TRIANGLE INEQUALITY FOR INVERSE OPTIMAL CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Inverse optimal control (IOC) is a problem of estimating a cost function based on the behaviors of an expert that behaves optimally with respect to the cost function. Although the Hamilton-Jacobi-Bellman (HJB) equation for the value function that evaluates the temporal integral of the cost function provides a necessary condition for the optimality of expert behaviors, the use of the HJB equation alone is insufficient for solving the IOC problem. In this study, we propose a triangle inequality which is useful for estimating the better representation of the value function, along with a new IOC method incorporating the triangle inequality. Through several IOC problems and imitation learning problems of time-dependent control behaviors, we show that our IOC method performs substantially better than an existing IOC method. Showing our IOC method is also applicable to an imitation of expert control of a 2-link manipulator, we demonstrate applicability of our method to real-world problems.

1 INTRODUCTION

The optimal control problem (OCP) is the problem of finding optimal controls which minimize a given objective function in a dynamical system, mostly in a continuous space of state, control signal (action), and time (Kirk, 2004). This is advantageous over reinforcement learning (RL) because RL targets Markov decision processes (MDPs), which are mostly defined over discrete-time domains (Sutton & Barto, 2018). Inverse optimal control (IOC), which is an inverse problem of OCP (Pauwels et al., 2014a), inherits the above advantage over the RL counterpart, inverse reinforcement learning (IRL) (Ng et al., 2000). That is, IOC can estimate a cost function based on the observation of expert behaviors, assuming that the expert has performed optimally based on the cost function. Note that the cost function that the expert would have used can be time-dependent in OCP (Kirk, 2004; Adida & Perakis, 2007); however, there have only been few applications of IOC methods to time-dependent problems. Interesting applications of inverse approaches such as IOC and IRL lie in imitation learning (Hussein et al., 2017). By solving an OCP with the cost function estimated by IOC, the optimal controlling behaviors demonstrated by the expert can be imitated. The setting in which an imitator solves an RL problem with the reward function estimated by the IRL is called apprenticeship learning (Abbeel & Ng, 2004). In these imitation methods, designing complex reward/cost functions can be avoided by performing optimal behaviors based on the estimated reward/cost function (Boularias et al., 2011). Imitation learning that incorporates the inverse approach can be further advantageous over other imitation learning methods such as behavior cloning (Bojarski et al., 2016; Pomerleau, 1988). Since the estimated cost function generalizes the expert objective, imitation learning with the inverse approach is effective even when the expert and imitator are in different environments (i.e., different system dynamics) (Boularias et al., 2011; Fu et al., 2018).

However, solving an IOC problem is difficult in high-dimensional domains. The value function, which evaluates the temporal integral of the cost function, is helpful in estimating the cost function. The Hamilton-Jacobi-Bellman (HJB) equation for the value function provides the necessary condition for the optimality of expert behaviors. However, the use of the HJB equation alone is insufficient for solving IOC well because the HJB equation presents the local optimality just around the expert behaviors; thus, it does not provide global optimality. In this study, we propose the use of a triangle inequality that presents the non-optimality of any bypath that goes through a via-point on a non-optimal trajectory. Because this inequality provides additional information about the value function, its use in IOC can improve the IOC solution by mitigating the ill-posedness possessed by the inverse problem (i.e., expert demonstrations are consistent with multiple cost and value functions). Although the idea of triangle inequality can be applied to general IOC problems, we show several applications to time-dependent IOC problems; that is, the underlying cost function that the expert has used is dependent on time. Time-dependent tasks can often be found in the real world; therefore, well-established modern control methods, such as model predictive control, have been applied to time-dependent tasks (Kirk, 2004; Oldewurtel et al., 2012). Considering this demand, we also demonstrate the application of our new IOC method to the imitation learning of time-dependent tasks.

In existing studies on IOC and IRL, the value and cost functions have been approximated in various forms, such as neural networks (Zou et al., 2018), linear combinations of features such as Gaussian RBFs (Self et al., 2019; Kamalapurkar, 2018; Dvijotham & Todorov, 2010), and polynomials (Pauwels et al., 2014a;b). In this study, we present a constrained linear-programming-based algorithm with polynomial approximation assumptions for the value and cost functions. Because this algorithm does not rely on a stochastic approximation, the implementation of the triagle inequality is straightforward.

2 Background

2.1 Optimal control theory

In optimal control theory, the value function $v(\mathbf{x})$, which represents the minimum total cost when moving from an arbitrary state $\mathbf{x}$ to a state in the terminal state set X_T, plays a central role. An OCP is a problem for obtaining optimal control sequence $\mathbf{u}(\cdot)$ that achieves the value function $v(\mathbf{x}_0)$ from a given initial state $\mathbf{x}_0$.

$$\begin{aligned} v(\mathbf{x}) = \min_{\mathbf{u}(\cdot),T(\geq t_0)} \int_{t_0}^{T} l(\mathbf{x}(\tau),\mathbf{u}(\tau))d\tau \\ s.t.\ \ \dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u}), \mathbf{x}(t_0) = \mathbf{x},\ \mathbf{x}(T) \in X_T \end{aligned} \tag{1}$$

Integrand l is the cost function that represents a scalar cost for a pair of state $\mathbf{x}$ and control $\mathbf{u}$. The value function $v(\mathbf{x})$ denotes the minimum cost integrated from the initial time t_0 to the terminal time T under the following three constraints: $\dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u})$ is the system dynamics, which is assumed to be known throughout this study, $\mathbf{x}(t_0) = \mathbf{x}$ is the initial condition, and $\mathbf{x}(T) \in X_T$ is the terminal condition.

Using the value function, we can obtain the HJB equation, which is a necessary condition for optimal control $\mathbf{u}$ at any state $\mathbf{x}$.

$$0 = \min_{\mathbf{u}} \left\{ l(\mathbf{x},\mathbf{u}) + \frac{\partial v}{\partial \mathbf{x}}^T(\mathbf{x}) f(\mathbf{x},\mathbf{u}) \right\} \tag{2}$$

Relaxing the HJB equation yields the following inequality:

$$\mathcal{L}(l,v)(\mathbf{x},\mathbf{u}) := l(\mathbf{x},\mathbf{u}) + \frac{\partial v}{\partial \mathbf{x}}^T(\mathbf{x}) f(\mathbf{x},\mathbf{u}) \geq 0 \tag{3}$$

This implies that given a cost function $l(\mathbf{x},\mathbf{u})$ and a value function $v(\mathbf{x})$, $\mathcal{L}(l,v)$ should not be negative for any pair of state $\mathbf{x}$ and control $\mathbf{u}$. The equality in Equation (3) holds only when the control $\mathbf{u}$ is optimal at the state $\mathbf{x}$.

2.2 Inverse optimal control with polynomial optimization

IOC is an inverse problem of OCP, which estimates the cost function given a trajectory of the optimal control $(\mathbf{x}_0,\mathbf{u}_0,t_0),...,(\mathbf{x}_{n-1},\mathbf{u}_{n-1},t_{n-1})$. Here, we explain the linear-programming-based IOC method presented by Pauwels et al. (2014a), which was used as a baseline method in this study. Although the authors did not show applications to time-dependent IOC problems, their method could address time-dependent cost and value functions. For convenience, we describe a simplified version of Pauwels' method in which the cost and value functions are assumed to be independent of time. This baseline IOC method estimates both the cost and value functions by optimizing the coefficients of the polynomial function approximators, which are designed to have all monomial bases up to the degree given as a hyperparameter. This method assumes that the dynamics of a system $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ is given by a polynomial vector, and the domains of the state and control space X and U, are compact basic semi-algebraic sets of the form $X = \{\mathbf{x} | g_i(\mathbf{x}) \geq 0, i = 1, ..., m\}, U = \{\mathbf{u} | k_j(\mathbf{u}) \geq 0, j = 1, ..., l\}$ with $g_i (i = 1, ..., m)$ and $k_j (j = 1, ..., l)$ being polynomials of $\mathbf{x}$ and $\mathbf{u}$, respectively.

IOC is a problem of estimating the hidden cost function of an optimally behaving expert given the trajectory of the expert, $(\mathbf{x}_0, \mathbf{u}_0, t_0), ..., (\mathbf{x}_{n-1}, \mathbf{u}_{n-1}, t_{n-1})$. In the baseline method, the cost function is recovered by solving the following constrained optimization problem:

$$\begin{aligned}
\inf_{l,v,\epsilon} \quad & \epsilon + \lambda \|l\|_1 && \text{(4a)} \\
\text{s.t.} \quad & \mathcal{L}(l, v)(\mathbf{x}, \mathbf{u}) \geq 0, \forall (\mathbf{x}, \mathbf{u}) \in X \times U && \text{(4b)} \\
& \frac{1}{n} \sum_{i=0}^{n-1} \mathcal{L}(l, v)(\mathbf{x}_i, \mathbf{u}_i) \leq \epsilon && \text{(4c)} \\
& v(\mathbf{x}) = 0, \forall \mathbf{x} \in X_T && \text{(4d)} \\
& \mathcal{A}(\mathcal{L}(l, v)) = 1 && \text{(4e)}
\end{aligned}$$

Equation (3) leads to Equations (4b) and (4c), and Equation (4c) is an epsilon relaxation of the equality condition. Equation (4d) requires the value function in the terminal state to be zero. In Equation (4e), $\mathcal{A}$ is a linear functional constraint for preventing the HJB function $\mathcal{L}$ from becoming a trivial function, such as the zero function; in our implementation the coefficient summation of the polynomial $\mathcal{L}(l, v)$ is restricted to unity. Equation (4a) attempts to minimize the slack variable ϵ plus L1-based regularizer of the coefficients of the cost function l; $\lambda > 0$ is a hyperparameter that controls the strength of the regularizer. $\inf_{l,v,\epsilon}$ indicates that this optimization problem is minimized by optimizing the coefficients of the polynomial function approximators for the cost l and value v functions, and the slack variable ϵ.

3 Inverse optimal control with triangle inequality

Although Pauwels' method is simple and widely applicable, its solution would not necessarily be good because of the shortage of constraints; the HJB equation only imposes constraints on the derivative of the value function, and represents the optimality condition of the expert behaviors only around the expert trajectory. To address this constraint shortage problem, we present a new IOC method based on the triangle inequality. Section 3.1 introduces the triangle inequality that should exist behind the expert optimal trajectory. Section 3.2 describes the IOC method incorporating the triangle inequality.

3.1 Triangle inequality

Here, we derive the triangle inequality in a simple time-independent setting, in which the cost and value functions taken by the expert are independent of time; however, its extension to address time-dependent settings is straightforward. The triangle inequality and IOC method for time-dependent settings are described in Appendix A.

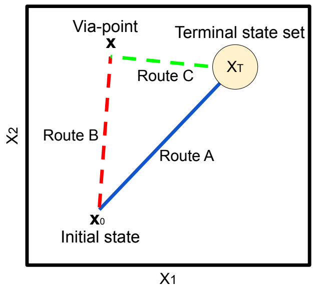


Figure 1: Conceptual diagram of triangle inequality

Figure 1 depicts the concept of triangle inequality in an OCP in two-dimensional state space. We assumed that route A is the optimal route with the minimal total cost from the initial state $\mathbf{x}_0$ to any terminal state in set X_T. Route B $\rightarrow$ C is the optimal route when it is constrained to pass through a via-point $\mathbf{x}$ that is not on the optimal route. The triangle inequality states that the total cost of B $\rightarrow$ C should be larger than that of A for any via-point $\mathbf{x}$.

Because the value function is defined as the minimum total cost to reach any terminal state, the total costs of A and C are given by

$v(\mathbf{x}_0)$ and $v(\mathbf{x})$, respectively. However, the minimal total cost of Route B cannot be represented by the value function. To this end, we introduce an alternative value function in the "time-reversed" OCP, which is given by

$$\begin{aligned} rv(\mathbf{x}) := \min_{\mathbf{u}(\cdot),t(\geq t_0)} \int_t^{t_0} -l(\mathbf{x}(\tau),\mathbf{u}(\tau))d\tau \\ s.t.\ \ \dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u}), \mathbf{x}(t)=\mathbf{x}, \mathbf{x}(t_0)=\mathbf{x}_0 \end{aligned} \tag{5}$$

The left-hand side of Equation (5), called the **reverse value function** in this study, denotes the minimal total *negative* cost from the via-point $\mathbf{x}$ to the initial state $\mathbf{x}_0$ in a backward manner. By reversing the integral interval, the reverse value function $rv(\mathbf{x})$ is shown to be equivalent to the minimum total cost from the initial state $\mathbf{x}_0$ to the arbitrary via-point $\mathbf{x}$. Therefore, the minimal total cost of Route B is given by $rv(\mathbf{x})$.

Accordingly, the triangle inequality is expressed as:

$$v(\mathbf{x}_0) \ = \ \min_{\mathbf{u}(\cdot),T(\geq t_0)} \int_{t_0}^{T} l(\mathbf{x}(\tau),\mathbf{u}(\tau))d\tau\ (\mathbf{x}(T)\in X_T) \tag{6a}$$

$$\leq \ \min_{\mathbf{u}(\cdot),T'(\geq t_0)} \int_{t_0}^{T'} l(\mathbf{x}(\tau),\mathbf{u}(\tau))d\tau\ (\mathbf{x}(T')\in X_T, \exists t\in[t_0,T']\ \mathbf{x}(t)=\mathbf{x}) \tag{6b}$$

$$= \ rv(\mathbf{x}) + v(\mathbf{x}), \tag{6c}$$

where the constraints $\mathbf{x}(t_0)=\mathbf{x}_0$ and $\dot{\mathbf{x}}=f(\mathbf{x},\mathbf{u})$ are omitted for visibility. When the via-point $\mathbf{x}$ is on optimal Route A, the equality in Equation (6) holds. If Route A is the sole optimal route in the state space, the inequality in Equation (6) should hold strictly for any via-point that is not on Route A.

HJB equation for the reverse value function Similar to the value function on the "time-forward" OCP, another HJB Equation (7) also holds for the reverse value function. The equality of this HJB equation is satisfied by the expert trajectory $(\mathbf{x}_0,\mathbf{u}_0,t_0),...,(\mathbf{x}_{n-1},\mathbf{u}_{n-1},t_{n-1})$ because the expert trajectory is optimal in both "time-forward" and "time-reversed" OCPs. The derivation is described in Appendix B.

$$\mathcal{RL}(l,rv)(\mathbf{x},\mathbf{u}) := l(\mathbf{x},\mathbf{u}) - \frac{\partial rv}{\partial \mathbf{x}}^T(\mathbf{x}) f(\mathbf{x},\mathbf{u}) \geq 0 \tag{7}$$

3.2 Implementation of the triangle inequality

In this subsection, we develop our new IOC method with the triangle inequality in time-independent settings. Our IOC method (8) is an extension of the baseline IOC method (4), with the additional constraints from (8b) to (8l), with the following assumptions: 1) the system dynamics is given by a polynomial vector, 2) $\mathbf{x}_{n-1} \in X_T$ is satisfied by the expert trajectory $(\mathbf{x}_0,\mathbf{u}_0,t_0),...,(\mathbf{x}_{n-1},\mathbf{u}_{n-1},t_{n-1})$, and 3) the domains of state and control space are given as compact basic semi-algebraic sets of the form $X=\{\mathbf{x}|g_i(\mathbf{x})\geq 0, i=1,...,m\}, U=\{\mathbf{u}|k_j(\mathbf{u})\geq 0, j=1,...,l\}$ with $g_i(i=1,...,m)$ and $k_j(j=1,...,l)$ being polynomials of $\mathbf{x}$ and $\mathbf{u}$, respectively. Below, we describe how the additional constraints from (8b) to (8l) have been derived.

Constraints (8k) and (8l) originate from the triangle inequality (6). Constraint (8k) requires the triangle inequality to hold for any state $\mathbf{x}$ in domain X. Constraint (8l) is the epsilon relaxation of the equality constraint, which should hold for any state $\mathbf{x}$ on the expert trajectory. The constraints on the reverse value function from (8f) to (8i) correspond to the constraints on the value function from (8b) to (8e), and have been introduced to identify the reverse value function using constrained linear programming. Furthermore, Equation (8j) should be satisfied because $rv(\mathbf{x}_{n-1})$ and $v(\mathbf{x}_0)$ share the same optimal trajectory.

Application to multiple trajectories settings Depending on the situation, we can observe multiple expert trajectories each from a different initial state to a state in the shared terminal state set. Our linear-programming-based method with triangle inequality can handle these situations by defining a reverse value function for each expert trajectory. For k expert trajectories, k reverse value functions $\{rv_j\}_{i=1,...k}$ can be defined, each with a different initial state as a terminal condition. Although the constraints from (8b) to (8e) are applied to the single value function, we should duplicate the constraints from (8f) to (8l) to cover multiple reverse value functions. We present the algorithm and some further details in Appendix C.

Implementation details The optimization problem (8) is solved in the same manner as solving the existing problem (4); the latter was solved based on polynomial optimization and linear matrix inequalities (Pauwels et al., 2014a). The inequalities (8f) and (8k) were reduced to linear matrix inequalities because each of them is an inequality over a compact basic semi-algebraic set, as is inequality (4b) in (4). For the optimization, we used the YALMIP toolbox in MATLAB (Lofberg, 2009), which is suitable for handling the polynomial constraints included in (8).

$$\inf_{l,v,rv,\epsilon_a,\epsilon_b,\epsilon_c} \quad \epsilon_a + \epsilon_b + \epsilon_c + \lambda \|l\|_1 \tag{8a}$$

s.t.

-Constraints from the baseline method-

$$\mathcal{L}(l,v)(\mathbf{x},\mathbf{u}) \geq 0, \forall (\mathbf{x},\mathbf{u}) \in X \times U \tag{8b}$$

$$\frac{1}{n}\sum_{i=0}^{n-1} \mathcal{L}(l,v)\left(\mathbf{x}_i,\mathbf{u}_i\right) \leq \epsilon_a \tag{8c}$$

$$v(\mathbf{x}) = 0, \forall \mathbf{x} \in X_T \tag{8d}$$

$$\mathcal{A}(\mathcal{L}(l,v)) = 1 \tag{8e}$$

-Constraints from the introduction of rv-

$$\mathcal{RL}(l,rv)(\mathbf{x},\mathbf{u}) \geq 0, \forall (\mathbf{x},\mathbf{u}) \in X \times U \tag{8f}$$

$$\frac{1}{n}\sum_{i=0}^{n-1} \mathcal{RL}(l,rv)\left(\mathbf{x}_i,\mathbf{u}_i\right) \leq \epsilon_b \tag{8g}$$

$$rv(\mathbf{x}_0) = 0 \tag{8h}$$

$$\mathcal{A}(\mathcal{RL}(l,rv)) = 1 \tag{8i}$$

$$rv(\mathbf{x}_{n-1}) = v(\mathbf{x}_0) \tag{8j}$$

-Constraints from the triangle inequality-

$$v(\mathbf{x}) + rv(\mathbf{x}) \geq v(\mathbf{x}_0), \forall \mathbf{x} \in X \tag{8k}$$

$$\frac{1}{n}\sum_{i=0}^{n-1} \{v(\mathbf{x}_i) + rv(\mathbf{x}_i) - v(\mathbf{x}_0)\} \leq \epsilon_c \tag{8l}$$

4 Experimental results

4.1 Experimental settings

We compared our IOC method with the baseline IOC method by using three types of simple OCP tasks. In particular, the third is in an imitation learning setting when the expert is taking time-dependent behaviors. The code is available at `https://github/AnonymousAuthor/ForDoubleBlindPolicy`.

Hyperparameters The polynomials to approximate l, v, and rv were set to the same degree, assuming that they were of comparable complexities. The regularization parameter λ was set to 10^{-6} regardless of the task. We found that a larger setting of this parameter value likely produced errors similar to those of the baseline method because the associated strong constraints on the coefficients would have made the triangle inequality condition less effective.

Tasks The following three OCPs were solved by the steepest descent method (Kirk, 2004). Although the domains of state and control space in the three OCPs were normalized as $[-1, 1]$ on each coordinate to stabilize the numerical calculation, its enlargement is straightforward.

OCP 1 **Time-independent control** to the origin of the two-dimensional state space with the terminal time T being free, starting from a randomly (uniformly) sampled initial state.

$$l(\mathbf{x},\mathbf{u}) = x_1^2 + x_2^2 + u_1^2 + u_2^2,\ \dot{\mathbf{x}} = \mathbf{u},\ X_T = \{(0,0)\}$$
$$X = \{\mathbf{x} |\ |x_i| \leq 1, i \in \{1,2\}\}, U = \{\mathbf{u} |\ |u_i| \leq 1, i \in \{1,2\}\}$$

OCP 2 **Time-dependent control** to chase a target that moves from $(0,0)$ to $(1,1)$ along time $[0,1]$, starting from a randomly (uniformly) sampled initial state.

$$l(\mathbf{x},\mathbf{u},t) = (x_1 - t)^2 + (x_2 - t)^2 + 0.3(u_1^2 + u_2^2),\ \dot{\mathbf{x}} = 5\mathbf{u},\ X_T = \{(1,1)\}$$
$$X = \{\mathbf{x} |\ 0 \leq x_i \leq 1, i \in \{1,2\}\}, U = \{\mathbf{u} |\ -1 \leq u_i \leq 0.3, i \in \{1,2\}\},\ 0 \leq t \leq 1$$

OCP 3 **Time-dependent non-polynomial control** with an intersecting trajectory, which requires different controls at the same state visited at different times. The generated trajectory will be used as an expert data in Section 4.3 (shown in Figure (3a)).

$$l(\mathbf{x},\mathbf{u},t) = \begin{cases} (x_1 - 7.6t + 1)^2 + (x_2 + 0.9)^2 + 2(u_1^2 + u_2^2) & (0 \leq t < 1/4) \\ (x_1 - 0.9)^2 + (x_2 - 7.2t + 2.7)^2 + 2(u_1^2 + u_2^2) & (1/4 \leq t < 2/4) \\ (x_1 + 7.2t - 4.5)^2 + (x_2 - 0.9)^2 + 2(u_1^2 + u_2^2) & (2/4 \leq t < 3/4) \\ (x_1 + 0.9)^2 + (x_2 + 7.6t - 6.6)^2 + 2(u_1^2 + u_2^2) & (3/4 \leq t \leq 1) \end{cases}$$
$$\dot{\mathbf{x}} = 10\mathbf{u},\ \mathbf{x}(t_0) = (-1, -0.9),\ X_T = \{(-0.9, -1)\}$$
$$X = \{\mathbf{x} |\ |x_i| \leq 1, i \in \{1,2\}\}, U = \{\mathbf{u} |\ |u_i| \leq 1, i \in \{1,2\}\},\ 0 \leq t \leq 1$$

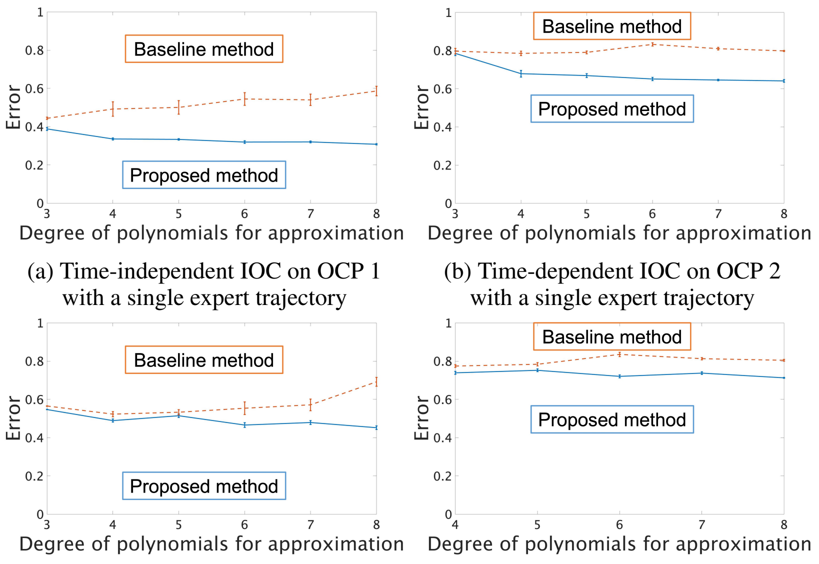
4.2 Accuracy of the estimated cost functions

First, we examined the accuracy of the cost functions estimated by our IOC and the baseline IOC methods in OCP 1 and OCP 2. The experiments were performed in two different situations: one in which a single expert trajectory was given, and the other in which multiple expert trajectories were given. Each experiment was evaluated using various degrees of polynomials for approximating the cost, value and reverse value functions. When applying IOC methods to the optimal (expert) trajectories in OCP 1 and OCP 2, we used time-independent and time-dependent polynomials, respectively. The following error function (9) was used to evaluate the estimated cost functions:

$$Error(l, \hat{l}) := \min_{\alpha} \sqrt{\frac{\int_{t_0}^{T} \int_U \int_X (l(\mathbf{x},\mathbf{u},t) - \alpha \hat{l}(\mathbf{x},\mathbf{u},t))^2 d\mathbf{x} d\mathbf{u} dt}{\int_{t_0}^{T} \int_U \int_X l(\mathbf{x},\mathbf{u},t)^2 d\mathbf{x} d\mathbf{u} dt}} \tag{9}$$

The function $Error(l, \hat{l})$ represents the difference between the correct cost function l and the estimated cost function $\hat{l}$ over the entire domain and was normalized to $0 \leq Error(l, \hat{l}) \leq 1$. Because there was an indeterminacy in the global magnitude of the cost function, we prepared a scalar α to compensate for this. Equation (9) was used to evaluate a time-dependent cost function, whereas the integral over time was eliminated when evaluating a time-independent cost function.

Results Figure 2 shows the accuracies of the cost function in terms of the normalized error (9) averaged over ten trials. In each trial, the expert trajectories were started from different initial states that were randomly sampled from the state space. Figures (2a) and (2b) show the accuracies with a single expert trajectory setting when our IOC and baseline IOC methods were applied to a single trajectory generated by OCP 1 and OCP 2 experts, respectively. Our IOC method exhibited consistently smaller errors than the baseline IOC method in both time-independent and time-dependent settings, particularly when the degree of the approximation polynomials was large. Visualization of the value functions estimated by the two IOC methods also evidenced that the triangle inequality contributed to the better estimation of the value function (Appendix D). The computation time of our IOC method was on average 2.68 times longer than that of the baseline method (Appendix E). Figures (2c) and (2d) show the accuracies with multiple trajectory settings, each applied to three expert trajectories generated by OCP 1 and OCP 2 experts. Our IOC method performed better than the baseline method, even when multiple expert trajectories were given.



(a) Time-independent IOC on OCP 1 with a single expert trajectory

(b) Time-dependent IOC on OCP 2 with a single expert trajectory

(c) Time-independent IOC on OCP 1 with three expert trajectories

(d) Time-dependent IOC on OCP 2 with three expert trajectories

Figure 2: Normalized errors by the two IOC methods for various degrees of approximation polynomials. The lines and error bars denote the mean and standard error over ten runs. In figures (2a) and (2b), IOC performed with a single expert trajectory. In figures (2c) and (2d), IOC performed with 3 expert trajectories. Each expert trajectory started from an initial state randomly sampled from the state space (hence different from the other initial states).

4.3 Imitation learning of time-dependent controls

Here, we applied the baseline and our IOC methods to imitate the expert trajectory for a time-dependent and non-polynomial control problem in OCP 3. In both mimickers, that is, the baseline and our IOC mimicker, we first estimated the cost function based on a single expert trajectory for OCP 3 (Figure 3a), then we solved the OCP for the estimated cost functions. Note that both IOC mimickers utilized an extended time-dependent version of the IOC methods, which are described in Appendix A. Because this is a time-dependent control problem, the control trajectory was intersected in the state space near the origin. In this experiment, the degree of the approximation polynomials and regularization parameter λ were chosen to best mimic the expert data (with respect to the squared error when reproducing the expert trajectory) among the following ranges: polynomial degree $\in \{4, 6, 8\}$ and $\lambda \in \{10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}\}$.

Figures 3b and 3c show the trajectories generated by the baseline and our IOC mimickers, respectively. While the two mimickers could reproduce circular motions in the state space, our IOC mimicker could imitate them more accurately than the baseline mimicker, demonstrating the usefulness of the proposed method even in the scenario of imitation learning.

As an additional experiment, we further examined the control generalization of imitation learning by the two IOC mimickers. As in the previous experiment, the OCP behaviors with the estimated cost functions were compared with the OCP behaviors with the correct cost function, but the initial states of these OCPs were different from that of the expert trajectory in Figure 3a. This setup is to examine the generalization capability of the IOC mimickers. The experiment was performed with the hyperparameters that had been shown to be optimal in the previous experiment. Figure 4 shows the imitation behaviors starting from a different initial state, where our IOC mimicker successfully produced a similar trajectory to the optimal trajectory while the baseline mimicker failed. For statistical evaluation, we repeated this imitation ten times starting from ten different initial states taken uniformly from the state space. Table 1 lists the mean squared error and its standard deviation over these ten trials, for the baseline and our IOC mimickers. From this table, we can see that our IOC-based imitation learning could reproduce the expert OCP's behaviors much better than the baseline method, even when initial states were different from the initial state with which the cost function had been estimated.

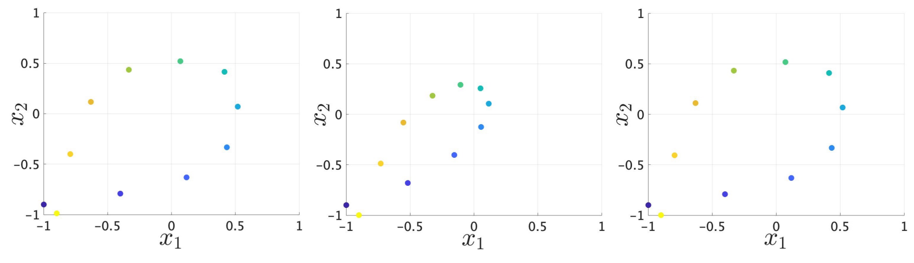

(a) Expert data (b) Baseline method (c) Proposed method

Figure 3: Imitation control by the baseline IOC method (figure (b)) and our IOC method (figure (c)), based on a single expert trajectory (figure (a)) that has solved OCP 3. The squared error of the baseline IOC method was 8.6×10^{-1} with the degree of approximation polynomials being 4 and $\lambda = 10^{-6}$. The squared error of the baseline IOC method was 2.5×10^{-4} with the degree of the approximation polynomials being 8 and $\lambda = 10^{-8}$. The gradual change in color from blue (dark) to yellow (light) indicates the progress of time.

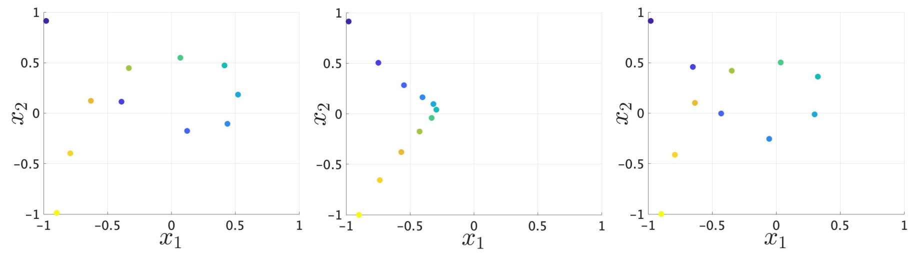

(a) Optimal trajectory (b) Baseline method (c) Proposed method

Figure 4: Imitation control starting at a different initial state $(-0.98, 0.91)$ from that $(-1.00, -0.90)$ of the original expert trajectory. Figures (a), (b), and (c) show the OCP trajectories with the correct cost function, with the cost function estimated by the baseline IOC method (i.e., baseline IOC mimicker), and with the cost function estimated by our IOC method (i.e., our IOC mimicker), respectively.

Table 1: Mean squared errors of imitation trajectories starting from different initial states

Baseline method	Proposed method
3.634±0.571	0.521±0.096

4.4 Imitation learning in a complex dynamics

We have so far assumed linear (or polynomial) dynamics, which is a requirement for utilizing our IOC algorithm based on polynomial optimization. To show further applicability to nonpolynomial dynamics, which often arises in real-world applications, here we examined an imitation learning task of control of a 2-link manipulator. Since the dynamics is represented as a complicated differential equation including trigonometric functions, we applied our IOC method to the approximated dynamics with the Taylor expansion up to the second order.

In this task, the state space was four-dimensional as $\mathbf{x} = [\theta_1, \theta_2, \dot{\theta_1}, \dot{\theta_2}]$ and the control space was two-dimensional as $\mathbf{u} = (\tau_1, \tau_2)$. Here, θ_i and τ_i are the angle and torque of the ith joint, respectively. Each coordinate of the state and control space was rescaled to be within the interval $[-1, 1]$ to avoid possible numerical instability. The upper and lower rows in Figure 5 visualize a series of expert and imitation behaviors, respectively. Our IOC method could well imitate back-and-forth control demonstrated by the expert, even when the dynamics was nonpolynomial and hence approximated by the Taylor expansion. Further discussion and details are provided in Appendix F.

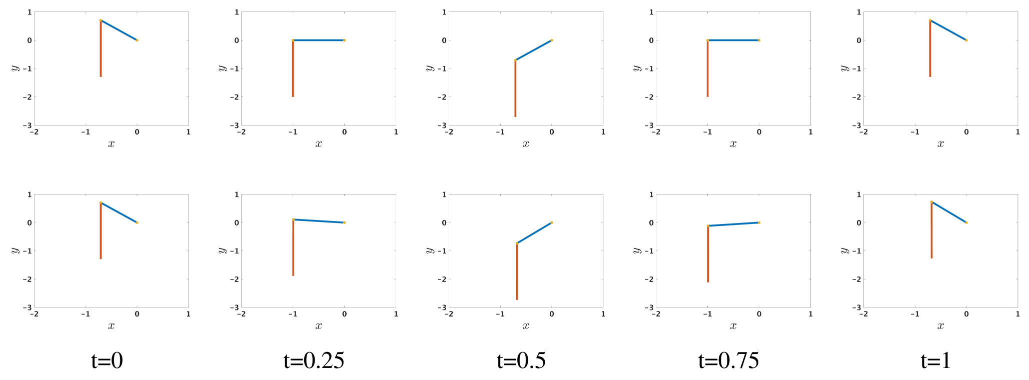


Figure 5: Imitation learning of control of a 2-link robot. The upper and lower figures display expert and imitation behaviors, respectively. Each column is a snapshot of the behaviors at a certain time.

5 Discussion

In this study, we presented a new IOC method based on the newly introduced triangle inequality. Although the HJB equation is a well-established condition for optimality in control theory, it is insufficient for effectively solving the inverse problem in IOC. In our experiments, we found that the triangle inequality was effective in estimating the better representation of the value function (Appendix D), thus improving the accuracy of the cost function estimated by the value-function-based IOC method (Section 4.2). We also found that the improved IOC method was preferable in the imitation learning scenario (Section 4.3). Our IOC-based mimicker imitated expert control well, even starting from an initial state that was different from that of the expert trajectory. Moreover, our imitation learning method worked even in a time-dependent OCP setting. We believe that this improved performance in the imitation of optimal controls would enlarge the application domains of the OCP and IOC.

Because our formulation is based on the optimal control theory, which assumes that the system dynamics are deterministic and known, an extension to the situations where the system is stochastic and/or unknown requires some additional devices. One possible direction in this regard would be to introduce spatial constraints similar to the triangle inequality to IRL. In many IRL formulations, we maximize the likelihood or posterior probability based on the gradient optimization method (Ziebart et al., 2008; Choi & Kim, 2011). Our IOC method with the triangle inequality might be extended in a similar way by seeking the zero point of the stochastically identified Bellman equation. However, the extension of our method to this IRL formulation has a disadvantage in that it has to dispose of the continuous action/state space and continuous time assumptions employed in optimal control theory. Moreover, in principle, conventional RL formulations based on MDPs cannot handle time-dependent cost functions. The path-integral-based RL formulation may become the foundation for extending our method to stochastic environments without losing continuous space and time-dependent cost function assumptions (Theodorou et al., 2010). Moreover, in this study, the expert trajectory was assumed to be optimal, which may not be satisfied in many practical applications. Although we showed the robustness of our method against observation noise in the expert trajectory (Appendix G), to handle non-optimal trajectories in a principled manner, it is necessary to extend our triangle inequality to weakened constraints, as in (Tschiatschek et al., 2019).

In this study, we relied on the existing software for constrained linear programming (Lofberg, 2009), which restricted our tasks to being relatively low-dimensional. Applications to high-dimensional tasks can be realized using kernel methods (Levine et al., 2011) or nonlinear neural networks (Wulfmeier et al., 2015). Because these function approximators are often optimized based on a (stochastic) gradient descent method to seek a zero-point of the Monte Carlo-based gradient function, we can utilize these function approximators by extending our constrained optimization problem to a gradient-based method.

Ethics Statement

Our study focuses on the theoretical aspects of IOC, thus there is little or no threats to health, safety, personal security, and privacy. However, as a potential future threat to safety, we discuss as follows:

When applying our IOC method to a real-world problem such as the imitation of professional drivers or plant operators, the evaluation of the estimated cost function is difficult because of the absence of the correct cost function. On the other hand, the estimated cost function includes errors, as shown in Section 4.2, causing potentially undesirable behaviors. Therefore, appropriate evaluation methods will be required to satisfy the high demand for safety.

Reproducibility Statement

All the codes are available at https://github/AnonymousAuthor/ForDoubleBlindPolicy. All the expert data used in this study were generated by the codes. The omitted explanations and derivations of our new IOC method are described in Appendix A,B,C.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine learning*, pp. 1, 2004.

Elodie Adida and Georgia Perakis. A nonlinear continuous time optimal control model of dynamic pricing and inventory control with no backorders. *Naval Research Logistics*, 54(7):767–795, 2007.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 182–189, 2011.

Jaedeug Choi and Kee-Eung Kim. Map inference for bayesian inverse reinforcement learning. *Advances in Neural Information Processing Systems*, 24, 2011.

Krishnamurthy Dvijotham and Emanuel Todorov. Inverse optimal control with linearly-solvable mdps. In *International Conference on Machine learning*, 2010.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys*, 50(2):1–35, 2017.

Rushikesh Kamalapurkar. Linear inverse reinforcement learning in continuous time and space. In *Annual American Control Conference*, pp. 1683–1688, 2018.

Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004.

Amit Kumar, Shrey Kasera, and L. B. Prasad. Optimal control of 2-link underactuated robot manipulator. In *International Conference on Innovations in Information, Embedded and Communication Systems*, pp. 1–6, 2017.

Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. *Advances in Neural Information Processing Systems*, 24, 2011.

Johan Lofberg. Pre- and post-processing sum-of-squares programs in practice. *IEEE Transactions on Automatic Control*, 54(5):1007–1011, 2009.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *International Conference on Machine learning*, volume 1, pp. 2, 2000.

Frauke Oldewurtel, Alessandra Parisio, Colin N. Jones, Dimitrios Gyalistras, Markus Gwerder, Vanessa Stauch, Beat Lehmann, and Manfred Morari. Use of model predictive control and weather forecasts for energy efficient building climate control. *Energy and Buildings*, 45:15–27, 2012.

Edouard Pauwels, Didier Henrion, and Jean-Bernard Lasserre. Inverse optimal control with polynomial optimization. In *IEEE Conference on Decision and Control*, pp. 5581–5586, 2014a.

Edouard Pauwels, Didier Henrion, and Jean-Bernard Lasserre. Linear conic optimization for inverse optimal control. *SIAM Journal on Control and Optimization*, 54, 2014b.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems*, 1, 1988.

Ryan Self, Michael Harlan, and Rushikesh Kamalapurkar. Online inverse reinforcement learning for nonlinear systems. In *IEEE Conference on Control Technology and Applications*, pp. 296–301, 2019.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 11:3137–3181, 2010.

Sebastian Tschiatschek, Ahana Ghosh, Luis Haug, Rati Devidze, and Adish Singla. Learner-aware teaching: Inverse reinforcement learning with preferences and constraints. *Advances in Neural Information Processing Systems*, 32, 2019.

Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, volume 8, pp. 1433–1438, 2008.

QiJie Zou, Haoyu Li, and Rubo Zhang. Inverse reinforcement learning via neural network in driver behavior modeling. In *IEEE Intelligent Vehicles Symposium*, pp. 1245–1250, 2018.

A Proposed method for time-dependent settings

In this section, we present the triangle inequality and an associated Inverse Optimal Control (IOC) method in time-dependent settings. Here, we assume that the expert trajectory $(\mathbf{x}_0, \mathbf{u}_0, t_0)$,.. ,$(\mathbf{x}_{n-1}, \mathbf{u}_{n-1}, t_{n-1}(= T))$ from the initial state $\mathbf{x}_0$ and time t_0 is observed, where T is the fixed terminal time and $\mathbf{x}_{n-1}$ is in the terminal state set, $\mathbf{x}_{n-1} \in X_T$. We also assume that the dynamics $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$ is given as a polynomial vector, and the domains of the state and control space, X and U, are given as compact basic semi-algebraic sets of the form $X = \{\mathbf{x}|g_i(\mathbf{x}) \geq 0, i = 1, ..., m\}, U = \{\mathbf{u}|k_j(\mathbf{u}) \geq 0, j = 1, ..., l\}$.

The objective of a time-dependent optimal control problem is to find the control sequence $\mathbf{u}(\cdot)$ that achieves the value function $v(t_0, \mathbf{x}_0)$ from the initial state $\mathbf{x}_0$ at the initial time t_0. This time-dependent value function is defined using the time-dependent cost function $l(\mathbf{x}, \mathbf{u}, t)$ and the fixed terminal time T. Arguments $t, \mathbf{x}$ should satisfy $t_0 \leq t \leq T, \mathbf{x} \in X$. The dynamics $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$ is allowed to be time-dependent.

$$\begin{aligned} v(t, \mathbf{x}) = \min_{\mathbf{u}(\cdot)} \int_t^T l(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) d\tau \\ \text{s.t. } \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t), \mathbf{x}(t) = \mathbf{x},\ \mathbf{x}(T) \in X_T \end{aligned} \tag{10}$$

The following HJB equation is derived from the value function (10).

$$-\frac{\partial v}{\partial t}(\mathbf{x}, t) = \min_{\mathbf{u}} \{ l(\mathbf{x}, \mathbf{u}, t) + \frac{\partial v}{\partial \mathbf{x}}^T (\mathbf{x}, t) f(\mathbf{x}, \mathbf{u}, t) \} \tag{11}$$

This HJB equation (11) is converted into the following inequality:

$$\mathcal{L}(l,v)(\mathbf{x},\mathbf{u},t) := l(\mathbf{x},\mathbf{u},t) + \frac{\partial v}{\partial \mathbf{x}}^T(\mathbf{x},t) f(\mathbf{x},\mathbf{u},t) + \frac{\partial v}{\partial t}(\mathbf{x},t) \geq 0 \tag{12}$$

Given the expert trajectory $(\mathbf{x}_0, \mathbf{u}_0, t_0), ..., (\mathbf{x}_{n-1}, \mathbf{u}_{n-1}, t_{n-1})$, the baseline method (Pauwels et al., 2014a), which can deal with time-dependent problems in principle, recovers the cost function using the following minimization problem (13):

$$\begin{aligned}
&\inf_{l,v,\epsilon} \epsilon + \lambda \|l\|_1 && \text{(13a)}\\
&\text{s.t.} \quad \mathcal{L}(l,v)(\mathbf{x},\mathbf{u},t) \geq 0, \forall (\mathbf{x},\mathbf{u},t) \in X \times U \times [t_0, T] && \text{(13b)}\\
&\qquad \frac{1}{n}\sum_{i=0}^{n-1} \mathcal{L}(l,v)\left(\mathbf{x}_i,\mathbf{u}_i,t_i\right) \leq \epsilon && \text{(13c)}\\
&\qquad v(T,\mathbf{x}) = 0, \forall \mathbf{x} \in X_T && \text{(13d)}\\
&\qquad \mathcal{A}(\mathcal{L}(l,v)) = 1 && \text{(13e)}
\end{aligned}$$

To derive the triangle inequality, the following time-dependent reverse value function rv is introduced. Arguments $t, \mathbf{x}$ must satisfy $t_0 \leq t \leq T, \mathbf{x} \in X$.

$$\begin{aligned}
rv(t,\mathbf{x}) := \min_{\mathbf{u}(\cdot)} \int_t^{t_0} -l(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau) d\tau \\
\text{s.t. } \dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u},t), \mathbf{x}(t) = \mathbf{x}, \mathbf{x}(t_0) = \mathbf{x}_0
\end{aligned} \tag{14}$$

Using this reverse value function, the triangle inequality can be derived as follows (constraints $\mathbf{x}(t_0) = \mathbf{x}_0, \dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u},t)$ are omitted from parentheses for simplicity):

$$\begin{aligned}
v(t_0,\mathbf{x}_0) &= \min_{\mathbf{u}(\cdot)} \int_{t_0}^{T} l(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau) d\tau \; (\mathbf{x}(T) \in X_T) && \text{(15a)}\\
&\leq \min_{\mathbf{u}(\cdot)} \int_{t_0}^{T} l(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau) d\tau \; (\mathbf{x}(T) \in X_T, \mathbf{x}(t) = \mathbf{x}) && \text{(15b)}\\
&= rv(t,\mathbf{x}) + v(t,\mathbf{x}) && \text{(15c)}
\end{aligned}$$

Equation (15a) represents the total cost of the optimal route from the initial state $\mathbf{x}_0$ and time t_0. Equation (15b) indicates that the total cost is greater than or equal to Equation (15a) when there is an additional constraint to pass through a via-point $\mathbf{x}$ at time t. Dividing the integral into two terms separated at time t, i.e., $rv(t,\mathbf{x})$ and $v(t,\mathbf{x})$, we have Equation (15c). The equality of this triangle inequality (15) is satisfied when pair $(t,\mathbf{x})$ is on the optimal trajectory.

Moreover, as derived in Appendix B, the HJB equation for the reverse value function is expressed as an inequality:

$$\mathcal{RL}(l,rv)(\mathbf{x},\mathbf{u},t) := l(\mathbf{x},\mathbf{u},t) - \frac{\partial rv}{\partial \mathbf{x}}^T(\mathbf{x},t) f(\mathbf{x},\mathbf{u},t) - \frac{\partial rv}{\partial t}(\mathbf{x},t) \geq 0 \tag{16}$$

Using the derived inequalities in Equations (15) and (16), the constraints from (17f) to (17l) can be derived in the same manner as for the time-independent case. All these derived constraints yield the following constrained optimization problem:

$$\inf_{l,v,rv,\epsilon_a,\epsilon_b,\epsilon_c} \quad \epsilon_a+\epsilon_b+\epsilon_c+\lambda\|l\|_1 \tag{17a}$$

s.t.

-Constraints from the baseline method-

$$\mathcal{L}(l,v)(\mathbf{x},\mathbf{u},t)\geq 0,\forall(\mathbf{x},\mathbf{u},t)\in X\times U\times[t_0,T] \tag{17b}$$

$$\tfrac{1}{n}\textstyle\sum_{i=0}^{n-1}\mathcal{L}(l,v)\left(\mathbf{x}_i,\mathbf{u}_i,t_i\right)\leq\epsilon_a \tag{17c}$$

$$v(T,\mathbf{x})=0,\forall\mathbf{x}\in X_T \tag{17d}$$

$$\mathcal{A}(\mathcal{L}(l,v))=1 \tag{17e}$$

-Constraints from the introduction of rv-

$$\mathcal{RL}(l,rv)(\mathbf{x},\mathbf{u},t)\geq 0,\forall(\mathbf{x},\mathbf{u},t)\in X\times U\times[t_0,T] \tag{17f}$$

$$\tfrac{1}{n}\textstyle\sum_{i=0}^{n-1}\mathcal{RL}(l,rv)\left(\mathbf{x}_i,\mathbf{u}_i,t_i\right)\leq\epsilon_b \tag{17g}$$

$$rv(t_0,\mathbf{x}_0)=0 \tag{17h}$$

$$\mathcal{A}(\mathcal{RL}(l,rv))=1 \tag{17i}$$

$$rv(T,\mathbf{x}_{n-1})=v(t_0,\mathbf{x}_0) \tag{17j}$$

-Constraints from the triangle inequality-

$$v(t,\mathbf{x})+rv(t,\mathbf{x})\geq v(t_0,\mathbf{x}_0),\forall(t,\mathbf{x})\in[t_0,T]\times X \tag{17k}$$

$$\tfrac{1}{n}\textstyle\sum_{i=0}^{n-1}\{v(t_i,\mathbf{x}_i)+rv(t_i,\mathbf{x}_i)-v(t_0,\mathbf{x}_0)\}\leq\epsilon_c \tag{17l}$$

B HAMILTON-JACOBI-BELLMAN (HJB) EQUATION FOR THE REVERSE VALUE FUNCTION

In this section, we derive the HJB equation for the reverse value function. Considering a small change in time, $\Delta t(>0)$, the reverse value function (14) can be transformed as follows:

$$\begin{aligned}
rv(t,\mathbf{x}) \quad &:= \quad \min_{\mathbf{u}(\cdot)}\int_t^{t_0}-l(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau)d\tau\\
&= \quad \min_{\mathbf{u}(\cdot)}\{\int_t^{t-\Delta t}-l(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau)d\tau+\int_{t-\Delta t}^{t_0}-l(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau)d\tau\}\\
&= \quad \min_{\mathbf{u}(\cdot)}\{\int_t^{t-\Delta t}-l(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau)d\tau+rv(t-\Delta t,\mathbf{x}(t-\Delta t))\}
\end{aligned}$$

Application of the Taylor expansion yields the following equation:

$$=\min_{\mathbf{u}}\{l(\mathbf{x},\mathbf{u},t)\Delta t+rv(t,\mathbf{x})-\frac{\partial rv}{\partial t}(t,\mathbf{x})\Delta t-\frac{\partial rv}{\partial\mathbf{x}}^T(t,\mathbf{x})f(\mathbf{x},\mathbf{u},t)\Delta t+o(\Delta t)\}$$

Dividing by Δt and taking the limit $\Delta t\to 0$ yields the following HJB equation:

$$0=\min_{\mathbf{u}}\{l(\mathbf{x},\mathbf{u},t)-\frac{\partial rv}{\partial t}(t,\mathbf{x})-\frac{\partial rv}{\partial\mathbf{x}}^T(t,\mathbf{x})f(\mathbf{x},\mathbf{u},t)\} \tag{18}$$

The corresponding inequality to this HJB equation (18) is obtained as follows:

$$RL(l,rv)(\mathbf{x},\mathbf{u},t)=l(\mathbf{x},\mathbf{u},t)-\frac{\partial rv}{\partial t}(t,\mathbf{x})-\frac{\partial rv}{\partial\mathbf{x}}^T(t,\mathbf{x})f(\mathbf{x},\mathbf{u},t)\geq 0$$

In the time-independent cases, this inequality becomes as simple as:

$$RL(l,rv)(\mathbf{x},\mathbf{u})=l(\mathbf{x},\mathbf{u})-\frac{\partial rv}{\partial\mathbf{x}}^T(\mathbf{x})f(\mathbf{x},\mathbf{u})\geq 0$$

C PROPOSED METHOD FOR MULTIPLE TRAJECTORIES SETTINGS

In this section, we describe our IOC method in multiple expert trajectories settings. There are k expert trajectories each indexed by j: $\{(\mathbf{x}_0^j, \mathbf{u}_0^j, t_0^j),...,(\mathbf{x}_{n_j-1}^j, \mathbf{u}_{n_j-1}^j, t_{n_j-1}^j)\}_{j=1,...,k}$, where each trajectory has n_j tuples of state, control and time stamp. The last states $\{\mathbf{x}_{n_j-1}^j\}_{j=1,...,k}$ in the trajectories are assumed to be in the terminal state set X_T.

The triangle inequality is derived for each trajectory. So, we define k reverse value functions $\{rv_j\}_{i=1,...k}$, each having a different initial state $\mathbf{x}_0^j$ in the expert trajectories as a terminal condition. We introduced a modified epsilon relaxation, Equation (19c), to deal with the k trajectories. Constraints (19f) to (19l) are defined for each trajectory because each trajectory utilizes its specific reverse value function.

$$\inf_{l,v,\epsilon_a,\{rv_j,\epsilon_b^j,\epsilon_c^j\}_{j=1,...,k}} \quad \epsilon_a + \textstyle\sum_{j=1}^k \{\epsilon_b^j + \epsilon_c^j\} + \lambda\|l\|_1 \tag{19a}$$

s.t.

-Constraints from the baseline method-

$$\mathcal{L}(l,v)(\mathbf{x},\mathbf{u}) \geq 0, \forall(\mathbf{x},\mathbf{u}) \in X \times U \tag{19b}$$

$$\frac{1}{\sum_{j=1}^k n_j} \sum_{j=1}^k \sum_{i=0}^{n_j-1} \mathcal{L}(l,v)\left(\mathbf{x}_i^j, \mathbf{u}_i^j\right) \leq \epsilon_a \tag{19c}$$

$$v(\mathbf{x}) = 0, \forall \mathbf{x} \in X_T \tag{19d}$$

$$\mathcal{A}(\mathcal{L}(l,v)) = 1 \tag{19e}$$

-Constraints from the introduction of $\{rv_j\}_{j=1,...,k}$-

$$\mathcal{RL}(l,rv_j)(\mathbf{x},\mathbf{u}) \geq 0, \forall(\mathbf{x},\mathbf{u}) \in X \times U \ (j=1,...,k) \tag{19f}$$

$$\frac{1}{n_j} \sum_{i=0}^{n_j-1} \mathcal{RL}(l,rv_j)\left(\mathbf{x}_i^j, \mathbf{u}_i^j\right) \leq \epsilon_b^j \ (j=1,...,k) \tag{19g}$$

$$rv_j(\mathbf{x}_0^j) = 0 \ (j=1,...,k) \tag{19h}$$

$$\mathcal{A}(\mathcal{RL}(l,rv_j)) = 1 \ (j=1,...,k) \tag{19i}$$

$$rv_j(\mathbf{x}_{n_j-1}^j) = v(\mathbf{x}_0^j) \ (j=1,...,k) \tag{19j}$$

-Constraints from the triangle inequality-

$$v(\mathbf{x}) + rv_j(\mathbf{x}) \geq v(\mathbf{x}_0^j), \forall \mathbf{x} \in X \ (j=1,...,k) \tag{19k}$$

$$\frac{1}{n_j} \sum_{i=0}^{n_j-1} \{v(\mathbf{x}_i^j) + rv_j(\mathbf{x}_i^j) - v(\mathbf{x}_0^j)\} \leq \epsilon_c^j \ (j=1,...,k) \tag{19l}$$

In time-dependent cases, k expert trajectories $\{(\mathbf{x}_0^j, \mathbf{u}_0^j, t_0^j),...,(\mathbf{x}_{n_j-1}^j, \mathbf{u}_{n_j-1}^j, t_{n_j-1}^j(=T))\}_{j=1,...,k}$ share the common terminal time T, and each last state $\{\mathbf{x}_{n_j-1}^j\}_{j=1,...,k}$ is in the terminal state set X_T. Initial time t_0^j for each trajectory can vary with the assumption $t_0^j \in [t_0, T]$ for some scalar t_0. Using Equation (14), the k time-dependent reverse value functions $\{rv_j\}_{i=1,...k}$ are defined as having initial time and state $t_0^j, \mathbf{x}_0^j$ as the terminal condition. Using the time-dependent constraints derived in Appendix A, the following program is formulated:

$$\inf_{l,v,\epsilon_a,\{rv_j,\epsilon_b^j,\epsilon_c^j\}_{j=1,..,k}} \quad \epsilon_a + \textstyle\sum_{j=1}^{k}\{\epsilon_b^j + \epsilon_c^j\} + \lambda \|l\|_1 \tag{20a}$$

s.t.

-Constraints from the baseline method-

$$\mathcal{L}(l,v)(\mathbf{x},\mathbf{u},t) \geq 0, \forall(\mathbf{x},\mathbf{u},t) \in X \times U \times [t_0, T] \tag{20b}$$

$$\frac{1}{\sum_{j=1}^{k} n_j} \sum_{j=1}^{k} \sum_{i=0}^{n_j-1} \mathcal{L}(l,v)\left(\mathbf{x}_i^j, \mathbf{u}_i^j, t_i^j\right) \leq \epsilon_a \tag{20c}$$

$$v(T,\mathbf{x}) = 0, \forall \mathbf{x} \in X_T \tag{20d}$$

$$\mathcal{A}(\mathcal{L}(l,v)) = 1 \tag{20e}$$

-Constraints from the introduction of $\{rv_j\}_{j=1,\ldots,k}$-

$$\mathcal{RL}(l,rv_j)(\mathbf{x},\mathbf{u},t) \geq 0, \forall(\mathbf{x},\mathbf{u},t) \in X \times U \times [t_0^j, T] \ (j=1,..,k) \tag{20f}$$

$$\frac{1}{n_j} \sum_{i=0}^{n_j-1} \mathcal{RL}(l,rv_j)\left(\mathbf{x}_i^j, \mathbf{u}_i^j, t_i^j\right) \leq \epsilon_b^j \ \ (j=1,...,k) \tag{20g}$$

$$rv_j(t_0^j, \mathbf{x}_0^j) = 0 \quad (j=1,...,k) \tag{20h}$$

$$\mathcal{A}(\mathcal{RL}(l,rv_j)) = 1 \quad (j=1,...,k) \tag{20i}$$

$$rv_j(T, \mathbf{x}_{n_j-1}^j) = v(t_0^j, \mathbf{x}_0^j) \quad (j=1,...,k) \tag{20j}$$

-Constraints from the triangle inequality-

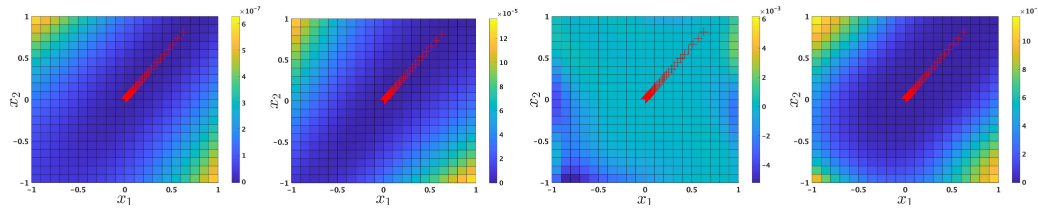
$$v(t,\mathbf{x}) + rv_j(t,\mathbf{x}) \geq v(t_0^j, \mathbf{x}_0^j), \forall(t,\mathbf{x}) \in [t_0^j, T] \times X \ (j=1,..,k) \tag{20k}$$

$$\frac{1}{n_j} \sum_{i=0}^{n_j-1} \{v(t_i^j, \mathbf{x}_i^j) + rv_j(t_i^j, \mathbf{x}_i^j) - v(t_0^j, \mathbf{x}_0^j)\} \leq \epsilon_c^j \ (j=1,..,k) \tag{20l}$$

D VALUE FUNCTION VISUALIZATION

To understand the benefits of the triangle inequality, we here visualize the value functions estimated by the baseline and our IOC methods. Figure 6 shows the value functions of OCP 1, estimated by the baseline IOC method (figures (6a) and (6c)) and our IOC method (figures (6b) and (6d)). We also examined two settings of the degree of approximation polynomials: degree of 3 (figures (6a) and (6b)) and 8 (figures (6c) and (6d)). The correct value function of OCP 1 is $v(\mathbf{x}) = x_1^2 + x_2^2$.

The baseline method and our method exhibited comparable approximations for the value functions with relatively small polynomial degree of 3 (figures (6a) and (6b)). However, their approximations were fairly different with a larger polynomial degree of 8 (figures (6c) and (6d)). Although the correct value function was mirror symmetric along the diagonal line in the two-dimensional state space and took its minimum at the origin, the baseline method approximated it as asymmetric. This can be seen as an over-adaptation; because the number of base monomials was relatively large with the polynomial degree being 8, the baseline method could not approximate well the value function for the distant region from the observed expert trajectory because of the shortage of constraints to extrapolate it. This result suggests that our IOC method improves the accuracy of the estimated value functions by utilizing the additional information from the triangle inequality, particularly when the number of monomial bases for the approximation is large.



(a) Baseline method degree 3 (b) Proposed method degree 3 (c) Baseline method degree 8 (d) Proposed method degree 8

Figure 6: Visualization of the estimated value functions. The red + marks show the expert trajectories used by the IOC methods. The correct value function is $v(\mathbf{x}) = x_1^2 + x_2^2$, which has circular contours.

E Computational resources

All the experiments were performed using a MacBook Pro with 16 GB memory and the $R2020b$ version of MATLAB. As we described in Section 3 of the main text, we followed the original method (Pauwels et al., 2014a) to let the inequality hold over the whole domain. In this method, the inequality constraints are reduced into the class of SOS (sum of squares) polynomials, and then can be transformed into linear matrix inequalities. Since our new method used the similar technique to implement the triangle inequality and the inequality from the HJB equation, the number of constraints further increased, but the computational time did not increase so much, at most three times that in a single trajectory setting.

We experimentally evaluate the computational resources in this section. Table 2 lists the mean computational time for estimating the cost function in OCP 1 and its standard deviation over 10 trials. Each column represents the degree of approximation polynomials. As shown in the degree 8 column in Table 2, the computational time required for our IOC method was 2.68 times longer than that of the baseline method.

Table 2: Computational time (sec) in the single expert trajectory setting

degree	4	5	6	7	8
Baseline method	1.05±0.03	1.44±0.04	2.67±0.06	4.47±0.24	9.37±0.59
Proposed method	2.04±0.10	2.88±0.07	5.57±0.18	9.47±0.25	25.12±1.79

In the multiple expert trajectories setting, the computational time for the baseline and our IOC methods increases with the number of trajectories. The computational time for these IOC methods was compared using OCP 1 with the approximation polynomial degree being 4 throughout this experiment. Table 3 lists the mean computational time and its standard deviation over 10 trials. Each column represents the number of expert trajectories. While the computational time for our IOC method increased almost linearly with the number of expert trajectories, that for the baseline method showed a small increase.

We consider this larger increase in our IOC method was caused by the additional constraints implemented for each trajectory. Since the number of constraints for HJB equations and triangle inequalities increased linearly with the number of expert trajectories, the computational time increased almost linearly as well.

Table 3: Computational time (sec) in the multiple expert trajectories setting

#trajectories	2	4	6	8	10
Baseline method	1.25±0.08	1.57±0.08	1.83±0.04	2.20±0.09	2.51±0.10
Proposed method	3.69±1.48	5.85±0.28	8.43±0.27	11.35±0.30	14.42±0.32

F Experimental details of a 2-link manipulator imitation

F.1 Experimental details

This section describes the details of the experiment presented in Section 4.4. Equation (21) represents the dynamics of a 2-link manipulator (Kumar et al., 2017) used in this study.

$$
\begin{aligned}
&\begin{bmatrix} M11 & M12 \\ M21 & M22 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} H1 \\ H2 \end{bmatrix} + \begin{bmatrix} G1 \\ G2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \\
&M_{11} = m_1 l_{a1}^2 + m_2 \left(l_1^2 + l_{a2}^2 + 2 l_1 l_{a2} \cos\theta_2 \right) + I_1 + I_2 \\
&M_{12} = M_{21} = m_2 \left(l_{a2}^2 + l_1 l_{a2} \cos\theta_2 \right) + I_2 \\
&M_{22} = m_2 l_{a2}^2 + I_2 \\
&H_1 = -m_2 l_1 l_{a2} \sin\theta_2 \dot{\theta}_2^2 - 2 m_2 l_1 l_{a2} \sin\theta_2 \dot{\theta}_2 \dot{\theta}_1 \\
&H_2 = m_2 l_{m1} l_2 \sin\theta_2 \dot{\theta}_1^2 \\
&G_1 = (m_1 l_{a1} + m_2 l_1)\, g \cos\theta_1 + m_2 l_{a2} g \cos(\theta_1 + \theta_2) \\
&G_2 = m_2 l_{a2} g \cos(\theta_1 + \theta_2)
\end{aligned} \tag{21}
$$

The state space was four-dimensional as $\mathbf{x} = (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$, where θ_1 and θ_2 are angles of the two joints of the manipulator. The control space was two-dimensional as $\mathbf{u} = (\tau_1, \tau_2)$, i.e., the torques applied to the two joints.

The parameters of the manipulator are shown in Table 4.

Table 4: Parameters of a 2-link manipulator

$m_1(Kg)$	$m_2(Kg)$	$I_1(Kg/m^2)$	$I_2(Kg/m^2)$	$l_{a1}(m)$	$l_{a2}(m)$	$l_1(m)$	$l_2(m)$	$g(m/s^2)$
1	1	1/12	1/3	1/2	1	1	2	0

The nonpolynomial dynamics was approximated by applying the Taylor expansion around $[\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2, \tau_1, \tau_2] = [-\pi/2, \pi/2, 0, 0, 0, 0]$ up to the second order.

Expert trajectory was given by the following equations:

$$
\begin{aligned}
\theta_1(t) &= \begin{cases} 96\pi t^5 - 120\pi t^4 + 40\pi t^3 - 3\pi/4 & (0 \le t < 1/2) \\ -96\pi(t-0.5)^5 + 120\pi(t-0.5)^4 - 40\pi(t-0.5)^3 - \pi/4 & (1/2 \le t \le 1) \end{cases} \\
\theta_2(t) &= \begin{cases} -96\pi t^5 + 120\pi t^4 - 40\pi t^3 + 3\pi/4 & (0 \le t < 1/2) \\ 96\pi(t-0.5)^5 - 120\pi(t-0.5)^4 + 40\pi(t-0.5)^3 + \pi/4 & (1/2 \le t \le 1) \end{cases}
\end{aligned} \tag{22}
$$

Equation (22) was designed to perform back-and-forth control between the two angles, $(\theta_1, \theta_2) = (-3\pi/4, 3\pi/4)$ and $(\theta_1, \theta_2) = (-\pi/4, \pi/4)$. The velocity and acceleration at each angle were 0. Obviously, the cost function associated with the expert trajectory given by Equation (22) is time-dependent.

The imitation behaviors were produced by solving the OCP with the following settings: 1) the cost function estimated by IOC with the approximated dynamics, 2) the initial and terminal states set to $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) = (-3\pi/4, 3\pi/4, 0, 0)$, and 3) the correct dynamics (21).

F.2 Additional experiment

As shown in Section 4.4, our method based on the triangle inequality successfully imitated the expert behaviors. As an additional experiment, we compared the baseline and our IOC methods in terms of imitation performance with various settings of the hyper-parameter λ. During this experiment, the degree of approximation polynomials was consistently set at 4 to keep the computation time feasible.

Table 5: Squared errors of imitation trajectories

λ	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}
Baseline method	23.71	36.65	30.09	9.46	9.46
Proposed method	19.11	15.58	11.02	6.01	6.01

Table 5 presents the squared error between the expert's and mimicker's control trajectories. Both methods performed well for a smaller setting of the hyper-parameter λ, and our method could imitate the expert's behaviors more accurately than the baseline method with every setting of λ. Figure 5 displayed the case of the smallest error in Table 5, i.e., that by our new IOC algorithm with $\lambda = 10^{-10}$.

G Sensitivity to noises

Most IOC methods, including the baseline (Pauwels et al., 2014a) and our method, assume that the demonstrated trajectories are optimal, i.e., the solutions of OCP. Here, we examined how well our IOC method works, when this optimality is not fully satisfied; in particular, we applied our IOC method to a situation in which there was a single expert trajectory that had been disturbed by observation noise. As an experimental setup, we re-used the setup for OCP1 with a single expert trajectory setting (Figure (2a) in the main text), but we applied two modifications: 1) the random noise from the uniform distribution of $[-0.05.0.05]$ was added to each state and control signal along the expert trajectory, and 2) the initial state was sampled from the uniform distribution of $[-0.95, 0.95]$ for each coordinate instead of $[-1, 1]$ to ensure the noisy trajectory stayed in the domain X. Figure 7 shows the accuracy of the estimated cost function over ten trials with this setting. Our IOC method estimated the cost function more accurately than the baseline method even when the expert trajectory included observation noise.

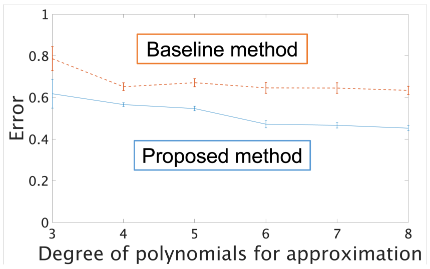


Figure 7: Normalized errors by the two IOC methods for various degrees of approximation polynomials. The lines and error bars denote the mean and standard error over ten trials.