biomimetics

Article

Online Inverse Optimal Control for Time-Varying Cost Weights

Sheng Cao *, Zhiwei Luo and Changqin Quan

Graduate School of System Informatics, Kobe University, 1-1 Rokkodai-cho, Nada-ku, Kobe 657-8501, Japan; luo@gold.kobe-u.ac.jp (Z.L.); quanchqin@gold.kobe-u.ac.jp (C.Q.)
* Correspondence: jasonsosen@gold.kobe-u.ac.jp

Abstract: Inverse optimal control is a method for recovering the cost function used in an optimal control problem in expert demonstrations. Most studies on inverse optimal control have focused on building the unknown cost function through the linear combination of given features with unknown cost weights, which are generally considered to be constant. However, in many real-world applications, the cost weights may vary over time. In this study, we propose an adaptive online inverse optimal control approach based on a neural-network approximation to address the challenge of recovering time-varying cost weights. We conduct a well-posedness analysis of the problem and suggest a condition for the adaptive goal, under which the weights of the neural network generated to achieve this adaptive goal are unique to the corresponding inverse optimal control problem. Furthermore, we propose an updating law for the weights of the neural network to ensure the stability of the convergence of the solutions. Finally, simulation results for an example linear system are presented to demonstrate the effectiveness of the proposed strategy. The proposed method is applicable to a wide range of problems requiring real-time inverse optimal control calculations.

Keywords: inverse optimal control; online calculation; time-varying cost weights; robust to noises

Citation: Cao, S.; Luo, Z.; Quan, C. Online Inverse Optimal Control for Time-Varying Cost Weights. *Biomimetics* **2024**, *9*, 84. https://doi.org/10.3390/biomimetics9020084

Academic Editor: Liang Li

Received: 16 December 2023
Revised: 25 January 2024
Accepted: 29 January 2024
Published: 31 January 2024

1. Introduction

The integration of biological principles with robotic technology heralds a new era of innovation, with a significant focus on applying optimal control and optimization methods to analyze animal motion. This approach guides robotic movement development evident in [1], which explores the intricate control systems in mammalian locomotion. Such research underpins the development of robots that emulate the efficiency and adaptability found in nature.

These advancements in understanding animal locomotion through optimal control methods set the stage for the relevance of inverse optimal control (IOC). IOC offers a retrospective analysis of expert movements—human or animal—to infer underlying cost functions optimized in these motions. This methodology is crucial when direct modeling of optimal strategies is complex or unknown.

The use of inverse optimal control (IOC) to identify suitable cost functions from the observable control input and state trajectories of experts is becoming increasingly important. Several successful applications of IOC in estimating the cost weights of multi-features have been reported. For example, the knowledge and expertise of specialists can be categorized and exploited in several fields, including robot control and autonomous driving. The authors of [2], who employed game theory in tailoring robot–human interactions, proposed a method for estimating the human cost function and selecting the robot's cost function based on the results, leading to the Nash equilibrium in human–robot interactions. The authors of [3] applied IOC to analyze taxi drivers' route choices. To investigate the cost combination of human motion, the authors of [4] conducted an experiment using IOC techniques to study human motion during the performance of a goal-achieving task using one arm. Additionally, the authors of [5] represented the learning of biological behavior as an inverse linear quadratic regulator (LQR) problem and proposed adaptive methods for

modeling and analyzing human reach-to-grasp behavior. Furthermore, the authors of [6] employed an IOC method to segment human movement.

Linear quadratic regulation is a common optimal control method for linear systems. In the 1960s and 1970s, numerous researchers offered solutions to the inverse LQR problem [7–9]. Recently, the theory of linear matrix inequality was employed to solve the inverse LQR problem [5,10,11]. Regarding the application of the IOC method for nonlinear systems, several approaches involving methods such as passivity-based condition monitoring [12] or robust design [13] have been reported.

Recent studies in the field of IOC have demonstrated significant advancements. The authors of [14] provided a comprehensive review of the methodologies and applications in inverse optimization, highlighting its growing importance across various domains. The authors of [15] introduced a novel method for sequential calculation in discrete-time systems, enhancing the IOC model's efficacy under noisy data conditions. The authors of [16] employed a multi-objective IOC approach to explore motor control objectives in human locomotion, which has implications for predictive simulations in rehabilitation technology. Furthermore, the authors of [17] delve into cost uniqueness in quadratic costs and control-affine systems, shedding light on the non-uniqueness cases in IOC. Moreover, a recent thesis [18] introduces a Collage-Based Approach for solving unique inverse optimal control problems, leveraging the Collage method for ODE inverse problems in conjunction with Pontryagin's Maximum Principle.

Feature-based IOC methods, which involve modeling the cost function as a linear combination of various feature functions with unknown weights, have gained acclaim in recent years [19–22]. However, it may be difficult to apply these methods to the analysis of complex, long-term behaviors using simple feature functions, e.g., analyzing human jumping [23]. To address this challenge, the authors of [24] proposed a technique for recovering phase-dependent weights that switch at unknown phase-transition points. This method employs a moving window along the observed trajectory to identify the phase-transition points, with the window length determined by a recovery matrix aimed at minimizing the number of observations required for successful cost-weight recovery. Although this method is effective in estimating phase-dependent cost weights, the complex computational requirements limit its use in real-time applications, such as human–robot collaboration tasks. Additionally, in this method, the cost weights in each phase are assumed to be fixed, which may not be generalizable. For example, the human jump motion in [23] was analyzed using time-varying, continuous cost weights.

Overall, the IOC still has several shortcomings that need to be addressed, particularly when applied in approximating complex, multi-phase, continuous cost functions in real time. In this paper, we propose a method for recovering the time-varying cost weights in the IOC problem for linear continuous systems using neural networks. Our approach involves constructing an auxiliary estimation system that closely approximates the behavior of the original system, followed by determining the necessary conditions for tuning the weights of the neurons in the neural network to obtain a unique solution for the IOC problem. We demonstrate that the unique solution corresponds to achieving a zero error between the original system state and the auxiliary estimated system state, as well as zero error between the original costate and the integral of the estimated costate. Based on this analysis, we develop two neural-network frameworks: one for approximating the cost-weight function and the other for addressing the error introduced by the auxiliary estimation system. Additionally, we discuss the necessary requirements for the feature functions to ensure the well-posedness of our online IOC method. Finally, we validate the effectiveness of our method through simulations.

This work makes several significant contributions:

- We provide a solution for the recovery of time-varying cost weights, essential for analyzing real-world animal or human motion.

- Our method operates online, suitable for a broad spectrum of real-time calculation problems. This contrasts with previous online IOC methods that mainly focused on constant cost weights for discrete system control.
- We introduce a neural network and state observer-based framework for online verification and refinement of estimated cost weights. This innovation addresses the critical need for solution uniqueness and robustness against data noise in IOC applications.

2. Problem Formulation

2.1. System Description and Problem Statement

Consider an object's system dynamics formulated as

$$\dot{x} = Ax + Bu \tag{1}$$

where $A \in R^{n\times n}$ and $B \in R^{n\times m}$ are two time-invariant matrixes, $x \in R^n$ represents the system states, and $u = [u_1, \ldots, u_m]^T \in R^m$ denotes the control input of the system [25].

To minimize the following cost function while accounting for dynamics (1), the classic optimal control problem is required to design the optimal control input $u^*(t)$, and generate a sequence of optimal states $x^*(t)$. (Superscript $*$ stands for the optimal condition.)

$$V(x,t) = \int_t^{t_f} L_0(x,u,\tau)d\tau \tag{2}$$

Here, L_0 has the following form:

$$L_0 = q^T F(x) + r^T G(u) \tag{3}$$

where $q = [q_1, q_2, \ldots, q_{n_f}]^T \in R^{n_f}$ and $r = [r_1, r_2, \ldots, r_m]^T \in R^m \ \forall r_i > 0$ represent the cost weight vectors, $F(x)$ is referred to as the general union feature vector with respect to x, and $G(u)$ indicates the feature vector that is only relevant to the control input u [26]. n_f represents the feature's number, which is different from the dimension of system states. For simplicity, we assume that $r^T G(u) = u^T R u$ where R is an unknown matrix with $R = \begin{bmatrix} r_1 & 0 & \ldots \\ \vdots & \ddots & \vdots \\ \ldots & 0 & r_m \end{bmatrix}$. Additionally, it is assumed that (A, B) is controllable, B is a full column rank matrix, and A and B are bounded such that $||A|| \leq \delta_A \ \ ||B|| \leq \delta_B$.

2.2. Maximum Principle in Forward Optimal Control

To minimize the cost function as is the case in (2) with L_0 defined in (3), there exists a costate variable vector λ that satisfies Pontryagin's maximum principle as follows:

$$\dot{\lambda} = -\bar{F}_x^T q - A^T \lambda \tag{4}$$

$$Ru + B^T \lambda = 0 \tag{5}$$

where $\bar{F}_x = \frac{\partial F(x)}{\partial x}$ and $\lambda \in R^n$ denote the costate variables. These two equations are derived from Pontryagin's Maximum Principle by taking the partial derivatives of the Hamiltonian function defined by $\mathbb{H}(x,u,\lambda) = L_0 + \lambda^T(Ax + Bu)$, specifically $\dot{\lambda} = -\frac{\partial \mathbb{H}}{\partial x}$ and $\frac{\partial \mathbb{H}}{\partial u} = 0$. The initial value of λ can be represented as λ_0.

The optimal control input u^* of the system expressed by (1) is given as

$$u^* = -R^{-1}B^T\lambda \tag{6}$$

where λ is unknown. Thus, using this optimal control input, we have

$$\dot{x} = Ax - H\lambda \tag{7}$$

where H denotes the matrix $H = BR^{-1}B^T$. Notably, given that B is a full column rank matrix, it is clear that H is invertible. In addition, since B is a bounded constant matrix, there exists a positive scalar δ_H such that H satisfies $||H|| \leq \delta_H$.

Additionally, the time derivatives of the system dynamics can be formulated as follows:

$$\ddot{x} = A\dot{x} - H\dot{\lambda} \tag{8}$$

2.3. Analysis of the IOC Problem

We assume that the system states $x[t, t_f]$ and the control input $u[t, t_f]$, which represent the time series of the system states and control inputs from time point t to t_f, provide the solution to the optimal minimization of the cost function (2). In addition, we assume that the optimal system states and control input satisfy the boundary conditions $||x|| \leq \delta_x$ $||u|| \leq \delta_u$ $||\dot{u}|| \leq \delta_{\dot{u}}$.

The objective of the IOC problem is to recover the unknown cost weight's vector $q(t)$. Furthermore, IOC, for example, may be employed to analyze different behaviors such as the effect of different occasions on the relative importance of certain human motion feature functions. A rigorous analysis of the derived cost weights that can recreate the original data $x[t, t_f], u[t, t_f]$ is required for the aforementioned applications. To begin, we consider two problems:

- What happens when a different feature function is selected?

In previous studies, it was assumed that the cost weight vector q is either a constant value [19] or a step function with multiple phases [24]. These assumptions have been effective in recovering the cost weights used in the analysis of optimal control methods for a robot's motion control, such as analyzing the motion of a robot controlled by a LQR approach. However, occasionally, it may be inappropriate to assume that the cost weights are constants or step functions when analyzing the complex behaviors of natural objects, such as human motion. In particular, deciding which feature function to adopt when evaluating the motion of natural objects could pose a challenge.

Proposition 1. *Depending on the different selections of feature functions $F(x)$ for the IOC, the original constant cost weight q may become a time-varying continuous function.*

Proof. From (8), for the objects' original feature function, we have

$$H^{-1}(-\ddot{x} + A\dot{x} + HA^T H^{-1}Bu) = \bar{F}_{ox}^T q_o \tag{9}$$

where q_o denotes the original time-invariant cost weight vector, and $\bar{F}_o(x)$ denotes the partial derivative with respect to x of the original feature function. When we choose a different feature function $F_n(x)$, the above equation becomes

$$H^{-1}(-\ddot{x} + A\dot{x} + HA^T H^{-1}Bu) = \bar{F}_{nx}^T q_n \tag{10}$$

where $\bar{F}_{nx}$ denotes the partial derivative with respect to x of the new selected feature function and q_n is the corresponding cost weights on $\bar{F}_{nx}$. Thus, we have

$$\bar{F}_{ox}^T q_o = \bar{F}_{nx}^T q_n \quad \forall t_0 \leq t \leq t_f$$

From this equation, it follows that q_n may be a time-varying function when $\bar{F}_{ox}$ and $\bar{F}_{nx}$ are not equivalent, and as $\bar{F}_{ox}$ and $\bar{F}_{nx}$ are continuous functions, we can reasonably conclude that q_n is also a continuous function. □

Based on this proposition, it is crucial to expand the definition of cost weights to include time-varying values, as this will facilitate a more accurate analysis of the motion of increasingly complex natural objects. Despite the need for time-varying cost weight recovery in many applications, it has received minimal research attention thus far.

- Whether or not the given set $x[t, t_f], u[t, t_f]$ in the IOC problem has a unique solution $\{q(t), r\}$.

The uniqueness of the solution to the IOC problem when cost weights are constant has been discussed in many studies [15,17,18,22]. In this work, we determine if there is still a unique solution to the IOC problem when q is a time-varying function.

From (10), we can find different continuous functions $q(t)$ such that the equation is satisfied for different values of R (different values of H). This implies that if q is considered as a time-varying function, the set $\{q(t), r\}$ will not have a unique solution.

Therefore, when we consider the unique solution of the IOC problem with the time-varying function $q(t)$, it is necessary to introduce additional conditions to ensure that the IOC problem has a unique solution and that the resulting unique solution is meaningful.

In this study, for simplicity, we assume that $R = I$ [27,28], where I is the identity matrix. In actual optimal control cost functions, when we focus on reducing one of the control inputs u_i, the convergence of the i-th system state x_i related to u_i will also be affected. Consequently, the final control result shows that the change in each state of the system is not solely influenced by the chosen cost weights $q(t)$, but also by $R(t)$. In the IOC problem, setting $R(t) = I$ allows the effect of different weights on different control inputs in the original system to be reflected in the current estimate of $q(t)$. This enables us to view the estimated weights on the system states as representing the relative importance of each state in the system's dynamic evolution, without considering the impact of the control input on these weights.

Based on our conclusion that q may be time-varying when different feature functions are chosen and on the corresponding conditions under which a unique solution exists, we can define the IOC problem to be solved in this study as follows:

Problem 1. *Online Estimation of Time-Varying Cost weights $q(t)$*

Given*: (1) Measured system state x as well as control input u (2) $R = I$*

Goal*: Online estimate of the time-varying $q(t)$ utilizing the given x and u.*

3. Adaptive Observer-Based Neural Network Approximation of Time-Varying Cost Weights

In this study, we estimate time-varying cost weight functions online using an observer-based adaptive neural network estimation approach, as opposed to earlier studies that required a large number of time series of x and u to recover fixed cost weights offline.

Construction of the Observer

Following the introduction of $\hat{q}(t) \in R^n$ denoting the estimation of $q(t)$, we define the estimation of the associated costate variable $\hat{\lambda}$ as follows:

$$\dot{\hat{\lambda}} = -\bar{F}_{\hat{x}}^T \hat{q}(t) - A^T \hat{\lambda} \tag{11}$$

where $\bar{F}_{\hat{x}} = \frac{\partial F(\hat{x})}{\partial \hat{x}}$ denotes the partial derivatives of the feature functions that are only relevant to the estimated system states $\hat{x}$ obtained by inserting $\hat{\lambda}$ into (7):

$$\dot{\hat{x}} = A\hat{x} - H\hat{\lambda} \tag{12}$$

where the initial state $\hat{x}_0$ of this system is selected to be $\hat{x}_0 = x_0$.

Thus, compared with that of the original system, the error generated by the new estimation system can be expressed as

$$\dot{\tilde{x}} = A\tilde{x} - H\tilde{\lambda} \tag{13}$$

$$\dot{\tilde{\lambda}} = -\bar{F}_x^T q(t) + \bar{F}_{\hat{x}}^T \hat{q}(t) - A^T \tilde{\lambda} \tag{14}$$

where $\tilde{\lambda} = \lambda - \hat{\lambda}$ and $\tilde{x} = x - \hat{x}$. Here, the feature function is selected such that its partial derivative with respect to x is bounded and it is assumed that $||\bar{F}_x^x|| \leq \delta_{nx}$, $||\bar{F}_{\hat{x}}|| \leq \delta_{n\hat{x}}$ and $||\bar{F}_x^x(x) - \bar{F}_{\hat{x}}(\hat{x})|| \leq \zeta||\tilde{x}||$ where δ_{nx}, $\delta_{n\hat{x}}$ and ζ denote a positive scalar.

Additionally, the time derivatives of (14) can be expressed as

$$\ddot{\tilde{x}} = A\dot{\tilde{x}} - H\dot{\tilde{\lambda}} \tag{15}$$

Thus, the following equation can be satisfied:

$$\dot{s} = A_r s + T_x\tilde{q} + (T_x - T_{\hat{x}})\hat{q} \tag{16}$$

where $s = \begin{bmatrix}\dot{\tilde{x}}\\ \tilde{\lambda}\end{bmatrix}$, $A_r = \begin{bmatrix}A & HA^T\\ 0 & -A^T\end{bmatrix}$, $T_x = \begin{bmatrix}H\bar{F}_x\\ -\bar{F}_x\end{bmatrix}$, $T_{\hat{x}} = \begin{bmatrix}H\bar{F}_{\hat{x}}\\ -\bar{F}_{\hat{x}}\end{bmatrix}$. $\tilde{q}$ denotes the error of estimating q. Here, $||\bar{F}_x^x(x) - \bar{F}_{\hat{x}}(\hat{x})|| \leq \zeta||\tilde{x}||$ implies that there exists a positive scalar ζ' such that $||T_x - T_{\hat{x}}|| \leq \zeta'||\tilde{x}||$ holds. Based on the bound of $\bar{F}_x^x(x), \bar{F}_{\hat{x}}(\hat{x}), H$, it follows that there are two positive scalars δ_{t_x} and $\delta_{t_{\hat{x}}}$ such that the following inequalities hold: $||T_x|| \leq \delta_{t_x}$ and $||T_{\hat{x}}|| \leq \delta_{t_{\hat{x}}}$.

Moreover, from (6) and (7), λ can be calculated as follows:

$$\lambda = -H^{-1}Bu \tag{17}$$

4. Neural Network-Based Approximation of Time Varying Cost Weights

In this section, a neural network-based cost weight approximation algorithm is proposed. To calculate an approximation of the time-varying vector q, we adopt a neural network in which the chosen inputs are $u_I = \begin{bmatrix}x_0\\ u\end{bmatrix}$, where x_0 denotes the initial state of the system (1). Based on this, we assume that time-invariant weight matrixes $W \in R^{n_f \times l}$ exist that satisfy the following expression:

$$q = W^T\phi(u_I) + \epsilon_1(u_I) \tag{18}$$

where $\phi(u_I)$ denotes the activation function and $\epsilon_1(u_I)$ denotes the structure approximation error of the neural networks. In addition, the activation function selected enables the activation function as well as its partial derivative to satisfy the following boundary condition: $||\phi(u_I)|| \leq \delta_p$ and $||\frac{\partial\phi(u_I)}{\partial u_I}|| \leq \delta_{pu}$ where δ_p and δ_{pu} represent two positive scalars. Additionally, $||\epsilon_1(u_I)|| \leq \epsilon_n$ where ϵ_n is a positive scalar.

The estimate of vector q is constructed as follows:

$$\hat{q} = \hat{W}^T\phi(u_I) \tag{19}$$

where $\hat{W}$ denotes the estimation of W. In this paper, we will combine two estimators $\hat{W}_1$ and $\hat{W}_2$ to estimate W, as shown in Section 4.1. Before presenting the details of the estimators, we first discuss the necessary conditions for the estimation.

Based on the setting of estimator $\hat{W}$, the error of estimating q can be expressed as

$$\tilde{q} = q - \hat{q} = \tilde{W}^T\phi(u_I) + \epsilon_1(u_I) \tag{20}$$

where $\tilde{W} = W - \hat{W}$ denotes the error of estimating W. Substituting $\tilde{q}$ into (16) yields

$$\dot{s} = A_r s + T_x\tilde{W}^T\phi(u_I) + (T_x - T_{\hat{x}})\hat{q} + T_x\epsilon_1(u_I) \tag{21}$$

To profoundly comprehend the necessary condition for the convergence of the estimation error $\tilde{W}$, we define uniformly ultimately bounded (UUB) below.

Definition 1. *A time-varying signal $\sigma(t)$ can be said as UUB if there exists a compact set $S \subset R^n$ so that for all $\sigma \in S$, there exists a bound $\mu \geq 0$ and a time T such that $||\sigma|| \leq \mu$ for all $t \geq t_0 + T$.*

Lemma 1. *If the following conditions are satisfied, $\tilde{W}$ becomes UUB.*

- *$\int_{t_0}^{t_i} sdt, s$ become UUB after a time point t_1 ($||\int_{t_0}^{t_i} sdt|| \leq \delta_1$, and $||s|| \leq \delta_2$)*
- *The change in $\hat{W}$ approaches zero*
- *Matrix C defined below will become a full row rank matrix.*

$$C = \begin{bmatrix} \int_{t_1+1}^{t_1+2} T_x(I \otimes \phi(u_I))^T dt \\ \vdots \\ \int_{t_i-1}^{t_i} T_x(I \otimes \phi(u_I))^T dt \end{bmatrix} \tag{22}$$

where $t_1 \leq t_i \leq t_f$ and any term in C satisfies the persistent excitation (PE) condition defined below.

$$|| \int_{t_j}^{t_j+1} T_x(I \otimes \phi(u_I))dt)^T || \geq \beta_j \ \ \forall t_1 \leq t_j \leq t_i \tag{23}$$

Here, β_j is a positive value.

Proof. From (21)

$$\begin{aligned} s = A_r \int_{t_0}^{t_i} sdt + \int_{t_0}^{t_i} T_x \tilde{W}^T \phi(u_I) dt \\ + \int_{t_0}^{t_i} (T_x - T_{\hat{x}}) \hat{q} dt + \int_{t_0}^{t_i} T_x \epsilon_1(u_I) dt \end{aligned} \tag{24}$$

Since $\int_{t_0}^{t_i} sdt \to 0, s \to 0$ reaches a steady state and A_r is a constant, we can obtain the following:

$$||s - A_r \int_{t_0}^{t_i} sdt|| \leq \delta_{si} \tag{25}$$

where δ_{si} denotes a small positive scalar. Additionally, with both $\epsilon_1(u_I)$ and T_x being bounded, this leads to

$$|| \int_{t_0}^{t_i} T_x \epsilon_1(u_I) dt|| \leq \delta_{T\epsilon} \tag{26}$$

where $\delta_{T\epsilon}$ denotes a small positive scalar. The term $\int_{t_0}^{t_i} T_x \epsilon_1(u_I) dt$ captures the effect of the structural error of the neural network on state s. Since T_x is bounded, when the neural network approximates the cost weight function adequately, the value of $\epsilon_1(u_I)$ decreases, which in turn minimizes the overall integral value. In other words, a well-selected neural network structure with a good approximation of the cost weight function will produce a small structure error and, therefore, a small overall integral value $\int_{t_0}^{t_i} T_x \epsilon_1(u_I) dt$.

(24)–(26) leads to

$$|| \int_{t_0}^{t_i} T_x \tilde{W}^T \phi(u_I) dt + \int_{t_0}^{t_i} (T_x - T_{\hat{x}}) \hat{q} dt|| \leq \delta_{si} + \delta_{T\epsilon} \tag{27}$$

Similarly, we can obtain a similar relation for the duration $[t_0, t_1]$

$$|| \int_{t_0}^{t_1} T_x \tilde{W}^T \phi(u_I) dt + \int_{t_0}^{t_1} (T_x - T_{\hat{x}}) \hat{q} dt|| \leq \delta_{si} + \delta_{T\epsilon} \tag{28}$$

From (27) and (28), it follows that

$$|| \int_{t_1+1}^{t_i} T_x \tilde{W}^T \phi(u_I) dt + \int_{t_1+1}^{t_i} (T_x - T_{\hat{x}}) \hat{q} dt|| \leq 2(\delta_{si} + \delta_{T\epsilon}) \tag{29}$$

Furthermore, considering $\int_{t_0}^{t_i} s dt \to 0$ after t_1, the definition of s and $||T_x - T_{\hat{x}}|| \leq \zeta' ||\tilde{x}||$, this implies that

$$\begin{aligned} || \int_{t_1+1}^{t_i} (T_x - T_{\hat{x}}) \hat{q} dt|| &\leq \int_{t_1+1}^{t_i} ||(T_x - T_{\hat{x}})|| ||\hat{q}|| dt \\ &\leq \int_{t_1+1}^{t_i} \zeta' \delta_{\tilde{x}} \delta_{\hat{q}} dt \equiv \delta_{\zeta(t_i - t_1 - 1)} \end{aligned} \tag{30}$$

where $\delta_{\tilde{x}}$ and $\delta_{\hat{q}}$ represent the bounds of $\tilde{x}$ and $\hat{q}$ respectively. Thus, this leads to the inequality

$$|| \int_{t_1+1}^{t_i} T_x \tilde{W}^T \phi(u_I) dt|| \leq 2(\delta_{si} + \delta_{T\epsilon}) + \delta_{\zeta(t_i - t_1 - 1)} \tag{31}$$

In this case, when $\dot{\hat{W}}$ approaches zero, the following relation emerges:

$$\begin{aligned} &|| \int_{t_1+1}^{t_i} T_x (I \otimes \phi(u_I))^T vec(\tilde{W}) dt|| \\ &= || \int_{t_1+1}^{t_i} T_x (I \otimes \phi(u_I))^T dt \; vec(\tilde{W})|| \\ &\leq 2(\delta_{si} + \delta_{T\epsilon}) + \delta_{\zeta(t_i - t_1 - 1)} \end{aligned} \tag{32}$$

Based on this relation, it follows that

$$\begin{aligned} || \int_{t_1+1}^{t_1+2} T_x (I \otimes \phi(u_I))^T dt \; vec(\tilde{W})|| \\ \leq 2(\delta_{si} + \delta_{T\epsilon}) + \delta_{\zeta(1)} \end{aligned} \tag{33}$$

where $\delta_{\zeta(1)} = \int_{t_1+1}^{t_1+2} \zeta' \delta_{\tilde{x}} \delta_{\hat{q}} dt = \cdots = \int_{t_i-1}^{t_i} \zeta' \delta_{\tilde{x}} \delta_{\hat{q}} dt$.

Thus, it implies that

$$||C \; vec(\tilde{W})|| \leq (t_i - t_1 - 1)(2(\delta_{si} + \delta_{T\epsilon}) + \delta_{\zeta(1)}) \tag{34}$$

where C is defined in (22). Due to C being full row rank, this leads to

$$\begin{aligned} ||vec(\tilde{W})|| &\leq ||C^+|| ||C \; vec(\tilde{W})|| \\ &\leq ||C^+|| (t_i - t_1 - 1)(2(\delta_{si} + \delta_{T\epsilon}) + \delta_{\zeta(1)}) \end{aligned} \tag{35}$$

From (23), we have $||C^+|| \leq \frac{1}{\sqrt{(t_i - t_1 - 1)\beta_j^2}}$

$$||vec(\tilde{W})|| \leq \sqrt{\frac{t_i - t_1 - 1}{\beta_j^2}} (2(\delta_{si} + \delta_{T\epsilon}) + \delta_{\zeta(1)}) \tag{36}$$

Thus, $\tilde{W}$ is UUB.

Notably, β_j evaluates the lower bound of the norm of $\int_{t_j}^{t_j+1} T_x (I \otimes \phi(u_I)) dt)^T$, it can increase when the data x cause the norm of the integral to deviate significantly from zero. The size of $\delta_{\zeta(1)}, \delta_{si}$ is related to the minimization of s and $\int_{t_0}^{t_i} s dt$, and the size of $\delta_{T\epsilon}$ is related to the approximation ability of the chosen neural network. The bound of $\tilde{W}$ after t_1 can be minimized by the excited x, successfully minimizing s and $\int_{t_0}^{t_i} s dt$ while appropriately designing the structure of the neural network. □

4.1. Construction of the Neural Network

As shown in Lemma 1, the convergence of $\int_{t_0}^{t} s d\tau$ is essential in the convergence of $\tilde{W}$ to 0. Therefore, it is necessary to incorporate this consideration in the approximation design.

First, we divide the estimation of the weights of the neural network into two parts:

$$\hat{W} = \hat{W}_1 + \hat{W}_2 \tag{37}$$

and

$$\hat{q} = \hat{q}_1 + \hat{q}_2 = (\hat{W}_1 + \hat{W}_2)^T \phi(u_I) \tag{38}$$

where $\hat{q}_1 = \hat{W}_1^T \phi(u_I)$ and $\hat{q}_2 = \hat{W}_2^T \phi(u_I)$.

The necessity for employing two distinct estimators, $\hat{W}_1$ and $\hat{W}_2$, is rooted in their specialized roles in minimizing the tracking error s. This dual-estimator approach ensures that $\hat{q}(t)$ closely aligns with the desired trajectory $q(t)$. While $\hat{W}_1$'s adaptive tuning is primarily aimed at steering s towards zero, its inherent residual errors in its adaptive process necessitate the deployment of $\hat{W}_2$ for error compensation and enhanced accuracy in tracking the ideal cost weight $q(t)$. To gain a deeper understanding of this system, we will begin by examining the error dynamics, which forms a fundamental basis for the subsequent detailed exploration of the tuning laws for each estimator.

The state equation describing the error dynamics can be obtained as follows:

$$\dot{s} = A_r s + T_x \tilde{q}_1 + (T_x - T_{\hat{x}})\hat{q}_1 - T_{\hat{x}}\hat{q}_2 \tag{39}$$

where $s = \begin{bmatrix} \tilde{x} \\ \tilde{\lambda} \end{bmatrix}$, $A_r = \begin{bmatrix} A & HA^T \\ 0 & -A^T \end{bmatrix}$, $T_x = \begin{bmatrix} H\bar{F}_x \\ -\bar{F}_x \end{bmatrix}$, $T_{\hat{x}} = \begin{bmatrix} H\bar{F}_{\hat{x}} \\ -\bar{F}_{\hat{x}} \end{bmatrix}$.

Further, to effectively minimize $\int_{t_0}^{t} s d\tau$, we define vector e as follows:

$$e = (T_x - T_{\hat{x}})\hat{q}_1 + Ks + K_p \int_{t_0}^{t} s d\tau - T_{\hat{x}}\hat{q}_2 + A_r s \tag{40}$$

where $K = diag([k, \ldots, k]) \in \mathbb{R}^{2n \times 2n}$ and $K_p = diag([k_p, \ldots, k_p]) \in \mathbb{R}^{2n \times 2n}$. Parameters k and k_p are two positive scalars, thus, (39) can be written as:

$$\dot{s} = -Ks - K_p \int_{t_0}^{t} s d\tau + T_x \tilde{q}_1 + e \tag{41}$$

We suppose that an ideal time-invariant weight matrix $W_2 \in R^{n_f \times l}$ exists, which guarantees that

$$\begin{aligned} &(T_x - T_{\hat{x}})\hat{q}_1 + Ks + A_r s + K_p \int_{t_0}^{t} s d\tau \\ &= T_{\hat{x}} q' = T_{\hat{x}}(W_2^T \phi(u_I) + \epsilon_2(u_I)) \end{aligned} \tag{42}$$

where $u_I = \begin{bmatrix} x_0 \\ u \end{bmatrix}$.

The estimation error of the neural network can be represented as

$$\begin{aligned} \tilde{q}_1 &\equiv q - \hat{q}_1 = \tilde{W}_1^T \phi(u_I) + \epsilon_1(u_I) \\ \tilde{q}_2 &\equiv q' - \hat{q}_2 = \tilde{W}_2^T \phi(u_I) + \epsilon_2(u_I) \end{aligned} \tag{43}$$

and e can be represented as

$$e = T_{\hat{x}}(\tilde{W}_2^T \phi(u_I) + \epsilon_2(u_I)) \tag{44}$$

Therefore, (41) becomes

$$\dot{s} = -Ks - K_p \int_{t_0}^{t} s d\tau + T_x(\tilde{W}_1^T \phi(u_I) + \epsilon_1(u_I)) + T_{\hat{x}}(\tilde{W}_2^T \phi(u_I) + \epsilon_2(u_I)) \quad (45)$$

4.2. Tuning Law of the Neural Network for the Estimation of $q(t)$

An updating law for a neural network that estimates $q(t)$ can be represented in Theorem 1, based on the error system's dynamics that were derived in (45).

Theorem 1. *If we choose the updating laws for the neural network weights $\hat{W}_1$ and $\hat{W}_2$ as shown in (46), respectively, where Γ_1 , Γ_2, and k_e are positive scalar constants, then state s, $\int_{t_0}^{t} s d\tau$ and error e will be UUB.*

$$\dot{\hat{W}}_1 = \Gamma_1 \phi(u_I) s^T T_x$$
$$\dot{\hat{W}}_2 = \Gamma_2 \phi(u_I)(s + k_e e)^T T_{\hat{x}} \quad (46)$$

In addition, if there exist positive constants t_δ, β_1, β_2, β_3, and β_4 such that the inequalities in (47) are satisfied for all initial times t_0, then the signals $\tilde{W}_1$ and $\tilde{W}_2$ will also be UUB.

$$\beta_2 I \geq \int_{t_0}^{t_0+t_\delta} C_{p1}(t)^T C_{p1}(t) dt \geq \beta_1 I$$
$$\beta_4 I \geq \int_{t_0}^{t_0+t_\delta} C_{p2}(t)^T C_{p2}(t) dt \geq \beta_3 I \quad (47)$$

Here, $C_{p1}(t) = T_x(I \otimes \phi(u_I)^T)$, $C_{p2}(t) = T_{\hat{x}}(I \otimes \phi(u_I)^T)$

Proof. A proof of this theorem can be found in Appendix A. □

Applying (46) results in s, $\int_{t_0}^{t} s d\tau$, and e being UUB, as shown in Theorem 1. Additionally, (46) shows that when s and e decreases, $\dot{\hat{W}}_1$ and $\dot{\hat{W}}_2$ decrease as well, resulting in a decrease in $\dot{\hat{W}} = \dot{\hat{W}}_1 + \dot{\hat{W}}_2$. At this point, as stated in Lemma 1, if the condition of matrix C (defined in Lemma 1), being a full row rank matrix, is satisfied, then $\tilde{W} = \tilde{W}_1 + \tilde{W}_2$ will also be UUB. Thus, the solution to the IOC problem can be derived by applying (38).

5. Simulations

5.1. Basic Simulation Conditions

To verify the effectiveness of our method, we performed the simulations using a sample linear system controlled by the optimal control method with the original cost weights R selected in two cases.

The sample linear system dynamics can be formulated as follows:

$$\dot{\theta} = A\theta + B\tau \quad (48)$$

where $\theta = [\theta_1, \theta_2]^T \in R^2$ represents the system states. We select $A = \begin{bmatrix} 30 & 80 \\ 60 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$ and $\tau \in R^2$ denoting the control input.

The cost function selected in these simulations is formulated as

$$V_r = \frac{1}{2} \int_0^{t_f} (\theta^T Q(t) \theta + \tau^T R \tau) dt \quad (49)$$

when all the elements of θ satisfying $|\theta_i| \leq \theta_{r_l}$ and $Q(t) = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix}$ is the continuous time-varying cost weights on system states θ. $R = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix}$ represents the cost weights on the control inputs.

Moreover, in our simulations, we select 0 as the initial value of all the elements of both $\hat{W}_1$ and $\hat{W}_2$. Actuation function $\phi(u_I)$ was selected as $\phi(u_I) = [\phi_1(u_I), \ldots, \phi_i(u_I), \ldots, \phi_l(u_I)]^T$ with $\phi_i(u_I)$ designed as

$$\phi_i(u_I) = exp(\frac{-(u_I - \psi_i)^T(u_I - \psi_i)}{\nu}) \tag{50}$$

where ν denotes a positive scalar and ψ_i denotes the center of the respective activation function. We initialized the activation function centers on a four-dimensional grid to match the dimension of u_i, ensuring a uniform distribution across the input space and enhancing network adaptability.

The overall implementation for recovering the time-varying cost weights is shown in Algorithm 1.

Algorithm 1 Online implementation

Input: $\{x_i, u_i\}$
Output: $\hat{q}(t)$
Initialization :
Initialize $\hat{\lambda}$, $\hat{x}$, $\hat{W}_1$, $\hat{W}_1$, $\hat{W}_2$, Γ_1, Γ_2 and $R = I$.
LOOP Process
for $i = 0$ to K **do**
  Calculate λ using $\lambda = -H^{-1}Bu$.
  Calculate $\dot{\hat{x}}$ and $\dot{\hat{\lambda}}$ using (11) and (12).
  Calculate $\dot{\tilde{x}}$ and $\dot{\tilde{\lambda}}$ using (14) and (13).
  Calculate $s = \begin{bmatrix} \dot{\tilde{x}} \\ \tilde{\lambda} \end{bmatrix}$.
  Calculate e following (40).
  Calculate $\phi(u_I)$ and update $\hat{W}_1$, $\hat{W}_2$ using (46).
  Calculate $\hat{q}(t)$ using (38).
end for
return $\hat{q}(t)$

Two cases are considered in the simulation:

- In the first case, we apply the optimal control of the sample system with cost weights θ as the signal ($q_1(t) = 1 + cos(t)$ and $q_2(t) = 2 + sin(t)$). The proposed IOC method is employed online to estimate the cost weights, with the simultaneous online recovery of the original system trajectory. Parameters Γ_1 and Γ_2 in the updating law are set to $\Gamma_1 = 1$ and $\Gamma_2 = 1$, respectively. Parameters k and k_p are set to $k = 50$ and $k_p = 625$, respectively. The initial values of $\hat{W}_1$ and $\hat{W}_2$ are set to matrixes with all elements equal to zero. The original r_1 and r_2 are set to $r_1 = 1$ and $r_2 = 1$, respectively. The simulation also uses 49 nodes in the neural network.
- In the second case, we perform the simulation of our IOC method, but with the original r_1 and r_2 set to $r_1 = 3$ and $r_2 = 4$, respectively. All other simulation settings are the same as in the first case.

Similar to the simulation sections in previous works ([6,24]), we use the control input from the simulation, which ignores the measurement issues with the control input and measurement errors that may occur in real-world applications. This allows us to purely evaluate the performance of our method in solving the IOC problem. In actual applications,

the control input can be calculated by substituting the measured $\dot{\theta}$ into (48), as described in [24].

5.2. Results

The simulation results are shown in the figures below.

In Figure 1, the blue solid line represents the original variation in the cost weights whereas the gray solid line represents the estimated cost weights. After a brief period of oscillation at the initial time, our method accurately recovers the original cost weights when $R = I$. Notably, similar to the case in other adaptive control methods and adaptive neural network based control methods, the initial oscillation is a result of the adaptive initialization of the weights in (46) due to the large initial errors in $\tilde{W}_1$ and $\tilde{W}_2$.

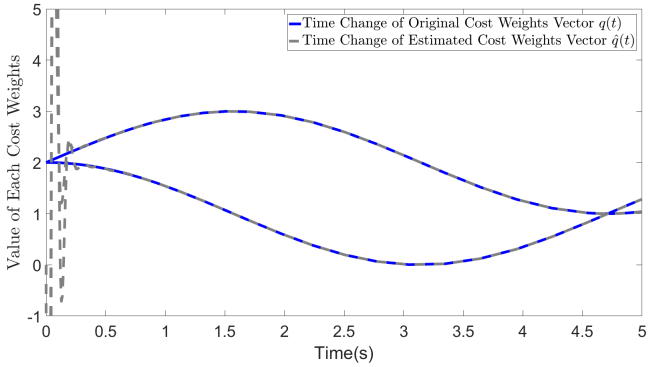

Figure 1. Estimated cost weights ($r_1 = 1, r_2 = 1$).

Figure 2 demonstrates the impact of selecting $R = I$ on the estimation results when the original R value is arbitrary. The solid blue line represents the original time-varying cost weights, whereas the dotted gray line represents the final estimated values. Although the estimated values differ from the original values, the general trend of the changes is preserved. In addition, the gray line represents the mutual weights in the dynamics of the system state, whereas the original weights among the control inputs are reflected in the current estimate of $q(t)$. From the figure, we can observe that the bottom lines in blue and gray colors represent the value of the original and estimated q_2. Evidently, the blue line for q_2 is larger than that for q_1 from 4.8 s to 5 s. Additionally, in the original settings, r_2 is 4, which confers greater importance to the decrease in u_2 compared with the case when $r_1 = 3$, leading to the weakening of the convergence of the θ_2 term associated with u_2. In our estimates, the value of the dashed line for the estimated q_2, which also considers the impact from original setting of R is not greater than the value of estimated q_1 between 4.8 s and 5 s. This indicates that the convergence of θ_2 is weakened by considering the impact from the cost weights on control input. Our dashed line more accurately reflects the actual situation compared to the blue line.

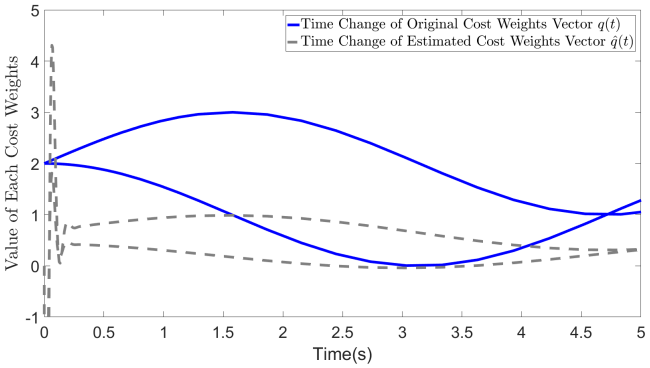

Figure 2. Estimated cost weights ($r_1 = 3, r_2 = 4$).

In Figures 3–5, we show the results of error e, states s and $\int_{t_0}^{t} s d\tau$ in two cases. The blue lines show the results of the first case, whereas the gray dotted lines show the results of the second case. From the figures, we can observe that all the values effectively decrease to a low range during the simulation, and most importantly, in the second case, the different selections of R do not affect the convergence of these values. This demonstrates the effectiveness of our method and highlights that even with different values of R, the recovered cost weights are still feasible solutions to the IOC problem, as they can be utilized to regenerate a similar system trajectory and control inputs ($\int_{t_0}^{t} s d\tau = \begin{bmatrix} \tilde{x} \\ \int_{t_0}^{t} \tilde{\lambda} d\tau \end{bmatrix} \to 0$).

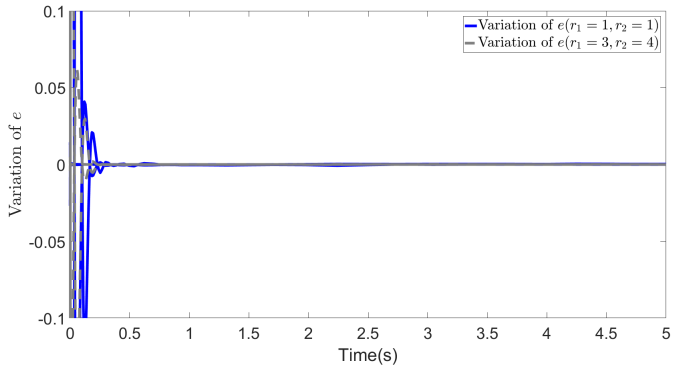

Figure 3. Variation of error e ($r_1 = 1, r_2 = 1$ and $r_1 = 3, r_2 = 4$).

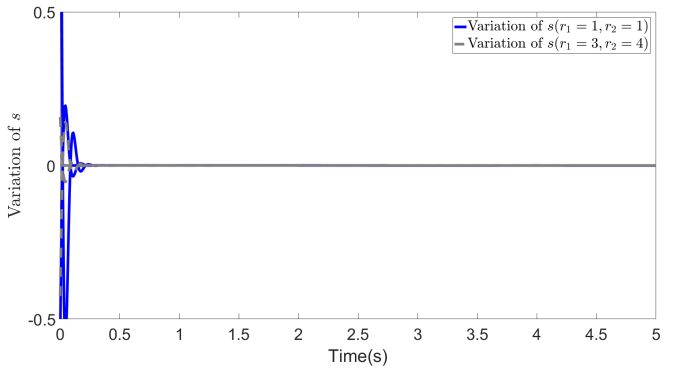

Figure 4. Variation of error s ($r_1 = 1, r_2 = 1$ and $r_1 = 3, r_2 = 4$).

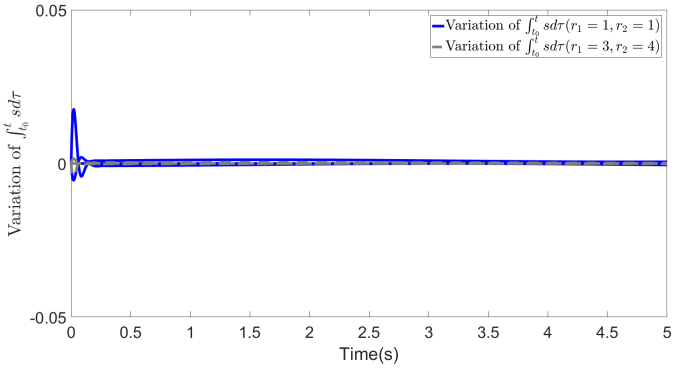

Figure 5. Variation of $\int_{t_0}^{t} s d\tau$ ($r_1 = 1, r_2 = 1$ and $r_1 = 3, r_2 = 4$).

6. Discussion

6.1. Robustness of the Proposed Method to Noisy Data

In (46), Γ_1 and Γ_2 decrease the error by regulating the updating speed of the estimated values. Adjusting these two terms may successfully reduce the impact of data noise to a

certain degree. Their roles are similar to that of a low-pass filter's time constant. For example, in the setting of the first case, when noise exists, $x \sim \mathcal{N}(0, 10^{-1})$ and $u \sim \mathcal{N}(0, 10^{-4})$, the simulation results show that different sets of Γ_1 and Γ_2 (e.g., $\Gamma_1 = 10$, $\Gamma_2 = 10$; $\Gamma_1 = 1$, $\Gamma_2 = 1$) can significantly influence the noise reduction performance.

As shown in Figure 6, while relatively small values of Γ_1 and Γ_2 may result in a low convergence rate, they effectively reduce the impact of data noise. Our method demonstrates robustness against noise by allowing for the adjustment of parameters Γ_1 and Γ_2.

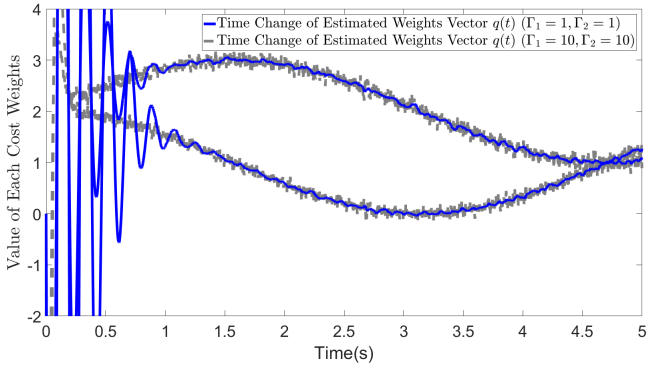

Figure 6. Estimated cost weights (Noisy Case): (1) $\Gamma_1 = 10$, $\Gamma_2 = 10$ (2) $\Gamma_1 = 1$, $\Gamma_2 = 1$.

6.2. Calculation Complexity and Real-Time Calculation

The proposed algorithm has a low computational complexity, as it only involves the calculation of dot products between matrixes and vectors as well as the summation of vectors. Additionally, it does not require any iterative or optimization calculations. This makes it an efficient solution for real-time calculations. In fact, our simulation shows that a single iteration of the algorithm using case 1 settings takes only approximately 0.23 ms in Matlab 2016b to complete the SIOC's calculation, which is fast enough to meet real-time calculation requirements.

6.3. Advantages of Using $R = I$

The simulation results suggest that one of the key advantages of setting R as a constant I is that it effectively consolidates the impact of cost weights on state convergence, which would have been influenced by different settings of R, into the estimated value of $q(t)$. This allows for a comprehensive evaluation of the system state convergence, as it only depends on $q(t)$, without needing to account for additional considerations. Furthermore, by maintaining a consistent value of $R = I$, it is possible to standardize the analysis of the same motion across multiple agents, which is crucial for various applications.

7. Conclusions

In this paper, we proposed a neural network based method for recovering the time-varying cost weights in the IOC problem for linear continuous systems. Our approach involved constructing an auxiliary estimation system that closely approximates the behavior of the original system, followed by determining the necessary conditions for tuning the weights of the neurons in the neural network to obtain a unique solution for the IOC problem. We discussed the necessary requirements for the previous settings to ensure the well-posedness of our online IOC method. We showed that the unique solution corresponds to achieving a nearly zero error between the original system state and the auxiliary estimated system state, as well as nearly zero error between the original costate and the integral of the estimated costate. Based on this analysis, we developed two neural network frameworks: one for approximating the cost weight function and the other for addressing the error introduced by the auxiliary estimation system and terms. Finally, we validated the effectiveness of our method through simulations, highlighting its ability to

recover time-varying cost weights and its robustness against different original choices of R. Overall, our method represents a significant advancement in the field of online IOC, and it is applicable to a wide range of problems requiring real-time IOC calculations.

Author Contributions: Conceptualization, S.C.; methodology, S.C.; software, C.Q.; writing—original draft, S.C.; writing—review and editing, S.C.; project administration, Z.L. and C.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

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

Appendix A. Proof of Theorem 1

Proof. Considering the Lyapunov candidate selected as follows

$$V = \frac{1}{2}s^T s + \frac{1}{2}(\int_{t_0}^{t} s d\tau)^T K_p \int_{t_0}^{t} s d\tau + \frac{1}{2}tr[\tilde{W}_1^T \Gamma_1^{-1} \tilde{W}_1 + \tilde{W}_2^T \Gamma_2^{-1} \tilde{W}_2] \tag{A1}$$

The derivative of V can be expressed as

$$\dot{V} = s^T \dot{s} + s^T K_p \int_{t_0}^{t} s d\tau - tr[\tilde{W}_1^T \Gamma_1^{-1} \dot{\hat{W}}_1 + \tilde{W}_2^T \Gamma_2^{-1} \dot{\hat{W}}_2] \tag{A2}$$

By introducing (45) and utilizing the proposed updating law of $\hat{W}_1$ and $\hat{W}_2$ in (46), $\dot{V}$ becomes

$$\begin{aligned}
\dot{V} =& -s^T k s + s^T T_x \tilde{q}_1 + s^T e \\
& - tr[\tilde{W}_1^T \Gamma_1^{-1} \dot{\hat{W}}_1 + \tilde{W}_2^T \Gamma_2^{-1} \dot{\hat{W}}_2] \\
=& -s^T k s + s^T T_x (\tilde{W}_1 \phi(u_I) + \epsilon_1(u_I)) \\
& + s^T T_{\hat{x}} (\tilde{W}_2 \phi(u_I) + \epsilon_2(u_I)) \\
& - tr[\tilde{W}_1^T \Gamma_1^{-1} \dot{\hat{W}}_1 + \tilde{W}_2^T \Gamma_2^{-1} \dot{\hat{W}}_2] \\
=& -s^T k s + s^T T_x \epsilon_1(u_I) + s^T T_{\hat{x}} \epsilon_2(u_I) \\
& + k_e e^T T_{\hat{x}} \epsilon_2(u_I) - e^T k_e e
\end{aligned} \tag{A3}$$

Here, with introducing a new vector p defined as $p = \begin{bmatrix} s \\ \frac{k_e}{\sqrt{k}} e \end{bmatrix}$ and considering (44), (A3) can be rewritten as

$$\dot{V} = -p^T K p + p^T p_\epsilon \tag{A4}$$

where $p_\epsilon = \begin{bmatrix} T_x \epsilon_1(u_I) + T_{\hat{x}} \epsilon_2(u_I) \\ \sqrt{k} T_{\hat{x}} \epsilon_2(u_I) \end{bmatrix}$.

By considering the boundedness condition of T_x,$T_{\hat{x}}$,$\epsilon_1(u_I)$ and $\epsilon_2(u_I)$, we have

$$||p_\epsilon|| \leq \sqrt{(\delta_{t_x} \epsilon_{n1} + \delta_{t_{\hat{x}}} \epsilon_{n2})^2 + k \delta_{t_{\hat{x}}} \epsilon_{n2}} \equiv \delta_{t_{p\epsilon}} \tag{A5}$$

From this boundedness condition, (A3) becomes

$$\begin{aligned}
\dot{V} &\leq -k||p||^2 + ||p||||p_\epsilon|| \\
&\leq -k||p||^2 + ||p||\delta_{t_{p\epsilon}} \\
&= -||p||(k||p|| - \delta_{t_{p\epsilon}})
\end{aligned} \tag{A6}$$

From (A6), the left hand side of (A6) would be negative when $||p|| \geq \frac{\delta_{tp\epsilon}}{k}$, implying that $\dot{V} \leq 0$ and p would maintain convergence when $||p|| \geq \frac{\delta_{tp\epsilon}}{k}$. Moreover, due to the vector $p = \begin{bmatrix} s \\ \frac{k_e}{\sqrt{k}} e \end{bmatrix}$, s as well as e would all be bounded satisfying

$$||s|| \leq \delta_s \tag{A7}$$

$$||e|| \leq \delta_e \tag{A8}$$

That is, s, e would all be UUB. Moreover, due to the continuity, $\dot{s}$ would also be UUB satisfying the following condition as

$$||s|| \leq \delta_{\dot{s}} \tag{A9}$$

Notably, with increasing k, the bound $\frac{\delta_{tp\epsilon}}{k}$ of p decreases. Furthermore, since V decreases continuously while $||p|| \geq \frac{\delta_{tp\epsilon}}{k}$, $\int_{t_0}^{t} s d\tau$ would also be UUB.

Conversely, from (41), we have

$$\begin{aligned} ||T_x \tilde{q}_1|| &= ||\dot{s} + Ks + K_p \int_{t_0}^{t} s d\tau - e|| \\ &\leq B_{fh} \end{aligned} \tag{A10}$$

where B_{fh} denotes a positive scalar. Furthermore, by considering (43), we have

$$\begin{aligned} ||T_x \tilde{W}_1^T \phi(u_I)|| &= ||T_x \tilde{q}_1 - T_x \epsilon_1(u_I)|| \\ &\leq ||T_x \tilde{q}_1|| + ||T_x|| ||\epsilon_1(u_I)|| \\ &= B_{fh} + \delta_{t_x} \epsilon_{n1} \end{aligned} \tag{A11}$$

Similarly, from the boundedness of e,$\epsilon_2(u_I)$ and (44), we have

$$||T_{\hat{x}} \tilde{W}_2^T \phi(u_I)|| \leq \delta_e + \delta_{t_{\hat{x}}} \epsilon_{n2} \tag{A12}$$

From (46), the dynamics related to $\tilde{W}_1$ and $\tilde{W}_2$ can be respectively given by

$$\begin{cases} \dot{\tilde{W}}_1 = -\Gamma_1 \phi(u_I) s^T T_x \\ y_1 = T_x \tilde{W}_1^T \phi(u_I) \end{cases} \tag{A13}$$

$$\begin{cases} \dot{\tilde{W}}_2 = -\Gamma_2 \phi(u_I)(s+e)^T T_{\hat{x}} \\ y_2 = T_{\hat{x}} \tilde{W}_2^T \phi(u_I) \end{cases} \tag{A14}$$

where y_1 and y_2 denote the outputs of two systems and are both bounded following (A11) and (A12).

Thus, the vector dynamics of the two systems can be given as

$$\begin{cases} \frac{d}{dt} vec(\tilde{W}_1) = -(I \otimes \Gamma_1 \phi(u_I)) T_x^T s = B_{p1}(t) s \\ y_1 = T_x (I \otimes \phi(u_I)^T) vec(\tilde{W}_1) = C_{p1}(t) vec(\tilde{W}_1) \end{cases} \tag{A15}$$

$$\begin{cases} \frac{d}{dt} vec(\tilde{W}_2) = -(I \otimes \Gamma_2 \phi(u_I)) T_{\hat{x}}^T (s + k_e e) \\ = B_{p2}(t)(s + k_e e)) \\ y_1 = T_{\hat{x}} (I \otimes \phi(u_I)^T) vec(\tilde{W}_2) = C_{p2}(t) vec(\tilde{W}_2) \end{cases} \tag{A16}$$

where $B_{p1}(t) = -(I \otimes \Gamma_1 \phi(u_I))T_x^T$ and $B_{p2} = -(I \otimes \Gamma_2 \phi(u_I))T_{\hat{x}}^T$ would be bounded with ensuring the boundedness of $\phi(u_I)$ and $T_x, T_{\hat{x}}$. Thus, from Lemma 4.2.1 in [29], if (47) is satisfied, the boundedness of y_1 y_2 as well as those of s and $s + k_e e$ assures the boundedness of $\tilde{W}_1$, $\tilde{W}_2$, that is, there exist two positive scalars $\delta_{\tilde{W}_1}$, $\delta_{\tilde{W}_2}$ such that $||\tilde{W}_1|| \leq \delta_{\tilde{W}_1}$, $||\tilde{W}_2|| \leq \delta_{\tilde{W}_2}$. Thus, $\tilde{W}_1$ and $\tilde{W}_2$ would be UUB. □

References

1. Frigon, A.; Akay, T.; Prilutsky, B.I. Control of Mammalian Locomotion by Somatosensory Feedback. *Compr. Physiol.* **2021**, *12*, 2877–2947. [CrossRef] [PubMed]
2. Li, Y.; Tee, K.P.; Yan, R.; Chan, W.L.; Wu, Y. A framework of human–robot coordination based on game theory and policy iteration. *IEEE Trans. Robot.* **2016**, *32*, 1408–1418. [CrossRef]
3. Ziebart, B.D.; Maas, A.L.; Bagnell, J.A.; Dey, A.K. Human Behavior Modeling with Maximum Entropy Inverse Optimal Control. In Proceedings of the AAAI Spring Symposium: Human Behavior Modeling, Stanford, CA, USA, 23–25 March 2009; Volume 92.
4. Berret, B.; Chiovetto, E.; Nori, F.; Pozzo, T. Evidence for composite cost functions in arm movement planning: An inverse optimal control approach. *PLoS Comput. Biol.* **2011**, *7*, e1002183. [CrossRef]
5. El-Hussieny, H.; Abouelsoud, A.; Assal, S.F.; Megahed, S.M. Adaptive learning of human motor behaviors: An evolving inverse optimal control approach. *Eng. Appl. Artif. Intell.* **2016**, *50*, 115–124. [CrossRef]
6. Jin, W.; Kulić, D.; Mou, S.; Hirche, S. Inverse optimal control from incomplete trajectory observations. *Int. J. Robot. Res.* **2021**, *40*, 848–865. [CrossRef]
7. Kalman, R.E. When is a linear control system optimal? *J. Fluids Eng.* **1964**, *86*, 51–60. [CrossRef]
8. Molinari, B. The stable regulator problem and its inverse. *IEEE Trans. Autom. Control* **1973**, *18*, 454–459. [CrossRef]
9. Obermayer, R.; Muckler, F.A. *On the Inverse Optimal Control Problem in Manual Control Systems*; NASA: Washington, DC, USA, 1965; Volume 208.
10. Boyd, S.; El Ghaoui, L.; Feron, E.; Balakrishnan, V. *Linear Matrix Inequalities in System and Control Theory*; SIAM: Philadelphia, PA, USA, 1994.
11. Priess, M.C.; Conway, R.; Choi, J.; Popovich, J.M.; Radcliffe, C. Solutions to the inverse LQR problem with application to biological systems analysis. *IEEE Trans. Control Syst. Technol.* **2014**, *23*, 770–777. [CrossRef] [PubMed]
12. Rodriguez, A.; Ortega, R. Adaptive stabilization of nonlinear systems: The non-feedback linearizable case. *IFAC Proc. Vol.* **1990**, *23*, 303–306. [CrossRef]
13. Freeman, R.A.; Kokotovic, P.V. Inverse optimality in robust stabilization. *SIAM J. Control Optim.* **1996**, *34*, 1365–1391. [CrossRef]
14. Chan, T.C.; Mahmood, R.; Zhu, I.Y. Inverse optimization: Theory and applications. *Oper. Res.* **2023** . . [CrossRef]
15. Cao, S.; Luo, Z.; Quan, C. Sequential Inverse Optimal Control of Discrete-Time Systems. *IEEE/CAA J. Autom. Sin.* **2024**, *11*, 1–14. [CrossRef]
16. Tomasi, M.; Artoni, A. Identification of motor control objectives in human locomotion via multi-objective inverse optimal control. *J. Comput. Nonlinear Dyn.* **2023**, *18*, 051004. [CrossRef]
17. Jean, F.; Maslovskaya, S. Injectivity of the inverse optimal control problem for control-affine systems. In Proceedings of the 2019 IEEE 58th Conference on Decision and Control (CDC), Nice, France, 11–13 December 2019; pp. 511–516.
18. Dewhurst, J. A Collage-Based Approach to Inverse Optimal Control Problems with Unique Solutions. Ph.D. Thesis, University of Guelph, Guelph, ON, Canada, 2021.
19. Johnson, M.; Aghasadeghi, N.; Bretl, T. Inverse optimal control for deterministic continuous-time nonlinear systems. In Proceedings of the 52nd IEEE Conference on Decision and Control, Firenze, Italy, 10–13 December 2013; pp. 2906–2913.
20. 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.
21. Ziebart, B.D.; Maas, A.L.; Bagnell, J.A.; Dey, A.K. Maximum entropy inverse reinforcement learning. In Proceedings of the Aaai, Chicago, IL, USA, 13–17 July 2008; Volume 8, pp. 1433–1438.
22. Molloy, T.L.; Ford, J.J.; Perez, T. Online inverse optimal control for control-constrained discrete-time systems on finite and infinite horizons. *Automatica* **2020**, *120*, 109109. [CrossRef]
23. Gupta, R.; Zhang, Q. Decomposition and Adaptive Sampling for Data-Driven Inverse Linear Optimization. *INFORMS J. Comput.* **2022**, *34*, 2720–2735. [CrossRef]
24. Jin, W.; Kulić, D.; Lin, J.F.S.; Mou, S.; Hirche, S. Inverse optimal control for multiphase cost functions. *IEEE Trans. Robot.* **2019**, *35*, 1387–1398. [CrossRef]
25. Athans, M.; Falb, P.L. *Optimal Control: An Introduction to the Theory and Its Applications*; Courier Corporation: Chelmsford, MA, USA, 2007.
26. 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]
27. Li, Y.; Yao, Y.; Hu, X. Continuous-time inverse quadratic optimal control problem. *Automatica* **2020**, *117*, 108977. [CrossRef]

28. Zhang, H.; Ringh, A. Inverse linear-quadratic discrete-time finite-horizon optimal control for indistinguishable homogeneous agents: A convex optimization approach. *Automatica* **2023**, *148*, 110758. [CrossRef]
29. Lewis, F.; Jagannathan, S.; Yesildirak, A. *Neural Network Control of Robot Manipulators and Non-Linear Systems*; CRC Press: Boca Raton, FL, USA, 2020.