

## Article

# Off-Policy Inverse Q-Learning Algorithm with Unobservable States

Zhenglin Li <sup>1,2,\*</sup>, Liqi Zhang <sup>1,2</sup>, Xisheng Dai <sup>3</sup> , Hao Fang <sup>4</sup> , Xuhao Liang <sup>1</sup> and Peizhong Li <sup>1</sup>

<sup>1</sup> School of Automation, Guangxi University of Science and Technology, Liuzhou 545006, China; m13317679386@163.com (L.Z.); 13266275236@163.com (X.L.); kndx949230901@163.com (P.L.)

<sup>2</sup> Guangxi Key Laboratory of Logistics Unmanned Aircraft Technology for Transportation Industry, Liuzhou 545006, China

<sup>3</sup> School of Computer and Science, Guangdong University of Finance, Guangzhou 510520, China; mathdxs@163.com

<sup>4</sup> School of Art and Design, Wuhan Institute of Technology, Wuhan 430205, China; fanghao@wit.edu.cn

\* Correspondence: 100000295@gxust.edu.cn

## Abstract

To address the problem of an unknown performance index in discrete-time (DT) linear systems with unobservable states, this paper investigates the output-feedback inverse reinforcement learning (IRL) problem and proposes a data-driven output-feedback off-policy inverse Q-learning algorithm. The proposed method relies solely on input–output trajectory data from an expert system. It does not require a system-dynamics model or state information to identify the unknown cost function and learn an optimal control policy. First, for cases where the system dynamics and the target control gain are known, we propose a model-based method. Building on this foundation, we further develop a model-free method that does not require a system-dynamics model or state information. The proposed algorithm uses state reconstruction to transform the state-feedback Q-function equation into an input–output form. Simulations using an F-16 aircraft model demonstrate that the learner system can approximate the expert trajectories and effectively reconstruct the performance-index function.

**Keywords:** unobservable state; off-policy; inverse reinforcement learning; optimal control

## 1. Introduction

With the advancement of optimal control theory, it has become possible to design control strategies based on known system models and performance indexes [1]. However, precise models and performance indexes for real world systems are often difficult to obtain, which makes the direct application of traditional optimal-control methods challenging [2,3]. Reinforcement learning, as a significant advancement in this field, relaxes the requirements for system models and allows control strategies to be learned incrementally through interaction with the environment. Nevertheless, it also relies on a predefined and well-designed cost function. When the performance index cannot be clearly defined, both approaches face fundamental challenges.

To address this issue, inverse reinforcement learning (IRL) [4,5], also known as imitation learning, was proposed. It reconstructs the latent cost function from expert demonstration data. Unlike traditional reinforcement learning, IRL can infer the underlying objective function by observing the expert's optimal behaviour [6–8]. However, most existing research focuses on state-feedback scenarios and typically assumes that the system state is



Academic Editor: Ioan Ursu

Received: 7 August 2026

Revised: 3 September 2026

Accepted: 7 September 2026

Published: 9 September 2026

**Copyright:** © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\) license](https://creativecommons.org/licenses/by/4.0/).

fully observable. Only [9,10] address the issue of unobservable system states. Xue et al. [9] proposed an algorithm based on static output-feedback control for continuous-time systems under external disturbances. However, this method employs a double-loop iterative structure, requiring repeated updates of the control policy and cost function in both inner and outer loops, resulting in high computational complexity. In recent years, IRL has been extensively studied in random systems [11–15], dynamic systems [16–19], and trajectory emulation [20–22]. It has gradually been integrated with methods such as data-driven control [21] and inverse optimal control, forming a class of important tools capable of simultaneously explaining expert behaviour and recovering control laws.

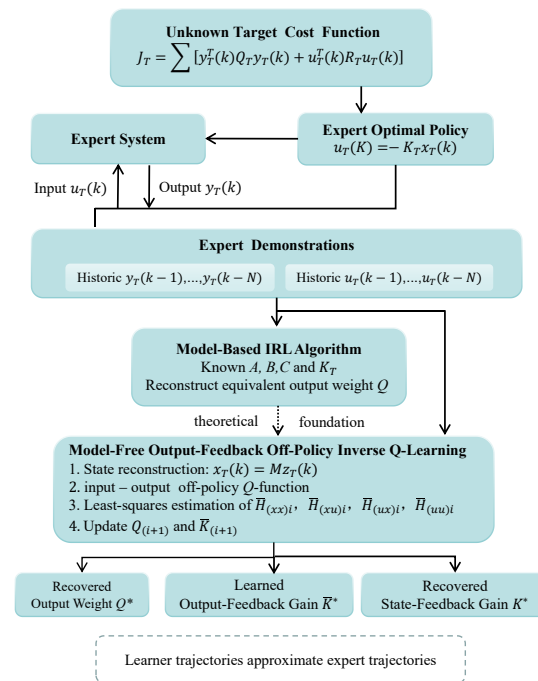
Related advances in control and optimization have also considered resilience and cooperation in complex networked systems. Adaptive fuzzy resilient decentralized control has been investigated for nonlinear large-scale cyber-physical systems under denial-of-service attacks [23]. Bilateral cooperative control has also been studied for nonlinear multi-agent systems with state and output quantization [24]. These studies highlight the importance of robustness to communication constraints, external attacks, quantization effects, and incomplete information. However, the existing literature has generally relied on prescribed system models and controller structures, whereas the present work focuses on data-driven inverse Q-learning for discrete-time linear systems with unobservable states.

Motivated by the above problems, this paper investigates a class of DT linear systems with unobservable states and focuses on the output-feedback inverse Q-learning problem. In contrast to existing IRL methods, the algorithm proposed in this paper does not rely on system state parameters and can be applied to output-feedback scenarios where the state is completely unobservable. First, for conditions where the system dynamics and the target control gain are known, a model-based IRL method is developed to reconstruct an output weighting matrix equivalent to the performance criterion underlying the expert policy. On this basis, by using the state reconstruction technique, the off-policy Q-function equation in the state-feedback form is transformed into an output-feedback form that depends only on expert input–output data. Furthermore, a model-free output-feedback off-policy inverse Q-learning algorithm is proposed. The proposed algorithm does not require the system dynamics or state information and is capable of reconstructing the unknown cost function and learning the control policy solely from the expert system's input–output trajectories.

Figure 1 outlines the proposed framework, in which a model-based IRL scheme is initially developed and subsequently generalized to a model-free method using accessible input–output measurements. The main contributions of this paper are summarized as follows:

- (1) For discrete-time linear systems with inaccessible state variables, a model-based IRL framework is initially constructed. Provided that the plant model and the target feedback gain are known, the framework recovers an equivalent output weighting matrix associated with the expert performance criterion. Theoretical results are then derived to verify both the convergence of the learning process and the stability of the resulting closed-loop system.

- (2) Compared with existing IRL methods, the algorithm proposed in this paper does not rely on system state parameters and can be applied to output-feedback scenarios where the state is completely unobservable. Furthermore, unlike previous work on IRL for DT systems [10,25], this study constructs an off-policy algorithm by introducing an auxiliary term. We then conduct a theoretical analysis of the algorithm's convergence and unbiasedness. Notably, the unbiasedness has not been addressed in existing IRL literature [9,10,16,25–27].



**Figure 1.** Overall framework of the proposed inverse Q-learning method for DT systems with unobservable states. The expert system generates input–output demonstration data. Based on the model-based IRL framework and the state reconstruction mechanism, the proposed model-free output-feedback off-policy inverse Q-learning algorithm reconstructs the equivalent output weight and learns a control gain close to the expert policy.

The organization of the paper is as follows. The problem formulation and the inverse reinforcement learning objective are introduced in Section 2. A model-based IRL design, together with its theoretical analysis, is given in Section 3. Section 4 presents the model-free output-feedback off-policy inverse Q-learning approach and discusses both its implementation and analytical results. Simulation examples are provided in Section 5, and the paper is concluded thereafter.

## 2. Problem Formulation

Consider a linear DT learner system with an unobservable state:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \\ y(k) &= Cx(k). \end{aligned} \quad (1)$$

where  $x(k) \in \mathbb{R}^n$  is the system state,  $u(k) \in \mathbb{R}^m$  is the control input, and  $y(k) \in \mathbb{R}^q$  is the system output.  $A \in \mathbb{R}^{n \times n}$  is the state matrix,  $B \in \mathbb{R}^{n \times m}$  is the input matrix, and  $C \in \mathbb{R}^{q \times n}$  is the output matrix.

The corresponding expert system is described as

$$\begin{aligned} x_T(k+1) &= Ax_T(k) + Bu_T(k), \\ y_T(k) &= Cx_T(k). \end{aligned} \quad (2)$$

where  $x_T(k) \in \mathbb{R}^n$  is the target state,  $u_T(k) \in \mathbb{R}^m$  is the target control input, and  $y_T(k) \in \mathbb{R}^q$  is the target system output. State matrix  $A$ , input matrix  $B$ , and output matrix  $C$  are the same as the learner system (1).

**Remark 1.** The learner system (1) and the expert system (2) share the same system dynamics.

**Assumption 1.** The matrix  $C$  is full-rank. The pair  $(A, B)$  is controllable, and the pair  $(A, C)$  is observable.

For expert systems, the following target cost function should be considered:

$$V_T^*(x_T(k)) = \min_{u_T(k)} \sum_{j=k}^{\infty} (y_T^T(j) Q_T y_T(j) + u_T^T(j) R_T u_T(j)) \quad (3)$$

where  $Q_T^T = Q_T \in \mathbb{R}^{q \times q}$  and  $R_T^T = R_T \in \mathbb{R}^{m \times m}$  are target weight matrices. Equation (3) describes the output-feedback optimal control problem corresponding to the expert system. Since the system state cannot be directly measured, the state will be reconstructed in the subsequent sections using input–output data.

Under the optimal control framework, the target optimal control law of the expert system (2) can be expressed as

$$u_T(k) = -K_T x_T(k) \quad (4)$$

where  $K_T$  denotes the optimal control gain, which satisfies

$$K_T = (R_T + B^T P_T B)^{-1} (B^T P_T A) \quad (5)$$

where  $P_T^T = P_T \in \mathbb{R}^{n \times n} > 0$  satisfies the following target DT algebraic Riccati equation (ARE):

$$P_T = A^T P_T A + C^T Q_T C - A^T P_T B (R_T + B^T P_T B)^{-1} B^T P_T A. \quad (6)$$

Substituting  $K_T$  in (5) into (6) yields

$$(A - B K_T)^T P_T (A - B K_T) + K_T^T R_T K_T + C^T Q_T C - P_T = 0. \quad (7)$$

Equation (7) shows that, under the optimal control strategy, the cost function of the expert system satisfies the corresponding ARE, thereby providing a theoretical foundation for the formulation of subsequent IRL problems.

**Assumption 2.** The target weights  $Q_T$  and  $R_T$  of the expert system (2), as well as the corresponding optimal control gain  $K_T$ , are unknown to the learner (1).

**Assumption 3.** The input–output trajectories  $u_T(k)$  and  $y_T(k)$  of the expert system (2) are available to the learner (1), while the states  $x_T(k)$  of both the expert system and the learner system (1) cannot be observed.

**Definition 1** (Equivalent Weight). Given any  $R = R^T \in \mathbb{R}^{m \times m} > 0$ , if there exists a  $Q \in \mathbb{R}^{q \times q} \geq 0$  such that the cost function determined by  $(Q, R)$  yields the same  $K_T$  as the expert (2), then  $Q$  is called an equivalent weight of  $Q_T$ , and the corresponding cost function is said to be equivalent to the target cost function (3) of the expert system.

**Definition 2** (IRL Problem). The objective of IRL is to reconstruct, based only on the input–output trajectory data  $(y_T(k), u_T(k))$  of the expert system (2), a set of cost-function weight parameters that are equivalent in terms of the target cost function, such that the learner system (1) can not only recover the unknown cost function but also generate the same optimal control policy as that of the expert system (2).

**Definition 3.** If there exists a sufficiently small constant  $\sigma > 0$  such that a matrix or vector  $X$  satisfies

$$\|X - X^*\| \leq \sigma \quad (8)$$

then  $X$  is called an approximate solution of  $X^*$ .

### 3. Model-Based IRL Algorithm

To address the aforementioned IRL problem, this section proposes a model-based method. A model-based IRL algorithm is firstly formulated to reconstruct the output weight matrix that equates to the target cost function (3). Furthermore, the algorithm is analysed to serve as the theoretical foundation for subsequent model-free algorithm.

#### 3.1. Optimal Control with Output Feedback and Cost Weighting

For the learner system (1), consider the following infinite-time cost function

$$V(x(k)) = \sum_{j=k}^{\infty} \left( y(j)^T Q y(j) + u(j)^T R u(j) \right) \quad (9)$$

where  $Q = Q^T \in \mathbb{R}^{q \times q} > 0$  denotes the output weight matrix, and  $R = R^T \in \mathbb{R}^{m \times m} > 0$  denotes the input weight matrix. In contrast to expert system (2), the objective of a learner system (1) is to solve the following cost function within the framework of optimal control.

$$V^*(x(k)) = \min_{u(k)} \sum_{j=k}^{\infty} \left( y(j)^T Q y(j) + u(j)^T R u(j) \right). \quad (10)$$

When the learner system (1) operates under the optimal control policy, its control input can be expressed as

$$u(k) = -Kx(k). \quad (11)$$

In particular, the control gain  $K$  satisfies

$$K = (R + B^T P B)^{-1} B^T P A \quad (12)$$

where  $P = P^T \in \mathbb{R}^{n \times n} > 0$  satisfies the following DT ARE:

$$P = A^T P A + C^T Q C - A^T P B (R + B^T P B)^{-1} B^T P A. \quad (13)$$

Furthermore, substituting (12) into (13) yields

$$(A - BK)^T P (A - BK) + K^T R K + C^T Q C - P = 0. \quad (14)$$

#### 3.2. Design of Model-Based IRL Algorithms

The objective is to construct a model-based IRL algorithm, given  $(A, B, C)$  and  $K_T$ , for reconstructing the output weight matrix that minimizes the target cost function (3) of an expert system (2). Before we proceed with the algorithm design, the following lemma is stated.

**Lemma 1.** Let  $R > 0$  for the learner system (1). If matrices  $Q$  and  $P$  both satisfy (14) and

$$(A - BK)^T P (A - BK) + K^T R K + C^T Q C - P - (K - K_T)^T R (K - K_T) = 0, \quad (15)$$

then the control gain  $K$  obtained is equal to the target control gain  $K_T$ .

**Proof.** Since  $Q$  and  $P$  satisfy both (14) and (15), subtracting (14) from (15) yields

$$(K - K_T)^T R (K - K_T) = 0, \quad (16)$$

then

$$K = K_T. \quad (17)$$

The proof is completed.  $\square$

Based on Lemma 1, a model-based IRL algorithm is proposed. Given an arbitrary symmetric positive definite matrix  $R$ , a learning rate  $\alpha \in (0, 1]$ , an initial output weight matrix  $Q_0$ , and an initial control gain  $K_0$ , let the iteration number be  $i = 0, 1, 2, \dots$ . Before designing the algorithm, we present the steps as follows.

**(1) Policy evaluation  $P_i$ .**

As the algorithm iterates, Equation (14) is rewritten as

$$(A - BK_i)^T P_i (A - BK_i) + K_i^T R K_i + C^T Q_i C - P_i = 0. \quad (18)$$

**(2) Output weight adjustment  $Q_{i+1}$ .**

Once  $P_i$  has been obtained, a correction term corresponding to inverse optimal control [28] is introduced, and the output weight matrix  $Q_{i+1}$  is updated using Equation (14). Let

$$Q_i = C^T Q_i C = P_i - (A - BK_i)^T P_i (A - BK_i) - K_i^T R K_i + (K_i - K_T)^T R (K_i - K_T). \quad (19)$$

Correcting the coupling term between the current  $Q_i$  in (19) and the  $Q_i$  in (15) to  $Q_{i+1}$  gives

$$C^T Q_{i+1} C = (1 - \alpha) C^T Q_i C + \alpha Q_i. \quad (20)$$

The pair  $(A, B)$  is controllable; the pair  $(A, C)$  is observable; and matrix  $C$  has full row rank. Substitute (15) and (19) into (20), then multiply  $(CC^T)^{-1}CC$  and  $C^T(CC^T)^{-1}$  by the left-hand and right-hand sides, respectively, to obtain

$$\begin{aligned} Q_{i+1} &= Q_i - \alpha (CC^T)^{-1} C (C^T Q_i C - \bar{Q}_i) C^T (CC^T)^{-1} \\ &= Q_i + \alpha (CC^T)^{-1} C (K_i - K_T)^T R (K_i - K_T) C^T (CC^T)^{-1}. \end{aligned} \quad (21)$$

**(3) Control policy update  $K_{i+1}$ .**

Evaluate  $P_i$  using the current policy, and update the control policy  $K_{i+1}$

$$K_{i+1} = (R + B^T P_i B)^{-1} B^T P_i A. \quad (22)$$

Then set  $i = i + 1$  and repeat (18)–(22) until the convergence condition  $\|K_{i+1} - K_i\| \leq \sigma$  is satisfied, where  $\sigma > 0$  is a predefined sufficiently small tolerance. The procedure of the model-based IRL algorithm is given in Algorithm 1.

---

**Algorithm 1** Model-Based IRL Algorithm

---

- 1: **Initialization.** Given  $R > 0$  and  $\alpha \in (0, 1]$ , initialize the algorithm with a reward weighting matrix  $Q_0 \geq 0$  and a stabilizing gain  $K_0$ , and set  $i = 0$ .
  - 2: Evaluate  $P_i$  as (18).
  - 3: Correct  $Q_{i+1}$  by (21).
  - 4: Update  $K_{i+1}$  by (22).
  - 5: Update  $i = i + 1$ , and repeat Step 1 until the stopping criterion  $\|K_{i+1} - K_i\| \leq \sigma$  is satisfied, where  $\sigma > 0$  is a prescribed tolerance.
-

### 3.3. Theoretical Analysis of Algorithm 1

This subsection presents theoretical results concerning the convergence and stability of the proposed algorithm.

**Theorem 1** (Convergence of Algorithm 1). *Given  $Q_0 \geq 0$ ,  $R > 0$ , and  $\alpha \in (0, 1]$ , if the system  $(A, C)$  is observable, the sequence  $\{Q_i\}$  will converge to some equivalent weight  $Q$  (see Definition 2), and Algorithm 1 will terminate within a finite number of steps under the given threshold condition.*

**Proof.** Following the convergence analysis reported in [4,26,29], one can conclude that Algorithm 1 requires only a finite number of iterations for the update of  $Q_i$  to stop.

Multiply the left-hand side and right-hand side of (21) by  $C^T$  and  $C$ , respectively, to obtain

$$\begin{aligned} C^T Q_{i+1} C &= (1 - \alpha) C^T Q_i C + \alpha (C^T Q_i C + (K_i - K_T)^T R (K_i - K_T)) \\ &= C^T Q_i C + \alpha C^T (CC^T)^{-1} C (K_i - K_T)^T R (K_i - K_T) C^T (CC^T)^{-1} C. \end{aligned} \quad (23)$$

Let

$$\Sigma_i = C^T (CC^T)^{-1} C (K_i - K_T)^T R (K_i - K_T) C^T (CC^T)^{-1} C. \quad (24)$$

Equation (23) can be written as

$$C^T (Q_{i+1} - Q_i) C = \alpha \Sigma_i. \quad (25)$$

Because  $\Sigma_i \geq 0$ , there is

$$Q_{i+1} \geq Q_i \geq \dots \geq Q_1 \geq 0, \quad i = 0, 1, \dots \quad (26)$$

That is,  $\{Q_i\}$  is monotonically increasing. This follows from (25):

$$C^T Q_{i+1} C = \alpha \Sigma_i + \alpha \Sigma_{i-1} + \dots + \alpha \Sigma_0 + C^T Q_0 C \quad (27)$$

From the Riccati equation theory [30],  $P_i$  varies monotonically with  $Q_i$ , and the associated gain  $K_i$  is consequently adjusted. It is also known that the target gain  $K_T$  may correspond to infinitely many equivalent weighting matrices  $Q$ . Hence, we aim to identify a particular equivalent matrix  $Q_e$  such that

$$\|Q_i - Q_e\| \leq \eta \quad (28)$$

and

$$\|K_i - K_T\| \leq \sigma \quad (29)$$

and the algorithm stops.

It follows from (26) that

$$C^T Q C \geq \|C^T Q_{i+1} C\| - \eta_0(\eta, C) = \|\alpha \Sigma_i + \dots + \alpha \Sigma_0 + C^T Q_0 C\| - \eta_0, \quad (30)$$

and it can be further shown that

$$C^T Q C \geq \alpha(i+1)\sigma^2 \lambda_{\min}(R + B^T P B) - \eta_0, \quad (31)$$

Therefore,

$$i \leq \frac{\|C^T Q C\| + \eta_0(\eta, C)}{\alpha \sigma^2 \lambda_{\min} R} - 1. \quad (32)$$

According to Definition 1,  $Q_i$  is monotonically increasing and  $\|Q_i\|$  is bounded. Therefore, Algorithm 1 converges within a finite number of steps, and the obtained control gain approaches  $K_T$ . The proof is completed.  $\square$

**Remark 2.** As discussed after Theorem 1, the equivalent output weight matrix is generally non-unique. Different choices of  $R$  and  $Q_0$  may yield different converged matrices  $Q_\infty$ , while the resulting pair  $(Q_\infty, R)$  generates the same optimal policy  $K_T$ . In practice,  $R = I$  and  $Q_0 = \lambda I$  with a small  $\lambda > 0$  provide a simple default choice for stable and monotonic convergence. In particular,  $Q_0 = 0$  and  $R = I$  can be used as a convenient minimum-norm selection criterion.

**Theorem 2** (Stability of the closed-loop system). *At every iteration step of Algorithm 1, the closed-loop system of the learner system (1) is stable.*

**Proof.** Given an initial  $Q_0$  and  $R > 0$ , the closed-loop stability of the system must be ensured [31] to prove that

$$V_i(x(k+1)) - V_i(x(k)) \leq 0. \quad (33)$$

If we define the Lyapunov function corresponding to the  $i$ th iteration as

$$V_i(x(k)) = x^T(k)P_i x(k), \quad (34)$$

then there is

$$V_i(x(k+1)) - V_i(x(k)) = x^T(k) [(A - BK_i)^T P_i (A - BK_i) - P_i] x(k), \quad (35)$$

where  $P_i > 0$ ; combining Equations (18) and (35) yields

$$V_i(x(k+1)) - V_i(x(k)) = -x^T(k) (K_i^T R K_i + C^T Q_i C) x(k). \quad (36)$$

Since  $R > 0$  and  $Q_i \geq 0$ , it follows that

$$K_i^T R K_i + C^T Q_i C \geq 0, \quad (37)$$

and thus,

$$V_i(x(k+1)) - V_i(x(k)) \leq 0. \quad (38)$$

Consequently, the stability of the learner system (1) is guaranteed throughout the entire iterative process. This completes the proof.  $\square$

## 4. Model-Free Output-Feedback Off-Policy Inverse Q-Learning Algorithm

In practical control systems, uncertainty in the system model limits the effectiveness of model-based methods. Moreover, unobservable states make it difficult to implement traditional control strategies that rely on full-state information. To address these issues, building upon Algorithm 1, we propose a data-driven, output-feedback method that reconstructs the unknown target cost function using only input–output data from the expert system (2).

### 4.1. Establishment of Off-Policy Q-Function Equation

For the  $i$ th iteration, multiply the left side of (18) by  $x_T^T(k)$  and the right side by  $x_T(k)$ , resulting in

$$\begin{aligned} & x_T^T(k) (A - BK_i)^T P_i (A - BK_i) x_T(k) + x_T^T(k) K_i^T R K_i x_T(k) \\ & + x_T^T(k) C^T Q_i C x_T(k) - x_T^T(k) P_i x_T(k) = 0. \end{aligned} \quad (39)$$

Define the expert's state Q-function as

$$\begin{aligned}
 Q_i(x_T(k), P_i) &= x_T^T(k) P_i x_T(k) \\
 &= (Ax_T(k) + BK_i x_T(k))^T P_i (Ax_T(k) + BK_i x_T(k)) + x_T^T(k) C^T Q_i C x_T(k) + x_T^T(k) K_i^T R K_i x_T(k) \\
 &= \begin{bmatrix} x_T(k) \\ K_i x_T(k) \end{bmatrix}^T H_i \begin{bmatrix} x_T(k) \\ K_i x_T(k) \end{bmatrix} \\
 &= x_T^T(k) W_i^T H_i W_i x_T(k).
 \end{aligned} \tag{40}$$

where

$$W_i = \begin{bmatrix} I \\ -K_i \end{bmatrix} \tag{41}$$

and

$$\begin{aligned}
 H_i &= \begin{bmatrix} H_{(xx)i} & H_{(xu)i} \\ H_{(ux)i} & H_{(uu)i} \end{bmatrix} \\
 &= \begin{bmatrix} A^T P_i A + C^T Q_i C & A^T P_i B \\ B^T P_i A & R + B^T P_i B \end{bmatrix};
 \end{aligned} \tag{42}$$

the block matrix is defined as

$$\begin{aligned}
 H_{(xx)i} &= A^T P_i A + C^T Q_i C \in \mathbb{R}^{n \times n}, \\
 H_{(xu)i} &= A^T P_i B \in \mathbb{R}^{n \times m}, \\
 H_{(ux)i} &= B^T P_i A \in \mathbb{R}^{m \times n}, \\
 H_{(uu)i} &= B^T P_i B + R \in \mathbb{R}^{m \times m}.
 \end{aligned} \tag{43}$$

Next, we construct the off-policy equation using the off-policy method [32]. The original expert system (2) is rewritten as

$$x_T(k+1) = A_c x_T(k) + B(u_T(k) + K_i x_T(k)) \tag{44}$$

where  $A_c = A - BK_i$ ,  $u_i = -K_i x_T(k)$  denotes the auxiliary control input for system (2), while  $u_T(k)$  denotes the target control input.

Multiplying the left-hand side and right-hand side of (14) by  $x_T^T(k)$  and  $x_T(k)$ , respectively, and substituting (40) and (44) into it, one obtains

$$\begin{aligned}
 x_T^T(k) W_i^T H_i W_i x_T(k) (x_T(k+1) - B(u_T(k) + K_i x_T(k)))^T P_i (x_T(k+1) - B(u_T(k) + K_i x_T(k))) \\
 = x_T^T(k) C^T Q_i C x_T(k) + x_T^T(k) K_i^T R K_i x_T(k).
 \end{aligned} \tag{45}$$

Substituting  $P_i = W_i^T H_i W_i$  from (40) into (45) and expanding the quadratic form on the left-hand side, one obtains

$$\begin{aligned}
 x_T^T(k) W_i^T H_i W_i x_T(k) - x_T^T(k+1) W_i^T H_i W_i x_T(k+1) \\
 + 2x_T^T(k+1) P_i B(u_T(k) + K_i x_T(k)) \\
 - (u_T(k) + K_i x_T(k))^T B^T P_i B(u_T(k) + K_i x_T(k)) \\
 = x_T^T(k) C^T Q_i C x_T(k) + x_T^T(k) K_i^T R K_i x_T(k).
 \end{aligned} \tag{46}$$

It follows from (40) and (42) that

$$\begin{aligned}
 Q_i(x_T(k), P_i) \\
 = x_T^T(k) H_{(xx)i} x_T(k) + 2x_T^T(k) H_{(xu)i} u_T(k) + u_T^T(k) H_{(uu)i} u_T(k).
 \end{aligned} \tag{47}$$

It follows from Q-functions (40) and (47) that

$$\begin{aligned} & x_T^T(k)H_{(xx)i}x_T(k) + 2x_T^T(k)H_{(xu)i}u_T(k) + u_T^T(k)H_{(uu)i}u_T(k) \\ & = x_T^T(k)C^TQ_iCx_T(k) + u_T^T(k)Ru_T(k) + x_T^T(k+1)P_ix_T(k+1). \end{aligned} \quad (48)$$

Combining this with (41) and (48) yields

$$\begin{aligned} & x_T^T(k)H_{(xx)i}x_T(k) - x_T^T(k+1)H_{(xx)i}x_T(k+1) \\ & + 2x_T^T(k)H_{(xu)i}u_T(k) + 2x_T^T(k+1)H_{(xu)i}K_ix_T(k+1) \\ & - u_T^T(k)H_{(uu)i}u_T(k) - x_T^T(k+1)K_i^TH_{(uu)i}K_ix_T(k+1) \\ & = x_T^T(k)C^TQ_iCx_T(k) + u_T^T(k)Ru_T(k). \end{aligned} \quad (49)$$

Thus the off-policy Q-function equation is obtained. Using the least-squares method and the expert state-input data  $x_T(k)$  and  $u_T(k)$ , Equation (49) yields a unique solution  $(H_{(xx)i}, H_{(xu)i}, H_{(ux)i}, H_{(uu)i})$  without knowledge of the system dynamics.

#### 4.2. Design of a Output-Feedback Off-Policy Inverse Q-Learning Algorithm

We now proceed to design a model-free method based on the aforementioned off-policy Q-function Equation (49). First, using a state reconstruction method, the off-policy Q-function Equation (49) is transformed into an input–output off-policy Q-function equation. We then iteratively update the output-reward weights  $Q_i$  and control gains  $K_i$  based on this input–output Q-function.

As shown in Assumption 3,  $x_T(k)$  is unobservable, whereas  $u_T(k)$  and  $y_T(k)$  are available. By introducing the lemma from Reference [33], the input and output data  $u_T(k)$  and  $y_T(k)$  are used to represent the Q-function without using the state  $x_T(k)$ .

**Lemma 2.** *If Assumption 1 holds, the system state  $x_T(k)$  at time  $k$  can be expressed in terms of the measurable input and output sequences from  $(k - N)$  to  $(k - 1)$ , as shown below:*

$$x_T(k) = M_y\bar{y}_{k-1,k-N} + M_u\bar{u}_{k-1,k-N} \quad (50)$$

where  $N \leq n$  is the upper bound of the observability index of the system, while  $\bar{u}_{k-1,k-N} \in \mathbb{R}^{m \times N}$  and  $\bar{y}_{k-1,k-N} \in \mathbb{R}^{s \times N}$ , respectively, denote the stacked input and output data vectors,

$$\begin{aligned} \bar{u}_{k-1,k-N} &= \begin{bmatrix} u_T^T(k-1) & u_T^T(k-2) & \cdots & u_T^T(k-N) \end{bmatrix}^T, \\ \bar{y}_{k-1,k-N} &= \begin{bmatrix} y_T^T(k-1) & y_T^T(k-2) & \cdots & y_T^T(k-N) \end{bmatrix}^T. \end{aligned} \quad (51)$$

The coupling matrix is defined as

$$\begin{aligned} M_u &= U_N - A^N(V_N^TV_N)^{-1}V_N^TG_N, \\ M_y &= A^N(V_N^TV_N)^{-1}V_N^T \end{aligned} \quad (52)$$

and

$$V_N = \begin{bmatrix} (CA^{N-1})^T & \cdots & (CA)^T & C^T \end{bmatrix}^T \quad (53)$$

$$U_N = \begin{bmatrix} B & AB & \cdots & A^{N-1}B \end{bmatrix} \quad (54)$$

$$G_N = \begin{bmatrix} 0 & CB & CAB & \cdots & CA^{N-2}B \\ 0 & 0 & CB & \cdots & CA^{N-3}B \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & CB \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \quad (55)$$

where  $V_N$  is the observable matrix,  $U_N$  is the control matrix representing the control inputs, and  $G_N$  is the Toeplitz matrix. Therefore, the state  $x_T(k)$  can be expressed in terms of the input–output data as

$$x_T(k) = Mz_T(k) \quad (56)$$

where

$$z_T(k) = [\bar{u}_{k-1,k-N} \quad \bar{y}_{k-1,k-N}]^T, \quad M = [M_u \quad M_y]. \quad (57)$$

By using the input–output data description (56) of the operating state to describe the state feedback Q-function (40), it is possible to obtain the input–output Q-function as follows:

$$Q_i(z_T(k), P_i) = z_T(k)^T M^T W_i^T H_i W_i M z_T(k). \quad (58)$$

The input–output Q-function matrix is further defined as

$$\bar{H}_i = \begin{bmatrix} \bar{H}_{(xx)i} & \bar{H}_{(xu)i} \\ \bar{H}_{(ux)i} & \bar{H}_{(uu)i} \end{bmatrix} \quad (59)$$

where  $\bar{H}_i = \bar{H}_i^T \in \mathbb{R}^{f \times f}$ ,  $f = mN + qN + m$ , and the block matrix is defined as follows:

$$\begin{aligned} \bar{H}_{(xx)i} &= M^T (A^T P_i A + C^T Q_i C) M \in \mathbb{R}^{(m+q)N \times (m+q)N}, \\ \bar{H}_{(xu)i} &= M^T A^T P_i B \in \mathbb{R}^{(m+q)N \times m}, \\ \bar{H}_{(ux)i} &= B^T P_i A M \in \mathbb{R}^{m \times (m+q)N}, \\ \bar{H}_{(uu)i} &= B^T P_i B + R \in \mathbb{R}^{m \times m}. \end{aligned} \quad (60)$$

Taking into account (22) and (59), the optimal control gain  $\bar{K}_i^T$  of the input–output Q function can be expressed as

$$\bar{K}_{i+1} = K_{i+1} M. \quad (61)$$

Here,  $K_{i+1}$  denotes the state feedback gain, while  $\bar{K}_{i+1}$  denotes the equivalent output-feedback gain acting on  $z_T(k)$ ; it follows from the block structure of the state-dependent Q-function matrix  $H_i$  that

$$K_{i+1} = H_{(uu)i}^{-1} H_{(ux)i}. \quad (62)$$

Therefore, there is

$$\bar{K}_{i+1} = K_{i+1} M = H_{(uu)i}^{-1} H_{(ux)i} M = \bar{H}_{(uu)i}^{-1} \bar{H}_{(ux)i}. \quad (63)$$

By introducing probing noise into the auxiliary control input,  $\hat{u}_T(k) = u_T(k) + \epsilon(k)$ , and substituting it into (49), the output-feedback off-policy Q-function equation can be obtained as

$$\begin{aligned} & z_T^T(k) \bar{H}_{(xx)i} z_T(k) - z_T^T(k+1) \bar{H}_{(xx)i} z_T(k+1) \\ & + 2z_T^T(k) \bar{H}_{(xu)i} \hat{u}_T(k) + 2z_T^T(k+1) \bar{H}_{(xu)i} \bar{K}_i z_T(k+1) \\ & + \hat{u}_T^T(k) \bar{H}_{(uu)i} \hat{u}_T(k) - z_T^T(k+1) \bar{K}_i^T \bar{H}_{(uu)i} \bar{K}_i z_T(k+1) \\ & = y_T^T(k) Q_i y_T(k) + \hat{u}_T^T(k) R \hat{u}_T(k) \end{aligned} \quad (64)$$

where  $\epsilon(k)$  represents the probing noise, while  $\bar{K}_i$  denotes the output-feedback control gain corresponding to the input–output Q-function. Without prior information on the system dynamics, the components  $(\bar{H}_{(xx)i}, \bar{H}_{(xu)i}, \bar{H}_{(ux)i}, \bar{H}_{(uu)i})$  are estimated using the least-squares method by solving Equation (64).

Next, as in Algorithm 1 for correcting  $Q_{i+1}$ , multiply both sides of (21) by  $Mz_T(k)$  and substitute  $y_T(k) = Cx_T(k)$ :

$$\begin{aligned} & y_T^T(k)Q_{i+1}y_T(k) \\ &= y_T^T(k)Q_i y_T(k) + \alpha(\hat{u}_T(k) + \bar{K}_i z_T(k) - \epsilon(k))^T R(\hat{u}_T(k) + \bar{K}_i z_T(k) - \epsilon(k)), \end{aligned} \quad (65)$$

where  $\bar{K}_{i+1}$  is updated iteratively according to (63). Finally, repeat the above steps until the convergence condition  $\|K_{i+1} - K_i\| < \sigma$  is satisfied; otherwise, set  $i = i + 1$  and proceed to (64). The above computational procedure is summarized in Algorithm 2.

---

**Algorithm 2** Model-Free Output-Feedback Off-Policy Inverse Q-Learning Algorithm

---

- 1: **Initialization:** Given  $R > 0$ , initialize the algorithm with a reward weighting matrix  $Q_0 \geq 0$  and a stabilizing gain  $K_0$ , and set  $i = 0$ .
  - 2: Evaluate  $\bar{H}_{(xx)i}, \bar{H}_{(xu)i}, \bar{H}_{(ux)i}, \bar{H}_{(uu)i}$  by (65).
  - 3: Improve  $Q_{i+1}$  as (65).
  - 4: Update  $\bar{K}_{i+1}$  by (63).
  - 5: Update  $i = i + 1$ , and repeat Step 1 until the stopping criterion  $\|\bar{K}_{i+1} - \bar{K}_i\| \leq \sigma$  is satisfied, where  $\sigma > 0$  is a prescribed tolerance.
- 

**Remark 3.** Lemma 2 reveals that the system state  $x_T(k)$  can be reconstructed from past input–output data. The parameter  $N$  is the observability index, which represents the minimum number of past output samples needed to uniquely determine the current state.

#### 4.3. Implementation of the Model-Free Algorithm

This subsection shows the implementation of steps (64) and (65) of Algorithm 2. The implementation of this algorithm primarily utilizes the following Kronecker product:

$$a^T W b = (b^T \otimes a^T) \text{vec}(W). \quad (66)$$

Using the Kronecker product in (66), estimate  $\bar{H}_{(xx)i}, \bar{H}_{(xu)i}, \bar{H}_{(ux)i}, \bar{H}_{(uu)i}$  via (64), where (64) is rewritten as

$$\begin{aligned} & \left( z_T^T(k) \otimes z_T^T(k) - z_T^T(k+1) \otimes z_T^T(k+1) \right) \text{vec}(\bar{H}_{(xx)i}) \\ &+ 2 \left( \hat{u}_T^T(k) \otimes z_T^T(k) + \bar{K}_i z_T(k+1) \otimes z_T^T(k+1) \right) \text{vec}(\bar{H}_{(xu)i}) \\ &+ \left( \hat{u}_T^T(k) \otimes \hat{u}_T^T(k) - (\bar{K}_i z_T(k+1))^T \otimes (\bar{K}_i z_T(k+1))^T \right) \text{vec}(\bar{H}_{(uu)i}) \\ &= y_T^T(k)Q_i y_T(k) + \hat{u}_T^T(k)R\hat{u}_T(k). \end{aligned} \quad (67)$$

Using a set of sample data indexed by  $l \in \mathbb{Z}^+$ ,

$$\phi_i = [\phi_i^1 \ \phi_i^2 \ \cdots \ \phi_i^l]^T \quad (68)$$

$$\psi_i = \begin{bmatrix} Z_{(xx)1} & Z_{(xu)1} & Z_{(uu)1} \\ Z_{(xx)2} & Z_{(xu)2} & Z_{(uu)2} \\ \vdots & \vdots & \vdots \\ Z_{(xx)l} & Z_{(xu)l} & Z_{(uu)l} \end{bmatrix}. \quad (69)$$

The scalar term corresponding to the  $l$ th sample is defined as

$$\phi_i^l = y_T^T(k+l-1)Q_i y_T(k+l-1) + \hat{u}_T^T(k+l-1)R\hat{u}_T(k+l-1), \quad (70)$$

$$\begin{aligned} Z_{(xx)l} &= z_T^T(k+l-1) \otimes z_T^T(k+l-1) - z_T^T(k+l) \otimes z_T^T(k+l), \\ Z_{(xu)l} &= 2\left(\hat{u}_T^T(k+l-1) \otimes z_T^T(k+l-1) + (\bar{K}_i z_T(k+l))^T \otimes z_T^T(k+l)\right), \\ Z_{(uu)l} &= \hat{u}_T^T(k+l-1) \otimes \hat{u}_T^T(k+l-1) - (\bar{K}_i z_T(k+l))^T \otimes (\bar{K}_i z_T(k+l))^T. \end{aligned} \quad (71)$$

Then, considering (68)–(71), (67) can be rewritten as

$$\psi_i \left[ \text{vec}(\bar{H}_{(xx)i}) \text{vec}(\bar{H}_{(xu)i}) \text{vec}(\bar{H}_{(uu)i}) \right]^T = \phi_i. \quad (72)$$

For (72), using the batch least-squares (BLS) method, one can determine from (68) that

$$\left[ \text{vec}(\bar{H}_{(xx)i}) \text{vec}(\bar{H}_{(xu)i}) \text{vec}(\bar{H}_{(uu)i}) \right]^T = (\psi_i^T \psi_i)^{-1} \psi_i^T \phi_i \quad (73)$$

where the total number of unknown parameters corresponding to  $\text{vec}(\bar{H}_{(xx)i})$ ,  $\text{vec}(\bar{H}_{(xu)i})$ , and  $\text{vec}(\bar{H}_{(uu)i})$  is required to satisfy  $l \geq ((m+s)N)^2 + (m+s)Nm + m^2$ . Therefore, in order to guarantee a unique solution to (73), the persistence of excitation condition must be satisfied, and at least  $l \geq ((m+s)N)^2 + (m+s)Nm + m^2$  sampling steps are required. Similar to (66), (65) can be rewritten as

$$\begin{aligned} & (y_T^T(k) \otimes y_T^T(k))^T \text{vec}(Q_{i+1}) \\ &= y_T^T(k)Q_i y_T(k) + \alpha \left( \hat{u}_T(k) + \bar{K}_i z_T(k) - \epsilon(k) \right)^T R \left( \hat{u}_T(k) + \bar{K}_i z_T(k) - \epsilon(k) \right). \end{aligned} \quad (74)$$

To further utilize multiple groups of sampled data to perform batch least-squares estimation of  $Q_{i+1}$ , the right-hand side of (74) corresponding to the  $p$ th sample group is denoted by  $\xi_p$ . By stacking the regression vectors corresponding to  $p$  consecutive groups of sampled data, one obtains the batch data matrix, denoted by  $\eta_p$ , and then one has

$$\begin{aligned} \xi_p &= \alpha \left( \hat{u}_T(k+p-1) + \bar{K}_i z_T(k+p-1) - \epsilon(k+p-1) \right)^T \\ &\quad \times R \left( \hat{u}_T(k+p-1) + \bar{K}_i z_T(k+p-1) - \epsilon(k+p-1) \right) \\ &\quad + y_T^T(k+p-1)Q_i y_T(k+p-1) \end{aligned} \quad (75)$$

and

$$\eta_p = \begin{bmatrix} y_T^T(k) \otimes y_T^T(k) \\ y_T^T(k+1) \otimes y_T^T(k+1) \\ \vdots \\ y_T^T(k+p-1) \otimes y_T^T(k+p-1) \end{bmatrix}. \quad (76)$$

where  $p \in \mathbb{Z}^+$  denotes the number of sample data groups.

After multiple sets of sample data are stacked, a batch of linear equations relating to the parameters to be estimated can be formulated and solved using the BLS method; rewriting (74) yields

$$\text{vec}(Q_{i+1}) = (\eta_p^T \eta_p)^{-1} \eta_p^T \xi_p. \quad (77)$$

In particular,  $Q_{i+1}$  is a symmetric positive definite matrix with  $\frac{q(q+1)}{2}$  distinct elements, and at least  $p \geq \frac{q(q+1)}{2}$  samples are required to solve (74).

After multiple sets of sample data are stacked, a system of linear equations involving  $\text{vec}(Q_{i+1})$  can be formulated, and the latest output weight matrix  $Q_{i+1}$  can be obtained using a least-squares approach.

**Remark 4.** In Algorithm 2, the dimension of the regressor  $z_T(k)$  is  $(m + q)N$ , and so the number of unknowns in the input–output  $Q$ -function matrix  $\hat{H}_i$  grows quadratically with  $N$ . The per-iteration complexity of the batch least-squares solution is  $O(((m + q)N + m)^3)$ , since the matrix to be inverted has dimension  $d = ((m + q)N + m)((m + q)N + m + 1)/2$ . Therefore, choosing  $N$  as the observability index balances reconstruction accuracy and computational efficiency.

**Remark 5.** It should be noted that, in order to satisfy the persistence of excitation condition for the BLS problem [32], the auxiliary control input is injected with probing noise in Algorithm 2. This ensures sufficient excitation of the regression data, such that the parameter matrices can be uniquely identified. The unbiasedness of the algorithm after the introduction of probing noise will be rigorously analysed in the subsequent subsection.

#### 4.4. Theoretical Analysis of Algorithm 2

In this subsection, some theoretical results are presented for the proposed algorithm, primarily concerning its unbiasedness and convergence.

**Theorem 3** (Unbiasedness of Algorithm 2). In Algorithm 2, by injecting probing noise into the control input  $\hat{u}_T(k)$ , the solutions of  $(\hat{H}_{(xx)i}, \hat{H}_{(xu)i}, \hat{H}_{(uu)i})$  obtained from (64) and  $Q_{i+1}$  obtained from (65) are unbiased,  $\hat{H}_{(xx)i} = \bar{H}_{(xx)i}$ ,  $\hat{H}_{(xu)i} = \bar{H}_{(xu)i}$ ,  $\hat{H}_{(uu)i} = \bar{H}_{(uu)i}$ ,  $\hat{Q}_i = Q_i$ .

**Proof.** When the control input is injected with probing noise, we have

$$\hat{u}_T(k) = u_T(k) + \epsilon(k). \quad (78)$$

According to Lemma 2, Equation (49) is equivalent to (64) and is derived from (44). Therefore, we revisit (44) to analyse the unbiasedness of Algorithm 2. Assume that the control input  $\hat{u}_T(k)$  with injected probing noise leads to biased estimates  $\hat{H}_i$  and  $\hat{Q}_i$  from (64) and (65), respectively. By substituting  $\hat{u}_T(k) = u_T(k) + \epsilon(k)$  and the biased estimates  $\hat{H}_i$  into (44), one obtains

$$\begin{aligned} & x_T^T(k)W_i^T\hat{H}_iW_ix_T(k) - x_T^T(k+1)W_i^T\hat{H}_iW_ix_T(k+1) \\ & + 2x_T^T(k+1)\hat{P}_iB(\hat{u}_T(k) + K_ix_T(k) - \epsilon(k)) \\ & - (B(u_T(k) + K_ix_T(k) - \epsilon(k)))^T\hat{P}_i(B(u_T(k) + K_ix_T(k) - \epsilon(k))) \\ & = x_T^T(k)C^TQ_iCx_T(k) + x_T^T(k)K_i^TRK_ix_T(k). \end{aligned} \quad (79)$$

If one substitutes  $P_i = W_i^TH_iW_i$  into the terms involving  $\epsilon(k)$  and cancelling the common terms, it follows that

$$\begin{aligned} & x_T^T(k)W_i^T\hat{H}_iW_ix_T(k) - x_T^T(k+1)W_i^T\hat{H}_iW_ix_T(k+1) \\ & + 2x_T^T(k+1)\hat{P}_iB(u_T(k) + K_ix_T(k)) \\ & - (B(u_T(k) + K_ix_T(k)))^T\hat{P}_i(B(u_T(k) + K_ix_T(k))) \\ & = x_T^T(k)C^TQ_iCx_T(k) + x_T^T(k)K_i^TRK_ix_T(k). \end{aligned} \quad (80)$$

By comparing the state feedback (44) without probing noise and the state feedback (71) with probing noise, it can conclude that  $\hat{H}_i = H_i$ . Furthermore, by combining this with the

one-to-one correspondence between the input–output form and the state feedback form, it follows that

$$\hat{H}_{(xx)i} = \bar{H}_{(xx)i}, \quad \hat{H}_{(xu)i} = \bar{H}_{(xu)i}, \quad \hat{H}_{(uu)i} = \bar{H}_{(uu)i}.$$

This implies that, after probing noise is introduced, (65) does not introduce bias into  $Q_i(z_T(k), P_i)$ .

$$\begin{aligned} & y_T^T(k) \hat{Q}_{i+1} y_T(k) \\ &= y_T^T(k) \hat{Q}_i y_T(k) + \alpha(u_T(k) + \bar{K}_i z_T(k))^T R(u_T(k) + \bar{K}_i z_T(k)). \end{aligned} \quad (81)$$

Thus, it follows that  $\hat{Q}_{i+1} = Q_{i+1}$ . Then, Algorithm 2 is unbiased with respect to the probing noise. This concludes the proof.  $\square$

**Theorem 4** (Convergence of Algorithm 2). *Given an initial matrix  $Q_0 > 0$ , and  $(A, \sqrt{Q_0}C)$  is observable, the sequences  $\{Q_i\}$ ,  $\{K_i\}$ , and  $\{H_i\}$  generated by Algorithm 2 converge to the limit matrices  $Q_\infty$ ,  $K_\infty$ , and  $H_\infty$ , respectively.*

**Proof.** The core of this proof is to establish the mathematical equivalence between Algorithm 2 and the model-based Algorithm 1, whose convergence has been proven in Theorem 1. We demonstrate that every computational step in Algorithm 2 is an algebraically identical transformation of the corresponding step in Algorithm 1, performed solely using input–output data.

The proof proceeds by analysing the three main steps of both algorithms.

#### Step 1: Equivalence of the Policy Evaluation Steps.

In Algorithm 2, the policy evaluation step solves for the input–output Q-function blocks  $(\bar{H}_{(xx)i}, \bar{H}_{(xu)i}, \bar{H}_{(uu)i})$  using (64). By Lemma 2,  $x_T(k) = Mz_T(k)$ , and from (59) and (60),  $\bar{H}_i = M^T H_i M$ . Substituting these into (64) transforms it back into the state-feedback off-policy Equation (49). Reversing the off-policy transformation from (44) to (49) yields the state Q-function form:

$$Q_i(x_T(k), P_i) = \begin{bmatrix} x_T(k) \\ -K_i x_T(k) \end{bmatrix}^T H_i \begin{bmatrix} x_T(k) \\ -K_i x_T(k) \end{bmatrix}. \quad (82)$$

By using  $P_i = W_i^T H_i W_i$  from (40) with  $W_i = [I \quad -K_i]^T$  and substituting the block structure of  $H_i$  from (42), we obtain exactly the Lyapunov Equation (18). Therefore, solving (64) in Algorithm 2 is equivalent to solving (18) in Algorithm 1.

#### Step 2: Equivalence of the Policy Improvement (Weight Update) Steps.

In Algorithm 2, the output weight update is given by (65). Using  $\hat{u}_T(k) = u_T(k) + \epsilon(k)$ ,  $u_T(k) = -K_T x_T(k)$ ,  $x_T(k) = Mz_T(k)$ ,  $\bar{K}_i = K_i M$  from (61), and  $y_T(k) = Cx_T(k)$ , the correction term becomes  $\hat{u}_T(k) + \bar{K}_i z_T(k) - \epsilon(k) = (K_i - K_T)x_T(k)$ . Substituting this into (65) yields

$$x_T^T(k) C^T Q_{i+1} C x_T(k) = x_T^T(k) C^T Q_i C x_T(k) + \alpha x_T^T(k) (K_i - K_T)^T R (K_i - K_T) x_T(k). \quad (83)$$

Since  $x_T(k)$  is arbitrary, we obtain

$$C^T Q_{i+1} C = C^T Q_i C + \alpha (K_i - K_T)^T R (K_i - K_T). \quad (84)$$

Pre-multiplying by  $(CC^T)^{-1}C$  and post-multiplying by  $C^T(CC^T)^{-1}$  yields exactly the model-based update (21). Thus, (65) in Algorithm 2 is equivalent to (21) in Algorithm 1.

#### Step 3: Equivalence of the Control Policy Updates.

In Algorithm 2, the control gain is updated using (63):  $\bar{K}_{i+1} = \bar{H}_{(uu)i}^{-1} \bar{H}_{(ux)i}$ . From (60),  $\bar{H}_{(uu)i} = B^T P_i B + R$  and  $\bar{H}_{(ux)i} = B^T P_i A M$ . Substituting these into (63) yields

$$\bar{K}_{i+1} = (R + B^T P_i B)^{-1} B^T P_i A M. \quad (85)$$

Given  $\bar{K}_{i+1} = K_{i+1} M$  from (61) and given that  $M$  is full column rank, cancelling  $M$  on both sides yields exactly the model-based gain update (22).

We have demonstrated that Step 1, Step 2, and Step 3 of Algorithm 2 are mathematically equivalent to Step 1, Step 2, and Step 3 of Algorithm 1, respectively. Therefore, Algorithm 2 is structurally equivalent to Algorithm 1. Furthermore, Theorem 3 has established that the probing noise injected in Algorithm 2 does not introduce any bias. Since Theorem 1 guarantees the convergence of Algorithm 1, it follows directly that Algorithm 2 also converges to  $(Q_\infty, \bar{K}_\infty, \bar{H}_\infty)$ .  $\square$

**Remark 6.** It should be pointed out that the output weight matrix  $Q_\infty$  obtained by Algorithm 2 does not necessarily coincide with the target weight matrix  $Q_T$ . However, the resulting output-feedback control gain  $K_\infty$  can approximate the target expert control gain  $K_T$ .

This indicates that the solution to the IRL problem is generally non-unique; different cost functions may correspond to the same or similar optimal control policies.

## 5. Simulation

To verify the effectiveness of the proposed algorithms, the F-16 aircraft autopilot function is adopted as an example, and the simulation results of both algorithms are presented. Regarding the F-16 aircraft autopilot model in [31,32,34], the system has three states  $x = [\alpha, q, \delta_e]^T$ , where  $\alpha$  denotes the angle of attack,  $q$  denotes the pitch rate, and  $\delta_e$  denotes the elevator deflection angle. The dynamics of the learner system (1) and the expert system (2) are as follows:

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

$$B = \begin{bmatrix} -0.00150808 \\ -0.0096 \\ 0.867345 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0.1 & -0.5 \end{bmatrix}.$$

The target output weight and input weight of the expert system are set as  $Q_T = 1$  and  $R_T = 1$ , respectively. Under the expert control policy, the optimal feedback gain is obtained as

$$K_T = \begin{bmatrix} -0.4293 & -0.1221 & 0.02492 \end{bmatrix},$$

and the corresponding state weighting matrix is given by

$$C^T Q_T C = \begin{bmatrix} 1 & 0.1 & -0.5 \\ 0.1 & 0.01 & -0.05 \\ -0.5 & -0.05 & 0.25 \end{bmatrix}.$$

The initial output weight of the learner system is set as  $Q_0 = 0$ , the input weight is  $R = 1$ , and the initial state is given by

$$x(0) = x_T(0) = \begin{bmatrix} 10 & 10 & -3 \end{bmatrix}^T.$$

### 5.1. Simulation Results for Algorithm 1

This subsection presents the results of Algorithm 1. Algorithm 1 is implemented under the condition that the system dynamics matrices  $A$ ,  $B$ , and  $C$  are known, and the expert optimal feedback gain  $K_T$  is available. Figure 2 illustrates the convergence performance of  $P_i$ ,  $Q_i$ , and  $K_i$  in Algorithm 1. It can be observed from Figure 3 that, as the number of iterations increases, the performance matrices of the learner system gradually converge. After the algorithm converges, the learned output weight is obtained as

$$Q_\infty = 0.9142.$$

The control gain and state weighting matrix are given by

$$K_\infty = \begin{bmatrix} -0.4001 & -0.1147 & 0.0231 \end{bmatrix},$$

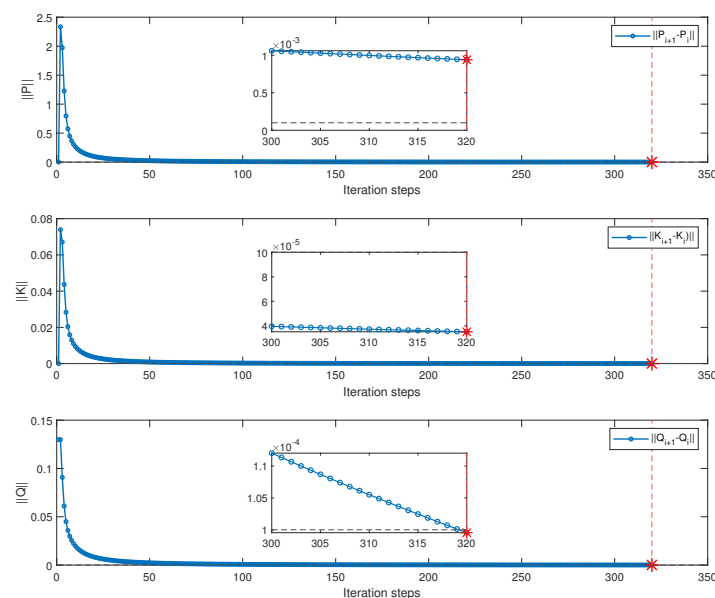
$$C^T Q_\infty C = \begin{bmatrix} 0.9649 & 0.09665 & -0.4824 \\ 0.0965 & 0.0096 & -0.0482 \\ -0.4824 & -0.0482 & 0.2412 \end{bmatrix}.$$

Specifically, the  $P$  matrix of the expert system and the  $P_T$  matrix ultimately learned by the learner are

$$P = \begin{bmatrix} 8.5491 & 5.5134 & -0.5444 \\ 5.5134 & 4.5822 & -0.0645 \\ -0.5444 & -0.0645 & 0.2449 \end{bmatrix}$$

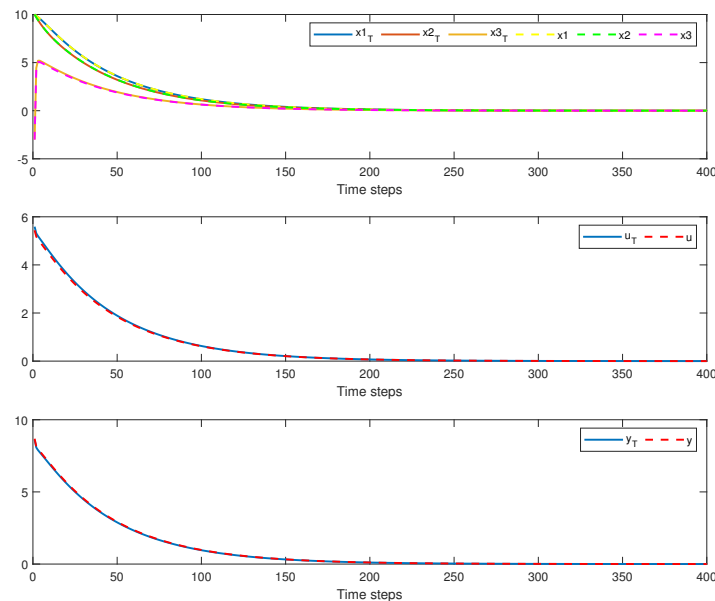
$$P_T = \begin{bmatrix} 8.7815 & 5.6536 & -0.5637 \\ 5.6536 & 4.6970 & -0.0667 \\ -0.5637 & -0.0667 & 0.2538 \end{bmatrix}$$

Figure 2 shows a trajectory comparison between the expert system (2) and the learner system (1) under the same initial condition. Here,  $x_T(k)$ ,  $u_T(k)$ , and  $y_T(k)$  denote the state trajectory, control input trajectory, and output trajectory of the expert system (2), respectively, while  $x(k)$ ,  $u(k)$ , and  $y(k)$  denote those of the learner system (1).



**Figure 2.** Convergence process of  $P_i$ ,  $K_i$  and  $Q_i$  using Algorithm 1.

It can be seen that, after convergence, the trajectories of the learner system (1) closely match those of the expert system (2), which verifies the effectiveness of Algorithm 1.



**Figure 3.** The learner trajectory  $(x(k), u(k), y(k))$  and expert trajectory  $(x_T(k), u_T(k), y_T(k))$  for Algorithm 1.

### 5.2. Simulation Results for Algorithm 2

This subsection reports the results of Algorithm 2. In contrast to Algorithm 1, Algorithm 2 operates without access to the system dynamics or state information. In the simulation, the input–output data vector  $z_T(k)$  is constructed, and the state-feedback Q-function equation is transformed into the input–output form using the reconstruction relation  $x_T(k) = Mz_T(k)$ .

For reproducibility, the initial output-weight matrix and the input-weight matrix are set to  $Q_0 = 0$  and  $R = 1$ , respectively. The batch least-squares estimation uses 43 consecutive data samples, and the algorithm is terminated when  $\|\bar{K}_{i+1} - \bar{K}_i\|_F \leq 10^{-4}$ . The common initial condition for the expert and learner systems is

$$x(0) = x_T(0) = \begin{bmatrix} 10 & 10 & -3 \end{bmatrix}^T.$$

The probing signals used for the sensitivity analysis are specified separately below, while all other simulation settings remain unchanged.

The convergence performance of  $Q_i$  and  $\bar{K}_i$  in Algorithm 2 is shown in Figure 4. The converged values of  $Q_\infty$  and  $\bar{K}_\infty$  are given by

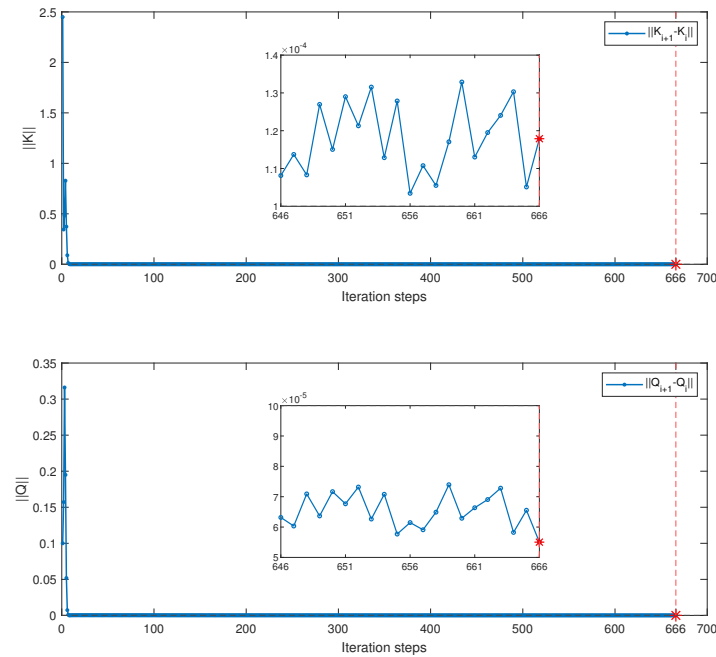
$$\bar{K}_\infty = \begin{bmatrix} 0.0217, -0.7160, 0.4800, -1.6526, -1.3257, -0.1469 \end{bmatrix}$$

$$Q_\infty = 0.9142.$$

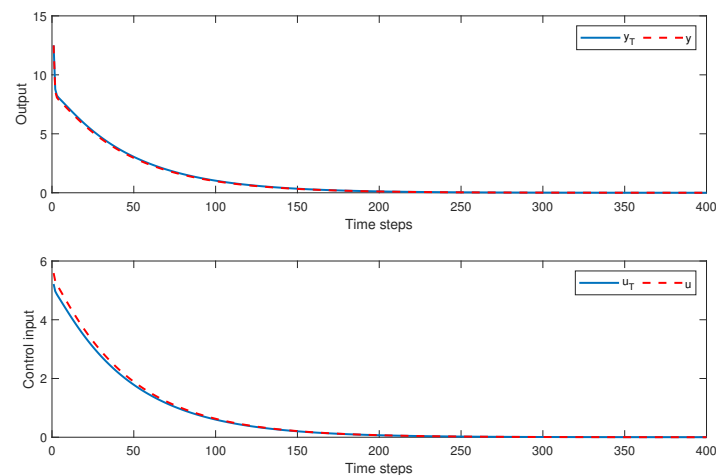
Using the state-reconstruction relation, the corresponding state-feedback control gain is obtained as

$$K_\infty = \begin{bmatrix} -0.4001, -0.1147, 0.0231 \end{bmatrix}.$$

Similarly, Figure 5 presents the trajectory comparison between the expert system (2) and the learner system (1) under Algorithm 2. Under the same initial condition, the trajectories of the learner system (1) can closely approximate those of the expert system (2). This indicates that Algorithm 2 can effectively reconstruct the performance index of the expert system without relying on a system model or state information and can learn a control policy close to that of the expert system (2).



**Figure 4.** Convergence process of  $K_i$  and  $Q_i$  using Algorithm 2.



**Figure 5.** The learner trajectory  $(u(k), y(k))$  and expert trajectory  $(u_T(k), y_T(k))$  for Algorithm 2.

$$\bar{H}_{xx} = \begin{bmatrix} 0.1781 & -0.6948 & 0.4310 & -1.6507 & 1.2069 & -0.1319 \\ -0.6932 & 134.7494 & -108.5490 & 309.7877 & -291.5275 & 33.2234 \\ 0.4325 & -108.5505 & 87.6898 & -249.5126 & 235.4112 & -26.8387 \\ -1.6498 & 309.7880 & -249.5147 & 712.2173 & -670.1239 & 76.3672 \\ 1.2089 & -291.5273 & 235.4117 & -670.1239 & 632.0192 & -72.0510 \\ -0.1324 & 33.2229 & -26.8387 & 76.3672 & -72.0510 & 8.2144 \end{bmatrix}, \quad (86)$$

$$\bar{H}_{xu} = \bar{H}_{ux}^T = \begin{bmatrix} 0.0256 & -0.8431 & 0.5651 & -1.9460 & 1.5610 & -0.1730 \end{bmatrix}^T, \bar{H}_{uu} = \begin{bmatrix} 1.1775 \end{bmatrix}$$

The Q-function matrix partition  $(\bar{H}_{xx}, \bar{H}_{xu}, \bar{H}_{ux}, \bar{H}_{uu})$  is given by (86).

Furthermore, Figure 6 shows the convergence process of the block matrices of the input–output Q-function matrix in Algorithm 2. Specifically, the convergence errors of  $\bar{H}_{(xx)i}$ ,  $\bar{H}_{(xu)i}$ ,  $\bar{H}_{(ux)i}$ , and  $\bar{H}_{(uu)i}$  gradually decrease and converge to a small neighbourhood of zero as the number of iterations increases. This demonstrates that the iteration process can effectively identify the Q-function matrix from the collected input–output data. Therefore, the convergence of the Q-function matrix further verifies the effec-

tiveness and feasibility of the proposed model-free output-feedback off-policy inverse Q-learning algorithm.

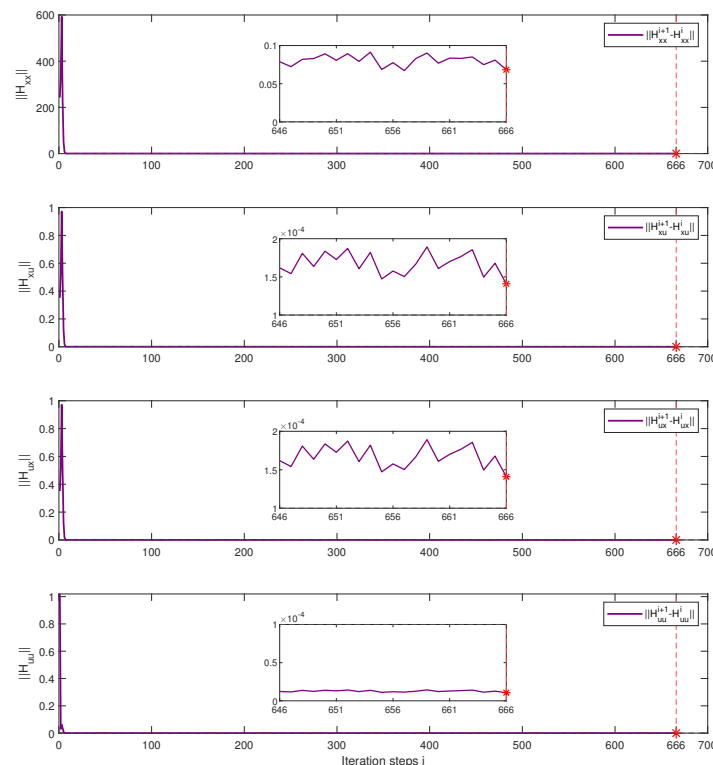
To investigate the influence of the probing-noise amplitude on the parameter estimation, three bounded multisine probing signals are considered:

$$\begin{aligned}\epsilon_1(k) &= 0.1 \sin(1.009k) + \cos^2(0.538k) + \sin(0.9k) + \cos(100k), \\ \epsilon_2(k) &= 0.2 \sin(1.009k) + \cos^2(0.538k) + \sin(0.9k) + \cos(100k), \\ \epsilon_3(k) &= 0.3 \sin(1.009k) + \cos^2(0.538k) + \sin(0.9k) + \cos(100k).\end{aligned}\quad (87)$$

The results in Table 1 show that Algorithm 2 converges under all three probing signals. The learned control gains remain close to the expert gain  $K_T$ , with gain errors of 0.0294, 0.0303, and 0.0304, respectively. The iteration steps further indicate the convergence efficiency of Algorithm 2 at different probing-noise amplitudes.

**Table 1.** Learned control gains at different probing-noise amplitudes.

| Probing Signal  | Learned Control Gain $K_\infty$ | $\ K_\infty - K_T\ _F$ | Iteration Steps |
|-----------------|---------------------------------|------------------------|-----------------|
| $\epsilon_1(k)$ | $[-0.4009, -0.1149, 0.0232]$    | 0.0294                 | 666             |
| $\epsilon_2(k)$ | $[-0.4001, -0.1147, 0.0231]$    | 0.0303                 | 666             |
| $\epsilon_3(k)$ | $[-0.3999, -0.1147, 0.0231]$    | 0.0304                 | 666             |



**Figure 6.** Convergence process of  $\hat{H}_{(xx)i}$ ,  $\hat{H}_{(xu)i}$ ,  $\hat{H}_{(ux)i}$ , and  $\hat{H}_{(uu)i}$  using Algorithm 2.

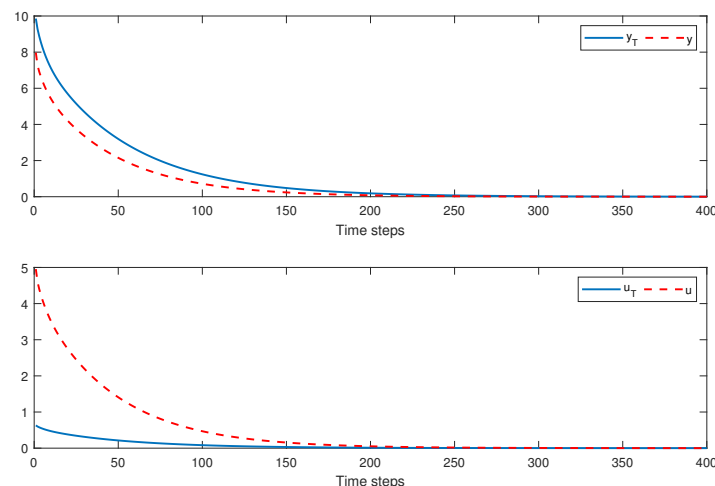
### 5.3. Comparison with the Output-Feedback RL-Based Optimal Control Method

In this subsection, the F-16 aircraft system is again used as an example for algorithm comparison, where Algorithm 2 presented in this paper is compared with the DT optimal control method based on output-feedback Q-learning proposed in [35]. In that work [35], the weight matrix for the cost function (3) must be specified manually in advance. Consequently, it can only address the problem of optimal reinforcement learning control under a known cost function.

Figures 5 and 7 show the comparison results between the RL-based optimal control method and our method in terms of output trajectory imitation performance. It can be observed that, using our method, the learner demonstrates superior imitation performance and is able to learn the optimal output-feedback control law without relying on system models or state information. In contrast, manually assigned cost function weights require constant adjustment, which may result in system performance failing to converge to the optimal control law as rapidly. Through imitation learning, satisfactory imitation performance can be guaranteed.

To quantitatively evaluate the trajectory imitation performance, the integral absolute error (IAE) and mean square error (MSE) are calculated for both the output and control-input trajectories. Quantitative comparison of trajectory imitation performance is shown in Table 2. They are defined as

$$\begin{aligned} \text{IAE}_y &= \sum_{k=1}^N |y_T(k) - y(k)|, & \text{MSE}_y &= \frac{1}{N} \sum_{k=1}^N (y_T(k) - y(k))^2, \\ \text{IAE}_u &= \sum_{k=1}^N |u_T(k) - u(k)|, & \text{MSE}_u &= \frac{1}{N} \sum_{k=1}^N (u_T(k) - u(k))^2. \end{aligned} \quad (88)$$



**Figure 7.** Simulations of the trajectories  $u$  and  $y$  were performed using an optimal control method based on RL.

**Table 2.** Quantitative comparison of trajectory imitation performance.

| Method                         | Signal | IAE      | MSE    |
|--------------------------------|--------|----------|--------|
| Output-feedback RL method [35] | $y$    | 141.7105 | 0.3605 |
|                                | $u$    | 164.7927 | 0.7949 |
| Proposed Algorithm 2           | $y$    | 12.2063  | 0.0029 |
|                                | $u$    | 14.1940  | 0.0066 |

## 6. Conclusions

This paper investigated the output-feedback inverse Q-learning problem for DT linear systems with unobservable states and proposed a data-driven output-feedback off-policy inverse Q-learning algorithm. First, a model-based IRL algorithm was presented as the basic framework. Building on this framework, a model-free output-feedback IRL algorithm was developed using only expert input–output trajectory data and without requiring a system model. Theoretical analysis demonstrated that the proposed algorithm achieves convergence and closed-loop stability under appropriate conditions. Finally, the effective-

ness of the proposed algorithm was verified through simulation examples, with results indicating that the learner can closely approximate the cost function structure and control performance of the expert system.

Compared with existing IRL methods, the approach has overcome the reliance of traditional state-feedback methods on the measurability of system states, enabling inverse Q-learning to be applied in output-feedback scenarios where states cannot be directly measured. Future work may further consider extending the proposed method to more general nonlinear systems and multi-agent game scenarios, conducting experimental validation on actual control objects, and combining robust data-estimation and  $H_\infty$  control techniques with the proposed framework to improve its robustness against measurement noise and model uncertainties.

**Author Contributions:** Z.L.: funding acquisition and supervision. L.Z.: methodology and writing (original draft). X.D.: software and validation. H.F.: data curation and visualization. X.L.: writing (review and editing). P.L.: investigation and resources. All authors have read and agreed to the published version of the manuscript. The co-authors finish the proofreading

**Funding:** This work was funded by the National Natural Science Foundation of China (Grant No. 62363002), the 2025 Hubei Provincial Key Laboratory Directed Commission Project (Basic Research Platform Program, Grant No. 2025CSA090), the Doctoral Foundation of Guangxi University of Science and Technology (Grant No. Z1009), and the Innovation Project of Guangxi Graduate Education (Grant No. JGY2025299).

**Data Availability Statement:** The raw data supporting the conclusions of this article will be made available by the authors on request.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Nomenclature

The main symbols used throughout this paper are summarized below.

Main symbols used in this paper

| Symbol              | Description                                                            |
|---------------------|------------------------------------------------------------------------|
| $k$                 | Discrete-time index.                                                   |
| $Q, R$              | Output-weight and input-weight matrices for the learner cost function. |
| $Q_T, R_T$          | Weight matrices of the expert cost function.                           |
| $Q_i$               | Estimated output-weight matrix at iteration $i$ .                      |
| $Q_\infty$          | Converged output-weight matrix.                                        |
| $P_i$               | Cost-function matrix at iteration $i$ .                                |
| $K_i$               | State-feedback control gain at iteration $i$ .                         |
| $K_T$               | Expert optimal state-feedback gain.                                    |
| $\bar{K}_i$         | Output-feedback control gain at iteration $i$ .                        |
| $M$                 | State-reconstruction matrix.                                           |
| $z_T(k)$            | Reconstructed input–output data vector at time $k$ .                   |
| $\bar{H}_{(xx)i}$   | State-data quadratic block of the Q-function matrix.                   |
| $\bar{H}_{(xu)i}$   | State-data/input cross-term block.                                     |
| $\bar{H}_{(ux)i}$   | Input/state-data cross-term block.                                     |
| $\bar{H}_{(uu)i}$   | Input quadratic block.                                                 |
| $\epsilon(k)$       | Probing signal injected into the auxiliary control input.              |
| $\alpha$            | Learning rate.                                                         |
| $\sigma$            | Stopping tolerance.                                                    |
| $\ \cdot\ _F$       | Frobenius norm.                                                        |
| $\text{vec}(\cdot)$ | Vectorization operator.                                                |

## References

- Lewis, F.L.; Vrabie, D.; Syrmos, V.L. *Optimal Control*; John Wiley & Sons: Hoboken, NJ, USA, 2012.
- Zhao, D.B.; Shao, K.; Zhu, Y.H.; Li, D.; Chen, Y.R.; Wang, H.T.; Liu, D.R.; Zhou, T.; Wang, C.H. Review of deep reinforcement learning and discussions on the development of computer Go. *Kongzhi Lilun Yu Yingyong/Control Theory Appl.* **2016**, *33*, 701–717.
- Zhang, H.G.; Zhang, X.; Yan-Hong, L.; Jun, Y. An overview of research on adaptive dynamic programming. *Acta Autom. Sin.* **2013**, *39*, 303–311. [[CrossRef](#)]
- Abbeel, P.; Ng, A.Y. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the Twenty-first International Conference on Machine Learning, Banff, AB, Canada, 4–8 July 2004; p. 1.
- Zhifei, S.; Joo, E.M. A review of inverse reinforcement learning theory and recent advances. In *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*; IEEE: Piscataway, NJ, USA, 2012; pp. 1–8.
- Ng, A.Y.; Russell, S. Algorithms for inverse reinforcement learning. In Proceedings of the Icml, Stanford, CA, USA, 29 June–2 July 2000; Volume 1, p. 2.
- Ab Azar, N.; Shahmansoorian, A.; Davoudi, M. From inverse optimal control to inverse reinforcement learning: A historical review. *Annu. Rev. Control* **2020**, *50*, 119–138. [[CrossRef](#)]
- Self, R.; Abudia, M.; Kamalapurkar, R. Online inverse reinforcement learning for systems with disturbances. In *Proceedings of the 2020 American Control Conference (ACC)*; IEEE: Piscataway, NJ, USA, 2020; pp. 1118–1123.
- Xue, W.; Lian, B.; Fan, J.; Chai, T.; Lewis, F.L. Inverse reinforcement learning for trajectory imitation using static output feedback control. *IEEE Trans. Cybern.* **2023**, *54*, 1695–1707. [[CrossRef](#)] [[PubMed](#)]
- Lian, B.; Xue, W.; Lewis, F.L.; Davoudi, A. Inverse Q-learning using input–output data. *IEEE Trans. Cybern.* **2023**, *54*, 728–738. [[CrossRef](#)] [[PubMed](#)]
- Wu, Z.; Sun, L.; Zhan, W.; Yang, C.; Tomizuka, M. Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving. *IEEE Robot. Autom. Lett.* **2020**, *5*, 5355–5362. [[CrossRef](#)]
- Lian, B.; Xue, W.; Lewis, F.L.; Chai, T. Inverse reinforcement learning for adversarial apprentice games. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *34*, 4596–4609. [[CrossRef](#)] [[PubMed](#)]
- Cao, K.; Xie, L. Game-theoretic inverse reinforcement learning: A differential pontryagin’s maximum principle approach. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *34*, 9506–9513. [[CrossRef](#)] [[PubMed](#)]
- Self, R.; Abudia, M.; Mahmud, S.N.; Kamalapurkar, R. Model-based inverse reinforcement learning for deterministic systems. *Automatica* **2022**, *140*, 110242. [[CrossRef](#)]
- Liu, W.; Wu, R.; Si, J.; Huang, H. A new robotic knee impedance control parameter optimization method facilitated by inverse reinforcement learning. *IEEE Robot. Autom. Lett.* **2022**, *7*, 10882–10889. [[CrossRef](#)]
- Xue, W.; Kolaric, P.; Fan, J.; Lian, B.; Chai, T.; Lewis, F.L. Inverse reinforcement learning in tracking control based on inverse optimal control. *IEEE Trans. Cybern.* **2021**, *52*, 10570–10581. [[CrossRef](#)] [[PubMed](#)]
- Self, R.; Coleman, K.; Bai, H.; Kamalapurkar, R. Online observer-based inverse reinforcement learning. *IEEE Control Syst. Lett.* **2020**, *5*, 1922–1927. [[CrossRef](#)]
- Clarke, S.G.; Byeon, S.; Hwang, I. A low complexity approach to model-free stochastic inverse linear quadratic control. *IEEE Access* **2022**, *10*, 9298–9308. [[CrossRef](#)]
- Lian, B.; Kartal, Y.; Lewis, F.L.; Mikulski, D.G.; Hudas, G.R.; Wan, Y.; Davoudi, A. Anomaly detection and correction of optimizing autonomous systems with inverse reinforcement learning. *IEEE Trans. Cybern.* **2022**, *53*, 4555–4566. [[CrossRef](#)] [[PubMed](#)]
- Choi, S.; Kim, S.; Jin Kim, H. Inverse reinforcement learning control for trajectory tracking of a multirotor UAV. *Int. J. Control Autom. Syst.* **2017**, *15*, 1826–1834. [[CrossRef](#)]
- Ho, J.; Ermon, S. Generative adversarial imitation learning. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 4565–4573.
- Lian, B.; Xue, W.; Xie, Y.; Lewis, F.L.; Davoudi, A. Off-policy inverse Q-learning for discrete-time antagonistic unknown systems. *Automatica* **2023**, *155*, 111171. [[CrossRef](#)]
- Zhao, Z.; Wang, T.; Peng, F.; Yu, J. Adaptive Fuzzy Resilient Decentralized Control for Nonlinear Large-Scale CPSs Under DoS Attacks. *IEEE Trans. Fuzzy Syst.* **2024**, *32*, 5899–5909. [[CrossRef](#)]
- Zhao, Z.; Wang, T.; Yu, J.; Basin, M.V. Bilateral Cooperative Control of Nonlinear Multiagent Systems With State and Output Quantification. *IEEE Trans. Cybern.* **2025**, *55*, 2949–2957. [[CrossRef](#)] [[PubMed](#)]
- Xue, W.; Lian, B.; Kartal, Y.; Fan, J.; Chai, T.; Lewis, F.L. Model-free inverse H-infinity control for imitation learning. *IEEE Trans. Autom. Sci. Eng.* **2024**, *22*, 5661–5672. [[CrossRef](#)]
- Lian, B.; Xue, W.; Lewis, F.L.; Chai, T. Inverse reinforcement learning for multi-player noncooperative apprentice games. *Automatica* **2022**, *145*, 110524. [[CrossRef](#)]
- Wu, H.; Hu, Q.; Zheng, J.; Dong, F.; Ouyang, Z.; Li, D. Discounted Inverse Reinforcement Learning for Linear Quadratic Control. *IEEE Trans. Cybern.* **2025**, *55*, 1995–2007. [[CrossRef](#)] [[PubMed](#)]
- Xue, W.; Lian, B.; Fan, J.; Kolaric, P.; Chai, T.; Lewis, F.L. Inverse reinforcement Q-learning through expert imitation for discrete-time systems. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *34*, 2386–2399. [[CrossRef](#)] [[PubMed](#)]

29. Lian, B.; Donge, V.S.; Lewis, F.L.; Chai, T.; Davoudi, A. Data-driven inverse reinforcement learning control for linear multiplayer games. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *35*, 2028–2041. [[CrossRef](#)] [[PubMed](#)]
30. Bittanti, S.; Laub, A.J.; Willems, J.C. *The Riccati Equation*; Springer: Berlin/Heidelberg, Germany, 1991; pp. 131–162.
31. Li, J.; Yuan, D.; Ding, Z. Optimal tracking control for discrete-time systems by model-free off-policy Q-learning approach. In *Proceedings of the 2017 11th Asian Control Conference (ASCC)*; IEEE: Piscataway, NJ, USA, 2017; pp. 7–12.
32. Al-Tamimi, A.; Lewis, F.L.; Abu-Khalaf, M. Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control. *Automatica* **2007**, *43*, 473–481. [[CrossRef](#)]
33. Kiumarsi, B.; Lewis, F.L.; Jiang, Z.P. H-infinity control of linear discrete-time systems: Off-policy reinforcement learning. *Automatica* **2017**, *78*, 144–152. [[CrossRef](#)]
34. Lewis, F.L.; Vamvoudakis, K.G. Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2010**, *41*, 14–25. [[CrossRef](#)] [[PubMed](#)]
35. Rizvi, S.A.A.; Lin, Z. Output Feedback Q-Learning Control for the Discrete-Time Linear Quadratic Regulator Problem. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 1523–1536. [[CrossRef](#)] [[PubMed](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.