Optimal and Autonomous Control Using Reinforcement Learning: A Survey

Bahare Kiumarsi, *Member, IEEE*, Kyriakos G. Vamvoudakis, *Senior Member, IEEE*, Hamidreza Modares, *Member, IEEE*, and Frank L. Lewis, *Fellow, IEEE*

***Abstract*— This paper reviews the current state of the art on reinforcement learning (RL)-based feedback control solutions to optimal regulation and tracking of single and multiagent systems. Existing RL solutions to both optimal $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control problems, as well as graphical games, will be reviewed. RL methods learn the solution to optimal control and game problems online and using measured data along the system trajectories. We discuss Q-learning and the integral RL algorithm as core algorithms for discrete-time (DT) and continuous-time (CT) systems, respectively. Moreover, we discuss a new direction of off-policy RL for both CT and DT systems. Finally, we review several applications.**

***Index Terms*— Autonomy, data-based optimization, reinforcement learning (RL).**

I. Introduction

OPTIMAL control theory [1]–[6] is a mature mathematical discipline that finds optimal control policies for dynamical systems by optimizing user-defined cost functionals that capture desired design objectives. The two main principles for solving such problems are the Pontryagin's maximum principle (PMP) and the dynamic programming (DP) principle. PMP provides a necessary condition for optimality. On the other hand, DP provides a sufficient condition for optimality by solving a partial differential equation, known as the Hamilton–Jacobi–Bellman (HJB) equation [7], [8]. Classical optimal control solutions are offline and require complete knowledge of the system dynamics. Therefore, they are not able to cope with uncertainties and changes in dynamics.

Machine learning [9]–[13] has been used for enabling adaptive autonomy. Machine learning is grouped in supervised, unsupervised, or reinforcement, depending on the amount and quality of feedback about the system or task. In supervised learning, the feedback information provided to learning algorithms is a labeled training data set, and the objective is to build the system model representing the learned relation between the input, output, and system parameters. In unsupervised learning, no feedback information is provided to the algorithm and the objective is to classify the sample sets to different groups based on the similarity between the input samples. Finally, reinforcement learning (RL) is a goal-oriented learning tool wherein the agent or decision maker learns a policy to optimize a long-term reward by interacting with the environment. At each step, an RL agent gets evaluative feedback about the performance of its action, allowing it to improve the performance of subsequent actions [14]–[19]. The term of RL includes all work done in all areas such as psychology, computer science, economic, and so on. A more modern formulation of RL is called approximate DP (ADP).

In a control engineering context, RL and ADP bridge the gap between traditional optimal control and adaptive control algorithms [20]–[26]. The goal is to learn the optimal policy and value function for a potentially uncertain physical system. Unlike traditional optimal control, RL finds the solution to the HJB equation online in real time. On the other hand, unlike traditional adaptive controllers that are not usually designed to be optimal in the sense of minimizing cost functionals, RL algorithms are optimal. This has motivated control system researchers to enable adaptive autonomy in an optimal manner by developing RL-based controllers.

A. Related Theoretical Work

The origin of RL is rooted in computer science and has attracted increasing attention since the seminal work of [27] and [28]. The interest in RL in control society dates back to the work of [15] and [29]–[31]. Watkins' Q-learning algorithm [32] has also made an impact by considering totally unknown environments. Extending RL algorithms to continuous-time (CT) and continuous-state systems was first performed in [33]. The work of [33] used the knowledge of the system models to learn the optimal control policy. The work of [34]–[36] formulated and developed such ideas in a control-theoretic framework for CT systems. The control of switching and hybrid systems using ADP is considered in [37]–[41].

There are generally two basic tasks in RL algorithms. One is called policy evaluation and the other is called policy improvement. Policy evaluation calculates the cost or value

Manuscript received April 3, 2017; revised August 29, 2017; accepted November 1, 2017. Date of publication April 3, 2017; date of current version May 15, 2018. This work was supported in part by the U.S. NSF under Grant ECCS-1405173, in part by ONR under Grant N00014-17-1-2239, in part by China NSFC under Grant 61633007, and in part by NATO through the Virginia Tech Startup Fund under Grant SPS G5176. *(Corresponding author: Bahare Kiumarsi.)*

B. Kiumarsi is with the UTA Research Institute, University of Texas at Arlington, Arlington, TX 76118 USA (e-mail: b_kiomarsi@yahoo.com).

K. G. Vamvoudakis is with the Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24061-0203 USA (e-mail: kyriakos@vt.edu).

H. Modares is with the Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO 65401 USA (e-mail: modaresh@mst.edu).

F. L. Lewis is with the UTA Research Institute, University of Texas at Arlington, Arlington, TX 76118 USA, and also with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110004, China (e-mail: lewis@uta.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2017.2773458

function related to the current policy, and policy improvement assesses the obtained value function and updates the current policy. Two main classes of RL algorithms that are used for performing these two steps are known as policy iteration (PI) [14] and value iteration (VI). PI and VI algorithms iteratively perform policy evaluation and policy improvement until an optimal solution is found. PI methods start with an admissible control policy [42], [43] and solve a sequence of Bellman equations to find the optimal control policy. In contrast to PI methods, VI methods do not require an initial stabilizing control policy. While most of RL-based control algorithms are PI, some VI algorithms have also been developed to learn optimal control solutions. We mostly survey PI-based RL algorithms that are used for feedback control design.

RL algorithms in control context have been mainly used to solve: 1) optimal regulation and optimal tracking of single-agent systems [1] and 2) optimal coordination of multiagent systems [44]. The objective of the optimal regulation problem is to design an optimal controller to assure the states or outputs of the systems converge to zero, or close to zero, while in the optimal tracking control problem, it is desired that the optimal controllers make the states or outputs of the systems track a desired reference trajectory. The goal in optimal coordination of multiagent systems is to design distributed control protocols based only on available local information of agents so that agents achieve some team objectives. This paper reviews existing RL-based algorithms in solving optimal regulation and tracking of single-agent systems and game-based coordination of multiagent systems.

Finally, RL algorithms that are used to solve optimal control problems are categorized in two classes of learning control methods, namely, on-policy and off-policy methods [14]. On-policy methods evaluate or improve the same policy as the one that is used to make decisions. In off-policy methods, these two functions are separated. The policy used to generate data, called the behavior policy, may in fact be unrelated to the policy that is evaluated and improved, called the estimation policy or target policy. The learning process for the target policy is online, but the data used in this step can be obtained offline by applying the behavior policy to the system dynamics. The off-policy methods are data efficient and fast since a stream of experiences obtained from executing a behavior policy is reused to update several value functions corresponding to different estimation policies. Moreover, off-policy algorithms take into account the effect of probing noise needed for exploration. Fig. 1 shows the schematic of on-policy and off-policy RL.

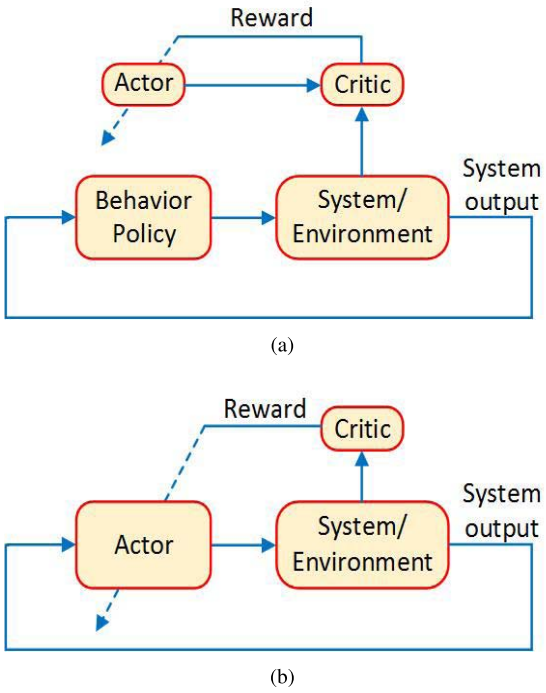

Fig. 1. Two different categories of RL. In on-policy RL, the policy that is applied to the system (behavior policy) to generate data for learning is the same as the policy that is being learned (learned policy) to find the optimal control solution. In off-policy RL, on the other hand, these two policies are separated and can be different. (a) Off-policy RL diagram. (b) On-policy RL diagram.

B. Structure

This paper surveys the literature on RL and autonomy. In Section II, we present the optimal control problems for discrete-time (DT) dynamical systems and their online solutions using RL algorithms. This section includes optimal regulation problem, tracking control problem, and $\mathcal{H}_\infty$ problem for both linear and nonlinear DT systems. In Section III, we discuss several recent developments of using RL for designing the optimal controllers for CT dynamical systems. Nash game problems and their online solutions are discussed at the end of this section. The RL solution to games on graphs is presented in Section IV. Finally, we talk about applications and provide open research directions in Sections V and VI.

II. Optimal Control of DT Systems and Online Solutions

Consider the nonlinear time-invariant system given as

$$\begin{aligned} x(k+1) &= f(x(k)) + g(x(k))u(k) \\ y(k) &= l(x(k)) \end{aligned} \tag{1}$$

where $x(k) \in \mathbb{R}^n$, $u(k) \in \mathbb{R}^m$, and $y(k) \in \mathbb{R}^p$ represent the state of the system, the control input, and the output of the system, respectively. $f(x(k)) \in \mathbb{R}^n$ is the drift dynamics, $g(x(k)) \in \mathbb{R}^{n\times m}$ is the input dynamics, and $l(x(k)) \in \mathbb{R}^p$ is the output dynamics. It is assumed that $f(0) = 0$ and $f(x(k)) + g(x(k))u(k)$ is locally Lipschitz and the system is stabilizable. This is a standard assumption to make sure the solution $x(t)$ of the system (1) is unique for any finite initial condition.

A. Optimal Regulation Problem

The goal of optimal regulation is to design an optimal control input to stabilize the system in (1) while minimizing a predefined cost functional. Such energy-related cost functional

can be defined as

$$J = \sum_{i=0}^{\infty} U(x(i), u(i)) \equiv \sum_{i=0}^{\infty} (Q(x) + u^T(i) R u(i))$$

where $Q(x) \succeq 0$ and $R = R^T \succ 0$. Hence, the problem to be solved can be defined as

$$V(x(k)) = \min_u \left[\sum_{i=k}^{\infty} (Q(x) + u^T(i) R u(i)) \right], \quad \forall x(k).$$

The value function given $u(k)$ can be defined as

$$\begin{aligned} V(x(k)) &= \sum_{i=k}^{\infty} U(x(i), u(i)) \\ &\equiv \sum_{i=k}^{\infty} (Q(x) + u^T(i) R u(i)), \quad \forall x. \end{aligned} \tag{2}$$

An equivalent to (2) is the Bellman equation

$$V(x(k)) = U(x(k), u(k)) + V(x(k+1)). \tag{3}$$

The associated Hamiltonian is defined by

$$\begin{aligned} &H(x(k), u(k), V) \\ &\quad = U(x(k), u(k)) + V(x(k+1)) - V(x(k)), \quad \forall x, u. \end{aligned}$$

The Bellman optimality principle [1] gives the optimal value function as

$$V^\star(x(k)) = \min_u [U(x(k), u(k)) + V^\star(x(k+1))]$$

which is termed the DT HJB equation. The optimal control is derived to be

$$\begin{aligned} u^\star(x(k)) &= \arg\min_u [U(x(k), u(k)) + V^\star(x(k+1))] \\ &= -\frac{1}{2} R^{-1} g^T(x) \left(\frac{\partial V^\star(x(k+1)}{\partial x(k+1)} \right). \end{aligned}$$

B. Special Case

For the DT linear systems, the dynamics (1) become

$$x(k+1) = Ax(k) + Bu(k) \quad y(k) = Cx(k) \tag{4}$$

where A, B, and C are constant matrices with appropriate dimensions.

By assuming that $Q(x) = x^T(k) Q x(k), Q \succ 0$ in the Bellman equation, the value function is quadratic in the current state so that

$$V(x(k)) = x^T(k) P x(k), \quad \forall x. \tag{5}$$

Then, the DT HJB becomes the DT algebraic Riccati equation (DARE)

$$Q - P + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A = 0$$

and the optimal control input is

$$u^\star(k) = -(R + B^T P B)^{-1} B^T P A x(k), \quad \forall x.$$

C. Approximate Solution Using RL

The HJB equation is generally extremely difficult or even impossible to solve analytically and one needs to approximate its solution. Existing methods for approximating the HJB equation and DARE require complete knowledge of the system dynamics. The following PI algorithm can be used to approximate the HJB equation and DARE solutions by performing successive iterations.

1) Offline PI Algorithm: Algorithm 1 presents an offline solution to the DT HJB equation but requires complete knowledge of the system dynamics.

Algorithm 1 PI Algorithm to Find the Solution of HJB

1: **procedure**
2: Given admissible policy $u_0(k)$
3: for $j = 0, 1, \ldots$ given u_j, solve for the value $V_{j+1}(x)$ using Bellman equation

$$V_{j+1}(x(k)) = Q(x) + u_j^T(k) R u_j(k) + V_{j+1}(x(k+1)),$$

on convergence, set $V_{j+1}(x(k)) = V_j(x(k))$.
4: Update the control policy $u_{j+1}(k)$ using

$$u_{j+1}(k) = -\frac{1}{2} R^{-1} g^T(x) \left(\frac{\partial V_{j+1}(x(k+1))}{\partial x(k+1)} \right).$$

5: Go to 3
6: **end procedure**

2) Actor and Critic Approximators: To approximate the solution of the HJB equation and obviate the requirement of the complete knowledge about the system dynamics, actor-critic structure has been widely presented to find the online solution to the HJB. The critic approximator estimates the value function and is updated to minimize the Bellman error. The actor approximator approximates the control policy and is updated to minimize the value function [15], [29], [30], [45].

The value function is represented at each step as

$$\hat{V}_j(x(k)) = \hat{W}_{cj}^T \phi_c(x(k)) = \sum_{i=1}^{N_c} \hat{w}_{cj}^i \phi_{c_i}(x(k)), \quad \forall x \tag{6}$$

and the control input as

$$\hat{u}_j(k) = \hat{W}_{aj}^T \sigma_a(x(k)) = \sum_{i=1}^{N_a} \hat{w}_{aj}^i \sigma_{a_i}(x(k)), \quad \forall x \tag{7}$$

where $\phi_{c_i}(x(k))$ and $\sigma_{a_i}(x(k))$ are the basis functions, $\hat{W}_{cj} = [\hat{w}_{cj}^1 \hat{w}_{cj}^2 \ldots \hat{w}_{cj}^{N_c}]^T$ is the weight vector of the critic approximator with N_c the number of basis functions used, and $\hat{W}_{aj} = [\hat{w}_{aj}^1 \hat{w}_{aj}^2 \ldots \hat{w}_{aj}^{N_a}]^T$ is the weight vector of the actor approximator with N_a the number of basis functions used.

The Bellman equation (3) in terms of the critic approximator (6) is written as

$$\hat{W}_{c(j+1)}^T \phi_c(x(k)) = U(x(k), \hat{u}_j(k)) + \hat{W}_{c(j+1)}^T \phi_c(x(k+1)).$$

The gradient descent tuning laws for the critic and actor approximators are given as

$$\hat{W}_{c(j+1)}^{l+1} = \hat{W}_{c(j+1)}^{l} - \alpha_1 \phi_c(x(k))((\hat{W}_{c(j+1)}^{l})^T \phi_c(x(k)) - U(x(k), \hat{u}_j(k))) \quad (8)$$

$$\hat{W}_{a(j+1)}^{l+1} = \hat{W}_{a(j+1)}^{l} - \alpha_2 \sigma_a(x(k)) \times \left(2R\left(W_{a(j+1)}^{l}\right)^T \times \sigma_a(x(k)) + g(x(k))^T \frac{\partial \phi(x(k+1))}{\partial x(k+1)} \hat{W}_{c(j+1)}\right)^T \quad (9)$$

where $\alpha_1, \alpha_2 > 0$ are the tuning gains. Note that using actor-critic structure to evaluate the value function and improve the policy for nonlinear systems does not need complete knowledge of the system dynamics. In [46], synchronous methods are given to tune the actor and critic approximators. In [47], an online approximator approach is used to find the solution HJB equation without requiring the knowledge of the internal system dynamics. In [48], a greedy iterative heuristic DP is introduced to obtain the optimal saturated controller using three neural networks to approximate the value function, the optimal control policy, and the model of the unknown plant.

Remark 1: Note that the structure of the value function approximator is an important factor in convergence and performance of Algorithm 1. If an inappropriate value function approximator is chosen, the algorithm may never converge to an optimal solution. For linear systems, the form of the value function is known to be quadratic, and, therefore, there would be no error in its approximation. Consequently, Algorithm 1 converges to the global optimal solution for the linear systems. That is, for linear systems, Algorithm 1 gives the exact solution to the DARE. For nonlinear system, using single-layer neural networks for value function approximation may require a large number of activation functions to assure a good approximation error, and consequently, a near-optimal solution. Two-layer neural networks are used in [49] to achieve a better approximation error with lesser number of activation functions. Moreover, error-tolerant ADP-based algorithms are presented for DT systems [50] that guarantees stability in the presence of approximation error in the value function approximation. □

3) Event-Triggered RL [51]: In order to reduce the communication between the controller and the plant, one needs to use an event-triggered control algorithm. The event-triggering mechanism determines when the control signal has to be transmitted so that the resulting event-based control executions still achieve some degree of performance and stabilize the system. For the nonlinear DT systems, three neural networks are used to find the event-triggered-based approximation of the HJB equation.

The nonlinear system can be represented as

$$x(k+1) = W_m^{\star T} \phi_m(x_m(k)) + \epsilon_m \quad (10)$$

where $W_m^{\star}$ is the target weights of model network from the hidden layer to the output layer, $x_m(k)$ is the input vector of the hidden layer, and ϵ_m is the bounded approximation error.

The system (10) using current estimates $\hat{W}_m$ of the ideal weights $W_m^{\star}$ is given as

$$\hat{x}(k+1) = \sum_{i=1}^{N_{mh}} w_{mi}^2 s_i(k)$$

with

$$s_i(k) = \frac{1 - e^{-h_i(k)}}{1 + e^{-h_i(k)}} \quad i = 1, \ldots, N_{mh}$$

$$h_i(k) = \sum_{j=1}^{n+m} w_{mi,j}^1 x_{mj}(k) \quad i = 1, \ldots, N_{mh}$$

where w_m^1 and w_m^2 are the weight matrices. h_i is the input of the ith hidden node, and s_i is its corresponding output. N_{mh} is the number of hidden neurons. $x_m(k)$ is the input of the model network that includes the sampled state vector and the corresponding control law.

The gradient descent can be used to update the weights of the network to minimize $e_m = \hat{x}(k+1) - x(k+1)$. The value function can be approximated by a network similar to (6). The gradient descent can be used to update the weights of the network to minimize

$$e_c(k) = J(x(k_i)) - [J(\hat{x}(k_i + 1)) + U(k)]$$

where J is the output of critic network and defined as

$$J(x(k_i)) = \sum_{l=1}^{N_{ch}} w_{cl}^2 q_l(k)$$

with

$$q_l(k) = \frac{1 - e^{-p_l(k)}}{1 + e^{-p_l(k)}} \quad l = 1, \ldots, N_{ch}$$

$$p_l(k) = \sum_{j=1}^{n} w_{cl,j}^1 x_{cj}(k_i) \quad l = 1, \ldots, N_{ch}$$

where p_l and q_l are the input and the output of the hidden nodes of the critic network, respectively, and N_{ch} is the total number of the hidden nodes. The input for the critic network, $x_c(k_i)$, is the sampled state vector only, so there are n input nodes in the critic network.

The sampled state $x(k_i)$ is used as input to learn the event-triggered control law, which is defined as

$$\mu(x(k_i)) = \sum_{l=1}^{N_{ah}} w_{al}^2 v_l(k)$$

with

$$v_l(k) = \frac{1 - e^{-t_l(k)}}{1 + e^{-t_l(k)}} \quad l = 1, \ldots, N_{ah}$$

$$t_l(k) = \sum_{j=1}^{n} w_{al,j}^1 x_{aj}(k_i) \quad l = 1, \ldots, N_{ah}$$

where t_l and v_l are the input and the output of the hidden nodes of the action network, respectively. N_{ah} is the total number of the hidden nodes in the action network.

The gradient descent can be used to update the weights of the actor network. In [52], the event-triggered finite-time optimal control scheme is designed for an uncertain nonlinear DT system.

4) Q Function for the DT Linear Quadratic Regulation: For linear systems, the work of [32] proposed an action-dependent function (Q-function) instead of value function in the Bellman equation to avoid knowing the system dynamics. This is termed in the literature as Q-learning [53], [54].

Based on (3) and (5), the DT Q-function is defined as

$$Q(x(k),u(k)) = x^T(k)Qx(k) + u^T(k)Ru(k) + x^T(k+1)Px(k+1), \quad \forall u, x. \quad (11)$$

Using the dynamics (4), the Q-function (11) becomes

$$Q(x(k),u(k)) = \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q + A^T P A & A^T P B \\ B^T P A & R + B^T P B \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}.$$

Define

$$Q(x(k),u(k)) = \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} S_{xx} & S_{xu} \\ S_{ux} & S_{uu} \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} = Z^T(k) S Z(k)$$

for a kernel matrix S.

By applying the stationarity condition $(\partial Q(x(k),u(k))/\partial u(k)) = 0$, one has

$$u^\star(k) = -(R + B^T P B)^{-1} B^T P A x(k)$$

and

$$u^\star(k) = -S_{uu}^{-1} S_{ux} x(k).$$

Algorithm 2 Q-Learning Algorithm for DARE

1: **procedure**
2: Given admissible policy $u_0(k)$
3: for $j = 0, 1, \ldots$ given u_j, solve for the value S_{j+1} using Bellman equation

$$Z^T(k)\, S_{j+1}\, Z(k) = x^T(k)\, Q\, x(k) + u_j^T(k)\, R\, u_j(k) + Z^T(k+1)\, S_{j+1}\, Z(k+1),$$

on convergence, set $S_{j+1} = S_j$.
4: Update the control policy $u_{j+1}(k)$ using

$$u_{j+1}(k) = -(S_{uu})_{j+1}^{-1} (S_{ux})_{j+1}\, x(k).$$

5: go to 3
6: **end procedure**

Algorithm 2 is a model-free learning approach. This algorithm converges to the global optimal solution, on condition that a persistence of excitation (PE) condition is satisfied [55]. The PE condition guarantees the uniqueness of the policy evaluation step at each iteration. However, full information of the states of the system is needed. In [56], output-feedback (OPFB) RL algorithms are derived for linear systems. These algorithms do not require any knowledge of the system dynamics and, as such, are similar to Q-learning and they have an added advantage of requiring only measurements of input/output data and not the full system state.

D. Optimal Tracking Problem

The goal now is to design an optimal control input to make the states of the system $x(k)$ follow a desired reference trajectory $x_d(k)$. Let us now define the tracking error $e(k)$ as

$$e(k) = x(k) - x_d(k).$$

In the tracking problem, the control input consists of two terms: a feedforward term that guarantees tracking and a feedback term that stabilizes the system.

The feedforward term can be obtained using the dynamics inversion concept as

$$u_d(k) = g(x_d(k))^{-1}(x_d(k+1) - f(x_d(k))).$$

Consider the following cost functional:

$$J(e(k), u_e(k)) = \sum_{i=k}^{\infty} \left(e^T(i) Q_e e(i) + u_e^T(i) R_e u_e(i) \right)$$

where $Q_e \succeq 0$ and $R_e = R_e^T \succ 0$. The feedback input can be found by applying the following stationarity condition, $\partial J(e, u_e)/\partial u_e = 0$ as:

$$u_e^\star(k) = -\frac{1}{2} R_e^{-1} g^T(k) \frac{\partial J(e(k+1))}{\partial e(k+1)}.$$

Then, the optimal control input including both feedback and feedforward terms is

$$u^\star(k) = u_d(k) + u_e^\star(k).$$

Obtaining the feedforward part of the control input needs complete knowledge of the system dynamics and the reference trajectory dynamics. In [57] and [58], a new formulation is developed that gives both feedback and feedforward parts of the control input simultaneously and thus enables RL algorithms to solve the tracking problems without requiring the complete knowledge of the system dynamics.

Assume now that the reference trajectory is generated by the following command generator model:

$$x_d(k+1) = \psi(x_d(k))$$

where $x_d(k) \in \mathbb{R}^n$. Then, an augmented system can be constructed in terms of the tracking error $e(k)$ and the reference trajectory $x_d(k)$ as

$$\begin{bmatrix} e(k+1) \\ x_d(k+1) \end{bmatrix} = \begin{bmatrix} f(e(k)+x_d(k)) - \psi(x_d(k)) \\ \psi(x_d(k)) \end{bmatrix} + \begin{bmatrix} g(e(k)+x_d(k)) \\ 0 \end{bmatrix} u(k) \equiv F(X(k)) + G(X(k))u(k)$$

where the augmented state is

$$X(k) = \begin{bmatrix} e(k) \\ x_d(k) \end{bmatrix}.$$

The new cost functional is defined as

$$J(x(0), x_d(0), u(k)) = \sum_{i=0}^{\infty} \gamma^{i-k} \times ((x(k) - x_d(k))^T Q (x(k) - x_d(k)) + u^T(i) R u(i))$$

where $Q \succeq 0$ and $R = R^T \succ 0$. The value function in terms of the states of the augmented system is written as

$$V(X(k)) = \sum_{i=k}^{\infty} \gamma^{i-k}(U(X(k), u(k)) = \sum_{i=k}^{\infty} \gamma^{i-k}(X^T(i)Q_T X(i) + u^T(i)Ru(i)) \quad (12)$$

where

$$Q_T = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}$$

and $0 < \gamma \leq 1$ is the discount factor.

Remark 2: Note that it is essential to use a discounted performance function for the proposed formulation. This is because if the reference trajectory does not go to zero, which is the case of most real applications, then the value is infinite, without the discount factor as the control input contains a feedforward part that depends on the reference trajectory and thus $u^T(k)Ru(k)$ does not go to zero as time goes to infinity. □

A difference equivalent to (12) is

$$V(X(k)) = U(X(k), u(k) + \gamma V(X(k+1)).$$

The Hamiltonian for this problem is given as

$$H(X(k), u(k), V) = X^T(k)Q_T X(k) + u^T(k)Ru(k) + \gamma V(x(k+1)) - V(x(k)).$$

The optimal value can be found [59]

$$V^\star(X(k)) = \min_u [U(X(k), u(k)) + \gamma V^\star(X(k+1))]$$

which is just the DT HJB equation. The optimal control is then given as

$$u^\star(X(k)) = \arg\min_u [U(X(k), u(k)) + \gamma V^\star(X(k+1))] = -\frac{\gamma}{2} R^{-1} G^T(X) \left(\frac{\partial V^\star(X(k+1)}{\partial X(k+1)} \right).$$

E. Approximate Solution Using RL

The solution of the DT HJB tracking equation can be approximated as follows.

1) Offline PI Algorithm: The PI algorithm is used to find the solution of DT HJB tracking by iterating on the solution of the Bellman equation.

The augmented system dynamics must be known in order to update the control input in Algorithm 3. Convergence properties of Algorithm 3 are similar to Algorithm 1 and are not discussed here.

2) Online Actor and Critic Approximators: To obviate the requirement of complete knowledge of the system dynamics or reference trajectory dynamics, an actor-critic structure, similar to (6) and (7), is developed in [57] for solving the nonlinear optimal tracking problem. In [58], the Q-learning is used to find the optimal solution for linear systems. Kiumarsi *et al.* [60] presented PI and VI algorithms to solve the linear quadratic tracker (LQT) ARE online without requiring any knowledge of the system dynamics and information of the states of the system only using the measured input and output data.

Algorithm 3 PI Algorithm to Find the Solution of Tracking HJB

1: **procedure**
2: Given admissible policy $u_0(k)$
3: for $j = 0, 1, \ldots$ given u_j, solve for the value $V_{j+1}(x)$ using Bellman equation

$$V_{j+1}(X(k)) = X^T(k)Q_T X(k) + u_j^T(k) R u_j(k) + \gamma V_{j+1}(X(k+1))$$

on convergence, set $V_{j+1}(X(k)) = V_j(X(k))$.
4: Update the control policy $u_{j+1}(k)$ using

$$u_{j+1}(X(k)) = -\frac{\gamma}{2} R^{-1} G^T(X) \left(\frac{\partial V_{j+1}(X(k+1))}{\partial X(k+1)} \right).$$

5: Go to 3
6: **end procedure**

F. $\mathcal{H}_\infty$ Control of DT Systems

The $\mathcal{H}_\infty$ control problem can be considered as a zero-sum game, where the controller and the disturbance inputs are considered as minimizing and maximizing players, respectively [61]–[64]. In the linear systems, the optimal solution to the zero-sum game problem leads to solving the GARE.

Consider now the dynamics with an added disturbance input

$$x(k+1) = f(x(k)) + g(x(k))u(k) + h(x(k))d(k) \quad (13)$$

where $x(k) \in \mathbb{R}^n$ is a measurable state vector, $u(k) \in \mathbb{R}^m$ is the control input, $d(k) \in \mathbb{R}^q$ is the disturbance input, $f(x(k)) \in \mathbb{R}^n$ is the drift dynamics, $g(x(k)) \in \mathbb{R}^{n \times m}$ is the input dynamics, and $h(x(k)) \in \mathbb{R}^{n \times q}$ is the disturbance input dynamics.

Define the cost functional to be optimized as

$$J(x(0), u(k), d(k)) = \sum_{i=0}^{\infty} (Q(x) + u^T(i)Ru(i) - \beta^2 d^T(i)d(i))$$

where $Q(x) \succeq 0$, $R = R^T \succ 0$, and $\beta \geq \beta^\star \geq 0$ with $\beta^\star$ the smallest β such that the system is stabilized. The value function with feedback control and disturbance policies can be defined as

$$V(x(k), u(k), d(k)) = \sum_{i=k}^{\infty} (Q(x) + u^T(i)Ru(i) - \beta^2 d^T(i)d(i)).$$

A difference equivalent to this is

$$V(x(k)) = Q(x) + u^T(k)Ru(k) - \beta^2 d^T(k)d(k) + V(x(k+1)).$$

We can find the optimal value, by solving a zero-sum differential game as

$$V^\star(x(k)) = \min_u \max_d J(x(0), u, d)$$

subject to (13). It is worth noting that $u(k)$ is the minimizing player while $d(k)$ is the maximizing one.

In order to solve this zero-sum game, one needs to solve the following Hamilton–Jacobi–Isaacs (HJI) equation:

$$V^\star(x(k)) = Q(x) + u^{\star T}(k) R u^\star(k) - \beta^2 d^{\star T}(k) d^\star(k) + V^\star(x(k+1)).$$

Given a solution $V^\star$ to this equation, one can find the optimal control and the worst case disturbance as

$$u^\star(k) = -\frac{1}{2}R^{-1}g^T(x)\left(\frac{\partial V^\star(x(k+1))}{\partial x(k+1)}\right)$$

and

$$d^\star(k) = \frac{1}{2\beta^2}h^T(x)\left(\frac{\partial V^\star(x(k+1))}{\partial x(k+1)}\right)$$

respectively.

G. Approximate Solution Using RL

In the following sections, on-policy and off-policy RL algorithms are presented to solve the HJI and GARE solutions by successive approximations to the Bellman equations.

1) On-Policy RL to Solve $\mathcal{H}_\infty$ Control of DT Systems: The following on-policy RL algorithm is given to solve the HJI.

Algorithm 4 PI Algorithm to Solve the HJI

1: **procedure**
2: Given admissible policies $u_0(k)$ and $d_0(k)$
3: for $j = 0, 1, \ldots$ given u_j and d_j, solve for the value $V_{j+1}(x(k))$ using Bellman equation

$$V_{j+1}(x(k)) = Q(x) + u_j^T(k)\,R\,u_j(k) - \beta^2 d_j^T(k)d_j(k) + V_{j+1}(x(k+1)),$$

on convergence, set $V_{j+1}(x(k)) = V_j(x(k))$.
4: Update the control policy $u_{j+1}(k)$ and disturbance policy $d_{j+1}(k)$ using

$$u_{j+1}(k) = -\frac{1}{2}R^{-1}g^T(x)\left(\frac{\partial V_{j+1}(k+1)}{\partial x(k+1)}\right),$$

$$d_{j+1}(k) = \frac{1}{2\beta^2}h^T(x)\left(\frac{\partial V_{j+1}(x(k+1))}{\partial x(k+1)}\right).$$

5: Go to 3.
6: **end procedure**

The PI algorithm (Algorithm 4) is an offline algorithm and requires complete knowledge of the system dynamics. The actor-critic structure can be used to design an online PI algorithm that simultaneously updates the value function and policies and does not require the knowledge of the drift dynamics. The value function and control input are approximated as (6) and (7), respectively. The disturbance input is approximated as

$$\hat{d}_j(k) = \hat{W}_{dj}^T\sigma_d(x(k)) = \sum_{i=1}^{N_d}\hat{w}_{dj}^i\sigma_{d_i}(x(k))$$

where $\sigma_{d_i}(x(k))$ is the basis function for the approximator, $\hat{W}_{dj} = [\hat{w}_{dj}^1\hat{w}_{dj}^2\ldots\hat{w}_{dj}^{N_d}]^T$ is the weight vector, and N_d is the number of basis functions used.

The weights of the value function and control input approximators are updated using gradient descent as (8) and (8), respectively. The gradient descent can also be used to update the weights of the disturbance input approximator.

A Q-learning algorithm is presented in [65] to find the optimal control input and worst case disturbance input for linear systems without requiring any knowledge of the system dynamics. However, the disturbance input needs to be updated in a prescribed manner. An off-policy RL algorithm is presented in [66] that does not need any knowledge of the system dynamics and the disturbance input does not need to be updated in a prescribed manner. Fig. 2 shows a schematic of off-policy RL for solving the zero-sum game problem.

2) Off-Policy RL for Solving Zero-Sum Game Problem of DT Systems [66]: Consider the following system description:

$$x(k+1) = Ax(k) + Bu(k) + Dd(k) \tag{14}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and $d \in \mathbb{R}^q$ are the state, control, and disturbance inputs, and A, B, and D are constant matrices of appropriate dimensions. To derive an off-policy RL algorithm, the original system (14) is rewritten as

$$x(k+1) = A_k x(k) + B(K_j^1 x(k) + u(k)) + D(K_j^2 x(k) + d(k)) \tag{15}$$

where $A_k = A - BK_j^1 - DK_j^2$.

In (15), the estimation policies are $u_j(k) = -K_j^1 x(k)$ and $d_j(k) = -K_j^2 x(k)$. By contrast, $u(k)$ and $d(k)$ are the behavior policies that are actually applied to (14) to generate the data for learning.

Using the Taylor expansion of the value function $V(x(k))$ at point $x(k+1)$ for (14), the value function (5) yields

$$\begin{aligned} x^T(k)P_{j+1}x(k) - x^T(k+1)P_{j+1}x(k+1) &= x^T(k)Qx(k) + x^T(k)\left(K_j^1\right)^T RK_j^1 x(k) - \beta^2 x^T(k)\left(K_j^2\right)^T \\ &\quad \times K_j^2 x(k) - \left(u(k) + K_j^1 x(k)\right)^T B^T P_{j+1}x(k+1) \\ &\quad - \left(u(k) + K_j^1 x(k)\right)^T B^T P_{j+1}A_k x(k) \\ &\quad - \left(K_j^2 x(k) + d(k)\right)^T D^T P_{j+1}x(k+1) \\ &\quad - \left(K_j^2 x(k) + d(k)\right)^T D^T P_{j+1}A_k x(k). \end{aligned} \tag{16}$$

Remark 3: Note that (16) does not explicitly depend on K_{j+1}^1 and K_{j+1}^2. Also, complete knowledge of the system dynamics is required for solving (16). In the following, it is shown how to find $(P_{j+1}, K_{j+1}^1, K_{j+1}^2)$ simultaneously by (16) without knowing any knowledge of the system dynamics. □

Given any matrix $M \in \mathbb{R}^{n\times m}$, $\text{vec}(M) \in \mathbb{R}^{nm\times 1}$ is transpose of a vector formed by stacking the rows of matrix M.

Using $a^T Wb = (b^T \otimes a^T)\text{vec}(W)$, (14), and $A_k = A - BK_j^1 - DK_j^2$, the off-policy Bellman equation (16) can be

rewritten as

$$\begin{aligned}
&(x^T(k)\otimes x^T(k))\mathrm{vec}(L^1_{j+1})\\
&-(x^T(k+1)\otimes x^T(k+1))\mathrm{vec}(L^1_{j+1})\\
&+2\big(x^T(k)\otimes\big(u(k)+K^1_j x(k)\big)^T\big)\mathrm{vec}(L^2_{j+1})\\
&-\big(\big(K^1_j x(k)-u(k)\big)^T\otimes\big(u(k)+K^1_j x(k)\big)^T\big)\mathrm{vec}(L^3_{j+1})\\
&+2\big(x^T(k)\otimes\big(d(k)+K^2_j x(k)\big)^T\big)\mathrm{vec}(L^4_{j+1})\\
&-\big(\big(K^1_j x(k)-u(k)\big)^T\otimes\big(d(k)+K^2_j x(k)\big)^T\big)\mathrm{vec}(L^5_{j+1})\\
&+\big(\big(d(k)-K^2_j x(k)\big)^T\otimes\big(u(k)+K^1_j x(k)\big)^T\big)\mathrm{vec}(L^6_{j+1})\\
&+\big(\big(d(k)-K^2_j x(k)\big)^T\otimes\big(d(k)+K^2_j x(k)\big)^T\big)\mathrm{vec}(L^7_{j+1})\\
&=x^T(k)Qx(k)+x^T(k)\big(K^1_j\big)^T RK^1_j x(k)\\
&\quad-\beta^2 x^T(k)\big(K^2_j\big)^T K^2_j x(k) \qquad (17)
\end{aligned}$$

with $L^1_{j+1}=P_{j+1}$, $L^2_{j+1}=B^TP_{j+1}A$, $L^3_{j+1}=B^TP_{j+1}B$, $L^4_{j+1}=D^TP_{j+1}A$, $L^5_{j+1}=D^TP_{j+1}B$, $L^6_{j+1}=B^TP_{j+1}D$, and $L^7_{j+1}=D^TP_{j+1}D$ that are the unknown variables in the Bellman equation (17). Then, using (17), one has

$$\begin{aligned}
\psi_j\big[&\mathrm{vec}\big(L^1_{j+1}\big)^T\mathrm{vec}\big(L^2_{j+1}\big)^T\mathrm{vec}\big(L^3_{j+1}\big)^T\mathrm{vec}\big(L^4_{j+1}\big)^T\\
&\mathrm{vec}\big(L^5_{j+1}\big)^T\mathrm{vec}\big(L^6_{j+1}\big)^T\mathrm{vec}\big(L^7_{j+1}\big)^T\big]^T=\phi_j \qquad (18)
\end{aligned}$$

where $\phi_j=[(\phi^1_j)^T\ldots(\phi^s_j)^T]^T$ and $\psi^j=[(\psi^1_j)^T\ldots(\psi^s_j)^T]^T$ with ϕ^l_j and ψ^l_j defined as in [66]. One can solve (18) using a least-squares (LS) method.

Algorithm 5 Off-Policy PI to Find the Solution of GARE

1: **procedure**
2: Set the iteration number $j=0$ and start with a admissible control policy $u(k)=-K^1x(k)+e(k)$ where $e(k)$ is probing noise.
3: **while** $\|K^1_{j+1}-K^1_j\|\geq\epsilon$ and $\|K^2_{j+1}-K^2_j\|\geq\epsilon$ **do**
4: For $j=0,1,2,\ldots$, solve (18) using LS

$$\begin{aligned}
\psi_j\big[&vec(L^1_{j+1})^T\,vec(L^2_{j+1})^T\,vec(L^3_{j+1})^T\,vec(L^4_{j+1})^T\\
&vec(L^5_{j+1})^T\,vec(L^6_{j+1})^T\,vec(L^7_{j+1})^T\big]^T=\phi_j.
\end{aligned}$$

5: Update the control and disturbance gains as

$$\begin{aligned}
K^1_{j+1}=&-(R+L^3_{j+1}+L^6_{j+1}(\beta^2\mathcal{I}-L^7_{j+1})^{-1}L^5_{j+1})^{-1}\\
&\times(L^2_{j+1}+L^6_{j+1}(\beta^2\mathcal{I}-L^7_{j+1})^{-1}L^4_{j+1}),\\
K^2_{j+1}=&(L^7_{j+1}-\beta^2\mathcal{I}-L^5_{j+1}(R+L^3_{j+1})^{-1}L^6_{j+1})^{-1}\\
&\times(L^4_{j+1}-L^5_{j+1}(R+L^3_{j+1})^{-1}L^2_{j+1}).
\end{aligned}$$

6: $j=j+1$
7: **end while**
8: **end procedure**

III. Optimal Control of CT Systems and Online Solutions

Consider the nonlinear time-invariant system given as

$$\dot{x}(t)=f(x(t))+g(x(t))u(t)\quad y(t)=l(x(t)) \qquad (19)$$

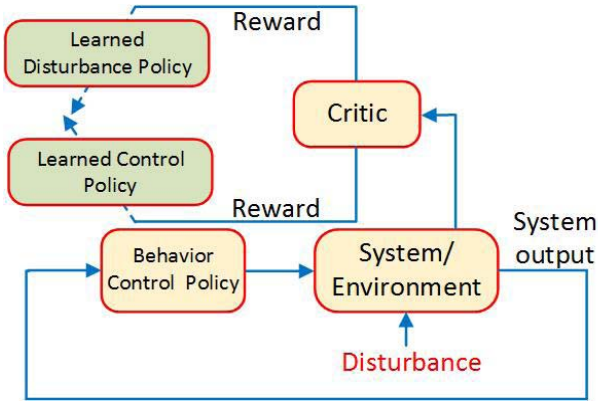

Fig. 2. Off-policy RL for $\mathcal{H}_\infty$ control. The behavior disturbance policy is the actual disturbance from the environment that cannot be specified. On the other hand, the learned disturbance policy uses the collected data to learn the worst case policy.

where $x(t)\in\mathbb{R}^n$, $u(t)\in\mathbb{R}^m$, and $y(t)\in\mathbb{R}^p$ represent the state of the system, the control input, and the output of the system, respectively. $f(x(t))\in\mathbb{R}^n$ is the drift dynamics, $g(x(t))\in\mathbb{R}^{n\times m}$ is the input dynamics, and $l(x(t))\in\mathbb{R}^p$ is the output dynamics. It is assumed that $f(0)=0$ and $f(x(t))+g(x(t))u(t)$ is locally Lipschitz and the system is stabilizable.

A. Optimal Regulation Problem

The goal in optimal regulation is to design an optimal control input to assure the states of the system (19) converge to zero by minimizing a cost functional. The cost functional is defined as

$$J(x(0),u)=\int_0^\infty r(x,u)dt\equiv\int_0^\infty(Q(x)+u^TRu))dt$$

where $Q(x)\succeq 0$ and $R=R^T\succ 0$. The value function for the admissible control policy can be defined as

$$V(x,u)=\int_t^\infty r(x,u)d\tau\equiv\int_t^\infty(Q(x)+u^TRu))d\tau.$$

A differential equivalent to this is

$$r(x,u)+\frac{\partial V}{\partial x}^T(f(x)+g(x)u))=0,\quad V(0)=0$$

and the Hamiltonian is given by

$$H\left(x,u,\frac{\partial V}{\partial x}^T\right)=r(x,u)+\frac{\partial V}{\partial x}^T(f(x)+g(x)u).$$

The optimal value is given by the Bellman optimality equation

$$r(x,u^\star)+\frac{\partial V^\star}{\partial x}^T(f(x)+g(x)u^\star)=0 \qquad (20)$$

which is just the CT HJB equation. The optimal control is then given as

$$\begin{aligned}
u^\star(t)&=\arg\min_u\left(r(x,u)+\frac{\partial V^\star}{\partial x}^T(f(x)+g(x)u)\right)\\
&=-\frac{1}{2}R^{-1}g^T(x)\left(\frac{\partial V^\star}{\partial x}\right). \qquad (21)
\end{aligned}$$

B. Approximate Solution Using RL

The following on-policy and off-policy integral RL (IRL) algorithms are presented to approximate HJB solution by iterating on the Bellman equations.

1) On-Policy IRL: In [35] and [67], an equivalent formulation of the Bellman equation that does not involve the dynamics is found to be

$$V(x(t)) = \int_t^{t+T} (Q(x(\tau)) + u^T(\tau) R u(\tau)) d\tau + V(x(t+T)) \tag{22}$$

for any time $t \geq 0$ and time interval $T > 0$. This equation is called IRL Bellman equation. The following PI algorithm can be implemented by iterating on the above IRL Bellman equation and updating the control policy.

Algorithm 6 On-Policy IRL Algorithm to Find the Solution of HJB

1: **procedure**
2: Given admissible policy u_0
3: for $j = 0, 1, \ldots$ given u_j, solve for the value $V_{j+1}(x)$ using Bellman equation

$$V_{j+1}(x(t)) = \int_t^{t+T} \left(Q(x) + u_j^T R u_j\right) d\tau + V_{j+1}(x(t+T)),$$

on convergence, set $V_{j+1}(x) = V_j(x)$.
4: Update the control policy $u_{j+1}(k)$ using

$$u_{j+1}(t) = -\frac{1}{2} R^{-1} g^T(x) \left(\frac{\partial V_{j+1}(x)}{\partial x}\right).$$

5: Go to 3.
6: **end procedure**

The IRL algorithm (Algorithm 6) is online and does not require the knowledge of the drift dynamics. To implement Step 3 of Algorithm 6, a neural-network-type structure, similar to (6) and (7), is used in [67] to approximate the value function. This algorithm is a sequential RL algorithm in the sense that the actor (policy improvement) and critic (policy evaluation) are updated sequentially. Synchronous update laws for actor and critic were first introduced in [36] to update both actor and critic simultaneously while assuring system stability. Later, in [68] and [69], synchronous actor-critic structure was augmented with system identification to avoid complete knowledge of the system dynamics.

2) On-Policy IRL With Experience Replay Learning Technique [70]–[72]: To speed up and obtain an easy-to-check condition for the convergence of the IRL algorithm, the recent transition samples are stored and repeatedly presented to the gradient-based update rule. A similar condition was proposed in [73] for adaptive control systems. This is a gradient-decent algorithm that does not only minimize the instantaneous temporal difference (TD) error but also minimize the TD errors for the stored transition samples. Assume now that the value function $V(x)$ can be uniformly approximated as in the IRL Bellman equation (22)

$$\hat{V}(x) = \hat{W}_c^T \phi_1(x), \quad \forall x$$

where $\phi_1(x) : \mathbb{R}^n \to \mathbb{R}^N$ is the basis function vector and N is the number of basis functions. Therefore, the approximate IRL Bellman equation becomes

$$e_B = \Delta\phi_1(t)^T \hat{W}_c + \int_{t-T}^{t} (Q(x) + \hat{u}^T R \hat{u}) d\tau$$

where $\Delta\phi_1(t) = \phi_1(t) - \phi_1(t-T)$ and e_B is the TD error after using current critic approximator weights. To collect data in the history stack, consider $\Delta\phi_1(t_j)$ as evaluated values of $\Delta\phi_1$ at the recorded time t_j. Then, define the Bellman equation error (TD error) at the recorded time t_j using the current critic weights estimation $\hat{W}_c$ as

$$(e_B)_j = \Delta\phi_1(t_j)^T \hat{W}_c + \int_{t_j-T}^{t_j} (Q(x(\tau)) + \hat{u}^T(\tau) R \hat{u}(\tau)) d\tau.$$

The experience replay-based gradient-decent algorithm for the critic NN is now given as

$$\dot{\hat{W}}_c = -\alpha_c \frac{\Delta\phi_1(t)}{(\Delta\phi_1(t)^T \Delta\phi_1(t) + 1)^2} e_B - \alpha_c \sum_{j=1}^{l} \frac{\Delta\phi_1(t_j)}{(\Delta\phi_1(t_j)^T \Delta\phi_1(t_j) + 1)^2} (e_B)_j.$$

The first term is a gradient update law for the TD error and the last term minimizes its stored samples in the history stack.

3) Event-Triggered On-Policy RL [74]: The event-triggered version of the optimal controller uses the sampled state information instead of the true one and (21) becomes

$$u^\star(\hat{x}_i) = -\frac{1}{2} R^{-1} g^T(\hat{x}_i) \left(\frac{\partial V^\star(\hat{x}_i)}{\partial x}\right) \quad \forall t \in (r_{i-1}, r_i] \text{ and } i \in N \tag{23}$$

where r_i is the ith consecutive sampling instant and $\hat{x}_i = x(r_i)$.

Using the event-triggered controller (23), the HJB equation (20) becomes $\forall x, \hat{x}_i \in \mathbb{R}^n$

$$\frac{\partial V^{\star T}}{\partial x} \left(f(x) - \frac{1}{2} g(x) R^{-1} g^T(\hat{x}_i) \left(\frac{\partial V^\star(\hat{x}_i)}{\partial x}\right)\right) + Q(x) + \frac{1}{4} \frac{\partial V^\star(\hat{x}_i)}{\partial x}^T g(\hat{x}_i) (R^{-1})^T R^{-1} g(\hat{x}_i)^T \frac{\partial V^\star(\hat{x}_i)}{\partial x} = (u^\star(\hat{x}_i) - u_c^\star)^T R (u^\star(\hat{x}_i) - u_c^\star) \tag{24}$$

where $u_c^\star$ is given by (21).

To solve the event-triggered HJB equation (24), the value function is approximated on a compact set Ω as

$$V^\star(x) = W_c^{\star T} \phi(x) + \epsilon_c(x), \quad \forall x \in \mathbb{R}^n \tag{25}$$

where $\phi(x) : \mathbb{R}^n \to \mathbb{R}^k$ is the basis function set vector, k is the number of basis functions, and $\epsilon_c(x)$ is the approximation error. Based on this, the optimal event-triggered controller in (23) can be rewritten as

$$u^\star(\hat{x}_i) = -\frac{1}{2} R^{-1} g^T(\hat{x}_i) \left(\frac{\partial \phi(\hat{x}_i)}{\partial x}^T W_c^\star + \frac{\partial \epsilon_c(\hat{x}_i)}{\partial x}\right) \quad t \in (r_{i-1}, r_{i1}]. \tag{26}$$

The optimal event-triggered controller (26) can be approximated by the actor for all $t \in (r_{i-1}, r_i]$ as

$$u^\star(\hat{x}_i) = W_u^{\star T} \phi_u(\hat{x}_i) + \epsilon_u(\hat{x}_i), \quad \forall \hat{x}_i, \ i \in N \tag{27}$$

where $\phi_u(\hat{x}_i) : \mathbb{R}^n \to \mathbb{R}^h$ is the basis function set vector, h is the number of basis functions, and $\epsilon_u(\hat{x}_i)$ is the approximation error.

The value function (25) and the optimal policy (27) using current estimates $\hat{W}_c$ and $\hat{W}_u$, respectively, of the ideal weights $W_c^\star$ and $W_u^\star$ are given by the following critic and actor approximators:

$$\hat{V}(x) = \hat{W}_c^T \phi(x), \quad \forall x$$
$$\hat{u}(\hat{x}_i) = \hat{W}_u^T \phi_u(\hat{x}_i), \quad \forall \hat{x}_i.$$

The Bellman error is defined as

$$e_c = \hat{W}_c^T \frac{\partial \phi}{\partial x}(f(x) + g(x)\hat{u}(\hat{x}_i))) + r(x, \hat{u})$$

with $r(x, \hat{u}) = Q(x) + \hat{u}(\hat{x}_i)^T R \hat{u}(\hat{x}_i)$.

The weights $\hat{W}_c$ are tuned to minimize $K = (1/2)e_c^T e_c$ as

$$\dot{\hat{W}}_c = -\alpha_c \frac{\partial K}{\partial \hat{W}_c} = -\alpha_c \frac{w}{(w^T w + 1)^2}(w^T \hat{W}_c + r(x, \hat{u}))$$

with $w = (\partial \phi / \partial x)(f(x) + g(x)\hat{u}(\hat{x}_i)))$.

In order to find the update law for the actor approximator, the following error is defined:

$$e_u = \hat{W}_u^T \phi_u(\hat{x}_i) + \frac{1}{2} R^{-1} g^T(\hat{x}_i) \left(\frac{\partial \phi(\hat{x}_i)}{\partial x} \right)^T \hat{W}_c, \quad \forall \hat{x}_i.$$

The weights $\hat{W}_u$ are tuned to minimize $E_u = (1/2)e_u^T e_u$ as

$$\dot{\hat{W}}_u = 0, \quad \text{for } t \in (r_{i-1}, r_i]$$

and the jump equation to compute $\hat{W}_u(r_j^+)$ given by

$$\hat{W}_u^+ = \hat{W}_u(t) - \alpha_u \phi_u(x(t)) \times \left(\hat{W}_u^T \phi_u(x(t)) + \frac{1}{2} R^{-1} g^T(x(t)) \frac{\partial \phi(x(t))}{\partial x}^T \hat{W}_c \right)^T \quad \text{for } t = r_i.$$

The convergence and stability are proved in [74]. The work of [75] extended [74] to systems with input constraints and without requiring the knowledge of the system dynamics. Event-based RL approaches are also presented in [76]–[78] for interconnected systems.

4) Off-Policy IRL [79], [80]: In order to develop the off-policy IRL algorithm, the system dynamics (19) is rewritten as

$$\dot{x}(t) = f(x(t)) + g(x(t))u_j(t) + g(x(t))(u(t) - u_j(t)) \quad (28)$$

where $u_j(t)$ is the policy to be updated. By contrast, $u(t)$ is the behavior policy that is actually applied to the system dynamics to generate the data for learning.

Differentiating $V(x)$ along with the system dynamics (28) and using $u_{j+1}(t) = -(1/2)R^{-1}g^T(x)(\partial V_j(x)/\partial x)$ give

$$\dot{V}_j = \left(\frac{\partial V_j(x)}{\partial x} \right)^T (f + g u_j) + \left(\frac{\partial V_j(x)}{\partial x} \right)^T g(u - u_j) = -Q(x) - {u_j}^T R u_j - 2{u_{j+1}}^T R(u - u_j).$$

Integrating from both sides of the above equation yields the off-policy IRL Bellman equation

$$V_j(x(t+T)) - V_j(x(t)) = \int_t^{t+T} \big(- Q(x) - u_j^T R u_j - 2{u_{j+1}}^T R(u - u_j) \big) d\tau. \quad (29)$$

Iterating on the IRL Bellman equation (29) yields the following off-policy IRL algorithm.

Algorithm 7 Off-Policy IRL Algorithm to Find the Solution of HJB

1: **procedure**
2: Given admissible policy u_0
3: for $j = 0, 1, \ldots$ given u_j, solve for the value V_j and u_{j+1} using off-policy Bellman equation

$$V_j(x(t+T)) - V_j(x(t)) = \int_t^{t+T} \big(- Q(x) - u_j^T R u_j - 2\,{u_{j+1}}^T R\,(u - u_j) \big) d\tau.$$

on convergence, set $V_{j+1} = V_j$.
4: Go to 3.
5: **end procedure**

Remark 4: Note that for a fixed control policy $u(t)$ (the policy that is applied to the system), the off-policy IRL Bellman equation (29) can be solved for both value function V_j and updated policy u_{j+1}, simultaneously without requiring any knowledge about the system dynamics. □

To implement the off-policy IRL algorithm (Algorithm 7), the actor-critic structure, similar to (6) and (7), is used to approximate the value function and control policy [79].

For a linear system, the off-policy IRL Bellman equation (see [81]) is given as

$$x^T(t+T)P_j x(t+T) - x^T(t)P_j x(t) = \int_t^{t+T} \big(- x^T Q x - u_j^T R u_j - 2{u_{j+1}}^T R(u - u_j) \big) d\tau. \quad (30)$$

Iterating on the Bellman equation (30) yields the off-policy IRL algorithm for the linear systems.

5) Robust Off-Policy IRL [79]: Consider the following uncertain nonlinear system:

$$\dot{x}(t) = f(x(t)) + g(x(t))[u(t) + \Delta(x, w)]$$
$$\dot{w}(t) = \Delta_w(x, w) \quad (31)$$

where $x(t) \in \mathbb{R}^n$ is the measured component of the state available for feedback control, $w(t) \in \mathbb{R}^p$ is the unmeasurable part of the state with unknown order p, $u(t) \in \mathbb{R}$ is the control input, $\Delta_w \in \mathbb{R}^p$ and $\Delta \in \mathbb{R}$ are unknown locally Lipschitz functions, and $f(x(t)) \in \mathbb{R}^n$ and $g(x(t)) \in \mathbb{R}^n$ are the drift dynamics and input dynamics, respectively. In order to develop the robust off-policy IRL algorithm, the system dynamics (31) is rewritten as

$$\dot{x}(t) = f(x(t)) + g(x(t))u_j(t) + g(x(t))v_j(t))$$

where $v_j = u + \Delta - u_j$.

Differentiating $V(x)$ along (31) and using $u_{j+1}(t) = -(1/2)R^{-1}g^T(x)(\partial V_j(x)/\partial x)$ give

$$\dot{V}_j = \left(\frac{\partial V_j(x)}{\partial x}\right)^T (f + gu_j + gv_j) = -Q(x) - {u_j}^T R u_j - 2{u_{j+1}}^T R v_j.$$

Integrating both sides of the above equation yields the robust off-policy IRL Bellman equation

$$V_j(x(t+T)) - V_j(x(t)) = \int_t^{t+T} \left(-Q(x) - u_j^T R u_j - 2{u_{j+1}}^T R v_j\right) d\tau. \quad (32)$$

Using an actor-critic structure, similar to (6) and (7), for V_j and u_{j+1} in (32), yields

$$\sum_{i=1}^{N_c} \hat{w}_{cj}^i \left[\phi_{c_i}(x(t+T)) - \phi_{c_i}(x(t))\right] = \int_t^{t+T} \left(-Q(x) - \hat{u}_j^T R \hat{u}_j - 2\left(\sum_{i=1}^{N_a} \hat{w}_{aj}^i \sigma_{a_i}(x(k))\right)^T R v_j\right) d\tau. \quad (33)$$

Iterating on the IRL Bellman equation (33) provides the following robust off-policy IRL algorithm.

Algorithm 8 Robust Off-Policy IRL Algorithm

1: **procedure**
2: Given admissible policy u_0
3: for $j = 0, 1, \ldots$ given u_j, solve for the value V_j and u_{j+1} using off-policy Bellman equation

$$\sum_{i=1}^{N_c} \hat{w}_{cj}^i \left[\phi_{c_i}(x(t+T)) - \phi_{c_i}(x(t))\right] = \int_t^{t+T} \left(-Q(x) - \hat{u}_j^T R \hat{u}_j - 2\left(\sum_{i=1}^{N_a} \hat{w}_{aj}^i \sigma_{a_i}(x(k))\right)^T R v_j\right) d\tau,$$

on convergence, set $V_{j+1} = V_j$.
4: Go to 3.
5: **end procedure**

C. Optimal Tracking Problem

The goal here is to design an optimal control input to make the states of the system $x(t)$ track a desired reference trajectory $x_d(t)$.

Define the tracking error as

$$e(t) = x(t) - x_d(t).$$

Similar to DT systems, standard techniques find the feedback and feedforward parts of the control input separately using complete knowledge of the system dynamics. In [82] and [83], a new formulation is developed that gives both feedback and feedforward parts of the control input simultaneously and thus enables RL algorithms to solve the tracking problems without requiring complete knowledge of the system dynamics.

Assume that the reference trajectory is generated by the command generator model

$$\dot{x}_d(t) = h_d(x_d(t))$$

where $x_d(t) \in \mathbb{R}^n$. An augmented system can be constructed in terms of the tracking error $e(t)$ and the reference trajectory $x_d(t)$ as

$$\dot{X}(t) = \begin{bmatrix} \dot{e}(t) \\ \dot{x}_d(t) \end{bmatrix} = \begin{bmatrix} f(e(t) + x_d(t)) - h_d(x_d(t)) \\ h_d(x_d(t)) \end{bmatrix} + \begin{bmatrix} g(e(t) + x_d(t)) \\ 0 \end{bmatrix} u(t) \equiv F(X(t)) + G(X(t))u(t) \quad (34)$$

where the augmented state is

$$X(t) = \begin{bmatrix} e(t) \\ x_d(t) \end{bmatrix}.$$

The cost functional is defined as

$$J(x(0), x_d(0), u(t)) = \int_0^\infty e^{-\eta(\tau - t)} \times ((x(\tau) - x_d(\tau))^T Q(x(\tau) - x_d(\tau)) + u^T(\tau) R u(\tau)) d\tau$$

where $Q \succeq 0$ and $R = R^T \succ 0$. The value function in terms of the states of the augmented system yields

$$V(X(t)) = \int_t^\infty e^{-\eta(\tau - t)} r(X, u) d\tau = (X^T(\tau) Q_T X(\tau) + u^T(\tau) R u(\tau)) d\tau \quad (35)$$

where

$$Q_T = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}$$

and $\eta \geq 0$ is the discount factor.

A differential equivalent to this is the Bellman equation

$$r(X, u) - \eta V + \frac{\partial V}{\partial X}^T (F(x) + G(X)u) = 0, \quad V(0) = 0$$

and the Hamiltonian is given by

$$H\left(X, u, \frac{\partial V}{\partial X}^T\right) = r(X, u) - \eta V + \frac{\partial V}{\partial X}^T (F(X) + G(X)u).$$

The optimal value is given by

$$r(X, u^\star) - \eta V^\star + \frac{\partial V^\star}{\partial X}^T (F(X) + G(X)u^\star) = 0$$

which is just the CT HJB tracking equation. The optimal control is then given as

$$u^\star(t) = \arg\min_u \left(r(X, u) - \eta V^\star + \frac{\partial V^\star}{\partial X}^T (F(X) + G(X)u)\right) = -\frac{1}{2} R^{-1} G^T(x) \left(\frac{\partial V^\star}{\partial X}\right).$$

D. Approximate Solution Using RL

Using the value function (35) and in a similar manner to the off-policy and on-policy IRL algorithms for the optimal regulation, the following off-policy and on-policy IRL algorithms are developed to find the optimal solution of CT HJB tracking equation.

1) On-Policy IRL: Similar to Algorithm 6 for the optimal regulation, the following on-policy IRL algorithm is presented in [83] to solve the CT HJB equation online and without requiring the knowledge of the internal system dynamics.

Algorithm 9 On-Policy IRL Algorithm to Find the Solution of HJB

1: **procedure**
2: Given admissible policy $u_{(0)}$
3: for $j = 0, 1, \ldots$ given u_j, solve for the value $V_{j+1}(X)$ using Bellman equation

$$V_{j+1}(X(t)) = \int_t^{t+T} e^{-\eta\tau}\left(X^T(\tau)Q_T X(\tau) + u_j^T(\tau)Ru_j(\tau)\right)d\tau + e^{-\eta T}V_{j+1}(X(t+T)),$$

on convergence, set $V_{j+1}(X) = V_j(X)$.
4: Update the control policy $u_{j+1}(t)$ using

$$u_{j+1}(t) = -\frac{1}{2}R^{-1}g^T(X)\left(\frac{\partial V_{j+1}(X)}{\partial X}\right).$$

5: Go to 3.
6: **end procedure**

The IRL algorithm (Algorithm 9) is an online algorithm and does not require the knowledge of the internal system dynamics. In [83], an actor-critic structure, similar to (6) and (7), is used to implement Algorithm 9. In [82], the quadratic form of value function, $x^T(t)Px(t)$, is used in the IRL Bellman equation for the linear systems to find the solution of LQT ARE.

2) Off-Policy IRL: Similar to off-policy IRL for optimal regulation, for the augmented system (34) and the discounted value function (35), the following off-policy IRL algorithm is developed in [84] to avoid the requirement of the knowledge of the system dynamics.

Algorithm 10 Off-Policy IRL Algorithm to Find the Solution of HJB

1: **procedure**
2: Given admissible policy u_0
3: for $j = 0, 1, \ldots$ given u_j, solve for the value V_{j+1} and u_{j+1} using off-policy Bellman equation

$$e^{-\eta T}V_{j+1}(X(t+T)) - V_{j+1}(X(t)) = \int_t^{t+T} e^{-\eta\tau}\left(-Q(X) - u_j^T Ru_j - 2u_{j+1}{}^T R(u-u_j)\right)d\tau,$$

on convergence, set $V_{j+1} = V_j$.
4: Go to 3.
5: **end procedure**

To implement off-policy IRL algorithm (Algorithm 10), the actor-critic structure, similar to (6) and (7), is used to approximate the value function and control policy [84].

Remark 5: Most of the existing solutions to optimal control problems are presented for affine systems. Extensions of RL algorithms for nonaffine systems are considered in [85] for optimal regulation and in [86] for optimal tracking control problems. □

E. $\mathcal{H}_\infty$ Control of CT Systems

The CT system dynamics in the presence of disturbance input is considered as

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) + h(x(t))d(t) \tag{36}$$

where $x(t) \in \mathbb{R}^n$ is a measurable state vector, $u(t) \in \mathbb{R}^m$ is the control input, $d(t) \in \mathbb{R}^q$ is the disturbance input, $f(x(t)) \in \mathbb{R}^n$ is the drift dynamics, $g(x(t)) \in \mathbb{R}^{n\times m}$ is the input dynamics, and $h(x(t)) \in \mathbb{R}^{n\times q}$ is the disturbance input dynamics. The performance index to be optimized is defined as

$$J(x(0), u, d) = \int_t^\infty (Q(x) + u^T Ru - \beta^2 d^T d)d\tau$$

where $Q(x) \succeq 0$, $R = R^T \succ 0$, and $\beta \geq \beta^\star \geq 0$ with $\beta^\star$ the smallest β such that the system is stabilized. The value function can be defined as

$$V(x(t), u, d) = \int_t^\infty (Q(x) + u^T Ru - \beta^2 d^T d)d\tau.$$

A differential equivalent to this is the Bellman equation

$$Q(x) + u^T Ru - \beta^2 d^T d + \frac{\partial V}{\partial x}^T (f(x) + g(x)u + h(x)d) = 0 \quad V(0) = 0$$

and the Hamiltonian is given by

$$H\left(x, u, d, \frac{\partial V}{\partial x}^T\right) = Q(x) + u^T Ru - \beta^2 d^T d + \frac{\partial V}{\partial x}^T (f(x) + g(x)u + h(x)d).$$

Define the two-player zero-sum differential game as

$$V^\star(x(t)) = \min_u \max_d J(x(0), u, d)$$

subject to (36).

In order to solve this zero-sum game, one needs to to solve the following HJI equation:

$$Q(x) + (u^\star)^T Ru^\star - \beta^2 (d^\star)^T d^\star + \frac{\partial V^\star}{\partial x}^T (f(x) + g(x)u^\star + h(x)d^\star) = 0.$$

Given a solution to this equation, one has

$$u^\star = -\frac{1}{2}R^{-1}g^T(x)\frac{\partial V^\star}{\partial x}$$
$$d^\star = \frac{1}{2\beta^2}h^T(x)\frac{\partial V^\star}{\partial x}.$$

F. Approximate Solution Using RL

The following PI algorithms can be used to approximate the HJI solution by iterating on the Bellman equation.

1) On-Policy IRL: In [87], an equivalent formulation of the Bellman equation that does not involve the dynamics is found to be

$$V(x(t-T)) = \int_{t-T}^{t} (r(x(\tau), u(\tau), d(\tau))d\tau + V(x(t))$$

for any time $t \geq 0$ and time interval $T > 0$. This equation is called the IRL Bellman equation.

Algorithm 11 On-Policy IRL for Regulation in Zero-Sum Games ($\mathcal{H}_\infty$ Control)

1: **procedure**
2: Given admissible policy u_0
3: for $j = 0, 1, \ldots$ given u_j
4: for $i = 0, 1, \ldots$ set $d^0 = 0$, solve for the value $V_j^{(i)}(x)$ using Bellman's equation

$$Q(x) + \left(\frac{\partial V_j^i}{\partial x}\right)^T \left(f(x) + g(x)u_j + h(x)d^i\right) + u_j^T R u_j - \beta^2 (d^i)^T d^i = 0, V_j^i(0) = 0,$$

$$d^{i+1} = \frac{1}{2\beta^2} h^T(x)\left(\frac{\partial V_j^i}{\partial x}\right),$$

on convergence, set $V_{j+1}(x) = V_j^i(x)$.
5: Update the control policy u_{j+1} using

$$u_{j+1} = -\frac{1}{2} R^{-1} g^T(x)\left(\frac{\partial V_{j+1}}{\partial x}\right).$$

6: Go to 3.
7: **end procedure**

The PI algorithm (Algorithm 11) is an offline algorithm and requires complete knowledge of the system dynamics. Actor-critic structure as (6) and (7) can be used to implement an online PI algorithm that simultaneously updates the value function and policies and does not require the knowledge of the drift dynamics. Various update laws are developed for learning actor, critic, and disturbance approximators' weights by minimizing the Bellman error [88]–[90].

2) Off-Policy IRL [80]: In order to develop the off-policy IRL algorithm for the $\mathcal{H}_\infty$ control, the system dynamics (36) is rewritten as

$$\dot{x}(t) = f(x(t)) + g(x(t))u_j(t) + g(x(t))(u(t) - u_j(t)) + h(x(t))d_j(t) + g(x(t))(d(t) - d_j(t))$$

where $u_j(t)$ and $d_j(t)$ are the policies to be updated. By contrast, $u(t)$ and $d(t)$ are the behavior policies that are actually applied to the system dynamics to generate the data for learning.

Differentiating $V(x)$ along with the system dynamics (28) and using $u_{j+1}(t) = -(1/2)R^{-1}g^T(x)(\partial V_j(x)/\partial x)$ and $d_{j+1}(t) = (1/2\beta^2)h^T(x)(\partial V_j(x)/\partial x)$ give

$$\dot{V}_j = \left(\frac{\partial V_j(x)}{\partial x}\right)^T (f + gu_j + hd_j) + \left(\frac{\partial V_j(x)}{\partial x}\right)^T g(u - u_j) + \left(\frac{\partial V_j(x)}{\partial x}\right)^T h(d - d_j) = -Q(x) - {u_j}^T R u_j + \beta^2 {d_j}^T d_j - 2{u_{j+1}}^T R(u - u_j) + 2\beta^2 {d_{j+1}}^T (d - d_j).$$

Integrating from both sides of the above equation yields the $\mathcal{H}_\infty$ off-policy IRL Bellman equation

$$V_j(x(t+T)) - V_j(x(t)) = \int_t^{t+T} \big(-Q(x) - {u_j}^T R u_j + \beta^2 {d_j}^T d_j - 2u_{j+1}^T R(u - u_j) + 2\beta^2 d_{j+1}^T (d - d_j)\big)d\tau. \quad (37)$$

Iterating on the IRL Bellman equation (37) yields the following off-policy IRL algorithm.

Algorithm 12 Off-Policy IRL Algorithm to Find the Solution of HJI

1: **procedure**
2: Given admissible policy u_0
3: for $j = 0, 1, \ldots$ given u_j and d_j, solve for the value V_j, u_{j+1} and d_{j+1} using off-policy Bellman equation

$$V_j(x(t+T)) - V_j(x(t)) = \int_t^{t+T} \big(-Q(x) - {u_j}^T R\, u_j + \beta^2\, {d_j}^T d_j - 2\, {u_{j+1}}^T R\,(u - u_j) + 2\,\beta^2\, {d_{j+1}}^T\,(d - d_j)\big)d\tau,$$

on convergence, set $V_{j+1} = V_j$.
4: Go to 3.
5: **end procedure**

Note that for a fixed control policy $u(t)$ and the actual disturbance (the policies that are applied to the system), the off-policy IRL Bellman equation (37) can be solved for value function V_j, learned control policy u_{j+1}, and learned disturbance policy d_{j+1} simultaneously, and without requiring any knowledge about the system dynamics.

To implement the off-policy IRL algorithm (Algorithm 12), the actor-critic structure, similar to (6) and (7), is used to approximate the value function, control, and disturbance policies [80].

In [88], an off-policy IRL algorithm is presented to find the solution of optimal tracking control problem without requiring any knowledge about the system dynamics and reference trajectory dynamics.

G. Nash Games

In this section, online RL-based solutions to noncooperative games, played among N agents, are considered.

Consider the N-player nonlinear time-invariant differential game

$$\dot{x}(t) = f(x(t)) + \sum_{j=1}^{N} g_j(x(t))u_j(t)$$

where $x \in \mathbb{R}^n$ is a measurable state vector, $u_j(t) \in \mathbb{R}^{m_j}$ are the control inputs, $f(x) \in \mathbb{R}^n$ is the drift dynamics, and $g_j(x) \in \mathbb{R}^{n\times m_j}$ is the input dynamics. It is assumed that $f(0) = 0$ and $f(x)+\sum_{j=1}^{N} g_j(x)u_j$ is locally Lipschitz and that the system is stabilizable.

The cost functional associated with each player is defined as

$$J_i(x(0), u_1, u_2, \ldots, u_N) = \int_0^\infty (r_i(x, u_1, \ldots, u_N))dt \equiv \int_0^\infty \left(Q_i(x) + \sum_{j=1}^{N} u_j^T R_{ij} u_j \right) dt \quad \forall i \in \mathcal{N} := \{1, 2, \ldots, N\}$$

where $Q_i(\cdot) \succeq 0$, $R_{ii} = R_{ii}^T \succ 0, \forall i \in \mathcal{N}$, and $R_{ij} = R_{ij}^T \succeq 0$, $\forall j \neq i \in \mathcal{N}$ with $\mathcal{N} := \{1, 2, \ldots, N\}$.

The value for each player can be defined as

$$V_i(x, u_1, u_2, \ldots, u_N) = \int_t^\infty (r_i(x, u_1, \ldots, u_N))d\tau, \quad \forall i \in \mathcal{N}, \ \forall x, u_1, u_2, \ldots, u_N.$$

Differential equivalents to each value function are given by the following Bellman equations:

$$r_i(x, u_1, \ldots, u_N) + \frac{\partial V_i}{\partial x}^T \left(f(x) + \sum_{j=1}^{N} g_j(x)u_j \right) = 0$$
$$V_i(0) = 0, \quad \forall i \in \mathcal{N}.$$

The Hamiltonian functions are defined as

$$H_i\left(x, u_1, \ldots, u_N, \frac{\partial V_i}{\partial x}^T\right) = r_i(x, u_1, \ldots, u_N) + \frac{\partial V_i}{\partial x}^T \left(f(x) + \sum_{j=1}^{N} g_j(x)u_j \right), \quad \forall i \in \mathcal{N}.$$

By applying the stationarity conditions, the associated feedback control policies are given by

$$u_i^\star = \arg\min_{u_i} H_i\left(x, u_1, \ldots, u_N, \frac{\partial V_i^\star}{\partial x}^T\right) = -\frac{1}{2} R_{ii}^{-1} g_i^T(x) \frac{\partial V_i}{\partial x}, \quad \forall i \in \mathcal{N}.$$

After substituting the feedback control policies into the Hamiltonian, one has the coupled Hamilton–Jacobi (HJ) equations

$$0 = Q_i(x) + \frac{1}{4}\sum_{j=1}^{N} \frac{\partial V_j}{\partial x}^T g_j(x) R_{jj}^{-T} R_{ij} R_{jj}^{-1} g_j^T(x) \frac{\partial V_j}{\partial x} + \frac{\partial V_i}{\partial x}^T \left(f(x) - \frac{1}{2}\sum_{j=1}^{N} g_j(x) R_{jj}^{-1} g_j^T(x) \frac{\partial V_j}{\partial x} \right), \quad \forall i \in \mathcal{N}. \tag{38}$$

H. Approximate Solution Using RL

A PI algorithm is now developed by iterating on the Bellman equation to approximate the solution to the coupled HJ equations.

Algorithm 13 PI for Regulation in Nonzero-Sum Games

1: **procedure**
2: Given N-tuple of admissible policies $\mu_i^k(0), \ \forall i \in \mathcal{N}$
3: **while** $\|V_i^{\mu^{(k)}} - V_i^{\mu^{(k-1)}}\| \geq \epsilon_{\text{iac}}, \forall i \in \mathcal{N}$ **do**
4: Solve for the N-tuple of costs $V_i^k(x)$ using the coupled Bellman equations

$$Q_i(x) + \frac{\partial V_i^k}{\partial x}^T \left(f(x) + \sum_{i=1}^{N} g_i(x)\mu_i^k\right) + \mu_i^{k\,T} R_{ii} \mu_i^k + \sum_{j=1}^{N} \mu_j^{k\,T} R_{ij} \mu_j^k = 0, \ V^{\mu_i^k}(0) = 0.$$

5: Update the N-tuple of control policies $\mu_i^{k+1}, \ \forall i \in \mathcal{N}$ using

$$\mu_i^{k+1} = -\frac{1}{2} R_{ii}^{-1} g_i^T(x) \frac{\partial V_i^k}{\partial x}^T.$$

6: $k = k + 1$
7: **end while**
8: **end procedure**

1) On-Policy IRL: It is obvious that (38) requires the complete knowledge of the system dynamics. In [87], an equivalent formulation of the coupled IRL Bellman equation that does not involve the dynamics is given as

$$V_i(x(t-T)) = \int_{t-T}^{t} r_i(x(\tau), u_1(\tau), \ldots, u_N(\tau))d\tau + V_i(x(t)), \quad \forall i \in \mathcal{N}$$

for any time $t \geq 0$ and time interval $T > 0$.

Let the value functions $V_i^\star$ be approximated on a compact set Ω as

$$V_i^\star(x) = W_i^T \phi_i(x) + \epsilon_i(x), \quad \forall x, \ \forall i \in \mathcal{N}$$

where $\phi_i(x) : \mathbb{R}^n \to \mathbb{R}^{K_i}, \ \forall i \in \mathcal{N}$, are the basis function basis set vectors, $K_i, \ \forall i \in \mathcal{N}$, is the number of basis functions, and $\epsilon_i(x)$ is the approximation error.

Assuming current weight estimates $\hat{W}_{ic}, \forall i \in \mathcal{N}$, the outputs of the critic are given by

$$\hat{V}_i(x) = \hat{W}_{ic}^T \phi_i(x), \quad \forall x, \ i \in \mathcal{N}.$$

Using a similar procedure as used for zero-sum games, the update laws can be rewritten as

$$\dot{W}_{ic} = -\alpha_i \frac{\Delta\phi_i(t)}{(\Delta\phi_i(t))^T \Delta\phi_i(t) + 1)^2} \times \left(\Delta\phi_i(t)^T \hat{W}_{ic} + \int_{t-T}^{t} \left(Q_i(x) + \hat{u}_i^T R_{ii} \hat{u}_i + \sum_{j=1}^{N} \hat{u}_j^T R_{ij} \hat{u}_j \right) d\tau \right), \quad \forall i \in \mathcal{N}$$

and

$$\dot{\hat{W}}_{iu} = -\alpha_{iu}\Bigg((F_i\hat{W}_{iu} - L_i\Delta\phi_i(t)^T\hat{W}_{ic}) - \frac{1}{4}\sum_{j=1}^{N}\left(\frac{\partial\phi_i}{\partial x}g_i(x)R_{ii}^{-T}R_{ij}R_{ii}^{-1}g_i(x)^T\frac{\partial\phi_i}{\partial x}^T\right)\hat{W}_{iu}\frac{\Delta\phi_i(t)^T}{(\Delta\phi_i(t)^T\Delta\phi_i(t)+1)^2}\hat{W}_{jc}\Bigg), \quad \forall i \in \mathcal{N}$$

respectively, with $\Delta\phi_i(t) := \phi_i(t) - \phi_i(t-T)$.

In [91], a model-free solution to nonzero-sum Nash games is presented. To obviate the requirement of complete knowledge of the system dynamics, system identification is used in [92].

IV. Graphical Games

Interactions among agents are modeled by a fixed strongly connected *graph* $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ defined by a finite set $\mathcal{V}_\mathcal{G} = \{n_1, \ldots, n_N\}$ of N agents and a set of edges $\mathcal{E}_\mathcal{G} \subseteq \mathcal{V}_\mathcal{G} \times \mathcal{V}_\mathcal{G}$ that represent interagent information exchange links. The set of *neighbors* of a node n_i is defined as the set of nodes with edges incoming to n_i and is denoted by $\mathcal{N}_i = \{n_j : (n_j, n_i) \in \mathcal{E}_\mathcal{G}\}$. The adjacency matrix is $\mathcal{A}_\mathcal{G} = [\alpha_{ij}]_{N\times N}$ with weights $\alpha_{ij} > 0$ if $(n_j, n_i) \in \mathcal{E}_\mathcal{G}$ and zero otherwise. The diagonal *degree* matrix D of the graph $\mathcal{G}$ is defined as $D = \text{diag}(d_i)$ with the weighted degree $d_i = \sum_{j\in\mathcal{N}_i}\alpha_{ij}$.

Let the agents dynamics be modeled as

$$\dot{x}_i(t) = Ax_i(t) + B_iu_i(t), \quad x_i(0) = x_{i0}, \ t \geq 0$$

where $x_i(t) \in \mathbb{R}^n$ is a measurable state vector, $u_i(t) \in \mathbb{R}^{m_i}$, $i \in \mathcal{N} := \{1, \ldots, N\}$, is each control input (or player), and $A \in \mathbb{R}^{n\times n}$ and $B_i \in \mathbb{R}^{n\times m_i}$, $i \in \mathcal{N}$, are the plant and input matrices, respectively.

The leader node/exosystem dynamics are

$$\dot{x}_0 = Ax_0.$$

The agents in the network seek to cooperatively asymptotically track the state of a leader node/exosystem, i.e., $x_i(t) \to x_0(t), \forall i \in \mathcal{N}$ while *simultaneously satisfying distributed cost functional*.

Define the *neighborhood tracking error* for every agent as

$$\delta_i := \sum_{j\in\mathcal{N}_i}\alpha_{ij}(x_i - x_j) + g_i(x_i - x_0), \quad \forall i \in \mathcal{N} \tag{39}$$

where g_i is the pinning gain. $g_i \neq 0$ if agent is pinned to the leader node. The dynamics of (39) are given by

$$\dot{\delta}_i = A\delta_i + (d_i + g_i)B_iu_i - \sum_{j\in\mathcal{N}_i}\alpha_{ij}B_ju_j, \quad \forall i \in \mathcal{N} \tag{40}$$

with $\delta_i \in \mathbb{R}^n$.

The cost functional associated to each agent $i \in \mathcal{N}$ has the following form:

$$J_i(\delta_i(0); u_i, u_{\mathcal{N}_i}) = \frac{1}{2}\int_0^\infty r_i(\delta_i, u_i, u_{\mathcal{N}_i})dt, \quad \forall i \in \mathcal{N} \equiv \frac{1}{2}\int_0^\infty\left(\delta_i^TQ_i\delta_i + u_i^TR_{ii}u_i + \sum_{j\in\mathcal{N}_i}u_j^TR_{ij}u_j\right)dt, \quad \forall i \in \mathcal{N}$$

with user-defined matrices $Q_i \succeq 0$, $R_{ii} \succ 0$, and $R_{ij} \succeq 0$, $\forall i, j \in \mathcal{N}$, of appropriate dimensions and $(\sqrt{Q_i}, A)$, $\forall i \in \mathcal{N}$, are detectable.

It is desired to find a graphical Nash equilibrium [93] $u_i^\star$ for all agents $i \in \mathcal{N}$ in the sense that

$$J_i(\delta_i(0); u_i^\star, u_{\mathcal{N}_i}^\star) \leq J_i(\delta_i(0); u_i, u_{\mathcal{N}_i}^\star), \quad \forall u_i, \ i \in \mathcal{N}.$$

This can be expressed by the following coupled distributed minimization problems:

$$J_i(\delta_i(0); u_i^\star, u_{\mathcal{N}_i}^\star) = \min_{u_i} J_i(\delta_i(0); u_i, u_{\mathcal{N}_i}^\star), \quad \forall i \in \mathcal{N}$$

with the dynamics given in (40).

The ultimate goal is to find the distributed optimal value functions $V_i^\star$, $\forall i \in \mathcal{N}$ defined by

$$V_i^\star(\delta_i(t)) := \min_{u_i}\int_t^\infty\frac{1}{2}\left(\delta_i^TQ_i\delta_i + u_i^TR_{ii}u_i + \sum_{j\in\mathcal{N}_i}u_j^TR_{ij}u_j\right)dt, \quad \forall t, \delta_i, i \in \mathcal{N}. \tag{41}$$

One can define the Hamiltonians associated with each agent's neighborhood tracking error (40) and each $V_i^\star$ given in (41) as follows:

$$H_i\left(\delta_i, u_i, u_{\mathcal{N}_i}, \frac{\partial V_i^\star}{\partial\delta_i}\right) = \frac{\partial V_i^{\star T}}{\partial\delta_i}\left(A\delta_i + (d_i + g_i)B_iu_i - \sum_{j\in\mathcal{N}_i}\alpha_{ij}B_ju_j\right) + \frac{1}{2}\delta_i^TQ_i\delta_i + \frac{1}{2}u_i^TR_{ii}u_i + \frac{1}{2}\sum_{j\in\mathcal{N}_i}u_j^TR_{ij}u_j, \quad \forall\delta_i, u_i, \quad \forall i \in \mathcal{N}.$$

According to the stationarity condition, the optimal control for each $i \in \mathcal{N}$ can be found to be

$$u_i^\star(\delta_i) = \arg\min_{u_i}H_i\left(\delta_i, u_i, u_{\mathcal{N}_i}, \frac{\partial V_i^\star}{\partial\delta_i}\right) = -(d_i + g_i)R_{ii}^{-1}B_i^T\frac{\partial V_i^\star}{\partial\delta_i}, \quad \forall\delta_i \tag{42}$$

that should satisfy the appropriate distributed coupled HJ equations

$$H_i\left(\delta_i, u_i^\star, u_{\mathcal{N}_i}^\star, \frac{\partial V_i^\star}{\partial\delta_i}\right) = 0, \quad \forall i \in \mathcal{N}.$$

Assume that the value functions are quadratic in the neighborhood tracking error, i.e., $V_i^\star(\delta_i) : \mathbb{R}^n \to \mathbb{R}$

$$V_i^\star(\delta_i) = \frac{1}{2}\delta_i^T P_i \delta_i, \quad \forall \delta_i, \forall i \in \mathcal{N} \tag{43}$$

where $P_i = P_i^T > 0 \in \mathbb{R}^{n\times n}$, $\forall i \in \mathcal{N}$, are the unique matrices that solve the following complicated distributed coupled equations:

$$\begin{aligned}
&\delta_i^T P_i \left(A\delta_i - (d_i+g_i)^2 B_i R_{ii}^{-1} B_i^T P_i \delta_i \right.\\
&\qquad \left. + \sum_{j\in\mathcal{N}} \alpha_{ij}(d_j+g_j) B_j R_{jj}^{-1} B_j^T P_j \delta_j \right)\\
&+ \left(A\delta_i - (d_i+g_i)^2 B_i R_{ii}^{-1} B_i^T P_i \delta_i \right.\\
&\qquad \left. + \sum_{j\in\mathcal{N}} \alpha_{ij}(d_j+g_j) B_j R_{jj}^{-1} B_j^T P_j \delta_j \right)^T\\
&\times P_i\delta_i + \sum_{j\in\mathcal{N}_i} (d_j+g_j)^2 \delta_j^T P_j B_j R_{jj}^{-T} R_{ij} R_{jj}^{-1} B_j^T P_j \delta_j\\
&+ (d_i+g_i)^2 \delta_i^T P_i B_i R_{ii} B_i^T P_i \delta_i + \delta_i^T Q_i \delta_i = 0, \quad \forall i \in \mathcal{N}.
\end{aligned}$$

Using (43), the optimal control (42) for every player $i \in \mathcal{N}$ can be written as

$$u_i^\star(\delta_i) = -(d_i+g_i) R_{ii}^{-1} B_i^T P_i \delta_i, \quad \forall \delta_i.$$

A. Approximate Solution Using RL

A PI algorithm is now developed to solve distributed coupled HJ equations as follows.

Algorithm 14 PI for Regulation in Graphical Games

1: **procedure**
2: Given N-tuple of admissible policies $\mu_i^k(0)$, $\forall i \in \mathcal{N}$
3: **while** $\|V_i^{\mu^{(k)}}(\delta_i) - V_i^{\mu^{(k-1)}}(\delta_i)\| \geq \epsilon_{\text{iac}}$, $\forall i \in \mathcal{N}$ **do**
4: Solve for the N-tuple of costs $V_i^k(x)$ using the coupled Bellman equations,

$$\begin{aligned}
&\delta_i^T Q_{ii} \delta_i + \frac{\partial V_i^k}{\partial \delta_i}^T (A\delta_i + (d_i+g_i) B_i \mu_i^k - \sum_{j\in\mathcal{N}_i} \alpha_{ij} B_j \mu_j^k)\\
&+ \mu_i^{kT} R_{ii} \mu_i^k + \sum_{j\in\mathcal{N}_i} \mu_j^{kT} R_{ij} \mu_j^k = 0, V^{\mu_i^k}(0) = 0.
\end{aligned}$$

5: Update the N-tuple of control policies $\mu_i^{k+1}, \forall i \in \mathcal{N}$ using

$$\mu_i^{k+1} = -(d_i+g_i) R_{ii}^{-1} B_i^T \frac{\partial V_i^k}{\partial \delta_i}^T.$$

6: $k = k+1$
7: **end while**
8: **end procedure**

The PI algorithm (Algorithm 14) is an offline algorithm. We now show how to develop an online RL-based algorithm for solving the coupled HJ equations.

Assume there exist constant weights W_i and such that the value functions $V_i^\star$ are approximated on a compact set Ω as

$$V_i^\star(\delta_i) = W_i^T \phi_i(\delta_i) + \epsilon_i(\delta_i), \quad \forall \delta_i, i \in \mathcal{N}$$

where $\phi_i(\delta_i) : \mathbb{R}^n \to \mathbb{R}^{K_i}$, $\forall i \in \mathcal{N}$, are the basis function set vectors, K_i is the number of basis functions, and $\epsilon_i(\delta_i)$ is the approximation error.

Assuming current weight estimates $\hat{W}_{ic}, \forall i \in \mathcal{N}$, the outputs of the critic are given by

$$\hat{V}_i(\delta_i) = \hat{W}_{ic}^T \phi_i(\delta_i), \quad \forall i \in \mathcal{N}.$$

Using the current weight estimates, the approximate Lyapunov-like equations are given by

$$\begin{aligned}
&\hat{H}_i(\cdot) = r_i(\delta_i(t), \hat{u}_i, \hat{u}_{\mathcal{N}_i})\\
&\qquad + \frac{\partial V_i}{\partial \delta_i}^T \left(A\delta_i + (d_i+g_i) B_i \hat{u}_i - \sum_{j\mathcal{N}_i} \alpha_{ij} B_j \hat{u}_j \right)\\
&\qquad = e_i, \quad \forall i \in \mathcal{N}
\end{aligned} \tag{44}$$

where $e_i \in \mathbb{R}, \forall i \in \mathcal{N}$, are the residual errors due to approximation and

$$\hat{u}_i = -(d_i+g_i) R_{ii}^{-1} B_i^T \frac{\partial \phi_i}{\partial \delta_i}^T \hat{W}_{iu}, \quad \forall i \in \mathcal{N}$$

where $\hat{W}_{iu}$ denote the current estimated values of the ideal weights W_i.

The critic weights are updated to minimize the square residual errors e_i in order to guarantee that $\hat{W}_{ic} \to W_i$, $\forall i \in \mathcal{N}$. Hence, the tuning law for the critic is given as

$$\begin{aligned}
&\dot{W}_{ic}\\
&= -\alpha_i \frac{\frac{\partial \phi_i}{\partial \delta_i} \left(A\delta_i + (d_i+g_i) B_i \hat{u}_i - \sum_{j\in\mathcal{N}_i} \alpha_{ij} B_j \hat{u}_j \right)}{\Delta^2}\\
&\times \left(\left(\frac{\partial \phi_i}{\partial \delta_i} \left(A\delta_i + (d_i+g_i) B_i \hat{u}_i - \sum_{j\in\mathcal{N}_i} \alpha_{ij} B_j \hat{u}_j \right) \right)^T \right.\\
&\left. \times \hat{W}_{ic} + \frac{1}{2}\delta_i^T Q_i \delta_i + \hat{u}_i^T R_{ii} \hat{u}_i + \sum_{j\in\mathcal{N}_i} \hat{u}_j^T R_{ij} \hat{u} \right), \quad \forall i \in \mathcal{N}
\end{aligned}$$

where $\alpha_i > 0$ is a tuning gain, and for the actor is given as

$$\dot{\hat{W}}_{iu} = -\alpha_{iu}\left(F_i\hat{W}_{iu} - L_i\left(\frac{\partial\phi_i}{\partial\delta_i}\left(A\delta_i + (d_i+g_i)B_i\hat{u}_i - \sum_{j\in\mathcal{N}_i}\alpha_{ij}B_j\hat{u}_j\right)^T\hat{W}_{ic}\right)\right) - \frac{1}{4}\sum_{j\in\mathcal{N}_i}\left(\frac{\partial\phi_i}{\partial\delta_i}B_iR_{ii}^{-T}R_{ij}R_{ii}^{-1}B_j^T\frac{\partial\phi_i}{\partial\delta_i}^T\right)\hat{W}_{iu} \times\left(\frac{\frac{\partial\phi_i}{\partial\delta_i}\left(A\delta_i+(d_i+g_i)B_i\hat{u}_i-\sum_{j\in\mathcal{N}_i}\alpha_{ij}B_j\hat{u}_j\right)}{\Delta}\right)^T\hat{W}_{ic}\Bigg) \quad \forall i\in\mathcal{N}$$

where $\alpha_{iu} > 0$ is a tuning gain, and $F_i, L_i \succ 0$ are matrices that guarantee stability of the system and

$$\Delta = \left(\left(\frac{\partial\phi_i}{\partial\delta_i}\left(A\delta_i + (d_i+g_i)B_i\hat{u}_i - \sum_{j\in\mathcal{N}_i}\alpha_{ij}B_j\hat{u}_j\right)\right)^T \times\frac{\partial\phi_i}{\partial\delta_i}\left(A\delta_i+(d_i+g_i)B_i\hat{u}_i-\sum_{j\in\mathcal{N}_i}\alpha_{ij}B_j\hat{u}_j\right)+1\right)^2.$$

V. Applications

RL has been successfully applied in many fields. Two of such fields are presented below.

A. Robot Control and Navigation

In the last decade or so, the application of RL in robotics has increased steadily. In [94], RL is used to design a kinematic and dynamic tracking control scheme for a nonholonomic wheeled mobile robot. Reference [95] uses the RL in the network-based tracking control system of the two-wheeled mobile robot, Pioneer 2-DX. The neural network RL is used to design a controller for an active simulated three-link biped robot in [96]. In [97], an RL based actor-critic framework is used in control of a fully actuated six degrees-of-freedom autonomous underwater vehicle. Luy *et al.* [98] use RL to design robust adaptive tracking control laws with optimality for multiwheeled mobile robots' synchronization in communication graph. Reference [99] presents an optimal adaptive consensus-based formation control scheme over finite horizon for networked mobile robots or agents in the presence of uncertain robot/agent dynamics. In [100], IRL approach is employed to design a novel adaptive impedance control for robotic exoskeleton with adjustable robot bebavior. IRL is used in control of a surface marine craft with unknown hydrodynamic parameters in [101]. In [102], RL is used along with a stabilizing feedback controller such as PID or linear quadratic regulation to improve the performance of the trajectory tracking in robot manipulators.

A mobile robot navigation task refers to plan a path with obstacle avoidance to a specified goal. RL-based mobile robot navigation mainly includes robot learning from expert demonstrations and robot self-learning and autonomous navigation. In the former, the examples of standard behaviors are provided to the robot and it learns from these data and generalize over all potential situations that are not given in the examples [103]–[112]. In the later, RL techniques are used to train the robot via interaction with the surrounding environment. Stability is not a concern in this type of the task. Yang *et al.* [113] used continuous Q-learning for autonomous navigation of mobile robots. Lagoudakis and Parr [114] proposed the LS PI for robot navigation task. In this paper, and some other similar works [115]–[117], neural networks are used to approximate the value function and the environment is assumed static. An autonomous agent needs to adapt its strategies to cope with changing surroundings and solve challenging navigation tasks [118], [119]. The methods in [115] and [116] are designed for environments with dynamic obstacle.

B. Power Systems

There have existed several applications of RL in power systems. The work of [120] proposes the PI technique based on actor-critic structure for automatic voltage regulator system for its both models neglecting and including sensor dynamics. In [121], a game-theory-based distributed controller is designed to provide the desired voltage magnitude and frequency at the load. The optimal switching between different typologies in step-down dc–dc voltage converters is investigated in [122]. The control of boost converter is presented in [123] using RL-based nonlinear control strategy at attaining a constant output voltage. In [124], RL is used for distributed control of dc microgrids to establish coordination among active loads to collectively respond to any load change. In [125], RL is applied to the control of an electric water heater with 50 temperature sensors.

VI. Conclusion

In this paper, we have reviewed several RL techniques for solving optimal control problems in real time using data measured along the system trajectories. We have presented families of online RL-based solutions for optimal regulation, optimal tracking, Nash, and graphical games. The complete dynamics of the systems do not need to be known for these RL-based online solution techniques. As another approach to speed up the learning by reuse of data, experience replay technique used in RL was discussed. These algorithms are fast and data efficient because of reuse of the data for learning.

The design of static RL-based OPFB controllers for nonlinear systems has not been investigated yet. Also existing RL-based $\mathcal{H}_\infty$ controllers require the disturbance to be measured during learning. Another interesting and important open research direction is to develop novel deep RL approaches for feedback control of nonlinear systems with high-dimensional inputs and unstructured input data. Deep neural networks can approximate a more accurate structure of the value function and avoid divergence of the RL algorithm and consequently

instability of the feedback control system. Deep RL for feedback control, however, requires developing new learning algorithms that assure the stability of the feedback system in the sense of Lyapunov during the learning.

REFERENCES

[1] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*. New York, NY, USA: Wiley, 2012.

[2] R. F. Stengel, *Optimal Control and Estimation* (Dover Books on Mathematics). Mineola, NY, USA: Dover, 1986.

[3] D. Liberzon, *Calculus of Variations and Optimal Control Theory—A Concise Introduction*. Princeton, NJ, USA: Princeton Univ. Press, 2011.

[4] M. Athans and P. L. Falb, *Optimal Control: An Introduction to the Theory and Its Applications* (Dover Books on Engineering). Mineola, NY, USA: Dover, 2006.

[5] D. E. Kirk, *Optimal Control Theory: An Introduction* (Dover Books on Electrical Engineering). Mineola, NY, USA: Dover, 2012.

[6] A. E. Bryson, *Applied Optimal Control: Optimization, Estimation and Control*. New York, NY, USA: Halsted, 1975.

[7] W. M. McEneaney, *Max-Plus Methods for Nonlinear Control and Estimation*. Basel, Switzerland: Birkhäuser, 2006.

[8] W. H. Fleming and W. M. McEneaney, "A max-plus-based algorithm for a Hamilton–Jacobi–Bellman equation of nonlinear filtering," *SIAM J. Control Optim.*, vol. 38, no. 3, pp. 683–710, 2000.

[9] M. T. Hagan, H. B. Demuth, M. Beale, and O. De Jesús, *Neural Network Design*, 2nd ed. New York, NY, USA: Martin, 2014.

[10] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[11] C. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.

[12] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer-Verlag, 2009.

[13] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1, 2nd ed. Cambridge, MA, USA: MIT Press, 2017.

[15] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA, USA: Athena Scientific, 1996.

[16] W. B. Powell, *Approximate Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2007.

[17] X.-R. Cao, *Stochastic Learning and Optimization*. New York, NY, USA: Springer, 2007.

[18] H. Zhang, D. Liu, Y. Luo, and D. Wang, *Adaptive Dynamic Programming for Control*. London, U.K.: Springer-Verlag, 2013.

[19] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li, *Adaptive Dynamic Programming With Applications in Optimal Control*. Springer, 2017.

[20] M. Krstić and I. Kanellakopoulos, *Nonlinear and Adaptive Control Design* (Adaptive and Cognitive Dynamic Systems: Signal Processing, Learning, Communications and Control). New York, NY, USA: Wiley, 1995.

[21] G. Tao, *Adaptive Control Design and Analysis* (Adaptive and Cognitive Dynamic Systems: Signal Processing, Learning, Communications and Control). New York, NY, USA: Wiley, 2003.

[22] P. Ioannou and B. Fidan, *Adaptive Control Tutorial* (Advances in Design and Control). Philadelphia, PA, USA: SIAM, 2006.

[23] N. Hovakimyan and C. Cao, *L1 Adaptive Control Theory: Guaranteed Robustness With Fast Adaptation* (Advances in Design and Control). Philadelphia, PA, USA: SIAM, 2010.

[24] S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence and Robustness* (Dover Books on Electrical Engineering). Mineola, NY, USA: Dover, 2011.

[25] K. J. Aström and B. Wittenmark, *Adaptive Control* (Dover Books on Electrical Engineering), 2nd ed. Mineola, NY, USA: Dover, 2013.

[26] P. Ioannou and J. Sun, *Robust Adaptive Control*. Mineola, NY, USA: Dover, 2013.

[27] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. SMC-13, no. 5, pp. 834–846, Sep./Oct. 1983.

[28] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, 1988.

[29] P. J. Werbos, "Neural networks for control and system identification," in *Proc. IEEE 28st Annu. Conf. Decision Control (CDC)*, Dec. 1989, pp. 260–265.

[30] P. J. Werbos, *A Menu of Designs for Reinforcement Learning Over Time*. Cambridge, MA, USA: MIT Press, 1991.

[31] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, D. A. White and D. A. Sofge, Eds. New York, NY, USA: Van Nostrand, 1992.

[32] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge Univ., Cambridge, U.K., 1989.

[33] K. Doya, "Reinforcement learning in continuous time and space," *Neural Comput.*, vol. 12, no. 1, pp. 219–245, 2000.

[34] M. Abu-Khalaf and F. L. Lewis, "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach," *Automatica*, vol. 41, no. 5, pp. 779–791, May 2005.

[35] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, no. 2, pp. 477–484, Feb. 2009.

[36] K. G. Vamvoudakis and F. L. Lewis, "Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[37] A. Heydari and S. N. Balakrishnan, "Optimal switching and control of nonlinear switching systems using approximate dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 6, pp. 1106–1117, Jun. 2014.

[38] A. Heydari and S. N. Balakrishnan, "Optimal switching between autonomous subsystems," *J. Franklin Inst.*, vol. 351, no. 5, pp. 2675–2690, 2014.

[39] A. Heydari, "Optimal scheduling for reference tracking or state regulation using reinforcement learning," *J. Franklin Inst.*, vol. 352, no. 8, pp. 3285–3303, 2015.

[40] A. Heydari and S. N. Balakrishnan, "Optimal switching between controlled subsystems with free mode sequence," *Neurocomputing*, vol. 149, pp. 1620–1630, Feb. 2015.

[41] A. Heydari, "Feedback solution to optimal switching problems with switching cost," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 10, pp. 2009–2019, Oct. 2016.

[42] R. J. Leake and R.-W. Liu, "Construction of suboptimal control sequences," *SIAM J. Control*, vol. 5, no. 1, pp. 54–63, Feb. 1967.

[43] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA, USA: MIT Press, 1960.

[44] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das, *Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches*. New York, NY, USA: Springer-Verlag, 2014.

[45] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 943–949, Aug. 2008.

[46] P. He and S. Jagannathan, "Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 2, pp. 425–436, Apr. 2007.

[47] T. Dierks and S. Jagannathan, "Online optimal control of nonlinear discrete-time systems using approximate dynamic programming," *J. Control Theory Appl.*, vol. 9, no. 3, pp. 361–369, 2011.

[48] Y. Luo and H. Zhang, "Approximate optimal control for a class of nonlinear discrete-time systems with saturating actuators," *Prog. Natural Sci.*, vol. 18, no. 8, pp. 1023–1029, 2008.

[49] Y. Sokolov, R. Kozma, L. D. Werbos, and P. J. Werbos, "Complete stability analysis of a heuristic approximate dynamic programming control design," *Automatica*, vol. 59, pp. 9–18, Sep. 2015.

[50] A. Heydari, "Theoretical and numerical analysis of approximate dynamic programming with approximation errors," *J. Guid., Control, Dyn.*, vol. 39, no. 2, pp. 301–311, 2016.

[51] L. Dong, X. Zhong, C. Sun, and H. He, "Adaptive event-triggered control based on heuristic dynamic programming for nonlinear discrete-time systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 7, pp. 1594–1605, Jul. 2016.

[52] A. Sahoo, H. Xu, and S. Jagannathan, "Near optimal event-triggered control of nonlinear discrete-time systems using neurodynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 9, pp. 1801–1815, Sep. 2016.

[53] S. J. Bradtke, B. E. Ydstie, and A. G. Barto, "Adaptive linear quadratic control using policy iteration," in *Proc. IEEE Amer. Control Conf.*, vol. 3. Jul. 1994, pp. 3475–3479.

[54] C. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, 1992.

[55] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 32–50, Aug. 2009.

[56] F. L. Lewis and K. G. Vamvoudakis, "Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 1, pp. 14–25, Feb. 2011.

[57] B. Kiumarsi and F. L. Lewis, "Actor–critic-based optimal tracking for partially unknown nonlinear discrete-time systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 140–151, Jan. 2015.

[58] B. Kiumarsi, F. L. Lewis, H. Modares, A. Karimpour, and M. B. Naghibi-Sistani, "Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics," *Automatica*, vol. 50, no. 4, pp. 1167–1175, Apr. 2014.

[59] R. Bellman, "Dynamic programming and Lagrange multipliers," *Proc. Nat. Acad. Sci. USA*, vol. 42, no. 10, pp. 767–769, 1956.

[60] B. Kiumarsi, F. L. Lewis, M. B. Naghibi-Sistani, and A. Karimpour, "Optimal tracking control of unknown discrete-time linear systems using input-output measured data," *IEEE Trans. Cybern.*, vol. 45, no. 12, pp. 2770–2779, Dec. 2015.

[61] G. Zames, "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses," *IEEE Trans. Autom. Control*, vol. AC-26, no. 2, pp. 301–320, Apr. 1981.

[62] A. J. van der Schaft, "L_2-gain analysis of nonlinear systems and nonlinear state-feedback H_∞ control," *IEEE Trans. Autom. Control*, vol. 37, no. 6, pp. 770–784, Jun. 1992.

[63] T. Başar and P. Bernhard, *H_∞ Optimal Control and Related Minimax Design Problems*. Boston, MA, USA: Birkhäuser, 1995.

[64] J. C. Doyle, K. Glover, P. P. Khargonekar, and B. A. Francis, "State-space solutions to standard H_2 and H_∞ control problems," *IEEE Trans. Autom. Control*, vol. 34, no. 8, pp. 831–847, Aug. 1989.

[65] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Model-free Q-learning designs for linear discrete-time zero-sum games with application to H_∞ control," *Automatica*, vol. 43, no. 3, pp. 473–481, 2007.

[66] B. Kiumarsi, F. L. Lewis, and Z.-P. Jiang, "H_∞ control of linear discrete-time systems: Off-policy reinforcement learning," *Automatica*, vol. 78, pp. 144–152, Apr. 2017.

[67] D. Vrabie and F. Lewis, "Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems," *Neural Netw.*, vol. 22, no. 3, pp. 237–246, 2009.

[68] S. Bhasin, R. Kamalapurkar, M. Johnson, K. G. Vamvoudakis, F. L. Lewis, and E. W. Dixon, "A novel actor-critic-identifier architecture for approximate optimal control of uncertain nonlinear systems," *Automatica*, vol. 49, no. 1, pp. 89–92, 2013.

[69] T. Dierks and S. Jagannathan, "Optimal control of affine nonlinear continuous-time systems," in *Proc. Amer. Control Conf.*, Jun./Jul. 2010, pp. 1568–1573.

[70] R. Kamalapurkar, J. R. Klotz, and W. E. Dixon, "Concurrent learning-based approximate feedback-Nash equilibrium solution of N-player nonzero-sum differential games," *IEEE/CAA J. Autom. Sin.*, vol. 1, no. 3, pp. 239–247, Jul. 2014.

[71] R. Kamalapurkar, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for approximate optimal regulation," *Automatica*, vol. 64, p. 94–104, Feb. 2016.

[72] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems," *Automatica*, vol. 50, no. 1, pp. 193–202, 2014.

[73] G. Chowdhary, T. Yucelen, M. Mühlegg, and E. N. Johnson, "Concurrent learning adaptive control of linear systems with exponentially convergent bounds," *Int. J. Adapt. Control Signal Process.*, vol. 27, no. 4, pp. 280–301, 2013.

[74] K. G. Vamvoudakis, "Event-triggered optimal adaptive control algorithm for continuous-time nonlinear systems," *IEEE/CAA J. Autom. Sinica*, vol. 1, no. 3, pp. 282–293, Jul. 2014.

[75] L. Dong, X. Zhong, C. Sun, and H. He, "Event-triggered adaptive dynamic programming for continuous-time systems with control constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 8, pp. 1941–1952, Aug. 2017.

[76] V. Narayanan and S. Jagannathan, "Distributed adaptive optimal regulation of uncertain large-scale interconnected systems using hybrid Q-learning approach," *IET Control Theory Appl.*, vol. 10, no. 12, pp. 1448–1457, 2016.

[77] V. Narayanan and S. Jagannathan, "Approximate optimal distributed control of uncertain nonlinear interconnected systems with event-sampled feedback," in *Proc. IEEE 55th Conf. Decision Control (CDC)*, Dec. 2016, pp. 5827–5832.

[78] V. Narayanan and S. Jagannathan, "Distributed event-sampled approximate optimal control of interconnected affine nonlinear continuous-time systems," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2016, pp. 3044–3049.

[79] Y. Jiang and Z.-P. Jiang, "Robust adaptive dynamic programming and feedback stabilization of nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 882–893, May 2014.

[80] B. Luo, H.-N. Wu, and T. Huang, "Off-policy reinforcement learning for H_∞ control design," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 65–76, Jan. 2015.

[81] Y. Jiang and Z.-P. Jiang, "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics," *Automatica*, vol. 48, no. 10, pp. 2699–2704, 2012.

[82] H. Modares and F. L. Lewis, "Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning," *IEEE Trans. Autom. Control*, vol. 59, no. 11, pp. 3051–3056, Nov. 2014.

[83] H. Modares and F. L. Lewis, "Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning," *Automatica*, vol. 50, no. 7, pp. 1780–1792, Jul. 2014.

[84] H. Modares, "Optimal tracking control of uncertain systems: On-policy and off-policy reinforcement learning approaches," Ph.D. dissertation, Dept. Electr. Eng., Univ. Texas Arlington, Arlington, TX, USA, 2015.

[85] T. Bian, Y. Jiang, and Z.-P. Jiang, "Adaptive dynamic programming and optimal control of nonlinear nonaffine systems," *Automatica*, vol. 50, no. 10, pp. 2624–2632, Oct. 2014.

[86] B. Kiumarsi, W. Kang, and F. L. Lewis, "H_∞ control of nonaffine aerial systems using off-policy reinforcement learning," *Unmanned Syst.*, vol. 4, no. 1, pp. 51–60, 2016.

[87] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis, *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles* (Control, Robotics and Sensors). Edison, NJ, USA: IET, 2013.

[88] H. Modares, F. L. Lewis, and Z.-P. Jiang, "H_∞ tracking control of completely unknown continuous-time systems via off-policy reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2550–2562, Oct. 2015.

[89] H. Li, D. Liu, and D. Wang, "Integral reinforcement learning for linear continuous-time zero-sum games with completely unknown dynamics," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 3, pp. 706–714, Jul. 2014.

[90] B. Luo and H.-N. Wu, "Computationally efficient simultaneous policy update algorithm for nonlinear H_∞ state feedback control with Galerkin's method," *Int. J. Robust Nonlinear Control*, vol. 23, no. 9, pp. 991–1012, 2013.

[91] K. G. Vamvoudakis, "Non-zero sum nash Q-learning for unknown deterministic continuous-time linear systems," *Automatica*, vol. 61, pp. 274–281, Nov. 2015.

[92] M. Johnson, R. Kamalapurkar, S. Bhasin, and W. E. Dixon, "Approximate N-player nonzero-sum game solution for an uncertain continuous nonlinear system," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 8, pp. 1645–1658, Aug. 2015.

[93] K. G. Vamvoudakis, F. L. Lewis, and G. R. Hudas, "Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality," *Automatica*, vol. 48, no. 8, pp. 1598–1611, 2012.

[94] N. T. Luy, "Reinforecement learning-based optimal tracking control for wheeled mobile robot," in *Proc. IEEE Int. Conf. Cyber Technol. Autom., Control, Intell. Syst. (CYBER)*, May 2012, pp. 371–376.

[95] M. Szuster and Z. Hendzel, "Discrete globalised dual heuristic dynamic programming in control of the two-wheeled mobile robot," *Math. Problems Eng.*, vol. 2014, Sep. 2014, Art. no. 628798.

[96] A. Ghanbari, Y. Vaghei, and S. M. R. S. Noorani, "Neural network reinforcement learning for walking control of a 3-link biped robot," *Int. J. Eng. Technol.*, vol. 7, no. 6, pp. 449–452, 2015.

[97] P. Walters, R. Kamalapurkar, and W. E. Dixon. (Sep. 2013). "Online approximate optimal station keeping of an autonomous underwater vehicle." [Online]. Available: https://arxiv.org/abs/1310.0063

[98] N. T. Luy, N. T. Thanh, and H. M. Tri, "Reinforcement learning-based robust adaptive tracking control for multi-wheeled mobile robots synchronization with optimality," in *Proc. IEEE Workshop Robot. Intell. Inform. Struct. Space (RiiSS)*, Apr. 2013, pp. 74–81.

[99] H. M. Guzey, H. Xu, and J. Sarangapani, "Neural network-based finite horizon optimal adaptive consensus control of mobile robot formations," *Optim. Control Appl. Methods*, vol. 37, no. 5, pp. 1014–1034, 2016.
[100] Z. Huang, J. Liu, Z. Li, and C.-Y. Su, "Adaptive impedance control of robotic exoskeletons using reinforcement learning," in *Proc. Int. Conf. Adv. Robot. Mechatron. (ICARM)*, 2016, pp. 243–248.
[101] Z. Bell, A. Parikh, J. Nezvadovitz, and W. E. Dixon, "Adaptive control of a surface marine craft with parameter identification using integral concurrent learning," in *Proc. IEEE 55th Conf. Decision Control (CDC)*, Dec. 2016, pp. 389–394.
[102] Y. P. Pane, S. P. Nageshrao, and R. Babuška, "Actor-critic reinforcement learning for tracking control in robotics," in *Proc. IEEE 55th Conf. Decision Control (CDC)*, Dec. 2016, pp. 5819–5826.
[103] M. van Lent and J. E. Laird, "Learning procedural knowledge through observation," in *Proc. ACM 1st Int. Conf. Knowl. Capture (K-CAP)*, 2001, pp. 179–186.
[104] A. Jain, B. Wojcik, T. Joachims, and A. Saxena, "Learning trajectory preferences for manipulators via iterative improvement," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 575–583.
[105] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn, "Imitation with ALICE: Learning to imitate corresponding actions across dissimilar embodiments," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 32, no. 4, pp. 482–496, Jul. 2002.
[106] B. Robins, K. Dautenhahn, R. te Boekhorst, and A. Billard, *Effects of Repeated Exposure to a Humanoid Robot on Children With Autism*. London, U.K.: Springer, 2004, pp. 225–236.
[107] A. Ude, C. G. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robot. Auto. Syst.*, vol. 47, nos. 2–3, pp. 93–108, 2004.
[108] J. Saunders, C. L. Nehaniv, and K. Dautenhahn, "Teaching robots by moulding behavior and scaffolding the environment," in *Proc. 1st ACM SIGCHI/SIGART Conf. Hum.-Robot Interaction (HRI)*, 2006, pp. 118–125.
[109] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *Int. J. Robot. Res.*, vol. 29, no. 13, pp. 1608–1639, 2010.
[110] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Trans. Robot.*, vol. 28, no. 6, pp. 1360–1370, Dec. 2012.
[111] E. Rombokas, M. Malhotra, E. A. Theodorou, E. Todorov, and Y. Matsuoka, "Reinforcement learning and synergistic control of the ACT hand," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 2, pp. 569–577, Apr. 2013.
[112] E. Alibekov, J. Kubalik, and R. Babuska, "Policy derivation methods for critic-only reinforcement learning in continuous action spaces," *IFAC-PapersOnLine*, vol. 49, no. 5, pp. 285–290, 2016.
[113] G.-S. Yang, E.-K. Chen, and C.-W. An, "Mobile robot navigation using neural Q-learning," in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 1. Aug. 2004, pp. 48–52.
[114] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, pp. 1107–1149, Dec. 2003.
[115] B.-Q. Huang, G.-Y. Cao, and M. Guo, "Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance," in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 1. Aug. 2005, pp. 85–89.
[116] C. Li, J. Zhang, and Y. Li, "Application of artificial neural network based on Q-learning for mobile robot path planning," in *Proc. IEEE Int. Conf. Inf. Acquisition*, Aug. 2006, pp. 978–982.
[117] J. Qiao, Z. Hou, and X. Ruan, "Application of reinforcement learning based on neural network to dynamic obstacle avoidance," in *Proc. Int. Conf. Inf. Autom.*, Jun. 2008, pp. 784–788.
[118] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 39, pp. 1–40, 2016.
[119] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. (Feb. 2015). "Trust region policy optimization." [Online]. Available: https://arxiv.org/abs/1502.05477
[120] L. B. Prasad, H. O. Gupta, and B. Tyagi, "Application of policy iteration technique based adaptive optimal control design for automatic voltage regulator of power system," *Int. J. Elect. Power Energy Syst.*, vol. 63, pp. 940–949, Dec. 2014.
[121] K. G. Vamvoudakis and J. P. Hespanha, "Online optimal operation of parallel voltage-source inverters using partial information," *IEEE Trans. Ind. Electron.*, vol. 64, no. 5, pp. 4296–4305, May 2016.
[122] A. Heydari, "Optimal switching of DC–DC power converters using approximate dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2016.2635586.
[123] D. J. Pradeep, M. M. Noel, and N. Arun, "Nonlinear control of a boost converter using a robust regression based reinforcement learning algorithm," *Eng. Appl. Artif. Intell.*, vol. 52, pp. 1–9, Jun. 2016.
[124] L.-L. Fan, V. Nasirian, H. Modares, F. L. Lewis, Y.-D. Song, and A. Davoudi, "Game-theoretic control of active loads in DC microgrids," *IEEE Trans. Energy Convers.*, vol. 31, no. 3, pp. 882–895, Sep. 2016.
[125] F. Ruelens, B. Claessens, S. Quaiyum, B. De Schutter, R. Babuska, and R. Belmans, "Reinforcement learning applied to an electric water heater: From theory to practice," *IEEE Trans. Smart Grid*, to be published, doi: 10.1109/TSG.2016.2640184.

Bahare Kiumarsi (M'17) received the B.S. degree from the Shahrood University of Technology, Shahrood, Iran, in 2009, the M.S. degree from the Ferdowsi University of Mashhad, Mashhad, Iran, in 2013, and the Ph.D. degree in electrical engineering from the University of Texas at Arlington, Arlington, TX, USA, in 2017.

Her current research interests include cyber-physical systems, game theory, reinforcement learning, and distributed control of multiagent systems.

Dr. Kiumarsi was a recipient of the UT Arlington N. M. Stelmakh Outstanding Student Research Award and the UT Arlington Graduate Dissertation Fellowship in 2017.

Kyriakos G. Vamvoudakis (SM'15) was born in Athens, Greece. He received the Diploma degree (High Hons.) in electronic and computer engineering from the Technical University of Crete, Chania, Greece, in 2006, and the M.S. and Ph.D. degrees in electrical engineering from The University of Texas at Arlington, Arlington, TX, USA, in 2008 and 2011, respectively.

From 2012 to 2016, he was a Project Research Scientist with the Center for Control, Dynamical Systems, and Computation, University of California, Santa Barbara, CA, USA. He is currently an Assistant Professor with the Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA, USA. He has co-authored one patent, more than 100 technical publications, and two books. His current research interests include game-theoretic control, network security, smart grid, and multiagent optimization.

Dr. Vamvoudakis was a recipient of several international awards including the 2016 International Neural Network Society Young Investigator Award, the Best Paper Award for Autonomous/Unmanned Vehicles at the 27th Army Science Conference in 2010, the Best Presentation Award at the World Congress of Computational Intelligence in 2010, and the Best Researcher Award from the Automation and Robotics Research Institute in 2011. He is currently an Associate Editor of the *Journal of Optimization Theory and Applications*, an Associate Editor of *Control Theory and Technology*, a Registered Electrical/Computer Engineer, and a member of the Technical Chamber of Greece.

Hamidreza Modares (M'15) received the B.Sc. degree from Tehran University, Tehran, Iran, in 2004, the M.Sc. degree from the Shahrood University of Technology, Shahrood, Iran, in 2006, and the Ph.D. degree from the University of Texas at Arlington (UTA), Arlington, TX, USA, in 2015.

From 2006 to 2009, he was with the Shahrood University of Technology as a Senior Lecturer. From 2015 to 2016, he was a Faculty Research Associate with UTA. He is currently an Assistant Professor with the Missouri University of Science and Technology, Rolla, MO, USA. He has authored several journal and conference papers on the design of optimal controllers using reinforcement learning. His current research interests include cyber-physical systems, machine learning, distributed control, robotics, and renewable energy microgrids.

Dr. Modares was a recipient of the Best Paper Award from the 2015 IEEE International Symposium on Resilient Control Systems, the Stelmakh Outstanding Student Research Award from the Department of Electrical Engineering, UTA, in 2015, and the Summer Dissertation Fellowship from UTA, in 2015. He is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Frank L. Lewis (F'94) received the bachelor's degree in physics/electrical engineering and the M.S.E.E. degree from Rice University, Houston, TX, USA, the M.S. degree in aeronautical engineering from the Univesity of West Florida, Pensacola, FL, USA, and the Ph.D. degree from Georgia Tech, Atlanta, GA, USA.

He holds seven U.S. patents, and has authored numerous journal special issues, 363 journal papers, and 20 books including *Optimal Control*, *Aircraft Control*, *Optimal Estimation*, and *Robot Manipulator Control* that are used as university textbooks worldwide. His current research interests include feedback control, intelligent systems, cooperative control systems, and nonlinear systems.

Dr. Lewis was a recipient of the Fulbright Research Award, the NSF Research Initiation Grant, the ASEE Terman Award, the International Neural Network Society Gabor Award, the U.K. Inst Measurement and Control Honeywell Field Engineering Medal, the IEEE Computational Intelligence Society Neural Networks Pioneer Award, the AIAA Intelligent Systems Award, the Texas Regents Outstanding Teaching Award 2013, and the China Liaoning Friendship Award. He is a Founding Member of the Board of Governors of the Mediterranean Control Association. He is a member of the National Academy of Inventors, a fellow of IFAC, AAAS, and the U.K. Institute of Measurement and Control, PE Texas, the U.K. Chartered Engineer, the UTA Distinguished Scholar Professor, the UTA Distinguished Teaching Professor, the Moncrief-ODonnell Chair with the University of Texas at Arlington Research Institute, Arlington, TX, USA, and the Qian Ren Thousand Talents Consulting Professor, Northeastern University, Shenyang, China.