Online Control-Informed Learning

Anonymous authors
Paper under double-blind review

Abstract

This paper proposes an Online Control-Informed Learning (OCIL) framework, which synthesizes the well-established control theories to solve a broad class of learning and control tasks in real time. This novel integration effectively handles practical issues in machine learning such as noisy measurement data, online learning, and data efficiency. By considering any robot as a tunable optimal control system, we propose an online parameter estimator based on extended Kalman filter (EKF) to incrementally tune the system in real time, enabling it to complete designated learning or control tasks. The proposed method also improves robustness in learning by effectively managing noise in the data. Theoretical analysis is provided to demonstrate the convergence and regret of OCIL. Three learning modes of OCIL, i.e. Online Imitation Learning, Online System Identification, and Policy Tuning On-the-fly, are investigated via experiments, which validate their effectiveness.

1 Introduction

Informed Machine Learning (IML) (Von Rueden et al., 2021) represents an emerging approach integrating prior knowledge into the machine learning (ML) process. While classic classification tasks in unsupervised, semi-supervised, or supervised ML primarily focus on extracting patterns from labeled or unlabeled data (LeCun et al., 2015), IML leverages prior knowledge such as physical laws, expert knowledge, or existing models to uncover underlying connections within data (Karniadakis et al., 2021). This integration enables models to produce more reliable predictions with enhanced interpretability, particularly in scenarios where data is scarce, noisy, or complex. This approach is especially advantageous in the domains where theoretical understanding is well-established and thus can guide ML. One notable example of IML is physics-informed machine learning (Karniadakis et al., 2021), which proves particularly valuable in solving partial differential equations for computational fluid dynamics.

Control-informed learning (CIL) is a subset of IML tailored for system control, autonomy, and robotics. This approach merges standard control theory, especially the established optimal control principles, with ML techniques to enhance the functionality of autonomous systems. This integration is beneficial for robotic applications, where optimal control systems typically govern robots. By incorporating prior knowledge about the system, such as dynamic models, control laws, and optimization strategies, into the learning process, CIL reduces the required training data, accelerates deployment, and enhances safety and efficiency by ensuring adherence to established control theories (Jin et al., 2020; 2021b).

Online control-informed learning (OCIL) represents a subset of CIL that continually updates its model as new data streams. This is beneficial for robotic applications, where the environment can change unpredictably. One advantage of online methods is their ability to learn and adapt on the fly. Unlike offline methods, which require a static dataset for training, online learning algorithms adjust their parameters or models in response to new information without training from scratch. This continual learning capability allows autonomous systems to improve their policies in real time. Additionally, OCIL can handle dynamic environments by continually updating its parameters. This feature is critical in applications such as autonomous driving, where adapting to new scenarios quickly is crucial for safety and performance. This concept intersects with many ML methods such as transfer learning (Pan & Yang, 2009), continual learning (Aljundi et al., 2019), and learning on-the-fly (Ornik et al., 2019).

1.1 Related Work

The proposed OCIL framework includes three modes for three sets of classic problems in both ML and system control communities. Hence, the following literature review presents some representative methods from both perspectives, for all three modes.

Online System Identification To identify a nonlinear system with possibly noisy measurement in an online fashion, Markov-decision-process-based methods are widely used, such as linear regression (Haruno et al., 2001), observation-transition modeling (Finn et al., 2016), latent-space modeling (Watter et al., 2015), (deep) neural networks (Fragkiadaki et al., 2015), Gaussian processes (Deisenroth & Rasmussen, 2011), and transition graphs (Zhang et al., 2018). Despite their widespread use, these methods often must balance data efficiency against prediction accuracy. To improve both metrics, physics-informed learning approaches Raissi et al. (2019); Saemundsson et al. (2020); Lutter et al. (2019); Zhong et al. (2019) incorporate physical laws into learning models. Recently, there has been an emerging trend to interpret (deep) neural networks through the lens of dynamical systems, leading to the development of several new algorithms (Chen et al., 2018; Han & E, 2016; Li et al., 2018; Li & Hao, 2018; Han et al., 2019; Zhang et al., 2019; Benning et al., 2019; Liu & Markowich, 2020). Koopman operator theory offers a method to lift states to an infinite-dimensional linear observable space (Mauroy et al., 2020; Williams et al., 2015). The recent deep Koopman representation (DKR) utilizes neural networks (NNs) to represent the observables that are difficult to formulate by hand without expertise in a particular dynamical system (Liang et al., 2023; Hao et al., 2023). This paper proposes an online learning framework to estimate system dynamics in real time as noisy measurement data continually comes in. We aim to incorporate inductive knowledge from optimal control theory to enhance both run-time computational efficiency, robustness against m measurement noise, and prediction accuracy.

Online Imitation Learning Online imitation and objective learning are typically referred to as inverse reinforcement learning (IRL) in the ML community and online inverse optimal control (IOC) in the system control community. IRL aims to deduce a control objective function with observed optimal demonstrations. The objective function is generally represented as a weighted sum of features (Abbeel & Ng, 2004; Ziebart et al., 2008; Ratliff et al., 2006). Approaches to find these unknown weights include feature matching (Abbeel & Ng, 2004), maximum entropy (Ziebart et al., 2008), and maximum margin (Ratliff et al., 2006). These IRL methods update the learned objective estimate within a defined feature space, exploiting the linearity of feature weights. As for learning nonlinear parameter mapping of objective functions, prior and system-dependent knowledge is required to further extend the methods above. On the other hand, with system dynamics, IOC aims for efficient learning approaches (Keshavarz et al., 2011; Mombaur et al., 2010; Liang et al., 2022; 2023; Jin et al., 2019; 2021a; Jin & Mou, 2021). For instance, some methods (Keshavarz et al., 2011; Liang et al., 2022; 2023; Jin et al., 2019; 2021a; Jin & Mou, 2021) directly calculate unknown weights by minimizing the violation of optimality conditions by the observed demonstration data, which avoid solving optimal control problems repetitively. This paper presents an online framework to facilitate learning objective functions online as noisy demonstration data continually streams in. By synthesizing the online state estimation techniques in control theory, the proposed method can recover the object function in real time while ensuring robustness against noise.

Tuning Policy On-the-fly Tuning policy on the fly is typically referred to as transfer learning and tuning optimal control (OC) systems in ML and control communities, respectively. Transfer learning exploits the generalization of existing knowledge such that it can be transferred across different domains (Taylor & Stone, 2009). Recently, transfer learning has been investigated in the reinforcement learning (RL) community because the knowledge gained in one task may improve learning performance in a related but different task. Such a concept of transfer learning may speed up the learning process in RL (Taylor & Stone, 2009). If the knowledge transfers from a domain of experts to a target domain, imitation learning (Ho & Ermon, 2016; Wu et al., 2019; Oh et al., 2018), behavior cloning (Torabi et al., 2018; Sasaki & Yamashina, 2021), or learning from demonstrations (Schaal, 1996; Kim et al., 2013; Jin et al., 2022a) can also be viewed as transfer learning. Knowledge from a source domain can also be represented by an action probability distribution of an expert policy (Czarnecki et al., 2019) or even directional corrections from expert (Jin et al., 2022b). In the control community, tuning OC systems initially refers to neighboring extremal optimal control (NEOC) (Bryson, 1975; Ghaemi et al., 2009). Recently, Jin et al. (2020) proposes a framework to tune an OC system based on differentiating Pontryagin's Maximum Principle. Tuning an OC system by either NEOC or differentiating

Pontryagin's Principle requires the knowledge of an entire optimal trajectory. However, for online tasks, the knowledge of the entire optimal trajectory is unavailable at run-time until the end. The proposed online learning framework in this paper includes a specific mode for model-based control tasks, where some information is unavailable until the current time.

1.2 Contributions

This paper introduces an online learning framework called Online Control-Informed Learning (OCIL). This framework is designed to be data efficient for various learning and control tasks while providing robustness against noisy data. In this paper, we consider a robot as an OC system, which is parameterized by tunable parameters within different components of the system, including dynamics, policy, and objective function. By tuning the OC system in real time, the proposed OCIL tackles three learning tasks in robotics, namely Online Imitation Learning, Online System Identification, and Policy Tuning On-the-fly. The proposed OCIL consists of two main components, both of which are inspired by control theory. Specifically, the framework first proposes an online parameter estimator based on the classic online state estimation techniques in control theory. The estimator continually updates the parameter estimates in real time as new data becomes available, aiming to minimize a cumulative loss defined for a specific task. To do so, the gradient information for the loss with respect to the tunable parameter is required. Therefore, OCIL employs a gradient generator (GG) based on Pontryagin Differential Programming in optimal control theory to calculate the exact gradient.

Notations. $\|\cdot\|$ denotes the Euclidean norm. Given a matrix $A \in \mathbb{R}^{n\times m}$, let A' denotes its transpose. For positive integers n and m, let $\mathbf{I}_n$ be the $n \times n$ identity matrix; $\mathbf{0}_n \in \mathbb{R}^n$ denotes a vector with all value 0; $\mathbf{0}_{n\times m}$ denotes a $n \times m$ matrix with all value 0. Let $\text{col}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_a\}$ denote a column stack of elements $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_a$, which may be scalars, vectors or matrices, i.e. $\text{col}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_a\} \triangleq [\boldsymbol{v}_1' \ldots \boldsymbol{v}_a']$.

2 Problem Formulation

Consider the following class of optimal control (OC) systems $\Sigma(\boldsymbol{\theta})$:

$$\begin{aligned} \text{system dynamics:} \quad & \boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta}), \quad \text{with } \boldsymbol{x}_0 \text{ given}, \\ \text{objective to be minimized:} \quad & J(\boldsymbol{\theta}) = \textstyle\sum_{t=0}^{T-1} c(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta}) + h(\boldsymbol{x}_T, \boldsymbol{\theta}), \end{aligned} \tag{1}$$

where $t = 0, 1, 2, \cdots, T$ is the time index with T being the final time; $\boldsymbol{x}_t \in \mathbb{R}^n$ and $\boldsymbol{u}_t \in \mathbb{R}^m$ denote the system state and control input, respectively; $\boldsymbol{\theta} \in \mathbb{R}^p$ denotes a tunable parameter; $\boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^n$ denotes a twice-differentiable time-invariant system dynamics; $c : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \mapsto \mathbb{R}$ and $h : \mathbb{R}^n \times \mathbb{R}^p \mapsto \mathbb{R}$ denote running cost the final cost, respectively, both of which are assumed to be twice-differentiable. For a choice of $\boldsymbol{\theta}$, the trajectory of optimal control system (1) can be determined by solving the optimal control problem:

$$\begin{aligned} \{\boldsymbol{x}_{0:T}(\boldsymbol{\theta}), \boldsymbol{u}_{0:T-1}(\boldsymbol{\theta})\} \in \arg \min_{\substack{\boldsymbol{x}_{0:T} \\ \boldsymbol{u}_{0:T-1}}} \quad & J(\boldsymbol{\theta}) \\ \text{subject to} \quad & \boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta}), \ \forall t \text{ with given } \boldsymbol{x}_0. \end{aligned} \tag{2}$$

where $\boldsymbol{x}_{0:T}(\boldsymbol{\theta}) \triangleq \text{col}\ \{\boldsymbol{x}_0(\boldsymbol{\theta}), \cdots, \boldsymbol{x}_T(\boldsymbol{\theta})\}$ and $\boldsymbol{u}_{0:T-1}(\boldsymbol{\theta}) \triangleq \text{col}\ \{\boldsymbol{u}_0(\boldsymbol{\theta}), \cdots, \boldsymbol{u}_{T-1}(\boldsymbol{\theta})\}$ denote the states and inputs trajectory given parameter $\boldsymbol{\theta}$, respectively. For notation simplicity, we define the trajectory of the optimal control system as $\boldsymbol{\xi}(\boldsymbol{\theta}) \triangleq \text{col}\{\boldsymbol{x}_{0:T}(\boldsymbol{\theta}), \boldsymbol{u}_{0:T-1}(\boldsymbol{\theta})\} \in \mathbb{R}^{(T+1)n+Tm}$. Since our goal is to develop an online update rule, we introduce the iteration index k, where $\boldsymbol{\theta}_k \in \mathbb{R}^p$ represents the tunable parameter at iteration k. However, since the parameter is updated at each time t in an online algorithm, k is equivalent to t, leading to $\boldsymbol{\theta}_k \equiv \boldsymbol{\theta}_t \in \mathbb{R}^p$. At each time t, given a slice of trajectory $\boldsymbol{\xi}(\boldsymbol{\theta}_t)$ at time t, denoted as $\boldsymbol{\xi}_t(\boldsymbol{\theta}_t) \triangleq \text{col}\{\boldsymbol{x}_t(\boldsymbol{\theta}_t), \boldsymbol{u}_t(\boldsymbol{\theta}_t)\} \in \mathbb{R}^{n+m}$. There is also information $\boldsymbol{O}_t \in \mathbb{R}^r$, which is used to evaluate $\boldsymbol{\xi}_t(\boldsymbol{\theta}_t)$ and remains inaccessible until time t. The information $\boldsymbol{O}_t$ could represent the desired state, control input, or reference measured output. Examples of $\boldsymbol{O}_t$ will be provided at the end of this section. We assume that the information $\boldsymbol{O}_t$ is subject to Gaussian measurement noise, denoted as $\boldsymbol{v}_t \sim \mathcal{N}(\mathbf{0}_r, \boldsymbol{R}_t)$, where $\boldsymbol{R}_t \in \mathbb{R}^{r\times r}$ represents the covariance matrix of the noise. In other words, at each time step t, the information received can be noisy and expressed as $\boldsymbol{O}_t = \boldsymbol{O}_t^* + \boldsymbol{v}_t$, where $\boldsymbol{O}_t^*$ represents the true but unknown information.

The *problem of interest* is to develop an online method to update $\boldsymbol{\theta}$ at every time t, such that its trajectory $\boldsymbol{\xi}(\boldsymbol{\theta})$ from (2) minimizes a task-specific cumulative loss $L(\boldsymbol{\xi}(\boldsymbol{\theta}))$:

$$\min_{\boldsymbol{\theta}} \quad L(\boldsymbol{\xi}(\boldsymbol{\theta})) \quad \text{subject to} \quad \boldsymbol{\xi}(\boldsymbol{\theta}) \text{ is in 2.} \tag{3}$$

To define $L(\boldsymbol{\xi}(\boldsymbol{\theta}))$, we consider a predefined differentiable stage loss function $\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{O}_t) : \mathbb{R}^{n+m} \times \mathbb{R}^r \to \mathbb{R}^r$. Then, the performance of the entire trajectory can be evaluated by the cumulative loss:

$$L(\boldsymbol{\xi}(\boldsymbol{\theta})) = \textstyle\sum_{t=0}^{T} \|\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{O}_t)\|^2. \tag{4}$$

To achieve a specific learning or control task, one needs to design a specific stage loss function $\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{O}_t)$ and select the most suitable information $\boldsymbol{O}_t$. Below, we will present several examples to illustrate how this can be done. It is worth noting that in different applications, adjustments to the configuration of system $\Sigma(\boldsymbol{\theta})$ are required according to the task.

Online SysID: For a SysID problem, the goal is to identify the dynamics model of a physical system from the state-input trajectory $\boldsymbol{\xi}^o = \{\boldsymbol{x}^o_{0:T}, \boldsymbol{u}_{0:T-1}\}$, where the superscript o denotes the observed trajectory. The trajectory is often generated by persistent excitation of the system without considering any control objectives (Keesman, 2011). Therefore, we can set $J(\boldsymbol{\theta}) = 0$:

$$\Sigma(\boldsymbol{\theta}) : \begin{matrix} \text{dynamics:} & \boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta}), \quad \text{with } \boldsymbol{x}_0 \text{ given,} \\ \text{objective:} & J(\boldsymbol{\theta}) = 0. \end{matrix} \tag{5}$$

To identify the model dynamics, namely finding the $\boldsymbol{\theta}$ in the dynamics $\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta})$, one could design the stage loss function to represent the discrepancy between the observed trajectory and the trajectory produced by $\boldsymbol{\theta}$, i.e. $\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{\xi}^o_t) = \boldsymbol{\xi}^o_t - \boldsymbol{\xi}_t(\boldsymbol{\theta})$, where $\boldsymbol{\xi}^o_t$ is a slice of $\boldsymbol{\xi}^o$ at time t. In the SysID mode, the information $\boldsymbol{O}_t$ received at time t is a slice of the trajectory of a physical system $\boldsymbol{\xi}^o_t$.

Online Imitation Learning: The objective function and the model dynamics are parameterized by an unknown $\boldsymbol{\theta}$. The OC system follows (1). Suppose one can observe the measurement of the optimal demonstration $\boldsymbol{y}^*_t$ at each time t, where $\boldsymbol{y}_t = \boldsymbol{g}(\boldsymbol{x}_t(\boldsymbol{\theta}), \boldsymbol{u}_t(\boldsymbol{\theta}))$ is a known differentiable mapping from the state and input to a measured output. Then, the stage loss function can be designed as $\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{y}^*_t) = \boldsymbol{y}^*_t - \boldsymbol{g}(\boldsymbol{x}_t(\boldsymbol{\theta}), \boldsymbol{u}_t(\boldsymbol{\theta}))$. In this case, the information $\boldsymbol{O}_t$ received at time t is the measurement of the optimal demonstration $\boldsymbol{y}^*_t$. The optimal demonstration can vary between being continuous or sparse, depending on practical application scenarios.

Tuning Policy On-the-fly: For a specific system, we want to obtain a feedback controller such that the trajectory minimizes certain task loss. We consider a feedback controller which is parameterized by $\boldsymbol{\theta}$, i.e. $\boldsymbol{u}_t = \mu(\boldsymbol{x}_t, \boldsymbol{\theta})$. Then the OC system is written as follows:

$$\Sigma(\boldsymbol{\theta}) : \begin{matrix} \text{dynamics:} & \boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \mu(\boldsymbol{x}_t, \boldsymbol{\theta})), \quad \text{with } \boldsymbol{x}_0 \text{ given,} \\ \text{objective:} & J = \sum_{t=0}^{T-1} c(\boldsymbol{x}_t, \mu(\boldsymbol{x}_t, \boldsymbol{\theta})) + h(\boldsymbol{x}_T). \end{matrix} \tag{6}$$

Then we can introduce any loss function to be minimized, which represents a specific tuning objective or specification, such as stabilizing the system on a particular point or tracking a desired trajectory.

3 Main Results

The proposed OCIL consists of two main components, both of which are inspired by control theory. Specifically, OCIL first proposes an online parameter estimator based on the extended Kalman filter (EKF). Going forward, we will show the challenge of obtaining the Kalman gain. To tackle this challenge, the gradient information for the loss with respect to the tunable parameter is required. Therefore, OCIL employs a gradient generator (GG) based on Pontryagin Differential Programming to calculate the exact gradient. Then the proposed OCIL framework will be introduced and supported with theoretical analysis.

3.1 Online Parameter Estimator

To minimize the cumulative task loss $L(\boldsymbol{\xi}(\boldsymbol{\theta}))$ with information $\boldsymbol{O}_t$, which is unavailable until time t, the optimization problem that needs to be solved in real time is:

$$\min_{\boldsymbol{\theta}} \quad \sum_{t=0}^{T} \|\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{O}_t)\|^2 \quad \text{subject to} \quad \boldsymbol{\xi}(\boldsymbol{\theta}) \text{ is in 2.} \tag{7}$$

The optimization problem (7) is essentially a least squares problem, although under constraints. One of the most famous methods to solve the least squares problems incrementally is the EKF (Bertsekas, 1996). The EKF was proposed to incrementally estimate the state of a system using measured output available at each time step. In our problem setting, instead of estimating the state of a system, our goal is to estimate the parameter $\boldsymbol{\theta}$ by utilizing the information $\boldsymbol{O}_t$ that is available at each time t. Therefore, by considering the tunable parameter $\boldsymbol{\theta}$ as the state to be estimated, its corresponding system can be written as:

$$\text{dynamics: } \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{w}_t, \quad \text{measurement: } \boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}_t), \boldsymbol{O}_t) + \boldsymbol{v}_t \equiv \boldsymbol{0}_s \tag{8}$$

where $\boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}_p, \boldsymbol{Q}_t)$ is the process noise which is assumed to be multivariate Gaussian noise. $\boldsymbol{Q}_t \in \mathbb{R}^{p\times p}$ is the covariance matrices of the process noise. In our experiments, we omit process noise since we are estimating a static parameter rather than the states of a dynamic system. Process noise usually accounts for dynamic uncertainty in such systems, which does not apply to our scenario. Nevertheless, we include the process noise here to support the convergence analysis. The measurement equation in (8) indicates that the stage loss with optimal $\boldsymbol{\theta}$ is assumed to be zero. The $\boldsymbol{\theta}$ estimate via EKF can be done as follows:

$$\hat{\boldsymbol{\theta}}_t^- := \hat{\boldsymbol{\theta}}_{t-1}, \ \boldsymbol{P}_t^- := \boldsymbol{P}_{t-1} + \boldsymbol{Q}_{t-1} \tag{9a}$$

$$\boldsymbol{K}_t := \boldsymbol{P}_t^- \boldsymbol{L}_t'(\boldsymbol{L}_t \boldsymbol{P}_t^- \boldsymbol{L}_t' + \boldsymbol{R}_t)^{-1}, \ \boldsymbol{P}_t := (\boldsymbol{I}_p - \boldsymbol{K}_t \boldsymbol{L}_t)\boldsymbol{P}_t^-, \ \hat{\boldsymbol{\theta}}_t := \hat{\boldsymbol{\theta}}_t^- + \boldsymbol{K}_t(\boldsymbol{0}_s - \boldsymbol{l}(*)), \tag{9b}$$

where $\boldsymbol{l}(*) \triangleq \boldsymbol{l}(\boldsymbol{\xi}_t(\hat{\boldsymbol{\theta}}_t^-), \boldsymbol{O}_t)$; $\boldsymbol{L}_t \triangleq \frac{d\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{O}_t)}{d\boldsymbol{\theta}}|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_t^-} \in \mathbb{R}^{r\times p}$; (9a) predicts the dynamics; (9b) updates the parameter estimate. Here, $\hat{\boldsymbol{\theta}} \in \mathbb{R}^p$ is the estimation of parameter $\boldsymbol{\theta}$, the superscript $^-$ means the term is not yet updated by measurement residual; $\boldsymbol{P}_t \in \mathbb{R}^{p\times p}$ is a positive-definite matrix that denotes the covariance of the estimate; $\boldsymbol{K}_t \in \mathbb{R}^{p\times r}$ denotes the Kalman gain. Throughout the estimation process, all of the terms are known except $\boldsymbol{L}_t$. It is challenging to obtain this term as the stage loss $\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{O}_t)$ is not an explicit function of $\boldsymbol{\theta}$. In the next subsection, we will present a *gradient generator* which computes the exact value for $\boldsymbol{L}_t$.

3.2 Gradient Generator

In this section, for brevity, the stage loss $\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{O}_t)$ is written as $\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}))$ since the gradient is not related to $\boldsymbol{O}_t$. Additionally, for the remainder of the paper, to maintain brevity, the notation $|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_t^-}$ will be omitted for all the partial derivatives that involve $\boldsymbol{\theta}$. To obtain the gradient $\frac{d\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}))}{d\boldsymbol{\theta}}$, one can employ the chain rule by definition,

$$\frac{d\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}))}{d\boldsymbol{\theta}} = \frac{\partial \boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}))}{\partial \boldsymbol{\xi}_t(\boldsymbol{\theta})} \frac{\partial \boldsymbol{\xi}_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}, \tag{10}$$

where $\frac{\partial \boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}))}{\partial \boldsymbol{\xi}_t(\boldsymbol{\theta})}$ is known since the stage loss is pre-designed. The challenge that remains is to find the partial derivative $\frac{\partial \boldsymbol{\xi}_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$, i.e. an analytical relation between trajectory $\boldsymbol{\xi}_t$ and the tunable parameter $\boldsymbol{\theta}$. To tackle this challenge, the gradient generator in Jin et al. (2020) is used to obtain the exact value of $\frac{\partial \boldsymbol{\xi}_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$. Given the OC system (2), one can obtain the Hamiltonian equation

$$H_t = c(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta}) + \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta})' \boldsymbol{\lambda}_{t+1} \tag{11}$$

for all $t = 0, \cdots, T-1$, where $\boldsymbol{\lambda}_t \in \mathbb{R}^n$ denotes the Lagrangian multiplier associated with the equality constraint of model dynamics. With the definition of $\boldsymbol{\xi}(\boldsymbol{\theta})$, one has $\frac{\partial \boldsymbol{\xi}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \text{col}\{\frac{\partial \boldsymbol{x}_{0:T}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}, \frac{\partial \boldsymbol{u}_{0:T-1}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\}$. By

defining

$$X_t \triangleq \frac{\partial \boldsymbol{x}_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{n\times p}, \quad U_t \triangleq \frac{\partial \boldsymbol{u}_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{m\times p}, \tag{12}$$

one can utilize the following lemma from Jin et al. (2020) to obtain the partial derivatives $\frac{\partial \boldsymbol{\xi}_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$:

Lemma 1. *Jin et al. (2020) By defining the Jacobian and Hessian matrices related to $\boldsymbol{\xi}(\boldsymbol{\theta})$:*

$$\begin{aligned}
&\boldsymbol{F}_t = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t},\ \boldsymbol{G}_t = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_t},\ \boldsymbol{E}_t = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}, \boldsymbol{H}_t^{xx} = \frac{\partial^2 H_t}{\partial \boldsymbol{x}_t \partial \boldsymbol{x}_t},\ \boldsymbol{H}_t^{xu} = \frac{\partial^2 H_t}{\partial \boldsymbol{x}_t \partial \boldsymbol{u}_t} = (\boldsymbol{H}_t^{ux})', \\
&\boldsymbol{H}_t^{uu} = \frac{\partial^2 H_t}{\partial \boldsymbol{u}_t \partial \boldsymbol{u}_t}, \boldsymbol{H}_t^{x\theta} = \frac{\partial^2 H_t}{\partial \boldsymbol{x}_t \partial \boldsymbol{\theta}},\ \boldsymbol{H}_t^{u\theta} = \frac{\partial^2 H_t}{\partial \boldsymbol{u}_t \partial \boldsymbol{\theta}},\ \boldsymbol{H}_T^{xx} = \frac{\partial^2 h}{\partial \boldsymbol{x}_T \partial \boldsymbol{x}_T},\ \boldsymbol{H}_T^{x\theta} = \frac{\partial^2 h}{\partial \boldsymbol{x}_T \partial \boldsymbol{\theta}},
\end{aligned} \tag{13}$$

if $\boldsymbol{H}_t^{uu}$ is invertible for all $t = 0, \cdots, T-1$, define the following recursions

$$\begin{aligned}
&\boldsymbol{V}_t = \boldsymbol{C}_t + \boldsymbol{A}_t'(\boldsymbol{I} + \boldsymbol{V}_{t+1}\boldsymbol{B}_t)^{-1}\boldsymbol{V}_{t+1}\boldsymbol{A}_t, \\
&\boldsymbol{W}_t = \boldsymbol{A}_t'(\boldsymbol{I} + \boldsymbol{V}_{t+1}\boldsymbol{B}_t)^{-1}(\boldsymbol{W}_{t+1} + \boldsymbol{V}_{t+1}\boldsymbol{M}_t) + \boldsymbol{N}_t,
\end{aligned} \tag{14}$$

with $\boldsymbol{V}_T = \boldsymbol{H}_T^{xx}$ and $\boldsymbol{W}_T = \boldsymbol{H}_T^{x\theta}$. Here, $\boldsymbol{A}_t = \boldsymbol{F}_t - \boldsymbol{G}_t(\boldsymbol{H}_t^{uu})^{-1}\boldsymbol{H}_t^{ux}$, $\boldsymbol{B}_t = \boldsymbol{G}_t(\boldsymbol{H}_t^{uu})^{-1}\boldsymbol{G}_t'$, $\boldsymbol{M}_t = \boldsymbol{E}_t - \boldsymbol{G}_t(H_t^{uu})'\boldsymbol{H}_t^{u\theta}$, $\boldsymbol{C}_t = \boldsymbol{H}_t^{xx} - \boldsymbol{H}_t^{xu}(\boldsymbol{H}_t^{uu})^{-1}\boldsymbol{H}_t^{ux}$, $\boldsymbol{N}_t = \boldsymbol{H}_t^{x\theta} - \boldsymbol{H}_t^{xu}(\boldsymbol{H}_t^{uu})'\boldsymbol{H}_t^{u\theta}$ are all known given (13). Then, the partial derivative $\frac{\partial \boldsymbol{\xi}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ can be obtained by recursively solving the following equations from $t = 0$ to $T-1$ with $\boldsymbol{X}_0(\boldsymbol{\theta}) = \boldsymbol{0}$:

$$\begin{aligned}
&\boldsymbol{U}_t = -(\boldsymbol{H}_t^{uu})^{\text{-}1}(\boldsymbol{H}_t^{ux}\boldsymbol{X}_t + \boldsymbol{H}_t^{u\theta} + \boldsymbol{G}_t'(\boldsymbol{I} + \boldsymbol{V}_{t+1}\boldsymbol{B}_t)^{\text{-}1}(\boldsymbol{V}_{t+1}\boldsymbol{A}_t\boldsymbol{X}_t + \boldsymbol{V}_{t+1}\boldsymbol{M}_t + \boldsymbol{W}_{t+1})), \\
&\boldsymbol{X}_{t+1} = \boldsymbol{F}_t\boldsymbol{X}_t + \boldsymbol{G}_t\boldsymbol{U}_t + \boldsymbol{E}_t.
\end{aligned} \tag{15}$$

The terms in (13) are based on the trajectory $\boldsymbol{\xi}(\boldsymbol{\theta})$ and the associated Lagrangian multiplier $\boldsymbol{\lambda}_{0:T-1}$. According to the discrete-time Pontryagin Maximum Principle (Jin et al., 2020), the trajectory of the Lagrangian multiplier can be obtained by

$$\boldsymbol{\lambda}_T = \frac{\partial h}{\partial \boldsymbol{x}_T}, \quad \boldsymbol{\lambda}_t \triangleq \frac{\partial H_t}{\partial \boldsymbol{x}_t} = \frac{\partial c}{\partial \boldsymbol{x}_t} + \frac{\partial h}{\partial \boldsymbol{x}_t}\boldsymbol{\lambda}_{t+1},\ \text{for } t = T-1, \cdots, 1. \tag{16}$$

3.3 OCIL Framework

With the online parameter estimator and the gradient generator, we propose the Online Control-Informed Learning framework in Fig. 1. The framework is summarized in Algorithm 2.

As shown in Fig. 1, at each time step, the predefined OC system $\Sigma(\boldsymbol{\theta})$ generates a system trajectory $\boldsymbol{\xi}(\boldsymbol{\theta})$ by performing optimal control with given $\boldsymbol{x}_0$ and $\boldsymbol{\theta}$. The trajectory $\boldsymbol{\xi}(\boldsymbol{\theta})$ is then fed into the stage loss function $\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{O}_t)$ and the gradient generator. Along with the information $\boldsymbol{O}_t$ obtained at time t, the stage loss function generates $\frac{\partial \boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{O}_t)}{\partial \boldsymbol{\xi}_t(\boldsymbol{\theta})}$, while $\frac{\partial \boldsymbol{\xi}_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ is generated by the gradient generator in Algorithm 1. The chain rule is then performed to obtain the Jacobian matrix $\boldsymbol{L}_t$, which is then passed into the online parameter estimator for the estimation of $\boldsymbol{\theta}$.

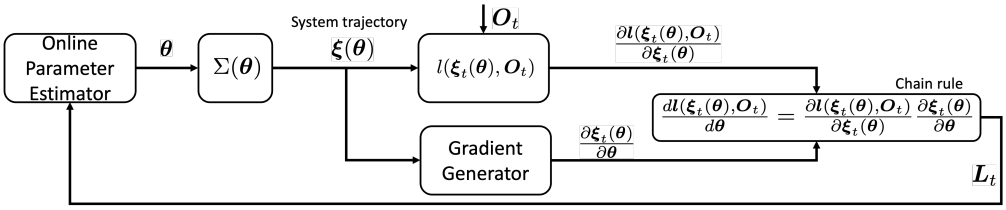


Figure 1: Framework of Online Control-Informed Learning.

Algorithm 1: Gradient Generator (GG)

Input: Trajectory $\boldsymbol{\xi}(\boldsymbol{\theta})$ from $\Sigma(\boldsymbol{\theta})$

Compute the coefficient matrices in (13) ;

Set $\boldsymbol{V}_T = \boldsymbol{H}_T^{xx}$ and $\boldsymbol{W}_T = \boldsymbol{H}_T^{x\theta}$;

for $t \leftarrow T$ *to* 0 *by* Δt **do**

  Update $\boldsymbol{V}_t$ and $\boldsymbol{W}_t$ using (14)

Set $\boldsymbol{X}_0(\boldsymbol{\theta}) = \boldsymbol{0}$;

for $t \leftarrow 0$ *to* T *by* Δt **do**

  Update $\boldsymbol{X}_t(\boldsymbol{\theta})$ and $\boldsymbol{U}_t(\boldsymbol{\theta})$ using (15)

Output: $\dfrac{\partial \boldsymbol{\xi}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \{\boldsymbol{X}_{0:T}(\boldsymbol{\theta}), \boldsymbol{U}_{0:T-1}(\boldsymbol{\theta})\}$

Algorithm 2: Online Control-Informed Learning

System and Stage Loss: $\Sigma(\boldsymbol{\theta})$ and $\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{O}_t)$

Initialize: $\boldsymbol{\theta}_0, \boldsymbol{P}_0$

for $t = t_0, t_1, \cdots$ **do**

  Obtain new information $\boldsymbol{O}_t$;

  Solve $\boldsymbol{\xi}(\boldsymbol{\theta}_t)$ from current OC system $\Sigma(\boldsymbol{\theta}_t)$;

  Obtain $\dfrac{\partial \boldsymbol{\xi}_t(\boldsymbol{\theta}_t)}{\partial \boldsymbol{\theta}_t}$ with GG in Algorithm 1;

  Obtain $\dfrac{\partial \boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}_t), \boldsymbol{O}_t)}{\partial \boldsymbol{\xi}_t(\boldsymbol{\theta}_t)}$ from $\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}_t), \boldsymbol{O}_t)$;

  Obtain $\boldsymbol{L}_t$ via the chain rule (10);

  Update $\boldsymbol{\theta}_t$ using the estimator (9);

3.4 Convergence Analysis

This subsection presents the convergence analysis of the online parameter estimator. The analysis employs a candidate Lyapunov function and introduces how the covariance matrices $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ affect the convergence of the cumulative loss $L(\boldsymbol{\xi}(\boldsymbol{\theta}))$. In this section, for brevity, the stage loss $\boldsymbol{l}(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{O}_t)$ is written as $\boldsymbol{l}(\boldsymbol{\theta})$. Suppose for a specific task, there is an unknown true parameter $\boldsymbol{\theta}_t$, where $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1}$ for all t, and $L(\boldsymbol{\xi}(\boldsymbol{\theta}_t)) = 0$. Then, we define the estimation error as $\tilde{\boldsymbol{\theta}}_t = \boldsymbol{\theta}_t - \hat{\boldsymbol{\theta}}_t$. Furthermore, we define

$$\begin{aligned} &\text{Measurement error: } \boldsymbol{e}_t = \boldsymbol{l}(\boldsymbol{\theta}_t) - \boldsymbol{l}(\hat{\boldsymbol{\theta}}_t^-) \\ &\text{Prediction error: } \tilde{\boldsymbol{\theta}}_t^- = \boldsymbol{\theta}_t - \hat{\boldsymbol{\theta}}_t^-. \end{aligned} \tag{17}$$

To perform the convergence analysis, a candidate Lyapunov function is employed:

$$V_t = \tilde{\boldsymbol{\theta}}_t' \boldsymbol{P}_t^{-1} \tilde{\boldsymbol{\theta}}_t. \tag{18}$$

The goal here is to determine conditions for which the candidate Lyapunov function $\{V_t\}_{t=1,2,\dots}$ is a decreasing sequence, i.e. $V_{t+1} - V_t \leq 0, \forall t$. For rigorous analysis of the candidate Lyapunov function, as proposed in Boutayeb et al. (1997), unknown diagonal matrices $\boldsymbol{\alpha}_t \in \mathbb{R}^{r\times r}$ and $\boldsymbol{\beta}_t \in \mathbb{R}^{p\times p}$ are introduced to model the measurement and prediction error defined in (17):

$$\boldsymbol{\alpha}_t \boldsymbol{e}_t = \boldsymbol{L}_t \tilde{\boldsymbol{\theta}}_t^-, \quad \tilde{\boldsymbol{\theta}}_t^- = \beta_t \tilde{\boldsymbol{\theta}}_{t-1}. \tag{19}$$

To ensure convergence of the proposed estimator, the following assumptions need to be made.

Assumption 1. *The derivative* $\boldsymbol{L}_t = \dfrac{d\boldsymbol{l}(\boldsymbol{\theta})}{d\boldsymbol{\theta}}$ *is of full rank for every* $\boldsymbol{\theta}$.

Remark 1. *The discrete-time dynamical system (8) satisfies the observability rank condition, i.e., for every* $\boldsymbol{\theta}$, $\text{rank}(\text{col}\{\dfrac{d\boldsymbol{l}(\boldsymbol{\theta})}{d\boldsymbol{\theta}}, \dfrac{d\boldsymbol{l}(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\boldsymbol{I}_p, \cdots, \dfrac{d\boldsymbol{l}(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\boldsymbol{I}_p^{p-1}\}) = p$ *(Song & Grizzle, 1992). That means if Assumption 1 is satisfied for every* $\boldsymbol{\theta}$*, the system (8) is observable for every* $\boldsymbol{\theta}$*. The observability condition assures that* $\boldsymbol{P}_t$ *is a bounded matrix from above and below (Song & Grizzle, 1992; Boutayeb & Aubry, 1999).*

As common in the EKF analysis, we adopt the following assumption:

Assumption 2. $\boldsymbol{L}_t$ *is a uniformly bounded matrix.*

We have the following lemma to show how the covariance matrices $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ affect the convergence of the tunable parameter. The proof can be found in Appendix A.

Lemma 2. *Let Assumptions 1 and 2 hold. If the matrices* $\boldsymbol{R}_t$ *and* $\boldsymbol{Q}_t$ *satisfy the following inequalities:*

$$(\boldsymbol{\alpha}_t - \boldsymbol{I}_s)^2 \leq \boldsymbol{R}_t (\boldsymbol{L}_t \boldsymbol{P}_t^- \boldsymbol{L}_t' + \boldsymbol{R}_t)^{-1}, \tag{20}$$

$$\boldsymbol{\beta}_t' (\boldsymbol{P}_t + \boldsymbol{Q}_t)^{-1} \boldsymbol{\beta}_t - \boldsymbol{P}_t^{-1} \leq 0, \tag{21}$$

Then the proposed estimator (9), when used as an observer for the system (8), ensures local asymptotic convergence, i.e. $\lim_{t\to\infty} \tilde{\boldsymbol{\theta}}_t = \boldsymbol{0}$.

Remark 2. *Lemma 2 provides sufficient conditions for the convergence of $\hat{\boldsymbol{\theta}}$. As the diagonal matrices $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ are unknown, one can design the matrices $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ to satisfy inequalities (20) and (21). For example, one can set the matrix $\boldsymbol{Q}_t$ to be sufficiently large so that (21) is satisfied, which means the parameter estimator can tolerate arbitrary large initial state estimation error, i.e. large $\boldsymbol{\beta}_t$. It is worth to note that as long as (20) and (21) are satisfied, $\hat{\boldsymbol{\theta}}_t$ converges to $\boldsymbol{\theta}_t$ and consequently $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ become identity matrix. In the case when there is no noise, i.e. $\boldsymbol{Q}_t = \mathbf{0}_{p\times p}$ and $\boldsymbol{R}_t = \mathbf{0}_{s\times s}$, $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ can only be identity matrices to satisfy the inequalities (20) and (21), indicating the convergence of $\hat{\boldsymbol{\theta}}_t$ to $\boldsymbol{\theta}_t$.*

Remark 3. *Equation (20) and (21) indicate one of the limitations of the estimator, which is the selection of initial guess. If the initial guess of $\boldsymbol{\theta}$ results in $\boldsymbol{\alpha}_0$ and $\boldsymbol{\beta}_0$ that does not satisfy (20) and (21), the value of the Lyapunov function (18) becomes larger, which leads to even larger $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$, causing the estimation to fail.*

We have the following main theorem show how the covariance matrix $\boldsymbol{R}_t$ and $\boldsymbol{Q}_t$ affect the convergence of cumulative loss $L(\boldsymbol{\xi}(\boldsymbol{\theta}_t))$ by utilizing the inequalities introduced in Lemma 2. The proof can be found in Appendix B.

Theorem 1. *Let Assumptions 1 and 2 hold. If $\boldsymbol{R}_t$ and $\boldsymbol{Q}_t$ satisfy the following inequalities:*

$$(\boldsymbol{\alpha}_t - \boldsymbol{I}_s)^2 \leq \boldsymbol{R}_t(\boldsymbol{L}_t\boldsymbol{P}_t^-\boldsymbol{L}_t' + \boldsymbol{R}_t)^{-1}, \quad \boldsymbol{\beta}_t'(\boldsymbol{P}_t + \boldsymbol{Q}_t)^{-1}\boldsymbol{\beta}_t - \boldsymbol{P}_t^{-1} \leq 0, \tag{22}$$

then finding $\boldsymbol{\theta}$ with the proposed estimator (9) employing the gradient generator in (14)-(15) ensures local asymptotic convergence of the cumulative loss L in (3) to 0 , i.e. $\lim_{t\to\infty} L(\boldsymbol{\xi}(\boldsymbol{\theta}_t)) = 0$.

To evaluate the effectiveness of online algorithms, the regret analysis is often adopted (Li et al., 2021; Zinkevich, 2003; Hazan et al., 2007). In our case, the regret at time T is defined as:

$$\text{Regret}_T = \textstyle\sum_{t=1}^{T}(\|\boldsymbol{l}(\hat{\boldsymbol{\theta}}_t)\|^2 - \|\boldsymbol{l}(\boldsymbol{\theta}_t)\|^2), \tag{23}$$

which represents the accumulative performance discrepancy between $\hat{\boldsymbol{\theta}}_t$ and $\boldsymbol{\theta}_t$. Assume $\|\boldsymbol{l}(\boldsymbol{\theta})\|^2$ is convex with respect to $\boldsymbol{\theta}$. As Lemma 2 states, the sufficient conditions (20) and (21) are satisfied for all t, then $\hat{\boldsymbol{\theta}}_t$ will converge to $\boldsymbol{\theta}_t$ asymptotically. Furthermore, according to (19), $\boldsymbol{e}_t$ will converge to zero, which indicates that $\|\boldsymbol{l}(\hat{\boldsymbol{\theta}}_t)\|^2$ will converge to $\|\boldsymbol{l}(\boldsymbol{\theta}_t)\|^2$ asymptotically. Therefore, we can say that the regret in (23) will converge to a bounded value as t goes to infinity. Please refer to C in Appendix for details.

4 Applications to Different Online Learning Modes and Experiments

This section demonstrates the capability of the proposed OCIL framework with its three modes by three applications, Online Imitation Learning, Online System Identification, and Learning Policy on-the-fly. This section includes a performance comparison with some state-of-the-art frameworks for three environments that are summarized in Table 1. For every environment, the unknown parameter is static, which means $\boldsymbol{Q}_t = \mathbf{0}_{p\times p}$. The covariance matrix of measurement noise is set to be $\boldsymbol{R}_t = 10^{-8}$ for OCIL, and $\boldsymbol{R}_t = 0$ for other methods as they fail to complete the tasks with measurement noise. To highlight the flexibility of OCIL, each experiment includes two phases: 1) online phase, where OCIL keeps learning the unknown parameter while new data comes in before the final time T; 2) offline phase, where OCIL keeps learning the parameter given the learned parameter at time T and the entire trajectory obtained from time $t = 0$ to time T. For each environment and task, a terminal time $T \in \mathbb{Z}$ is defined to represent a desired time duration where the system shall finish the task.

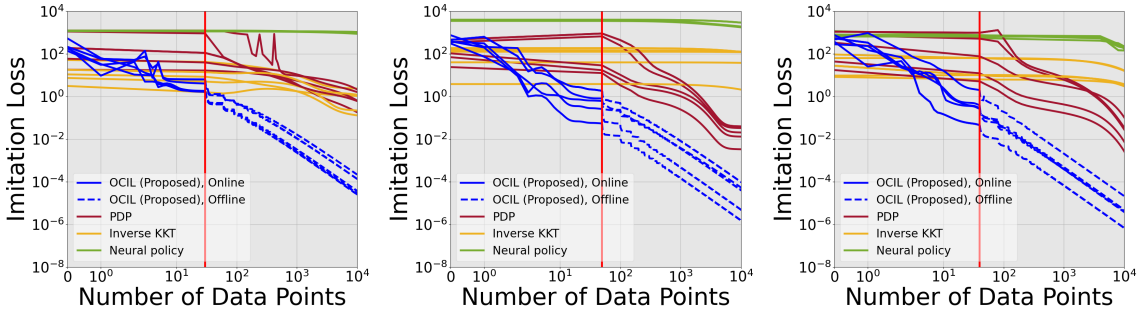
To unify the data visualization of both online and offline phases, the horizontal axis represents the number of data points, where a vertical red line corresponds to the final time T, i.e. the end of the online phase. The number of data points reflects the number of iterations multiplied by the total number of time steps for each iteration. The solid blue curves indicate the online portion of OCIL, whereas the dashed blue curves indicate the offline portion. For every environment and every method, 5 trials are performed given random initial conditions due to the high computational cost for other methods. The computational performance and analysis for OCIL are shown in Section 5 of the Appendix.

Online Imitation Learning. The control objective is parameterized as a weighted distance to the goal. Set the stage loss of imitation learning $l(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{y}_t^*) = \boldsymbol{y}_t^* - \boldsymbol{g}(\boldsymbol{x}_t(\boldsymbol{\theta}), \boldsymbol{u}_t(\boldsymbol{\theta}))$. Four existing methods are used for

Table 1: Experiment Environments

Systems	Dynamics parameter $\boldsymbol{\theta}_{dyn}$	Objective parameter $\boldsymbol{\theta}_{obj}$
Cartpole 6-DoF Quadrotor 6-DoF Rocket	cart mass, pole mass and length mass, wing length, inertia matrix mass, rocket length, inertia matrix	$c(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{\theta}_{obj}\|\boldsymbol{x} - \boldsymbol{x}_g\|^2 + \|\boldsymbol{u}\|^2$ $h(\boldsymbol{x}) = \boldsymbol{\theta}_{obj}\|\boldsymbol{x} - \boldsymbol{x}_g\|^2$

comparisons: (i) inverse KKT (Englert et al., 2017) (ii) neural policy cloning (Bojarski et al., 2016) and (iii) PDP (Jin et al., 2020). These methods don't handle measurement noise well because of their limitations, so we performed the experiments without including measurement noise for these methods. Fig. 2a-2c summarize the comparison result, where OCIL converges faster and obtains lower loss than the other offline methods, in both online and offline phases. The initial loss for each method is different because the learning representation (parameterization) is different. Thus, it is hard to guarantee that an initial neural network has the same loss as another initial parameter vector. Nevertheless, the initial representation of each method is adjusted such that OCIL does not take advantage of good initialization. Fig. 2a-2c validate the effectiveness of OCIL's both online and offline performance, even with measurement noise.



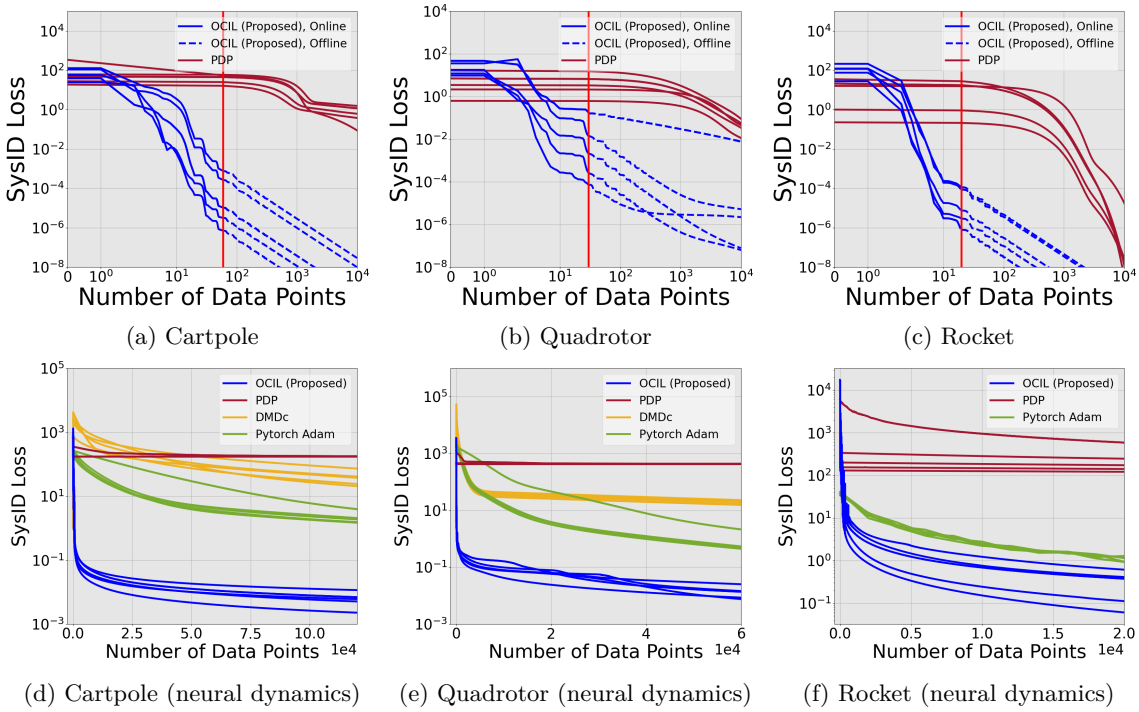
(a) Cartpole (b) Quadrotor (c) Rocket

Figure 2: Imitation loss v.s. number of data points

Online System Identification The stage loss is set to be $l(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{\xi}_t^o) = \boldsymbol{\xi}_t^o - \boldsymbol{\xi}_t(\boldsymbol{\theta})$. Three other methods are used for comparison: (i) Pytorch Adam solver, (ii) DMDc and (iii) PDP. No measurement noise are injected into observed data for existing methods due to their inherent limitations. Fig. 3a-3c summarize the result, where OCIL outperforms PDP for faster convergence and lower loss, in both online and offline phases. Different than Online Imitation Learning, OCIL does not decrease its SysID loss significantly at first because the number of data points is not sufficient for online learning. Once the number of data points becomes sufficient, the SysID loss starts decreasing significantly. This phenomenon can also be observed in the other methods, but their critical number of data points is significantly larger than OCIL's. In Fig. 3d-3f, OCIL and other methods are applied to learn the neural dynamics using the same observed trajectory. It can be seen that OCIL outperforms other methods for lower loss. Fig. 4 demonstrates the capability of OCIL dealing with neural dynamics that has different size of NN.

Policy Tuning On-the-fly. The parameterized OC system in 6 is used here, where the policy is in a state-feedback form and parameterized by the tunable parameter $\boldsymbol{\theta}$. The stage loss is set to be $l(\boldsymbol{\xi}_t(\boldsymbol{\theta}), \boldsymbol{\xi}_t^*) = \boldsymbol{\xi}_t^* - \boldsymbol{\xi}_t(\boldsymbol{\theta})$. where $\boldsymbol{\xi}_t^*$ is the trajectory that needs to be tracked. Other methods are used for comparison (i) iLQR (ii) GPS, and (iii) PDP. No measurement noise is included for existing methods due to their limitations. Fig. 5a-5c summarize the result, where the loss and its variation of OCIL converge very quickly. The buffers in Fig. 5a-5f indicate 3 times of standard deviation. Fig. 5d-5f presents the online phase of OCIL given 1000 random trials, which further validates the effectiveness and robustness of OCIL given measurement noise.

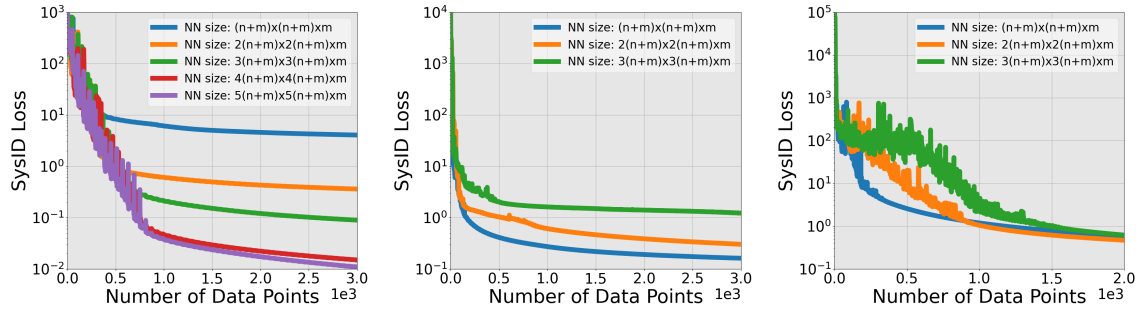
In general, OCIL from all figures does not have a smooth loss trajectory as the other offline methods. This is because at the online phase, an optimal gain matrix $\boldsymbol{K}_t$ from (9a) is computed to update $\boldsymbol{\theta}$, whereas the other methods either use a constant or iteration-dependent step size. The optimal gain is conceptually similar



(a) Cartpole (b) Quadrotor (c) Rocket

(d) Cartpole (neural dynamics) (e) Quadrotor (neural dynamics) (f) Rocket (neural dynamics)

Figure 3: SysID loss v.s. number of data points



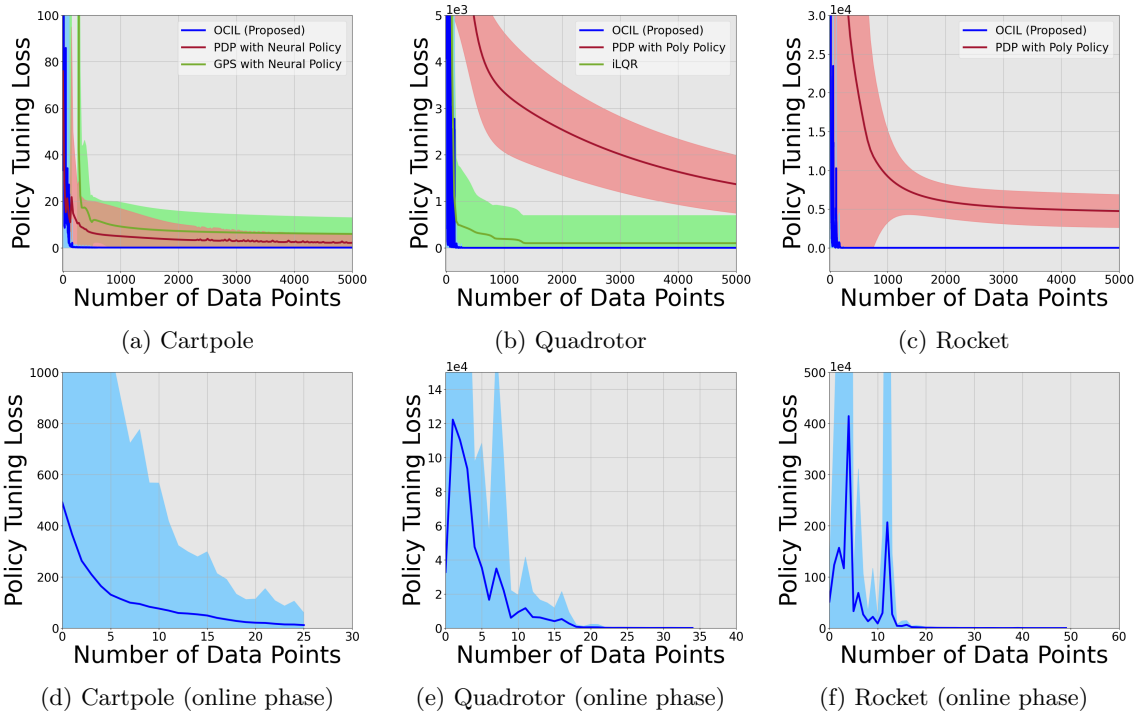
(a) Cartpole (b) Quadrotor (c) Rocket

Figure 4: SysID Loss v.s. number of data points with different size of neural dynamics

to searching an optimal step size in the line-search optimization algorithms. Thus, it is observed that the loss variation, as represented by blue buffers, is relatively high initially but starts decreasing significantly as new data comes in because $\boldsymbol{K}_t$ is continually updated. In contrast, the loss variation barely changes for the other offline methods after some data points.

5 Real-time Computational Performance

The experiments with OCIL were performed on a desktop with one Intel Core i7-8700k CPU with 8GB RAM. No GPU was used. The experiments with other methods were performed on a desktop with one AMD Ryzen 9 5900X CPU, one Nvidia Geforce RTX 4070ti, and 32 GB RAM. A more powerful PC was selected for the other methods because of their high computational cost. As noted at the beginning of Section 4, only 5 trials were conducted due to the computational expense.



(a) Cartpole (b) Quadrotor (c) Rocket

(d) Cartpole (online phase) (e) Quadrotor (online phase) (f) Rocket (online phase)

Figure 5: Policy Tuning Loss v.s. number of data points. Buffers represent loss variation under 3σ with random initial conditions.

To demonstrate that the computational performance of OCIL is enough to be used in an online fashion, we recorded the computational time for OCIL in different modes for 100 trials. Table 3 summarizes the OCIL's computational performance for the system identification task for three environments, where OCIL Time indicates the computational time of running OCIL at each time t, i.e. the iteration within the for-loop of Algorithm 2; GG Time indicates the computational time of running gradient generator (GG) at each time t, i.e. Algorithm 1; Estimator Time indicates the computational time of updating $\hat{\boldsymbol{\theta}}$, i.e. Line 7 of Algorithm 2; Δ indicates the time step of each environment, i.e., the time duration between two consecutive data measurements or the maximum allowed time duration of online algorithms to perform computation; Percentage indicates the percentage of the average OCIL time with respect to Δ. The header of Table 4 and Table 2 are the same. Roughly speaking, OCIL time = GG time + Estimator time + Optimal Control computation time.

Table 3 illustrates that OCIL can estimate the dynamical system with neural network representation in real time, within the system frequency of getting new data. Table 4 illustrates that OCIL can tune the neural policy in real time. As indicated in Line 2 of Algorithm 2, the most computationally heavy part is solving optimal control trajectory in real time, instead of GG and the parameter estimator. As demonstrated at the beginning of this section, OCIL does not require huge computational resources, such as GPU. Therefore, this paper believes that OCIL has the capability to run in an online fashion.

Table 2: Computational Performance for Online Imitation Learning

Env.	OCIL Time [ms]	GG Time [ms]	Estimator Time [ms]	Δ [ms]	Percentage
Cartpole	62.10 ± 6.63	7.47 ± 0.25	0.031 ± 0.0023	100	62.10 %
Quadrotor	81.70 ± 2.51	21.72 ± 0.84	0.058 ± 0.039	100	81.70 %
Rocket	72.25 ± 13.91	19.77 ± 6.26	0.060 ± 0.012	100	72.25%

Table 3: Computational Performance for SysID with Neural System

Env.	OCIL Time [ms]	GG Time [ms]	Estimator Time [ms]	Δ [ms]	Percentage
Cartpole	17.18 ± 5.15	6.05 ± 2.59	1.93 ± 0.85	50	34.36 %
Quadrotor	35.53 ± 8.98	12.18 ± 4.77	16.18 ± 6.11	100	35.53 %
Rocket	29.54 ± 8.29	11.25 ± 4.67	12.89 ± 5.29	200	14.77 %

Table 4: Computational Performance for Policy Tuning with Neural Policy

Env.	OCIL Time [ms]	GG Time [ms]	Estimator Time [ms]	Δ [ms]	Percentage
Cartpole	16.41 ± 5.35	7.72 ± 3.07	3.96 ± 1.91	50	32.82 %
Quadrotor	62.67 ± 9.51	30.86 ± 2.25	22.47 ± 8.33	100	62.67 %
Rocket	59.02 ± 7.25	33.99 ± 2.98	12.94 ± 5.62	100	59.02 %

6 Limitations.

This section discusses the major limitations of the proposed framework from three perspectives.

Local convergence: Since OCIL is based on first-order gradient, it can only achieve local minima for general non-convex optimization problems in (3). Furthermore, the general problem in (3) belongs to a bi-level optimization framework. Under certain assumptions such as convexity and smoothness on models (e.g., dynamics model, policy, loss function, and control objective function), global convergence of the bi-level optimization can be established. However, this paper believes that such conditions are too restrictive in the context of dynamical control systems. Therefore, this paper believes that the local convergence analysis based on general nonlinear optimization is enough.

Parameterization matters for global convergence: When performing experiments, we find that how models are parameterized matters for good convergence performance. For example, in online SysID mode, we observe that using a neural network dynamics (in Fig. 3d-3f) is more likely to get trapped in local minima than using the true dynamics with unknown parameters (in Fig. 3a-3c)). In general, more complex parameterization will bring extreme non-convexity to the optimization problem, making the algorithm more easily trapped in local minima. Determining the parameterization of an object to be learned requires prior or expert knowledge, which is common in ML.

Initialization matters: As OCIL borrows how optimal gain updates from EKF, they share the same drawback that convergence depends on the selection of initialization. As shown in Remark 3, a bad initial guess might cause the estimator to diverge according to Lemma 2. Therefore, if a relatively good initial guess is hard to retrieve, one might need to use other methods to cold start OCIL.

7 Conclusions

This paper proposes Online Control-Informed Learning (OCIL), an online learning method tailored for diverse learning tasks. By considering an optimal control system tunable through a tunable parameter, OCIL effectively addresses tasks such as online imitation learning, online system identification, and tuning policy on-the-fly. By designing a stage loss specific to each task and treating the tunable parameter as a state of a new system, we employ the online parameter estimator to estimate the parameter in real-time and minimize loss at each time step. Theoretical analysis establishes the convergence conditions for OCIL, while experiments on various environments, tasks, and existing methods are done to validate its data efficiency, versatility, and robustness against measurement noise.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 1–8, 2004.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11254–11263, 2019.

Martin Benning, Elena Celledoni, Matthias J Ehrhardt, Brynjulf Owren, and Carola-Bibiane Schönlieb. Deep learning as optimal control problems: models and numerical methods. *arXiv preprint arXiv:1904.05657*, 2019.

Dimitri P. Bertsekas. Incremental least squares methods and the extended kalman filter. *SIAM Journal on Optimization*, 6(3):807–822, 1996. doi: 10.1137/S1052623494268522.

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.

M. Boutayeb and D. Aubry. A strong tracking extended kalman observer for nonlinear discrete-time systems. *IEEE Transactions on Automatic Control*, 44(8):1550–1556, 1999. doi: 10.1109/9.780419.

M. Boutayeb, H. Rafaralahy, and M. Darouach. Convergence analysis of the extended kalman filter used as an observer for nonlinear deterministic discrete-time systems. *IEEE Transactions on Automatic Control*, 42(4):581–586, 1997. doi: 10.1109/9.566674.

Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pp. 6571–6583, 2018.

Wojciech M Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1331–1340. PMLR, 2019.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 465–472, 2011.

Peter Englert, Ngo Anh Vien, and Marc Toussaint. Inverse kkt: Learning cost functions of manipulation tasks from demonstrations. *The International Journal of Robotics Research*, 36(13-14):1474–1488, 2017.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, pp. 64–72, 2016.

Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *IEEE International Conference on Computer Vision*, pp. 4346–4354, 2015.

Reza Ghaemi, Jing Sun, and Ilya V Kolmanovsky. Neighboring extremal solution for nonlinear discrete-time optimal control problems with state inequality constraints. *IEEE Transactions on Automatic Control*, 54 (11):2674–2679, 2009.

Jiequn Han and Weinan E. Deep learning approximation for stochastic control problems. *Deep Reinforcement Learning Workshop, Advances in Neural Information Processing Systems*, 2016.

Jiequn Han, Qianxiao Li, et al. A mean-field optimal control formulation of deep learning. *Research in the Mathematical Sciences*, 6(1):1–41, 2019.

Wenjian Hao, Paulo C Heredia, Bowen Huang, Zehui Lu, Zihao Liang, and Shaoshuai Mou. Policy learning based on deep koopman representation. *arXiv preprint arXiv:2305.15188*, 2023.

Masahiko Haruno, Daniel M Wolpert, and Mitsuo Kawato. Mosaic model for sensorimotor learning and control. *Neural Computation*, 13(10):2201–2220, 2001.

Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems*, 29, 2016.

Wanxin Jin and Shaoshuai Mou. Distributed inverse optimal control. *Automatica*, 129:109658, 2021. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2021.109658. URL `https://www.sciencedirect.com/science/article/pii/S0005109821001783`.

Wanxin Jin, Dana Kulić, Jonathan Feng-Shun Lin, Shaoshuai Mou, and Sandra Hirche. Inverse optimal control for multiphase cost functions. *IEEE Transactions on Robotics*, 35(6):1387–1398, 2019.

Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. Pontryagin differentiable programming: An end-to-end learning and control framework. *Advances in Neural Information Processing Systems*, 33: 7979–7992, 2020.

Wanxin Jin, Dana Kulić, Shaoshuai Mou, and Sandra Hirche. Inverse optimal control from incomplete trajectory observations. *The International Journal of Robotics Research*, 40(6-7):848–865, 2021a.

Wanxin Jin, Shaoshuai Mou, and George J Pappas. Safe pontryagin differentiable programming. *Advances in Neural Information Processing Systems*, 34:16034–16050, 2021b.

Wanxin Jin, Todd D Murphey, Dana Kulić, Neta Ezer, and Shaoshuai Mou. Learning from sparse demonstrations. *IEEE Transactions on Robotics*, 39(1):645–664, 2022a.

Wanxin Jin, Todd D Murphey, Zehui Lu, and Shaoshuai Mou. Learning from human directional corrections. *IEEE Transactions on Robotics*, 39(1):625–644, 2022b.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Karel J Keesman. *System identification: an introduction*. Springer Science & Business Media, 2011.

Arezou Keshavarz, Yang Wang, and Stephen Boyd. Imputing a convex objective function. In *IEEE International Symposium on Intelligent Control*, pp. 613–619, 2011.

Beomjoon Kim, Amir-massoud Farahmand, Joelle Pineau, and Doina Precup. Learning from limited demonstrations. *Advances in Neural Information Processing Systems*, 26, 2013.

Jack B Kuipers. *Quaternions and rotation sequences*, volume 66. Princeton University Press, 1999.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pp. 1071–1079, 2014.

Qianxiao Li and Shuji Hao. An optimal control approach to deep learning and applications to discrete-weight neural networks. *arXiv preprint arXiv:1803.01299*, 2018.

Qianxiao Li, Long Chen, Cheng Tai, and E Weinan. Maximum principle based algorithms for deep learning. *Journal of Machine Learning Research*, 18(165):1–29, 2018.

Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *International Conference on Informatics in Control, Automation and Robotics*, pp. 222–229, 2004.

Yingying Li, Guannan Qu, and Na Li. Online optimization with predictions and switching costs: Fast algorithms and the fundamental limit. *IEEE Transactions on Automatic Control*, 66(10):4761–4768, 2021. doi: 10.1109/TAC.2020.3040249.

Zihao Liang, Wanxin Jin, and Shaoshuai Mou. An iterative method for inverse optimal control. In *2022 13th Asian Control Conference (ASCC)*, pp. 959–964, 2022. doi: 10.23919/ASCC56756.2022.9828009.

Zihao Liang, Wenjian Hao, and Shaoshuai Mou. A data-driven approach for inverse optimal control. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 3632–3637, 2023. doi: 10.1109/CDC49753.2023.10383220.

Hailiang Liu and Peter Markowich. Selection dynamics for deep neural networks. *Journal of Differential Equations*, 269(12):11540–11574, 2020.

Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*, 2019.

Alexandre Mauroy, Y Susuki, and Igor Mezić. *Koopman operator in systems and control*. Springer, 2020.

Katja Mombaur, Anh Truong, and Jean-Paul Laumond. From human to humanoid locomotion—an inverse optimal control approach. *Autonomous Robots*, 28(3):369–383, 2010.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, pp. 3878–3887. PMLR, 2018.

Melkior Ornik, Steven Carr, Arie Israel, and Ufuk Topcu. Control-oriented learning on the fly. *IEEE Transactions on Automatic Control*, 65(11):4800–4807, 2019.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *International Conference on Machine Learning*, pp. 729–736, 2006.

Steindor Saemundsson, Alexander Terenin, Katja Hofmann, and Marc Deisenroth. Variational integrator networks for physically structured embeddings. In *International Conference on Artificial Intelligence and Statistics*, pp. 3078–3087, 2020.

Fumihiro Sasaki and Ryota Yamashina. Behavioral cloning from noisy demonstrations. In *International Conference on Learning Representations*, 2021.

Stefan Schaal. Learning from demonstration. *Advances in Neural Information Processing Systems*, 9, 1996.

Yongkyu Song and Jessy W. Grizzle. The extended kalman filter as a local asymptotic observer for nonlinear discrete-time systems. In *1992 American Control Conference*, pp. 3365–3369, 1992. doi: 10.23919/ACC.1992.4792775.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

Laura Von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, et al. Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):614–633, 2021.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, pp. 2746–2754, 2015.

Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6): 1307–1346, 2015.

Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, pp. 6818–6827. PMLR, 2019.

Amy Zhang, Sainbayar Sukhbaatar, Adam Lerer, Arthur Szlam, and Rob Fergus. Composable planning with attributes. In *International Conference on Machine Learning*, pp. 5842–5851, 2018.

Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Painless adversarial training using maximal principle. *arXiv preprint arXiv:1905.00877*, 2019.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. *arXiv preprint arXiv:1909.12077*, 2019.

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

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 928–936, 2003.

Appendix

A Proof of Lemma 2

Since the matrices $\boldsymbol{P}_t$ and $\boldsymbol{L}_t$ are bounded according to Assumption 1 and 2, from (9b), one will have:

$$\begin{aligned}\boldsymbol{K}_t &= \boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1} &(24)\\ &= \boldsymbol{P}_t^-\boldsymbol{L}_t'(\boldsymbol{L}_t\boldsymbol{P}_t^-\boldsymbol{L}_t' + \boldsymbol{R}_t)^{-1}. &(25)\end{aligned}$$

Then, by taking the inverse of (24) and (25), one will get:

$$\boldsymbol{P}_t^{-1} = (\boldsymbol{P}_t^-)^{-1} + \boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t. \tag{26}$$

Substituting (24) into (9b) and subtracting both sides from $\boldsymbol{\theta}_t$, one will have:

$$\tilde{\boldsymbol{\theta}}_t = \tilde{\boldsymbol{\theta}}_t^- - \boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{e}_t. \tag{27}$$

Then, plug (27) into the Lyapunov function (18):

$$\begin{aligned}V_t &= \tilde{\boldsymbol{\theta}}_t'\boldsymbol{P}_t^{-1}\tilde{\boldsymbol{\theta}}_t &(28)\\ &= (\tilde{\boldsymbol{\theta}}_t^- - \boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{e}_t)'\boldsymbol{P}_t^{-1}(\tilde{\boldsymbol{\theta}}_t^- - \boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{e}_t) &(29)\\ &= (\tilde{\boldsymbol{\theta}}_t^-)'\boldsymbol{P}_t^{-1}\tilde{\boldsymbol{\theta}}_t^- - (\tilde{\boldsymbol{\theta}}_t^-)'\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{e}_t - \boldsymbol{e}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t\tilde{\boldsymbol{\theta}}_t^- + \boldsymbol{e}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t\boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{e}_t &(30)\end{aligned}$$

Next, we plug (26) into (30):

$$\begin{aligned}V_t &= (\tilde{\boldsymbol{\theta}}_t^-)'((\boldsymbol{P}_t^-)^{-1} + \boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t)\tilde{\boldsymbol{\theta}}_t^- - (\tilde{\boldsymbol{\theta}}_t^-)'\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{e}_t - \boldsymbol{e}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t\tilde{\boldsymbol{\theta}}_t^- + \boldsymbol{e}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t\boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{e}_t &(31)\\ &= V_t^- + (\tilde{\boldsymbol{\theta}}_t^-)'\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t\tilde{\boldsymbol{\theta}}_t^- - (\tilde{\boldsymbol{\theta}}_t^-)'\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{e}_t - \boldsymbol{e}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t\tilde{\boldsymbol{\theta}}_t^- + \boldsymbol{e}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t\boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{e}_t, &(32)\end{aligned}$$

where

$$\begin{aligned}V_t^- &= (\tilde{\boldsymbol{\theta}}_t^-)'(\boldsymbol{P}_t^-)^{-1}\tilde{\boldsymbol{\theta}}_t^- &(33)\\ &= (\tilde{\boldsymbol{\theta}}_{t-1})'\boldsymbol{\beta}_t'(\boldsymbol{P}_{t-1} + \boldsymbol{Q}_{t-1})^{-1}\boldsymbol{\beta}_t\tilde{\boldsymbol{\theta}}_{t-1}. &(34)\end{aligned}$$

Using (19), (32) becomes:

$$\begin{aligned}V_t &= V_t^- + (\tilde{\boldsymbol{\theta}}_t^-)'\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t\tilde{\boldsymbol{\theta}}_t^- - (\tilde{\boldsymbol{\theta}}_t^-)'\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{e}_t - \boldsymbol{e}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t\tilde{\boldsymbol{\theta}}_t^- + \boldsymbol{e}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t\boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{e}_t &(35)\\ &= V_t^- + \boldsymbol{e}_t'\boldsymbol{\alpha}_t\boldsymbol{R}_t^{-1}\boldsymbol{\alpha}_t\boldsymbol{e}_t - \boldsymbol{e}_t'\boldsymbol{\alpha}_t\boldsymbol{R}_t^{-1}\boldsymbol{e}_t - \boldsymbol{e}_t'\boldsymbol{R}_t^{-1}\boldsymbol{\alpha}_t\boldsymbol{e}_t + \boldsymbol{e}_t'\boldsymbol{R}_t^{-1}\boldsymbol{L}_t\boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1}\boldsymbol{e}_t &(36)\\ &= V_t^- + \boldsymbol{e}_t'(\boldsymbol{\alpha}_t\boldsymbol{R}_t^{-1}\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t\boldsymbol{R}_t^{-1} - \boldsymbol{R}_t^{-1}\boldsymbol{\alpha}_t + \boldsymbol{R}_t^{-1}\boldsymbol{L}_t\boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1})\boldsymbol{e}_t. &(37)\end{aligned}$$

To ensure that the Lyapunov function $\{V_t\}_{t=1,2,\ldots}$ is a decreasing sequence, $V_t - V_{t-1} \leq 0$.

$$\begin{aligned}&V_t - V_{t-1} &(38)\\ &\quad= \boldsymbol{e}_t'(\boldsymbol{\alpha}_t\boldsymbol{R}_t^{-1}\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t\boldsymbol{R}_t^{-1} - \boldsymbol{R}_t^{-1}\boldsymbol{\alpha}_t + \boldsymbol{R}_t^{-1}\boldsymbol{L}_t\boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1})\boldsymbol{e}_t &(39)\\ &\quad+ (\tilde{\boldsymbol{\theta}}_{t-1})'(\boldsymbol{\beta}_t'(\boldsymbol{P}_{t-1} + \boldsymbol{Q}_{t-1})^{-1}\boldsymbol{\beta}_t - \boldsymbol{P}_{t-1}^{-1})\tilde{\boldsymbol{\theta}}_{t-1} \leq 0. &(40)\end{aligned}$$

Therefore, to ensure the Lyapunov function is a decreasing sequence,

$$\boldsymbol{\alpha}_t\boldsymbol{R}_t^{-1}\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t\boldsymbol{R}_t^{-1} - \boldsymbol{R}_t^{-1}\boldsymbol{\alpha}_t + \boldsymbol{R}_t^{-1}\boldsymbol{L}_t\boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1} \leq 0, \tag{41}$$

and

$$\boldsymbol{\beta}_t'(\boldsymbol{P}_{t-1} + \boldsymbol{Q}_{t-1})^{-1}\boldsymbol{\beta}_t - \boldsymbol{P}_{t-1}^{-1} \leq 0. \tag{42}$$

With some manipulations:

$$\begin{aligned}(\boldsymbol{\alpha}_t - \boldsymbol{I}_s)\boldsymbol{R}_t^{-1}(\boldsymbol{\alpha}_t - \boldsymbol{I}_s) - \boldsymbol{R}_t^{-1} + \boldsymbol{R}_t^{-1}\boldsymbol{L}_t\boldsymbol{P}_t\boldsymbol{L}_t'\boldsymbol{R}_t^{-1} \leq 0, &(43)\\ (\boldsymbol{\alpha}_t - \boldsymbol{I}_s)\boldsymbol{R}_t^{-1}(\boldsymbol{\alpha}_t - \boldsymbol{I}_s) - \boldsymbol{R}_t^{-1}(\boldsymbol{I}_s - \boldsymbol{L}_t\boldsymbol{P}_t^-\boldsymbol{L}_t'(\boldsymbol{L}_t\boldsymbol{P}_t^-\boldsymbol{L}_t' + \boldsymbol{R}_t)^{-1}) \leq 0. &(44)\end{aligned}$$

By letting $\boldsymbol{I}_s = (\boldsymbol{L}_t \boldsymbol{P}_t^- \boldsymbol{L}_t' + \boldsymbol{R}_t)(\boldsymbol{L}_t \boldsymbol{P}_t^- \boldsymbol{L}_t' + \boldsymbol{R}_t)^{-1}$, we have

$$(\boldsymbol{\alpha}_t - \boldsymbol{I}_s)\boldsymbol{R}_t^{-1}(\boldsymbol{\alpha}_t - \boldsymbol{I}_s) - (\boldsymbol{L}_t \boldsymbol{P}_t^- \boldsymbol{L}_t' + \boldsymbol{R}_t)^{-1} \leq 0. \tag{45}$$

Since $\boldsymbol{\alpha}_t$ and $\boldsymbol{R}_t$ are diagonal matrices, we will have

$$\boldsymbol{R}_t^{-1}(\boldsymbol{\alpha}_t - \boldsymbol{I}_s)^2 - (\boldsymbol{L}_t \boldsymbol{P}_t^- \boldsymbol{L}_t' + \boldsymbol{R}_t)^{-1} \leq 0, \tag{46}$$

which at the end yields:

$$(\boldsymbol{\alpha}_t - \boldsymbol{I}_s)^2 \leq \boldsymbol{R}_t(\boldsymbol{L}_t \boldsymbol{P}_t^- \boldsymbol{L}_t' + \boldsymbol{R}_t)^{-1}, \tag{47}$$

therefore the proof is completed.

B Proof of Theorem 1

This proof is straightforward once Lemma 2 is provided. Consider the assumptions 1 and 2 are met, according to Lemma 2, with the exact gradient generated by the gradient generator in (14)-(15), $\lim_{t\to\infty} \tilde{\boldsymbol{\theta}}_t = 0$. As the estimated $\hat{\boldsymbol{\theta}}$ converges to the true $\boldsymbol{\theta}$, where the true parameter gives zero cumulative loss, the cumulative loss $L(\boldsymbol{\xi}(\hat{\boldsymbol{\theta}}))$ goes to 0.

C Scratch for Regret Analysis

In the regret analysis for online optimization algorithms Zinkevich (2003); Hazan et al. (2007), the regret rate is often determined by the step size used for their gradient descent, which is typically a constant or an iteration-dependent value. Such step size can make the sequence of regret over time a Cauchy sequence. Consequently, one can induce that the accumulative regret is bounded.

As for the proposed OCIL framework, as mentioned at the end of Section 4, the optimal gain matrix $\boldsymbol{K}_t$ is updated every time t as new data comes in. Then given the proposed estimate update (9b), $\boldsymbol{K}_t$ updates the parameter estimate $\hat{\boldsymbol{\theta}}_t$ by minimizing the loss function. Therefore, the combination $\boldsymbol{K}_t(\boldsymbol{0} - \boldsymbol{l}(\boldsymbol{\xi}_t(\hat{\boldsymbol{\theta}}_t^-))$ is conceptually similar to the product of the gradient and optimal step size, where the optimal step is size calculated by line search algorithms in the context of optimization. This statement is also verified by our experiments in Section 4.

To sum up, if 20 and 21 are satisfied for all t, $\boldsymbol{l}(\hat{\boldsymbol{\theta}}_t^-)$ diminishes as time increases, which indicates that the accumulative regret will converge to a bounded value.

D Experiment Details

D.1 System/Environment Setups

Cartpole. We consider the following continuous dynamics of the cartpole

$$\begin{bmatrix} \dot{p} \\ \ddot{p} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{p} \\ (F + \frac{m_p l \dot{\theta}^2 \sin(\theta)}{m_t}) - \frac{m_p l \ddot{\theta} \cos(\theta)}{m_t} \\ \dot{\theta} \\ \frac{g \sin(\theta) - \cos(\theta)(F + \frac{m_p l \dot{\theta}^2 \sin(\theta)}{m_t})}{l(\frac{4}{3} - \frac{m_p \cos(\theta)^2}{m_t})} \end{bmatrix}, \tag{48}$$

where $p \in \mathbb{R}$ is the horizontal displacement of the cart; $\theta \in \mathbb{R}$ is the pole angle; $F \in \mathbb{R}$ denotes the horizontal force applied to the cart which is between -1 and $+1$; $l \in \mathbb{R}$ is the length of the pole; $m_p, m_t \in \mathbb{R}$ are the masses of the pole and total cartpole, respectively. By defining the states and control inputs of the cartpole

$$\boldsymbol{x} \triangleq \begin{bmatrix} p & \dot{p} & \theta & \dot{\theta} \end{bmatrix}' \quad \text{and} \quad \boldsymbol{u} \triangleq F \tag{49}$$

respectively.

Quadrotor UAV. We consider a quadrotor UAV with the following dynamics

$$\begin{aligned} \dot{\boldsymbol{p}}_I &= \boldsymbol{v}_I, \\ m\dot{\boldsymbol{v}}_I &= m\boldsymbol{g}_I + \mathbf{F}_I, \\ \dot{\boldsymbol{q}}_{B/I} &= \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_B)\boldsymbol{q}_{B/I}, \\ J_B\dot{\boldsymbol{\omega}}_B &= \mathbf{M}_B - \boldsymbol{\omega} \times J_B\boldsymbol{\omega}_B. \end{aligned} \tag{50}$$

Here, the subscription $_B$ and $_I$ denote a quantity is expressed in the body frame and inertial (world) frame, respectively; m and $J_B \in \mathbb{R}^{3\times3}$ are the mass and moment of inertia with respect to body frame of the UAV, respectively. g is the gravitational constant ($g = 10\text{kg/m}^2$), $\boldsymbol{g}_I = [0, 0, g]'$. $\boldsymbol{p} \in \mathbb{R}^3$ and $\boldsymbol{v} \in \mathbb{R}^3$ are the position and velocity vector of the UAV; $\boldsymbol{\omega}_B \in \mathbb{R}^3$ is the angular velocity vector of the UAV; $\boldsymbol{q}_{B/I} \in \mathbb{R}^4$ is the unit quaternion Kuipers (1999) that describes the attitude of UAV with respect to the inertial frame; $\boldsymbol{\Omega}(\boldsymbol{\omega}_B)$ is defined as:

$$\boldsymbol{\Omega}(\boldsymbol{\omega}_B) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}, \tag{51}$$

$\mathbf{M}_B \in \mathbb{R}^3$ is the torque applied to the UAV; $\mathbf{F}_I \in \mathbb{R}^3$ is the force vector applied to the UAV center of mass. The total force magnitude $f = \|\mathbf{F}_I\| \in \mathbb{R}$ (along z-axis of the body frame) and torque $\mathbf{M}_B = [M_x, M_y, M_z]'$ are generated by thrust from four rotating propellers $[T_1, T_2, T_3, T_4]'$, their relationship can be expressed as:

$$\begin{bmatrix} f \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -l_w/2 & 0 & l_w/2 \\ -l_w/2 & 0 & l_w/2 & 0 \\ c & -c & c & -c \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}, \tag{52}$$

where l_w is the wing length of the UAV and c is a fixed constant. The state and input vectors of the UAV are defined as:

$$\begin{aligned} \boldsymbol{x} &\triangleq \begin{bmatrix} \boldsymbol{p}' & \boldsymbol{v}' & \boldsymbol{q}' & \boldsymbol{\omega}' \end{bmatrix}' \in \mathbb{R}^{13}, \\ \boldsymbol{u} &\triangleq \begin{bmatrix} T_1 & T_2 & T_3 & T_4 \end{bmatrix}' \in \mathbb{R}^4. \end{aligned} \tag{53}$$

Rocket. The rocket is treated as a rigid body subject to constant gravitational acceleration, $g_I \in \mathbb{R}^3$, and neglect aerodynamic forces. The vehicle is assumed to actuate a single gimbaled rocket engine to generate a thrust vector within a feasible range of magnitudes and gimbal angles. We assume that at the landing phase, the depletion of fuel is insignificant. Therefore, we omit the dynamics of rocket mass. The rocket has the following dynamics:

$$\begin{aligned} \dot{\boldsymbol{p}}_{\mathcal{I}} &= \boldsymbol{v}_{\mathcal{I}}, \\ \dot{\boldsymbol{v}}_{\mathcal{I}} &= \frac{1}{m}C_{\mathcal{I}/\mathcal{B}}\mathbf{T}_{\mathcal{B}} + \mathbf{g}_{\mathcal{I}}, \\ \dot{\boldsymbol{q}}_{\mathcal{B}/\mathcal{I}} &= \frac{1}{2}\Omega(\boldsymbol{\omega}_{\mathcal{B}})\boldsymbol{q}_{\mathcal{B}/\mathcal{I}}, \\ J_{\mathcal{B}}\dot{\boldsymbol{\omega}}_{\mathcal{B}} &= \mathbf{M}_{\mathcal{B}} - [\boldsymbol{\omega}_{\mathcal{B}}\times]J_{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}}. \end{aligned} \tag{54}$$

Here, the subscription $_{\mathcal{B}}$ and $_{\mathcal{I}}$ denote a quantity is expressed in the body frame and inertial (world) frame, respectively; m and $J_{\mathcal{B}} \in \mathbb{R}^{3\times3}$ are the mass and moment of inertia with respect to body frame of the rocket, respectively. $\boldsymbol{p} \in \mathbb{R}^3$ and $\boldsymbol{v} \in \mathbb{R}^3$ are the position and velocity vector of the rocket; $\boldsymbol{\omega}_{\mathcal{B}} \in \mathbb{R}^3$ is the angular velocity vector of the rocket; $\boldsymbol{q}_{\mathcal{B}/\mathcal{I}} = [q_0, q_1, q_2, q_3]$ is the unit quaternion that describes the attitude of rocket with respect to the inertial frame; $\mathbf{T}_{\mathcal{B}} \in \mathbb{R}^3$ is the commanded thrust vector; $\mathbf{M}_{\mathcal{B}} \in \mathbb{R}^3$ is the torque applied to the rocket; $C_{\mathcal{B}/\mathcal{I}}$ is the direction cosine matrix that encodes the attitude transformation from body frame to inertia frame and related to $\boldsymbol{q}_{\mathcal{B}/\mathcal{I}}$ by the following relationship:

$$C_{\mathcal{B}/\mathcal{I}} = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & 1-2(q_1^2+q_1^2) \end{bmatrix},$$

The inverse transformation is denoted as $C_{\mathcal{I}/\mathcal{B}} = C_{\mathcal{B}/\mathcal{I}}^T$;

The skew-symmetric matrices $[\boldsymbol{\omega}_{\mathcal{B}}\times]$ and $\Omega(\boldsymbol{\omega}_{\mathcal{B}})$ are defined as follow:

$$[\boldsymbol{\omega}_{\mathcal{B}}\times] \triangleq \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \quad \Omega(\boldsymbol{\omega}_{\mathcal{B}}) \triangleq \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix},$$

The state and input vectors of the rocket are defined as:

$$\begin{aligned} \boldsymbol{x} &= \begin{bmatrix} \boldsymbol{p}_{\mathcal{I}}' & \boldsymbol{v}_{\mathcal{I}}' & \boldsymbol{q}_{\mathcal{B}/\mathcal{I}}' & \boldsymbol{\omega}_{\mathcal{B}}' \end{bmatrix}' \in \mathbb{R}^{13}, \\ \boldsymbol{u} &= \mathbf{T}_{\mathcal{B}} = \begin{bmatrix} T_x & T_y & T_z \end{bmatrix}' \in \mathbb{R}^3, \end{aligned} \tag{55}$$

Discretization Discretization is done by the following discrete-time form

$$\boldsymbol{x}_{t+1} \approx \boldsymbol{x}_t + \Delta \cdot \boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{u}_t) \triangleq \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{56}$$

where Δ is the discretization interval.

D.2 Online Imitation Learning

Data acquisition. The dataset of expert demonstrations is generated by solving an optimal control system with the true dynamics and control objective parameter $\boldsymbol{\theta}^* = \{\boldsymbol{\theta}_{dyn}, \boldsymbol{\theta}_{obj}\}$ given. We generate a number of five trajectories, where different trajectories have different initial conditions $\boldsymbol{x}_0$ and time horizons T.

PDP. We employed the PDP in Jin et al. (2020) to solve this problem. The learning rate is set to be $\eta = 10^{-4}$. Five trials were run given random initial $\boldsymbol{\theta}_0$.

Inverse KKT method. We choose the inverse KKT method Englert et al. (2017) for comparison because it is suitable for learning objective functions for high-dimensional continuous-space systems. We adapt the inverse KKT method, and define the KKT loss as the norm-2 violation of the KKT condition by the demonstration data:

$$\min_{\boldsymbol{\theta}, \boldsymbol{\lambda}_{1:T}} \left(\left\| \frac{\partial L}{\partial \boldsymbol{x}_{0:T}}(\boldsymbol{x}^*_{0:T}, \boldsymbol{u}^*_{0:T-1}) \right\|^2 + \left\| \frac{\partial L}{\partial \boldsymbol{u}_{0:T-1}}(\boldsymbol{x}^*_{0:T}, \boldsymbol{u}^*_{0:T-1}) \right\|^2 \right).$$

Neural policy cloning. For the neural policy cloning, we directly learn a neural network policy $\boldsymbol{u} = \boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta})$ from the dataset using supervised learning, that is

$$\min_{\boldsymbol{\theta}} \sum_{t=0}^{T-1} \|\boldsymbol{u}_t^* - \boldsymbol{\mu}(\boldsymbol{x}_t^*, \boldsymbol{\theta})\|^2 \tag{57}$$

D.3 Online System Identification

Data acquisition. In the system identification experiment, we collect a total number of five trajectories from systems with dynamics known, wherein different trajectories $\boldsymbol{\xi}^o = \{\boldsymbol{x}^o_{0:T}, \boldsymbol{u}_{0:T-1}\}$ have different initial conditions $\boldsymbol{x}_0$ and horizons T (T ranges from 10 to 20 depending on different environment and task), with random inputs $\boldsymbol{u}_{0:T-1}$ drawn from uniform distribution.

PDP. We employed the PDP in Jin et al. (2020) to solve this problem. The learning rate is set to be $\eta = 10^{-4}$. Five trials were run given random initial $\boldsymbol{\theta}_0$. For the neural objective function case, the learning rate is set to be $\eta = 10^{-5}$.

Pytorch Adam We use Pytorch Adam to learn a neural network $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta})$ to represent the system dynamics, where the input of the network is state and control vectors, and output is the state of next step. We train the neural network by minimizing the following residual

$$\min_{\boldsymbol{\theta}} \sum_{t=0}^{T-1} \|\boldsymbol{x}_{t+1}^o - \boldsymbol{f}(\boldsymbol{x}_t^o, \boldsymbol{u}_t, \boldsymbol{\theta})\|^2. \tag{58}$$

DMDc. The DMDc method Proctor et al. (2016) is a method that based on Koopman theory to reprsent a nonlinear dynamics with linear dynamics of observables. Observables are basis functions of states. The observable space has much higher dimension compared to state space. The estimation of the dynamics is achieved by the following optimization:

$$\min_{\boldsymbol{A},\boldsymbol{B}} \sum_{t=0}^{T-1} \|\boldsymbol{x}_{t+1}^o - \boldsymbol{A}\boldsymbol{x}_t^o - \boldsymbol{B}\boldsymbol{u}_t\|^2. \tag{59}$$

Learning a neural dynamics. We consider the dynamics of each system (cartpole, quadrotor and rocket) are represented by a fully-connected feed-forward neural network $\hat{\boldsymbol{f}}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta})$. The neural network has a layer structure of $(n+m)$-$2(n+m)$-n with *tanh* activation functions, i.e., there is an input layer with $(n+m)$ neurons equal to the dimension of state, one hidden layer with $2(n+m)$ neurons and one output layer with n neurons. The $\boldsymbol{\xi}^o = \{\boldsymbol{x}_{0:T}^o, \boldsymbol{u}_{0:T-1}\}$ obtained previously are used in stage loss. We conducted five trials for each method with different initial $\boldsymbol{\theta}$.

D.4 Policy Tuning On-the-fly

Neural State Feedback Policy. In this application, we learn a the parameter of a neural state feedback policy by minimizing given control objective functions. Specifically, we use a fully-connected feed-forward neural network which has a layer structure of $3n$-$3n$-m with *tanh* activation functions, i.e., there is an input layer with $3n$ neurons equal to the dimension of state, one hidden layer with $3n$ neurons and one output layer with m neurons. The policy parameter $\boldsymbol{\theta}$ is the neural network parameter. For comparison, we apply the guided policy search (GPS) method Levine & Abbeel (2014) and iLQR Li & Todorov (2004) to solve the same problem.

PDP. We employed the PDP in Jin et al. (2020) to solve this problem. The learning rate is set to be $\eta = 10^{-4}$ or $= 10^{-6}$. Five trials were ran given random initial $\boldsymbol{\theta}_0$. For the neural objective function case, the learning rate is set to be $\eta = 10^{-5}$.