ADAPTIVE OPTIMAL TRACKING OF UNCERTAIN DISCRETE-TIME SYSTEMS

by

BAHARE KIUMARSI KHOMARTASH

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2017

To my love, Reza

ACKNOWLEDGEMENT

I would like to express my special appreciation and thanks to my advisor Professor Frank L. Lewis for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. I could not have imagined having a better advisor and mentor for my Ph.D study. I wish to thank the members of my PhD committee, Dr. Michael Manry, Dr. William Dillon, Dr. Manfred Huber and Dr. Ramitn Madani for their helpful suggestions and for taking time to serve in my dissertation committee.

I also thank Ms. Gail Paniuski and Ms. Priscila Walsh for their assistance with finding the answer to every questions I had. I wish to thank my friend Hamideh Habibi who has helped me throughout my career.

A special thanks to my family. Words cannot express how grateful I am to my mother, father and my brothers for supporting me spiritually throughout my life.

Last but not the least, I would like to thank my beloved husband, Reza, Thank you for supporting me for everything, and especially I can't thank you enough for encouraging me throughout this experience.

26 June 2017

ABSTRACT

ADAPTIVE OPTIMAL TRACKING OF UNCERTAIN DISCRETE-TIME SYSTEMS

BAHARE KIUMARSI KHOMARTASH, Ph.D.

The University of Texas at Arlington, 2017

Suppervising professor: Frank L. Lewis

Optimal feedback control design has been responsible for much of the successful performance of engineered systems in aerospace, manufacturing, industrial processes, vehicles, ships, robotics, and elsewhere. Although most control design methods concern only about the stability of the controlled systems, the stability is a bare minimum requirement and it is desired to design a controller by optimizing some predefined performance criteria. However, the classical optimal control methods rely on offline solutions to complicated Hamilton-Jacobi-Bellman (HJB) equations which require complete knowledge about the system dynamics. Therefore, they are not able to cope with uncertainties and changes in dynamics.

This research presents adaptive control structures based on reinforcement learning (RL) for computing online the solutions to H_2 optimal tracking and H_∞ control of single-agent systems and optimal coordination of multi-agent systems. A family of adaptive controllers is designed that converge in real time to optimal control and game theoretic solutions by using data measured along the system trajectories. First, an alternative approach for formulating the optimal tracking problem in a causal manner is developed that enables us to use RL to find the optimal solutions. On-policy RL is used to solve linear and nonlinear H_2 optimal control problems. In contrast to the

existing methods, the proposed approach for nonlinear systems takes into account the input constraints in the optimization problem by using a nonquadratic performance function. Then, a new model-free method of off-policy learning is presented to find the solution to the H_∞ control problem online in real-time. The proposed method has two main advantages compared to the other mode-free methods. First, the disturbance input does not need to be adjusted in a specific manner. This makes it more practical as the disturbance cannot be specified in most real-world applications. Second, there is no bias as a result of adding a probing noise to the control input to maintain persistence of excitation (PE) condition. Finally, an optimal mode-free solution to the output synchronization problem of heterogeneous discrete-time systems is developed. It is shown that this control protocol implicitly solves the output regulator equations. The relationship between the solution to the output regulator equations and the proposed solution is also shown.

TABLE OF CONTENTS

List of illustrations

Chapter 1

INTRODUCTION

Optimal control of dynamical systems [1]-[6] is an important topic in control engineering. Although most control design methods concern only about the stability of the controlled systems, the stability is a bare minimum requirement and it is desired to design a controller by optimizing some predefined performance criteria. Optimal control problems can be divided into two major groups: optimal regulation and optimal tracking. The aim of the optimal regulation is to make the states of the system go to zero in an optimal manner and the aim of the optimal tracking is to make the states of the system track a reference trajectory. It is well known that finding the optimal regulation solution requires solving the Hamilton–Jacobi–Bellman (HJB) equation, which is a nonlinear partial differential equation. For linear systems with quadratic performance function, the HJB equation reduces to the algebraic Riccati equation (ARE). In contrast to the solution to the optimal regulation problem, which consists of only a feedback term obtained by solving an HJB equation, the solution to the optimal tracking problems consists of two terms: a feedforward term that guarantees perfect tracking and a feedback term that stabilizes the system dynamics. Existing solutions to the optimal tracking problems find the feedforward term using the dynamics inversion concept [7] and the feedback term by solving an HJB equation. However, the classical optimal control methods rely on offline solutions to complicated HJB equations which require complete knowledge about the system dynamics. Therefore, they are not able to cope with uncertainties and changes in dynamics.

Reinforcement learning (RL) [8]-[20] has been widely used in several disciplines to find an optimal policy in an uncertain environment online in real time without requiring complete knowledge about the system dynamics. RL, inspired by learning mechanisms observed in mammals, 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.

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 function related to the current policy, and policy improvement assesses the obtained value function and updates the current policy. There are three well-known types of RL algorithms: policy iteration (PI), value iteration (VI) and generalized policy iteration (GPI). In PI, at each iteration, the critic weights are tunded until convergence while the actor weights are holding constant. This algorithm involves full solution of a Lyapunov equation at each step and is called a full backup in RL terms. In VI, the critic network is tuned for a single step at each iteration, then the actor network is also tuned for a single step. This algorithm involves only a Lyapunov recursion and is called a partial backup in RL. In GPI, the critic weights are tuned a few steps at each iteration and then, the actor wights are tuned a single step.

The interest in RL in control society dates back to the work of [9], [21]-[23]. Watkins' Q-Learning algorithm [24] has also made an impact by considering totally unknown environments. Later, considerable research was conducted for developing RL techniques to find optimal feedback solutions for both discrete-time and continuous-time systems [25]-[54]. Moreover, RL methods have been used to find the solution to zero-sum game and non-zero sum game problems [55]-[62].

Among the existing RL methods, the policy iteration (PI) technique [8], [63] has been widely used for designing feedback controllers. In particular, PI algorithms are used to solve the linear quadratic regulator (LQR) problem for both discrete-time systems and continuous-time systems [26], [28], [29], [64], [65]. It is well known that solving the LQR requires solving an ARE. To find a solution to the ARE, the PI technique starts with an admissible control policy and then iteratively alternates between policy evaluation and policy improvement steps until there is no change in the value or the policy. To avoid the requirement for knowledge of the system dynamics,

in [28], a PI algorithm is developed that converges to the optimal solution of the discrete-time LQR problem using Q-functions [8], [24]. Q-learning does not require any knowledge of the system dynamics.

Although some RL-based algorithms are developed to find the solution to the optimal tracking problem for both nonlinear discrete-time (DT) systems and nonlinear continuous-time (CT) systems [66]-[71], these methods employ the dynamic inversion concept to find the feedforward part of the control input a priori and they only use the RL to find the optimal feedback part of the control input. However, the dynamic inversion technique requires the control input matrix be invertible and complete knowledge of the system dynamics be known or identified a priori.

Moreover, the problem of optimal tracking control of DT systems with input constraints has not been considered yet. This is because the feedback and feedforward control terms are obtained separately and only the feedback term of the control input appears in the performance function, it is not possible to encode input constraints into the performance function, as it was done for the optimal regulation problem [72]–[74]. If the input constraints are not considered a priori, the control input may exceed its permitted bound because of the actuator saturation and this leads to performance degradation or even system instability. Therefore, the existing RL-based solutions to the optimal tracking problem offer no guarantee on the remaining control inputs on their permitted bounds during and after learning.

The H_∞ control is a well-known robust control approach which is used to attenuate the effects of disturbances on the performance of dynamical systems [75]-[77]. It has a strong connection to the zero-sum game problem [78], where the controller and the disturbance are considered as minimizing and maximizing players, respectively. Finding the solution to the zero sum game problem leads to solving the game algebraic Riccati equation (GARE) for the linear systems. Numerical and iterative methods have been widely used to solve the GARE. However, they mostly require complete knowledge of the system dynamics. Q-learning algorithm has also

been used to find the solution to the zero-sum game arising in H_∞ optimal control problem [56]. Although elegant, there are two main problems with this algorithm. First, Q-learning requires the disturbance input to be updated in a prescribed manner. However, the disturbance input cannot be updated in a prescribed manner in more real-world applications. Second, Q-learning algorithm does not cancel out the effects of probing noise (which is used to excite the system) in the Bellman equation while evaluating the value function. This may result in bias and can affect the convergence of the algorithm.

Finally, the state synchronization of the leader-following homogeneous multi-agent systems, where all agents and leader have identical dynamics, has been well established for both DT and CT systems [79]-[83]. However, in many practical applications, the agents' dynamics may not be the same. Therefore, it is desired to design distributed output synchronization control protocols for heterogeneous systems, which may have non identical dynamics. The output synchronization problem for DT multi-agent systems has received considerably less attention [84]-[88]. Existing output synchronization methods require solution of the output regulator equations. However, this requires complete knowledge of all agents's and the leader's dynamics. Moreover, existing results only consider making the steady-state tracking errors go to zero and do not provide an optimal solution that not only provides a zero steady-state tracking error but also minimizes the transient response. The output synchronization problem for DT systems with unknown dynamics is not considered in the literature.

Based on the above elaborated problems, the research objectives of this dissertation are to address these mentioned issues and provide efficient RL-based methods.

The rest of the dissertation is organized as follows.

- In Chapter 2, online model-free solution using RL algorithms to the infinite-horizon linear quadratic tracking (LQT) for DT systems is developed. The LQT problem is first transformed into minimizing a discounted performance function subject to an augmented system composed of the original system and the

command generator system. An LQT ARE equation is then developed which gives both feedforward and feedback parts of optimal control solution simultaneously. In the first part, full state feedback is assumed available for control. Then, a Q-learning algorithm is proposed to solve the LQT without requiring any knowledge of the augmented system dynamics. In the second part, to obviate the requirement for the knowledge of the state of the system, the state of the augmented system is constructed from the delayed input, output, and reference trajectory data. A Bellman equation is then developed which only requires the available measured data to evaluate a policy. Finally, based on this Bellman equation, policy iteration algorithm is presented to learn the solution to the LQT without requiring the knowledge of the system dynamics and the state of the system.

- In chapter 3, a partially model-free adaptive optimal control solution to the nonlinear DT tracking control problem in the presence of input constraints is presented. In contrast to the standard solution, which finds the feedforward and feedback terms of the control input separately, the minimization of the proposed discounted performance function gives both feedback and feedforward parts of the control input simultaneously. This enables us to encode the input constraints into the optimization problem by using a nonquadratic performance function. An actor-critic based reinforcement learning algorithm is used to learn the solution to the tracking HJB equation online without requiring knowledge of the system drift dynamics.
- In Chapter 4, a model-free solution to the H_∞ control of linear discrete-time systems is presented. The proposed approach employs off-policy RL to solve the game algebraic Riccati equation online using the measured data along the system trajectories. Like existing model-free RL algorithms, no knowledge of the

system dynamics is required. However, the proposed method has two main advantages. First, the disturbance input does not need to be adjusted in a specific manner. Second, there is no bias as a result of adding a probing noise to the control input to maintain persistence of excitation (PE) condition. Consequently, the convergence of the proposed algorithm is not affected by probing noise.

- Chapter 5 presents an optimal model-free solution to the output synchronization of heterogeneous multi-agent DT systems. First, local discounted performance functions are defined for all agents and the optimal synchronization control protocols are found by solving a set of algebraic Riccati equations (AREs) and without requiring the explicit solution to the output regulator equations. It is shown that the proposed method implicitly solves the output regulator equations and therefore solves the output synchronization problem, provided that the discount factor is bigger than a lower bound. This formulation enables us to develop a Q-learning algorithm to solve the AREs using only measured data and so find the optimal distributed control protocols for each agent without requiring complete knowledge of the agents's or leader's dynamics. It is shown that the combination of a distributed adaptive observer and the controller guarantees synchronization. The relationship between the standard solution and the proposed solution is also shown.
- Chapter 6 summarizes and concludes the dissertation and recommends the future research works that extend the proposed materials in this dissertation.

The publications resulted from this dissertation are listed below:

[1] 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, 2014.

[2] 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 Transactions on Cybernetics, vol. 45, no. 12, pp. 2770–2779, 2015.

[3] B. Kiumarsi and F. L. Lewis, "Actor-critic-based optimal tracking for partially unknown nonlinear discrete-time systems," IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 1, pp. 140–151, 2015.

[4] 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, 2017.

[5] B. Kiumarsi and F. L. Lewis," Output synchronization of heterogeneous discrete-time systems: A model-free optimal approach," Automatica, 2017.

Chapter 2

OPTIMAL TRACKING CONTROL OF COMPLETELY UNKNOWN LINEAR DISCRETE-TIME SYSTEMS: STATE FEEDBACK AND OUTPUT FEEDBACK

2.1. Introduction

In this chapter, the online model-free solution using reinforcement learning (RL) algorithms to the infinite-horizon linear quadratic tracking (LQT) for discrete-time systems is developed. It is assumed that the reference trajectory is generated by a linear command generator. Although the value function for the LQT is not quadratic in general, it is shown that for the given command generator and the reward function, the LQT value function is quadratic in the state of the system and the reference trajectories. The quadratic nature of the value function for the LQT allows development of a Bellman equation which uses only knowledge of the state of the system and the reference trajectories to find the value related to a control policy. Then, an augmented system composed of the original system dynamics and the command generator dynamics is formed. Based on this augmented system, an augmented ARE is derived whose solution yields the solution to the LQT. That is, once the augmented ARE is solved, both the feedback and feedforward terms of the control input are obtained simultaneously. In the first part, full state feedback is assumed available for control. Then, a Q-learning algorithm is proposed to solve the LQT without requiring any knowledge of the augmented system dynamics. It is verified that starting from an admissible control policy, the proposed Q-learning algorithm converges to the optimal control solution.

In the second part, to obviate the requirement for the knowledge of the state of the system, the state of the augmented system is constructed from the delayed input, output, and reference trajectory data. A Bellman equation is then developed which only requires the available measured data to evaluate a policy. Finally, based on this Bellman equation, policy iteration algorithm is presented to learn the solution to the LQT without requiring the knowledge of the system dynamics and the state of the system.

The rest of this chapter is organized as follows. The review of the standard solution for the LQT problem is given in section 2.2. An alternative approach for formulating the infinite-horizon LQT in a causal manner is presented in Section 2.3. In Section 2.4, a novel Q-learning algorithm is proposed to solve LQT without any knowledge of the augmented system dynamics using state feedback. In Section 2.5, a novel Bellman equation is presented which uses only measured data along the system trajectories to evaluate a fixed control policy. The proposed online model-free policy iteration method and its convergence proof are developed in Section 2.6 which finds the solution to the LQT using only measured data along the system trajectories. Finally, Sections 2.7 and 2.8 present the simulation results and the conclusion, respectively.

2.2. LQT Problem and its Standard Solution

In this section, we review the standard solution for the linear quadratic tracker (LQT) problem. It is assumed that the reference trajectory approaches zero as time goes to infinity.

Consider the linear discrete-time (DT) system

$$\begin{aligned} x_{k+1} &= A\,x_k + B\,u_k \\ y_k &= C\,x_k \end{aligned} \tag{2.1}$$

where $x_k \in \mathbb{R}^n$ is the measured state, $u_k \in \mathbb{R}^m$ is the control input, $y_k \in \mathbb{R}^p$ is the output and A, B and C are constant matrices with compatible dimensions.

For the infinite-horizon LQT problem, the goal is to design an optimal controller for the system (2.1) which ensures that the output y_k tracks a reference trajectory r_k and guarantees stability. This can be achieved by minimizing the following infinite-horizon performance index

$$J_k = \frac{1}{2}\sum_{i=k}^{\infty} U_i = \frac{1}{2}\sum_{i=k}^{\infty}\Big[(C\,x_i - r_i)^T Q\,(C\,x_i - r_i) + u_i^T R\,u_i\Big] \tag{2.2}$$

where $\overline{r}_k = \{r_k, r_{k+1}, \ldots\}$, U_i is the utility function at time step i, and $Q > 0$ and $R > 0$ are symmetric matrices.

The standard solution using the calculus of variation is provided as follows. Considering the system (2.1) with the performance index (2.2), the costate equation is given by [2]

$$\lambda_k = A^T \lambda_{k+1} + C^T Q C x_k - C^T Q r_k \tag{2.3}$$

where λ_k is the costate variable. The stationarity condition for finding the optimal control

$$0 = B^T \lambda_{k+1} + R u_k \tag{2.4}$$

Therefore, the optimal control is

$$u_k = -R^{-1} B^T \lambda_{k+1} \tag{2.5}$$

It is clear that the optimal control is a linear costate feedback, but because of the last term in the costate equation, it is no longer possible to express it as a linear state feedback as for the LQ regulator. However, u_k can be expressed as a combination of a linear state variable feedback plus a term depending on r_k [2]. Thus,

$$\lambda_k = S x_x - v_k^{SS} \tag{2.6}$$

for some as yet unknown auxiliary sequence v_k^{SS} and gain S. This will turn out to be a valid assumption if a consistent equation can be found for v_k^{SS}. Using (2.1), (2.3), (2.5) and (2.6), and some manipulations yields

$$u_k = -K_x x_k + K_v v_k^{SS} \tag{2.7}$$

where $v_k^{SS} = \lim_{T \to \infty} v_k$ with

$$v_k = (A - B K_x)^T v_{k+1} + C^T Q r_k, \qquad v(T) = 0 \tag{2.8}$$

and

$$K_x = (B^T S B + R)^{-1} B^T S A \tag{2.9}$$

$$K_v = (B^T S B + R)^{-1} B^T \tag{2.10}$$

where S is obtained from solving the following algebraic Riccati equation (ARE)

$$C^T Q\, C - S + A^T S\, A - A^T S\, B(B^T S\, B + R)^{-1} B^T S\, A = 0 \tag{2.11}$$

Sufficient conditions for existence of a solution $S = S^T > 0$ to the ARE are (A, B) stabilizable and $(A, \sqrt{Q}C)$ observable.

Remark 2.1. From (2.7) it is observed that the control input consists of a feedback term linear in x_k plus a feedforward term independent of x_k. The gain K_x of the first term depends on the solution of the ARE (2.11) and the second term depends on the difference equation (2.8). A drawback of this formulation of the LQT problem is the need to solve for v_k backwards in time. That is, the standard LQT solution is noncausal.

Remark 2.2. Note that the assumption that the reference trajectory approaches zero as time goes to infinity is essential for minimizing the performance index (2.2). This is because the control input contains a part depending on the reference trajectory which makes (2.2) unbounded if the reference trajectory does not approach zero. Therefore, the meaning of minimality is lost. In the subsequent sections it is shown that we can relax this restrictive assumption by using a discount factor in the performance index.

Remark 2.3. A disadvantage to the standard LQT solution in Section 2.2 is that it can only be used for a class of reference trajectories that are generated by an asymptotically stable command generator. Another disadvantage of this solution is the need to compute the noncausal signal v_k using backward recursion (2.8). Therefore, the infinite-horizon LQT problem has not received much attention in the literature.

2.3. Causal Solution to the LQT Problem and Quadratic Form of the LQT Value Function

In this section, we propose an alternative approach for formulating the infinite-horizon LQT problem in a causal manner. First, it is assumed that the reference trajectory is generated by a linear command generator and it is shown that in this case the value function of the LQT

problem can be expressed as a quadratic form in terms of x_k and r_k. Then, a Bellman equation is developed for the LQT, and an augmented LQT ARE is given. This allows us to use reinforcement learning (RL) to solve the LQT problem online in Section 2.4.

2.3.1 Quadratic form for the LQT value function

Before proceeding, the following assumption is made.

Assumption 2.1. The reference trajectory for the LQT problem is produced by the command generator model

$$r_{k+1} = F\, r_k \tag{2.12}$$

This command generator model does not assume that F is Hurwitz. As such, it can generate a large class of useful command trajectories, including unit step (useful, e.g., in position command), sinusoidal waveforms (useful, e.g., in hard disk drive control), the ramp (useful in velocity tracking systems, e.g., satellite antenna pointing), and more.

Based on the system dynamics (2.1) and the reference trajectory dynamics (2.12), construct the augmented system

$$X_{k+1} = \begin{bmatrix} x_{k+1} \\ r_{k+1} \end{bmatrix} = \begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & F \end{bmatrix} \begin{bmatrix} x_k \\ r_k \end{bmatrix} + \begin{bmatrix} B \\ \mathbf{0} \end{bmatrix} u_k \equiv T\, X_k + B_1 u_k \tag{2.13}$$

where the augmented state is

$$X_k = \begin{bmatrix} x_k \\ r_k \end{bmatrix} \tag{2.14}$$

The performance index (2.2) can be only used if F is Hurwitz. In practice this is not true, for instance, tracking of unit step and sinusoidal commands. In the following, it is shown that by introducing a discount factor in the performance index one can implement the infinite-horizon LQT even for the cases that the command generator dynamics F is not Hurwitz. Consider the following discounted performance index or value function

$$V(x_k, \overline{r}_k) = \frac{1}{2}\sum_{i=k}^{\infty} \gamma^{i-k} U_i = \frac{1}{2}\sum_{i=k}^{\infty} \gamma^{i-k} \left[(C x_i - r_i)^T Q (C x_i - r_i) + u_i^T R u_i \right] \tag{2.15}$$

where $0 < \gamma \leq 1$ is the discount factor. Note that $\gamma = 1$ can be only used if one knows a priori that the reference trajectory is generated by an asymptotically stable command generator system. That is, if F in (2.12) is Hurwitz.

Note that the value function (2.15) can be written in terms of the augmented state as

$$V(x_k, \overline{r}_k) = \frac{1}{2}\sum_{i=k}^{\infty} \gamma^{i-k} \left[X_i^T Q_1 X_i + u_i^T R u_i \right] \tag{2.16}$$

where

$$Q_1 = C_1^T Q C_1 \tag{2.17}$$

with $C_1 = [C \quad -I]$.

The next Lemma shows that the value function is quadratic in the state of the augmented system.

Lemma 2.1. Quadratic form for the value function

For the infinite-horizon LQT problem, under Assumption 2.1, for any fixed stabilizing policy

$$u_i = K_x x_i + K_r r_i \tag{2.18}$$

the value function (2.15) can be written as

$$V(x_k, \overline{r}_k) = V(x_k, r_k) = V(X_k) = \frac{1}{2} X_k^T P X_k \tag{2.19}$$

for some matrix $P = P^T > 0$.

Proof. Using (2.18) in (2.15) yields

$$V(x_k,\overline{r}_k)=\frac{1}{2}\sum_{i=k}^{\infty}\gamma^{i-k}\Big[(Cx_i-r_i)^TQ(Cx_i-r_i)+(K_xx_i+K_rr_i)^TR(K_xx_i+K_rr_i)\Big]=$$
$$\frac{1}{2}\sum_{i=0}^{\infty}\gamma^i\Big[x_{i+k}^T(C^TQC+K_x^TRK_x)x_{i+k}+x_{i+k}^T(-C^TQ+K_x^TRK_r)r_{i+k} \qquad (2.20)$$
$$+r_{i+k}^T(-QC+K_r^TRK_x)x_{i+k}+r_{i+k}^T(Q+K_r^TRK_r)r_{i+k}\Big]$$

Note that using (2.18), the solution of system dynamics (2.1) and reference trajectory (2.12) for specific initial condition x_k and r_k are

$$r_{i+k}=F^i\,r_k \qquad (2.21)$$

$$x_{i+k}=G^ix_k+M\,r_k \qquad (2.22)$$

where $G=A+BK_x$ and $M=\sum_{n=0}^{i-1}G^{i-n-1}BK_rF^n$.

Putting (2.21) and (2.22) in (2.20), results in

$$V(x_k,r_k)=\frac{1}{2}x_k^TP_{11}x_k+\frac{1}{2}x_k^TP_{12}r_k+\frac{1}{2}r_k^TP_{21}x_k+\frac{1}{2}r_k^TP_{22}r_k \qquad (2.23)$$

where

$$P_{11}=\sum_{i=0}^{\infty}\gamma^i\Big[(G^i)^T(C^TQC+K_x^TRK_x)G^i\Big] \qquad (2.24)$$

$$P_{12}=\sum_{i=0}^{\infty}\gamma^i\Big[(G^i)^T(-C^TQ+K_x^TRK_r)F^i+(G^i)^T(C^TQC+K_x^TRK_x)M\Big] \qquad (2.25)$$

$$P_{21}=\sum_{i=0}^{\infty}\gamma^i\Big[(F^i)^T(-QC+K_r^TRK_x)G^i+M^T(C^TQC+K_x^TRK_x)G^i\Big] \qquad (2.26)$$

$$P_{22}=\sum_{i=0}^{\infty}\gamma^i\Big[M^T(-C^TQ+K_x^TRK_r)F^i+M^T(C^TQC+K_x^TRK_x)M+ \qquad (2.27)$$
$$(F^i)^T(-QC+K_r^TRK_x)M+(F^i)^T(Q+K_r^TRK_r)F^i\Big]$$

Therefore (2.19) holds with

$$P=\begin{bmatrix}P_{11} & P_{12}\\ P_{21} & P_{22}\end{bmatrix} \qquad (2.28)$$

∎

This completes the proof**.**

2.3.2 Bellman equation and ARE for the LQT problem

In this subsection, we derive a LQT Bellman equation and an augmented LQT ARE in terms of *P* in (2.19).

On the basis of (2.15) and (2.19), one has

$$V(x_k, r_k) = \frac{1}{2}(Cx_k - r_k)^T Q(Cx_k - r_k) + \frac{1}{2}u_k^T R\, u_k + \frac{\gamma}{2}\sum_{i=k+1}^{\infty}\gamma^{i-(k+1)}\left[(Cx_i - r_i)^T Q\,(Cx_i - r_i) + u_i^T R\, u_i\right] \tag{2.29}$$

which yields the LQT Bellman equation

$$V(x_k, r_k) = \frac{1}{2}(Cx_k - r_k)^T Q(Cx_k - r_k) + \frac{1}{2}u_k^T R\, u_k + \gamma\, V(x_{k+1}, r_{k+1}) \tag{2.30}$$

Using (2.19) in (2.30), we obtain the LQT Bellman equation in terms of value function kernel matrix *P* as

$$X_k^T P\, X_k = X_k^T Q_1 X_k + u_k^T R\, u_k + \gamma\, X_{k+1}^T P\, X_{k+1} \tag{2.31}$$

where Q_1 is defined (2.17).

Define the LQT Hamiltonian

$$H(X_k, u_k) = X_k^T Q_1 X_k + u_k^T R\, u_k + \gamma X_{k+1}^T P\, X_{k+1} - X_k^T P\, X_k \tag{2.32}$$

or equivalently

$$H(X_k, u_k) = X_k^T Q_1 X_k + u_k^T R\, u_k + \gamma V(X_{k+1}) - V(X_k) \tag{2.33}$$

The next theorem shows how the LQT problem can be solved in a causal manner using an augmented ARE.

Theorem 2.1. ARE for causal solution of the LQT problem. Under Assumption 2.1 and using (2.19), any optimal policy for the LQT problem has the form

$$u_k = -K_1 X_k \tag{2.34}$$

where

$$K_1 = (R + \gamma B_1^T P B_1)^{-1} \gamma B_1^T P T \tag{2.35}$$

and P satisfies the augmented LQT ARE

$$Q_1 - P + \gamma T^T P T - \gamma^2 T^T P B_1 (R + \gamma B_1^T P B_1)^{-1} B_1^T P T = 0 \tag{2.36}$$

Proof. A necessary condition for optimality [30] is the stationary condition

$$\frac{\partial H(X_k, u_k)}{\partial u_k} = 2 R u_k + \gamma \frac{\partial X_{k+1}}{\partial u_k}^T \frac{\partial V(X_{k+1})}{\partial X_{k+1}} = 2 R u_k + 2 \gamma B_1^T P X_{k+1} = 0$$

Then

$$u_k = -(R + \gamma B_1^T P B_1)^{-1} \gamma B_1^T P T X_k \tag{2.37}$$

Substituting (2.13) and (2.37) in the Bellman equation (2.31) results in the LQT ARE (2.36).

In the following theorem, the stability of tracking error for the optimal control input, given by solving the LQT ARE (2.36), is discussed. It is shown that the convergence of the tracking error to zero cannot be guaranteed because of using the discount factor in the value function. However, it is discussed that by choosing a proper discount factor and a weighting matrix *Q* in the value function, one can make the tracking error as small as desired. ∎

Theorem 2.2. Stability and optimality of the LQT ARE solution

Consider the LQT problem for the systems (2.1) with the command generator (2.12) and the value function (2.15). Define $\bar{e}_k = \gamma^{k/2} e_k$, where $e_k = C x_k - r_k$ is the tracking error at sample time k. Then, the optimal control input obtained by solving the LQT ARE (2.36) asymptotically stabilizes $\bar{e}_k$. Moreover, it minimizes the value function (2.15) over all stabilizing controls.

Proof. We first show $\bar{e}_k$ is asymptotically stable. Consider the augmented systems (2.13) with the state X_k. Define the new state $\bar{X}_k = \gamma^{k/2} X_k$. Since $e_k = [C \ -I] X_k$ and $[C \ -I] \neq 0$, if $\bar{X}_k$

goes to zero, then $\bar{e}_k$ goes to zero. In the following, it is shown that $\bar{X}_k$ and consequently $\bar{e}_k$ converges to zero as k goes to infinity.

Consider the following Lyapunov function

$$V(\bar{X}_k) = \frac{1}{2}\bar{X}_k{}^T P \bar{X}_k \tag{2.38}$$

where P is the solution of the LQT ARE (2.36). Then we have

$$V(\bar{X}_{k+1}) - V(\bar{X}_k) = \frac{1}{2}\bar{X}_{k+1}{}^T P \bar{X}_{k+1} - \frac{1}{2}\bar{X}_k{}^T P \bar{X}_k \tag{2.39}$$

Using $X_k = \gamma^{-k/2}\bar{X}_k$ and the control input (2.34) in (2.13), one has

$$\bar{X}_{k+1} = \gamma^{1/2} T \bar{X}_k + \gamma^{1/2} B_1 \bar{u}_k = \gamma^{1/2}(T - B_1(R + \gamma B_1^T P B_1)^{-1}\gamma B_1^T P T)\bar{X}_k \tag{2.40}$$

where $\bar{u}_k = -K_1\bar{X}_k$. Putting (2.40) in (2.39) and adding and subtracting $K_1^T R K_1$ and some manipulations yields

$$V(\bar{X}_{k+1}) - V(\bar{X}_k) = \frac{1}{2}\bar{X}_k{}^T\left[-P + \gamma T^T P T - \gamma^2 T^T P B_1(R + \gamma B_1^T P B_1)^{-1} B_1^T P T - K_1{}^T R K_1\right]\bar{X}_k \tag{2.41}$$

where K_1 is defined in (2.35). From (2.36) one has

$$-P + \gamma T^T P T - \gamma^2 T^T P B_1(R + \gamma B_1^T P B_1)^{-1} B_1^T P T = -Q_1 \tag{2.42}$$

Putting (2.42) in (2.41) yields

$$V(\bar{X}_{k+1}) - V(\bar{X}_k) = \frac{1}{2}\bar{X}_k{}^T(-Q_1 - K_1{}^T R K_1)\bar{X}_k < 0 \tag{2.43}$$

This completes the proof of the stability.

To show the optimality, note that

$$\frac{1}{2}(\bar{X}_{\infty}{}^{T}P\,\bar{X}_{\infty}-\bar{X}_{k}{}^{T}P\,\bar{X}_{k})=\frac{1}{2}\sum_{i=k}^{\infty}[\bar{X}_{i+1}{}^{T}P\,\bar{X}_{i+1}-\bar{X}_{i}{}^{T}P\,\bar{X}_{i}]=$$
$$\frac{1}{2}\sum_{i=k}^{\infty}[\gamma(T\bar{X}_{i}+B_{1}\bar{u}_{i})^{T}P\,(T\bar{X}_{i}+B_{1}\bar{u}_{i})-\bar{X}_{i}{}^{T}P\,\bar{X}_{i}]=\frac{1}{2}\sum_{i=k}^{\infty}[\bar{X}_{i}{}^{T}(\gamma T^{T}PT-P)\bar{X}_{i}+\gamma\bar{X}_{i}{}^{T}T^{T}B_{1}\,\bar{u}_{i}+$$
$$\gamma\,\bar{u}_{i}{}^{T}B_{1}^{T}PT\,\bar{X}_{i}+\gamma\,\bar{u}_{i}{}^{T}B_{1}^{T}P\,B_{1}\,\bar{u}_{i}] \tag{2.44}$$

Using the LQT ARE (2.36) in (2.44) and since $\bar{X}_{\infty}=0$, one has

$$\frac{1}{2}\,\bar{X}_{k}{}^{T}P\,\bar{X}_{k}+\frac{1}{2}\sum_{i=k}^{\infty}[\bar{X}_{i}{}^{T}\big(-Q_{1}+\gamma^{2}\,T^{T}\,P\,B_{1}(R+\gamma\,B_{1}^{T}P\,B_{1})^{-1}B_{1}^{T}P\,T\big)\bar{X}_{i}+$$
$$\gamma\bar{X}_{i}{}^{T}T^{T}PB_{1}\,\bar{u}_{i}+\gamma\bar{u}_{i}{}^{T}B_{1}^{T}PT\,\bar{X}_{i}+\gamma\bar{u}_{i}{}^{T}B_{1}^{T}P\,B_{1}\,\bar{u}_{i}]=0 \tag{2.45}$$

On the other hand, the value function (2.15) in terms of $\bar{X}_{k}$ and $\bar{u}_{k}$ can be written as

$$V(\bar{X}_{k},\bar{u}_{k})=\frac{1}{2}\gamma^{-k}\sum_{i=k}^{\infty}[\bar{X}_{i}^{T}Q_{1}\bar{X}_{i}+\bar{u}_{i}^{T}R\,\bar{u}_{i}] \tag{2.46}$$

In fact, minimizing the value function (2.15) with respect to the system (2.13) is equivalent to minimizing the value function (2.46) with respect to (2.40).

Multiplying the right-hand side of (2.45) by γ^{-k} and adding its result to (2.46) yields

$$V(\bar{X}_{k},\bar{u}_{k})=\frac{1}{2}\,X_{k}{}^{T}P\,X_{k}+\frac{\gamma^{-k}}{2}\sum_{i=k}^{\infty}[\bar{X}_{i}{}^{T}\big(\gamma^{2}\,T^{T}\,P\,B_{1}(R+\gamma\,B_{1}^{T}P\,B_{1})^{-1}B_{1}^{T}P\,T\big)\bar{X}_{i}+$$
$$\gamma\bar{X}_{i}{}^{T}T^{T}PB_{1}\,\bar{u}_{i}+\gamma\bar{u}_{i}{}^{T}B_{1}^{T}PT\,\bar{X}_{i}+\bar{u}_{i}{}^{T}(R+\gamma B_{1}^{T}P\,B_{1})\,\bar{u}_{i}] \tag{2.47}$$

Completing the square gives

$$V(\bar{X}_{k},\bar{u}_{k})=\frac{1}{2}\,X_{k}{}^{T}P\,X_{k}+\frac{\gamma^{-k}}{2}\sum_{i=k}^{\infty}[\bar{u}_{i}+(R+\gamma\,B_{1}^{T}P\,B_{1})^{-1}\gamma B_{1}^{T}P\,T\,\bar{X}_{i}]^{T}(R+\gamma\,B_{1}^{T}P\,B_{1})\times$$
$$[\bar{u}_{i}+(R+\gamma\,B_{1}^{T}P\,B_{1})^{-1}\gamma B_{1}^{T}P\,T\,\bar{X}_{i}] \tag{2.48}$$

Since $R>0$, the (2.46) achieves its minimum when $\bar{u}_{k}=-K_{1}\bar{X}_{k}$, where K_{1} is given in (2.35). Consequently, $u_{k}=-K_{1}X_{k}$ minimizes the value function (2.15) and this completes the proof of the optimality. ∎

Remark 2.4. Theorem 2.2 shows that the tracking error is bounded when the optimal control input obtained by the LQT ARE is applied to the system. Moreover, Eq. (2.43) shows that the larger the *Q* in the value function is the faster the tracking error decreases (see (2.17)). Therefore, by choosing a smaller discount factor and/or larger *Q* one can make the tracking error as small as desired before the value of γ^i becomes very small.

2.4. Reinforcement Learning to Solve LQT Online

In this section, we use the causal LQT formulation of Section 2.3 to develop RL algorithms for the LQT, where the value function and control law are updated by recursive iterations online using data measured along the system trajectories.

For an arbitrary stabilizing gain K_1 in (2.34), the augmented LQT Bellman equation (2.31) becomes the LQT Lyapunov equation

$$Q_1 - P + K_1^T R K_1 + \gamma (T - B_1 K_1)^T P (T - B_1 K_1) = 0 \tag{2.49}$$

Instead of directly solving the LQT ARE (2.36), the following policy iteration (PI) algorithm based on repeated solutions of (2.49) can be employed.

Algorithm 2.1. Offline policy iteration for LQT solution

Initialization: Start with a stabilizing control policy K_1.

1. Policy evaluation, solve for P^{j+1} using the LQT Lyapunov equation

$$P^{j+1} = Q_1 + (K_1^j)^T R K_1^j + \gamma (T - B_1 K_1^j)^T P^{j+1} (T - B_1 K_1^j) \tag{2.50}$$

2. Policy improvement

$$K_1^{j+1} = (R + \gamma B_1^T P^{j+1} B_1)^{-1} \gamma B_1^T P^{j+1} T \tag{2.51}$$

This algorithm is an extension of Hewer's method [89] to the LQT problem. The proof shows that P^j in Algorithm 2.1 converges to the solution to the LQT ARE (2.36) and that K_1 is stabilizing at each step.

The Lyapunov equation (2.50) in Algorithm 2.1 evaluates a fixed control policy in an offline manner and it requires complete knowledge of the system dynamics. However, one can use the Bellman equation (2.31), instead of the Lyapunov equation (2.50), to evaluate a control policy in an online manner and without requiring knowledge of the system dynamics. The next algorithm uses the LQT Bellman equation (2.31) to solve the LQT online.

Algorithm 2.2. Online policy iteration for LQT solution

Initialization: Start with a stabilizing control policy K_1.

1. Policy evaluation, solve for P^{j+1} using the LQT Bellman equation

$$X_k^T P^{j+1} X_k = X_k^T (Q_1 + (K_1^j)^T R K_1^j) X_k + \gamma X_{k+1}^T P^{j+1} X_{k+1} \tag{2.52}$$

2. Policy improvement

$$K_1^{j+1} = (R + \gamma B_1^T P^{j+1} B_1)^{-1} \gamma B_1^T P^{j+1} T \tag{2.53}$$

Policy iteration Algorithm 2.2 can be implemented online using Least-squares (LS) using the data tuple X_k, X_{k+1} and ρ_k measured along the system trajectories with $\rho_k = X_k^T \ Q_1 + (K_1^j)^T R K_1^j \ X_k$. In fact (2.52) is a scalar equation and *P* is a symmetric $(n+p)\times(n+p)$ matrix with $(n+p)\times(n+p+1)/2$ independent element. Therefore at least $(n+p)\times(n+p+1)/2$ data tuples are required before (2.52) can be solved using LS. Both batch LS and recursive LS methods can be used to perform policy evaluation step (2.52). The system dynamics (T, B_1) is not needed to solve Bellman equation (2.52), but must be known to update the control policy using (2.53).

To obviate the requirement for complete knowledge of the system dynamics, a Q-learning algorithm in the Section 2.5 and a policy iteration algorithm in Section 2.6 are developed which use the states and outputs information of the system to find the optimal solution, respectively.

2.5. Q-learning to Solve the LQT Online

The online LQT policy iteration Algorithm 2.2 requires knowledge of the system dynamics (T, B_1). In this section a Q-learning algorithm [24], [90] is developed that solves the LQT ARE (2.36) online without requiring any knowledge of the system dynamics (A, B) or command generator dynamics (F).

2.5.1. Q-function for the LQT

Based on the LQT Bellman equation (2.31), the discrete-time LQT Q-function is defined as

$$Q(x_k, r_k, u_k) = \frac{1}{2} X_k^T Q_1 X_k + \frac{1}{2} u_k^T R u_k + \frac{1}{2} \gamma X_{k+1}^T P X_{k+1} \tag{2.54}$$

where Q_1 is defined in (2.17).

By using augmented system dynamics (2.13), (2.54) becomes

$$\begin{aligned} Q(X_k, u_k) &= \frac{1}{2} X_k^T Q_1 X_k + \frac{1}{2} u_k^T R u_k + \frac{1}{2} \gamma (T X_k + B_1 u_k)^T P (T X_k + B_1 u_k) \\ &= \frac{1}{2} \begin{bmatrix} X_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q_1 + \gamma T^T P T & \gamma T^T P B_1 \\ \gamma B_1^T P T & R + \gamma B_1^T P B_1 \end{bmatrix} \begin{bmatrix} X_k \\ u_k \end{bmatrix} \end{aligned} \tag{2.55}$$

Therefore, Define

$$Q(X_k, u_k) = \frac{1}{2} \begin{bmatrix} X_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} X_k \\ u_k \end{bmatrix} = \frac{1}{2} \begin{bmatrix} X_k \\ u_k \end{bmatrix}^T \begin{bmatrix} H_{XX} & H_{Xu} \\ H_{uX} & H_{uu} \end{bmatrix} \begin{bmatrix} X_k \\ u_k \end{bmatrix} \tag{2.56}$$

for kernel matrix $H = H^T$.

Applying $\frac{\partial Q(X_k, u_k)}{\partial u_k} = 0$ to (2.56) yields

$$u_k = -H_{uu}^{-1} H_{uX} X_k \tag{2.57}$$

and to (2.55) yields

$$u_k = -(R + \gamma B_1^T P B_1)^{-1} \gamma B_1^T P T X_k \tag{2.58}$$

as in equation (2.34).

Eq. (2.58) requires knowledge of the augmented system dynamics (T, B_1) to compute the LQT control. On the other hand, (2.57) requires knowledge only of the Q-function matrix kernel *H*. RL is used in the next subsection to determine the kernel matrix *H* online without knowing the augmented system dynamics using data measured along the system trajectories.

2.5.2. Q-learning for the LQT

Based on the definition of Q-function (2.54), one can introduce a Q-learning algorithm to solve the LQT ARE (2.36) online without knowing the augmented system dynamics (T, B_1).

The infinite-horizon Q-function is given by (2.54). Hence the Q-function satisfies the Bellman equation

$$Q(X_k, u_k) = \frac{1}{2} X_k^T Q_1 X_k + \frac{1}{2} u_k^T R u_k + \gamma Q(X_{k+1}, u_{k+1}) \tag{2.59}$$

where the policy $u_{k+1} = -K_1 X_{k+1}$ is followed after time *k*.

Define

$$Z_k = \begin{bmatrix} X_k \\ u_k \end{bmatrix} \tag{2.60}$$

to write (2.56) as

$$Q(X_k, u_k) = \frac{1}{2} Z_k^T H Z_k \tag{2.61}$$

By substituting (2.61) into (2.59), the Q-function Bellman equation (2.59) becomes

$$Z_k^T H Z_k = X_k^T Q_1 X_k + u_k^T R u_k + \gamma Z_{k+1}^T H Z_{k+1} \tag{2.62}$$

Policy iteration is especially easy to implement in terms of the Q-function, as follows.

Algorithm 2.3. LQT Policy Iteration solution using the LQT Q-function

1. Policy evaluation

$$Z_k^T H^{j+1} Z_k = X_k^T Q_1 X_k + (u_k^j)^T R (u_k^j) + \gamma Z_{k+1}^T H^{j+1} Z_{k+1} \tag{2.63}$$

2. Policy improvement

$$u_k^{j+1} = -(H_{uu}^{-1})^{j+1} H_{uX}^{j+1} X_k \tag{2.64}$$

Note that in contrast to the Bellman equation (2.52) in Algorithm 2.2, the control input appears in quadratic form of the Q-function Bellman equation (2.63). Therefore, in contrast to Algorithm 2.2, the policy improvement step (2.64) in Algorithm 2.3, which is given by minimizing the Q-function (2.63) with respect to the control input, can be carried out in terms of the learned kernel matrix H^{j+1} without resorting to the system dynamics.

The convergence of Algorithm 2.3 can be proven as in [56]. Note that, policy iteration using Q-function is performed online and can be implemented without requiring any knowledge of the augmented system dynamics based on Least-squares (LS) using the data tuple Z_k, Z_{k+1} and ρ_k measured along the system trajectories with $\rho_k = X_k^T Q_1 X_k + (u_k^j)^T R u_k^j$. In fact (2.63) is a scalar equation and *H* is a symmetric $(n+p+m)\times(n+p+m)$ matrix with $(n+p+m)\times(n+p+m+1)/2$ independent elements. Therefore at least $(n+p+m)\times(n+p+m+1)/2$ data tuples are required before (2.63) can be solved using LS. Both batch LS and recursive LS methods can be used to perform policy evaluation step (2.63).

Remark 2.5. Policy iteration based adaptive optimal control schemes require a persistent excitation condition (PE) [28], [53], [54], [56] to ensure sufficient exploration of the state space. If the state almost converges to the desired position and becomes stationary, the PE is no longer satisfied. An exploratory signal consisting of sinusoids of varying frequencies can be added to the control input to ensure PE qualitatively.

2.6. Policy Iteration Using Input-output Measured Data to Solve the LQT Online

In this section, first it is shown how to construct the state of the augmented system and the value function in terms of the past input, output and reference trajectory sequences. Then, a Bellman equation is defined that allows us to use PI to solve the LQT problem online, without requiring knowledge of the system dynamics and the reference trajectory dynamics, only by measuring past input, output, and reference trajectory data.

2.6.1 Constructing the State of the Augmented System and Value Function in Terms of Available Measured Data

The augmented system (2.13) can be written on a time horizon $\left[k-N,k\right]$ as the expanded state equation [31]

$$X_k = T^N X_{k-N} + \begin{bmatrix} B_1 & TB_1 & T^2B_1 & \dots & T^{N-1}B_1 \end{bmatrix} \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ u_{k-3} \\ \vdots \\ u_{k-N} \end{bmatrix} \tag{2.65}$$

or equivalently

$$\begin{bmatrix} x_k \\ r_k \end{bmatrix} = \begin{bmatrix} A^N & 0 \\ 0 & F^N \end{bmatrix} \begin{bmatrix} x_{k-N} \\ r_{k-N} \end{bmatrix} + \begin{bmatrix} B & AB & A^2B & \dots & A^{N-1}B \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ u_{k-3} \\ \vdots \\ u_{k-N} \end{bmatrix} \tag{2.66}$$

Define $U_N = \begin{bmatrix} B & AB & A^2B & \dots & A^{N-1}B \end{bmatrix}$ and $\overline{u}_{k-1,k-N} = \begin{bmatrix} u_{k-1}^T & u_{k-2}^T & u_{k-3}^T & \dots & u_{k-N}^T \end{bmatrix}^T$, where $\overline{u}_{k-1,k-N}$ is input signals over the time interval $\left[k-N,k-1\right]$. Using these definitions and (2.1), the output can be written as follows

$$y_k = C\,x_k = C\,A^N x_{k-N} + C\,U_N \overline{u}_{k-1,k-N} \tag{2.67}$$

Then, the sequence of the output vector over the time interval $\left[k-N,k-1\right]$ is

$$\overline{y}_{k-1,k-N} = \begin{bmatrix} y_{k-1} \\ y_{k-2} \\ y_{k-3} \\ \vdots \\ y_{k-N} \end{bmatrix} = \begin{bmatrix} CA^{N-1} \\ CA^{N-2} \\ \vdots \\ CA \\ C \end{bmatrix} x_{k-N} + \begin{bmatrix} 0 & CB & CAB & \cdots & CA^{N-2}B \\ 0 & 0 & CB & \cdots & CA^{N-3}B \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & CB \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ u_{k-3} \\ \vdots \\ u_{k-N} \end{bmatrix} \tag{2.68}$$

or equivalently

$$\overline{y}_{k-1,k-N} = W_N x_{k-N} + D_N \overline{u}_{k-1,k-N} \tag{2.69}$$

with W_N being the observability matrix

$$W_N = \begin{bmatrix} CA^{N-1} \\ CA^{N-2} \\ \vdots \\ CA \\ C \end{bmatrix} \tag{2.70}$$

Since (A,C) is observable, there exists a *K*, the observability index, such that rank $(W_N) < n$ for $N < K$ and that rank $(W_N) = n$ for $N \geq K$. Note that K satisfies $K\,p \geq n$.

Substituting x_{k-N} from (2.69) into (2.66) yields

$$\begin{bmatrix} x_k \\ r_k \end{bmatrix} = \begin{bmatrix} A^N & 0 \\ 0 & F^N \end{bmatrix} \begin{bmatrix} W_N^+ & \overline{y}_{k-1,k-N} - D_N \overline{u}_{k-1,k-N} \\ & r_{k-N} \end{bmatrix} + \begin{bmatrix} U_N \\ 0 \end{bmatrix} \overline{u}_{k-1,k-N} \tag{2.71}$$

where $W_N^+ = (W_N^T W_N)^{-1} W_N^T$. Then, (2.71) becomes

$$\begin{aligned} \begin{bmatrix} x_k \\ r_k \end{bmatrix} &= \begin{bmatrix} A^N W_N^+ & 0 \\ 0 & F^N \end{bmatrix} \begin{bmatrix} \overline{y}_{k-1,k-N} \\ r_{k-N} \end{bmatrix} + \begin{bmatrix} U_N - A^N W_N^+ D_N \\ 0 \end{bmatrix} \overline{u}_{k-1,k-N} \\ &= \begin{bmatrix} U_N - A^N W_N^+ D_N & A^N W_N^+ & 0 \\ 0 & 0 & F^N \end{bmatrix} \begin{bmatrix} \overline{u}_{k-1,k-N} \\ \overline{y}_{k-1,k-N} \\ r_{k-N} \end{bmatrix} \end{aligned} \tag{2.72}$$

Define

$$M = \begin{bmatrix} U_N - A^N W_N^+ D_N & A^N W_N^+ & 0 \\ 0 & 0 & F^N \end{bmatrix} \tag{2.73}$$

and

$$\bar{Z}_k = \begin{bmatrix} \bar{u}_{k-1,k-N} \\ \bar{y}_{k-1,k-N} \\ r_{k-N} \end{bmatrix} \tag{2.74}$$

Then, using the definition of the augmented state in (2.14) and definitions (2.73) and (2.74), (2.72) becomes

$$X_k = \begin{bmatrix} x_k \\ r_k \end{bmatrix} = M \bar{Z}_k \tag{2.75}$$

Substituting (2.75) into (2.19), yields

$$V(x_k, r_k) = \frac{1}{2} X_k^T P X_k = \frac{1}{2} \bar{Z}_k^T M^T P M \bar{Z}_k = \frac{1}{2} \bar{Z}_k^T \bar{P} \bar{Z}_k \tag{2.76}$$

Eq. (2.76) expresses that the value function at time *k* has a quadratic form in terms of the past inputs, outputs and reference trajectory. Note that the inner kernel matrix $\bar{P}$ depends on the system dynamics (A, B, C) and the reference trajectory dynamics (F).

2.6.2. Bellman Equation in Terms of Available Measured Data

In this subsection, a LQT Bellman equation in terms of measured data is derived to evaluate a fixed control policy.

Using (2.76) in the LQT Bellman (2.30) yields

$$\bar{Z}_k^T \bar{P} \bar{Z}_k = (y_k - r_k)^T Q (y_k - r_k) + u_k^T R u_k + \gamma \bar{Z}_{k+1}^T \bar{P} \bar{Z}_{k+1} \tag{2.77}$$

where (2.77) is the LQT Bellman equation in terms of measured data. Using this LQT Bellman equation for evaluation of the value of the current policy is the first key concept in developing our model-free RL algorithms.

Based on (2.77), define the LQT Hamiltonian function in terms of observed data as

$$H(\overline{Z}_k, u_k) = (y_k - r_k)^T Q(y_k - r_k) + u_k^T R u_k + \gamma \overline{Z}_{k+1}^T \overline{P} \overline{Z}_{k+1} - \overline{Z}_k^T \overline{P} \overline{Z}_k \tag{2.78}$$

$\overline{Z}_{k+1}^T \overline{P}\, \overline{Z}_{k+1}$ can be rewritten as a following form

$$\overline{Z}_{k+1}^T \overline{P}\, \overline{Z}_{k+1} = \begin{bmatrix} u_k \\ \overline{u}_{k-1,k-N+1} \\ \overline{y}_{k,k-N+1} \\ r_{k-N+1} \end{bmatrix}^T \begin{bmatrix} p_o & p_u & p_y & p_r \\ p_u^T & P_{22} & P_{23} & P_{24} \\ p_y^T & P_{32} & P_{33} & P_{34} \\ p_r^T & P_{42} & P_{43} & P_{44} \end{bmatrix} \begin{bmatrix} u_k \\ \overline{u}_{k-1,k-N+1} \\ \overline{y}_{k,k-N+1} \\ r_{k-N+1} \end{bmatrix} \tag{2.79}$$

Then, putting (2.79) in (2.78) and using the stationarity condition $dH/du = 0$, one has

$$\begin{aligned} u_k &= -\gamma(R + \gamma p_0)^{-1}(p_u \overline{u}_{k-1,k-N+1} + p_y \overline{y}_{k,k-N+1} + p_r r_{k-N+1}) \\ &= -K \times \begin{bmatrix} (\overline{u}_{k-1,k-N+1})^T & (\overline{y}_{k,k-N+1})^T & (r_{k-N+1})^T \end{bmatrix}^T \end{aligned} \tag{2.80}$$

This gives the control input u_k in terms of previous controls, outputs, and reference trajectory values. It is a dynamic regulator in terms of current and previous observed data that is equivalent to the state feedback control (2.35).

Note that for any vector $a \in \mathbb{R}^{n_a}$, $b \in \mathbb{R}^{n_b}$, and matrix $W \in \mathbb{R}^{n_a \times n_b}$, one has

$$a^T W b = (b^T \otimes a^T) vec(W) \tag{2.81}$$

where $\otimes$ is Kronecker product and $vec(W)$ is the vector formed by stacking the columns of matrix W.

Using (2.81), the LQT Bellman equation (2.77) can be written as

$$\phi_k(vec(\overline{P})) = (y_k - r_k)^T Q(y_k - r_k) + u_k^T R u_k + \gamma \phi_{k+1}(vec(\overline{P})) \tag{2.82}$$

where

$$\phi_k = \overline{Z}_k^T \otimes \overline{Z}_k^T \tag{2.83}$$

In the next section, it is shown how to use ADP to learn the solution to the LQT problem by learning online the kernel matrix $\overline{P}$ in the Bellman equation (2.82) for a control policy and finding an improved control policy using the update law (2.80), without knowing *A*, *B*, *C* and *F*.

The state of the system is not needed and only the measured input, output and reference data are required.

2.6.3. RL to solve the LQT problem using measured data

In this section, we develop PI algorithm to solve the LQT problem online in real time using only available measured data. They use the Bellman equation in form of (2.82) in the policy evaluation step and a policy update law in form of (2.80) in the policy improvement step. This is online iterative algorithm that converge to the solution to the LQT problem using only measured data along the system trajectories. The PI algorithm is as follows.

Algorithm 2.4. Policy iteration using measured data

Initialization: Start with an admissible control policy u_k^0.

1. Policy evaluation: Solve for $\bar{P}^{j+1}$ such that

$$\phi_k(vec(\bar{P})^{j+1}) = (y_k - r_k)^T Q (y_k - r_k) + (u_k^j)^T R (u_k^j) + \gamma \phi_{k+1}(vec(\bar{P})^{j+1}) \tag{2.84}$$

2. Policy improvement:

$$u_k^{j+1} = -\gamma (R + \gamma p_0^{j+1})^{-1} \times (p_u^{j+1} \overline{u}_{k-1,k-N+1} + p_y^{j+1} \overline{y}_{k,k-N+1} + p_r^{j+1} r_{k-N+1}) \tag{2.85}$$

The Bellman equation (2.84) can be solved online using LS using the measured data sets $\bar{Z}_k$, $\bar{Z}_{k+1}$ and ρ_k at each step which $\rho_k = (y_k - r_k)^T Q (y_k - r_k) + (u_k^j)^T R u_k^j$. In fact (2.84) is a scalar equation and $\bar{P}$ is a symmetric $(Nm + Np + p) \times (Nm + Np + p)$ matrix with $(Nm + Np + p) \times (Nm + Np + p + 1) / 2$ independent element. Therefore at least $L = (Nm + Np + p) \times (Nm + Np + p + 1) / 2$ data sets are required before (2.84) can be solved using LS. Assume that we collect $s \geq L$ number of samples and form the matrices

$$H_k = \begin{bmatrix} \phi_k \\ \phi_{k+1} \\ \vdots \\ \phi_{k+s-1} \end{bmatrix} \tag{2.86}$$

and

$$\psi_k = \begin{bmatrix} (y_k - r_k)^T Q (y_k - r_k) + (u_k^j)^T R (u_k^j) \\ (y_{k+1} - r_{k+1})^T Q (y_{k+1} - r_{k+1}) + (u_{k+1}^j)^T R (u_{k+1}^j) \\ \vdots \\ (y_{k+s-1} - r_{k+s-1})^T Q (y_{k+s-1} - r_{k+s-1}) + (u_{k+s-1}^j)^T R (u_{k+s-1}^j) \end{bmatrix} \tag{2.87}$$

Then based on (2.84), we have

$$\Delta H_k (vec(\bar{P})^{j+1}) = \psi_k \tag{2.88}$$

where $\Delta H_k = H_k - \gamma H_{k+1}$. The least-squares solution for $(\bar{P})^{j+1}$ in (2.84) becomes

$$vec(\bar{P})^{j+1} = (\Delta H_k^T \Delta H_k)^{-1} \Delta H_k^T \psi_k \tag{2.89}$$

The following Lemma is required to prove the convergence of Algorithm 2.4.

Lemma 2.2. Consider a fixed control policy

$$u_k = -K_1 X_k = -\bar{K}_1 \bar{Z}_k \tag{2.90}$$

where $\bar{K}_1 = K_1 M$. Assume that the value function for the policy $u_k = -K_1 X_k$ is $V_k = \frac{1}{2} X_k^T P X_k$ and is found by solving the Lyapunov equation (2.49) for P. Also, assume that the value function for the policy $u_k = -\bar{K}_1 \bar{Z}_k$ is $\bar{V}_k = \frac{1}{2} \bar{Z}_k^T \bar{P} \bar{Z}_k$ and is found by solving the least squares equation (2.89) for $\bar{P}$. Then, $V_k = \bar{V}_k$ provided that the matrix H_k in (2.86) is full rank.

Proof. By using (2.75), one has $V_k = \frac{1}{2} X_k^T P X_k = \frac{1}{2} \bar{Z}_k^T M^T P M \bar{Z}_k$. Therefore, $V_k = \bar{V}_k$ if and only if

$$\bar{P} = M^T P M \tag{2.91}$$

Thus, in order to prove that the Lyapunov function (2.49) and the least squares equation (2.89) give the same value function for the given policy in (2.90), we need to show that $\bar{P}$ found by solving the least squares equation (2.89) is equal to $M^T P M$ where P is the solution to the Lyapunov equation (2.49).

Assume that there exists a matrix P_1 such that

$$\bar{P} = M^T P_1 M \tag{2.92}$$

To prove that $V_k = \bar{V}_k$ it remains to show that P_1 satisfies the Lyapunov equation (2.49), provided that H_k in (2.86) is full rank. That is, to show that P_1 in (2.92) is equal to P in (2.91). To this end, using (2.13), (2.75) and (2.90), one has

$$X_{k+1} = (T - B_1 K_1) X_k = (T - B_1 K_1) M \bar{Z}_k = M \bar{Z}_{k+1} \tag{2.93}$$

Therefore,

$$\bar{Z}_{k+1} = M^+ (T - B_1 K_1) M \bar{Z}_k \tag{2.94}$$

where $M^+ = M^T (M M^T)^{-1}$ is the generalized right inverse of M with $M^+ M = I$.

Using (2.75), (2.90), (2.92) and (2.94) in (2.77), we get

$$\bar{Z}_k^T M^T [Q_1 - P_1 + K_1^T R K_1 + \gamma (T - B_1 K_1)^T P_1 (T - B_1 K_1)] M \bar{Z}_k = 0 \tag{2.95}$$

where Q_1 is defined in (2.17). After collecting s equations to perform the least squares and using (2.81) ,we end up with

$$H_k \, vec(M^T [Q_1 - P_1 + K_1^T R K_1 + \gamma (T - B_1 K_1)^T P_1 (T - B_1 K_1)] M) = 0 \tag{2.96}$$

Equation (2.96) is equivalent to the least squares equation (2.89). Since M and H_k are full rank, equation (2.96) is satisfied if and only if $Q_1 - P_1 + K_1^T R K_1 + \gamma (T - B_1 K_1)^T P_1 (T - B_1 K_1) = 0$. That is, if P_1 satisfies the Lyapunov equation (2.49). This completes the proof. ∎

Theorem 2.3. Convergence proof for Algorithm 2.4.

Consider the sequence $\bar{P}^j$ and u^j in (2.84) and (2.85). Then, if the rank condition of Lemma 2.2 is satisfied, as $j \to \infty$, $\bar{P}^j \to \bar{P}^*$ and $u^j \to u^*$.

Proof. It was shown in Lemma 2.2 that for a fixed control policy, the value function found by the Bellman equation (2.84) in Algorithm 2.4 is equal to the value function found by the Lyapunov equation (2.49). Therefore, the policy evaluation steps of Algorithms 2.1 and 2.4 give the same solution for the value function. Moreover, the policy improvement steps (2.50) and (2.85) in Algorithms 2.1 and 2.4 are found by minimizing the value function and since the value functions for both algorithms are the same, both policy improvement steps (2.50) and (2.85) in Algorithms 2.1 and 2.4 lead to the same results and thus Algorithms 2.1 and 2.4 have the same convergence properties. Convergence of Algorithm 2.1 is shown in [89]. This completes the proof. ∎

Remark 2.6. PI Algorithms 2.4 is implemented online in real time using the past input, output and reference data measured along the system trajectories without requiring any knowledge of the augmented system dynamics.

Remark 2.7. In the policy evaluation step of the policy iteration algorithm, it is required to solve the Lyapunov equation (2.84) at each step. This requires a stabilizing gain u^j at each step. This is called a full back up in reinforcement learning terms [8], [13]. On the other hand, in the policy evaluation step of value iteration algorithm it is required to solve the Lyapunov recursion (2.97) at each step, which is very easy to compute, and does not require a stabilizing gain. This is called a partial backup in reinforcement learning.

Remark 2.8. An initial admissible policy is required in the proposed policy iteration Algorithms 2.1 and 2.4. In many cases the system is itself stable and the initial policy can be chosen as $u = 0$ Therefore, the admissibility of the initial policy is guaranteed without requiring any knowledge of T. If the system itself is not stable, one can obtain the initial admissible policy by using some

knowledge of T. Suppose the system (2.13) has a nominal model T_N satisfying $T = T_N + \Delta T$, where ΔT is unknown part of T. In this case, an admissible initial policy can be obtained by using robust control methods such as H_∞ control with the nominal model T_N. Note that the learning process does not require any knowledge of T.

2.7. Simulation Results

In this section, simulation examples are carried out to illustrate the design procedures and verify the effectiveness of the proposed schemes.

A linear system is considered as

$$\begin{aligned} x_{k+1} &= \begin{bmatrix} -1 & 2 \\ 2.2 & 1.7 \end{bmatrix} x_k + \begin{bmatrix} 2 \\ 1.6 \end{bmatrix} u_k \\ y_k &= \begin{bmatrix} 1 & 2 \end{bmatrix} x_k \end{aligned} \tag{2.97}$$

The open-loop poles are $z_1 = -2.1445$ and $z_2 = 2.8445$, so the system is unstable. The performance index is considered as (2.15) with $Q = 6$, $R = 1$ and $\gamma = 0.8$. It is supposed that the sinusoid reference trajectory is generated by the command generator dynamics given by

$$r_{k+1} = -r_k \tag{2.98}$$

2.7.1. Policy iteration using value function

In this subsection Algorithms 2.1 and 2.2, which use value function structure (2.19) to evaluate the performance of a policy, are applied for the system (2.97) and the reference trajectory (2.98).

The optimal matrix *P* satisfying the ARE (2.36) for this problem is

$$P^* = \begin{bmatrix} 133.3840 & 16.0531 & 31.1402 \\ 16.0531 & 25.1604 & -10.8271 \\ 31.1402 & -10.8271 & 18.4825 \end{bmatrix} \tag{2.99}$$

First, offline policy iteration Algorithm 2.1 is implemented as in (2.50) and (2.51). Fig. 2.1 shows that the *P* matrix parameters converge to their optimal values. After 12 iterations the *P* matrix parameters converge to

$$P = \begin{bmatrix} 133.3840 & 16.0531 & 31.1402 \\ 16.0531 & 25.1604 & -10.8271 \\ 31.1402 & -10.8271 & 18.4825 \end{bmatrix} \tag{2.100}$$

The results of applying the optimal control given by substituting the *P* matrix (2.100) in (2.34), (2.35) to the system (2.97) are now presented. Fig. 2.2 shows that the output y_k tracks the reference trajectory r_k and guarantees the stability for the offline policy iteration Algorithm 2.1 is presented. The optimal control signal input is shown in Fig. 2.3.

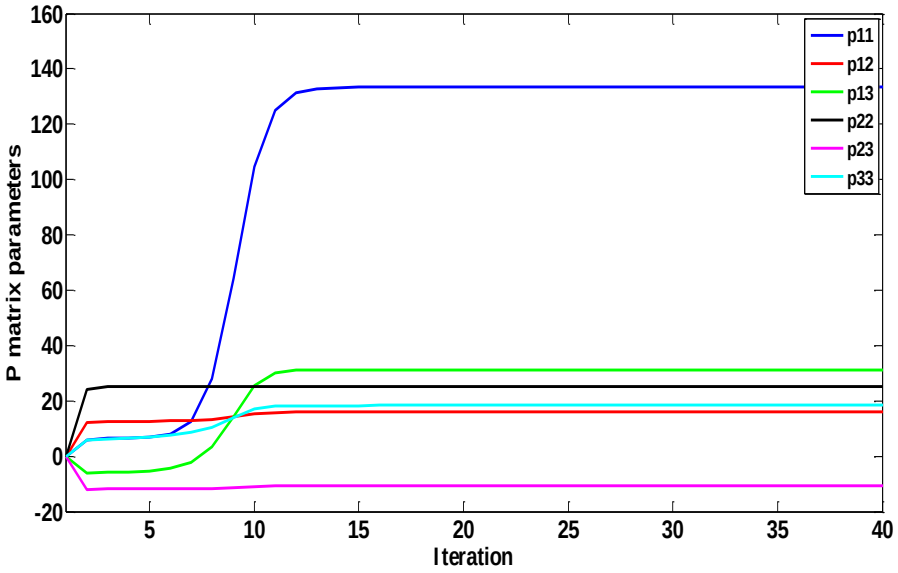

Fig. 2.1. Convergence of the P matrix parameters to their optimal values for offline PI Algorithm 2.1

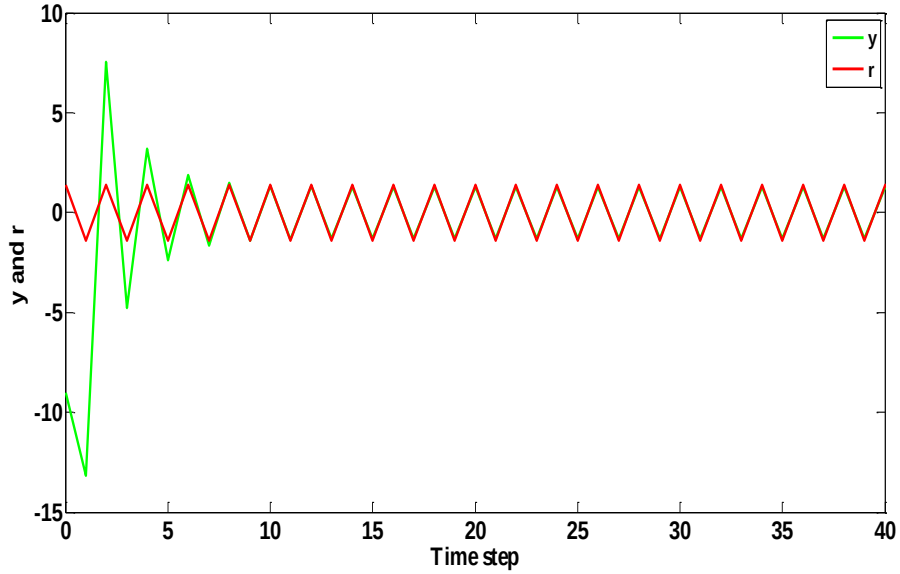

Fig. 2.2. Evaluation of the output and the reference trajectory for offline PI Algorithm 2.1

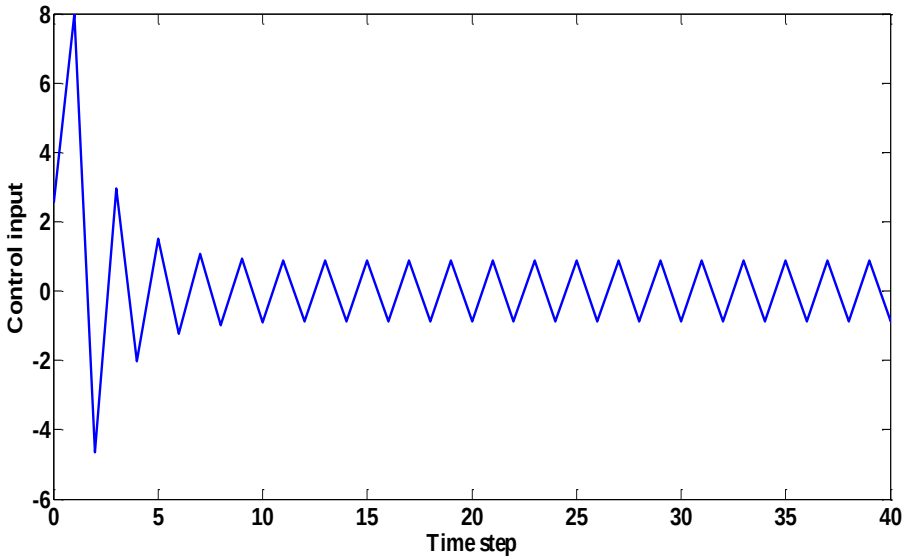

Fig. 2.3. The control input during learning

Next, the online policy iteration Algorithm 2.2 is implemented as in (2.52) and (2.53). PE was ensured by adding a probing noise to the control input. Fig. 2.4 shows how the P matrix parameters converge to their optimal values. After 20 iterations the P matrix parameters converge to

$$P = \begin{bmatrix} 133.3840 & 16.0531 & 31.1402 \\ 16.0531 & 25.1604 & -10.8271 \\ 31.1402 & -10.8271 & 18.4825 \end{bmatrix}$$

Comparing this *P* matrix with the P^* matrix, it is seen that the online Algorithm 2.2 converges very close to the optimal controller.

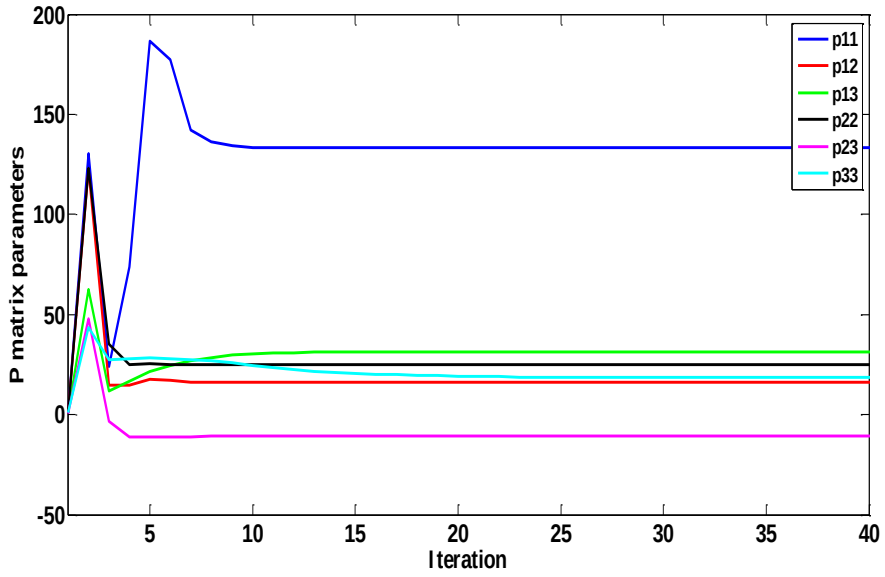

Fig. 2.4. Convergence of the P matrix parameters to their optimal values for online PI Algorithm 2.2

2.7.2. Policy iteration using Q-function

The policy iteration Algorithm 2.3, which uses the *Q*-function to evaluate a policy, is implemented as in (2.63) and (2.64).

The optimal *P* matrix for this problem by solving the ARE (2.36) is (2.99). Using this *P* matrix and considering the definition of *Q*-function in (2.55) and (2.56) and definitions of B_1 and *T* in (2.13), the optimal *H* function becomes

$$H^* = \begin{bmatrix} 153.6214 & -91.4595 & 37.9679 & -106.6035 \\ -91.4595 & 596.3286 & -47.0995 & 566.3383 \\ 37.9679 & -47.0995 & 20.7860 & -35.9657 \\ -106.6035 & 566.3383 & -35.9657 & 561.5493 \end{bmatrix}$$

Consequently, using (2.58) for the optimal control $u_k^* = -K^* X_k$, the control gain K^* is given as

$$K^* = \begin{bmatrix} -0.1898 & 1.0085 & -0.0640 \end{bmatrix}$$

Now, Algorithm 2.3 is used to solve the problem. It is assumed the dynamics *T* and $\boldsymbol{B}_1$ are completely unknown. For the purpose of demonstrating the algorithm, the initial state of the augmented system is chosen as $X_0 = [5 \quad -5 \quad 5]^T$ and initial control input is chosen as $K_0 = [0.3 \quad 1.3 \quad 0.75]$. In each iteration, 21 data samples are collected to perform the LS. PE was ensured by adding a probing noise to the control input. Fig. 2.5 and Fig. 2.6. show norm of the difference of the optimal and the computed *H* matrices as well as norm of the difference between the optimal control gain and the computed gain, respectively. After 6 iterations the *H* matrix parameters and the control gain converge to

$$H = \begin{bmatrix} 153.6214 & -91.4595 & 37.9681 & -106.6935 \\ -91.4595 & 596.3286 & -47.1000 & 566.3383 \\ 37.9681 & -47.1000 & 19.0389 & -35.9662 \\ -106.6935 & 566.3383 & -35.9662 & 561.5493 \end{bmatrix}$$

and

$$K = \begin{bmatrix} -0.1898 & 1.0085 & -0.0640 \end{bmatrix}$$

Fig. 2.7. shows the output of the system and the reference trajectory during learning process. Fig. 2.8. shows the probing noise injected to the control input during learning process. It is clear that after 300 time step the PE condition is no longer needed. Therefore, probing noise is turned off. Thereafter, the output of the system is very close to the reference trajectory as it is required.

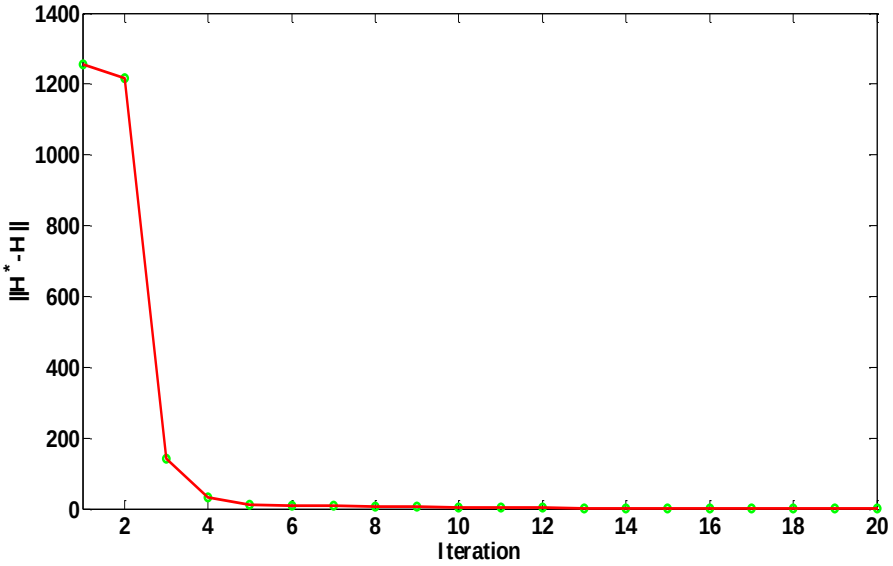

Fig. 2.5. Convergence of *H* matrix to its optimal values H^* during the learning process

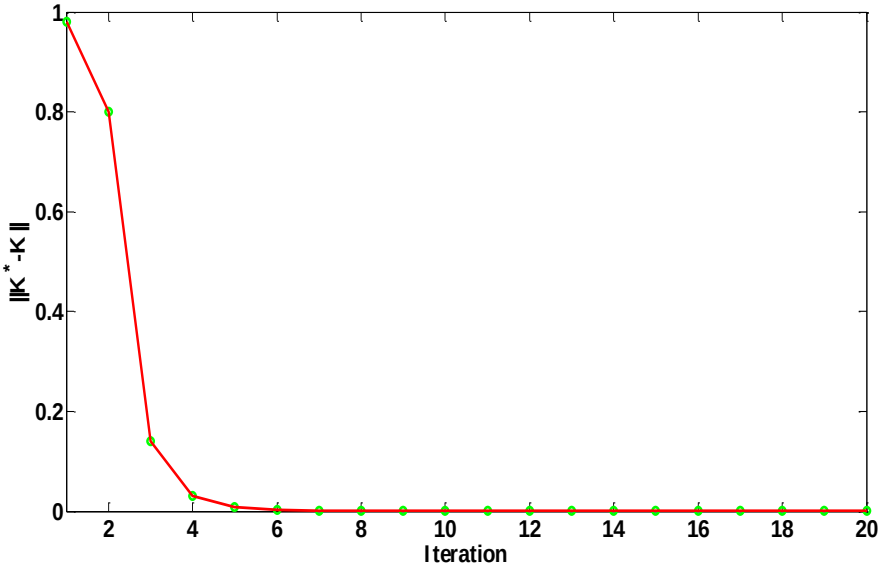

Fig. 2.6. Convergence of *K* matrix to its optimal values K^* during the learning process

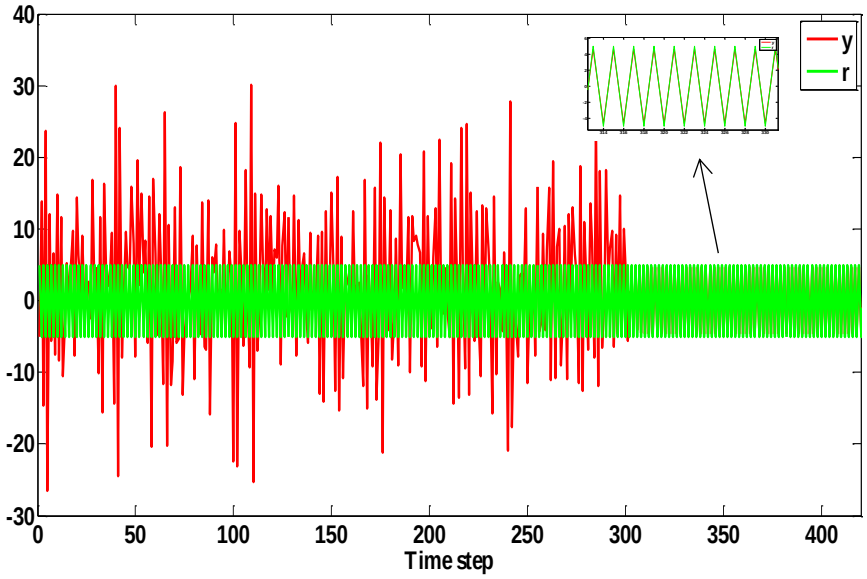

Fig. 2.7. Evaluation of the output and the reference trajectory during the learning process

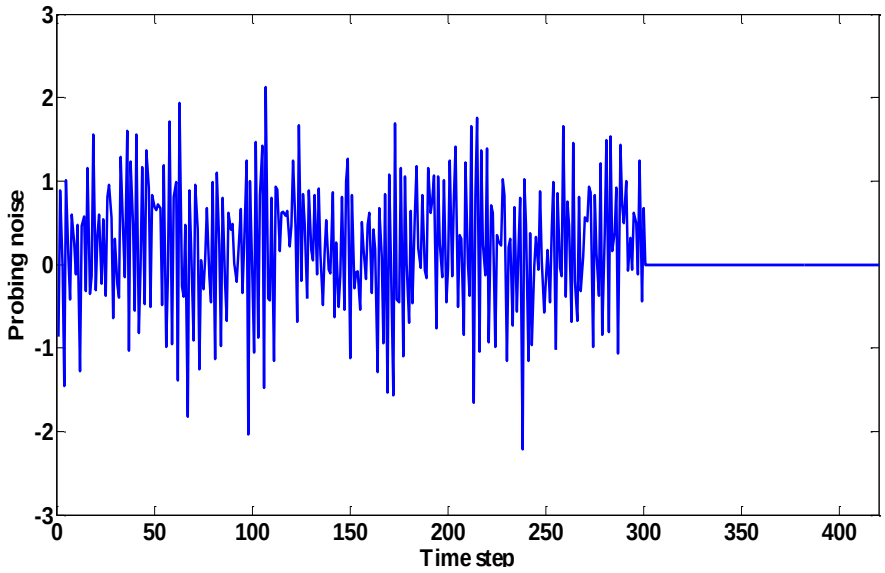

Fig. 2.8. Probing noise during the learning process

2.7.3. Policy iteration using input-output measured data

In this subsection policy iteration Algorithm 2.4 is used to solve the problem. It is assumed the dynamics T and B_1 are completely unknown. The value of *N* should be bigger than 2 to make sure W_N is a full rank matrix. Here, we select $N = 2$, then $\bar{P} \in \Re^{5\times5}$. In each iteration, 25 data samples are collected to perform the LS. PE was ensured by adding a probing noise to the control input. By using $\bar{P} = M^T P M$ and (2.99), the optimal $\bar{P}$ matrix is

$$\bar{P}^* = \begin{bmatrix} 700 & -8210 & 1750 & -4250 & 40 \\ -8210 & 140010 & -27840 & 72380 & -1300 \\ 1750 & -27840 & 5590 & -14390 & 240 \\ -4250 & 72380 & -14390 & 37420 & -670 \\ 40 & -1300 & 240 & -670 & 20 \end{bmatrix}$$

and the optimal control gain is

$$K^* = \begin{bmatrix} 11.7006 & -2.4874 & 6.0486 & -0.0640 \end{bmatrix}$$

Fig. 2.9. and Fig. 2.10 show norm of the difference of the optimal and the computed $\bar{P}$ matrices as well as norm of the difference between the optimal control gain and the computed gain, respectively. After 13 iterations the $\bar{P}$ matrix parameters and the control gain converge to

$$\bar{P} = \begin{bmatrix} 680 & -7810 & 1660 & -4150 & 45 \\ -7810 & 136610 & -27160 & 70530 & -1370 \\ 1660 & -27160 & 5450 & -14020 & 260 \\ -4150 & 70530 & -14020 & 36420 & -710 \\ 45 & -1370 & 260 & -710 & 30 \end{bmatrix}$$

and

$$K = \begin{bmatrix} 11.4642 & -2.4367 & 6.0917 & -0.0661 \end{bmatrix}$$

Fig. 2.11. shows the output of the system and the reference trajectory. It is clear that the output of the system is very close to the reference trajectory as it is required.

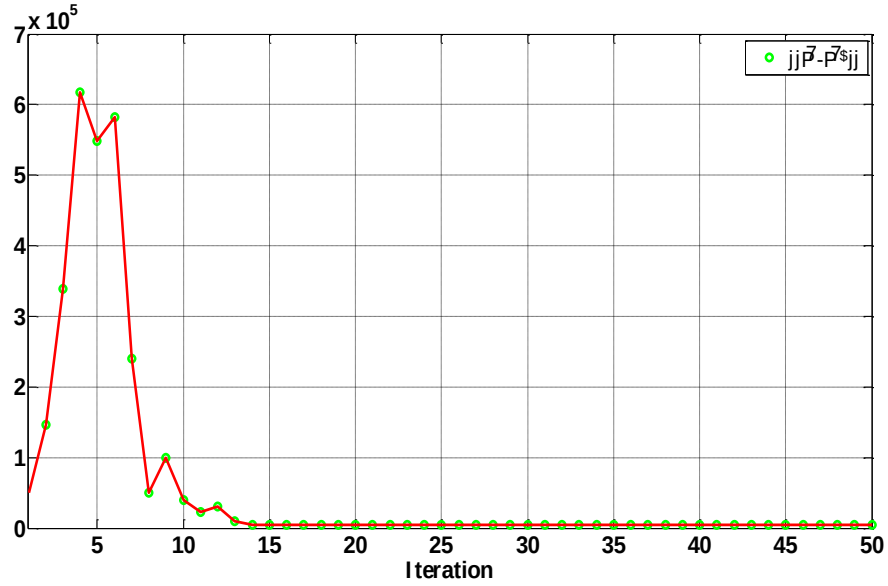

Fig. 2.9. Convergence of $\bar{P}$ matrix to its optimal values $\bar{P}^*$ during the learning process of policy iteration Algorithm 2.4

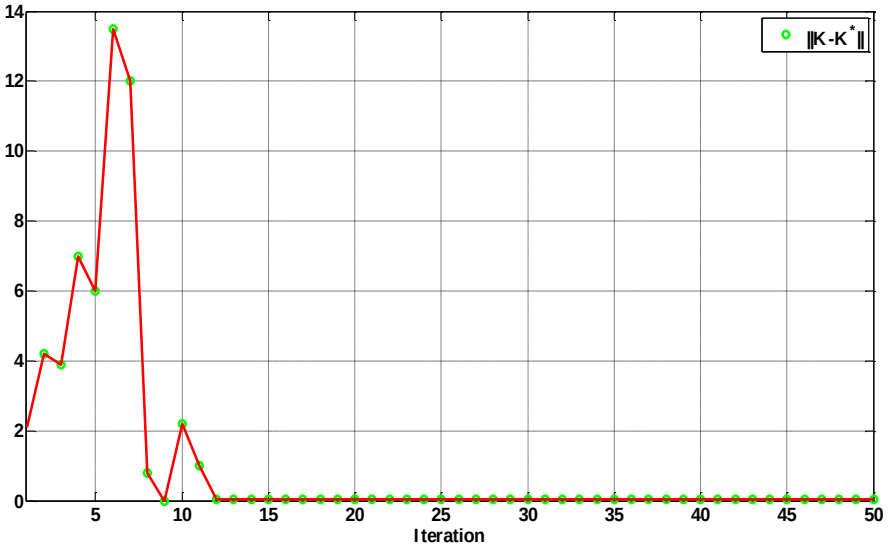

Fig. 2.10. Convergence of K matrix to its optimal values K^* during the learning process of policy iteration Algorithm 2.4

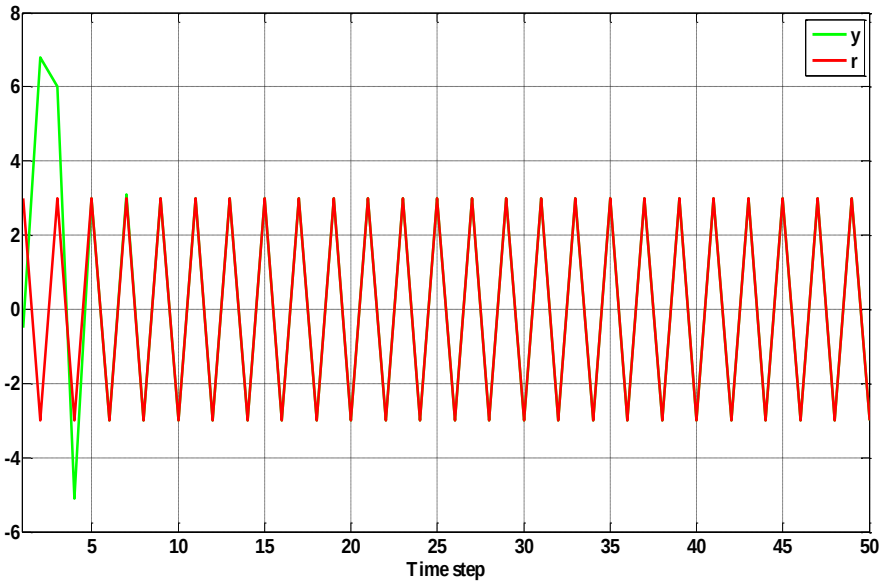

Fig. 2.11 Output and Reference trajectory

2.8. Conclusion

An alternative solution to the LQT problem using RL was presented. It was shown that the value function has a quadratic form in terms of the state of the augmented system. On the basis of this value function, a LQT ARE was obtained and Q-learning and policy iteration algorithms were developed to solve the LQT ARE online without requiring knowledge of the system dynamics and using only the measured states, input, output, and reference trajectory data. The simulation results showed that the proposed formulation for the LQT problem gave good tracking performance.

Chapter 3

ACTOR-CRITIC-BASED OPTIMAL TRACKING CONTROL FOR PARTIALLY-UNKNOWN NONLINEAR CONSTRAINED-INPUT SYSTEMS

3.1. Introduction

This chapter presents a partially model-free adaptive optimal control solution to the nonlinear discrete-time (DT) tracking control problem in the presence of input constraints. The tracking error dynamics and reference trajectory dynamics are first combined to form an augmented system. Then, a new discounted performance function based on the augmented system is presented for the optimal nonlinear tracking problem. In contrast to the standard solution, which finds the feedforward and feedback terms of the control input separately, the minimization of the proposed discounted performance function gives both feedback and feedforward parts of the control input simultaneously. This enables us to encode the input constraints into the optimization problem by using a nonquadratic performance function. The DT tracking Bellman equation and tracking Hamilton-Jacobi-Bellman (HJB) are derived. An actor-critic based reinforcement learning algorithm is used to learn the solution to the tracking HJB equation online without requiring knowledge of the system drift dynamics. That is, two neural networks (NNs), namely actor NN and critic NN, are tuned online and simultaneously to generate the optimal bounded control policy.

The rest of the paper is organized as follows. The next section formulates the optimal tracking control problem and discusses its standard solution and its shortcomings. In the Section 3.3, the new formulation to the optimal tracking problem of deterministic nonlinear DT constrained-input systems is presented. Also, a DT tracking HJB equation is obtained which gives both feedback and feedforward parts of the control input simultaneously. An actor-critic based controller is given in Section 3.4 which learns the solution to the DT tracking HJB online and without requiring knowledge of the drift system dynamics or the reference trajectory dynamics. Finally, the simulation results are presented in Section 3.5 to confirm the suitability of the

proposed method.

3.2. Optimal Tracking Control for Nonlinear Systems

In this section, we formulate the nonlinear optimal tracking problem and give the standard solution. The standard solution has several deficiencies. We fix these in the next sections where we give our results.

Consider the deterministic nonlinear affine discrete-time system given by

$$x(k+1) = f(x(k)) + g(x(k))u(k) \tag{3.1}$$

where $k > 0$, $x(k) \in \mathbb{R}^n$ represents the state vector of the system, $u(k) \in \mathbb{R}^m$ represents the control vector, $f(x(k)) \in \mathbb{R}^n$ is the drift dynamics of the system and $g(x(k)) \in \mathbb{R}^{n \times m}$ is the input dynamics of the system. Assume that $f(0) = 0$ and $f(x) + g(x)u$ is Lipschitz continuous on a compact set Ω which contains the origin and the system (3.1) is controllable in the sense that there exists a continuous control on Ω which stabilizes the system.

For the infinite-horizon tracking problem, the goal is to design an optimal controller for the system (3.1) which ensures that the state $x(k)$ tracks the reference trajectory $r(k)$ in an optimal manner. Define the tracking error as

$$e(k) = x(k) - r(k) \tag{3.2}$$

For the standard solution to the tracking problem, the control input consists of two parts, including a feedforward part and a feedback part [2]. In the following, it is discussed how each of these parts are obtained using the standard method.

The feedforward or steady-state part of the control input is used to assure perfect tracking. The perfect tracking is achieved if $x(k) = r(k)$. In order for this to happen, a feedforward control input $u(k) = u_d(k)$ must exist to make $x(k)$ equal to $r(k)$. By substituting $x(k) = r(k)$ and $u(k) = u_d(k)$ in the system dynamics (3.1), one has

$$r(k+1) = f(r(k)) + g(r(k))\, u_d(k) \tag{3.3}$$

If the system dynamics is known, $u_d(k)$ is obtained by

$$u_d(k) = g(r(k))^{+} \left(r(k+1) - f(r(k))\right) \tag{3.4}$$

where $g(r(k))^{+} = (g(r(k))^{T} g(r(k)))^{-1} g(r(k))^{T}$ is the generalized invers of $g(r(k))$ with $g(r(k))^{+} g(r(k)) = I$.

Remark 3.1. Note that equation (3.4) cannot be solved for $u_d(k)$ unless the reference trajectory is given a priori. Moreover, finding $u_d(k)$ requires the inverse of the input dynamics $g(r(k))$ and the function $f(r(k))$ be known. We shall propose a new method in Section 3.3 that does not need the reference trajectory to be given or the input dynamics to be invertible.

The feedback control input is computed as follows. By using (3.1) and (3.3), the tracking error dynamics can be expressed in terms of the tracking error $e(k)$ and the reference trajectory $r(k)$ as

$$\begin{aligned} e(k+1) &= x(k+1) - r(k+1) = f(e(k) + r(k)) + g(e(k) + r(k))u_e(k) + \\ & g(e(k) + r(k))g^{-1}(r(k))(r(k+1) - f(r(k))) - r(k+1) \end{aligned} \tag{3.5}$$

By defining

$$\begin{aligned} g_e(k) &\triangleq g(e(k) + r(k)) \\ f_e(k) &\triangleq f(e(k) + r(k)) - g(e(k) + r(k))g^{-1}(r(k))(r(k+1) - f(r(k))) - r(k+1) \end{aligned} \tag{3.6}$$

Eq. (3.5) can be rewritten as

$$e(k+1) = f_e(k) + g_e(k)u_e(k) \tag{3.7}$$

where $u_e(k)$ is the feedback control input that is designed to stabilize the tracking error dynamics in an optimal manner by minimizing the following performance function

$$J(e(k),u_e(k))=\sum_{i=k}^{\infty}\Big[e^T(i)Q_e\,e(i)+u_e^T(i)R_e\,u_e(i)\Big] \tag{3.8}$$

where $Q_e \in \mathbb{R}^{n\times n}$ and $R_e \in \mathbb{R}^{m\times m}$ are symmetric positive definite. The feedback input that minimizes (3.8) is found by solving the stationary condition $\partial J(e,u_e)\big/\partial u_e = 0$ [2]. The result is

$$u_e^*(k)=-\frac{1}{2}R_e^{-1}g_e^T(k)\frac{\partial J(e(k+1))}{\partial e(k+1)} \tag{3.9}$$

Then the standard control input is given by

$$u^*(k)=u_d(k)+u_e^*(k) \tag{3.10}$$

where u_d is given by (3.4) and u_e^* is given by (3.9).

Remark 3.2. The feedback part of the control input (3.9) is designed to stabilize the tracking error dynamics. The RL algorithm for finding the feedback part could be implemented by measuring the state $x(k)$ and the reference trajectory $r(k)$ to obtain $e(k)$ and without requiring the system dynamics [2]. In contrast, obtaining the feedforward part of the control input needs complete knowledge of the system dynamics and the reference trajectory dynamics.

Remark 3.3. Since the feedforward part of the control input (3.4) is found separately and does not involve in the optimization of the performance (3.8), it is not possible to encode the constraints of the control input u into the optimization problem. In the following, a new formulation of the problem is given that gives both feedback and feedforward parts of the control input simultaneously by minimizing a predefined performance function.

3.3. New Formulation for the Nonlinear Input Constraints Tracking Problem

A disadvantage to the standard tracking problem solution in Section II is that it needs complete knowledge of the system dynamics, and also the reference trajectory should satisfy (3.3) to compute the feedforward control input (3.4). Moreover, the standard solution does not

take into account the input constraints caused by physical limitations of the actuator. In this section, we propose an alternative approach for formulating the nonlinear tracking problem that gives both parts of the control input simultaneously and also takes into account the input constraints. First, an augmented system composed of the original system and the reference trajectory is constructed. Based on this augmented system a new performance function is presented that consists of both feedback and feedforward parts of the control input. Then, the Bellman and HJB equations for the nonlinear tracking problem are obtained.

3.3.1. Augmented system dynamics and discounted performance function

Before proceeding, the following assumption is made for the reference trajectory.

Assumption 3.1. The reference trajectory dynamics for the nonlinear tracking problem is generated by the command generator model

$$r(k+1) = \psi(r(k)) \tag{3.11}$$

where $\psi(r(k))$ is a C^{∞} function with $\psi(0) = 0$ and $r(k) \in \mathbb{R}^n$ [37].

Definition 3.1. A equilibrium point is said to be Lyapunov stable if for every $\varepsilon > 0$, there exists a $\delta = \delta(\varepsilon) > 0$ such that, if $\left\| x(0) - x_e \right\| < \delta$, then for every $t \geq 0$ we have $\left\| x(t) - x_e \right\| < \varepsilon$.

Remark 3.4. Note that Assumption 3.1 is a standard assumption made in accordance with other work on tracking control in the literature [91]. This command generator dynamics can generate a large class of command trajectories, including unit step, sinusoidal waveforms, damped sinusoids, and more. Human factor studies show that after learning a task, the skilled human operator behaves predictably like a command generator dynamics. Assume, the model (3.11) covers this case from human robot interaction.

The tracking error dynamics (3.5) in terms of the control input $u(k)$ is given by

$$e(k+1) = x(k+1) - r(k+1) = f(e(k) + r(k)) - \psi(r(k)) + g(e(k) + r(k))u(k) \tag{3.12}$$

Based on (3.11) and (3.12), the augmented system is constructed in terms of the tracking error $e(k)$ and the reference trajectory $r(k)$ as

$$\begin{bmatrix} e(k+1) \\ r(k+1) \end{bmatrix} = \begin{bmatrix} f(e(k)+r(k)) - \psi(r(k)) \\ \psi(r(k)) \end{bmatrix} + \begin{bmatrix} g(e(k)+r(k)) \\ 0 \end{bmatrix} u(k) \equiv F(X(k)) + G(X(k))u(k) \tag{3.13}$$

where the augmented state is

$$X(k) = \begin{bmatrix} e(k) \\ r(k) \end{bmatrix} \in \mathbb{R}^{2n} \tag{3.14}$$

and

$$u(k) \in \mathbb{R}^m \Big| \quad |\, u_j(k) \,| \leq \overline{u}_j,\ j = 1, \cdots, m \tag{3.15}$$

where $\overline{u}_j$ is the saturating bound for the j-th actuator.

Based on the augmented system (3.13), define the value function for the tracking problem as

$$V(X(k)) = \sum_{i=k}^{\infty} \gamma^{i-k} U_i = \sum_{i=k}^{\infty} \gamma^{i-k} \left[X(i)^T Q_1 X(i) + W(u(i)) \right] \tag{3.16}$$

where

$$Q_1 = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} \tag{3.17}$$

Here $0 < \gamma \leq 1$ is the discount factor and $W(u(i)) \in \mathbb{R}$ is used to penalize the control input. Q and $W(u(i))$ are positive definite.

For the unconstrained control problem, $W(u)$ may have the quadratic form $W(u(i)) = u(i)^T R\, u(i)$. However, to deal with the constrained control input [72]-[74], [92] [93], we employ a nonquadratic functional defined as

$$W(u(i)) = 2 \int_0^{u(i)} \varphi^{-T}(\bar{U}^{-1} s) \bar{U} R\, ds \tag{3.18}$$

where $\bar{U} \in \mathbb{R}^{m\times m}$ is the constant diagonal matrix given by $\bar{U} = diag\{\bar{u}_1, \bar{u}_2, ..., \bar{u}_m\}$ and R is positive definite and assumed to be diagonal. $\varphi(.)$ is a bounded one-to-one function satisfying $|\varphi(.)| < 1$. Moreover, it is a monotonic increasing odd function with its first derivative bounded by a constant *M*. It is clear that $W(u(i))$ in (3.18) is a scalar for $u(i) \in \mathbb{R}^m$. $W(u)$ is ensured to be positive definite because φ is monotonic odd function and R is positive definite. This function is written in terms of scalar components as follows,

$$\varphi^{-1}(\bar{U}^{-1}s) = \left[\varphi^{-1}(\frac{u_1(i)}{\bar{u}_1(i)}) \quad \varphi^{-1}(\frac{u_2(i)}{\bar{u}_2(i)}) \quad \dots \quad \varphi^{-1}(\frac{u_m(i)}{\bar{u}_m(i)})\right]^T \tag{3.19}$$

where $s \in \mathbb{R}^m$. Denote $\omega(s) = \varphi^{-T}(\bar{U}^{-1}s)\bar{U}R = \left[\omega_1(s_1) \dots \omega_m(s_m)\right]$. Then, the integral in (3.18) is defined in terms of the components of $u(i)$ as [41]

$$W(u(i)) = 2\int_0^{u(i)} \omega(s)\, ds = 2\sum_{j=1}^{m} \int_0^{u_j(i)} \omega_j(s_j)\, ds_j \tag{3.20}$$

Note that by defining the nonquadratic function $W(u(i))$ (3.18), the control input resulting by minimizing the value function (3.16) is bounded (see equation (3.29)). The function (3.16) has been used to generate bounded control for continuous-time systems in [72]-[74].

Remark 3.5. 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 performance function (3.16) becomes infinite without the discount factor as the control input contains a feedforward part which depends on the reference trajectory and thus $W(u)$ does not go to zero as time goes to infinity.

3.3.2 Bellman and HJB equations for the nonlinear tracking problem

In this subsection, based on the augmented system and the value function presented in the previous subsection, the Bellman and HJB equations for the nonlinear tracking problem are

given.

By using (3.16) and (3.18), we have

$$\begin{aligned} V(X(k)) = {} & X(k)^T Q_1 X(k) + 2\int_0^{u(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\,R\,ds \\ & + \sum_{i=k+1}^{\infty} \gamma^{i-k}\left[X(i)^T Q_1 X(i) + 2\int_0^{u(i)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\,R\,ds \right] \end{aligned} \tag{3.21}$$

which yields the nonlinear tracking Bellman equation

$$V(X(k)) = X(k)^T Q_1 X(k) + 2\int_0^{u(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\,R\,ds + \gamma V(X(k+1)) \tag{3.22}$$

Based on (3.22), define the Hamiltonian function

$$H(X(k),u(k),V) = X(k)^T Q_1 X(k) + 2\int_0^{u(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}R\,ds + \gamma V(X(k+1)) - V(X(k)) \tag{3.23}$$

From Bellman's optimality principle, it is well known that, for the infinite-horizon optimization case, the value function $V^*(X(k))$ is time-invariant and satisfies the DT HJB equation

$$V^*(X(k)) = \min_{u(k)}\left[X(k)^T Q_1 X(k) + 2\int_0^{u(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\,R\,ds + \gamma V^*(X(k+1)) \right] \tag{3.24}$$

Or equivalently,

$$H(X(k),u^*(k),V^*) = X(k)^T Q_1 X(k) + 2\int_0^{u^*(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\,R\,ds + \gamma V^*(X(k+1)) - V^*(X(k)) \tag{3.25}$$

Necessary condition for optimality is stationarity condition [2]

$$\begin{aligned} & \frac{\partial H(X(k),u(k),V^*)}{\partial u(k)} = \frac{\partial\left(X(k)^T Q_1 X(k) + 2\int_0^{u(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\,R\,ds \right)}{\partial u(k)} \\ & +\gamma\left(\frac{\partial X(k+1)}{\partial u(k)} \right)^T \frac{\partial V^*(X(k+1))}{\partial X(k+1)} = 0 \end{aligned} \tag{3.26}$$

By using (3.13), we have

$$\frac{\partial X(k+1)}{\partial u(k)} = G(X) \tag{3.27}$$

also,

$$\frac{\partial\left(X(k)^T Q_1\, X(k) + 2\int_0^{u(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\, R\, ds\right)}{\partial u(k)} = 2\varphi^{-T}(\bar{U}^{-1}u(k))\bar{U}\, R \tag{3.28}$$

By substituting (2.27) and (2.28) into (2.26), yields the optimal control input,

$$u^*(k) = \bar{U}\varphi\left(-\frac{\gamma}{2}(\bar{U}R)^{-T} G(X)^T \frac{\partial V^*(X(k+1))}{\partial X(k+1)}\right) \tag{3.29}$$

By substituting (2.29) in the Bellman equation, the DT tracking HJB equation (2.24) becomes

$$V^*(X(k)) = X(k)^T Q_1\, X(k) + W(u^*(k)) + \gamma V^*(X(k+1)) \tag{3.30}$$

where the $V^*(X(k))$ is the value function corresponding to the optimal control input.

In order to find the optimal control solution, first the DT tracking HJB equation (3.30) is solved and then the optimal control solution is given by (3.29) using the solution of the DT tracking HJB equation. However, in general, the DT tracking HJB equation cannot be solved analytically. In the subsequent sections, it is shown how to solve the DT tracking HJB equation online in real time without complete knowledge of the augmented system dynamics.

Remark 3.6. Note that in (3.29) both feedback and feedforward parts of the control input are obtained simultaneously by minimizing the performance function (3.16) subject to the augmented system (3.13). This enables us to extend reinforcement learning techniques for solving the nonlinear optimal tracking problem without using complete knowledge of the system dynamics. Also, it enables us to encode the input constraints into the optimization problem using the nonquadratic function (3.18).

Remark 3.7. It is clear from (3.29) that the optimal control input never exceeds its permitted bounds. This is a result of reformulation of the nonlinear tracking problem to minimize of the nonquadratic performance function (3.16) subject to the augmented system (3.13). In this formulation, both feedback and feedforward parts of the control input are obtained simultaneously by minimizing the performance function (3.16) and the control input stays in its permitted bounds because of using the nonquadratic function (3.18) for penalizing the control input.

3.3.3 Optimal tracking problem with input constraints

The following Theorem 3.1 shows that the solution obtained by the DT tracking HJB equation gives the optimal solution and locally asymptotically stabilizes the error dynamics (3.12) in the limit as the discount factor goes to one.

Lemma 3.1. For any admissible control policy $u(k)$, Suppose $V(X(k)) \in C^1 : \mathbb{R}^n \to \mathbb{R}$ is a smooth positive definite solution to the Bellman equation (3.22). Define $u^* = u(V(X))$ by (3.29) in terms of $V(X)$. Then

$$H(X(k),u(k),V) = H(X(k),u^*(k),V) + 2\int_{u^*(k)}^{u(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\,R\,ds + \gamma V(F(X(k)) + G(X(k))u) \\ -\gamma V(F(X(k)) + G(X(k))u^*) \tag{3.31}$$

Proof . The Hamiltonian function is

$$H(X(k),u(k),V) = X(k)^T Q_1\, X(k) + 2\int_0^{u(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\,R\,ds + \gamma V(X(k+1)) - V(X(k)) \tag{3.32}$$

By adding and subtracting the terms $2\int_0^{u^*(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\,R\,ds$ and $\gamma V(F(X(k)) + G(X(k))\,u^*)$ to (3.32) yields

$$\begin{aligned} &H(X(k),u(k),V) = X(k)^T Q_1\, X(k) + 2\int_0^{u^*(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\,R\,ds + \\ &\gamma V(F(X(k)) + G(X(k))u^*) - V(X(k)) + \\ &2\int_{u^*(k)}^{u(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\,R\,ds + \gamma V(F(X(k)) + G(X(k))u) - \gamma V(F(X(k)) + G(X(k))u^*) \end{aligned} \tag{3.33}$$

The proof is complete. ∎

Theorem 3.1. Solution to the Optimal Control Problem. Consider the augmented system (3.13) with performance function (3.16). Suppose $V^*(X(k)) \in C^1 : \mathbb{R}^n \to \mathbb{R}$ is a smooth positive definite solution to the DT HJB equation (3.30). Define control $u^* = u(V^*(k))$ as given by (3.29). Then, the closed-loop system

$$X(k+1) = F(X(k)) + G(X(k))u^* = F(X(k)) + G(X(k))\bar{U}\varphi\left(-\frac{\gamma}{2}(\bar{U}R)^{-T}G(X)^T\frac{\partial V^*(X(k+1))}{\partial(X(k+1))}\right) \tag{3.34}$$

is locally asymptotically stable in the limit, as the discount factor goes to one. Moreover, $u^* = u(V^*(k))$ is bounded and minimizes the value function (3.16) over all bounded controls, and the optimal value on $[0,\infty)$ is given by $V^*(X(k))$.

Proof. Clearly, (3.29) is bounded**.** The proof of the stability and optimality of the DT tracking HJB equation is given separately as follows.

1. Stability. Suppose that the value function $V^*(X(k))$ satisfies the DT tracking HJB equation. Then, using (3.30) one has

$$V^*(X(k)) - \gamma V^*(X(k+1)) = X(k)^T Q_1 X(k) + W(u^*(k)) \tag{3.35}$$

Multiplying both sides of (3.35) by γ^k gives

$$\gamma^k V^*(X(k)) - \gamma^{k+1}V^*(X(k+1)) = \gamma^k(X(k)^T Q_1 X(k) + W(u^*(k))) \tag{3.36}$$

Defining the difference of the Lyapunov function as $\Delta(\gamma^k V^*(X(k))) = \gamma^{(k+1)}V^*(X(k+1)) - \gamma^k V^*(X(k))$ and using (3.36) yields

$$\Delta(\gamma^k V^*(X(k))) = -\gamma^k(X(k)^T Q_1 X(k) + W(u^*(k))) \tag{3.37}$$

Eq. (3.37) shows that the tracking error is bounded for the optimal solution, but its asymptotic stability cannot be concluded. However, if $\gamma = 1$ (which can be chosen only if the reference trajectory goes to zero), LaSalle's extension can be used to show that the tracking error is locally asymptotically stable. In fact, the LaSalle's extension says that the states of the system converge to a region wherein $\dot{V} = 0$. By using equation (3.37), we have $\dot{V} = 0$ if $e(k) = 0$ and $W(u(k)) = 0$. Since $W(u(k)) = 0$ if $e(k) = 0$, therefore for $\gamma = 1$, the tracking error is locally asymptotically stable. This confirms that in the limit as the discount factor goes to one, the optimal

control input resulted from solving the DT tracing HJB equation makes the error dynamics asymptotically stable.

If the discount factor is chosen as a nonzero value, one can make the tracking error as small as desired by choosing a discount factor close to one in the performance function (3.16).

2. Optimality. We now prove that the HJB equation provides the sufficient condition for optimality. Note that for any admissible control input $u(k)$ and initial condition $X(0)$, one can write the value function (3.16) as

$$\begin{aligned} V(X(0),u) &= \sum_{i=0}^{\infty} \gamma^i [X(i)^T Q_1\, X(i) + W(u(i))] = \sum_{i=0}^{\infty} \gamma^i \,[X(i)^T Q_1\, X(i) + W(u(i))] \\ &+ \sum_{i=0}^{\infty} \gamma^i \,[\gamma V(X(k+1)) - V(X(k))] + V(X(0)) \end{aligned} \tag{3.38}$$

Then

$$V(X(0),u) = \sum_{i=0}^{\infty} \gamma^i \, H(X,u,V) + V(X(0)) \tag{3.39}$$

Substituting (3.31) into (3.39) and considering $H(X^*,u^*,V^*) = 0$ and (3.38) yields

$$\begin{aligned} V(X(0),u) = \sum_{i=0}^{\infty} \gamma^i \Big[2\int_{u^*(k)}^{u(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\, R\, ds + \gamma V^*(F(X(k)) + \\ G(X(k))u) - \gamma V^*(F(X(k)) + G(X(k))u^*) \Big] + V^*(X(0)) \end{aligned} \tag{3.40}$$

Or equivalent

$$V(X(0),u) = M + V^*(X(0)) \tag{3.41}$$

where

$$M = \sum_{i=0}^{\infty} \gamma^i \Big[2\int_{u^*(k)}^{u(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\, R\, ds + \gamma V^*(F(X(k)) + G(X(k))u) - \gamma V^*(F(X(k)) + G(X(k))\, u^*) \Big] \tag{3.42}$$

To show that u^* is an optimal control and the optimal value function is $V^*(X(0))$, it is needed to show that the term M is bigger than zero when $u \neq u^*$ and it is zero when $u = u^*$.

Taking $\varphi^{-1}(.)$ from two sides of (3.29), yields

$$2\,(\bar{U}R)^T\varphi^{-1}(\bar{U}^{-1}u^*) = -\gamma G(X)^T\,\frac{\partial V^*(X(k+1))}{\partial X(k+1)} \tag{3.43}$$

Also, we have

$$\frac{dV^*(X(k+1))}{du(k)} = (\frac{\partial V^*(X(k+1))}{\partial X(k+1)})^T\,\frac{\partial X(k+1)}{\partial u(k)} = (\frac{\partial V^*(X(k+1))}{\partial X(k+1)})^T G(X) \tag{3.44}$$

Using (3.44) in (3.43) yields

$$2\,(\bar{U}R)^T\varphi^{-1}(\bar{U}^{-1}u^*) = -\gamma(\frac{dV^*(X(k+1))}{du(k)})^T \tag{3.45}$$

By integrating (3.45) one has

$$\begin{aligned}&\int_{u^*}^{u} 2\ \varphi^{-T}(\bar{U}^{-1}u^*)\bar{U}R\ ds = -\gamma\int_{u^*}^{u}\frac{dV^*(X(k+1))}{ds(k)}\ ds\\ &2(u-u^*)\varphi^{-T}(\bar{U}^{-1}u^*)\bar{U}R = -\gamma[V^*(F(X(k)) + G(X(k))u)\\ &-V^*(F(X(k)) + G(X(k))u^*)\Big]\end{aligned} \tag{3.46}$$

By using (3.46) and considering $\alpha(s) = \varphi^{-T}(\bar{U}^{-1}s)\bar{U}\,R$, M becomes

$$M = \sum_{i=0}^{\infty}\gamma^i\left[2\int_{u^*(k)}^{u(k)}\alpha\,(s)\,ds - 2(u-u^*)\,\alpha(u^*)\right] \tag{3.47}$$

We consider

$$l = \int_{u^*(k)}^{u(k)}\alpha\,(s)ds - (u-u^*)\,\alpha(u^*) \tag{3.48}$$

By using (3.20), we rewrite (3.48) as

$$l = \sum_{j=1}^{m}\int_{u_j^*(k)}^{u_j(k)}\alpha_j(s_j)ds_j - (u_j - u_j^*)\,\alpha_j(u_j^*) \tag{3.49}$$

By defining $l_j = \int_{u_j^*(k)}^{u_j(k)}\alpha_j(s_j)ds_j - (u_j - u_j^*)\,\alpha_j(u_j^*)$, (3.49) is

$$l = \sum_{j=1}^{m} l_j \tag{3.50}$$

To complete the proof, it is needed to show that l_j , $j = 1,..,m$ is bigger than zero for $u_j \neq u_j^*$ and they are zero for $u_j = u_j^*$. It is clear that $l_j = 0$ for $u_j = u_j^*$. Now it should be shown that $l_j > 0$ for $u_j \neq u_j^*$. First it is assumed $u_j > u_j^*$. Then using mean value theorem for the integrals, there exists a $\bar{u}_j \in [u_j^*, u_j]$ such that

$$\int_{u_j^*(k)}^{u_j(k)} \alpha_j(s_j) ds_j = (u_j - u_j^*)\,\alpha(\bar{u}_j) > (u_j - u_j^*)\,\alpha_j(u_j^*) \tag{3.51}$$

Therefore, $l_j > 0$ for $u_j > u_j^*$. Then, it is assumed $u_j < u_j^*$. Using mean value theorem, there exists a $\bar{u}_j \in [u_j, u_j^*]$ such that

$$\int_{u_j^*(k)}^{u_j(k)} \alpha_j(s_j) ds_j = (u_j - u_j^*)\,\alpha(\bar{u}_j) > (u_j - u_j^*)\,\alpha_j(u_j^*) \tag{3.52}$$

Therefore, $l_j > 0$ for $u_j < u_j^*$. Then the proof is completed. ∎

3.4. RL for Solving the Nonlinear Tracking Problem

The DT HJB equation (3.30) cannot be solved exactly and it also requires complete knowledge of the system dynamics. In this section, an online policy iteration algorithm is given to find the solution to the DT tracking HJB equation. This algorithm still requires complete knowledge of the system dynamics. To obviate the requirement of complete knowledge of the system dynamics, an actor-critic algorithm is presented in the next subsection to solve the nonlinear tracking problem.

Note that instead of the solving the DT HJB equation directly, one can use the following iterative policy iteration algorithm which uses the tracking Bellman equation (3.22) to evaluate a fixed control policy and the update policy in form of (3.29) to find an improved control policy.

3.4.1. Algorithm 3.1. Online Policy Iteration Algorithm

Initialize the control input $u^0(k)$.

1. Policy evaluation: Find the value function related to the policy $u^i(k)$ by solving the Bellman equation (3.22)

$$V^i(X(k)) = X(k)^T Q_1 X(k) + W(u^i(k)) + \gamma V^i(X(k+1)) \tag{3.53}$$

2. Policy improvement: Update the policy using

$$u^{i+1}(X) = \bar{U}\varphi\left(-\frac{\gamma}{2}(\bar{U}R)^{-T} G(X(k))^T \frac{\partial V^i(X(k+1))}{\partial X(k+1)}\right) \tag{3.54}$$

Remark 3.8. This policy iteration algorithm can be implemented online using the least squares method by standard technique [13]. This requires a persistent excitation (PE) condition to allow solution of repeated Bellman equation (3.53) at successive time instants in a batch fashion.

The augmented system dynamics is not needed to solve the Bellman equation (3.53) but must be known for updating the control input using (3.54). To obviate the requirement complete knowledge of the system dynamics or reference trajectory dynamics, an actor-critic algorithm [13] is developed in the next subsection to solve the nonlinear tracking problem.

3.4.2. actor-critic structure for solving the nonlinear tracking problem

In this section, the solution to the DT tracking HJB equation is learned using an actor-critic structure which does not require knowledge of drift system dynamics or reference trajectory dynamics. In contrast to the sequential online Algorithm 3.1, the proposed algorithm is an online synchronized algorithm. That is, instead of sequentially updating the value function and the policy, as in Algorithm 3.1, the value function and the policy are updated simultaneously. This method has been developed for continuous-time systems in [44] and for discrete-time systems with one-layer NN in [43]. The value function and the control input are approximated with two separate two-layer perceptron neural networks (NNs) [21]-[23]. The critic NN estimates the value function

and is a function of the tracking error and the reference trajectory. The actor NN represents a control policy and is a function of the tracking error and the reference trajectory. The critic NN is updated to minimize the tracking Bellman error and the actor NN is to minimize the value function.

Actor NN

A two-layer NN with one hidden layer is used as the actor NN to approximate the control input for the nonlinear tracking problem. Fig. 3.1 shows the actor NN. The input for the actor NN is $X(k)$ defined in (3.14).

The output of the actor NN are given as

$$\begin{aligned} &\hat{u}(X(k),W_a^{(1)}(k),W_a^{(2)}(k)) = [\hat{u}_1,...,\hat{u}_m], \\ &\hat{u}_i = \sigma_i(W_{a_i}^{(2)}(k)\,\phi_a(W_a^{(1)}(k)X(k))) = \sigma_i(\sum_{l=1}^{N_a}\hat{w}_{a_{il}}^{(2)}(k)\,\phi_{a_l}(\sum_{j=1}^{2n}\hat{w}_{a_{lj}}^{(1)}(k)X_j(k))) \end{aligned} \tag{3.55}$$

where $\phi(k) = [\phi_1(k),...,\phi_{N_a}(k)]^T \in \mathbb{R}^{N_a\times 1}$ is the vector of hidden-layer activation functions with $\phi(.) = \tanh(.)$, and $\sigma_i(.) = \overline{u}_i \tanh(.)$, $i = 1,...,m$ is the *i-th* output-layer activation function.

The weight matrix between the neurons in the input and the hidden layers is defined as

$$W_a^{(1)} = \begin{bmatrix} w_{a_{11}}^{(1)} & ... & w_{a_{1(2n)}}^{(1)} \\ \vdots & \vdots & \vdots \\ w_{a_{N_a1}}^{(1)} & ... & w_{a_{N_a(2n)}}^{(1)} \end{bmatrix} \in \mathbb{R}^{N_a\times 2n}$$

where N_a is number of the hidden-layer neurons and each element $w_{a_{ij}}^{(1)}$ denotes the weight from *j-th* input to *i-th* hidden neurons. Also, define the matrix of the weights between the neurons in the hidden and the output layers as

$$W_a^{(2)} = \begin{bmatrix} w_{a_{11}}^{(2)} & ... & w_{a_{1N_a}}^{(2)} \\ \vdots & \vdots & \vdots \\ w_{a_{m1}}^{(2)} & ... & w_{a_{mN_a}}^{(2)} \end{bmatrix} \in \mathbb{R}^{m\times N_a}$$

where $\hat{w}_{a_{ij}}^{(2)}(k)$ is the weight between *j-th* hidden layer and *i-th* output layer.

Note that the output of the actor NN satisfies the input bounds defined in (3.15).

Critic NN

A two-layer NN with one hidden layer is used as critic NN to approximate the value function. Fig. 3.2 shows the critic NN. The input for critic NN is $X(k)$.

The output of the critic NN is given as

$$\hat{V}(X(k)) = W_c^{(2)}(k)\phi_c(W_c^{(1)}(k)X(k)) = \sum_{i=1}^{N_c} \hat{w}_{c_i}^{(2)}(k)\,\phi_{c_i}(\sum_{j=1}^{2n} \hat{w}_{c_{ij}}^{(1)}(k)X_j(k)) \tag{3.56}$$

where $\phi(k) = [\phi_1(k),...,\phi_{N_c}(k)]^T \in \mathbb{R}^{N_c \times 1}$ is the vector of hidden layer activation functions, with $\phi(.) = \tanh(.)$ and the matrix of the weights between the neurons in the input and the hidden layers is defined as

$$W_c^{(1)} = \begin{bmatrix} w_{c_{11}}^{(1)} & ... & w_{c_{1(2n)}}^{(1)} \\ \vdots & \vdots & \vdots \\ w_{c_{N_c1}}^{(1)} & ... & w_{c_{N_c(2n)}}^{(1)} \end{bmatrix} \in \mathbb{R}^{N_c \times 2n},$$

where N_c is number of the hidden layer neurons and each element $w_{c_{ij}}^{(1)}$ denotes the weight from *j-th* input to *i-th* hidden neurons. Also, define the vector of the weights between the neurons in the hidden and the output layer as

$$W_c^{(2)} = [w_{c_1}^{(2)}, w_{c_2}^{(2)},..., w_{c_{N_c}}^{(2)}] \in \mathbb{R}^{1 \times N_c}$$

where $\hat{w}_{c_i}^{(2)}(k)$ is the weight between *i-th* hidden layer and output layer.

3.5. Learning Rules for Actor and Critic NNs

In this section the approximate gradient descent rules for updating the critic and actor networks are developed. The objective of tuning the critic weights is to minimize the Bellman equation error and the objective of tuning the actor weights is to minimize the approximate value function. In order to obviate the need for $V(\mathrm{X}(k+1))$, which requires the system model to predict

$X(k+1)$, the previous values of the system state $X(k)$ and the state value $V(X(k))$ are stored and used for updating the actor and critic NNs weights.

Updating rule for the critic network

The prediction error of the tracking Bellman equation is defined as

$$e_c(k) = \gamma \hat{V}(X(k)) + U_k - \hat{V}(X(k-1)) \tag{3.57}$$

where

$$U_k = X(k)^T Q_1 X(k) + 2\int_0^{u(k)} \varphi^{-T}(\bar{U}^{-1}s)\bar{U} R\, ds$$

is the reinforcement signal or the temporal difference signal. It is desired to select the weights of the critic network to minimize the square Bellman error

$$E_c(k) = \frac{1}{2} e_c^2(k) \tag{3.58}$$

The tuning laws for the critic weights are selected as the normalized approximation to gradient descent algorithm, that is,

$$\hat{w}_{c_{ij}}^{(1)}(k+1) = \hat{w}_{c_{ij}}^{(1)}(k) - l_c \frac{\partial E_c(k)}{\partial \hat{w}_{c_{ij}}^{(1)}(k)} \tag{3.59}$$

$$\hat{w}_{c_i}^{(2)}(k+1) = \hat{w}_{c_i}^{(2)}(k) - l_c \frac{\partial E_c(k)}{\partial \hat{w}_{c_i}^{(2)}(k)} \tag{3.60}$$

where $l_c > 0$ is learning rate.

Using the chain rule one has

$$\begin{aligned} \frac{\partial E_c(k)}{\partial \hat{w}_{c_{ij}}^{(1)}(k)} &= \frac{\partial E_c(k)}{\partial e_c(k)} \frac{\partial e_c(k)}{\partial \hat{V}(k)} \frac{\partial \hat{V}(k)}{\partial \phi_{c_i}(k)} \frac{\partial \phi_{c_i}(k)}{\partial \hat{w}_{c_{ij}}^{(1)}(k)} \\ &= \gamma\, e_c(k) \hat{w}_{c_i}^{(2)}(k)(1-\phi_{c_i}^2(k)) X_j(k) \end{aligned} \tag{3.61}$$

$$\frac{\partial E_c(k)}{\partial \hat{w}_{c_i}^{(2)}(k)} = \frac{\partial E_c(k)}{\partial \hat{V}(k)} \frac{\partial \hat{V}(k)}{\partial \hat{w}_{c_i}^{(2)}(k)} = \gamma\, e_c(k)\, \phi_{c_i}(k) \tag{3.62}$$

Updating rule for the actor network

The actor (3.55) is updated to minimize the approximated value function. This is done by minimizing the error between a target control input which is obtained by minimizing the value function and the actual control input which is applied to the system. Let the current estimation of the value function be $\hat{V}$. Then, based on the policy update law (3.54), the target control input at time k is

$$\tilde{u}(X) = \bar{U}\varphi\left(-\frac{\gamma}{2}(\bar{U}R)^{-1}G(X(k))^T \frac{\partial \hat{V}(X(k+1))}{\partial X(k+1)}\right) \tag{3.63}$$

However, to obtain the value at time $k+1$, the states are required to be predicted by using a model network. But, we do not use a model network to predict the future value. Rather, we store the previous value of the system state and the state value and try to minimize the error between the target control input and the actual control input given current actor and critic estimate weights while the previous stored state $X(k-1)$ is used as the input to the actor and critic. That is to minimize

$$e_a(k) = \tilde{u}(X(k-1)) - \hat{u}(k,k-1) \tag{3.64}$$

where $\hat{u}(k,k-1) = \hat{u}(X(k-1), W_a^{(1)}(k), W_a^{(2)}(k))$ is the output of the actor NN at time $(k-1)$ if the current NN weights are used. It is desired to select the weights of the actor network to minimize the square actor NN error

$$E_a(k) = \frac{1}{2}e_a^2(k) \tag{3.65}$$

The tuning laws for the actor weights are selected as the normalized gradient descent algorithm, that is,

$$\hat{w}_{a_{ij}}^{(1)}(k+1) = \hat{w}_{a_{ij}}^{(1)}(k) - l_a \frac{\partial E_a(k)}{\partial \hat{w}_{a_{ij}}^{(1)}(k)} \tag{3.66}$$

$$\hat{w}^{(2)}_{a_{ij}}(k+1) = \hat{w}^{(2)}_{a_{ij}}(k) - l_a \frac{\partial E_a(k)}{\partial \hat{w}^{(2)}_{a_{ij}}(k)} \tag{3.67}$$

where $l_a > 0$ is learning rate. The chain rule for the weights between the input layer and the hidden layer yields

$$\frac{\partial E_a(k)}{\partial \hat{w}^{(1)}_{a_{ij}}(k)} = \frac{\partial E_a(k)}{\partial e_a(k)} \frac{\partial e_a(k)}{\partial \hat{u}(k,k-1)} \frac{\partial \hat{u}(k,k-1)}{\partial \hat{w}^{(1)}_{a_{ij}}(k)} = -e_a(k) \sum_{l=1}^{m} [\overline{u}_l (1 - \frac{1}{\overline{u}_l^2} \sigma_l{}^2 (X(k-1), W_a^{(1)}(k), W_a^{(2)}(k))) \times$$
$$\sum_{i=1}^{N_a} \hat{w}^{(2)}_{a_{li}}(k)(1 - \phi_{a_l}(k)^2)] \times \sum_{j=1}^{2n} X_j(k-1)$$

(3.68)

and for the weights between the hidden layers and the output layers is

$$\frac{\partial E_a(k)}{\partial \hat{w}^{(2)}_{a_{ij}}(k)} = \frac{\partial E_a(k)}{\partial e_a(k)} \frac{\partial e_a(k)}{\partial \hat{u}(k,k-1)} \frac{\partial \hat{u}(k,k-1)}{\partial \hat{w}^{(2)}_{a_{ij}}(k)} = e_a(k) \times$$
$$-\overline{u}_i (1 - \frac{1}{\overline{u}_i^2} \sigma_i{}^2 (X(k-1), W_a^{(1)}(k), W_a^{(2)}(k)) \times \phi_j(k)$$

(3.69)

Note that the stability proof and convergence of the actor-critic network during learning are provided in [43], [94].

Fig 3.3 depicts the schematic of the proposed update law for the actor-critic structure. As can be seen from this figure, the previous values of the state of the system and the state of the system are required to be stored and used in updating the actor and critic NNs weights.

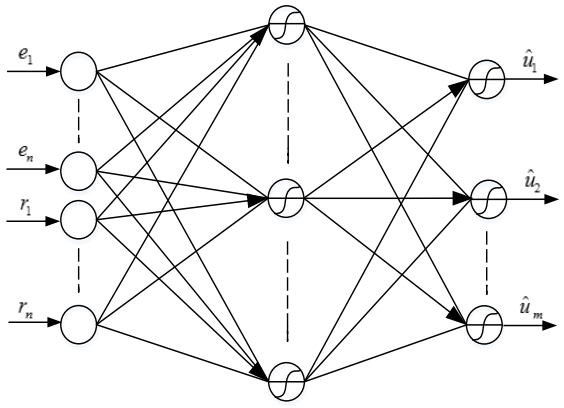

Fig 3.1. The actor network

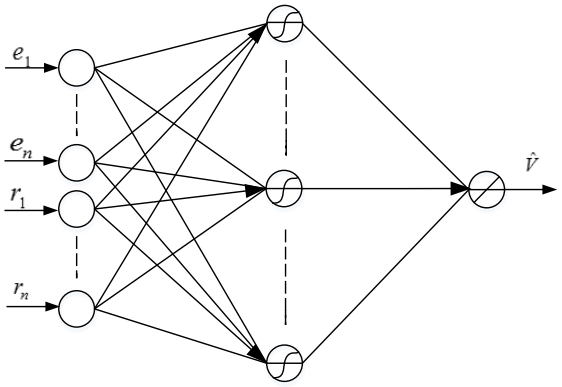

Fig 3.2. The critic network

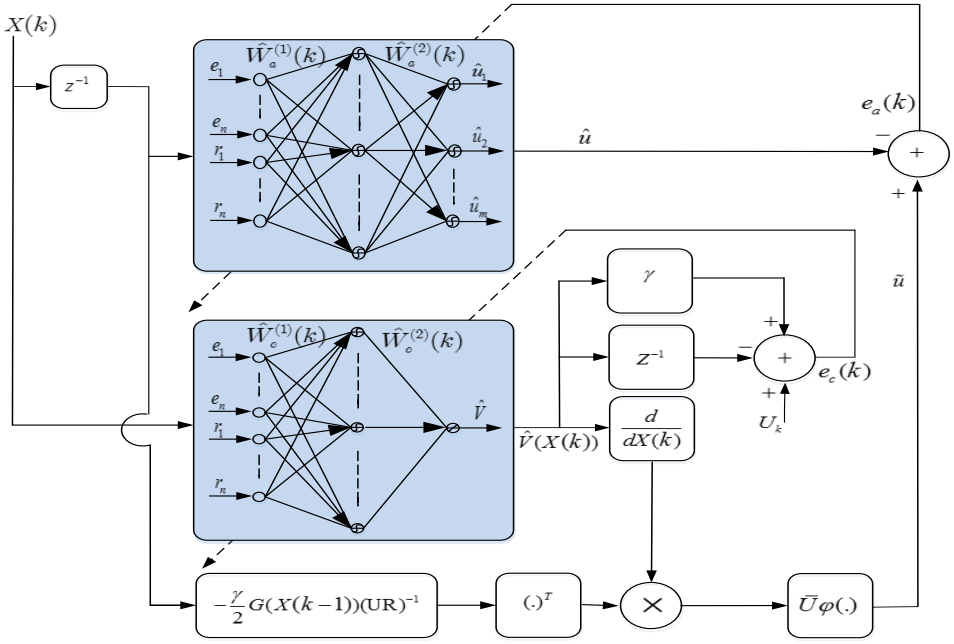

Fig 3.3. Schematic of the proposed actor and critic learning

Remark 3.9. Note that in the proposed algorithm both the actor and critic NNs are updated simultaneously. In fact, the control input given by (3.55) is applied to system continually while converging to the optimal solution. Since the weight update laws for actor and critic are coupled,

the critic NN is also required to be updated continually and simultaneously with the control input. This simultaneous update rule for actor and critic makes convergence guarantees more difficult to prove.

Remark 3.10. Note that for solving the optimal tracking problem using reinforcement learning, the control input should be persistently exciting (PE). This ensures sufficient exploration of the state of the systems. An exploratory signal consisting of sinusoids of varying frequencies can be added to the control input to insure PE qualitatively.

Remark 3.10. A challenge of the proposed RL methods based on value function approximation is to find an accurate structure for the value function. If the structure of the network for approximator (eg., the number of neural network layers or neurons) is not selected appropriately, the learning algorithm may never converge and this can consequently lead to instability of the feedback control systems. Due to the difficulty to find an appropriate structure for the value function approximators for the high-dimensional problems, these methods have typically been restricted to simple demonstration problems.

3.6. Simulation Results

In this section, a simulation example is provided to illustrate the design procedures and verify the effectiveness of the proposed scheme.

A nonlinear system dynamics is considered as

$$\begin{aligned} x_1(k+1) &= -0.8\,x_2(k) \\ x_2(k+1) &= -0.45\,x_1(k) - \sin(x_2(k)) + 0.2\,x_2(k)u(k) \end{aligned} \tag{3.70}$$

It is assumed that the control input is bounded by $|u(k)| \le 0.4$.

The sinusoid reference trajectory dynamics is generated by the command generator given by

$$\begin{aligned} r_1(k+1) &= -r_1(k) \\ r_2(k+1) &= -r_2(k) \end{aligned} \tag{3.71}$$

The performance index is consider as (3.16) with $Q = 20I$, $R = 1$ and $\gamma = 0.3$ that *I* is the identity matrix with appropriate dimension.

The proposed actor-critic algorithm is applied to the system (3.70). The weights of neural network (NN) for both actor and critic networks are initialized randomly between -1 and 1. A small probing noise is added to the control input to excite the system during learning.

The critic is a two-layer NN with 5 activation functions in the hidden layer and one activation function in the output layer. The type of activation functions in the hidden layer is tanh(.) and in the output layer is the identity function.

At the end of learning, the weight matrix between input layer and hidden layer converges to

$$W_c^{(1)} = \begin{bmatrix} -0.9910 & 1.6093 & 5.5495 & 3.8788 \\ 1.1450 & 3.6761 & 2.4605 & 1.9193 \\ 3.0482 & 1.5314 & -1.0640 & -0.2953 \\ 2.1615 & 1.7524 & 2.9390 & 3.2034 \\ -0.8530 & 5.7600 & -1.4361 & -1.8192 \end{bmatrix} \tag{3.72}$$

and the weight vector between the hidden layer and output layer converges to

$$W_c^{(2)} = \begin{bmatrix} 2.7743 & 1.3367 & 5.2477 & 1.1616 & 0.0228 \end{bmatrix} \tag{3.73}$$

Fig. 3.4. shows that the convergence of the weight vector between the hidden layer and the output layer in the critic network.

The actor is also a two-layer NN with 5 activation functions in the hidden layer and one $\bar{u}\tanh(.)$ activation function in the output layer that $\bar{u}$ is the bound for control input. The type of the activation functions in the hidden layer is tanh(.).

At the end of learning, the weight matrix between input layer and hidden layer converges to

$$W_a^{(1)} = \begin{bmatrix} 3.0711 & 6.5380 & -1.8147 & 0.9523 \\ -2.9429 & 3.7955 & 5.5219 & 1.2470 \\ -1.4521 & -1.9017 & 2.1629 & 0.8028 \\ 2.2684 & 0.2858 & 3.8524 & 4.7390 \\ 1.6565 & 2.6921 & 3.8270 & 4.9242 \end{bmatrix} \tag{3.74}$$

and the weight vector between the hidden layer and output layer converges to

$$W_a^{(2)} = \begin{bmatrix} 4.9707 & 3.5331 & 3.5613 & 7.4699 & -2.1780 \end{bmatrix} \tag{3.75}$$

Fig. 3.5. shows that the convergence of the weight vector between the hidden layer and the output layer in the actor network. Fig. 3.6. and Fig. 3.7. show that the states of the system $x(k)$ track the reference trajectory $r(k)$ and guarantee the stability for the proposed method after the learning is finished and the probing noise is removed. This confirms that our proposed method successfully finds an optimal tracking controller for system (3.70).

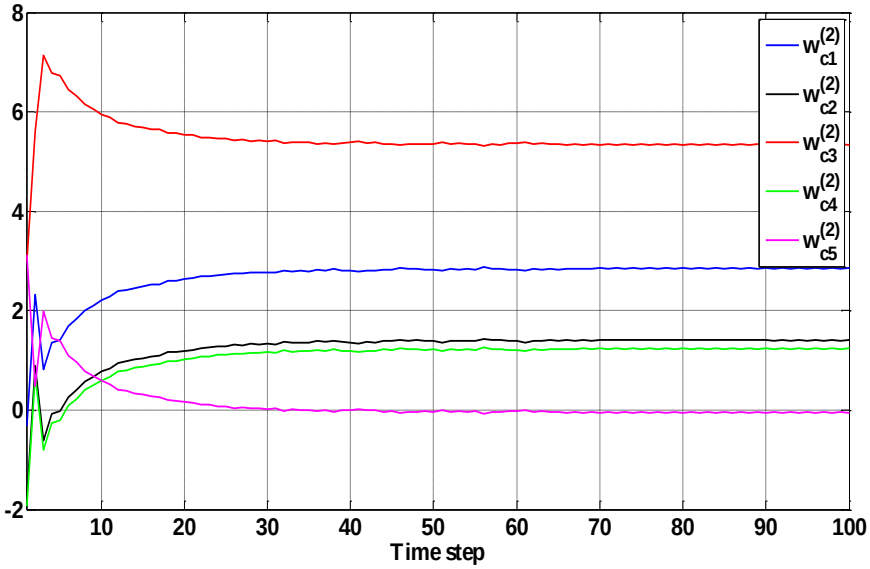

Fig 3.4. The weights of output layer of critic network

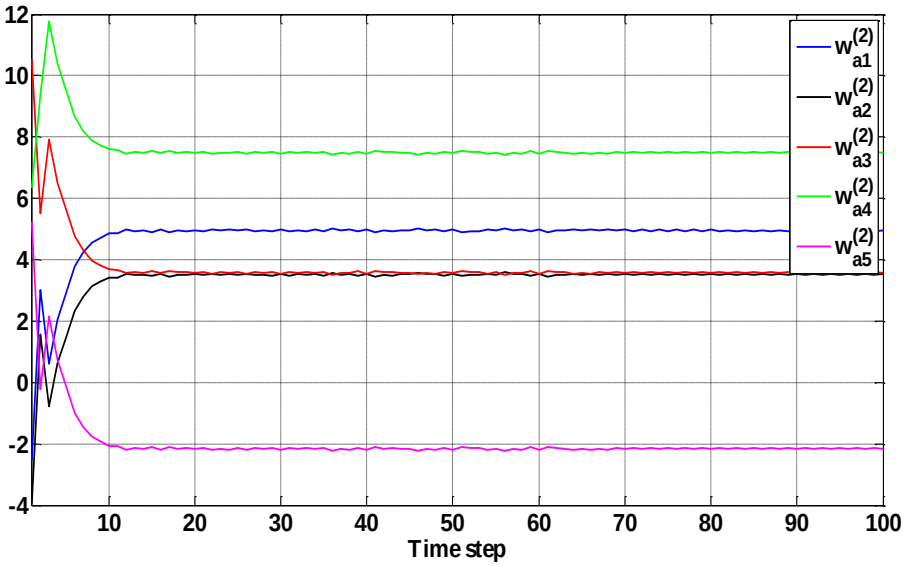

Fig. 3.5. The weights of output layer of actor network

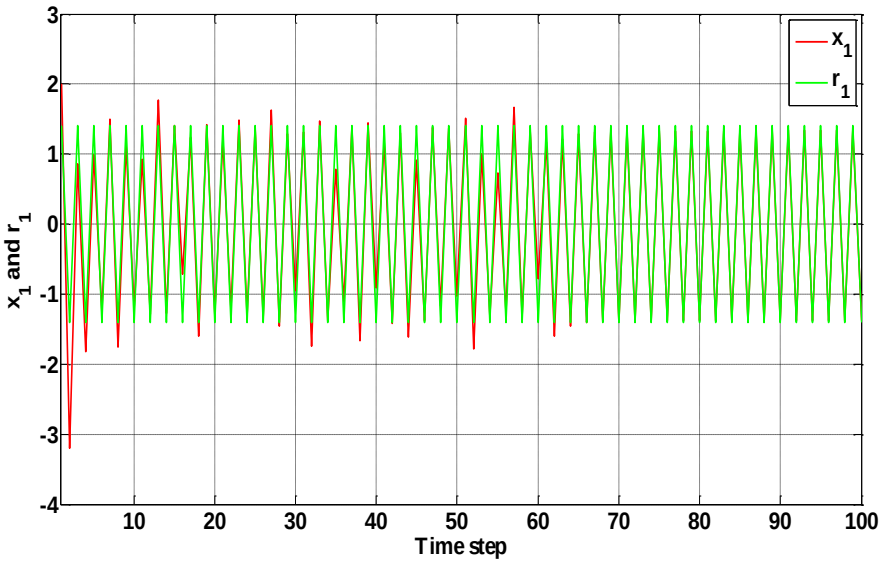

Fig. 3.6. Evaluation of x_1 and r_1 during the learning process

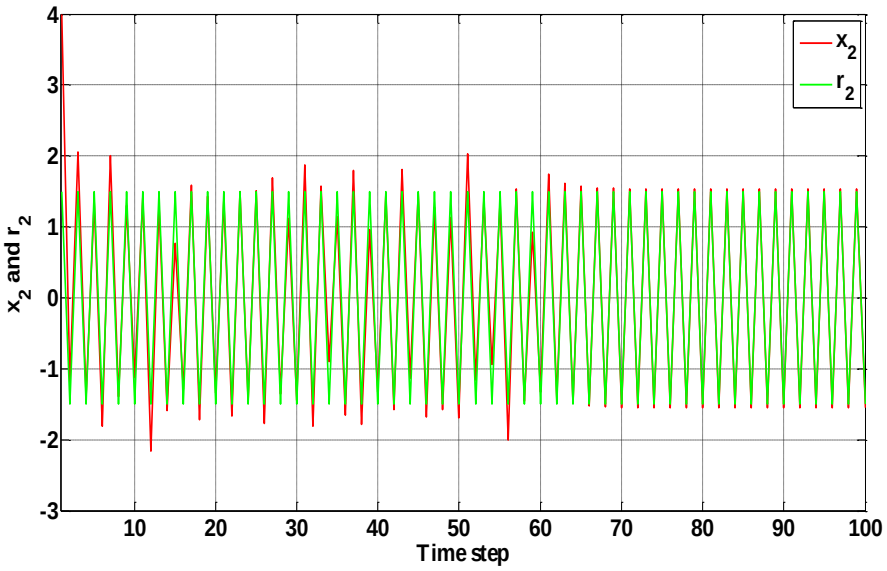

Fig. 3.7. Evaluation of x_2 and r_2 during the learning process

3.7. Conclusion

A new formulation of the nonlinear DT tracking control problem in the presence of input constraints was presented in this chapter. A novel discounted performance function was presented and it was shown that the minimization of this performance function gives both feedback and feedforward parts of the bounded optimal control input simultaneously. An actor-critic structure was used to learn the solution of the tracking problem online without requiring knowledge of the system drift dynamics. The actor and critic NNs were updated simultaneously and used previous stored state of the system and value function, instead of their current values, to avoid the requirement of the system model dynamics.

Chapter 4

H_∞ CONTROL OF LINEAR DISCRETE-TIME SYSTEMS: OFF-POLICY REINFORECMENT LEARNING

4.1. Introduction

The H_∞ control is a well-known robust control approach which is used to attenuate the effects of disturbances on the performance of dynamical systems [75]-[77]. It has a strong connection to the zero-sum game problem [78], where the controller and the disturbance are considered as minimizing and maximizing players, respectively. Finding the solution to the zero-sum game problem leads to solving the game algebraic Riccati equation (GARE) for the linear systems. Numerical and iterative methods have been widely used to solve the GARE. However, they mostly require complete knowledge of the system dynamics.

Q-learning algorithm has also been used to find the solution to the optimal control problem for systems with disturbances by solving the GARE (Al-Tamimi, Lewis, & Abu-Khalaf, 2007). Although elegant, there are two main problems with this algorithm. First, Q-learning requires the disturbance input to be updated in a prescribed manner. However, the disturbance input cannot be updated in a prescribed manner in more real-world applications. Second, Q-learning algorithm does not cancel out the effects of probing noise (which is used to excite the system) in the Bellman equation while evaluating the value function. This may result in bias and can affect the convergence of the algorithm.

To avoid these mentioned problems, in this paper, an off-policy RL algorithm [8] is developed. In off-policy methods, two separate policies are used. 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. Off-policy RL is presented for solving the

optimal control problem of continuous-time (CT) systems with partially-unknown or completely-unknown dynamics [64], [94]-[100]..

To our knowledge, off-policy RL for DT systems has not been developed yet. Although Q-learning is originally off-policy, what is called Q-learning in control society is actually SARSA [8], which is on-policy. Developing off-policy RL algorithms for DT systems is not straightforward because of the appearance of both system matrix *A* and control matrix *B* in the policy update equation.

In this chapter, a model-free solution to the H_∞ control of linear discrete-time systems is presented. The proposed approach employs off-policy RL to solve the game algebraic Riccati equation online using the measured data along the system trajectories. Like existing model-free RL algorithms, no knowledge of the system dynamics is required. However, the proposed method has two main advantages. First, the disturbance input does not need to be adjusted in a specific manner. This makes it more practical as the disturbance cannot be specified in most real-world applications. Second, there is no bias as a result of adding a probing noise to the control input to maintain persistence of excitation (PE) condition. Consequently, the convergence of the proposed algorithm is not affected by probing noise.

The chapter is organized as follows. In Section 4.2, H_∞ control problem, and the effect of adding probing noise are given. The proposed off-policy RL algorithm is presented in Section 4.3. The effectiveness of this method is shown by performing an H_∞ control autopilot design for an F-16 aircraft in Section 4.4.

4.2. Background and Problem Formulation

4.2.1. Discrete-time H_∞ control problem

Consider the following linear discrete-time system

$$x_{k+1} = A x_k + B u_k + D w_k \tag{4.1}$$

where $x_k \in \mathbb{R}^n$ is the system state, $u_k \in \mathbb{R}^{m_1}$ is the control input, and $w_k \in \mathbb{R}^{m_2}$ is the external disturbance input.

Definition 4.1. The system (1) has L_2 -gain less than or equal to γ if

$$\sum_{k=0}^{\infty} \left[x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k \right] \le \gamma^2 \sum_{k=0}^{\infty} w_k^{\mathrm{T}} w_k \tag{4.2}$$

for all $w_k \in L_2[0,\infty)$**,** where Q and R are symmetric positive definite matrices, and $\gamma \ge 0$ is a prescribed constant discturbance attenuation level.

Note that functions in $L_2[0,\infty)$ represent the signals having finite energy over infinite interval $[0,\infty)$. That is,

$$\sum_{k=0}^{\infty} w_k^{\mathrm{T}} w_k < \infty .$$

The H_∞ control is to develop a control input such that the system (4.1) with $w_k = 0$ is asymptotically stable and it satisfies the disturbance attenuation condition (4.2). Based on (4.2), define the infinite horizon performance function as

$$J(x_k, u_k, w_k) = \sum_{i=k}^{\infty} U_i = \sum_{i=k}^{\infty} \left[x_i^{\mathrm{T}} Q x_i + u_i^{\mathrm{T}} R u_i - \gamma^2 w_i^{\mathrm{T}} w_i \right] \tag{4.3}$$

Moreover, using (4.3), for an admissible control policy u_k and a disturbance policy w_k, the value function is defined as

$$V(x_k) = \sum_{i=k}^{\infty} [x_i^{\mathrm{T}} Q x_i + u_i^{\mathrm{T}} R u_i - \gamma^2 w_i^{\mathrm{T}} w_i] \tag{4.4}$$

Assumption 4.1. The pair (A, B) is stabilizable and the pair $(A, \sqrt{Q})$ is observable.

4.2.2. Formulation of the H_∞ control as a zero-sum game

The H_∞ control problem can be expressed as a two-player zero-sum differential game in which the control policy player u_k seeks to minimize the value function, while the disturbance policy player w_k desires to maximize it. The goal is to find the feedback saddle point (u_k^*, w_k^*) such that

$$V^*(x_k) = \min_{u_k} \max_{w_k} J(x_k, u_k, w_k) = \min_{u_k} \max_{w_k} \sum_{i=k}^{\infty} \left[x_i^{\mathrm{T}} Q x_i + u_i^{\mathrm{T}} R u_i - \gamma^2 w_i^{\mathrm{T}} w_i \right] \tag{4.5}$$

By using the value function (4.4), one has

$$V(x_k) = x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k - \gamma^2 w_k^{\mathrm{T}} w_k + \sum_{i=k+1}^{\infty} \left[x_i^{\mathrm{T}} Q x_i + u_i^{\mathrm{T}} R u_i - \gamma^2 w_i^{\mathrm{T}} w_i \right] \tag{4.6}$$

which yields the Bellman equation

$$V(x_k) = x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k - \gamma^2 w_k^{\mathrm{T}} w_k + V(x_{k+1}) \tag{4.7}$$

It is known that for the linear system (4.1) with the quadratic value function (4.4), the value function is given as

$$V(x_k) = x_k^{\mathrm{T}} P x_k \tag{4.8}$$

By using (4.8) in (4.7), the Bellman equation (4.7) becomes

$$x_k^{\mathrm{T}} P x_k = x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k - \gamma^2 w_k^{\mathrm{T}} w_k + x_{k+1}^{\mathrm{T}} P x_{k+1} \tag{4.9}$$

The Hamiltonian function is defined as

$$H(x_k, u_k, w_k) = x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k - \gamma^2 w_k^{\mathrm{T}} w_k + V(x_{k+1}) - V(x_k) \tag{4.10}$$

The optimal control policy u_k^* and the worst-case disturbance w_k^* should satisfy $\partial H(x_k, u_k, w_k)/\partial u_k = 0$ and $\partial H(x_k, u_k, w_k)/\partial w_k = 0$, respectively. Therefore, one has

$$u_k^* = -K_1^* x_k \tag{4.11}$$

$$w_k^* = -K_2^* x_k \tag{4.12}$$

where

$$K_1^* = (R + B^{\mathrm{T}} P B + B^{\mathrm{T}} P D(\gamma^2 I - D^{\mathrm{T}} P D)^{-1} D^{\mathrm{T}} P B)^{-1} (B^{\mathrm{T}} P A + B^{\mathrm{T}} P D(\gamma^2 I - D^{\mathrm{T}} P D)^{-1} D^{\mathrm{T}} P A) \tag{4.13}$$

$$K_2^* = (D^{\mathrm{T}}PD - \gamma^2 I - D^{\mathrm{T}}PB(R + B^{\mathrm{T}}PB)^{-1}B^{\mathrm{T}}PD)^{-1}(D^{\mathrm{T}}PA - D^{\mathrm{T}}PB(R + B^{\mathrm{T}}PB)^{-1}B^{\mathrm{T}}PA) \tag{4.14}$$

and *P* satisfies the game algebraic Riccati equation (GARE)

$$P = A^{\mathrm{T}}PA + Q - \begin{bmatrix} A^{\mathrm{T}}PB & A^{\mathrm{T}}PD \end{bmatrix} \begin{bmatrix} R + B^{\mathrm{T}}PB & B^{\mathrm{T}}PD \\ D^{\mathrm{T}}PB & D^{\mathrm{T}}PD - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^{\mathrm{T}}PA \\ D^{\mathrm{T}}PA \end{bmatrix} \tag{4.15}$$

It is shown in [78], on one hand, that (4.13)-(4.15) solve the zero-sum game problem defined in (4.5), and, on the other hand, that solving zero-sum game problem defined in (4.5) is equivalent to finding a control policy that satisfies the disturbance attenuation condition (4.2). Therefore, the solution of (4.13)-(4.15) guarantees that the disturbance attenuation condition (4.2) is satisfied.

Remark 4.1. It is shown in [75] that there exists a γ^* such that for $\gamma < \gamma^*$, the H_∞ control problem has no solution. In [101], an explicit expression for the infimum of γ, i.e., γ^*, is found. It is shown that if $\gamma \geq \gamma^* \geq 0$, the GARE (4.15) has a unique positive semi-definite solution, the closed-loop system is asymptotically stable, and the disturbance attenuation condition is satisfied.

4.2.3. Online H_∞ PI algorithm

Various algorithms have been developed to solve the GARE (4.15). Policy iteration (PI) algorithm is one of the most used algorithms for solving the GARE (4.15) online which is as follows.

Algorithm 4.1. Online PI algorithm

Initialization: Set the iteration number $j = 0$ and start with a stabilizing control policy u_k^0 and disturbance w_k^0.

1. Solve for P^{j+1} using the Bellman equation

$$x_k^{\mathrm{T}}P^{j+1}x_k = x_k^{\mathrm{T}}Q\,x_k + (u_k^j)^{\mathrm{T}}R\,u_k^j - \gamma^2(w_k^j)^{\mathrm{T}}w_k^j + x_{k+1}^{\mathrm{T}}P^{j+1}x_{k+1} \tag{4.16}$$

2. Update the control and disturbance gains as

$$\begin{aligned} u_k^{j+1} &= -K_1^{j+1}x_k = -(R + B^{\mathrm{T}}P^{j+1}B + B^{\mathrm{T}}P^{j+1}D(\gamma^2 I - D^{\mathrm{T}}P^{j+1}D)^{-1}D^{\mathrm{T}}P^{j+1}B)^{-1} \\ &\times (B^{\mathrm{T}}P^{j+1}A + B^{\mathrm{T}}P^{j+1}D(\gamma^2 I - D^{\mathrm{T}}P^{j+1}D)^{-1}D^{\mathrm{T}}P^{j+1}A)x_k \end{aligned} \tag{4.17}$$

$$w_k^{j+1} = -K_2^{j+1} x_k = -(D^{\mathrm{T}} P^{j+1} D - \gamma^2 I - D^{\mathrm{T}} P^{j+1} B(R + B^{\mathrm{T}} P^{j+1} B)^{-1} B^{\mathrm{T}} P^{j+1} D)^{-1} \times (D^{\mathrm{T}} P^{j+1} A - D^{\mathrm{T}} P^{j+1} B(R + B^{\mathrm{T}} P^{j+1} B)^{-1} B^{\mathrm{T}} P^{j+1} A) x_k \tag{4.18}$$

3. Stop if

$$\left| K_1^{j+1} - K_1^j \right| \le \varepsilon \quad \text{and} \quad \left| K_2^{j+1} - K_2^j \right| \le \varepsilon \tag{4.19}$$

for a small positive value of ε, otherwise set $j = j + 1$ and go to 1.

To implement Algorithm 4.1, the Bellman equation (4.16) can be written as

$$(x_k^{\mathrm{T}} \otimes x_k^{\mathrm{T}} - x_{k+1}^{\mathrm{T}} \otimes x_{k+1}^{\mathrm{T}}) vec(P^{j+1}) = x_k^{\mathrm{T}} Q x_k + (u_k^j)^{\mathrm{T}} R u_k^j - \gamma^2 (w_k^j)^{\mathrm{T}} w_k^j \tag{4.20}$$

Here, (4.20) is a scalar equation and $P^{j+1} \in \mathbb{R}^{n \times n}$ is a symmetric matrix with $n \times (n+1)/2$ independent elements. Therefore, at least $n \times (n+1)/2$ data sets are required to solve (4.20) using least squares (LS). To solve (4.20), one has $\zeta\, vec(P^{j+1}) = \upsilon$ where $\zeta = [\zeta_1^{\mathrm{T}} \quad \cdots \quad \zeta_{q_1}^{\mathrm{T}}]^{\mathrm{T}}$ with $\zeta_i = x_{k+i-1}^{\mathrm{T}} \otimes x_{k+i-1}^{\mathrm{T}} - x_{k+i}^{\mathrm{T}} \otimes x_{k+i}^{\mathrm{T}}$, $\upsilon = [\upsilon_1^{\mathrm{T}} \quad \cdots \quad \upsilon_{q_1}^{\mathrm{T}}]^{\mathrm{T}}$ with $\upsilon_i = x_{k+i-1}^{\mathrm{T}} Q x_{k+i-1} + (u_{k+i-1}^j)^{\mathrm{T}} R u_{k+i-1}^j - \gamma^2 (w_{k+i-1}^j)^{\mathrm{T}} w_{k+i-1}^j$ and q_1 is the number of unknown elements of P^{j+1}. Matrix ζ must have independent rows.

The requirement of independent rows of these matrices is equivalent to the following condition.

Definition 4.2. A *q*-vector sequence $h = \begin{bmatrix} h_1 & \dots & h_q \end{bmatrix}^{\mathrm{T}}$ is said to be persistently exciting over an interval $[k+1, k+l]$ if for some constant $\beta > 0$

$$\sum_{i=k+1}^{k+l} h_i h_i^{\mathrm{T}} \ge \beta I \tag{4.21}$$

Note that if $l < q$, (4.21) cannot be satisfied.

To satisfy the persistence of excitation (PE) condition (4.21) in Algorithm 4.1, probing noise is added to the system dynamics (4.1). To this end, the actual control input which is applied to the system to collect data is considered as

$$\hat{u}_k^j = u_k^j + e_k \tag{4.22}$$

with e_k being a probing noise or dither and u_k^j given by (4.17). The following Lemma shows that probing noise may lead to incorrect solutions when solving the Bellman equation.

Lemma 4.1. *Effect of adding probing noise on PI Algorithm.*

Let P^{j+1} be the solution to (4.16) with $e_k = 0$ in (4.22) and $\hat{P}^{j+1}$ be the solution to (4.16) with $e_k \neq 0$ in (4.22). Then, $P^{j+1} \neq \hat{P}^{j+1}$.

Proof. Let (4.16) be the undithered Bellman equation with $e_k = 0$ in (4.22), i.e. $\hat{u}_k^j = u_k^j$. On the other hand, using (4.22) with $e_k \neq 0$ in (4.16), the dithered Bellman equation yields

$$\begin{aligned} &x_k^{\mathrm{T}} \hat{P}^{j+1} x_k = x_k^{\mathrm{T}} Q x_k + (\hat{u}_k^j)^{\mathrm{T}} R \hat{u}_k^j - \gamma^2 (w_k^j)^{\mathrm{T}} w_k^j + \hat{x}_{k+1}^{\mathrm{T}} \hat{P}^{j+1} \hat{x}_{k+1} = \\ &x_k^{\mathrm{T}} Q x_k + (u_k^j + e_k)^{\mathrm{T}} R (u_k^j + e_k) - \gamma^2 (w_k^j)^{\mathrm{T}} w_k^j + \\ &(A x_k + B u_k^j + B e_k + D w_k^j)^{\mathrm{T}} \hat{P}^{j+1} (A x_k + B u_k^j + B e_k + D w_k^j) \end{aligned} \tag{4.23}$$

By considering (4.1) in (4.23), one has

$$\begin{aligned} &x_k^{\mathrm{T}} \hat{P}^{j+1} x_k = x_k^{\mathrm{T}} Q x_k + (u_k^j)^{\mathrm{T}} R u_k^j - \gamma^2 (w_k^j)^{\mathrm{T}} w_k^j + x_{k+1}^{\mathrm{T}} \hat{P}^{j+1} x_{k+1} \\ &e_k^{\mathrm{T}} (R + B^{\mathrm{T}} \hat{P}^{j+1} B) e_k + 2 e_k^{\mathrm{T}} R u_k^j + 2 e_k^{\mathrm{T}} B^T \hat{P}^{j+1} x_{k+1} \end{aligned} \tag{4.24}$$

which is the undithered Bellman equation (4.16) plus three terms depending on probing noise. Then, P^{j+1} is not the same as $\hat{P}^{j+1}$. ∎

Remark 4.2. It is seen that on-policy PI Algorithm 4.1 actually solves (4.24) online and hence obtains an incorrect solution $\hat{P}^{j+1}$ that is not the desired solution P^{j+1} to the Bellman equation (4.16). Since $\hat{P}^{j+1} \neq P^{j+1}$, this result shows that the control update (4.17) may not be correct if probing noise is added to Algorithm 4.1.

Remark 4.3. The online PI Algorithm 4.1 needs complete knowledge of the system dynamics to obtain the optimal control input and the worst-case disturbance input. In [55], a Q-learning algorithm was presented to solve the H_∞ control problem online without any knowledge of the system dynamics.

Remark 4.4. In Algorithm 4.1, the disturbance input must be updated in the prescribed optimal fashion (4.18) and applied to system dynamics to collect data. However, in practical applications, the disturbance is independent and cannot be specified. This issue is fixed in Section 4.3.

Remark 4.5. Algorithm 4.1 is the standard method for solving the discrete-time H_∞ optimal control problem online using RL. Note that if disturbance $w_k = 0$, for example, Algorithm 4.1 is the basis for the Heuristic Dynamic Programming (HDP) algorithm in Approximate Dynamic Programming (ADP). Lemma 4.1 shows that all standard approaches based on Algorithm 4.1 are vulnerable to bias caused by probing noise**.**

4.3. Off-policy RL algorithm for solving zero-sum game problem

In this section, an off-policy RL algorithm is presented to solve the zero-sum game problem arising in the H_∞ control. It is shown furture that this off-policy algorithm does not suffer bias if probing noise is used.

4.3.1. Off-policy RL algorithm

To derive off-policy RL algorithm, the original system (4.1) is rewritten as

$$x_{k+1} = A_k x_k + B(K_1^j x_k + u_k) + D(K_2^j x_k + w_k) \tag{4.25}$$

where $A_k = A - BK_1^j - DK_2^j$.

In (4.25), the target policies are $u_k^j = -K_1^j x_k$ and $w_k^j = -K_2^j x_k$. They are the policies that are being learned and updated by the PI algorithm. By contrast, u_k and w_k are the behavior policies that are actually applied to the system dynamics (4.1) to generate data for learning.

For fixed policies u_k^j and w_k^j, the Bellman equation (4.7) yields

$$V^{j+1}(x_k, u_k, w_k) - V^{j+1}(x_{k+1}, u_k, w_k) = x_k^{\mathrm{T}} Q x_k + (u_k^j)^{\mathrm{T}} R u_k^j - \gamma^2 (w_k^j)^{\mathrm{T}} w_k^j \tag{4.26}$$

The Taylor expansion of the value function $V(x_k)$ at point x_{k+1} is

$$V(x_k) = V(x_{k+1}) + (\nabla V)^{\mathrm{T}}(x_{k+1})(x_k - x_{k+1}) + \frac{1}{2}(x_k - x_{k+1})^{\mathrm{T}} \nabla^2 V(x_{k+1})(x_k - x_{k+1}) \tag{4.27}$$

By using (4.8) and (4.27), the left-hand side of (4.26) becomes

$$V^{j+1}(x_k) - V^{j+1}(x_{k+1}) = 2x_{k+1}^{\mathrm{T}} P^{j+1}(x_k - x_{k+1}) + (x_k - x_{k+1})^{\mathrm{T}} P^{j+1}(x_k - x_{k+1}) \tag{4.28}$$

By substituting (4.25) in (4.28) and performing some manipulations, one has

$$\begin{aligned} &V^{j+1}(x_k) - V^{j+1}(x_{k+1}) = -x_k^{\mathrm{T}} A_k^{\mathrm{T}} P^{j+1} A_k x_k + x_k^{\mathrm{T}} P^{j+1} x_k \\ &-(u_k + K_1^j x_k)^{\mathrm{T}} B^{\mathrm{T}} P^{j+1} x_{k+1} - (u_k + K_1^j x_k)^{\mathrm{T}} B^{\mathrm{T}} P^{j+1} A_k x_k \\ &-(K_2^j x_k + w_k)^{\mathrm{T}} D^{\mathrm{T}} P^{j+1} x_{k+1} - (K_2^j x_k + w_k)^{\mathrm{T}} D^{\mathrm{T}} P^{j+1} A_k x_k \end{aligned} \tag{4.29}$$

On the other hand, the Bellman equation (4.9) can be written as the Lyapunov equation

$$Q - P^{j+1} + (K_1^j)^{\mathrm{T}} R K_1^j - \gamma^2 (K_2^j)^{\mathrm{T}} K_2^j + A_k^{\mathrm{T}} P^{j+1} A_k = 0 \tag{4.30}$$

Using (4.8) and (4.30) in (4.29) yields the following off-policy H_∞ Bellman equation (4.31). We now show that this equation can be iteratively solved to find the solution to the GARE (4.15) which gives the following off-policy RL algorithm.

Algorithm 4.2. Model-based off-policy RL

Initialization: Set the iteration number $j = 0$ and start with a stabilizing control policy u_k.

1. Solve the following off-policy Bellman equation for $(P^{j+1}, K_1^{j+1}, K_2^{j+1})$ simultaneously

$$\begin{aligned} &x_k^{\mathrm{T}} P^{j+1} x_k - x_{k+1}^{\mathrm{T}} P^{j+1} x_{k+1} = x_k^{\mathrm{T}} Q x_k + x_k^{\mathrm{T}} (K_1^j)^{\mathrm{T}} R K_1^j x_k - \gamma^2 x_k^{\mathrm{T}} (K_2^j)^{\mathrm{T}} K_2^j x_k \\ &-(u_k + K_1^j x_k)^{\mathrm{T}} B^{\mathrm{T}} P^{j+1} x_{k+1} - (u_k + K_1^j x_k)^{\mathrm{T}} B^{\mathrm{T}} P^{j+1} A_k x_k \\ &-(K_2^j x_k + w_k)^{\mathrm{T}} D^{\mathrm{T}} P^{j+1} x_{k+1} - (K_2^j x_k + w_k)^{\mathrm{T}} D^{\mathrm{T}} P^{j+1} A_k x_k \end{aligned} \tag{4.31}$$

2. Stop if

$$\left| K_1^{j+1} - K_1^j \right| \le \varepsilon \quad \text{and} \quad \left| K_2^{j+1} - K_2^j \right| \le \varepsilon \tag{4.32}$$

for a small positive value of ε, otherwise set $j = j + 1$ and go to 1.

Note that (4.31) does not explicitly depend on K_1^{j+1} and K_2^{j+1}. Also, the complete knowledge of the system dynamics is required for solving (4.31). In Subsection 4.3.2, it is shown in Algorithm 4.3 how to find $(P^{j+1}, K_1^{j+1}, K_2^{j+1})$ simultaneously by (4.31) without requiring any knowledge of the system dynamics.

The function of the on-policy Algorithm 4.1 is to solve the GARE (4.15) online in real-time. The next results show that Algorithm 4.2 also solves the GARE (4.15).

Theorem 4.1. On-policy Algorithm 4.1 and off-policy Algorithm 4.2 are equivalent in the sense that (4.16) and (4.31) are equivalent.

Proof. Substituting $A_k = A - BK_1^j - DK_2^j$ and system dynamics (4.1) in the off-policy Bellman equation (4.31), yields

$$\begin{aligned} &x_k^{\mathrm{T}} P^{j+1} x_k - (Ax_k + Bu_k + Dw_k)^{\mathrm{T}} P^{j+1} (Ax_k + Bu_k + Dw_k) = x_k^{\mathrm{T}} Q\, x_k + x_k^{\mathrm{T}} (K_1^j)^{\mathrm{T}} R\, K_1^j x_k - \gamma^2 x_k^{\mathrm{T}} (K_2^j)^{\mathrm{T}} K_2^j x_k \\ &-(u_k + K_1^j x_k)^{\mathrm{T}} B^{\mathrm{T}} P^{j+1} (Ax_k + Bu_k + Dw_k) - (u_k + K_1^j x_k)^{\mathrm{T}} B^{\mathrm{T}} P^{j+1} (A - BK_1^j - DK_2^j) x_k \\ &-(w_k + K_2^j x_k)^{\mathrm{T}} D^{\mathrm{T}} P^{j+1} (Ax_k + Bu_k + Dw_k) - (w_k + K_2^j x_k)^{\mathrm{T}} D^{\mathrm{T}} P^{j+1} (A - BK_1^j - DK_2^j) x_k \end{aligned} \tag{4.33}$$

By eliminating the common terms in the left hand and right hand side of (4.33), one has

$$\begin{aligned} &x_k^{\mathrm{T}} P^{j+1} x_k - x_k^{\mathrm{T}} A^{\mathrm{T}} P^{j+1} A^{\mathrm{T}} x_k = x_k^{\mathrm{T}} Q\, x_k + x_k^{\mathrm{T}} (K_1^j)^{\mathrm{T}} R\, K_1^j x_k - \gamma^2 x_k^{\mathrm{T}} (K_2^j)^{\mathrm{T}} K_2^j x_k \\ &+x_k^{\mathrm{T}} (K_1^j)^{\mathrm{T}} B^{\mathrm{T}} P^{j+1} B\, K_1^j x_k + x_k^{\mathrm{T}} (K_1^j)^{\mathrm{T}} B^{\mathrm{T}} P^{j+1} D\, K_2^j x_k \\ &-2x_k^{\mathrm{T}} (K_2^j)^{\mathrm{T}} D^{\mathrm{T}} P^{j+1} A\, x_k + x_k^{\mathrm{T}} (K_2^j)^{\mathrm{T}} D^{\mathrm{T}} P^{j+1} B\, K_1^j x_k \\ &-2x_k^{\mathrm{T}} (K_1^j)^{\mathrm{T}} B^{\mathrm{T}} P^{j+1} A\, x_k + x_k^{\mathrm{T}} (K_2^j)^{\mathrm{T}} D^{\mathrm{T}} P^{j+1} D\, K_2^j x_k \end{aligned} \tag{4.34}$$

Eq. (4.34) can be rewritten as

$$\begin{aligned} &x_k^{\mathrm{T}} Q\, x_k + x_k^{\mathrm{T}} (K_1^j)^{\mathrm{T}} R\, K_1^j x_k - \gamma^2 x_k^{\mathrm{T}} (K_2^j)^{\mathrm{T}} K_2^j x_k - x_k^{\mathrm{T}} P^{j+1} x_k \\ &+x_k^{\mathrm{T}} (A - BK_1^j - DK_2^j)^{\mathrm{T}} P^{j+1} (A - BK_1^j - DK_2^j) x_k = 0 \end{aligned} \tag{4.35}$$

which is equal to (4.16). Therefore, the on-policy Bellman equation (4.16) in Algorithm 4.1 and the off-policy Bellman equation (4.31) in Algorithm 4.2 are equivalent and the proof is completed.

■

Remark 4.6. In off-policy RL Algorithm 4.2, behavior policies applied to the system do not need to be the same as target policies which are improved and updated. In fact, in the proposed Algorithm 4.2, the policies u_k and w_k are behavior policies applied to the system dynamics (4.1) to collect data, while the policies $u_k^j = -K_1^j x_k$ and $w_k^j = -K_2^j x_k$ are the target policies and updated using measured data generated from the policies u_k and w_k. In Algorithm (4.2), u_k is assumed to be a stabilizing exploratory control policy and w_k is the xternal disturbance which is applied to

the system. Therefore, the actual disturbance w_k applied to the system is not required to be updated in a prescribed manner according to $w_k^j = -K_2^j x_k$. This makes the proposed algorithm more practical than standard methods based on Algorithm 4.1.

Theorem 4.2. Convergence of the off-policy RL Algorithm 4.2.

The off-policy RL Algorithm 4.2 converges to the optimal control solution given by (4.13) and (4.14) where the matrix *P* satisfies the GARE (4.15).

Proof. The convergence is shown in two steps:

Step 1. In Theorem 4.1, it is shown that the off-policy Algorithm 4.2 is equivalent to Algorithm 4.1 at every iteration j.

Step 2. Substituting updated policies (4.17) and (4.18) into (4.35) yields

$$P^{j+1} = A^{\mathrm{T}} P^{j+1} A + Q - \begin{bmatrix} A^{\mathrm{T}} P^j B & A^{\mathrm{T}} P^j D \end{bmatrix} \begin{bmatrix} R + B^{\mathrm{T}} P^j B & B^{\mathrm{T}} P^j D \\ D^{\mathrm{T}} P^j B & D^{\mathrm{T}} P^j D - \gamma^2 I \end{bmatrix}^{-1} \begin{bmatrix} B^{\mathrm{T}} P^j A \\ D^{\mathrm{T}} P^j A \end{bmatrix} \tag{4.36}$$

By using the result of Theorem 4.1, it can be concluded that iterating on (4.31) is equivalent to iterating on (4.36). In [102], it is shown that iterating on (4.36) converges to the solution of GARE (4.15). Therefore, Algorithm 4.2 converges to the optimal solution. ■

To satisfy the PE condition, probing noise must be added to solve (4.16) in Algorithm 4.1 and (4.31) in Algorithm 4.2. Lemma 4.1 shows that this may result in incorrect solutions in Algorithm 4.1. The next result shows that adding probing noise does not lead to incorrect solutions while solving the Bellman equation (4.31) in the proposed Algorithm 4.2.

Theorem 4.3. Effect of adding probing noise on off-policy RL Algorithm 4.2.

Let $\hat{P}^{j+1}$ be the solution to (4.31) with $\hat{u}_k = u_k + e_k$ where $e_k \neq 0$ is the probing noise and $\bar{P}^{j+1}$ be the solution to (4.31) with $\bar{u}_k = u_k$. Then $\hat{P}^{j+1} = \bar{P}^{j+1}$.

Proof.

1. The off-policy Bellman equation (4.31) for control input $\hat{u}_k$ *is*

$$
\begin{aligned}
&x_k^{\mathrm{T}}\hat{P}^{j+1}x_k-(A_kx_k+B(K_1^jx_k+\hat{u}_k)+D(K_2^jx_k+w_k))^{\mathrm{T}}\hat{P}^{j+1}\times(A_kx_k+B(K_1^jx_k+\hat{u}_k)+D(K_2^jx_k+w_k))=\\
&x_k^{\mathrm{T}}Qx_k+x_k^{\mathrm{T}}(K_1^j)^{\mathrm{T}}RK_1^jx_k-\gamma^2x_k^{\mathrm{T}}(K_2^j)^{\mathrm{T}}K_2^jx_k-(\hat{u}_k+K_1^jx_k)^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}(A_kx_k+B(K_1^jx_k+\hat{u}_k)+D(K_2^jx_k+w_k))\\
&-(\hat{u}_k+K_1^jx_k)^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}A_kx_k-(K_2^jx_k+w_k)^{\mathrm{T}}D^{\mathrm{T}}\hat{P}^{j+1}A_kx_k-(K_2^jx_k+w_k)^{\mathrm{T}}D^{\mathrm{T}}\hat{P}^{j+1}\times\\
&(A_kx_k+B(K_1^jx_k+\hat{u}_k)+D(K_2^jx_k+w_k))
\end{aligned}
\tag{4.37}
$$

Substituting $\hat{u}_k=u_k+e_k$ into (4.37) yields

$$
\begin{aligned}
&x_k^{\mathrm{T}}\hat{P}^{j+1}x_k-(A_kx_k+B(K_1^jx_k+u_k+e_k)+D(K_2^jx_k+w_k))^{\mathrm{T}}\hat{P}^{j+1}\\
&\times(A_kx_k+B(K_1^jx_k+u_k+e_k)+D(K_2^jx_k+w_k))=\\
&x_k^{\mathrm{T}}Qx_k+x_k^{\mathrm{T}}(K_1^j)^{\mathrm{T}}RK_1^jx_k-\gamma^2x_k^{\mathrm{T}}(K_2^j)^{\mathrm{T}}K_2^jx_k-(u_k+e_k+K_1^jx_k)^{\mathrm{T}}\\
&\times B^{\mathrm{T}}\hat{P}^{j+1}(A_kx_k+B(K_1^jx_k+u_k+e_k)+D(K_2^jx_k+w_k))\\
&-(u_k+e_k+K_1^jx_k)^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}A_kx_k-(K_2^jx_k+w_k)^{\mathrm{T}}D^{\mathrm{T}}\hat{P}^{j+1}A_kx_k\\
&-(K_2^jx_k+w_k)^{\mathrm{T}}D^{\mathrm{T}}\hat{P}^{j+1}(A_kx_k+B(K_1^jx_k+u_k+e_k)+D(K_2^jx_k+w_k))
\end{aligned}
\tag{4.38}
$$

Substituting (4.25) into (4.38) yields

$$
\begin{aligned}
&x_k^{\mathrm{T}}\hat{P}^{j+1}x_k-(x_{k+1}+Be_k)^{\mathrm{T}}\hat{P}^{j+1}(x_{k+1}+Be_k)=x_k^{\mathrm{T}}Qx_k+x_k^{\mathrm{T}}(K_1^j)^{\mathrm{T}}RK_1^jx_k-\gamma^2x_k^{\mathrm{T}}(K_2^j)^{\mathrm{T}}K_2^jx_k\\
&-(u_k+e_k+K_1^jx_k)^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}(x_{k+1}+Be_k)\\
&-(u_k+e_k+K_1^jx_k)^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}A_kx_k-(K_2^jx_k+w_k)^{\mathrm{T}}D^{\mathrm{T}}\hat{P}^{j+1}A_kx_k\\
&-(K_2^jx_k+w_k)^{\mathrm{T}}D^{\mathrm{T}}\hat{P}^{j+1}(x_{k+1}+Be_k)
\end{aligned}
\tag{4.39}
$$

Expanding terms in both sides of (4.39) yields

$$
\begin{aligned}
&x_k^{\mathrm{T}}\hat{P}^{j+1}x_k-x_{k+1}^{\mathrm{T}}\hat{P}^{j+1}x_{k+1}-2x_{k+1}^{\mathrm{T}}\hat{P}^{j+1}Be_k-e_k^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}Be_k=x_k^{\mathrm{T}}Qx_k+x_k^{\mathrm{T}}(K_1^j)^{\mathrm{T}}RK_1^jx_k-\gamma^2x_k^{\mathrm{T}}(K_2^j)^{\mathrm{T}}K_2^jx_k\\
&-(u_k+K_1^jx_k)^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}x_{k+1}-(u_k+K_1^jx_k)^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}Be_k-x_{k+1}^{\mathrm{T}}\hat{P}^{j+1}Be_k-e_k^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}Be_k-(u_k+K_1^jx_k)^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}A_kx_k\\
&-e_k^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}A_kx_k-(K_2^jx_k+w_k)^{\mathrm{T}}D^{\mathrm{T}}\hat{P}^{j+1}A_kx_k-(K_2^jx_k+w_k)^{\mathrm{T}}D^{\mathrm{T}}\hat{P}^{j+1}x_{k+1}-(K_2^jx_k+w_k)^{\mathrm{T}}D^{\mathrm{T}}\hat{P}^{j+1}Be_k
\end{aligned}
\tag{4.40}
$$

Eliminating the common terms and considering

$$
x_{k+1}^{\mathrm{T}}\hat{P}^{j+1}Be_k=x_k^{\mathrm{T}}A_k^{\mathrm{T}}\hat{P}^{j+1}Be_k+(u_k+K_1^jx_k)^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}Be_k+(K_2^jx_k+w_k)^{\mathrm{T}}D^{\mathrm{T}}\hat{P}^{j+1}Be_k \tag{4.41}
$$

in (4.40) yield

$$
\begin{aligned}
&x_k^{\mathrm{T}}\hat{P}^{j+1}x_k-x_{k+1}^{\mathrm{T}}\hat{P}^{j+1}x_{k+1}=x_k^{\mathrm{T}}Qx_k+x_k^{\mathrm{T}}(K_1^j)^{\mathrm{T}}RK_1^jx_k-\gamma^2x_k^{\mathrm{T}}(K_2^j)^{\mathrm{T}}K_2^jx_k\\
&-(u_k+K_1^jx_k)^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}x_{k+1}-(u_k+K_1^jx_k)^{\mathrm{T}}B^{\mathrm{T}}\hat{P}^{j+1}A_kx_k\\
&-(K_2^jx_k+w_k)^{\mathrm{T}}D^{\mathrm{T}}\hat{P}^{j+1}x_{k+1}-(K_2^jx_k+w_k)^{\mathrm{T}}D^{\mathrm{T}}\hat{P}^{j+1}A_kx_k
\end{aligned}
\tag{4.42}
$$

$\hat{P}^{j+1}$ is obtained by solving (4.42).

2. Substituting $\bar{u}_k = u_k$ into the off-policy Bellman equation (4.31) yields

$$\begin{aligned} & x_k^{\mathrm{T}} \bar{P}^{j+1} x_k - x_{k+1}^{\mathrm{T}} \bar{P}^{j+1} x_{k+1} = x_k^{\mathrm{T}} Q x_k + x_k^{\mathrm{T}} (K_1^j)^{\mathrm{T}} R K_1^j x_k - \gamma^2 x_k^{\mathrm{T}} (K_2^j)^{\mathrm{T}} K_2^j x_k \\ & -(u_k + K_1^j x_k)^{\mathrm{T}} B^{\mathrm{T}} \bar{P}^{j+1} x_{k+1} - (u_k + K_1^j x_k)^{\mathrm{T}} B^{\mathrm{T}} \bar{P}^{j+1} A_k x_k \\ & -(K_2^j x_k + w_k)^{\mathrm{T}} D^{\mathrm{T}} \bar{P}^{j+1} x_{k+1} - (K_2^j x_k + w_k)^{\mathrm{T}} D^{\mathrm{T}} \bar{P}^{j+1} A_k x_k \end{aligned} \tag{4.43}$$

$\bar{P}^{j+1}$ is obtained by solving (4.43).

By comparing (4.42) and (4.43), it can be concluded that is the same as $\bar{P}^{j+1}$. Therefore, the off-policy Bellman equation (4.31) is insensitive to probing noise. ■

Remark 4.7. It was discussed in Section 4.2 that Algorithm 4.1 may result in a bias. This is because the control policy $\hat{u}_k^j = -K_1^j x_k + e_k$ is applied to the system dynamics to generate data while the value function is estimated for the control policy $u_k^j = -K_1^j x_k$. Therefore, the measured state and input data used for learning are generated by a slightly different policy than the one under evaluation (while they are supposed to be the same in on-policy), which may cause a bias. On the other hand, Algorithm 4.2 is an off-policy RL algorithm, which allows us to separate the behavior policies (μ_k, w_k) and target policies ($u_k^j = -K_1^j x_k$, $w_k^j = -K_2^j x_k$). Since the behavior policy is an arbitrary policy and unrelated to the estimated policy, as shown in Theorem 4.3, probing noise does not affect the estimation of the value for the policy under evaluation. In fact, the probing noise added to the behavior policy is explicitly incorporated when solving the Bellman equation which leads to eliminating the bias.

4.3.2. Obtaining the optimal control input and the disturbance without system dynamics

Algorithm 4.2 requires the system dynamics to solve (4.31). In this section, the solution of Bellman equation (4.31) for $(P^{j+1}, K_1^{j+1}, K_2^{j+1})$ is presented in Algorithm 4.3. This solution does not require any knowledge of the system dynamics.

Based on Kronecker product, one has

$$a^{\mathrm{T}} W\, b = (b^{\mathrm{T}} \otimes a^{\mathrm{T}}) vec(W) \tag{4.44}$$

with vectors $a \in \mathbb{R}^{va}, b \in \mathbb{R}^{vb}$, and matrix $W \in \mathbb{R}^{va \times vb}$. Then, by using (4.1), (4.44), and $A_k = A - BK_1^j - DK_2^j$, the off-policy Bellman equation (4.31) can be rewritten as

$$\begin{aligned}
&(x_k^{\mathrm{T}} \otimes x_k^{\mathrm{T}}) vec(P^{j+1}) - (x_{k+1}^{\mathrm{T}} \otimes x_{k+1}^{\mathrm{T}}) vec(P^{j+1}) + 2(x_k^{\mathrm{T}} \otimes (u_k + K_1^j x_k)^{\mathrm{T}}) vec(B^{\mathrm{T}} P^{j+1} A) \\
&-((K_1^j x_k - u_k)^{\mathrm{T}} \otimes (u_k + K_1^j x_k)^{\mathrm{T}}) vec(B^{\mathrm{T}} P^{j+1} B) + 2(x_k^{\mathrm{T}} \otimes (w_k + K_2^j x_k)^{\mathrm{T}}) vec(D^{\mathrm{T}} P^{j+1} A) \\
&-((K_1^j x_k - u_k)^{\mathrm{T}} \otimes (w_k + K_2^j x_k)^{\mathrm{T}}) vec(D^{\mathrm{T}} P^{j+1} B) + ((w_k - K_2^j x_k)^{\mathrm{T}} \otimes (u_k + K_1^j x_k)^{\mathrm{T}}) vec(B^{\mathrm{T}} P^{j+1} D) \\
&+((w_k - K_2^j x_k)^{\mathrm{T}} \otimes (w_k + K_2^j x_k)^{\mathrm{T}}) vec(D^{\mathrm{T}} P^{j+1} D) = x_k^{\mathrm{T}} Q x_k + x_k^{\mathrm{T}} (K_1^j)^{\mathrm{T}} R K_1^j x_k - \gamma^2 x_k^{\mathrm{T}} (K_2^j)^{\mathrm{T}} K_2^j x_k
\end{aligned} \tag{4.45}$$

Using LS method, the unique solution $(P^{j+1}, K_1^{j+1}, K_2^{j+1})$ can be obtained simultaneously and without any knowledge of the system dynamics. The Bellman equation (4.45) has $n^2 + m_1^2 + m_2^2 + 2m_1 m_2 + n(m_1 + m_2)$ unknown parameters. Therefore, at least $n^2 + m_1^2 + m_2^2 + 2m_1 m_2 + n(m_1 + m_2)$ data sets are required before (4.45) can be solved using LS at each iteration. For the positive integer $s \geq n^2 + m_1^2 + m_2^2 + 2m_1 m_2 + n(m_1 + m_2)$, one defines

$$\phi^j = \begin{bmatrix} x_k^{\mathrm{T}} Q\, x_k + x_k^{\mathrm{T}} (K_1^j)^{\mathrm{T}} R K_1^j x_k - \gamma^2 x_k^{\mathrm{T}} (K_2^j)^{\mathrm{T}} K_2^j x_k \\ x_{k+1}^{\mathrm{T}} Q\, x_{k+1} + x_{k+1}^{\mathrm{T}} (K_1^j)^{\mathrm{T}} R K_1^j x_{k+1} - \gamma^2 x_{k+1}^{\mathrm{T}} (K_2^j)^{\mathrm{T}} K_2^j x_{k+1} \\ \vdots \\ x_{k+s-1}^{\mathrm{T}} Q\, x_{k+s-1} + x_{k+s-1}^{\mathrm{T}} (K_1^j)^{\mathrm{T}} R K_1^j x_{k+s-1} - \gamma^2 x_{k+s-1}^{\mathrm{T}} (K_2^j)^{\mathrm{T}} K_2^j x_{k+s-1} \end{bmatrix} \tag{4.46}$$

$$\psi^j = \begin{bmatrix} \mathrm{H}_{(xx)1} & \mathrm{H}_{(xu)1} & \mathrm{H}_{(uu)1} & \mathrm{H}_{(xw)1} & \mathrm{H}_{(uw)1} & \mathrm{H}_{(wu)1} & \mathrm{H}_{(ww)1} \\ \mathrm{H}_{(xx)2} & \mathrm{H}_{(xu)2} & \mathrm{H}_{(uu)2} & \mathrm{H}_{(xw)2} & \mathrm{H}_{(uw)2} & \mathrm{H}_{(wu)2} & \mathrm{H}_{(ww)2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathrm{H}_{(xx)s} & \mathrm{H}_{(xu)s} & \mathrm{H}_{(uu)s} & \mathrm{H}_{(xw)s} & \mathrm{H}_{(uw)s} & \mathrm{H}_{(wu)s} & \mathrm{H}_{(ww)s} \end{bmatrix} \tag{4.47}$$

where

$$\begin{aligned}
\mathrm{H}_{(xx)\mathrm{i}} &= x_{k+i-1}^{\mathrm{T}} \otimes x_{k+i-1}^{\mathrm{T}} - x_{k+\mathrm{i}}^{\mathrm{T}} \otimes x_{k+\mathrm{i}}^{\mathrm{T}} \\
\mathrm{H}_{(xu)\mathrm{i}} &= 2(x_{k+i-1}^{\mathrm{T}} \otimes (u_{k+i-1} + K_1^j x_{k+i-1})^{\mathrm{T}}) \\
\mathrm{H}_{(uu)\mathrm{i}} &= -(K_1^j x_{k+i-1} - u_{k+i-1})^{\mathrm{T}} \otimes (u_{k+i-1} + K_1^j x_{k+i-1})^{\mathrm{T}} \\
\mathrm{H}_{(xw)\mathrm{i}} &= 2(x_{k+i-1}^{\mathrm{T}} \otimes (w_{k+i-1} + K_2^j x_{k+i-1})^{\mathrm{T}}) \\
\mathrm{H}_{(uw)\mathrm{i}} &= -(K_1^j x_{k+i-1} - u_{k+i-1})^{\mathrm{T}} \otimes (w_{k+i-1} + K_2^j x_{k+i-1})^{\mathrm{T}} \\
\mathrm{H}_{(wu)\mathrm{i}} &= (w_{k+i-1} - K_2^j x_{k+i-1})^{\mathrm{T}} \otimes (u_{k+i-1} + K_1^j x_{k+i-1})^{\mathrm{T}} \\
\mathrm{H}_{(ww)\mathrm{i}} &= (w_{k+i-1} - K_2^j x_{k+i-1})^{\mathrm{T}} \otimes (w_{k+i-1} + K_2^j x_{k+i-1})^{\mathrm{T}}
\end{aligned}$$

Define the unknown variables in the Bellman equation (4.45), which the control input and disturbance input gains depend on, as

$$L_1{}^{j+1} = P^{j+1} \ , \ L_2{}^{j+1} = B^T P^{j+1} A \ , \ L_3{}^{j+1} = B^T P^{j+1} B \ , \ L_4{}^{j+1} = D^T P^{j+1} A$$
$$L_5{}^{j+1} = D^T P^{j+1} B \ , \ L_6{}^{j+1} = B^T P^{j+1} D \ , \ L_7{}^{j+1} = D^T P^{j+1} D$$

Then, using (4.45)-(4.47), one has

$$\begin{aligned} \psi^j \Big[& vec(L_1{}^{j+1})^{\mathrm{T}} \ vec(L_2{}^{j+1})^{\mathrm{T}} \ vec(L_3{}^{j+1})^{\mathrm{T}} \ vec(L_4{}^{j+1})^{\mathrm{T}} \\ & vec(L_5{}^{j+1})^{\mathrm{T}} \ vec(L_6{}^{j+1})^{\mathrm{T}} \ vec(L_7{}^{j+1})^{\mathrm{T}} \Big]^{\mathrm{T}} = \phi^j \end{aligned} \tag{4.48}$$

The equation (4.48) can be solved by LS method as

$$\begin{aligned} & \Big[vec(L_1{}^{j+1})^{\mathrm{T}} \ vec(L_2{}^{j+1})^{\mathrm{T}} \ vec(L_3{}^{j+1})^{\mathrm{T}} \ vec(L_4{}^{j+1})^{\mathrm{T}} \\ & vec(L_5{}^{j+1})^{\mathrm{T}} \ vec(L_6{}^{j+1})^{\mathrm{T}} \ vec(L_7{}^{j+1})^{\mathrm{T}} \Big]^{\mathrm{T}} = ((\psi^j)^{\mathrm{T}} \psi^j)^{-1} (\psi^j)^{\mathrm{T}} \phi^j \end{aligned} \tag{4.49}$$

Note that in (4.49), ψ^j and ϕ^j are known matrices and $L_1{}^{j+1}$ through $L_7{}^{j+1}$ are unknown values. This solution requires full rank of (4.47) which amounts to the PE condition and requires at least *s* time steps. *s* is a positive integer which is at least equal to the number of unknown parameters of the off-policy Bellman equation (4.45). That is, $s \geq n^2 + m_1^2 + m_2^2 + 2m_1 m_2 + n(m_1 + m_2)$.

Using the solution of (4.49) for $L_1{}^{j+1}$ through $L_7{}^{j+1}$, (4.17), and (4.18), the gains K_1^{j+1} and K_2^{j+1} can be obtained as

$$K_1^{j+1} = (R + L_3{}^{j+1} + L_6{}^{j+1} (\gamma^2 I - L_7{}^{j+1})^{-1} L_5{}^{j+1})^{-1} \Big[L_2{}^{j+1} + L_6{}^{j+1} (\gamma^2 I - L_7{}^{j+1})^{-1} L_4{}^{j+1} \Big] \tag{4.50}$$

$$K_2^{j+1} = (L_7{}^{j+1} - \gamma^2 I - L_5{}^{j+1} (R + L_3{}^{j+1})^{-1} L_6{}^{j+1})^{-1} \Big[L_4{}^{j+1} - L_5{}^{j+1} (R + L_3{}^{j+1})^{-1} L_2{}^{j+1} \Big] \tag{4.51}$$

The following off-policy algorithm uses LS (4.49) and update laws (4.50) and (4.51) to find the solution to the H_∞ control problem of linear discrete-time systems.

Algorithm 4.3. Model-free off-policy RL

Initialization: Set the iteration number j=0 and start with a stabilizing control policy $u_k = -K_1 x + e_k$ where e_k is probing noise.

1. For j=0,1,2, …., solve (4.52) to obtain ${L_i}^{j+1}$, i=1,…, 7 using LS

$$\psi^j \left[vec({L_1}^{j+1})^{\mathrm{T}} \; vec({L_2}^{j+1})^{\mathrm{T}} \; vec({L_3}^{j+1})^{\mathrm{T}} \; vec({L_4}^{j+1})^{\mathrm{T}} \; vec({L_5}^{j+1})^{\mathrm{T}} \; vec({L_6}^{j+1})^{\mathrm{T}} \; vec({L_7}^{j+1})^{\mathrm{T}} \right]^{\mathrm{T}} = \phi^j \tag{4.52}$$

2. Update the control and disturbance gains using learned gains ${L_1}^{j+1}$ through ${L_7}^{j+1}$

$$K_1^{j+1} = (R + {L_3}^{j+1} + {L_6}^{j+1}(\gamma^2 I - {L_7}^{j+1})^{-1} {L_5}^{j+1})^{-1} \left[{L_2}^{j+1} + {L_6}^{j+1}(\gamma^2 I - {L_7}^{j+1})^{-1} {L_4}^{j+1} \right] \tag{4.53}$$

$$K_2^{j+1} = ({L_7}^{j+1} - \gamma^2 I - {L_5}^{j+1}(R + {L_3}^{j+1})^{-1} {L_6}^{j+1})^{-1} \left[{L_4}^{j+1} - {L_5}^{j+1}(R + {L_3}^{j+1})^{-1} {L_2}^{j+1} \right] \tag{4.54}$$

3. Stop if

$$\left| K_1^{j+1} - K_1^j \right| \le \varepsilon \quad \text{and} \quad \left| K_2^{j+1} - K_2^j \right| \le \varepsilon \tag{4.55}$$

for a small positive value of ε, otherwise set $j = j + 1$ and go to 1. ■

Remark 4.8. The proposed off-policy RL Algorithm 4.3 iteratively solves (4.49). This iterative algorithm does not require any knowledge of the system dynamics. The cost of control and disturbance policies are evaluated using measured data along the system trajectories. In fact, the algorithm has two steps. In the first step, in (4.52), the gains ${L_1}^{j+1}$ through ${L_7}^{j+1}$ are found using measured data ψ^j and ϕ^j (see (4.46) and (4.47)). In the second step, the control and disturbance policies are updated using the gains learned in the first step. Therefore, no knowledge of the system dynamics is required. Moreover, the disturbance policy which is specified and updated in (4.54) does not need to be applied to the system. This is in contrast to the existing RL and Q-learning methods that require this specified disturbance policy be applied to the system, which is not practical as the disturbance applied to the system cannot be specified.

4.4. Simulation Results

In this section, the proposed scheme is used for control of an F-16 aircraft autopilot. Four cases are considered to show the effect of probing noise. In cases 1 through 3, different magnitudes of probing noise are considered with fix frequencies. In case 4, the frequencies of probing noise are changed. It is seen that the new off-policy H_∞ algorithm is insensitive to both magnitude and frequency of probing noise and always converges.

The F-16 short period dynamics has three states given as $x = \begin{bmatrix} \alpha & q & \delta_e \end{bmatrix}^{\mathrm{T}}$ where α is the angle of attack, *q* is the pitch rate, and δ_e is the elevator deflection angle. The discrete-time plant model of this aircraft dynamics is

$$x_{k+1} = Ax_k + Bu_k + Dw_k \tag{4.56}$$

where

$$A = \begin{bmatrix} 0.906488 & 0.0816012 & -0.0005 \\ 0.074349 & 0.90121 & -0.000708383 \\ 0 & 0 & 0.132655 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.00150808 \\ -0.0096 \\ 0.867345 \end{bmatrix} \qquad D = \begin{bmatrix} 0.00951892 \\ 0.00038373 \\ 0 \end{bmatrix}$$

The performance index is considered as (4.4) with $Q = diag(1,1,1)$, $R = I$, and the disturbance attenuation $\gamma = 1$. Using (4.13) for the optimal control $u_k^* = -K_1^* x_k$ and for the worst-case disturbance $w_k^* = -K_2^* x_k$, the gains K_1^* and K_2^* are given as

$$K_1^* = [-0.0842 \;\; -0.0961 \;\; 0.0661]$$
$$K_2^* = [-0.1477 \;\; -0.1244 \;\; 0]$$

Now the results of the proposed off-policy RL Algorithm 4.3 are given. The model-free off-policy RL Algorithm 4.3 is implemented as in (4.52)-(4.55). It is assumed that the dynamics

A, B, and *D* are completely unknown. The initial state and the initial gains are chosen as

$$x_0 = \begin{bmatrix} 10 & -10 & -3 \end{bmatrix}^{\mathrm{T}}$$
$$K_1 = \begin{bmatrix} 3 & 2.5 & 1.1 \end{bmatrix}$$
$$K_2 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

In each iteration, 25 data samples are collected to perform the LS solution of the Bellman equations.

Case 1: The probing noise is considered as

$$e_k = 0.2\sin(1.009k) + \cos^2(0.538k) + \sin(0.9k) + \cos(100k)$$

After 5 iterations, the control and disturbance gains converge to

$$K_1 = [-0.0844 \ \ -0.096 \ \ 0.066]$$
$$K_2 = [-0.1477 \ \ -0.1244 \ \ 0]$$

Fig. 4.1 shows norm of the difference of the optimal control K_1^* gain and disturbance gain K_2^* and the computed their values during the learning process. Fig. 4.2 shows the states of the system during and after learning with probing noise added up to time step 400. The probing noise is turned off after 400 time steps and the optimal control solution found by learning makes all states go to zeto.

Case 2: The probing noise is increased as

$$e_k = \sin(1.009k) + \cos^2(0.538k) + \sin(0.9k) + \cos(100k)$$

After 5 iterations, the control gain and the disturbance gain converge to

$$K_1 = [-0.0841 \ \ -0.096 \ \ 0.066], K_2 = [-0.1477 \ \ -0.1244 \ \ 0]$$

Fig. 4.3 shows norm of the difference of the optimal control and disturbance gains K_1^* and K_2^* and the computed their values during the learning process. Fig. 4.4 shows the states of the system during and after learning with probing noise added up to time step 400. The probing noise is turned off after 400 time steps and the optimal control solution found by learning makes all states go to zeto.

Case 3: The probing noise is increased as

$$e_k = 4\sin(1.009k) + \cos^2(0.538k) + \sin(0.9k) + \cos(100k)$$

After 4 iterations, the control gain and the disturbance gain converge to

$$K_1 = [-0.0841 \ \ -0.096 \ \ 0.066]$$
$$K_2 = [-0.1476 \ \ -0.1245 \ \ 0]$$

Fig. 4.5 shows norm of the difference of the optimal control and disturbance gains K_1^* and K_2^* and the computed their values during the learning process. In Fig. 4.6, the states of the system are shown during and after learning. The algorithm converges.

Case 4: The frequencies of probing noise is changed as

$$e_k = 1\sin(9.7k) + \cos^2(10.2k) + \sin(10k) + \cos(10k)$$

The control gain and the disturbance gain converge to

$$K_1 = [-0.0841 \ \ -0.096 \ \ 0.066]$$
$$K_2 = [-0.1477 \ \ -0.1244 \ \ 0]$$

Fig. 4.7 shows norm of the difference of the optimal control and disturbance gains K_1^* and K_2^* and the computed their values during the learning process. In Fig. 4.8, the states of the system is shown during and after learning.

Fig. 4.9 shows the attenuation

$$\sum_{k=0}^{\infty}\left[x_k^{\mathrm{T}} Q\, x_k + u_k^{\mathrm{T}} R\, u_k\right] \Big/ \sum_{k=0}^{\infty} w_k^{\mathrm{T}} w_k$$

for the optimal control input and $w_k = \sin(k)e^{-0.01*k}$. It can be seen that the disturbance attenuation condition (4.2) is satisfied.

Remark 4.9. From the results of cases 1 to 4, it can be concluded the proposed off-policy algorithm converges to the optimal solution regardless of the level and frequency of the probing noise. This is in contrast to other model-free but on-policy RL approaches. This is because, if the magnitude of the probing noise is too small, the PE condition may not be satisfied and if the

magnitude of the probing noise is too large, its covariance is increased and then based on Lemma 4.1, deleterious effect of the probing noise can be increased.

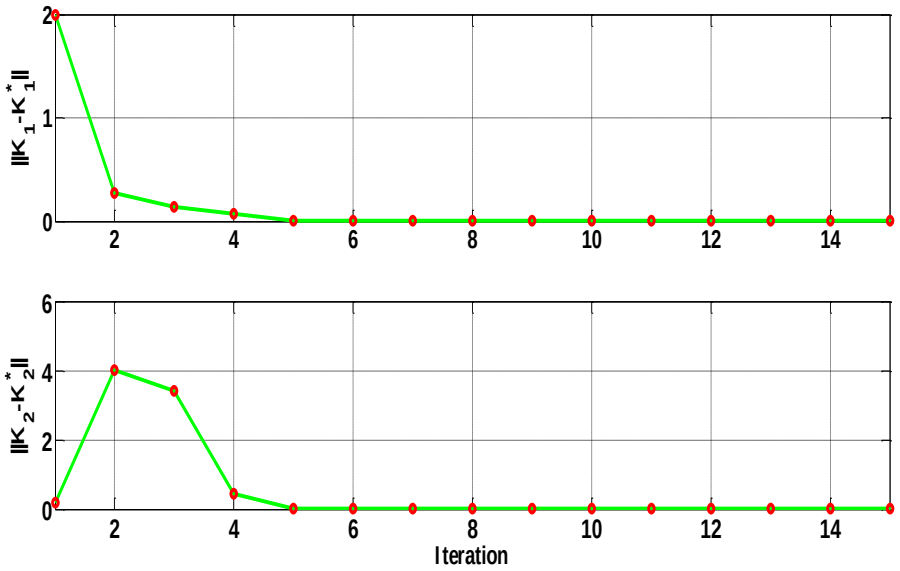

Fig 4.1. Case 1: Convergence K_1 and K_2 in off-policy RL

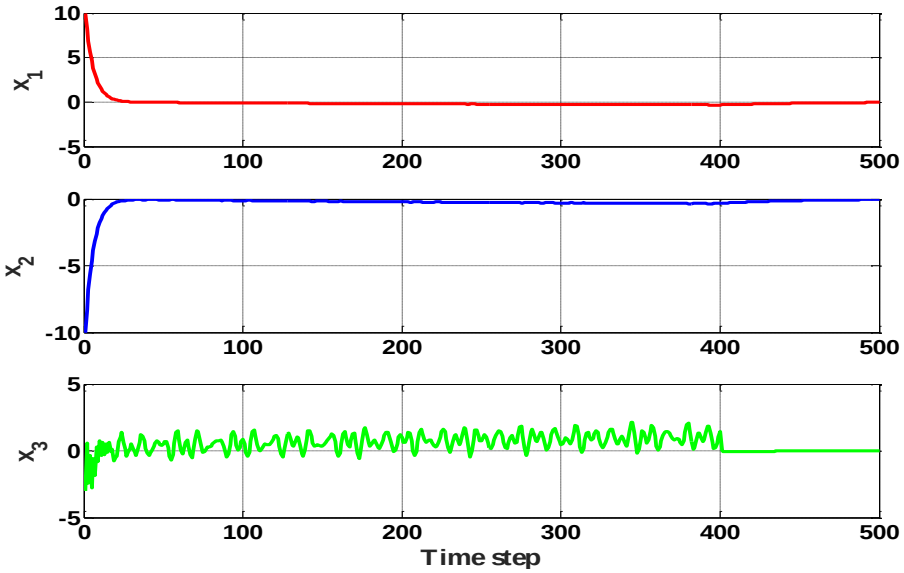

Fig 4.2. Case 1: The system states in off-policy RL

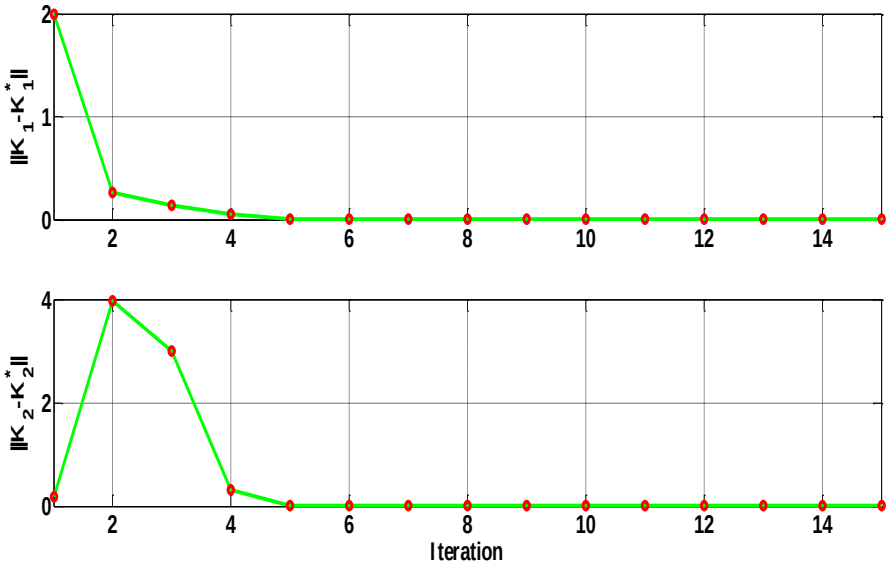

Fig 4.3. Case 2: Convergence K_1 and K_2 in off-policy RL

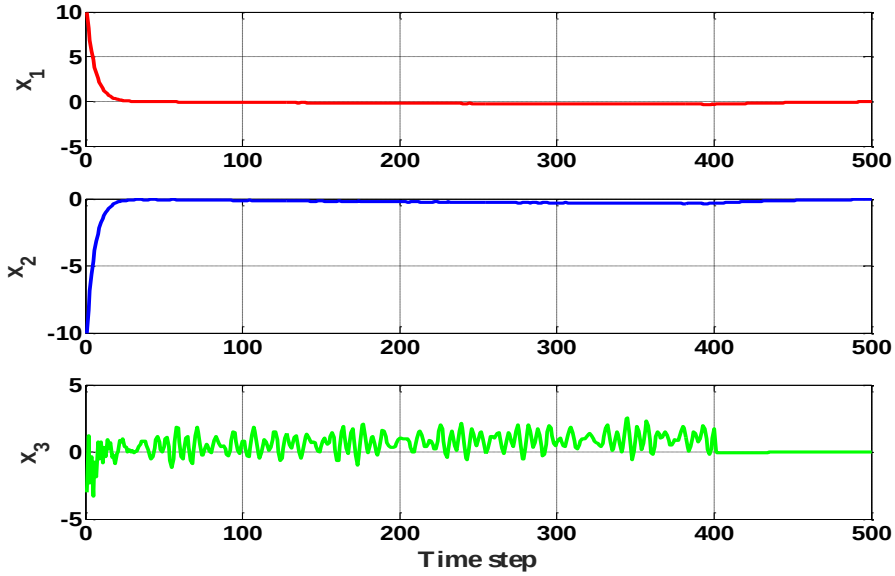

Fig 4.4. Case 2: The system states in off-policy RL

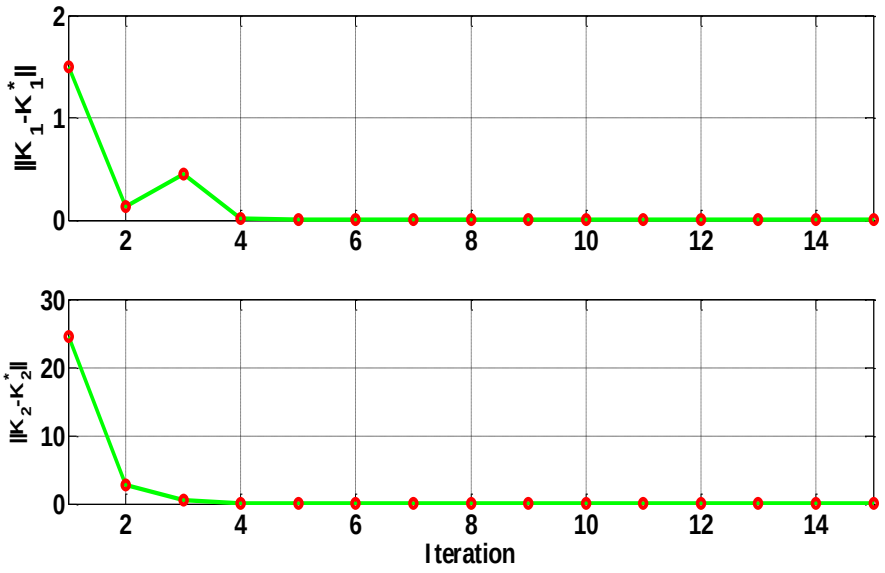

Fig 4.5. Case 3: Convergence K_1 and K_2 in off-policy RL

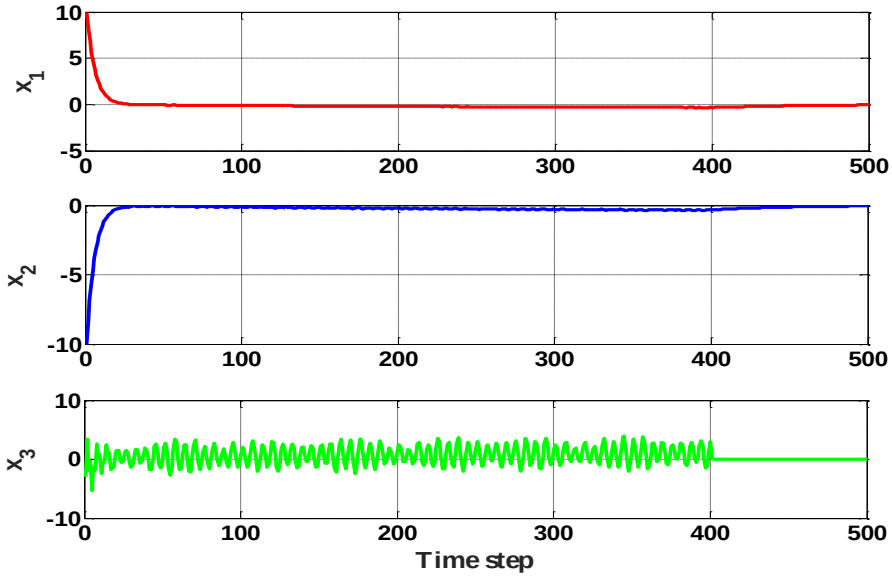

Fig 4.6. Case 3: The system states in off-policy RL

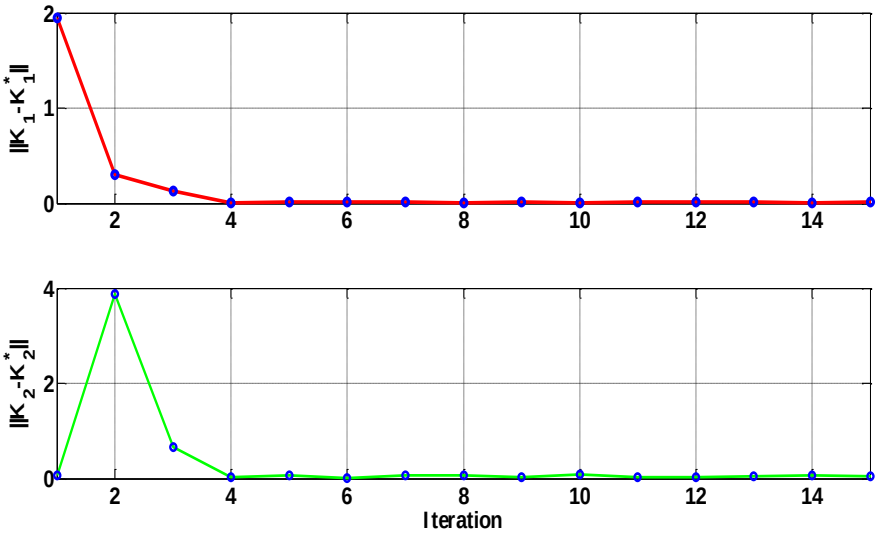

Fig. 4.7. Case 4: Convergence K_1 and K_2 in off-policy RL

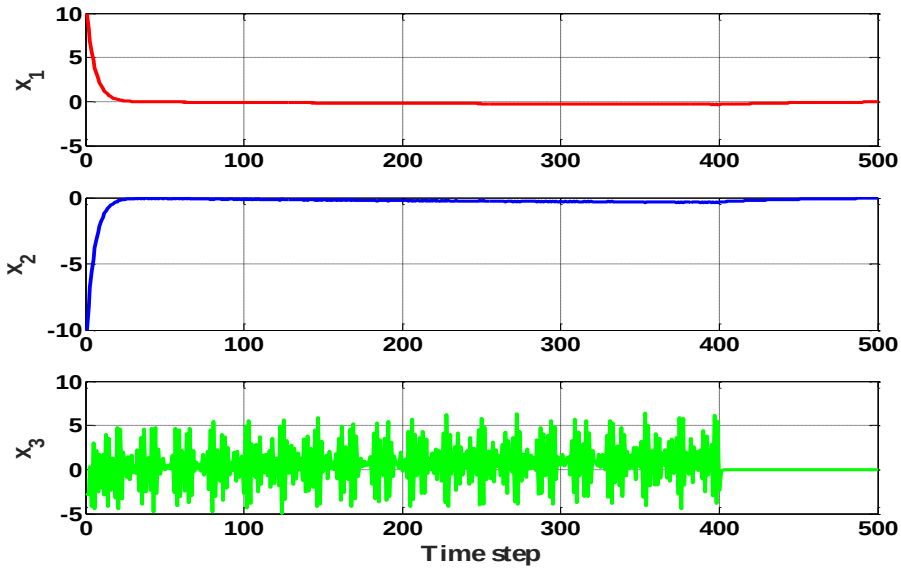

Fig 4.8. Case 4: The system states in off-policy RL

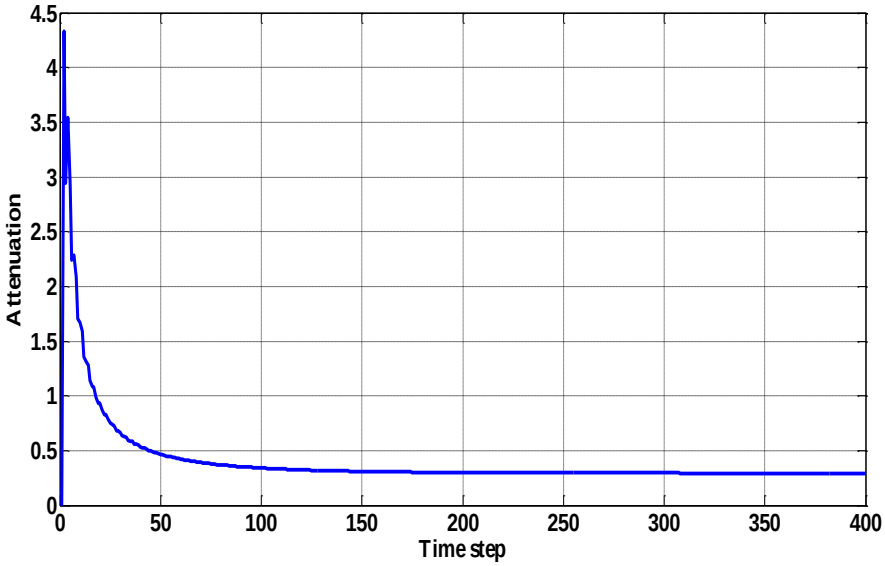

Fig 4.9. Disturbance Attenuation

4.5. Conclusion

An off-policy RL algorithm was presented to solve the H_∞ control problem of completely unknown linear discrete-time systems. The proposed method was inspired by [64], which was presented for continuous-time systems, and it had two main advantages over other model-free RL algorithms exist in the literature for discrete-time. First, the proposed model-free RL algorithm did not require an adjustable disturbance input and so it is more practical. Second, there was no bias in the Bellman equation as a result of probing noise and thus the probing noise did not affect the convergence of the algorithm.

Chapter 5

OUTPUT SYNCHRONIZATION OF HETEROGENEOUS DISCRETE-TME SYSTEMS:A MODEL-FREE OPTIMAL APPROACH

5.1. Introduction

Distributed control of multi-agent systems has received considerable attention in the control community motivated by its variety of applications in physics, social sciences, biology and engineering in recent years [103]-[107]. A large class of these results are for consensus and synchronization problems where a local control protocol is designed for each agent, based on its local and its neighbor's information, so that all agents reach an agreement on their outputs or states. These problems are divided into two categories: leaderless consensus and leader-following synchronization. In the leaderless consensus problem, all agents reach agreement on some common values of interest and in the leader-following problem, all follower agents track the trajectories of an agent called leader.

The state synchronization of the leader-following homogeneous multi-agent systems, where all agents and leader have identical dynamics, has been well established for both discrete-time (DT) and continuous-time (CT) systems [79]-[83]. However, in many practical applications, the agents' dynamics may not be the same. Therefore, it is desired to design distributed output synchronization control protocols for heterogeneous systems, which may have non identical dynamics.

The output synchronization problem has been studied for CT systems [108]-[119]. On the other hand, the output synchronization problem for DT multi-agent systems has received considerably less attention. In [87] and [88], the output synchronization problem of DT systems is solved for general set of networks such as those with unknown topology, partial state infromation and delay.[120] considered the leaderless consensus problem and [121] used an observer which requires continuous exchange of information among agents. In [86], an adaptive observer is designed to solve the leader-following synchronization problem of DT systems. Existing output

synchronization methods require solution of the output regulator equations. However, this requires complete knowledge of all agents's and the leader's dynamics. Moreover, existing results only consider making the steady-state tracking errors go to zero and do not provide an optimal solution that not only provides a zero steady-state tracking error but also minimizes the transient response. The output synchronization problem for DT systems with unknown dynamics is not considered in the literature.

In this chapter, for the first time, an optimal model-free solution to the output synchronization problem of heterogeneous DT multi-agent systems is provided. First, local discounted performance functions are defined for all agents and the optimal synchronization control protocols are found by solving a set of algebraic Riccati equations (AREs) and without requiring the explicit solution to the output regulator equations. It is shown that the proposed method implicitly solves the output regulator equations and therefore solves the output synchronization problem, provided that the discount factor is bigger than a lower bound. This formulation enables us to develop a Q-learning algorithm to solve the AREs using only measured data and so find the optimal distributed control protocols for each agent without requiring complete knowledge of the agents's or leader's dynamics. It is shown that the combination of a distributed adaptive observer and the controller guarantees synchronization. The relationship between the standard solution and the proposed solution is also shown. A simulation example is given to show the effectiveness of the proposed method.

5.2. Background: Graph communication and output synchronization

This section introduces some basic concepts of graph theory and the output synchronization problem.

5.2.1. Graph and Communication topology

A graph is a pair $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ with $\mathcal{V}_\mathcal{G} = \{v_1, v_2, ..., v_N\}$ a set of N nodes and $\mathcal{E}_\mathcal{G}$ a set of edges. Elements of $\mathcal{E}_\mathcal{G}$ are denoted as (v_i, v_j) which is termed an edge from v_i to v_j. The

adjacency matrix is $A=[a_{ij}]$ with weights $a_{ij}>0$ if $(v_j,v_i)\in\mathcal{E}_{\mathcal{G}}$, $a_{ij}=0$ if $(v_j,v_i)\notin\mathcal{E}_{\mathcal{G}}$ and $a_{ii}=0$ for all $i=1,2,...,N$. The in-degree of node v_i is $d_i(v_i)=\sum_{j=1}^{N}a_{ij}$. The diagonal in-degree matrix D is defined as $D=diag\{d_i(v_i)\}$. The graph Laplacian matrix is defined as $L=D-A$. Graph $\mathcal{G}$ is strongly connected if v_i and v_j are connected for all distinct nodes $v_i,v_j\in\mathcal{V}_{\mathcal{G}}$. A directed tree is a connected digraph where every node except one, called the root, has in-degree equal to one. A spanning tree of a digraph is a directed tree formed by graph edges that connects all the nodes of the graph. The pinning gain g_i is one if the leader is pined to agent i and zero otherwise. The pinning matrix G is defined as $G=diag(g_i), i=1,...,N$.

5.2.2 Output synchronization of multi-agent discrete-time systems

Consider linear heterogeneous multi-agent systems as

$$\begin{aligned} x_i(k+1)&=A_i x_i(k)+B_i u_i(k)\\ y_i(k)&=C_i x_i(k) \qquad i=1,...,N \end{aligned} \tag{5.1}$$

where $x_i\in\mathbb{R}^{n_i}$, $y_i\in\mathbb{R}^{p}$, and $u_i\in\mathbb{R}^{m_i}$ are the state, measurement output, and control input of the i-th agent.

The leader dynamics is assumed to be given by

$$\begin{aligned} r_0(k+1)&=F r_0(k)\\ y_0(k)&=H r_0(k) \end{aligned} \tag{5.2}$$

with $r_0\in\mathbb{R}^{l}$, $y_0\in\mathbb{R}^{p}$.

Assumption 5.1. The graph has a spanning tree and the leader is pinned to at least one root node.

Assumption 5.2. The leader has all poles on the unit circle and non-repeated.

Under Assumption 5.2, the state variables of the leader dynamics can generate any periodic or constant signals for the tracking trajectory [114], [118].

Remark 5.1. Assumption 5.2 is a standard assumption. The stable modes in the leader dynamics do not have an impact in the steady state response of the system as they asymptotically go to zero.

Problem 5.1. The output synchronization problem is to design a distributed control input $u_i(k)$ for all agents in (5.1) such that their outputs synchronize to the output of the leader. That is,

$$\lim_{k\to\infty} e_i(k) = \lim_{k\to\infty}\big(y_i(k) - y_0(k)\big) = 0, \quad \forall i = 1,..., N \tag{5.3}$$

Problem 1 can be solved using the solution to the output regulator equations given by

$$\begin{aligned} A_i\Pi_i + B_i\Gamma_i &= \Pi_i F \\ C_i\Pi_i &= H \end{aligned} \tag{5.4}$$

Based on the solution of (5.4), a standard control input that solves Problem 5.1 is given by

$$u_i(k) = \overline{K_i}\big(x_i(k) - \Pi_i r_i(k)\big) + \Gamma_i r_i(k) \tag{5.5}$$

Here, $r_i(k)$ is the estimation of leader's state $r_0(k)$ for the node i and is designed by the distributed observer

$$r_i(k+1) = F r_i(k) + c(1 + d_i + g_i)^{-1} K\Big[\sum_{j=1}^{N} a_{ij}\big(r_j(k) - r_i(k)\big) + g_i\big(r_0(k) - r_i(k)\big)\Big] \tag{5.6}$$

where $g_i \geq 0$ is the pinning gain, $c > 0$ is a coupling gain, and K is a designed matrix.

Define the observer estimation error $\delta_i(k)$ and state tracking error $\varepsilon_i(k)$ as

$$\delta_i(k) = r_i(k) - r_0(k) \tag{5.7}$$

$$\varepsilon_i(k) = x_i(k) - \Pi_i r_i(k) \tag{5.8}$$

By using (5.2), (5.6) and (5.7), the observer estimation error dynamics is defined as

$$\delta_i(k+1) = F\delta_i(k) + c(1 + d_i + g_i)^{-1} K\Big[\sum_{j=1}^{N} a_{ij}\big(\delta_j(k) - \delta_i(k)\big) - g_i\delta_i(k)\Big] \tag{5.9}$$

Then, the global observer estimation error dynamics is

$$\delta(k+1) = \big[I_N \otimes F - c(I_N + D + G)^{-1}(L + G) \otimes K\big]\delta(k) = A_\delta \delta(k) \tag{5.10}$$

where $\delta(k)=\left[\delta_1^{\mathrm{T}}(k) \quad \delta_2^{\mathrm{T}}(k) \quad ... \quad \delta_N^{\mathrm{T}}(k)\right]^{\mathrm{T}}$.

The dynamics of the state tracking errors are

$$\varepsilon_i(k+1)=x_i(k+1)-\Pi_i r_i(k+1)=A_i x_i(k)+B_i u_i(k)-\Pi_i\Big(F r_i(k)+c(1+d_i+g_i)^{-1}K\times \Big[\sum_{j=1}^{N}a_{ij}\big(r_j(k)-r_i(k)\big)+g_i\big(r_0(k)-r_i(k)\big)\Big]\Big) \tag{5.11}$$

Substituting the distributed control (5.5) into (4.11) and considering (5.4) and (5.7) yields

$$\varepsilon_i(k+1)=(A_i+B_i\overline{K_i})\varepsilon_i(k)+c\Pi_i(1+d_i+g_i)^{-1}K\Big[\sum_{j=1}^{N}a_{ij}\big(\delta_i(k)-\delta_j(k)\big)+g_i\delta_i(k)\Big] \tag{5.12}$$

The next results verify the performance of (5.4)-(5.6).

Theorem 5.1. Consider the heterogeneous multi-agent systems (5.1) with the distributed control protocol (5.4)-(5.6). Then, Problem 5.1 is solved if and only if $A_i+B_i\overline{K_i}$ is stable for $i=1,...,N$ and $A_\delta=I_N\otimes F-c(I_N+D+G)^{-1}(L+G)\otimes K$ is Schur.

Proof. It is shown in [82] how to design c and K to assure the convergence of the observer, i.e., A_δ has all eigenvalues inside the unit circle if the leader dynamics F is known. Moreover, if (A_i,B_i) be stabilizable, one can find $\overline{K_i}$ to stabilize $A_i+B_i\overline{K_i}$ for $i=1,...,N$. Then, by using (5.4), $\lim_{k\to\infty}\delta_i(k)=0$ and $\lim_{k\to\infty}\varepsilon_i(k)=0$, one has

$$\begin{aligned} &y_i(k)=C_i x_i(k)\to C_i\Pi_i r_i(k)=H r_i(k)\\ &y_i(k)=H r_i(k)\to H r_0(k)=y_0(k)\end{aligned} \tag{5.13}$$

Therefore, Problem 5.1 is solved and the proof is completed. ∎

Remark 5.2. Designing the control input in the output synchronization problem for each agent using (5.4)-(5.6) requires complete knowledge of the its own dynamics and the leader's dynamics.

5.3. Optimal output synchronization for heterogeneous systems

In this section, the output synchronization problem is formulated as a set of local optimal tracking problems. This leads to solving a set of algebraic Riccati equations (AREs). This formulation enables us to provide a model-free solution in Section 5.5. Moreover, it is shown that

the solutions of these AREs satisfy the output regulator equations (5.4) and make the output tracking errors (5.3) go to zero. It is assumed the leader's state $r_0(k)$ is available to all agents. This assumption is relaxed in Section 5.4.

5.3.1. Optimal Design

The goal is to find a control policy $u_i(k)$ for all agents to make the output of the all agents $y_i(k)$ in (5.1) follow the output of the leader $y_0(k)$ in (5.2) while minimizing a predefined performance function. Define the discounted performance function

$$V_i(x_i(k),u_i(k)) = \sum_{j=k}^{\infty} \gamma^{j-k} \left[\left(y_i(j) - y_0(j)\right)^{\mathrm{T}} Q_i \left(y_i(j) - y_0(j)\right) + u_i^{\mathrm{T}}(j) W_i u_i(j) \right] \tag{5.14}$$

where $0 < \gamma \leqslant 1$ is a discount factor, $Q_i \succeq 0$ and $W_i = W_i^{\mathrm{T}} \succ 0$.

Assumption 5.3. For $i = 1,....,N$, (A_i, B_i) are stabilizable and $(A_i, \sqrt{Q_i} C_i)$ are observable.

The control input is given as

$$u_i(k) = K_{x_i} x_i(k) + K_{r_0} r_0(k) = K_i X_i(k) \tag{5.15}$$

where $X_i(k) = \begin{bmatrix} x_i^{\mathrm{T}}(k) & r_0^{\mathrm{T}}(k) \end{bmatrix}^{\mathrm{T}}$ is the augmented system state and $K_i = \begin{bmatrix} K_{x_i} & K_{r_0} \end{bmatrix}$. Then, the augmented system dynamics is defined as

$$X_i(k+1) = T_i X_i(k) + B_{1i} u_i(k) \tag{5.16}$$

where

$$T_i = \begin{bmatrix} A_i & 0 \\ 0 & F \end{bmatrix} \quad \text{and} \quad B_{1i} = \begin{bmatrix} B_i \\ 0 \end{bmatrix}$$

The value function (5.14) can be rewritten as

$$V_i(X_i(k)) = \sum_{j=k}^{\infty} \gamma^{j-k} X_i^{\mathrm{T}}(j) (C_{1i}^{\mathrm{T}} Q_i C_{1i} + K_i^{\mathrm{T}} W_i K_i) X_i(j) = X_i^{\mathrm{T}}(k) P_i X_i(k) \tag{5.17}$$

where $C_{1i} = \begin{bmatrix} C_i & -H \end{bmatrix}$. The Bellman equation is then,

$$V_i(X_i(k)) = X_i^{\mathrm{T}}(k) C_{1i}^{\mathrm{T}} Q_i C_{1i} X_i(k) + u_i^{\mathrm{T}}(k) W_i u_i(k) + \gamma V_i(X_i(k+1)) \tag{5.18}$$

and the Hmaitonian is defined as

$$H(X_i(k),u_i(k)) = X_i^{\mathrm{T}}(k)C_{1i}^{\mathrm{T}}Q_iC_{1i}X_i(k) + u_i^{\mathrm{T}}(k)W_iu_i(k) + \gamma V_i(X_i(k+1)) - V_i(X_i(k))$$

By following the procedures in Chapter 2, the optimal control input $u_i^*(k)$ is given as

$$u_i^*(k) = K_i^*X_i(k) = \begin{bmatrix} K_{x_i}^* & K_{r_0}^* \end{bmatrix} X_i(k) = -\gamma(W_i + \gamma B_{1i}^{\mathrm{T}}P_iB_{1i})^{-1}B_{1i}^{\mathrm{T}}P_iT_iX_i(k) \tag{5.19}$$

where

$$P_i = \begin{bmatrix} P_i^{11} & P_i^{12} \\ P_i^{21} & P_i^{22} \end{bmatrix} \tag{5.20}$$

is found by solving the ARE

$$C_{1i}^{\mathrm{T}}Q_iC_{1i} - P_i + \gamma T_i^{\mathrm{T}}P_iT_i - \gamma^2T_i^{\mathrm{T}}P_iB_{1i}(W_i + \gamma B_{1i}^{\mathrm{T}}P_iB_{1i})^{-1}B_{1i}^{\mathrm{T}}P_iT_i = 0 \tag{5.21}$$

Lemma 5.1. Existence of the solution to the ARE.

Let Assumption 5.3 is satisfied. The ARE (5.21) has a unique positive semi-definite solution if $\gamma^{1/2}F$ has all its eigenvalues inside the unit circle.

Proof. Note that the ARE (5.21) can be written as

$$C_{1i}^{\mathrm{T}}Q_iC_{1i} - P_i + \overline{T}_i^{\mathrm{T}}P_i\overline{T}_i - \overline{T}_i^{\mathrm{T}}P_i\overline{B}_{1i}(W_i + \overline{B}_{1i}^{\mathrm{T}}P_i\overline{B}_{1i})^{-1}\overline{B}_{1i}^{\mathrm{T}}P_i\overline{T}_i = 0 \tag{5.22}$$

where $\overline{T}_i = \gamma^{1/2}T_i$ and $\overline{B}_{1i} = \gamma^{1/2}B_{1i}$. Equation (5.22) is the standard ARE without discount factor for a system with dynamics $(\gamma^{1/2}T_i, \gamma^{1/2}B_{1i})$. The necessary and sufficient condition for the standard ARE (5.22) to have a unique solution is stabilizability of the dynamics $(\gamma^{1/2}T_i, \gamma^{1/2}B_{1i})$ and observability of the $(\gamma^{1/2}T_i, \sqrt{Q_i}C_{1i})$. This requires that $(\gamma^{1/2}A_i, \gamma^{1/2}B_i)$ be stabilizable, $\gamma^{1/2}F$ be stable and $(\gamma^{1/2}A_i, \sqrt{Q_i}C_i)$ be observable. By considering Assumption 5.3, then $(\gamma^{1/2}A_i, \gamma^{1/2}B_i)$ are also stablizable and $(\gamma^{1/2}A_i, \sqrt{Q_i}C_i)$ are observable for any $0 < \gamma \leqslant 1$. Therefore, the ARE (5.21) has a unique positive semi-definite solution if conditions of Lemma 5.1 are satisfied. ■

Remark 5.3. Note that the control input (5.19) depends on the leader's states and the leader's state does not generally go to zero. Therefore, if the discount factor in the performance function

(5.14) is chosen as one, it becomes infinity for any control input and therefore the meaning of optimality is lost. $\gamma = 1$ can be only use if one knows a priori that the leader is generated by an asymptotically stable command generator system, which is the trivial case. That is, F has all its eigenvalues inside the unit circle. These results are consistent with the results of Lemma 5.1 which requires $\gamma^{1/2} F$ be stable. Based on Assumption 5.2, which excludes the trivial case that the leader states converge to zero, $\gamma^{1/2} F$ is stable if $0 < \gamma < 1$. However, as shown in the following example, the existence of the solution does not guarantee stability of the tracking error. An extra condition on the discount factor will be found in the sequel to guarantee stability as well as existence of the solution.

Example 5.1. Consider the first order system dynamics

$$\begin{aligned} x_i(k+1) &= 3x_i(k) + u_i(k) \\ y_i(k) &= x_i(k) \end{aligned} \tag{5.23}$$

The performance function is defined as

$$V_i(x_i)(k) = \sum_{j=k}^{\infty} \gamma^{j-k} [q_i x_i^2(j) + w_i u_i^2(j)] = p_i x_i^2(k) \tag{5.24}$$

The Hamiltonian function is defined as

$$H_i(k) = q_i x_i^2(j) + w_i u_i^2(j) + \gamma V_i(x_i(k+1)) - V_i(x_i(k)) \tag{5.25}$$

Considering $w_i = q_i = 1$ and applying the stationarity condition yields the following optimal control input

$$u_i^*(k) = -\gamma(1 + \gamma p_i)^{-1} 3 p_i x_i(k) \tag{5.26}$$

where p_i is found by solving the ARE

$$1 - p_i + 9\gamma p_i - 9\gamma^2 (1 + \gamma p_i)^{-1} p_i^2 = 0 \tag{5.27}$$

Substituting the solution of (5.27) into (5.26) yields,

$$u_i^*(k) = -3(1 - 2(1 + 10\gamma + \sqrt{100\gamma^2 - 16\gamma + 1})^{-1}) x_i(k) \tag{5.28}$$

However, this optimal solution found by solving ARE does not make the system stable for all values of $0 < \gamma < 1$. In fact, if $\gamma < \gamma^{\star} = 0.286$, then the system is unstable. The next theorem shows how to find an lower bound $\gamma^{\star}$ for the discount factor to assure the stability of the system.

Remark 5.4. In (5.19), the feedback gain is computed from the ARE solution (5.21). In Section 5.5, it is shown how to compute the ARE solution without solving (5.21) by using only measured data.

5.3.2. Convergence of Output Tracking Error

The following Theorem 5.3 shows that the control protocol (5.19) solves the output synchronization problem provided that the discount factor is close enough to one. To show this, the results of the following theorem are required.

Theorem 5.2. Lower bound for the discount factor.

Let Assumptions 5.2 and 5.3 are satisfied. Then, $K_{x_i}^{\star}$ given by (5.19) and (5.20) makes $A_i + B_i K_{x_i}^{\star}$ Schur if the discount factor satisfies

$$0 < (I - C_i^{\mathrm{T}} Q_i C_i (P_i^{11})^{-1}) < \gamma I < I \tag{5.29}$$

Proof. Considering the optimal control input (5.19) in the ARE (5.21) yields the Lyapunov equation

$$C_{1i}^{\mathrm{T}} Q_i C_{1i} - P_i + (K_i^{\star})^{\mathrm{T}} W_i K_i^{\star} + \gamma (T_i + B_{1i} K_i^{\star})^{\mathrm{T}} P_i (T_i + B_{1i} K_i^{\star}) = 0 \tag{5.30}$$

The Lyapunov equation is used to show the stability of the ARE solution. Using P_i in (5.20), the upper left-hand side of the Lyapunov equation (5.30) is

$$C_i^{\mathrm{T}} Q_i C_i - P_i^{11} + (K_{x_i}^{\star})^{\mathrm{T}} W_i^{11} K_{x_i}^{\star} + \gamma (A_i + B_i K_{x_i}^{\star})^{\mathrm{T}} P_i^{11} (A_i + B_i K_{x_i}^{\star}) = 0 \tag{5.31}$$

By considering $Q_i = M_i^{\mathrm{T}} M_i \;\; and \;\; W_i^{11} = D_i^{\mathrm{T}} D_i$, (5.31) can be rewritten as

$$P_i^{11} = \gamma (A_i + B_i K_{x_i}^{\star})^{\mathrm{T}} P_i^{11} (A_i + B_i K_{x_i}^{\star}) + \begin{bmatrix} C_i^{\mathrm{T}} M_i^{\mathrm{T}} & (K_{x_i}^{\star})^{\mathrm{T}} D_i^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} M_i C_i \\ D_i K_{x_i}^{\star} \end{bmatrix} \tag{5.32}$$

Eq. (5.32) has a unique positive definite solution if $\gamma^{1/2}(A_i + B_i K_{x_i}^{\star})$ is stable and the following system is observable

$$\left(\gamma^{1/2}(A_i + B_i K_{x_i}^{\star}), \begin{bmatrix} M_i C_i \\ D_i K_{x_i}^{\star} \end{bmatrix}\right) \tag{5.33}$$

In Chapter 1, it is proved that $\gamma^{1/2}(A_i + B_i K_{x_i}^{\star})$ is stable for every value of $0 < \gamma \leqslant 1$. Now, the observability of (5.33) should be shown. The observability matrix is defined as

$$O = \begin{bmatrix} Z I - \gamma^{1/2}(A_i + B_i K_{x_i}^{\star}) \\ M_i C_i \\ D_i K_{x_i}^{\star} \end{bmatrix} \tag{5.34}$$

The system (5.33) is observable if the observability matrix (5.34) be full rank. On the other hand,

$$\operatorname{rank}\left(\begin{bmatrix} Z I - \gamma^{1/2}(A_i + B_i K_{x_i}^{\star}) \\ M_i C_i \\ D_i K_{x_i}^{\star} \end{bmatrix}\right) = \operatorname{rank}\left(\begin{bmatrix} Z I - \gamma^{1/2}(A_i + B_i K_{x_i}^{\star}) \\ M_i C_i \end{bmatrix}\right) \tag{5.35}$$

The state feedback preserves observability. Then,

$$\operatorname{rank}\left(\begin{bmatrix} Z I - \gamma^{1/2}(A_i + B_i K_{x_i}^{\star}) \\ M_i C_i \end{bmatrix}\right) = \operatorname{rank}\left(\begin{bmatrix} Z I - \gamma^{1/2} A_i \\ M_i C_i \end{bmatrix}\right) \tag{5.36}$$

Then, the observability of (5.33) is equivalent to observability of $(\gamma^{1/2} A_i, M_i C_i) = (\gamma^{1/2} A_i, \sqrt{Q_i} C_i)$. By considering Assumption 5.3, it can be concluded that the Lyapunov equation makes the system stable and observable. Therefore, the solution of (5.31) is unique positive definite.

Now, it is desired to show that $A_i + B_i K_{x_i}^{\star}$ is stable. By considering $A_{c_i} = A_i + B_i K_{x_i}^{\star}$ in (5.31), one has

$$\gamma A_{c_i}^{\mathrm{T}} P_i^{11} A_{c_i} = P_i^{11} - C_i^{\mathrm{T}} Q_i C_i - (K_{x_i}^{\star})^{\mathrm{T}} W_i^{11} K_{x_i}^{\star} \tag{5.37}$$

Multiplying the left-hand and right-hand sides of (5.37) by $z_i^{\mathrm{T}}(k)$ and $z_i(k)$, respectively yields,

$$\gamma z_i^{\mathrm{T}}(k)A_{c_i}^{\mathrm{T}}P_i^{11}A_{c_i}z_i(k) = z_i^{\mathrm{T}}(k)P_i^{11}z_i(k) - z_i^{\mathrm{T}}(k)C_i^{T}Q_iC_iz_i(k) - z_i^{\mathrm{T}}(k)(K_{x_i}^{\star})^{\mathrm{T}}W_i^{11}K_{x_i}^{\star}z_i(k) \tag{5.38}$$

Also, let λ_i be an eigenvalue of A_{c_i} *and* $z_i(k)$ its corresponding eigenvector. Then,

$$A_{c_i}z_i(k) = \lambda_i z_i(k) \tag{5.39}$$

By substituting (5.39) into (5.38), one has

$$\gamma z_i^{\mathrm{T}}(k)\lambda_i P_i^{11}\lambda_i z_i(k) = z_i^{\mathrm{T}}(k)P_i^{11}z_i(k) - z_i^{\mathrm{T}}(k)C_i^{\mathrm{T}}Q_iC_iz_i(k) - z_i^{\mathrm{T}}(k)(K_{x_i}^{\star})^{\mathrm{T}}W_i^{11}K_{x_i}^{\star}z_i(k) \tag{5.40}$$

Eq. (5.40) can be written as

$$(\gamma\|\lambda_i\|^2 - 1)P_i^{11} = -C_i^{\mathrm{T}}Q_iC_i - \gamma^2 A_i^{\mathrm{T}}P_i^{11}B_i(W_i^{11} + \gamma B_i^{\mathrm{T}}P_i^{11}B_i)^{-1}W_i^{11}(W_i^{11} + \gamma B_i^{\mathrm{T}}P_i^{11}B_i)^{-1}B_i^{\mathrm{T}}P_i^{11}A_i \tag{5.41}$$

This gives

$$(\gamma\|\lambda_i\|^2 - 1)P_i^{11} < -C_i^{\mathrm{T}}Q_iC_i \tag{5.42}$$

Then,

$$\|\lambda_i\|^2 I < \frac{1}{\gamma}\left(I - C_i^{\mathrm{T}}Q_iC_i(P_i^{11})^{-1}\right) \tag{5.43}$$

For the stability of the system, the eigenvalues of the closed-loop system $A_{c_i} = A_i + B_iK_{x_i}^{\star}$ should be in the unit circle, that is,

$$\|\lambda_i\|^2 I < \frac{1}{\gamma}\left(I - C_i^{\mathrm{T}}Q_iC_i(P_i^{11})^{-1}\right) < I \tag{5.44}$$

Then, for $\left(I - C_i^{\mathrm{T}}Q_iC_i(P_i^{11})^{-1}\right) < \gamma I < I$, the closed-loop system is stable and the proof is completed. ∎

Remark 5.5. Note that Theorem 5.2 shows that the discount factor affects the stability and there is a minimum value for it to make the system stable which depends on the system dynamics. However, we do not need to calculate this value as one can always choose the discount factor close enough to 1 to assure that the ARE solution is stabilizing.

Theorem 5.3. Consider the system dynamics (5.1) and the leader dynamics (5.2). Assume the discount factor satisfies (5.29). Then, the control input (5.19) makes the output tracking errors $e_i(k) = y_i(k) - y_0(k)$ go to zero.

Proof. First, it is shown that null space of P_i is a zero-error and invariant subspace. The proof is then completed by showing that this subspace is also attractive.

Multiply both sides of Lyapunov equation (5.30) by X_i and X_i^{T}, respectively. Then,

$$X_i^{\mathrm{T}} C_{1i}^{\mathrm{T}} Q_i C_{1i} X_i - X_i^{\mathrm{T}} P_i X_i + X_i^{\mathrm{T}} (K_i^*)^{\mathrm{T}} W_i K_i^* X_i + \gamma X_i^{\mathrm{T}} (T_i + B_{1i} K_i^*)^{\mathrm{T}} P_i (T_i + B_{1i} K_i^*) X_i = 0 \tag{5.45}$$

If $X_i \in null(P_i)$ in (5.45), then $X_i^{\mathrm{T}} P_i X_i = 0$. Therefore, null space of P_i yields

$$X_i^{\mathrm{T}} C_{1i}^{\mathrm{T}} Q_i C_{1i} X_i + X_i^{\mathrm{T}} (K_i^*)^{\mathrm{T}} W_i K_i^* X_i + \gamma X_i^{\mathrm{T}} (T_i + B_{1i} K_i^*)^{\mathrm{T}} P_i (T_i + B_{1i} K_i^*) X_i = 0 \tag{5.46}$$

Eq. (5.46) is summation of three quadratic terms and is equal to zero if and only if all quadratic terms are zero. That means, $X_i^{\mathrm{T}} C_{1i}^{\mathrm{T}} Q_i C_{1i} X_i = 0$, $X_i^{\mathrm{T}} (K_i^*)^{\mathrm{T}} W_i K_i^* X_i = 0$, $X_i^{\mathrm{T}} (T_i + B_{1i} K_i^*)^{\mathrm{T}} P_i (T_i + B_{1i} K_i^*) X_i = 0$. Since Q_i, W_i, and P_i are positive definite matrices, (5.46) is satisfied if and only if $C_{1i} X_i = 0$, $K_i^* X_i = 0$, and $(T_i + B_{1i} K_i^*) X_i = 0$. Since $(T_i + B_{1i} K_i^*) X_i = T_{i_{cl}} X_i = 0 \; if \; P_i X_i = 0$, one concludes that the null space of P_i is $T_{i_{cl}}$-invariant. On the other hand, $C_{1i} X_i = 0$ gives $(y_i(k) - y_0(k)) = 0$ which yields $e_i(k) = y_i(k) - y_0(k) = 0$. Therefore, null space of is a zero error and invariant subspace. To complete the proof, it remains to show that the null space of P_i is attractive. To this end, consider the following Lyapunov function

$$V_i(X_i(k)) = X_i^{\mathrm{T}}(k) P_i X_i(k) \geqslant 0 \tag{5.47}$$

which yields

$$\begin{aligned} &\Delta V_i(X_i) = V_i(X_i(k+1)) - V_i(X_i(k)) = \\ &X_i^{\mathrm{T}}(k)(T_i + B_{1i} K_i^*)^{\mathrm{T}} P_i (T_i + B_{1i} K_i^*) X_i(k) - X_i^{\mathrm{T}}(k) P_i X_i(k) = (\eta_i^2 - 1) X_i^{\mathrm{T}}(k) P_i X_i(k) \end{aligned} \tag{5.48}$$

where η_i is the eigenvalue of the augmented system (5.16) with the optimal control input (5.19). Theorem 5.2 results in $\eta_i < 1$. Then, one has $\eta_i^2 - 1 < 0$ and $\Delta V_i(X_i) \leqslant 0$. By La Salle's extension, $X_i(k)$ converges to an invariant set contained in null space of P_i. ∎

5.3.3 Relationship between output regulator equations and ARE

The solution to the output regulator equations (5.4) can be expressed in terms of the solution to the ARE (5.21).

Define the null space of P_i as

$$\Omega_i = \{X_i(k) \neq 0 \,|P_i X_i(k) = 0\}$$

where $X_i(k) = \begin{bmatrix} x_i^{\mathrm{T}}(k) & r_0^{\mathrm{T}}(k) \end{bmatrix}^{\mathrm{T}}$ and P_i is defined as (5.20).

Lemma 5.2. Given the solution (5.20) to the optimal tracking problem ARE (5.21) for any $X_i(k) \in \Omega_i$, one has

Proof. for any $X_i(k) \in \Omega_i$ one has

$$P_i X_i(k) = \begin{bmatrix} P_i^{11} & P_i^{12} \\ P_i^{21} & P_i^{22} \end{bmatrix} \begin{bmatrix} x_i(k) \\ r_0(k) \end{bmatrix} = 0 \tag{5.50}$$

which results in

$$x_i(k) = -(P_i^{11})^{-1} P_i^{12} r_0(k) \tag{5.51}$$

$$P_i^{22} = -P_i^{21}(P_i^{11})^{-1} P_i^{12} \tag{5.52}$$

■

Theorem 5.4. Let P_i be the solution to (5.21) and $K_i^\star = \begin{bmatrix} K_{x_i}^\star & K_{r_0}^\star \end{bmatrix}$ be given as (5.19). Then, the output regulator equations (5.4) are satisfied with

$$\begin{aligned} \Pi_i &= -(P_i^{11})^{-1} P_i^{12} \\ \Gamma_i &= K_{r_0}^\star - K_{x_i}^\star (P_i^{11})^{-1} P_i^{12} \end{aligned} \tag{5.53}$$

Proof. The proof has two parts. In part a, it is shown that the first equation of the output regulator equations is satisfied and in part b, it is shown that its second equation is satisfied.

a. By using (5.16) and (5.19), the closed-loop system can be written as

$$\begin{aligned} X_{i_{cl}}(k+1) &= \begin{bmatrix} A_i + B_i K_{x_i}^\star & B_i K_{r_0}^\star \\ 0 & F \end{bmatrix} X_{i_{cl}}(k) \\ &= T_{i_{cl}} X_{i_{cl}}(k) \end{aligned} \tag{5.54}$$

Using (5.51) and the fact that the null space of P_i is closed-loop invariant, which was shown in Theorem 5.3, yields

$$x_i(k+1) = -(P_i^{11})^{-1} P_i^{12} r_0(k+1) \tag{5.55}$$

By substituting (5.1), (5.2), and (5.19) into (5.55), one has

$$A_i x_i(k) + B_i K_{x_i}^{\star} x_i(k) + B_i K_{r_0}^{\star} r_0(k) = -(P_i^{11})^{-1} P_i^{12} F r_0(k) \tag{5.56}$$

In Theorem 5.1, it is proved that $r_i(k) \to r_0(k)$. Therefore, using the results of Theorem 5.1 in (5.19) yields

$$u_i(k) = K_{x_i}^{\star} x_i(k) + K_{r_i}^{\star} r_i(k) \tag{5.57}$$

Now, comparing the control inputs (5.5) and (5.57) results in

$$K_{x_i}^{\star} = \bar{K}_i \quad \text{and} \quad K_{r_i}^{\star} = \Gamma_i - \bar{K}_i \Pi_i \tag{5.58}$$

Substituting (5.51) and (5.58) into (5.56) and performing some manipulation yields

$$A_i (P_i^{11})^{-1} P_i^{12} + B_i \bar{K}_i ((P_i^{11})^{-1} P_i^{12} + \Pi_i) + B_i \Gamma_i = (P_i^{11})^{-1} P_i^{12} F \tag{5.59}$$

Eq.(5.59) is equal to (5.4) with $\Pi_i = -(P_i^{11})^{-1} P_i^{12}$. Then, by using (5.58), $\Gamma_i = K_{r_i}^{\star} + \bar{K}_i \Pi_i = K_{r_i}^{\star} + K_{x_i}^{\star} (P_i^{11})^{-1} P_i^{12}$. That is, the optimal tracking solution (5.19) solves the output regulator equations implicitly.

b. In Theorem 5.3, it is shown that the null space of P_i is a subspace of the null space of $C_{1i}^{\mathrm{T}} Q_i C_{1i}$. Thus, if $X_i^{\mathrm{T}} P_i X_i = 0$, then $X_i^{\mathrm{T}} C_{1i}^{\mathrm{T}} Q_i C_{1i} X_i = 0$ which is equivalent to,

$$(C_i x_i - H r_0)^{\mathrm{T}} Q_i (C_i x_i - H r_0) = 0 \tag{5.60}$$

On the other hand, Lemma 5.2 and part a give

$$x_i = \Pi_i r_0 \tag{5.61}$$

Substituting (5.61) into (5.60), yields

$$r_0^{\mathrm{T}} (C_i \Pi_i - H)^{\mathrm{T}} Q_i (C_i \Pi_i - H) r_0 = 0 \tag{5.62}$$

since Q_i is a positive definite matrix and r_i is not always zero, then (5.62) is zero if and only if $(C_i\Pi_i - H) = 0$. In fact, the null space of P_i is the space that $(C_i\Pi_i - H) = 0$. Therefore, as it was shown in Theorem 5.3, starting from anywhere in the space, the system trajectories converge to the space in which $(C_i\Pi_i - H) = 0$. Then, the proof is completed.

As it is shown in Theorem 5.4, the solution to the ARE (5.21) gives the solution to the output regulator equations and therefore, solving the ARE equation is a sufficient condition for solving the output regulator problem. In Section 5.5 we show how to compute the solution to the ARE (5.21) online using measured data without knowing the dynamics of the leader or agents.

5.4. Adaptive distributed observer design

In Section 5.3, it is assumed that all agents have the state of the leader r_0. This assumption can be relaxed by the distributed observer (5.6). But, the distributed observer (5.6) needs the leader dynamics F. In this section, an adaptive distributed observer is designed to estimate the state of the leader for all agents without requiring complete knowledge of the leader's dynamics (F, H).

To accomplish this, the following distributed observer is used,

$$r_i(k+1) = \hat{F}_i(k) r_i(k) + c(1 + d_i + g_i)^{-1} K[\sum_{j=1}^{N} a_{ij}(r_j(k) - r_i(k)) + g_i(r_0(k) - r_i(k))] \tag{5.63}$$

where $\hat{F}_i(k)$ is an estimation of F, and K and are designed in the following Lemma 5.3.

Lemma 5.3. Let Assumptions 5.1 and 5.2 be satisfied and $F = F_0 + \Delta F$ where the nominal F_0 has all its poles on the unit circle and $\| \Delta F \|_\infty < \xi$ for some known bound ξ. Then, for a general graph, $A_\delta = I_N \otimes F - c(I_N + D + G)^{-1}(L + G) \otimes K$ is Schur if $K = F_0$ and

$$|1 - c\lambda_i| < \frac{1}{\xi \sigma_{\max}(F_0)} \tag{5.64}$$

where λ_i $i = 1, ..., N$ are the eigenvalues of $(I_N + D + G)^{-1}(L + G)$.

Proof. Showing A_δ is Schur is equivalent to show (5.10) is stable. By considering $F = F_0 + \Delta F$, (5.10) can be rewritten as

$$\delta(k+1) = [I_N \otimes F_0 - c(I_N + D + G)^{-1}(L+G) \otimes K]\delta(k) + (I_N \otimes \Delta F)\delta(k) \equiv F_{C0}\,\delta(k) + w_s \qquad (5.65)$$

The system (5.65) can be considered as a feedback connection of two systems given by

$$\begin{aligned} H_1(Z) &= (ZI - F_{C0})^{-1} \\ H_2(Z) &= w_s \end{aligned} \qquad (5.66)$$

Based on the small-gain theorem, if F_{C0} is Schur and the gain of H_1 is less than or equal to the inverse of the gain of H_2, i.e., $\frac{1}{\xi}$, then the system (5.66) is locally asymptotically stable.

By choosing $K = F_0$, one has

$$F_{C0} = I_N \otimes F_0 - c(I_N + D + G)^{-1}(L+G) \otimes F_0$$

It is shown in [83] that F_{C0} is Schur if and only if all matrices $I_N \otimes F_0 - c\lambda_i I_N \otimes F_0$ have their eigenvalues inside the unit circle. $\lambda_i \; i = 1, ..., N$ are locates inside a circle with center 1 and radius one. Then, for the set of eigenvalues of F_{C0}, i.e. $\mu(F_{C0})$, one has

$$\mu(F_{C0}) = \mu(I_N \otimes F_0 - c\lambda_i I_N \otimes F_0) = (1 - c\lambda_i)\mu(I_N \otimes F_0)$$

on the other hand, one has

$$h_1(k) = (F_0(1 - c\lambda_i))^{k-1}$$

$$\| H_1 \|_\infty = \sup_k | h_1(k) | = \| F_0(1 - c\lambda_i) \|$$

Then, based on the small-gain theorem, the system (5.65) is stable if and only if

$$\begin{aligned} | \mu(F_{C0}) | &= | (1 - c\lambda_i)\mu(I_N \otimes F_0) | < 1 \\ \| H_1 \|_\infty &= \| F_0(1 - c\lambda_i) \| < \frac{1}{\xi} \end{aligned} \qquad (5.67)$$

This results in

$$| 1 - c\lambda_i | < \frac{1}{\xi \sigma_{\max}(F_0)} \qquad (5.68)$$

■

Lemma 5.4 (Young's inequality). Let $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^n$ be arbitrary vectors. Then,

$$X^T Y \leqslant \frac{X^T X + Y^T Y}{2} \tag{5.69}$$

■

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

The next result provides an update law for $\hat{F}_i(k)$ in (5.63).

Theorem 5.5. Consider the distributed observer (5.63) with K and c designed using the results of Lemma 5.3. The observer estimation error (5.7) by considering(5.63) is

$$\delta_i(k+1) = \hat{F}_i(k) r_i(k) + c(1 + d_i + g_i)^{-1} K[\sum_{j=1}^{N} a_{ij}(r_j - r_i) + g_i(r_0 - r_i)] - F r_0(k) \tag{5.70}$$

Then, the observer estimation error (5.70) converges to zero for all $i = 1,...,N$ by selecting the update law for $\hat{F}_i(k)_{\text{vec}} \in \mathbb{R}^{l^2 \times 1}$ as

$$\hat{F}_i(k+1)_{\text{vec}} = \hat{F}_i(k)_{\text{vec}} + \alpha \varphi_i(k+1) \delta_i(k+1), \ \forall i \in N \tag{5.71}$$

with

$$\varphi_i(k+1) = -R_i^{\mathrm{T}} (R_i R_i^{\mathrm{T}} + \zeta I_{l \times l})^{-1} \ , \ \zeta > 0$$

$$R_i^{\mathrm{T}} = I_l \otimes r_i(k)$$

$$0 < \alpha < \frac{1}{2}$$

Proof. The global observer estimation error dynamics is,

$$\delta(k+1) = [I_N \otimes F - c(I + D + G)^{-1}(L + G) \otimes K]\delta(k) + R(\hat{F}(k) - F)_{\text{vec}} = A_\delta \delta(k) + R\tilde{F}(k)_{\text{vec}} \tag{5.72}$$

with

$$R^{\mathrm{T}} = diag(R_1^{\mathrm{T}}, R_2^{\mathrm{T}}, ..., R_N^{\mathrm{T}})$$

The Lyapunov stability theorem is used to prove that the observer estimation error converges to zero. Consider the Lyapunov function as

$$V(k) = \delta^{\mathrm{T}}(k) P \delta(k) + \tilde{F}(k)_{\mathrm{vec}}^{\mathrm{T}} \Gamma^{-1} \tilde{F}(k)_{\mathrm{vec}} \tag{5.73}$$

Then,

$$\begin{aligned} \Delta V(k) = V(k+1) - V(k) = & -\delta^{\mathrm{T}}(k)(P - A_\delta^{\mathrm{T}} P A_\delta)\delta(k) + 2\delta^{\mathrm{T}}(k) A_\delta^{\mathrm{T}} P R \tilde{F}(k)_{\mathrm{vec}} + \tilde{F}(k)_{\mathrm{vec}}^{\mathrm{T}} R^{\mathrm{T}} P R \tilde{F}(k)_{\mathrm{vec}} \\ & + \tilde{F}(k+1)_{\mathrm{vec}}^{\mathrm{T}} \Gamma^{-1} \tilde{F}(k+1)_{\mathrm{vec}} - \tilde{F}(k)_{\mathrm{vec}}^{\mathrm{T}} \Gamma^{-1} \tilde{F}(k)_{\mathrm{vec}} \end{aligned} \tag{5.74}$$

Substituting update law (5.71) into(5.74) yields,

$$\begin{aligned} \Delta V(k) = & -\delta^{\mathrm{T}}(k)(P - A_\delta^{\mathrm{T}} P A_\delta)\delta(k) + 2\delta^{\mathrm{T}}(k) A_\delta^{\mathrm{T}} P R \tilde{F}(k)_{\mathrm{vec}} + \tilde{F}(k)_{\mathrm{vec}}^{\mathrm{T}} R^{\mathrm{T}} P R \tilde{F}(k)_{\mathrm{vec}} \\ & + 2\alpha \tilde{F}(k)_{\mathrm{vec}}^{\mathrm{T}} \Gamma^{-1} \varphi(k+1) A_\delta \delta(k) + 2\alpha \tilde{F}(k)_{\mathrm{vec}}^{\mathrm{T}} \Gamma^{-1} \varphi(k+1) R \tilde{F}(k)_{\mathrm{vec}} \\ & + \alpha^2 \delta^{\mathrm{T}}(k) A_\delta^{\mathrm{T}} \varphi^{\mathrm{T}}(k+1) \Gamma^{-1} \varphi(k+1) A_\delta \delta(k) + 2\alpha^2 \delta^{\mathrm{T}}(k) A_\delta^{\mathrm{T}} \varphi^{\mathrm{T}}(k+1) \Gamma^{-1} \varphi(k+1) R \tilde{F}(k)_{\mathrm{vec}} \\ & + \alpha^2 \tilde{F}(k)_{\mathrm{vec}}^{\mathrm{T}} R^{\mathrm{T}} \varphi^{\mathrm{T}}(k+1) \Gamma^{-1} \varphi(k+1) R \tilde{F}(k)_{\mathrm{vec}} \end{aligned} \tag{5.75}$$

Select

$$\begin{aligned} \varphi(k+1) &= diag(\varphi_1(k+1), ..., \varphi_N(k+1)) \\ &= -R^{\mathrm{T}}(RR^{\mathrm{T}} + \zeta I)^{-1} \\ P &= (RR^{\mathrm{T}} + \zeta I)^{-1} \end{aligned} \tag{5.76}$$

Then,

$$L = \varphi^{\mathrm{T}}(k+1) \Gamma^{-1} \varphi(k+1) < \Gamma^{-1} P \tag{5.77}$$

By using (5.76) and (5.77), the gradient of the value function (5.75) can be rewritten as

$$\begin{aligned} \Delta V = & -\delta^{\mathrm{T}}(k)(P - A_\delta^{\mathrm{T}} P A_\delta - \alpha^2 A_\delta^{\mathrm{T}} L A_\delta)\delta(k) + \tilde{F}(k)_{\mathrm{vec}}^{\mathrm{T}}(R^{\mathrm{T}} P R + \alpha^2 R^{\mathrm{T}} L R - 2\alpha \Gamma^{-1} R^{\mathrm{T}} P R)\tilde{F}(k)_{\mathrm{vec}} \\ & + 2\delta^{\mathrm{T}}(k)(\alpha^2 A_\delta^{\mathrm{T}} L R + A_\delta^{\mathrm{T}} P R - \alpha \Gamma^{-1} A_\delta^{\mathrm{T}} P R)\tilde{F}(k)_{\mathrm{vec}} \end{aligned} \tag{5.78}$$

By considering Lemma 5.3, one has

$$\begin{aligned} \Delta V \leqslant & -\delta^{\mathrm{T}}(k)(P - A_\delta^{\mathrm{T}} P A_\delta)\delta(k) + \delta^{\mathrm{T}}(k)(A_\delta^{\mathrm{T}}(2\alpha^2 L + (1 - \alpha\Gamma^{-1})P) A_\delta)\delta(k) + \\ & \tilde{F}(k)_{\mathrm{vec}}^{\mathrm{T}}(R^{\mathrm{T}}(2P + 2\alpha^2 L - 3\alpha\Gamma^{-1} P)R)\tilde{F}(k)_{\mathrm{vec}} \end{aligned} \tag{5.79}$$

The observer estimation error converges to zero if $\Delta V \leqslant 0$ in (5.79). In Lemma 5.3, it is proved that A_δ is Schur for desired K and c. Therefore, $(P - A_\delta^{\mathrm{T}} P A_\delta) > 0$. By using the results of Lemma 5.3, $\Delta V \leqslant 0$ if the following inequalities are satisfied,

$$\begin{aligned} &2P-3\alpha\Gamma^{-1}P+2\alpha^2 L<0\\ &P+2\alpha^2 L-\alpha\Gamma^{-1}P<0 \end{aligned} \tag{5.80}$$

by considering (5.77), these conditions yield

$$\begin{aligned} &2P-3\alpha\Gamma^{-1}P+2\alpha^2\Gamma^{-1}P<0\\ &P-\alpha\Gamma^{-1}P+2\alpha^2\Gamma^{-1}P<0 \end{aligned} \tag{5.81}$$

Conditions to (5.80) are guaranteed if

$$\Gamma^{-1}>\frac{1}{\alpha-2\alpha^2} \text{ and } 0<\alpha<\frac{1}{2} \tag{5.82}$$

Therefore, the observer estimation error (5.70) converges to zero for all $i=1,...,N$ and the proof is completed. ■

Remark 5.6. The optimality is not considered in the observer design but the overall control strategy found in the steady-state,i.e., after the observer is converged, is optimal.

5.5. Model-free solution of optimal output synchronization problem

In this section, it is shown how to use reinforcement Q-learning algorithm to compute the solution to the ARE (5.21) online using measured data without knowing the dynamics of the leader or agents.

5.5.1 Combining the optimal tracking control and adaptive observer design

Theorems 5.3 and 5.4 require the use of protocol (5.19), so that every agent must know the leader's state $r_0(k)$. In Section 5.4, it is shown that by using the local adaptive observer (5.63) and the update law (5.71), every agent can get a local estimation of the leader's state. By using the local estimation of $r_i(k)$ in (5.19), the optimal tracking control input for each agent is obtained using

$$u_i^\star(k)=K_{x_i}^\star x_i(k)+K_{r_i}^\star r_i(k)=K_i^\star\hat{X}_i(k) \tag{5.83}$$

where $K_i^\star=\begin{bmatrix}K_{x_i}^\star & K_{r_i}^\star\end{bmatrix}$ and $\hat{X}_i(k)=\begin{bmatrix}x_i^{\mathrm{T}}(k) & r_i^{\mathrm{T}}(k)\end{bmatrix}^{\mathrm{T}}$.

Theorem 6. Consider the system dynamics (5.1) and the leader dynamics (5.2) with observer (5.63) and (5.71) and control input (5.83). Then, the output tracking error $e_i(k) = y_i(k) - y_0(k)$ goes to zero.

Proof. The augmented system (5.16) using the control input (5.83) and $e_i(k)$ are,

$$\begin{aligned}\begin{bmatrix} x_i(k+1) \\ r_0(k+1) \end{bmatrix} &= \begin{bmatrix} A_i + B_i K_{x_i}^{\star} & B_i K_{r_i}^{\star} \\ 0 & F \end{bmatrix} \begin{bmatrix} x_i(k) \\ r_0(k) \end{bmatrix} \\ &+ \begin{bmatrix} B_i K_{r_i}^{\star} \\ 0 \end{bmatrix} \delta_i(k)\end{aligned} \tag{5.84}$$

$$e_i(k) = Cx_i(k) - Hr_0(k) \tag{5.85}$$

In Theorem 5.2, it is shown that $A_i + B_i K_{x_i}^{\star}$ has all its eigenvalues inside the unit circle. Then, based on Assumption 5.2 and since in Theorem 5.1, it is proved that $\delta_i(k) \to 0, \forall i \in N$, the augmented system (5.84) is BIBO stable. Therefore, there exists a constant M such that,

$$\| e_i(k) \| \leqslant M \| \delta_i(k) \| \tag{5.86}$$

Then, $e_i(k) \to 0, \forall i \in N$.

5.5.2 Q-learning

Based on the Bellman equation (5.18), the Q-function is defined as

$$Q_i(X_i(k), u_i(k)) = X_i^{\mathrm{T}}(k)(C_{1i}^{\mathrm{T}} Q_i C_{1i}) X_i(k) + u_i^{\mathrm{T}}(k) W_i u_i(k) + \gamma X_i^{\mathrm{T}}(k+1) P_i X_i(k+1) \tag{5.87}$$

By considering (5.16), the Q-function (5.87) becomes

$$Q_i(X_i(k), u_i(k)) = Z_i^{\mathrm{T}}(k) H_i Z_i(k) \tag{5.88}$$

where

$$\begin{aligned} Z_i(k) &= \begin{bmatrix} X_i^{\mathrm{T}} & u_i^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \\ H_i &= \begin{bmatrix} h_{X_i X_i} & h_{X_i u_i} \\ h_{u_i X_i} & h_{u_i u_i} \end{bmatrix} = \begin{bmatrix} C_{1i}^{\mathrm{T}} Q_i C_{1i} + \gamma T_i^{\mathrm{T}} P_i T_i & \gamma T_i^{\mathrm{T}} P_i B_{1i} \\ \gamma B_{1i}^{\mathrm{T}} P_i T_i & W_i + \gamma B_{1i}^{\mathrm{T}} P_i B_{1i} \end{bmatrix} \end{aligned} \tag{5.89}$$

Then, the Bellman equation(5.18) in terms of Q-function (5.88) becomes

$$Z_i^{\mathrm{T}}(k)H_iZ_i(k) = X_i^{\mathrm{T}}(k)(C_{1i}^{\mathrm{T}}Q_iC_{1i})X_i(k) + u_i^{\mathrm{T}}(k)W_iu_i(k) + \gamma Z_i^{\mathrm{T}}(k+1)H_iZ_i(k+1) \quad (5.90)$$

with

$$X_i(k) = \begin{bmatrix} x_i(k) \\ r_0(k) \end{bmatrix} = \begin{bmatrix} x_i(k) \\ r_i(k) \end{bmatrix} + \begin{bmatrix} 0_{n_i\times 1} \\ \delta_i(k) \end{bmatrix} = \hat{X}_i(k) + \delta_{1i}(k) \quad (5.91)$$

Considering (5.91) in the Bellman equation (5.90) yields

$$\begin{aligned} &\hat{Z}_i^{\mathrm{T}}(k)H_i\hat{Z}_i(k) + \begin{bmatrix}\delta_{1i}^{\mathrm{T}}(k) & 0^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}} H_i \begin{bmatrix}\delta_{1i}^{\mathrm{T}}(k) & 0^{\mathrm{T}}\end{bmatrix} = \hat{X}_i^{\mathrm{T}}(k)C_{1i}^{\mathrm{T}}Q_iC_{1i}\hat{X}_i(k) + \delta_{1i}^{\mathrm{T}}(k)C_{1i}^{\mathrm{T}}Q_iC_{1i}\hat{X}_i(k) \\ &+ \hat{X}_i^{\mathrm{T}}(k)C_{1i}^{\mathrm{T}}Q_iC_{1i}\delta_{1i}(k) + \delta_{1i}^{\mathrm{T}}(k)C_{1i}^{\mathrm{T}}Q_iC_{1i}\delta_{1i}(k) + u_i^{\mathrm{T}}(k)W_iu_i(k) + \gamma\hat{Z}_i^{\mathrm{T}}(k+1)\hat{H}_iZ_i(k+1) + \\ &\gamma\begin{bmatrix}\delta_{1i}^{\mathrm{T}}(k+1) & 0^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}} H_i \begin{bmatrix}\delta_{1i}^{\mathrm{T}}(k+1) & 0^{\mathrm{T}}\end{bmatrix} \end{aligned} \quad (5.92)$$

with $\hat{Z}_i(k) = \begin{bmatrix}\hat{X}_i^{\mathrm{T}}(k) & u_i^{\mathrm{T}}(k)\end{bmatrix}^{\mathrm{T}}$. By considering the result of Lemma 5.3 in (5.92), one has

$$\hat{Z}_i^{\mathrm{T}}(k)H_i\hat{Z}_i(k) = \hat{X}_i^{\mathrm{T}}(k)C_{1i}^{\mathrm{T}}Q_iC_{1i}\hat{X}_i(k) + u_i^{\mathrm{T}}(k)W_iu_i(k) + \gamma\hat{Z}_i^{\mathrm{T}}(k+1)\hat{H}_iZ_i(k+1) \quad (5.93)$$

Applying the optimality condition gives

$$u_i^*(k) = -(h_{u_iu_i}^{-1}h_{u_i\hat{X}_i})\hat{X}_i(k) \quad (5.94)$$

Algorithm 5.1. Model-free Q-learning Algorithm

Initialization: Set the iteration number $j = 0$ and start with a stabilizing control policy $u_i^0(k)$.

1. For $j = 0,1,2,...$ solve using LS

$$\hat{Z}_i^j(k)^{\mathrm{T}}H_i^j\hat{Z}_i^j(k) = \hat{X}_i^j(k)^{\mathrm{T}}C_{1i}^{\mathrm{T}}Q_iC_{1i}\hat{X}_i^j(k)^{\mathrm{T}} + u_i^j(k)^{\mathrm{T}}W_iu_i^j(k) + \gamma\hat{Z}_i^j(k+1)^{\mathrm{T}}H_i^j\hat{Z}_i^j(k+1) \quad (5.95)$$

2. Update the control input as

$$u_i^{(j+1)}(k) = -((h_{u_iu_i}^j)^{-1}h_{u_i\hat{X}_i}^j)\hat{X}_i^j(k) \quad (5.96)$$

Remark 5.7. To implement Q-learning, (5.95) must be solved at each step. This equation can be solved online using least-squares method. This requires a persistence of excitation (PE) condition to allow solution of repeated Bellman equation (5.95) at successive time instants in a batch fashion.

Remark 5.8. The ARE (5.21) and the optimal control input (5.19) solve the output synchronization problem 5.1 but they require the dynamics of the leader and agents from (5.16). However, Q-

learning finds the solution to the ARE and the optimal control input without requiring these dynamics and using only measured data along the system trajectories.

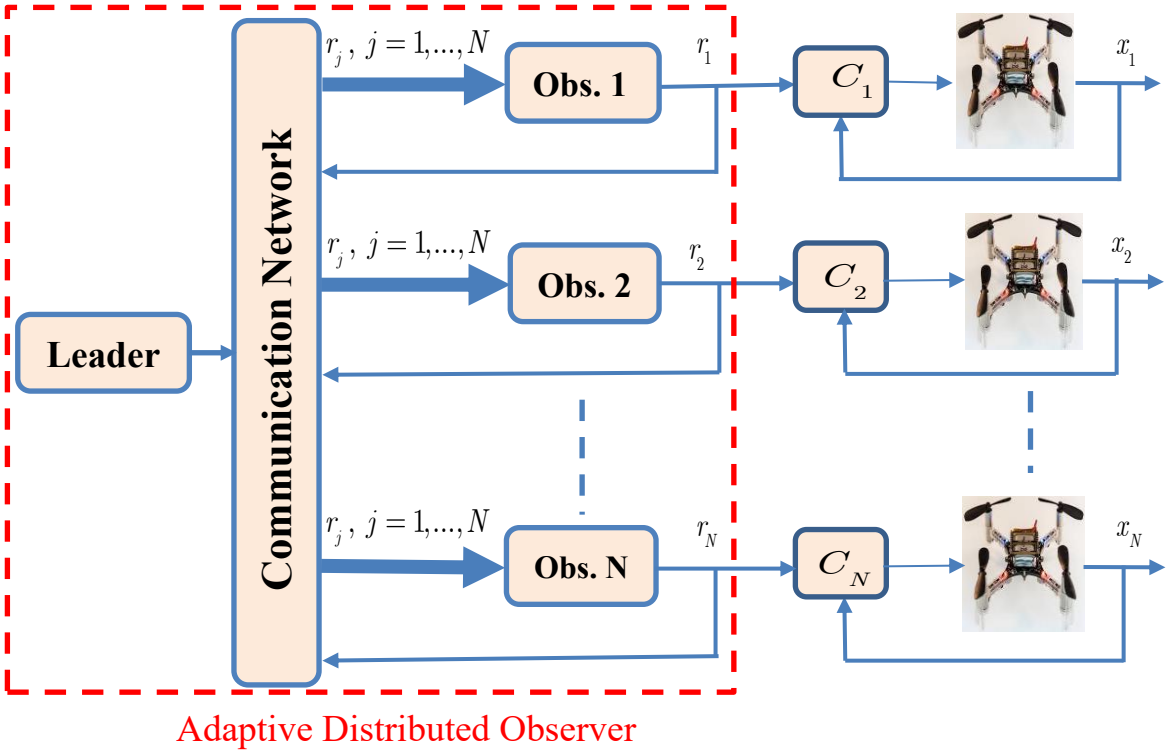

Fig. 5.1. Schematic of the proposed approach

5.4. Simulation Results

In this section, a numerical example is provided to verify the effectiveness of the proposed method. Consider a heterogeneous discrete-time multi-agent system with six agents. The first agent is considered as a leader and its dynamics is given by

$$\begin{aligned} r_0(k+1) &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} r_0(k) \\ y_0(k) &= \begin{bmatrix} 1 & 2 \end{bmatrix} r_0(k) \end{aligned} \tag{5.97}$$

All five other agents are considered as followers and their dynamics are given by (5.1) with

$$
\begin{aligned}
&A_1 = 2, \qquad B_1 = 3, \qquad C_1 = 1 \\
&A_2 = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 2 & 1 \end{bmatrix} \\
&A_3 = \begin{bmatrix} 1 & 1 \\ 10 & 3 \end{bmatrix}, \quad B_3 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \quad C_3 = \begin{bmatrix} 1 & 1 \end{bmatrix} \\
&A_4 = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 3 & 5 \\ 0 & 0 & 4 \end{bmatrix}, B_4 = \begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix}, \quad C_4 = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \\
&A_5 = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 4 \\ 0 & 0 & 4 \end{bmatrix}, B_5 = \begin{bmatrix} 1 & 1 & 1 \\ 5 & 5 & 5 \\ 5 & 5 & 5 \end{bmatrix}, C_5 = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}
\end{aligned}
\tag{5.98}
$$

The directed communication graph for this example is shown in Fig. 5.2.

First, the distributed adaptive observer is implemented to estimate the leader's state for all agents. The gains are chosen as $c = 0.11$ and $\alpha = 0.35$ in (5.63) and (5.71). The weighting matrices and discount factor are given as $Q_1 = 10, Q_2 = 10, Q_3 = 10, Q_4 = 8, Q_5 = 10, \gamma = 0.9$. Fig. 5.3 shows the error between the observer and the leader state for all agents. The solution of the output regulator equation (5.4) for the given heterogeneous systems (5.98) are

$$
\begin{aligned}
&\Pi_1 = \begin{bmatrix} 0.5 & 1 \end{bmatrix} && \Gamma_1 = \begin{bmatrix} 0 & -0.5 \end{bmatrix} \\
&\Pi_2 = \begin{bmatrix} 0.418 & 0.781 \\ 0.163 & 0.436 \end{bmatrix} && \Gamma_2 = \begin{bmatrix} -0.054 & -1.145 \end{bmatrix} \\
&\Pi_3 = \begin{bmatrix} 0.081 & -0.543 \\ 0.918 & 2.543 \end{bmatrix} && \Gamma_3 = \begin{bmatrix} -0.514 & -0.639 \end{bmatrix} \\
&\Pi_4 = \begin{bmatrix} -0.057 & 0.057 \\ -1.485 & -1.514 \\ 1.342 & 1.657 \end{bmatrix} && \Gamma_4 = \begin{bmatrix} -0.743 & -1.057 \end{bmatrix} \\
&\Pi_5 = \begin{bmatrix} -1.370 & -1.629 \\ 1.407 & 1.592 \\ -0.148 & 0.148 \end{bmatrix} && \Gamma_5 = \begin{bmatrix} 0.049 & -0.049 \\ 0.049 & -0.049 \\ 0.049 & -0.049 \end{bmatrix}
\end{aligned}
\tag{5.99}
$$

The solution of ARE (5.21) for all agents are

$$
\begin{aligned}
P_1^\star &= \begin{bmatrix} 40.8840 & -20.4348 & -40.2209 \\ -20.4348 & 34.4260 & 20.6295 \\ -40.2209 & 20.6295 & 64.0252 \end{bmatrix} \\
P_2^\star &= \begin{bmatrix} 2398.2 & -3536.3 & -402.1 & -328.8 \\ -3536.3 & 5390.9 & 563.9 & 414.6 \\ -402.1 & 563.9 & 189.4 & 88.4 \\ -328.8 & 414.6 & 88.4 & 191.5 \end{bmatrix} \\
P_3^\star &= \begin{bmatrix} 354.3886 & 96.6323 & -114.7498 & -50.5744 \\ 96.6323 & 31.9168 & -36.2106 & -27.6901 \\ -114.7498 & -36.2106 & 106.8674 & 92.5375 \\ -50.5744 & -27.6901 & 92.5375 & 107.4225 \end{bmatrix} \\
P_4^\star &= \begin{bmatrix} 14526 & -6187 & -5359 & -1063 & -1212 \\ -6187 & 2825 & 2406 & 562 & 593 \\ -5359 & 2406 & 2093 & 418 & 441 \\ -1063 & 562 & 418 & 350 & 347 \\ -1212 & 593 & 441 & 347 & 373 \end{bmatrix} \\
P_5^\star &= \begin{bmatrix} 146.533 & 167.326 & 122.279 & -16.617 & -45.762 \\ 167.326 & 250.469 & 121.810 & -105.215 & -144.213 \\ 122.279 & 121.810 & 180.276 & 22.774 & -21.367 \\ -16.617 & -105.215 & 22.774 & 129.414 & 136.380 \\ -45.762 & -144.213 & -21.367 & 136.380 & 158.973 \end{bmatrix}
\end{aligned} \tag{5.100}
$$

By using (5.99) and (5.100), it is obvious that (5.53) is satisfied. It can be concluded that the following simulation results are consistent with the results of Theorem 5.4.

Now, the Q-learning Algorithm 5.1 is used to solve the problem. It is assumed that the dynamics of all agents and leader are completely unknown. The control gains converge to

$$
\begin{aligned}
K_1 &= \begin{bmatrix} -0.6630 & 0.3261 & 0.1657 \end{bmatrix} \\
K_2 &= \begin{bmatrix} 3.1679 & -7.7434 & -0.1194 & -0.2251 \end{bmatrix} \\
K_3 &= \begin{bmatrix} -1.0703 & -0.5175 & 0.0462 & 0.0930 \end{bmatrix} \\
K_4 &= \begin{bmatrix} -0.8601 & -0.0194 & -0.6357 & 0.0306 & 0.0142 \end{bmatrix} \\
K_5 &= \begin{bmatrix} -0.0874 & -0.0947 & -0.3254 & 0.0146 & 0.0072 \\ -0.0874 & -0.0947 & -0.3254 & 0.0146 & 0.0072 \\ -0.0874 & -0.0947 & -0.3254 & 0.0146 & 0.0072 \end{bmatrix}
\end{aligned} \tag{5.101}
$$

The optimal gains are computed from the above P_i using (5.19). Fig. 5.4 shows the norm of the difference between the optimal control gain (5.19) and the computed gain (5.96) for all agents. The outputs of the leader and all agents when the control protocol (5.19) with the optimal gains is applied is given in Fig. 5.5.

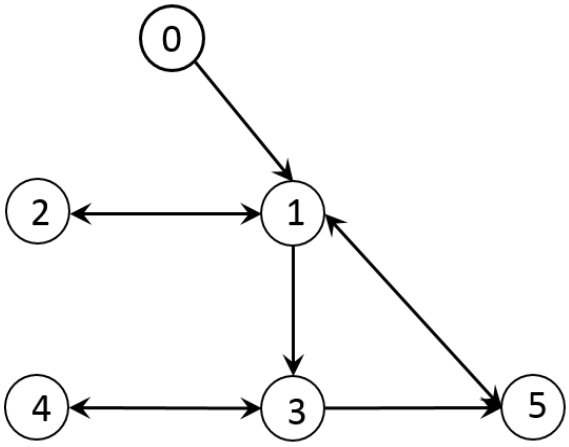

Fig. 5.2. Communication network for the agents and leader

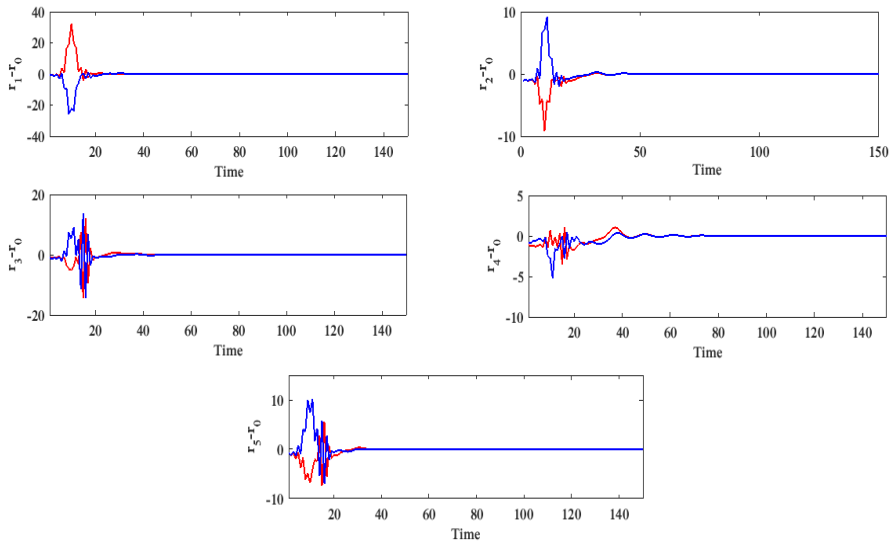

Fig. 5.3. The error between leader state and observer for all agents

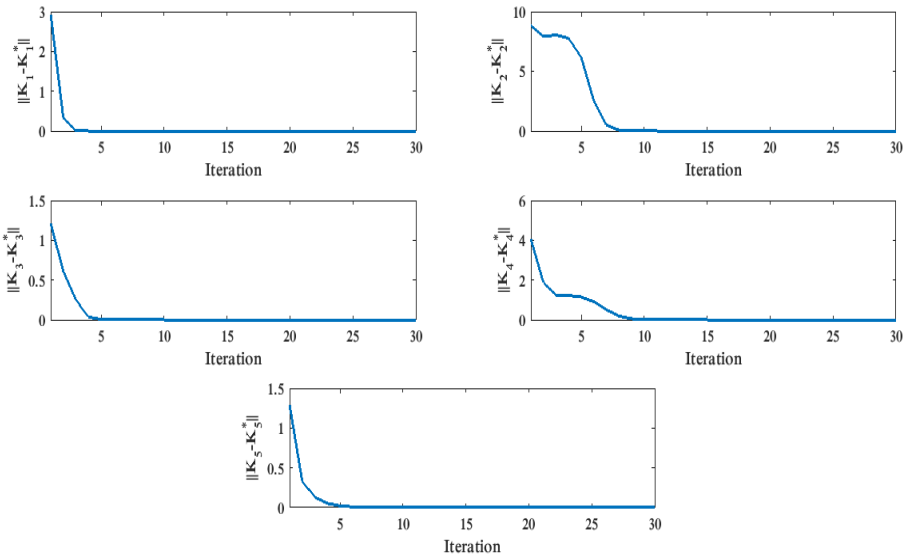

Fig. 5.4. Convergence of the control gains to their optimal values during the learning process for all agents

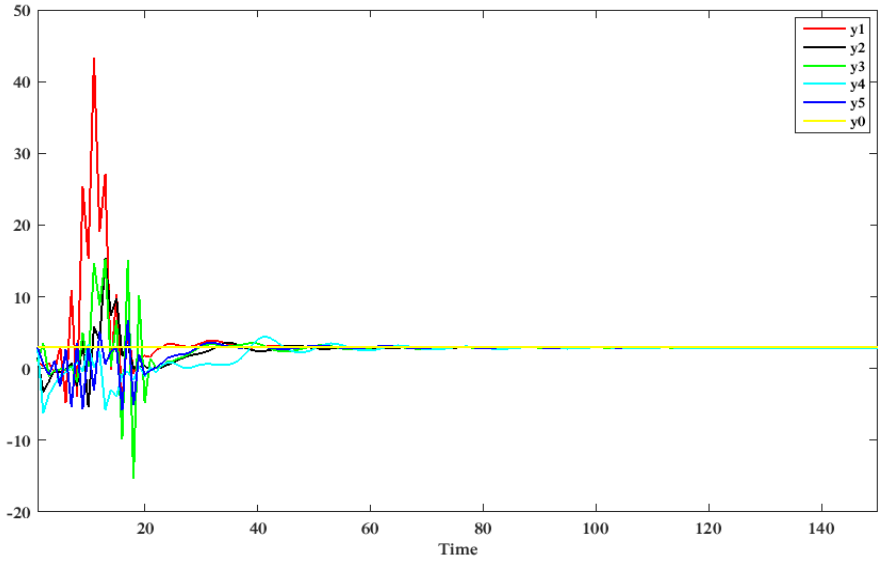

Fig 5.5. The outputs of the leader and all agents

5.5. Conclusion

In thischapter, the optimal model-free output synchronization problem for discrete-time systems is studied. An adaptive distributed observer is designed to provide the estimation of the

leader state to all agents. Q-learning is employed to find the optimal solutions online using only measured data and without requiring complete knowledge of the leader or agents dynamics.

Chapter 6

CONCLUSION AND FUTURE WORK

In this dissertation, reinforcement learning (RL) algorithms are developed to find the optimal solutions to optimal tracking control problems of both linear and nonlinear discrete-time systems. The proposed approaches are online, and do not require complete knowledge of the systems dynamics. Hamilton-Jacobi-Bellman (HJB) equation arising in the H_2 optimal control problem and Hamilton-Jacobi-Isaacs (HJI) equation arising in the H_∞ optimal control problem are solved online in real time and using measured data along the system trajectory. Both on-policy and off-policy RL algorithms are developed. On-policy RL is used to solve linear and nonlinear H_2 optimal control problems. In contrast to the existing methods, the proposed approach for nonlinear systems takes into account the input constraints in the optimization problem by using a nonquadratic performance function. Off-policy RL is employed to solve the game algebraic Riccati equation (GARE) online using the measured data along the system trajectories. The proposed method has two main advantages compared to the other model-free methods. First, the disturbance input does not need to be adjusted in a specific manner. Second, there is no bias as a result of adding a probing noise to the control input to maintain persistence of excitation (PE) condition. Extension to multi-agent systems is also considered. Optimal model-free solution is presented to the output synchronization of heterogeneous multi-agent discrete-time systems. It is shown that the proposed method implicitly solves the output regulator equations and therefore solves the output synchronization problem.

The following are some of the directions for continuation of this work.

1. Design of an online model-free solution to the optimal tracking control of nonlinear affine systems.
2. Propose a deep neural network to approximate a more accurate structure of the value function for nonlinear systems and avoid divergence of the RL algorithm and consequently

instability of the feedback control system in the presence of high-dimensional inputs and unstructured input data such as images.

3. Extend the proposed results to output synchronization of multi-agent discrete-time systems with nonlinear dynamics.
4. Apply the proposed methods to practical systems such as autonomous robots, and power and energy systems.

REFERENCES

[1] R. Stengel, Optimal theory and estimation. Dover books on advanced mathematics, Dover publications, Mineola, NY, 1986.

[2] F. L. Lewis, D. Vrabie, and V. L. Syrmos, Optimal Control. John Wiley, 2012.

[3] D. Liberzon, Calculus of variations and Optimal control theory: A concise of introduction. Princeton university press, Princeton, NJ, 2011.

[4] M. Athans and P. Falb, Optimal control: An introduction to the theory and its applications. Dover books on engineering series, Dover publications, Mineola, NY, 2006.

[5] D. Kirk, Optimal control theory: An introduction. Dover books on electrical engineering, Dover publications, Mineola, NY, 2012.

[6] A. Bryson, Applied Optimal control:optimization, estimation and control. Halsted Press book, Taylor and Francis, NewYork, 1975.

[7] A. Mannava, S. N. Balakrishnan, L. Tang, and R. G. Landers. (2012). Optimal tracking control of motion systems. IEEE Transactions on Control Systems and Technology, 20(6), 1548–1556.

[8] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction (2nd Edition, in preparation), vol. 1. MIT press Cambridge, 2017.

[9] D. P. Bertsekas and J. N. Tsitsiklis, Neuro-dynamic programming. Athena Scientific, MA, 1996.

[10] W. B. Powell, Approximate Dynamic Programming. Hoboken, NJ, Wiley, 2007.

[11] A. G. Barto, J. Si, W. B. Powell, and D. Wunch, Handbook of Learning and Approximate Dynamic Programming. New York, NY, USA: Wiley, 2004.

[12] R. A. Howard. Dynamic programming and markov processes. Cambridge, MA: MIT Press,1960.

[13] F. L. Lewis and D. Vrabie. (2009). Reinforcement learning and adaptive dynamic programming for feedback control. IEEE Circuits Syst. Mag., 9(3), 32–50.

[14] F. L. Lewis, K. G. Vamvoudakis, and D. Vrabie. Optimal adaptive control and differential games by reinforcement learning principles. London: Institution of Engineering and Technology, 2013.

[15] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. IEEE Control Syst. 32(6), 76–105.

[16] F. Y. Wang, H. Zhang, and D. Liu. (2009). Adaptive dynamic programming: an introduction, IEEE Computational Intelligence Magazine, 39-47.

[17] F. L. Lewis and D. Liu. Reinforcement Learning and Approximate Dynamic Programming for Feedback Control. Hoboken, NJ, USA: Wiley, 2013.

[18] H. Zhang, D. Liu, Y. Luo, and D. Wang. Adaptive dynamic programming for control: algorithms and stability. London: Springer-Verlag, 2012.

[19] X.-R. Cao. Stochastic Learning and Optimization. Springer US, 2007.

[20] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li. Adaptive dynamic programming with application in optimal control. Springer International Publishing, 2017.

[21] P. J. Werbos. (1989). Neural networks for control and system identification. In Decision and Control (CDC), IEEE 28st Annual Conference on. 260–265.

[22] P. J. Werbos. A menu of designs for reinforcement learningover time. MIT press, 1991.

[23] P.J. Werbos. (1992). Approximate dynamic programming for real time control and neural modeling. In D. A. White, & D. A. Sofge (Eds.), Handbook of intelligent control. Multiscience Press.

[24] C. Watkins. Learning from delayed rewards. Ph.D. thesis, Cambridge Univ., Cambridge, England, 1989.

[25] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf. (2008). Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 38(4), 943– 949.

[26] L. C. Baird. (1994). Reinforcement learning in continuous time: advantage updating. In Proc. of ICNN.

[27] S. Bhasin, R. Kamalapurkar, M. Johnson, K. G. Vamvoudakis, F. L. Lewis, and W. E. Dixon (2012). A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems. Automatica, 49, 82–92.

[28] S. J. Bradtke, B. E. Ydestie, and A. G. Barto. (1994). Adaptive linear quadratic control using policy iteration. In: Proc. of ACC, 3475–3476.

[29] K. Doya. (2000). Reinforcement learning in continuous time and space. Neural Computation, 12, 219-245.

[30] J. Y. Lee, J. B. Park, and Y. H. Choi. (2012). Integral Q-learning and explorized policy iteration for adaptive optimal control of continuous-time linear systems. Automatica, 48, 2850–2859.

[31] F. L. Lewis and K. G. Vamvoudakis. (2011). Reinforcement learning for partially observable dynamic processes: adaptive dynamic programming using measured output data. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 41(1), 14–23.

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

[33] D. Liu and Q. Wei. (2013). Finite-approximation-error-based optimal control approach for discretetime nonlinear systems. IEEE Transactions on Cybernetics, 43, 779–789.

[34] D. Liu and Q. Wei. (2014). Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. IEEE Transactions on Neural Networks and Learning Systems, 25(3), 621–634.

[35] P. He and S. Jagannathan. (2007). Reinforcement learning neural network- based controller for nonlinear discrete-time systems with input constraints. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics). 37(2), 425–436.

[36] D. Liu, X. Yang, and H. Li. (2013). Adaptive optimal control for a class of continuoustime affine nonlinear systems with unknown internal dynamics. Neural Computing and Applications, http://dx.doi.org/10.1007/s00521-012-1249-y.

[37] J. J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks. (2002). Adaptive dynamic programming. IEEE Transactions on Systems Man, and Cybernetics, Part C: Applications and Reviews, 32, 140–153.

[38] D. V. Prokhorov and D. C. Wunsch. (1997). Adaptive critic designs. IEEE Trans. Neural Netw. 8(5). 997–1007.

[39] Y. Luo and H. Zhang. Approximate optimal control for a class of nonlinear discrete-time systems with saturating actuators. (2008). Prog. Natural Sci. 18(8). 1023–1029.

[40] R. Song, W. Xiao, and Q. Wei. (2013). Optimal Tracking Control for a Class of Nonlinear Time-Delay Systems with Actuator Saturation. Advances in Brain Inspired Cognitive Systems-Lecture Notes in Computer Science, 7888, 208-215.

[41] R. Song, W. Xiao, H. Zhang, and C. Sun. (2014). Adaptive dynamic programming for a class of complex-valued nonlinear systems. IEEE Transactions on Neural Networks and Learning Systems, 25(9), 1733-1739. IEEE Trans. Neural Netw. 12(2). 264–276.

[42] 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.

[43] K. Vamvoudakis and F. L. Lewis. (2010). Online actor-critic algorithm to solve the continuous infinite-time horizon optimal control problem. Automatica, 46, 878–888.

[44] K. Vamvoudakis, D. Vrabie, and F. L. Lewis. (2013). Online adaptive algorithm for optimal control with integral reinforcement learning. International Journal of Robust and Nonlinear Control, http://dx.doi.org/10.1002/rnc, in press.

[45] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis. Optimal adaptive control and differential games by reinforcement learning principles. London: Institution of Engineering and Technology, 2013.

[46] D. Wang, D. Liu, and Q. Wei. (2012). Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach. Neurocomputing, 78, 14–22.

[47] Q. Wei, D. Liu, G. Shi, and Y. Liu. (2015). Multi-batter optimal coordination control for home energy management systems via distributed iterative adaptive dynamic programming. IEEE Transactions on Industrial Electronics, 62(7), 4203- 4214.

[48] Q. Wei, D. Liu, and D. Shi. (2015). A novel dual iterative Q-learning method for optimal battery management in smart residential environments, IEEE Transactions on Industrial Electronics, 62(4), 2509-2518.

[49] Q. Wei and D. Liu. (2014). Data-driven neuro-optimal temperature control of water gas shift reaction using stable iterative adaptive dynamic programming. IEEE Transactions on Industrial Electronics, 61(11), 6399–6408.

[50] H. N. Wu and B. Luo. (2013). Simultaneous policy update algorithms for learning the solution of linear continuous-time H∞ state feedback control. Information Sciences, 222, 472–485.

[51] H. Xu, S. Jagannathan, and F. L. Lewis. (2012). Stochastic optimal control of unknown linear networked control system in the presence of random delays and packet losses, Automatica, 48, 1017–1030.

[52] D. Vrabie and F. L. Lewis. (2009). Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. Neural Networks, 22, 237–246.

[53] D. Vrabie, O. Pastravanu, M. Abou-Khalaf, and F. L. Lewis. (2009). Adaptive optimal control for continuous-time linear systems based on policy iteration. Automatica, 45, 477–484.

[54] M. Abu-Khalaf and F. L. Lewis. (2008). Neurodynamic Programming and Zero-Sum Games for Constrained Control Systems, IEEE Transactions on Neural Networks, 19(7), 1243-1252.

[55] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf. (2007). Model-free Q-learning designs for linear discrete-time zero-sum games with application to H∞ control. Automatica, 43(3), 473–481.

[56] H. Li, D. Liu, and D. Wang. (2014). Integral Reinforcement Learning for Linear Continuous-Time Zero-Sum Games With Completely Unknown Dynamics, IEEE Transactions on Automation Science and Engineering, 11(3), 706-714.

[57] K. Vamvoudakis and F. L. Lewis. (2010). Online solution of nonlinear two-player zero sum games using synchronous policy iteration. In Proceedings of 49th IEEE Conference on Decision and Control, Atlanta, (pp. 3040–3047).

[58] K. Vamvoudakis and F. L. Lewis. (2011). Multi-player non-zero-sum games: online adaptive learning solution of coupled Hamilton–Jacobi equations. Automatica, 47(8), 1556–1569.

[59] H. N. Wu and B. Luo. (2012). Neural network based online simultaneous policy update algorithm for solving the HJI equation in nonlinear H∞ control. IEEE Transactions on Neural Networks and Learning Systems, 23 (12), 1884–1895.

[60] H. Zhang, Q. Wei, and D. Liu. (2011). An iterative approximate dynamic programming method to solve for a class of nonlinear zero-sum differential games. Automatica, 47(1), 207–214.

[61] D. Liu, H. Li, and D. Wang. (2014). Online synchronous approximate optimal learning algorithm for multi-player non-zero-sum games with unknown dynamics. IEEE Trans. Syst., Man, Cybern., Syst. 44(8). 1015–1027.

[62] D. A. White and D. A. Sofge. (Eds.) (1992). Handbook of Intelligent Control, New York: Van Nostrand Reinhold.

[63] Y. Jiang and Z.P. Jiang. (2012). Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. Automatica, 48, 2699–2704.

[64] J. Y. Lee, J. B. Park, and Y. H. Choi. (2012). Integral Q-learning and explorized policy iteration for adaptive optimal control of continuous-time linear systems. Automatica, 48, 2850–2859.

[65] D. Liu, Y. Huang, and Q. Wei. (2013). Neural Network H∞ Tracking Control of Nonlinear Systems Using GHJI Method, Advances in Neural Networks – ISNN 2013 Lecture Notes in Computer Science, 7952, 186-195.

[66] D. Wang, D. Liu, and Q. Wei. (2012). Finite-horizon neurooptimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach. Neurocomputing, 78, 14–22.

[67] H. Zhang, Q. Wei, and Y. Luo. (2008). A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 38, 937–942.

[68] H. Zhang, L. Cui, X. Zhang, and X. Luo. (2011). Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. IEEE Transactions on Neural Networks, 22, 2226– 2236.

[69] Y. Huang and D. Liu. (2014). Neural-network-based optimal tracking control scheme for a class of unknown discrete-time nonlinear systems using iterative ADP algorithm. Neurocomputing, 125, 46–56.

[70] T. Dierks and S. Jagannathan. (2009). Optimal tracking control of affine nonlinear discrete-time systems with unknown internal dynamics. In Joint 48th IEEE conference on decision and control and 28th Chinese control conference Shanghai, PR China (pp. 6750–6755).

[71] M. Abu-Khalaf and F. L. Lewis. (2005). Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. Automatica, 41(5). 779–791.

[72] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani. (2013). Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks." IEEE Trans. Neural Netw. Learn. Syst. 24(10). 1513–1525.

[73] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani (2014). Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. Automatica. 50(1). 193–202.

[74] A. J. Van der Schaft. (1992). L2-gain analysis of nonlinear systems and nonlinear state feedback H∞control. IEEE Transactions on Automatic Control, 37(6), 770–784.

[75] G. Zames. (1981). Feedback and optimal sensitivity: model reference transformations, multiplicative seminorms, and approximate inverses. IEEE Transactions on Automatic Control, 26(2), 301–320.

[76] J. C. Doyle, K. Glover, P. P. Khargonekar, and B. A. Francis. (1989). State-space solutions to standard H2 and H∞ control problems. IEEE Transactions on Automatic Control, 34(8), 831–847.

[77] T. Basar and P. Bernard. H∞ optimal control and related minimax design problems. Boston, MA: Birkhäuser, 1995.

[78] H. Zhang, T. Feng, H. Liang, and Y. Luo. (2015). Lqr-based optimal distributed cooperative design for linear discrete-time multiagent systems. Neural Networks and Learning Systems, IEEE Transactions on, 1(13).

[79] H. Zhang, F. L. Lewis, and A. Das. (2011). Optimal design for synchronization of cooperative systems: State feedback, observer and output feedback. Automatic Control, IEEE Transactions on, 56(8).1948-1952.

[80] Z. Li, Z. Duan, G. Chen, and L. Huang. (2010). Consensus of multiagent systems and synchronization of complex networks: A unified viewpoint. IEEE Transactions on Circuits and Systems I: Regular Papers, 57(1). 213-224.

[81] Z. Qu. Cooperative control of dynamical systems: Applications to autonomous vehicle. Springer-Verlag, London, 2009.

[82] K. Hengster-Movric, K. You, F. L. Lewis, and L. Xie. (2013). Synchronization of discrete-time multi-agent systems on graphs using riccati design. Automatica, 49(2). 414-423.

[83] S. Li, G. Feng, X. Luo, and X. Guan. (2015). Output consensus of heterogeneous linear discrete-time multiagent systems with structural uncertainties. Cybernetics, IEEE Transactions on, 45(12). 2868-2879.

[84] H. Liang, H. Zhang, and Z.Wang. (2015). Distributed-observer-based cooperative control for synchronization of linear discrete-time multi-agent systems. ISA Transaction, 59. 72-78.

[85] Y. Yan and J. Huang. (2016). Coperative output regulation of discrete-time linear time-delay multi-agent systems. IET Control Theory and Applications, 10(16). 2019-2026.

[86] T. Yang, X. Wang, A. Saberi, and A.A. Stoorvogel. (2013). Output synchronization for heterogeneous networks of discrete-time introspective right-invertible agents with uniform constant communication delay. American Control Conference, pp. 516-521.

[87] X. Wang, A. Saberi, and T. Yang. (2014). Synchronization in heterogeneous networks of discrete-time introspective right invertible agents. Internation journal of robust and nonlinear control, 24, 3255-3281.

[88] G. A. Hewer.(1971). An iterative technique for the computation of steady state gains for the discrete optimal regulator. IEEE Transactions on Automatic Control, 16(4), 382–384.

[89] T. Landelius and H. Knutsson. Greedy adaptive critics for LQR problem: convergence proof. Technical report, Linkoping, Sweden, Computer vision laboratory, 1996.

[90] A. Isidori, Nonlinear Control Systems, 3rd ed. London, U.K.: Springer-Verlag, 1995.

[91] S. E. Lyshevski. (1998). Optimal control of nonlinear continuous-time systems: Design of bounded controllers via generalized nonquadratic functionals. In Proc. IEEE ACC, 205–209.

[92] K. Doya. (2000). Reinforcement learning in continuous time and space. Neural Comput. 2(1), 219–245.

[93] X. Luo and J. Si. (2013). Stability of direct heuristic dynamic programming for nonlinear tracking control using PID neural network. In Proc. IJCNN, Dallas, TX, USA, 1–7.

[94] B. Luo, H. Wu, and T. Huang. (2015). Off-policy reinforcement learning for control design. IEEE Transactions on Cybernetics, 45(1), 65-76.

[95] B. Luo, H. Wu, T. Huang, and D. Liu. (2014). Data-based approximate policy iteration for affine nonlinear continuous-time optimal control design. Automatica, 50(12), 3281-3290.

[96] B. Luo, H. Wu, T. Huang, and D. Liu. (2015). Reinforcement learning so-lution for HJB equation arising in constrained optimal control prob-lem. Neural Networks, 71, 150-158.

[97] B. Luo, T. Huang, H. Wu, and X. Yang. (2015). Data-drivencontrol for nonlinear distributed parameter systems. IEEE Transactions on Neural Networks and Learning Systems, 26(11), 2949-2961.

[98] H. Modares, F. L. Lewis, and Z. P. Jiang. (2015). Tracking control of completely unknown continuous-time systems via off-policy reinforcement learning. IEEE Transactions on Neural Networks and Learning Systems, 26(10), 2550-2562.

[99] H. Li, D. Liu, and D. Wang. (2014). Integral reinforcement learning for linear continuous-time zero-sum games with completely unknown dy-namics. IEEE Transactions on Automation Science and Engineering, 11(3), 706-714.

[100] B. M. Chen. Robust and H∞ control. London: Springer –Veriag, 2000.

[101] A. A. Stoorvogel and A. J. T. M. Weeren. (1994). The discrete-time Riccati equation related to the H∞ control problem. IEEE Transactions on Automatic Contro. 39(3), 686-691.

[102] A. Jadbabaie, L. Jie, and A. S. Morse. (2003). Coordination of groups of of mobile autonomous agents using nearest neighbor rules. Automatic Control, IEEE Transactions on, 48(6):988-1001.

[103] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das. Cooperative control of multi-agent systems: optimal and adaptive design approaches. Springer-Verlag, 2014.

[104] R. Olfati-Saber and R. M. Murray. (2004). Consensus problems in networks of agents with switching topology and time-delays. Automatic Control, IEEE Transactions on, 49(9),1520-1533.

[105] W. Ren and R. W. Beard. Distributed consensus in multi- vehicle cooperative control. Springer-Verlag, London, 2008.

[106] L. Scardovi and R. Sepulchre. (2009). Synchronization in networks of identical linear systems. Automatica, 45(11), 2557-2562.

[107] L. Marconi A. Isidori and G. Casadei. (2014). Robust output synchronization of a network of heterogeneous nonlinear agents via nonlinear regulation theory. Automatic Control, IEEE Transactions on, 59(10), 2680-2691.

[108] Z. Ding. (2015). Adaptive consensus output regulation of a class of nonlinear systems with unknown high-frequency gain. Automatica, 51, 348-55.

[109] Y. Hong, X. Wang, and Z.-P. Jiang. (2013). Distributed output regulation of leaderfollower multi-agent systems. Int. J. Robust Nonlinear Control, 23, 48-66.

[110] J. Huang and Z. Chen. (2004). A general framework for tackling the output regulation problem. Automatic Control, IEEE Transactions on, 49(12), 2203-2218.

[111] L. Liu and J. Huang. (2008). Global robust output regulation of lower triangular systems with unknown control direction. Automatica, 44(5),1278-1284.

[112] H. Modares, S. P. Nageshra, G. A. Delgado Lopes, R. Babuka, and F. L. Lewis. (2016). Optimal model-free output synchronization of heterogeneous systems using off-policy reinforcement learning. Automatica, 71, 334-341.

[113] Y. Su and J. Huang. (2012). Cooperative output regulation of linear multi-agent systems. Automatic Control, IEEE Transactions on, 57(4), 1062-1066.

[114] P. Wieland, R. Sepulchre, and F. Allgwer. (2011). An internal model principle is necessary and su_cient for linear output synchronization. Automatica, 47(5),1068-1074.

[115] J. Xiang, W. Wei, and Y. Li. (2009). Synchronized output regulation of linear networked systems. Automatic Control, IEEE Transactions on, 54(6), 1336-1341.

[116] T. Yang, A. Saberi, A. A. Stoorvogel, and H. F. Grip. (2014). Output synchronization for heterogeneous networks of ntrospective right-invertible agents. Int. J. Robust Nonlinear Control, 24, 1821-1844.

[117] T. Yang, X. Wang, A. Saberi, and A.A. Stoorvogel. (2013). Output synchronization for heterogeneous networks of discrete-time introspective right-invertible agents with uniform constant communication delay. American Control Conference, 516-521.

[118] L. Yu and J. Wang. (2013). Robust cooperative control for multi- agent systems via distributed output regulation. Systems and Control Letters, 62(11),1049-1056.

[119] S. Li, G. Feng, X. Luo, and X. Guan. (2015). Output consensus of heterogeneous linear discrete-time multiagent systems with structural uncertainties. Cybernetics, IEEE Transactions on, 45(12), 2868-2879.

[120] H. Liang, H. Zhang, and Z.Wang. (2015). Distributed-observer-based cooperative control for synchronization of linear discrete- time multi-agent systems. ISAg Transactions, 59, 72-78.

Biographical Information

Bahare Kiumarsi received the B.S. degree from Shahrood University of Technology, Iran, 2009 and the M.S. degree from Ferdowsi University of Mashhad, Iran, 2013. She achieved the Ph.D. degree from the University of Texas at Arlington, Arlington, TX, USA in August 2017. She was the winner of UT-Arlington N.M. Stelmakh Outstanding Student Research Award and also recipient of the UT-Arlington Graduate Dissertation Fellowship for summer 2017. Her research interests include distributed control, optimal control, reinforcement learning and neural networks.