^i(n)} \frac{\partial \underline{\Omega}^i(n)}{\partial \underline{O}_{ij}^{(2)}(n)} \frac{\partial \underline{O}_{ij}^{(2)}(n)}{\partial \sigma_j^i(n)}$$
$$= \begin{cases} l^i \underline{\Omega}^i(n) \dfrac{\left(O_j^1(n) - m_{j2}^i(n)\right)^2}{\left(\sigma_j^i(n)\right)^3}, & O_j^1(n) \le \dfrac{m_{j1}^i(n) + m_{j2}^i(n)}{2} \\ l^i \underline{\Omega}^i(n) \dfrac{\left(O_j^1(n) - m_{j1}^i(n)\right)^2}{\left(\sigma_j^i(n)\right)^3}, & O_j^1(n) > \dfrac{m_{j1}^i(n) + m_{j2}^i(n)}{2} \end{cases}, \tag{71}$$

$$\frac{\partial \underline{O}_i^{(3)}(n)}{\partial l^i(n)} = \underline{\Omega}^i(n) - \underline{O}_i^{(3)}(n-1), \frac{\partial \bar{O}_i^{(3)}(n)}{\partial l^i(n)} = \bar{\Omega}^i(n) - \bar{O}_i^{(3)}(n-1), \tag{72}$$

$$\frac{\partial \bar{O}^{(3)}(n)}{\partial \alpha_j(n)} = \frac{\partial \bar{O}^{(3)}(n)}{\partial \bar{\Omega}^i(n)} \frac{\partial \bar{\Omega}^i(n)}{\partial \bar{O}_{ij}^{(2)}(n)} \frac{\partial \bar{O}_{ij}^{(2)}(n)}{\partial O_j^1(n)} \frac{\partial O_j^1(n)}{\partial \alpha_j(n)}$$
$$= \begin{cases} -l^i \bar{\Omega}^i(n) O_j^1(n-1) \dfrac{\left(O_j^1(n) - m_{j1}^i(n)\right)^2}{\left(\sigma_j^i(n)\right)^3}, & O_j^1(n) < m_{j1}^i(n) \\ -l^i \bar{\Omega}^i(n) O_j^1(n-1) \dfrac{\left(O_j^1(n) - m_{j2}^i(n)\right)^2}{\left(\sigma_j^i(n)\right)^3}, & O_j^1(n) < m_{j2}^i(n) \\ 0, & \text{otherwise} \end{cases}, \tag{73}$$

$$\frac{\partial \underline{O}^{(3)}(n)}{\partial \alpha_j(n)} = \frac{\partial \underline{O}^{(3)}(n)}{\partial \underline{\Omega}^i(n)} \frac{\partial \underline{\Omega}^i(n)}{\partial \underline{O}_{ij}^{(2)}(n)} \frac{\partial \underline{O}_{ij}^{(2)}(n)}{\partial O_j^1(n)} \frac{\partial O_j^1(n)}{\partial \alpha_j(n)}$$
$$= \begin{cases} -l^i \underline{\Omega}^i(n) O_j^1(n-1) \dfrac{\left(O_j^1(n) - m_{j2}^i(n)\right)^2}{\left(\sigma_j^i(n)\right)^3}, & O_j^1(n) \le \dfrac{m_{j1}^i(n) + m_{j2}^i(n)}{2} \\ -l^i \underline{\Omega}^i(n) O_j^1(n-1) \dfrac{\left(O_j^1(n) - m_{j1}^i(n)\right)^2}{\left(\sigma_j^i(n)\right)^3}, & O_j^1(n) > \dfrac{m_{j1}^i(n) + m_{j2}^i(n)}{2} \end{cases}. \tag{74}$$

Appendix B

Proof Consider the following Lyapunov candidate:

$$V_g(n) = \frac{1}{2} e_g^2(n). \tag{75}$$

□

For stable training algorithm, $\Delta V_g(n)$ should be less than zero. Hence, the $\Delta V_g(n)$ is calculated as the following equation:

$$\begin{aligned} \Delta V_g(n) &= \frac{1}{2}\left(e_g^2(n+1) - e_g^2(n)\right) \\ &= \frac{1}{2}\left(\left(e_g(n) + \Delta e_g(n)\right)^2 - e_g^2(n)\right) \\ &= \Delta e_g(n)\left(e_g(n) + \frac{1}{2}\Delta e_g(n)\right). \end{aligned} \tag{76}$$

The $\Delta V_g(n)$ can be rewritten as:

$$\Delta V_g(n) = \left(\frac{\partial e_g(n)}{\partial \theta_g(n)}\right) \Delta\theta_g(n) \left(e_g(n) + \frac{1}{2}\left(\frac{\partial e_g(n)}{\partial \theta_g(n)}\right) \Delta\theta_g(n)\right). \tag{77}$$

Thus,

$$\left(\frac{\partial e_g(n)}{\partial \theta_g(n)}\right) \Delta\theta_g(n) = \left(\frac{\partial e_g(n)}{\partial \theta_g(n)}\right) \Phi \left(\frac{\partial e_g(n)}{\partial \theta_g(n)}\right)^{\mathrm{T}} e_g(n), \tag{78}$$

where

$$\Phi = \left(\left(\frac{\partial e_g(n)}{\partial \theta_g(n)}\right)^{\mathrm{T}} \frac{\partial e_g(n)}{\partial \theta_g(n)} + \lambda_g(n) I\right)^{-1}. \tag{79}$$

By using a matrix inversion lemma, we get [55]:

$$(E + FGH)^{-1} = E^{-1} - E^{-1}F\left(G^{-1} + HE^{-1}F\right)^{-1} HE^{-1}. \tag{80}$$

According to (80), (79) can be rewritten as:

$$\begin{aligned} \Phi &= \lambda_g^{-1}(n) I - \lambda_g^{-2}(n) \left(\frac{\partial e_g(n)}{\partial \theta_g(n)}\right)^{\mathrm{T}} \left(I + \frac{\partial e_g(n)}{\partial \theta_g(n)} \lambda_g^{-1}(n) \left(\frac{\partial e_g(n)}{\partial \theta_g(n)}\right)^{\mathrm{T}}\right)^{-1} \frac{\partial e_g(n)}{\partial \theta_g(n)} \\ &= \lambda_g^{-1}(n) I - \lambda_g^{-1}(n) \left(\frac{\partial e_g(n)}{\partial \theta_g(n)}\right)^{\mathrm{T}} \left(\lambda_g(n) I + \frac{\partial e_g(n)}{\partial \theta_g(n)} \left(\frac{\partial e_g(n)}{\partial \theta_g(n)}\right)^{\mathrm{T}}\right)^{-1} \frac{\partial e_g(n)}{\partial \theta_g(n)}. \end{aligned} \tag{81}$$

Using (81) and (78), we obtain

$$\begin{aligned} \left(\frac{\partial e_g(n)}{\partial \theta_g(n)}\right) \Delta\theta_g(n) &= \lambda_g^{-1}(n) \left\|\frac{\partial e_g(n)}{\partial \theta_g(n)}\right\|^2 e_g(n) \\ &\quad - \lambda_g^{-1}(n) \left\|\frac{\partial e_g(n)}{\partial \theta_g(n)}\right\|^4 \left(\lambda_g + \left\|\frac{\partial e_g(n)}{\partial \theta_g(n)}\right\|^2\right)^{-1} e_g(n). \end{aligned} \tag{82}$$

The time difference of the Lyapunov function $\Delta V_g(n)$ can be given as:

$$\begin{aligned} &\Delta V_g(n) \\ &= -\frac{1}{2} e_g^2(n) \left(\lambda_g^{-1}(n) \left\|\frac{\partial e_g(n)}{\partial \theta_g(n)}\right\|^2 - \lambda_g^{-1}(n) \left\|\frac{\partial e_g(n)}{\partial \theta_g(n)}\right\|^4 \left[\lambda_g(n) + \left\|\frac{\partial e_g(n)}{\partial \theta_g(n)}\right\|^2\right]^{-1}\right) \\ &\times \left(2 - \lambda_g^{-1}(n) \left\|\frac{\partial e_g(n)}{\partial \theta_g(n)}\right\|^2 + \lambda_g^{-1}(n) \left\|\frac{\partial e_g(n)}{\partial \theta_g(n)}\right\|^4 \left[\lambda_g(n) + \left\|\frac{\partial e_g(n)}{\partial \theta_g(n)}\right\|^2\right]^{-1}\right). \end{aligned} \tag{83}$$

Since

$$0 \leq \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^2, \tag{84}$$

thus

$$\lambda_g^{-1}(n) \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^4 \leq \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^2 + \lambda_g^{-1}(n) \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^4. \tag{85}$$

By multiplying both sides of (85) by $\left(\lambda_g(n) + \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^2 \right)^{-1}$, we get

$$\lambda_g^{-1}(n) \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^4 \left(\lambda_g(n) + \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^2 \right)^{-1} \leq \lambda_g(n) \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^2, \tag{86}$$

so that

$$0 \leq \lambda_g(n) \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\| - \lambda_g^{-1}(n) \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^4 \left(\lambda_g(n) + \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^2 \right)^{-1}. \tag{87}$$

Hence, in order that $\Delta V_g(n) \leq 0$,

$$0 \leq \left(2 - \lambda_g^{-1}(n) \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^2 + \lambda_g^{-1}(n) \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^4 \left(\lambda_g(n) + \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^2 \right)^{-1} \right). \tag{88}$$

Thus, the following constraint for the stability is

$$0 \leq 2 - \lambda_g^{-1}(n) \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^2, \tag{89}$$

which means

$$\lambda_g(n) \geq \frac{1}{2} \left\| \frac{\partial e_g(n)}{\partial \theta_g(n)} \right\|^2. \tag{90}$$

References

1. El-Nagar AM, El-Bardini M (2014) Practical realization for the interval type-2 fuzzy PD + I controller using a low-cost microcontroller. Arab J Sci Eng 39(8):6463–6476
2. Shaheen O, El-Nagar AM, El-Bardini M, El-Rabaie NM (2018) Probabilistic fuzzy logic controller for uncertain nonlinear systems. J Frankl Inst 355(3):1088–1106
3. El-Nagar AM, El-Bardini M (2016) Hardware-in-the-loop simulation of interval type-2 fuzzy PD controller for uncertain nonlinear system using low cost microcontroller. Appl Math Model 40(3):2346–2355
4. Sozhamadevi N, Sathiyamoorthy S (2015) A probabilistic fuzzy inference system for modeling and control of nonlinear process. Arab J Sci Eng 40(6):1777–1791
5. Chemachema M (2012) Output feedback direct adaptive neural network control for uncertain SISO nonlinear systems using a fuzzy estimator of the control error. Neural Netw 36:25–34
6. Rossomando FG, Soria CM (2017) Discrete-time sliding mode neuro-adaptive controller for SCARA robot arm. Neural Comput Appl 28(12):3837–3850
7. Wen G, Ge SS, Tu F (2018) Optimized backstepping for tracking control of strict-feedback systems. IEEE Trans Neural Netw Learn Syst 29(8):3850–3862
8. Aghababa MP (2016) Optimal design of fractional-order PID controller for five bar linkage robot using a new particle swarm optimization algorithm. Soft Comput 20(10):4055–4067
9. Kiumarsi B, Vamvoudakis KG, Modares H, Lewis FL (2018) Optimal and autonomous control using reinforcement learning: a survey. IEEE Trans Neural Netw Learn Syst 29(6):2042–2062
10. Liu YJ, Li S, Tong S, Chen CP (2018) Adaptive reinforcement learning control based on neural approximation for nonlinear discrete-time systems with unknown nonaffine dead-zone input. IEEE Trans Neural Netw Learn Syst 99:1–11
11. Khan SG, Herrmann G, Lewis FL, Pipe T, Melhuish C (2012) Reinforcement learning and optimal adaptive control: an overview and implementation examples. Annu Rev Control 36(1):42–59
12. Murray JJ, Cox CJ, Lendaris GG, Saeks R (2002) Adaptive dynamic programming. IEEE Trans Syst Man Cybern Part C (Appl Rev) 32(2):140–153
13. 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(8):2389–2406
14. Radac MB, Precup RE, Roman RC (2017) Model-free control performance improvement using virtual reference feedback tuning and reinforcement Q-learning. J Syst Sci 48(5):1071–1083
15. Boubertakh H, Tadjine M, Glorennec PY, Labiod S (2010) Tuning fuzzy PD and PI controllers using reinforcement learning. ISA Trans 49(4):543–551
16. Lewis FL, Liu D (2013) Reinforcement learning and approximate dynamic programming for feedback control. Wiley, Hoboken
17. Hendzel Z, Szuster M (2011) Discrete neural dynamic programming in wheeled mobile robot control. Commun Nonlinear Sci Numer Simul 16(5):2355–2362
18. Zhang J, Zhang H, Luo Y, Feng T (2014) Model-free optimal control design for a class of linear discrete-time systems with multiple delays using adaptive dynamic programming. Neurocomputing 135:163–170
19. Wang D, Liu D, Wei Q, Zhao D (2012) Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming. Automatica 48(8):1825–1832
20. Zhong X, He H, Zhang H, Wang Z (2015) A neural network based online learning and control approach for Markov jump systems. Neurocomputing 149:116–123
21. Maharajan C, Raja R, Cao J, Rajchakit G (2018) Novel global robust exponential stability criterion for uncertain inertial-type BAM neural networks with discrete and distributed time-varying delays via Lagrange sense. J Frankl Inst 355(11):4727–4754
22. Sowmiya C, Raja R, Cao J, Rajchakit G (2018) Enhanced result on stability analysis of randomly occurring uncertain parameters, leakage, and impulsive BAM neural networks with time-varying delays: discrete-time case. Int J Adapt Control Signal Process 32(7):1010–1039
23. Sowmiya C, Raj R, Cao J, Li X, Rajchakit G (2018) Discrete-time stochastic impulsive BAM neural networks with leakage and mixed time delays: an exponential stability problem. J Frankl Inst 355(10):4404–4435
24. Sowmiya C, Raja R, Cao J, Rajchakit G, Alsaedi A (2018) Exponential stability of discrete-time cellular uncertain BAM

neural networks with variable delays using halanay-type inequality. Appl Math Inf Sci 12(3):545–558

25. Sundara V, Raja R, Agarwal R, Rajchakit G (2018) A novel controllability analysis of impulsive fractional linear time invariant systems with state delay and distributed delays in control. Discontin Nonlinearity Complex 7(3):275–290
26. Saravanakumar R, Rajchakit G, Ahn CK, Karimi HR (2017) Exponential stability, passivity, and dissipativity analysis of generalized neural networks with mixed time-varying delays. IEEE Trans Syst Man Cybern Syst 49(2):395–405
27. Song R, Xiao W, Zhang H, Sun C (2014) Adaptive dynamic programming for a class of complex-valued nonlinear systems. IEEE Trans Neural Netw Learn Syst 25(9):1733–1739
28. Si J, Wang YT (2001) Online learning control by association and reinforcement. IEEE Trans Neural Netw 12(2):264–276
29. Huang X, Naghdy F, Du H, Naghdy G, Todd C (2015) Reinforcement learning neural network (RLNN) based adaptive control of fine hand motion rehabilitation robot. In: IEEE conference on control applications (CCA), pp 941–946
30. Shen H, Guo C (2016) Path-following control of underactuated ships using actor-critic reinforcement learning with MLP neural networks. In: IEEE conference information science and technology (ICIST), pp 317–321
31. Niedzwiedz C, Elhanany I, Liu Z, Livingston S (2008) A consolidated actor-critic model with function approximation for high-dimensional POMDPs. In: AAAI conference, pp 37–42
32. He H, Ni Z, Fu J (2012) A three-network architecture for on-line learning and optimization based on adaptive dynamic programming. Neurocomputing 78(1):3–13
33. Ni Z, Tang Y, Sui X, He H, Wen J (2016) An adaptive neurocontrol approach for multimachine power systems. Int J Electr Power Energy Syst 75:108–116
34. Lv Y, Na J, Ren X (2017) Online H∞ control for completely unknown nonlinear systems via an identifier–critic-based ADP structure. Int J Control 92:1–12
35. Khater AA, El-Nagar AM, El-Bardini M, El-Rabaie NM (2018) Adaptive TS fuzzy controller using reinforcement learning based on Lyapunov stability. J Frankl Inst 355(14):6390–6415
36. Fung RF, Lin FJ, Wai RJ, Lu PY (2000) Fuzzy neural network control of a motor-quick-return servomechanism. Mechatronics 10:145–167
37. Juang CF, Huang RB, Lin YY (2009) A recurrent self-evolving interval type-2 fuzzy neural network for dynamic system processing. IEEE Trans Fuzzy Syst 17(5):1092–1105
38. Lin CJ, Chin CC (2004) Prediction and identification using wavelet-based recurrent fuzzy neural networks. IEEE Trans Syst Man Cybern Part B (Cybern) 34(5):2144–2154
39. El-Nagar AM, El-Bardini M (2014) Simplified interval type-2 fuzzy logic system based on new type-reduction. J Intell Fuzzy Syst 27(4):1999–2010
40. El-Nagar AM (2016) Embedded intelligent adaptive PI controller for an electromechanical system. ISA Trans 64:314–327
41. Deng Z, Choi KS, Cao L, Wang S (2014) T2FELA: type-2 fuzzy extreme learning algorithm for fast training of interval type-2 TSK fuzzy logic system”. IEEE Trans Neural Netw Learn Syst 25(4):664–676
42. Zhang Z (2018) Trapezoidal interval type-2 fuzzy aggregation operators and their application to multiple attribute group decision making. Neural Comput Appl 29(4):1039–1054
43. Lin CM, La VH, Le TL (2018) DC–DC converters design using a type-2 wavelet fuzzy cerebellar model articulation controller. Neural Comput Appl. https://doi.org/10.1007/s00521-018-3755-z
44. El-Bardini M, El-Nagar AM (2014) Interval type-2 fuzzy PID controller: analytical structures and stability analysis. Arab J Sci Eng 39(10):7443–7458
45. El-Bardini M, El-Nagar AM (2014) Interval type-2 fuzzy PID controller for uncertain nonlinear inverted pendulum system. ISA Trans 53(3):732–743
46. Juang CF, Tsao YW (2008) A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning. IEEE Trans Fuzzy Syst 16(6):1411–1424
47. Lin YY, Chang JY, Lin CT (2014) A TSK-type-based self-evolving compensatory interval type-2 fuzzy neural network (TSCIT2FNN) and its applications. IEEE Trans Ind Electron 61(1):447–459
48. El-Nagar AM (2018) Nonlinear dynamic systems identification using recurrent interval type-2 TSK fuzzy neural network—A novel structure. ISA Trans 72:205–217
49. Lin YY, Liao SH, Chang JY, Lin CT (2014) Simplified interval type-2 fuzzy neural networks. IEEE Trans Neural Netw Learn Syst 25(5):959–969
50. Wiering M, Van Otterlo M (2012) Reinforcement learning. Adapt Learn Optim 12:3
51. Juang CF, Lin CT (1998) An online self-constructing neural fuzzy inference network and its applications. IEEE Trans Fuzzy Syst 6(1):12–32
52. Kermani BG, Schiffman SS, Nagle HT (2005) Performance of the Levenberg–Marquardt neural network training method in electronic nose applications. Sensors Actuators B Chem 110(1):13–22
53. Fu X, Li S, Fairbank M, Wunsch DC, Alonso E (2015) Training recurrent neural networks with the Levenberg–Marquardt algorithm for optimal control of a grid-connected converter. IEEE Trans Neural Netw Learn Syst 26(9):1900–1912
54. Liu H (2010) On the Levenberg–Marquardt training method for feed-forward neural networks. In: IEEE international conference on natural computation (ICNC), vol 1. pp 456–460
55. Astrom KJ, Wittenmark B (2013) Adaptive control. Courier Corporation
56. Zhang X, Zhang H, Sun Q, Luo Y (2012) Adaptive dynamic programming-based optimal control of unknown nonaffine nonlinear discrete-time systems with proof of convergence. Neurocomputing 91:48–55
57. Xu D, Jiang B, Shi P (2014) Adaptive observer based data-driven control for nonlinear discrete-time processes. IEEE Trans Autom Sci Eng 11(4):1037–1045
58. Eskinat E, Johnson SH, Luyben WL (1991) Use of Hammerstein models in identification of nonlinear systems. AIChE J 37(2):255–268
59. Berger MA, da Fonseca Neto JV (2013) Neurodynamic programming approach for the PID controller adaptation. IFAC Proc 46(11):534–539