Online Model-Free Reinforcement Learning for Output Feedback Tracking Control of a Class of Discrete-Time Systems With Input Saturation

AHMAD JOBAN AL-MAHASNEH, (Member, IEEE), SREENATHA G. ANAVATTI, (Member, IEEE), AND MATTHEW A. GARRATT, (Senior Member, IEEE)

1Mechatronics Engineering Department, Philadelphia University, Amman 19392, Jordan
2School of Engineering and Information Technology, University of New South Wales Canberra, Canberra, ACT 2612, Australia

Corresponding author: Ahmad Jobran Al-Mahasneh (almahasneh87@gmail.com)

This work was supported by the University of New South Wales Canberra, Australia.

ABSTRACT In this paper, a new model-free Model-Actor (MA) reinforcement learning controller is developed for output feedback control of a class of discrete-time systems with input saturation constraints. The proposed controller is composed of two neural networks, namely a model-network and an actor network. The model-network is utilized to predict the output of the plant when a certain control action is applied to it. The actor network is utilized to estimate the optimal control action that is required to drive the output to the desired trajectory. The main advantages of the proposed controller over the previously proposed controllers are its ability to control systems in the absence of explicit knowledge of these systems’ dynamics and its ability to start learning from scratch without any offline training. Also, it can explicitly handle the control constraints in the controller design. Comparison results with a previously published reinforcement learning output feedback controller and other controllers confirm the superiority of the proposed controller.

INDEX TERMS Reinforcement learning, adaptive control, nonlinear control, optimal control.

I. INTRODUCTION
Reinforcement Learning (RL) is a natural way of decision-making based on the interactions between an agent and its environment where the agent learns from these interactions by means of rewards and punishments [1]. RL have been used in optimal control tasks as it is based on the Bellman’s optimality equation [2]. RL can be classified into two main categories: model-based such as Adaptive Dynamic Programming (ADP) and model-free RL such as Q-learning. The first one requires a mathematical model of the system to design the control policy and second one can be used to control a system without the condition of presence of mathematical model of the system. The lack of the mathematical models is prominent in real-world systems such as industrial systems and even if they exist, they would be too complex to be used for the design of the controllers [3].

Model-free RL methods have been widely utilized for feedback control, mainly because they can be utilized as Data-Driven Controllers (DDC) for dynamic systems without explicitly relying on the mathematical models of these systems [4], [5]. Model-free control is a DDC where the model of the system’s dynamics is not required to design the controllers rather the input/output data of the system are utilized to design the controller [6]. Thus, the dynamics of the system might be unknown [6]. Hence, this type of control is practical for real-world systems [7]. Also, it accommodates for a wide range of dynamic systems that might not be controlled with model-based controllers due to the fact that the accurate models for these systems might not be available, these models’ complexity is high, or due to the necessity for huge efforts to build such models [6].

The typical RL-based algorithms such as Q-learning, contain actions/states mapping tables that are updated online to improve the abilities of the agents to make decisions. These tables can be large when the actions/states combinations are
large. Thus, Neural Networks (NNs) are utilized to approximate these mapping tables due to their universal approximation abilities, i.e., they can approximate any nonlinear continuous function under certain conditions [8], [9]. Deep NNs have also been recently utilized in deep RL [10], [11]. In addition to their approximation abilities, NNs have parallel structures permitting better deployment in practical applications [12]. NNs have plentiful applications in control systems, particularly in the estimation of the states/outputs and the design of the controllers for dynamic systems. To utilize NNs in control systems they should be trained using a particular algorithm that optimizes their parameters. As dynamic systems operate online in a sequential manner, the NNs should have the ability to learn online from the incoming data stream. Thus, typically the parameters of the NNs are adapted online to minimize a predetermined objective function. Gradient-Descent (GD) [13], Recursive Least Squares (RLS) [14] and Extended Kalman Filter (EKF) [15] are among the most prominent training approaches of NNs. NNs have been utilized in controlling different types of systems such as robotic systems [16], [17], industrial processes [18], autonomous systems [19], [20], automotive systems [21], and power systems [22]. Fuzzy Logic Systems (FLSs) can also be used in estimation-based control [23]. In fact, Takagi-Sugeno (TSK) fuzzy is equivalent to a NN. However, NNs can accomplish faster and more accurate responses than FLSs [24].

Also, Deterministic Policy Gradient (DPG)/Temporal Difference (TD) and DP have been used to control an autonomous underwater vehicle that has been employed for plume tracing [25]. One could notice that most of the formerly suggested ADP/RL-based controllers have been utilized in the regulation of the states, i.e., driving the system’s states to constant values such as zero [26], [27], [28]. It should be noted that the tracking control problem is a more challenging task than the regulation problem as the desired trajectory might be varying with time. Also, most of the published work in ADP/RL control utilize offline trained networks [29], [30], [31] which makes their approaches unsuitable for problems that require learning from scratch. In addition, most of the previously developed ADP/RL controllers require explicit knowledge of the dynamics of the system to be controlled [29], [32], [33]. This negates the RL concept as a model-free learning approach where the agents acquire knowledge from the interactions with their environments.

**A. RELATED WORKS AND MOTIVATION**

The RL/ADP based controllers have been previously utilized in output feedback control of nonlinear systems in a few papers. In [34], an RL-based controller has been utilized to control a class of MIMO Discrete-Time (DT) systems where three networks are required: observer, actor and critic networks. The main issues with such a controller are its dependence on the system’s dynamics and the number of the required networks is more than two, as typically required in RL-based control. In [35], a critic-actor structure has been utilized to control a class of pure-feedback DT systems. The major shortcomings of this controller include, it requires a transformation of the system into a specific form and it requires the system’s dynamics to be known to design the controller. In [36] and [37], RL-based controllers have been developed. The main issues with these approaches include, they have been developed for linear systems. Also, they are dependent on the knowledge of the system’s dynamics. In addition, some works have been developed to handle the input constraints using the RL-based control. For instance, in [38] and [39], policy iteration RL controller has been developed to handle the input constraints problem for different classes of continuous-time systems. However, these approaches are mathematically complex and they are model-based controllers that are not suitable for model-free control tasks. A similar approach has been reported in [40] but the dynamics of the system have been approximated using NNs to avoid the requirement of the system’s dynamics knowledge. The proposed approach utilizes three NNs, namely, identifier, action and critic networks, which makes it computationally expensive. Also, it has been adopted for the problem of state regulation, i.e., drive the system’s states to zero. In comparison with the most recent works in DT control such as in [41], [42], and [43], the proposed controller is online and no offline training of the NNs is required and thus, it starts learning from scratch. Also, the controller design does not require knowledge of the system dynamics. Furthermore, the model-actor design is unique and provides an alternative path for RL controllers design rather than the critic-actor methods. These features together, render the proposed controller suitable for a wide range of dynamic systems.

**B. PAPER CONTRIBUTIONS**

Motivated by the research gaps in the previous research works, the paper has the following main contributions:

- Distinct from most of the previous literature in model-free RL which utilize critic-actor RL structure, a new online Model-Actor (MA) structure RL-based NNs controller is developed. The method is applied for output feedback control of a class of DT systems with input saturation constraints.
- The proposed controller is capable of learning on the fly and achieve an acceptable solution of the control problem.
- The proposed method can explicitly handle the system’s input saturation and adapt to external disturbances and parametric uncertainties.
- In comparison with the existing results, the proposed method does not incorporate complex computations and can be utilized to control a wide range of dynamics without explicit knowledge of the model dynamics.

The rest of the paper is structured as follows: In Section II, the controller design including the update laws of the weights of the model and the actor networks are developed. In Section III, the stability of the closed-loop system is discussed using Lyapunov’s approach. In Section IV, numerical
results including the tracking performance, comparison analysis and sensitivity analysis are presented. Finally, Section V concludes the paper.

II. PRELIMINARIES AND CONTROLLER DESIGN

Consider a class of DT dynamical systems described as:

\[
x_2(k+1) = f_1(x(k)) \\
\vdots \\
x_n(k+1) = f_n(x(k)) + g(x(k))u(k) + d(k) \\
y(k) = h(x(k))
\]

(1)

where \( x(k+1) \) is the system state at \( k+1 \), \( f_1(\bullet), f_n(\bullet) \) and \( g(\bullet) \) are unknown continuous functions, \( u(k) \) is the control input, \( d(k) \) is the external disturbance and \( h(x(k)) \) is a known output function.

The control input \( u(k) \) is constrained as follows:

\[
u_{\text{low}} \leq u(k) \leq u_{\text{high}}
\]

(2)

where \( u_{\text{low}} \) and \( u_{\text{high}} \) are the lower and higher saturation limits respectively as demonstrated in FIGURE 1.

Remark 1: In most of the real-world applications, these saturation limits exist as the actuators usually operate in a specific range. These are hard input constraints and thus, the control signal is not allowed to violate these constraints at any time.

Assumption 1: The system in (1) is controllable.

\[\text{FIGURE 1. The saturated control input.}\]

The aim of the controller is to force the output of the plant \( y(k) \) to follow the desired trajectory \( y_d(k) \) in the presence of input saturations.

Remark 2: There are a few optimal control methods that can handle the input constraints such as Model Predictive Control (MPC). However, MPC is computationally expensive and hence, it can be utilized for only slowly varying systems such as chemical processes [44]. In our proposed controller, we utilize these constraints for our benefit to decide ranges of the control actions.

Assumption 2: The reference trajectory for one-step ahead \( y_d(k+1) \) is always known at any step \( k \).

Remark 3: Since the reference trajectory \( y_d \) is known before starting the control process, the reference trajectory in the future \( y_d(k+n) \), \( n \geq 1 \) is also known. Alternatively, one could think of the step \( k \) as the past step and the step \( k+1 \) as the current step.

A. THE CONTROLLER DESIGN

The RL-based controller has two major stages: exploration and exploitation. Both of the exploration and exploitation co-exist in the proposed method. However, they are distinct from the typical exploration and exploitation in the RL method. The exploration is conducted by uniformly dividing the feasible control range into points. Each of these points is an exploration point. This will permit the algorithm to explore a wide range of actions. However, these actions are not tested on the system itself rather their outcomes are estimated using the model-network which represents an emulation of the system's dynamics. After the outcomes of all the actions are estimated using the model-network, their costs are calculated using (14). In the exploitation stage, the action that results in the lowest cost for the next state will be selected. This will guarantee that the actor network will always be trained with optimal control actions. Randomness is not introduced anywhere in the algorithm. All the steps including the exploration and the initialization are deterministic to avoid any reproducibility issues and to eliminate any non-functional scenarios. The exploration is crucial for the controller to learn the effects of different control actions. The exploitation is the most significant stage as it determines the convergence speed of the controller to the optimal control actions.

The controller consists of the model and the actor networks. The control process has the following parts:

- The model-network is utilized to predict the next output of the plant using the plant’s output and the control input \( u \).
- The cost function is used to determine the optimal control action \( u \) at every step.
- The actor network is employed to predict the optimal control signal.

A block-diagram of the proposed controller is depicted in FIGURE 2.

The output tracking error \( e(k) \) is defined as:

\[
e(k) = y(k) - y_d(k)
\]

(3)

The augmented one step ahead output tracking error \( \hat{e}(k+1) \) is the difference between the output estimated by the model-network \( \hat{y}(k+1) \) and the desired output at the next step \( y_d(k+1) \) and it can be mathematically expressed as:

\[
\hat{e}(k+1) = \hat{y}(k+1) - y_d(k+1)
\]

(4)

The utility function \( p(k) \) is defined as:

\[
p(k) = e^2(k)
\]

(5)

the utility function \( p(k) \) is a measure of the current performance of the system, where small values of \( p(k) \) stand for high performance and vice versa.

The one step ahead utility function can be expressed as:

\[
p(k+1) = e^2(k+1)
\]

(6)
Since the exact output tracking error of every control action at step \( k + 1 \) is not exactly known, the difference between the model network output \( \hat{y}(k+1) \) and the desired one-step ahead output \( y_d(k+1) \) and will be used to approximate it according to (4). Thus, (6) can be rewritten as:
\[
\hat{p}(k + 1) = \hat{e}^2(k + 1)
\] (7)

The strategic utility function is defined as [19]:
\[
Q(k) = \sum_{i=0}^{N-k} \alpha^{N+1-i} \hat{p}(k+i)
\]
\[
Q(k) = \alpha^N \hat{p}(k+1) + \alpha^{N-1} \hat{p}(k+2) + \cdots + \alpha^{N+1-N+k} \hat{p}(K + N - K)
\]
\[
Q(k) = \alpha^N \hat{p}(k+1) + \alpha^{N-1} \hat{p}(k+2) + \cdots + \alpha^{k+1} \hat{p}(N)
\]
\[
Q(k) = \alpha^N \hat{p}(k+1) + \alpha^{N-1} \hat{p}(k+2) + \cdots + \alpha^{k+1} \hat{p}(N)
\]

where \( \alpha \in (0, 1) \) is a positive design parameter and \( N \) is the horizon index and it is an indication of how far in the future the algorithm will predict the utility function \( Q(k) \). \( Q(k) \) is a measure of the long term performance of the system.

**Remark 4:** One step ahead policy learning is a standard approach in RL called Temporal Difference (0) (TD(0)) [45]. However, the algorithm can be used to predict as many steps \( N \) as the user wishes (TD(n)) but the predictor will be more vulnerable to errors and biases as this is a dynamic environment and it is likely for the data distribution to change with time. In addition, longer predictions will be more computationally expensive.

\( Q(k) \) can also be expressed in a way similar to the standard Bellman equation [34] as:
\[
Q(k) = \min_{u(k)} \{ \alpha Q(k - 1) - \alpha^{N+1} \hat{p}(k) \}
\]
subject to \( u_{low} \leq u(k) \leq u_{high} \)

(9)

The solution of the above equation is the local optimal control signal in the saturation range. The solution is denoted as \( u^*(k) \).

**Proof:** This is to prove that the the standard Bellman equation in (9) equals to the strategic utility function in (8).

Firstly, modify (8) one step back from \( k \) to \( k - 1 \) will lead to:
\[
Q(k - 1) = \alpha^N \hat{p}(k) + \alpha^{N-1} \hat{p}(k+1) + \cdots + \alpha^{k} \hat{p}(N)
\] (10)

Secondly, multiply (10) by \( \alpha \) leads to:
\[
\alpha Q(k - 1) = \alpha^{N+1} \hat{p}(k) + \alpha^N \hat{p}(k+1) + \cdots + \alpha^{k+1} \hat{p}(N)
\] (11)

Thirdly, subtract the term \( \alpha^{N+1} \hat{p}(k) \) from both sides of (11) yields:
\[
\alpha Q(k - 1) - \alpha^{N+1} \hat{p}(k) = \alpha^{N+1} \hat{p}(k) + \alpha^N \hat{p}(k+1) + \cdots + \alpha^{k+1} \hat{p}(N) - \alpha^{N+1} \hat{p}(k)
\]

Finally, simplify (12) leads to:
\[
\alpha Q(k - 1) - \alpha^{N+1} \hat{p}(k) = \alpha^N \hat{p}(k+1) + \cdots + \alpha^{k+1} \hat{p}(N)
\]

(13)

The right hand side of (13) equals \( Q(k) \) and the min operator is used to seek the control action that minimizes the strategic utility function.

In our paper, we are interested in one step look ahead policy, which means rather than waiting for the last time step to calculate the utility function, the prediction at step \( k + 1 \) can be utilized to find the best control action to apply to the next state of the system. This is a standard approach in RL named Temporal Difference one-step predictor (TD(0)) learning and it is considered an efficient way of learning value functions online and incrementally [46]. Hence, (8) can be simplified to consider one step in the future as follows:
\[
Q(k) = \alpha^N \hat{p}(k+1) + \alpha^{k+1} \hat{p}(N)
\]

(14)

By solving (9), the control action that will minimize the utility one step ahead in the future will be found.

**B. CONVERGENCE ANALYSIS**

The one step ahead policy is considered in this paper as it will ensure the cost will approach zero when an optimal control action is selected. Firstly, the strategic utility function at step \( k - 1 \) can be written as:
\[
Q(k - 1) = \alpha p(k) + \alpha^k p(k)
\]

(15)

Substituting (15) into (9) leads to:
\[
Q(k) = \min_{u(k)} \{ \alpha(\alpha p(k) + \alpha^k p(k)) - \alpha^{N+1} \hat{p}(k) \}
\]
subject to \( u_{low} \leq u(k) \leq u_{high} \)

(16)

Considering \( N = 1 \) then (16) becomes:
\[
Q(k) = \min_{u(k)} \{ \alpha^2 p(k) + \alpha^{k+1} \hat{p}(k) - \alpha^{1+1} \hat{p}(k) \}
\]
subject to \( u_{low} \leq u(k) \leq u_{high} \)

(17)

Simplifying (17) leads to:
\[
Q(k) = \min_{u(k)} \{ \alpha^2 p(k) - \alpha^2 \hat{p}(k) + \alpha^{k+1} \hat{p}(k) \}
\]
\[
= \min_{u(k)} \{ \alpha^{k+1} \hat{p}(k) \}
\]
subject to \( u_{low} \leq u(k) \leq u_{high} \)

(18)

According to (18), the solution \( u(k) \) will be the control action that minimizes the output tracking error which is the aim of control. Also, the solution guarantees that the control actions can be selected only within the saturation limits. The solution of (18) is the optimal control action \( u^*(k) \) that minimizes the cost function \( Q(k) \), this solution will be used to update the actor network weights \( W_a \) and the actor network will then predict the next optimal control action \( u^*(k+1) \). Modifying (18) one step back at \( k - 1 \) and comparing it with (15), the following can be obtained:
\[
\alpha^k \hat{p}(k-1) = \alpha \hat{p}(k) + \alpha^k \hat{p}(k)
\]

(19)
From the above we have two main cases, for the initial policy when \( k = 1 \) we have:

\[
\alpha T p(0) = \alpha p(1) + \alpha T p(1)
\]  
(20)

Since \( k \geq 1 \), then the above equation can be simplified as:

\[
0 = 2\alpha p(1)
\]  
(21)

From the definition of \( p(k) \) in (5), the following can be written:

\[
2 \epsilon^2(1) = 0
\]  
(22)

Which can imply that the initial control policy will lead the tracking error to approach zero.

For large values of \( k \) the following can be written:

\[
\alpha p(k) = \alpha k((p(k) - 1) - \alpha k p(k))
\]  
(23)

Since \( \alpha \in (0, 1) \), the term \( \alpha k \) will approach a neighborhood around the zero for large enough \( k \) values and (23) can be rewritten as:

\[
\alpha p(k) \approx 0
\]  
(24)

From the definition of \( p(k) \), one can write the following:

\[
\alpha e^2(k) \approx 0
\]  
(25)

From the above equation one can know, the tracking error will asymptotically approach a neighborhood around the zero when the values of \( k \) increase.

\section*{C. MODEL-NETWORK}

The model-network is trained online to predict the output of the plant one step ahead in the future, based on the control input \( u(k) \) and the current output of the plant \( y(k) \).

Remark 5: The model-network acts as a predictor of the system’s output when a certain control action is applied to it. Also, the model-network is crucial in real-world systems to avoid driving the agents into undesirable states that could cause their malfunction or damage. In the typical critic-actor framework, the critic network is utilized to estimate the value function or Q function in the Q-learning framework.

According to the universal approximation property of NNs [8], [9], there is an optimal weights vector \( W_m^\ast \) that makes the model-network approximates the future output with an arbitrary error \( \epsilon_m \), then \( y(k + 1) \) can be expressed as follows:

\[
y(k + 1) = W_m^T \sigma_m(u(k), y(k)) + \epsilon_m
\]  
(26)

The actual output estimated by the model-network is expressed as:

\[
\hat{y}(k + 1) = W_m^T \sigma_m(u(k), y(k))
\]  
(27)

where \( W_m \) is the current vector of the hidden-output weights of the model-network.

The weighting error of the model-network is defined as:

\[
\tilde{W}_m(k) = W_m^\ast - W_m(k)
\]  
(28)

The estimation error of the model-network \( e_m(k) \) can be expressed as:

\[
e_m(k) = \hat{y}(k) - y(k)
\]  
(29)

The model-network is trained online to minimize the following objective function:

\[
E_m(k) = \frac{1}{2} \epsilon_m^2(k)
\]  
(30)

The hidden-output weights of the model-network are updated according to the gradient descent method as follows:

\[
W_m(k + 1) = W_m(k) - \gamma_m \frac{\partial E_m(k)}{\partial W_m(k)}
\]

\[
= W_m(k) - \gamma_m \frac{\partial E_m(k)}{\partial e_m(k)} \frac{\partial e_m(k)}{\partial W_m(k)}
\]

\[
= W_m(k) - \gamma_m e_m(k) \sigma_m(u(k), y(k))
\]

\[
= W_m(k) - \gamma_m (\hat{y}(k) - y(k)) \sigma_m(u(k), y(k))
\]  
(31)

\section*{D. ACTOR NETWORK}

The actor network is trained online to predict the optimal control signal \( u(k + 1) \) using the predicted control signal from the solution of (9) and the current plant output \( y(k) \).

According to the universal approximation property of NNs [8], [9], there is an optimal weights vector \( W_a^\ast \) that makes the actor network predict the optimal control \( u^\ast(k + 1) \) with an arbitrary error \( \epsilon_a \), then \( u^\ast(k + 1) \) can be written as follows:

\[
u^\ast(k + 1) = W_a^T \sigma_a(u(k), y(k)) + \epsilon_a
\]  
(32)

The actual output which is predicted by the actor network is expressed as:

\[
u(k + 1) = W_a^T \sigma_a(u(k), y(k))
\]  
(33)

where \( W_a \) is the current actor network hidden-output weights vector.

The weighting error of the actor-network is defined as:

\[
\tilde{W}_a(k) = W_a^\ast - W_a(k)
\]  
(34)

The estimation error of the actor-network \( e_a(k) \) is expressed as:

\[
e_a(k) = u^\ast(k) - u(k)
\]  
(35)

The actor network is trained online to minimize the following objective function:

\[
E_a(k) = \frac{1}{2} \epsilon_a^2(k)
\]  
(36)
The hidden-output weights of the actor network are adapted according to the gradient-descent method as follows:

$$W_a(k+1) = W_a(k) + \gamma_a \frac{\partial E_a(k)}{\partial W_a(k)}$$

$$= W_a(k) + \gamma_a e_a(k) \frac{\partial e_a(k)}{\partial W_a(k)}$$

$$= W_a(k) + \gamma_a e_a(k) \sigma_a(u(k), y(k))$$

$$= W_a(k) + \gamma_a (\alpha^2 - u(k)) \sigma_a(u(k), y(k))$$

(37)

**Remark 6**: There are some rules to follow for the selection of the design parameters. The learning rates $\gamma_a$ and $\gamma_m$ are selected as follows: start with a very small values such as 0.01 and then increase it in a fixed step such as 0.01 until the desired convergence behavior is observed. For the error threshold, $e_{th}$, one can select it as $1e^{-6}$ or less. If the application does not require high accuracy, one can select it around $1e^{-3}$.

### III. STABILITY ANALYSIS

To study the stability of the proposed controller, Lyapunov second method is utilized. The stability of the system is independent of the mathematical model of the system.

**Assumption 3**: The norms of the actor and model networks weights are bounded as follows: $||W_a(k)|| \leq \alpha$ where $\alpha \in (0, \infty)$ and $||\hat{W}_m(k)|| \leq \beta$ where $\beta \in (0, \infty)$.

If the learning rates $\gamma_a$ and $\gamma_m$ are selected as:

$$\gamma_a < -\frac{||\hat{W}_m(k)||^2}{e^2(k)}$$

(38)

$$\gamma_m > \sqrt{\frac{||\hat{W}_m(k)||^2}{2\sigma^2_m(k)||\sigma_m||^2}}$$

(39)

then the closed-loop system is stable.

**Proof**: The following Lyapunov functions are defined as:

$$V(k) = \sum_{i=1}^{2} V_i(k)$$

(40)

The first Lyapunov function is designed as:

$$V_1(k) = e^2(k - 1) + \frac{1}{\gamma_a} \hat{W}_a(k)^T \hat{W}_a(k)$$

(41)

The first difference of $V_1$ can be written as:

$$\Delta V_1(k) = e^2(k) - e^2(k - 1)$$

$$+ \frac{1}{\gamma_a} (\hat{W}_a(k + 1)\hat{W}_a(k + 1) - \hat{W}_a(k)^T \hat{W}_a(k))$$

(42)

Substitute (37) into (42) leads to:

$$\Delta V_1(k) = e^2(k) - e^2(k - 1)$$

$$+ \frac{1}{\gamma_a} (\hat{W}_a(k) + (\gamma_a e_a(k) \sigma_a(k))^T$$

$$\times (\hat{W}_a(k) + \gamma_a e_a(k) \sigma_a(k)) - \hat{W}_a^T(k) \hat{W}_a(k))$$

(43)

Incorporate Cauchy-Schwarz inequality [47] into (43) leads to:

$$\Delta V_1(k) \leq \frac{1}{\gamma_a} \left( 2||\hat{W}_a(k)||^2 + 2\gamma_a e^2_a(k)||\sigma_a(k)||^2$$

$$- ||\hat{W}_a(k)||^2 \right)$$

$$+ e^2(k) - e^2(k - 1)$$

(44)

Simplify (44) further leads to:

$$\Delta V_1(k) \leq \frac{||\hat{W}_a(k)||^2}{\gamma_a} + 2\gamma_a e^2_a(k)||\sigma_a(k)||^2$$

(45)
The following nonlinear nonaffine DT second-order system
the tracking error can be arbitrarily small.

\[ +e^2(k) - e^2(k - 1) \]

Substitute (38) into (45) and simplify leads to:
\[ \Delta V_1(k) \leq -2\frac{||\tilde{W}_m(k)||^2}{e^2(k)}e_\sigma(k)||\sigma_m(k)||^2 + e^2(k - 1) \]
\[ \leq 0 \]

The second Lyapunov function \( V_2(k) \) is written as:
\[ V_2(k) = \tilde{W}_m(k)^T \tilde{W}_m(k) \]

The first difference of \( V_2 \) can be written as:
\[ \Delta V_2(k) = \tilde{W}_m^T(k + 1)\tilde{W}_m(k + 1) - \tilde{W}_m^T(k)\tilde{W}_m(k) \]

Substitute (31) into (48) leads to:
\[ \Delta V_2(k) = (\tilde{W}_m(k) - \gamma_m e_m(k)\sigma_m(k))^T \]
\[ \times (\tilde{W}_m(k) - \gamma_m e_m(k)\sigma_m(k)) - \tilde{W}_m^T(k)\tilde{W}_m(k) \]

Incorporate Cauchy-Schwarz inequality into (49) leads to:
\[ \Delta V_2(k) \leq 2||\tilde{W}_m(k)||^2 - 2\gamma_m^2 e_m^2(k)||\sigma_m(k)||^2 \]
\[ -||\tilde{W}_m(k)||^2 \]

Simplify (50) further leads to:
\[ \Delta V_2(k) \leq ||\tilde{W}_m(k)||^2 - 2\gamma_m^2 e_m^2(k)||\sigma_m(k)||^2 \]

Substitute (39) into (51) and simplify further leads to:
\[ \Delta V_2(k) \leq 0 \]

As both \( V_1 \) and \( V_2 \) satisfy the Lyapunov stability conditions according to the above analysis, all the signals of the closed-loop system are semi-globally uniformly ultimately bounded (SUUB) [48]. The proof is complete. \( \square \)

Remark 7: Since the online learning of the neural networks incorporates a single iteration at every time step rather than many iterations as typically practiced in offline learning, the effect of the gradient-descent’s local minimum is not significant as the actual error values are used to update the neural networks’ weights in a single shot by means of performance feedback. Also, the limitations on the learning rates \( \gamma_a \) and \( \gamma_m \) explicitly determine the ranges of the learning rates to maintain the stability of the system.

Remark 8: The tracking error \( e(k) \) is arbitrarily small as the NNs have universal approximation capabilities by which they can approximate any nonlinear function, under some mild conditions, to an arbitrarily small error. Since the NNs are used for the model and the actor parts of the controller, the tracking error can be arbitrarily small.

IV. NUMERICAL RESULTS
A. TRACKING PERFORMANCE ANALYSIS
The following nonlinear nonaffine DT second-order system is utilized to test the proposed controller performance:
\[ x_1(k + 1) = 0.2 \frac{x_1(k)^2 x_2(k)}{1 + x_1(k)^2} + 0.5x_2(k) \]
\[ x_2(k + 1) = \frac{x_1(k)}{1 + x_1(k)^2 + x_2(k)^2} + u(k) + \sin (u(k)) + d(k) \]
\[ y(k) = x_1(k) \]
subject to \(-2 \leq u(k) \leq 2\) \[(53)\]
The external disturbance is given by:
\[ d(k) = 0.05 \cos (0.01k) \cos (x_1(k)) \]
\[(54)\]
The desired output trajectory is given by:
\[ y_d(k) = \begin{cases} \sin \left(\frac{\pi kT}{2}\right) & \text{if } k < 3000 \\ 0.5 \cos \left(\frac{2\pi kT}{5}\right) + 0.5 \sin \left(\frac{2\pi kT}{5}\right) & \text{if } k \geq 3000 \end{cases} \]

The initial conditions are \( x_1(0), x_2(0) = 0, 0.3 \). The hidden-output weights for the model and the actor networks are initialized to zeros. The learning rates are selected as \( \gamma_a = 0.3 \) and \( \gamma_m = 0.1 \). The error threshold \( e_{th} \) is \( 1 \times 10^{-5} \). The controller design consists of the model-network that is trained online to predict \( \hat{y}(k + 1) \). Also, the cost function in (14) is utilized to find the best control action that minimizes the tracking error for the next step. Finally, the actor network is trained online to predict the next control action. The NNs in this paper are General Regression Neural Networks (GRNNs).

The trajectory tracking as shown in FIGURE 3 depicts the learning ability of the proposed controller wherein the initial exploration, the output oscillates between 1 and -1 then it converges to the desired output. The control signal from the actor network is depicted in FIGURE 4. The control signal follows the shape of the reference and it is bounded between -0.8 and 0.8.

![FIGURE 3. The output tracking performance for the proposed MA controller.](image-url)

The optimal cost evolution with time as shown in FIGURE 5 demonstrates that the optimal cost increases to more than 0.01 and then it gradually reduces to almost zeros where it converges after that. This figure ensures the ability of the proposed controller to quickly reach an optimal cost and converge to it.

B. COMPARISON ANALYSIS
In this section, we compare our proposed method with the method published in [35] using the same plant and exactly the
FIGURE 4. The control signal for the proposed MA controller.

FIGURE 5. Optimal cost evolution with time.

same parameters as mentioned in their paper. The following performance indices are utilized for the comparison:

1) Root Mean Square Error (RMSE) and it can be expressed as:

\[
RMSE = \sqrt{\frac{1}{N} \sum_{k=1}^{N} (y_d(k) - y(k))^2}
\]  

(55)

where \( N \) is the total number of simulation steps.

2) Mean Absolute Error (MAE) and it can be expressed as:

\[
MAE = \frac{1}{N} \sum_{k=1}^{N} |y_d(k) - y(k)|
\]  

(56)

3) Integral Absolute Error (IAE) and it can be expressed as:

\[
IAE = \int_{t=0}^{T} |e(k)|dt
\]  

(57)

4) Integral Square Error (ISE) and it can be expressed as:

\[
ISE = \int_{t=0}^{T} e^2(k)dt
\]  

(58)

The performance comparison results are shown in TABLE 1 and in FIGURE 6. These results depict the superiority of our proposed controller over the controller in [35]. After the initial learning stage, our controller accurately tracks the desired trajectory while the controller in [35] does not accurately follow the desired trajectory, does not reach the maximum value of the desired trajectory and suffers a delay in the output tracking. The bold numbers indicate that our controller has better tracking performance than the compared controller in terms of lower RMSE, MAE, IAE and ISE error performance indices.

C. SENSITIVITY ANALYSIS

One of the main design parameters in our controller design is the number of control actions in the exploration stage. Thus, a sensitivity analysis of this parameter effect on the controller tracking performance is considered in this subsection. The control actions are uniformly selected in a range of input saturation constraints. It is evident from TABLE 2 that the number of control actions has a significant impact on the tracking accuracy. When the number of control actions is increased, the controller accuracy also increases until a certain limit where the accuracy will not significantly improve. This is sensible as in the basic idea of RL, the agent needs to learn as much control actions as possible. However, there is a particular limit where increasing the number of control actions will not significantly improve its learning ability. For instance, in our example, as depicted in TABLE 2, after 81 control actions, the improvements in the tracking accuracy are marginal. It is also clear that even when the number of control actions is small, the controller can still perform reasonably well.

D. ROBUSTNESS ANALYSIS

In this subsection, the robustness of the controller is examined by systematically increasing the disturbance magnitude by two, three, five, seven and ten folds from the original one. The results as depicted in TABLE 3 and FIGURE 7 demonstrate the ability of the controller to reject disturbances even when the disturbance magnitudes significantly increase.

| Controller | Performance index |
|------------|-------------------|
|             | RMSE   | MAE    | IAE    | ISE    |
| Model-actor controller | 0.3665 | 0.0099 | 0.2974 | 0.0400 |
| The critic-actor controller | 0.3250 | 0.2797 | 8.3918 | 3.1683 |

TABLE 1. Comparison results with the critic-actor controller [35].
TABLE 2. The effects of the number of control actions in the exploration stage.

| # of control of actions | RMSE | MAE | IAE | ISE |
|-------------------------|------|-----|-----|-----|
| 6                       | 0.1029 | 0.0823 | 2.4689 | 0.3176 |
| 9                       | 0.0892 | 0.0523 | 1.5690 | 0.1435 |
| 21                      | 0.0425 | 0.0273 | 0.6513 | 0.0541 |
| 41                      | 0.0378 | 0.0138 | 0.4155 | 0.0438 |
| 81                      | 0.0368 | 0.0109 | 0.3273 | 0.0406 |
| 201                     | 0.0365 | 0.0100 | 0.3009 | 0.0400 |
| 401                     | 0.0365 | 0.0099 | 0.2974 | 0.0400 |
| 801                     | 0.0365 | 0.0090 | 0.2964 | 0.0400 |

TABLE 3. Performance comparison for different disturbances.

| Disturbance | RMSE | MAE | IAE | ISE |
|-------------|------|-----|-----|-----|
| d           | 0.0365 | 0.0099 | 0.2974 | 0.0400 |
| 2d          | 0.0363 | 0.0099 | 0.2964 | 0.0396 |
| 3d          | 0.0364 | 0.0101 | 0.3018 | 0.0398 |
| 5d          | 0.0365 | 0.0105 | 0.3164 | 0.0400 |
| 7d          | 0.0412 | 0.0141 | 0.4231 | 0.0509 |
| 10d         | 0.0381 | 0.0142 | 0.4260 | 0.0436 |

FIGURE 7. Effects of disturbance variation on the tracking performance of the MA controller.

FIGURE 8. Parameters variations with time.

FIGURE 9. Trajectory tracking with time-varying parameters.

without substantial loss in the accuracy. In fact, for the different disturbance scenarios, after the initial learning stage, the tracking trajectories converge to almost the same final values.

E. UNCERTAINTY ANALYSIS

In this subsection, the ability of the proposed controller for handling plant time-varying parameters is tested.

If the system in (53) is rewritten with time-varying parameters as:

\[
\begin{align*}
x_1(k+1) &= a(k)x_1^2(k)x_2(k) + b(k)x_2(k) \\
x_2(k+1) &= \frac{x_1(k)}{1 + x_1^2(k) + x_2^2(k)} + c(k)u(k) + \sin(u(k)) + d(k) \\
y(k) &= x_1(k) \\
\text{subject to } & -2 \leq u(k) \leq 2
\end{align*}
\]

(59)

where \(a(k), b(k)\) and \(c(k)\) are the time-varying parameters and their nominal values are 0.2, 0.5 and 1 respectively. The aim is to vary these parameters with \(\pm 20\%\) from their nominal values as follows:

\[
\begin{align*}
a(k) &= 0.04\sin(0.01k) + 0.2 \\
b(k) &= 0.1\sin(0.01k) + 0.5 \\
c(k) &= 0.2\sin(0.01k) + 1.0
\end{align*}
\]

(60)

The parameters variation with time is depicted in FIGURE 8. Also, the trajectory tracking performance with time-varying parameters as shown in FIGURE 9 depicts that the parameter variations have a negligible effect on controller performance.

F. TRACKING WITH APERIODIC SIGNAL

In this subsection, aperiodic signal, with variable frequency and amplitude, is developed to test the MA controller as follows:

\[
y_d(k) = \begin{cases} 
\text{square}(0.005k), & \text{for } 1 \leq n \leq 2000 \\
0.8 \times \text{square}(0.004k), & \text{for } 2000 < n \leq 4000 \\
0.5 \times \text{square}(0.003k), & \text{for } 4000 < n \leq 6000 
\end{cases}
\]

1) TRACKING PERFORMANCE

The tracking results, as depicted in FIGURE 10, shows the ability of the controller to track aperiodic signal. The initial overshoots reduce when the learning progresses.

2) SENSITIVITY ANALYSIS

The parameters \(a, b\) and \(c\) are varied as in subsection C to test the ability of the controller to handle the plant’s parameter variation in the case of aperiodic signal. The results,
FIGURE 10. MA tracking for aperiodic signal.

FIGURE 11. MA tracking for aperiodic signal with varying parameters.

FIGURE 12. MA tracking for aperiodic signal with five folds disturbance.

as depicted in FIGURE 11, reveal the ability of the controller to keep the system close to the desired reference even when the parameters vary significantly.

3) ROBUSTNESS ANALYSIS

Also, to verify the ability of the controller in the case of increasing disturbances when aperiodic reference is applied, five folds of the original magnitude of the disturbance is used. The performance results, as shown in FIGURE 12, depict the ability of the controller to perform reasonably well in the case of the increased disturbances.

G. PENDULUM CONTROL

In this example, a nonlinear pendulum is utilized to test the proposed controller. The pendulum dynamic model can be represented as:

\[
\dot{x}_1(t) = x_2(t)
\]

\[
\dot{x}_2(t) = -\sin(x_1(t)) - 0.03x_2(t) + u(t) + d(t)
\]

\[
y(t) = x_2(t)
\]

subject to \(-2 \leq u(k) \leq 2\) \hspace{1cm} (61)

The initial states are \(X(0) = [\pi/2, 0]\). The DT version of the pendulum can be expressed as:

\[
x_1(k+1) = x_2(k)\Delta T + x_1(k)
\]

\[
x_2(k+1) = (-\sin(x_1(k)) - 0.03x_2(k) + u(k))\Delta T + x_2(k)
\]

\[
y(k) = x_2(k)
\]

where \(\Delta T = 0.01\) is the sampling time.

The desired output \(y_d(k)\) is defined as:

\[
y_d(t) = \sin(k\Delta T/3)
\]

(63)

The external disturbance \(d(k)\) is given by:

\[
d(k) = 0.1 \sin(k\Delta T) \cos(0.5k\Delta T)
\]

(64)

The learning rates are selected as \(\gamma_a = 0.1\) and \(\gamma_m = 2.3\). The error threshold \(e_{th}\) is \(1 \times 10^{-7}\). The controller design contains three main blocks. Firstly, the model-network is trained online to predict the pendulum output \(\hat{y}(k+1)\). Secondly, the cost function in (14) is utilized to find the best control action that minimizes the tracking error for the next step. Finally, the actor network is trained online to predict the next control action. The NNs are GRNNs.

The output trajectory tracking of the pendulum shown in FIGURE 13 reveals the ability of the proposed controller to accurately track the desired output with a very low tracking error. There are some minor oscillations occurring between 10 and 15 seconds due to the learning.

H. MA-CONTROLLER COMPARISON WITH OTHER CONTROLLERS

In this subsection, the proposed controller is compared with a proportional integral derivative controller (PID) and adaptive RBFNN controller for the pendulum example with different inputs including step, sine wave and square wave. Three performance indices are utilized in the comparison namely, RMSE, ISE and IAE. The parameters of the PID are 0.1 for \(K_P\), \(K_D\) and \(K_I\). RBFNN has 21 nodes in the hidden layer and its learning rate is 0.01. The comparison results as depicted
TABLE 5. Step response info.

| Controller | Rising Time (Sec) | Settling Time (Sec) | Overshoot % |
|------------|-------------------|---------------------|-------------|
| MA         | 0.5475            | 3.5653              | 6.1265      |
| PID        | 0.3729            | 8.4164              | 56.6314     |
| RBFNN      | 0.2712            | 8.7789              | 66.9475     |

FIGURE 14. MA controller tracking comparison with other controllers for step input in the pendulum example.

FIGURE 15. MA controller tracking comparison with other controllers for sine input in the pendulum example.

in Table 4 demonstrate the capability of the proposed controller to perform better than most of the other controllers in the tested performance indices.

To further illustrate the comparison between the proposed controller and the PID and RBFNN controllers, the figures of the output trajectory tracking are constructed for the step, sine and square inputs as demonstrated in FIGURE 14, FIGURE 15 and FIGURE 16. The figures depict the superiority of the proposed controller over the other controllers in two aspects, less overshoot and oscillations from the setpoint and better convergence behavior. For instance, for the step response in FIGURE 14 and as reported in Table. 5, PID and RBFNN reach about 56.63% and 66.95% overshoot respectively from the setpoint in the transient response in the interval after two seconds. The proposed controller, on the other hand, has 8.65% overshoot from the setpoint which leads to significantly less oscillation in the transient response than the PID and RBFNN. The proposed controller has the lowest settling time of 4.76 seconds. The RBFNN controller has the lowest rising time as it is an adaptive controller and also, as it starts from a reasonable structure. In the case of the sine wave input as depicted in FIGURE 15, most of the controllers behave reasonably well in terms of the overshoot and the convergence. The proposed controller has some oscillations at the beginning of the learning stage but they vanish after the learning converges. In the case of the square wave input as shown in FIGURE 16, PID and RBFNN have significantly higher overshoot than the proposed controller. The proposed controller has the most reasonable performance in terms of the low overshoot and reasonably short convergence time.

FIGURE 16. MA controller tracking comparison with other controllers for square input in the pendulum example.

I. CONTROL ACTIONS ANALYSIS

In this subsection, the ability of the controller to handle the saturation is discussed for the pendulum example with the different references, namely, step, sine and square inputs. The results of the evolution of the MA's control signals with time for the various references are depicted in FIGURE 17, FIGURE 18, and FIGURE 19. The results reveal that for the step reference, the control signal initially reaches the saturation limit 2 and then the controller gradually reduces the signal to around 1 for the rest of the simulation. For the sine reference, the control signal reaches the saturation limits 2 and $-2$ and then the controller reduces the control signal to between $-1$ and 1. For the square wave, the control signal reaches the saturation limits multiple times due to the nature of the square wave, however, the controller gradually reduces the control signal to 1 or $-1$. Since the controller starts learning from scratch (i.e. the NNs have zero output weights initially), it is initially looking for an acceptable solution.
among all the possible solutions, which is usually referred as the exploration stage in the RL framework. Thus, these oscillations are inevitable initially. However, as these oscillations are bounded and does not violate the control saturation constrains, they should not generate any stability issue for the system.

J. COMPUTATION ANALYSIS

In this subsection, the focus is on analyzing the average time required for each iteration of the algorithm and comparing it with the other controllers. To accomplish that, the required time for each iteration of the algorithm is recorded and then the statistical mean and the standard deviation (mean±std) of these times are calculated. The results, as depicted in TABLE 6, indicate that the MA controller takes a little bit longer time for each iteration that the other controllers as predicted because it incorporates two NNs and cost evaluation. However, in practical applications, these differences are not significant and it will not affect the applicability in real-world applications.

Remark 9: The algorithm is not mathematically complex. All the simulation results are generated from running the algorithm in real-time. However, for simple systems, this algorithm might be replaced with more efficient algorithms particularly when the hardware specifications are low. To fully answer the question, we have calculated the update rate of the average difference between the time steps 2.7465e-06 sec and the average time required to execute the controller code update 1.19158e-06 sec. Since the time required to execute the controller’s update is less than the difference between the time steps, then the controller will be working in the real-time.

Remark 10: In comparison with our controller, the controllers proposed in [49], [50], and [51] utilize a variety of neuro-fuzzy systems to implement the structure of the critic-actor RL controller to control nonlinear systems with uncertainties and disturbances.

V. CONCLUSION

A new MA RL-based controller is designed to control dynamic systems without explicitly relying on the systems’ dynamics. It can be even utilized when the dynamics of the system are unknown. The proposed method considers an approach using two networks designated the model and actor networks. This dual-network structure allows the controller to explore the whole feasible control actions range and at the same time, the action that has the best performance will always be selected. The controller performance is proven to be effective in learning the best control actions based on the RL-concept and the optimal control method. The comparison results with a critic-actor controller reveal that the performance of our controller is far better. In addition, the robustness and uncertainty analysis of the proposed controller ensures its ability to reject disturbances and handle uncertainty. Also, the performance of the proposed controller in comparison with the existing controllers such as PID and adaptive RBFNN ensures its effectiveness to handle the tracking control problem. Although the algorithm contains two neural networks, it is not computationally expensive. Also, it can be optimized by reducing the number of control actions in the exploration stage and the size of the hidden-layers of the NNs to suite a variety of hardware with different processing abilities.

TABLE 6. Controller’s computational times comparison.

| Controller | mean±std (seconds) |
|------------|-------------------|
| MA         | 7.6000e-03 ± 0.0013 |
| PID        | 6.3684e-04 ± 0.0012 |
| RBFNN      | 6.6459e-04 ± 9.5591e-04 |

REFERENCES

[1] R. Kamalapurkar, P. Walters, J. Rosenfeld, and W. Dixon, Optimal Control. Cham, Switzerland: Springer, 2018, pp. 1–16.
[2] Y. Guan, S. E. Li, J. Duan, J. Li, Y. Ren, Q. Sun, and B. Cheng, “Direct and indirect reinforcement learning,” Int. J. Intell. Syst., vol. 36, no. 8, pp. 4439–4467, Aug. 2021, doi: 10.1002/int.22466.

[3] Z. Hou, H. Gao, and F. Lewis, “Data-driven control and learning systems,” IEEE Trans. Ind. Electron., vol. 64, no. 5, pp. 4070–4075, May 2017.

[4] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, “Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers,” IEEE CONTROL Syst., vol. 32, no. 6, pp. 76–105, Dec. 2012.

[5] F. Lewis and D. Liu, Reinforcement Learning and Approximate Dynamic Programming for Feedback Control (IEEE Press Series on Computational Intelligence). Hoboken, NJ, USA: Wiley, 2013.

[6] Z. Hou and S. Jin, Model Free Adaptive Control: Theory and Applications. Boca Raton, FL: CRC Press, 2013.

[7] Z.-S. Hou and Z. Wang, “From model-based control to data-driven control: Survey, classification and perspective,” Inf. Sci., vol. 235, pp. 3–35, Jun. 2013.

[8] J. Park and I. W. Sandberg, “Universal approximation using radial-basis-function networks,” Neural Comput., vol. 3, no. 2, pp. 246–257, Mar. 1991.

[9] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” Neural Netw., vol. 2, no. 5, pp. 359–366, Dec. 1989.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and S. Petersen, “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, p. 529, 2015.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” 2015, arXiv:1509.02971.

[12] S. Haykin, Neural Networks and Learning Machines, 3/E. London, U.K.: Pearson, 2010.

[13] M. T. Hagan and M. B. Menhaj, “Training feedforward networks with the Marquardt algorithm,” IEEE Trans. Neural Netw., vol. 5, no. 6, pp. 989–993, Nov. 1994.

[14] J. Biski and L. Rutkowski, “A fast training algorithm for neural networks,” IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 45, no. 6, pp. 749–753, Jun. 1998.

[15] G. V. Puskorius and L. A. Feldkamp, Parameter-Based Kalman Filter Training: Theory and Implementation. Hoboken, NJ, USA: Wiley, 2002, ch. 2, pp. 23–67.

[16] W. He, Y. Chen, and Z. Yin, “Adaptive neural network control of an uncertain robot with full-state constraints,” IEEE Trans. Cybern., vol. 46, no. 3, pp. 620–629, Apr. 2016.

[17] W. He, Y. Dong, and C. Sun, “Adaptive neural impedance control of a robotic manipulator with input saturation,” IEEE Trans. Syst., Man, Cybern., Syst., vol. 46, no. 3, pp. 334–344, Mar. 2016.

[18] W. Wang, H. Gao, and J. Qiu, “A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control,” IEEE Trans. Neural Netw. Learn. Syst., vol. 27, no. 2, pp. 425–428, Feb. 2016.

[19] R. Cui, C. Yang, Y. Li, and S. Sharma, “Adaptive neural network control of aues with control input nonlinearities using reinforcement learning,” IEEE Trans. Man. Cybern., Syst., vol. 47, no. 6, pp. 1019–1029, Jan. 2017.

[20] W. He, Z. Yan, C. Sun, and Y. Chen, “Adaptive neural network control of a flapping wing micro aerial vehicle with disturbance observer,” IEEE Trans. Cybern., vol. 47, no. 10, pp. 3452–3465, Oct. 2017.

[21] Y.-J. Liu, Q. Zeng, S. Tong, C. L. P. Chen, and L. Liu, “Adaptive neural network control for a class of nonstrict-feedback discrete-time systems with time-varying vertical displacement and speed constraints,” IEEE Trans. Ind. Electron., vol. 66, no. 12, pp. 9458–9466, Dec. 2019.

[22] R. J. Wai, M. W. Chen, and Y. K. Liu, “Design of adaptive control and fuzzy neural network control for single-stage boost inverter,” IEEE Trans. Ind. Electron., vol. 62, no. 9, pp. 5434–5445, Sep. 2015.

[23] X. Zhang, Y. Wang, C. Wang, C.-Y. Su, Z. Li, and X. Chen, “Adaptive estimated inverse output-feedback quantized control for piezoelectric positioning stage,” IEEE Trans. Ind. Electron., vol. 49, no. 6, pp. 2106–2118, Jun. 2019.

[24] H. Yu, T. Xie, S. Pasczynski, and B. M. Wilamowski, “Advantages of radial basis function networks for dynamic system design,” IEEE Trans. Ind. Electron., vol. 58, no. 12, pp. 4536–4540, Dec. 2011.

[25] H. Hu, S. Song, and C. L. P. Chen, “Plume tracing via model-free reinforcement learning method,” IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 8, pp. 2515–2527, Aug. 2019.
[48] S. S. Ge, J. Zhang, and T. H. Lee, “Adaptive neural network control for a class of MIMO nonlinear systems with disturbances in discrete-time,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 4, pp. 1630–1645, Aug. 2004.

[49] A. A. Khater, A. M. El-Nagar, M. El-Bardini, and N. El-Rabaie, “A novel structure of actor-critic learning based on an interval type-2 TSK fuzzy neural network,” *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 11, pp. 3047–3061, Nov. 2020.

[50] A. A. Khater, A. M. El-Nagar, M. El-Bardini, and N. M. El-Rabaie, “Improving the performance of a class of adaptive fuzzy controller based on stable and fast on-line learning algorithm,” *Eur. J. Control*, vol. 51, pp. 39–52, Jan. 2020.

[51] A. A. Khater, A. M. El-Nagar, M. El-Bardini, and N. M. El-Rabaie, “Online learning of an interval type-2 TSK fuzzy logic controller for nonlinear systems,” *J. Franklin Inst.*, vol. 356, no. 16, pp. 9254–9285, Nov. 2019.

AHMAD JOBRAH AL-MAHASNEH (Member, IEEE) received the B.Sc. and M.Sc. degree in mechatronics/mechanical engineering from the Jordan University of Science and Technology, in 2010 and 2014, respectively, and the Ph.D. degree in mechanical engineering from the University of New South Wales Canberra, in 2020. He is an Assistant Professor with the Department of Mechatronics Engineering, Philadelphia University, Jordan. His research interests include data-driven modeling and control, adaptive and intelligent control, evolving systems and neural networks, nonlinear control, and unmanned aerial vehicles control. He is a member of IEEE Control Systems Society (CSS) and IEEE Systems, Man, and Cybernetics Society (SMC).

SREENATHA G. ANAVATTI (Member, IEEE) received the Ph.D. degree from the Indian Institute of Science, Bengaluru, India. He was an Associate Professor at the IIT Bombay, Mumbai, India. He moved to Australia in 1998. He is a Senior Lecturer with the School of Engineering and Information Technology, University of New South Wales Canberra, Canberra, ACT, Australia. He has supervised a number of Ph.D. scholars over these years. His current research interests include application of artificial intelligence techniques like fuzzy and neural systems for UAVs, underwater vehicles, ground vehicles, identification and control of dynamic systems, navigation and path planning for autonomous vehicles, evolutionary fuzzy systems, and their applications to dynamic systems.

MATTHEW A. GARRATT (Senior Member, IEEE) received the B.E. degree in aeronautical engineering from Sydney University, Sydney, NSW, Australia, the Graduate Diploma degree in applied computer science from Central Queensland University, Rockhampton, QLD, Australia, and the Ph.D. degree in biologically inspired robotics from Australian National University, Canberra, ACT, Australia, in 2008. He is a Full Professor with the School of Engineering and Information Technology (SEIT), University of New South Wales Canberra. He is a member of the CIS task force on the Ethics and Social Implications of CI. His current research interests include sensing, guidance, and control for autonomous systems with particular emphasis on biologically inspired and computational intelligence approaches.