2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

A Data-Driven Approach for Inverse Optimal Control

Zihao Liang, Wenjian Hao, Shaoshuai Mou

Abstract— **This paper proposes a data-driven, iterative approach for inverse optimal control (IOC), which aims to learn the objective function of a nonlinear optimal control system given its states and inputs. The approach solves the IOC problem in a challenging situation when the system dynamics is unknown. The key idea of the proposed approach comes from the deep Koopman representation of the unknown system, which employs a deep neural network to represent observables for the Koopman operator. By assuming the objective function to be learned is parameterized as a linear combination of features with unknown weights, the proposed approach for IOC is able to achieve a Koopman representation of the unknown dynamics and the unknown weights in objective function together. Simulation is provided to verify the proposed approach.**

I. Introduction

As one of the key techniques developed in control theories, *optimal control* aims to find control inputs for a target system such that inputs and systems' states optimize a known objective function, which usually represents various mission objectives in practical applications. Such objective functions are usually unknown especially for complicated and newly developed missions such as human motion analysis [1], manipulation [2], human-robot interaction [3], and autonomous driving [4], for which limited knowledge is available and objective functions are usually implicit. To address this, researchers have recently devoted a large amount of attentions to ***Inverse Optimal Control*** **(IOC)**, which aims to learn the objective function from observations of an expert system's trajectories (namely, inputs and states).

Most IOC methods typically assume the unknown objective function parameterized as a linear combination of selected prescribed features (or basis functions), where each feature characterizes one aspect of the system behavior, such as energy cost, time consumption, risk levels, etc. The problem of solving IOC is changed to estimate the unknown weights in constructing the objective function based on such selected features. A direction to solving IOC problems is by adopting a double-layer architecture [3]–[9], in which the weights are updated in an outer layer while optimal control systems are solved in the inner layer with the cost of high computation for repeatedly solving optimal control problems. To further reduce the computational burden in solving IOC, researchers started to leverage the optimality conditions such as Karush-Kuhn-Tucker (KKT) conditions, for which the observed trajectory must satisfy, and the unknown weights can thus be solved directly solved by constructing the optimal equations [10], [11]. Along this direction, the authors of [12], [13] have solved the problem of IOC in the case when observed trajectories are not complete, with a further generalization of the proposed approach to a distributed algorithm for IOC in multi-agent systems in [14].

Note that all the IOC methods mentioned above heavily depend on the exact knowledge of the underlying dynamics of an optimal control system, i.e. these methods are not applicable if the dynamics are unknown. Obtaining a dynamics model is sometimes effort-demanding especially for high-dimensional systems as it requires a large amount of expertise and knowledge in systems and their motion [15], [16]. This requirement in turn weakens one of the most prominent benefits of IOC techniques, which claims to empower non-expert users to program the robot without much effort and only by providing demonstrations. Recognition of this has motivated the goal of this paper, which aims to solve the IOC problem even when the exact knowledge of system dynamics is not available. This requires us to develop a method that not only learns the control objective function but also the dynamics model from demonstration data as well. We note that the Koopman operator has recently been attractive in representing an unknown nonlinear system by a linear time-varying system [17]–[20], based on which controllers could be designed [21]. System identification based on the Koopman operator relies on carefully selecting the observables, for which the introduction of deep neural networks (DNN) has recently proved to be helpful [22], [23].

Motivated by the aforementioned limitation of existing IOC methods and the recent progress in applying Koopman-operator theory in solving data-driven control problems, this paper develops a data-driven IOC approach, where jointly learn the unknown objective function and the underlying dynamics together. We first represent the unknown dynamics of an optimal control system using the Koopman operator, and then iteratively learn the Koopman operator and the control objective function in the same learning framework. Figure 1 illustrates the arrangement of the framework. Compared to existing IOC techniques, the proposed method does not require information on system dynamics. Furthermore, the method does not necessarily require complete demonstration data of an optimal control system, and the input data is allowed to be segments of optimal trajectories.

Notations. Let $\|\cdot\|$ denote the Euclidean norm. For a matrix $A \in \mathbb{R}^{n\times m}$, A' denotes its transpose; $A^\dagger$ denotes its Moore-Penrose pseudoinverse. Let $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_t}$ denotes the Jacobian matrix of a differentiable vector-valued function $\boldsymbol{g}(\boldsymbol{x})$ with respect

The research is supported in part by a grant from the NASA University Leadership Initiative (80NSSC20M0161) and a gift funding from Northrop Grumman Corporation.

The authors are with the School of Aeronautics and Astronautics, Purdue University, IN 47907, USA {liang331, hao93, mous}@purdue.edu

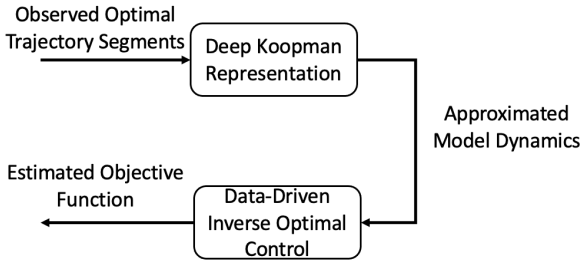

Fig. 1: Data-Driven Inverse Optimal Control

to $\boldsymbol{x}$ evaluated at $\boldsymbol{x}_t$.

II. Problem Formulation

Consider a discrete-time optimal control system with the dynamics

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^n$ is the system state; $\boldsymbol{u}_t \in \mathbb{R}^m$ is the control input; $t = 0, 1, \cdots, T$ is the time step; and $\boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is *unknown* and assumed to be differentiable. The control objective function of the optimal control system is considered to be a linear combination of known features and *unknown* weights:

$$J(\boldsymbol{x}_{0:T}, \boldsymbol{u}_{0:T}, \boldsymbol{\omega}) = \sum_{t=0}^{T} \boldsymbol{\omega}' \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{2}$$

where $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{u}) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^r$ is a specified feature vector function and assumed to be differentiable; $\boldsymbol{\omega} \in \mathbb{R}^r$ is a vector of weights, which are *unknown*; and T is the time horizon.

Since the system considered here is an optimal control system, any trajectory of time horizon T, denoted as a sequence of states-inputs $\boldsymbol{\xi}_{0:T} = \{\boldsymbol{x}_{0:T}, \boldsymbol{u}_{0:T}\}$, minimizes the cost function (2) and satisfies the dynamics (1). Suppose a set of observed sequences of states-inputs is given, which is denoted by

$$\mathcal{D} = \{\boldsymbol{\zeta}_1, \boldsymbol{\zeta}_2, \cdots, \boldsymbol{\zeta}_D\} \tag{3}$$

with each sequence being a segment of a system trajectory that minimizes the cost function (2):

$$\boldsymbol{\zeta}_i = \{\boldsymbol{x}^*_{\underline{t}_i:\bar{t}_i}, \boldsymbol{u}^*_{\underline{t}_i:\bar{t}_i}\} \subseteq \boldsymbol{\xi}_{0:T}, \tag{4}$$

with $\underline{t}_i, \bar{t}_i$ being the starting time and end time of ith state-input sequence.

The **goal** of this paper is to develop an algorithm to estimate the unknown weight vector $\boldsymbol{\omega}$ in (2) via IOC with the given dataset $\mathcal{D}$ without knowing the system dynamics.

III. Main Results

This section develops the data-driven inverse optimal control (IOC) algorithm. Here, we employ the deep Koopman representation (DKR) to approximate the unknown dynamics (1). The first part below presents the deep Koopman representation of unknown dynamics, the second part presents the IOC method based on Koopman operator dynamics, and the third part develops the data-driven IOC algorithm, where the objective function and the Koopman operator dynamics are jointly learned.

A. Dynamics Approximation using Deep Koopman Representation

In this section, we focus on the data-driven approximation of unknown system dynamics. We employ DKR as in [22]. DKR uses the nonlinear mapping $\boldsymbol{\psi}(\cdot, \boldsymbol{\theta}) : \mathbb{R}^n \to \mathbb{R}^N$, parameterized by $\boldsymbol{\theta} \in \mathbb{R}^q$, as the finite-dimension Koopman observable. $\boldsymbol{\psi}(\cdot, \boldsymbol{\theta})$ is represented by a Deep Neural Network (DNN) with a known structure but an unknown parameter $\boldsymbol{\theta}$ to be determined by the set of observed trajectories $\mathcal{D}$. We also denote the number of hidden layers nodes as n_h. One can approximate the unknown dynamics (1) by finding $\boldsymbol{\psi}(\cdot, \boldsymbol{\theta})$ and matrices $\mathcal{K}_x \in \mathbb{R}^{N \times N}$, $\mathcal{K}_u \in \mathbb{R}^{N \times m}$, $\mathcal{C} \in \mathbb{R}^{n \times N}$ based on dataset $\mathcal{D}$ such that for $t \leq T$,

$$\begin{aligned} \boldsymbol{\psi}(\boldsymbol{x}_{t+1}, \boldsymbol{\theta}) &= \mathcal{K}_x \boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta}) + \mathcal{K}_u \boldsymbol{u}_t, \\ \hat{\boldsymbol{x}}_{t+1} &= \mathcal{C} \boldsymbol{\psi}(\boldsymbol{x}_{t+1}, \boldsymbol{\theta}), \end{aligned} \tag{5}$$

where $\hat{\boldsymbol{x}}_t \in \mathbb{R}^n$ is the estimated states vector obtained by DKR. By rewriting (5), one can achieve:

$$\begin{aligned} \hat{\boldsymbol{x}}_{t+1} &= \hat{\boldsymbol{f}}(\boldsymbol{x}_t, \boldsymbol{u}_t) \\ &= \mathcal{C} \boldsymbol{\psi}(\boldsymbol{x}_{t+1}, \boldsymbol{\theta}) \\ &= \mathcal{C} \mathcal{K}_x \boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta}) + \mathcal{C} \mathcal{K}_u \boldsymbol{u}_t, \end{aligned} \tag{6}$$

where $\hat{\boldsymbol{f}}$ denotes the approximated system of (1).

It is noted from (6) that the Koopman operator for approximation of the dynamical system is to transfer the system dynamics (1) into a linear system which has the observables $\boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta})$ as its state. This linear system facilitates the analysis of the original non-linear control system, especially in the field of system learning [24], nonlinear control [21], etc.

We define a vector $\boldsymbol{z}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta}) \in \mathbb{R}^{N+m}$ consists of the observables $\boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta}) \in \mathbb{R}^N$ over states (normally $N \gg n$), and the inputs $\boldsymbol{u}_t$:

$$\boldsymbol{z}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta}) \\ \boldsymbol{u}_t \end{bmatrix}. \tag{7}$$

Then, the finite-dimensional Koopman operator $\mathcal{K} : \mathbb{R}^{N+m} \to \mathbb{R}^{N+m}$ that acts on the space spanned by all observables in $\boldsymbol{z}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta})$ can be written as:

$$\mathcal{K} =: \begin{bmatrix} \mathcal{K}_x & \mathcal{K}_u \end{bmatrix} \in \mathbb{R}^{N \times (N+m)}. \tag{8}$$

Thus, by combining (6)-(8), with a given pair of states-inputs $\{\boldsymbol{x}_t, \boldsymbol{u}_t\}$, the DKR approximation of dynamical system (1) is

$$\begin{aligned} \hat{\boldsymbol{x}}_{t+1} &= \mathcal{C} \mathcal{K}_x \boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta}) + \mathcal{C} \mathcal{K}_u \boldsymbol{u}_t \\ &= C \mathcal{K} \boldsymbol{z}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta}). \end{aligned} \tag{9}$$

For any sequence $\boldsymbol{\zeta}_i \in \mathcal{D}$, we can define the following dynamics approximation loss:

$$l^i_{\mathcal{K}}(\mathcal{K}, \boldsymbol{\zeta}_i, \boldsymbol{\theta}) = \frac{1}{\tau} \sum_{t=\underline{t}_i}^{\bar{t}_i - 1} \| \boldsymbol{\psi}(\boldsymbol{x}_{t+1}, \boldsymbol{\theta}) - \mathcal{K} \boldsymbol{z}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\theta}) \|^2, \tag{10}$$

where $\tau = \bar{t}_i - \underline{t}_i$. The $\mathcal{C}$ matrix is computed by minimizing the following loss function:

$$l^i_{\mathcal{C}} = \frac{1}{\tau} \sum_{t=\underline{t}_i}^{\bar{t}_i - 1} \| \boldsymbol{x}_t - \mathcal{C} \boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta}) \|^2. \tag{11}$$

To solve (10) analytically, we first define:

$$\boldsymbol{\Psi}_i^x = [\boldsymbol{\psi}(\boldsymbol{x}_{\underline{t}_i}, \boldsymbol{\theta}), ..., \boldsymbol{\psi}(\boldsymbol{x}_{\bar{t}_i-1}, \boldsymbol{\theta})], \tag{12}$$

$$\boldsymbol{\Psi}_i^{x+1} = [\boldsymbol{\psi}(\boldsymbol{x}_{\underline{t}_i+1}, \boldsymbol{\theta}), ..., \boldsymbol{\psi}(\boldsymbol{x}_{\bar{t}_i}, \boldsymbol{\theta})], \tag{13}$$

$$\boldsymbol{U}_i = [\boldsymbol{u}_{\underline{t}_i}, ..., \boldsymbol{u}_{\bar{t}_i-1}], \tag{14}$$

$$\boldsymbol{Z}_i = \begin{bmatrix} \boldsymbol{\Psi}_i^x \\ \boldsymbol{U}_i \end{bmatrix}. \tag{15}$$

The Koopman operator is computed analytically by solving:

$$\mathcal{K} = \boldsymbol{\Psi}_i^{x+1} \boldsymbol{Z}_i' (\boldsymbol{Z}_i \boldsymbol{Z}_i')^{-1}. \tag{16}$$

Any solution to (16) is a solution to (10) [21]. We are also able to solve the equation (11) analytically for the matrix $\mathcal{C}$ by:

$$\mathcal{C} = X_i (\boldsymbol{\Psi}_i^x)^\dagger \tag{17}$$

where $X_i = [\boldsymbol{x}_{\underline{t}_i}, \cdots, \boldsymbol{x}_{\bar{t}_i-1}]$.

Equation (16) computes the Koopman operator by only utilizing a segment of trajectory $\boldsymbol{\zeta}_i$ in the provided set of observed data $\mathcal{D}$. To incorporate other segments of the trajectory, one needs to compute the inverse in (16) and the pseudo-inverse in (17) repeatedly, which is computationally expensive as i increases. To fully utilize the whole data set in a computationally efficient way, we borrow the iterative update law of the Koopman operator proposed by [23]. To utilize this update law, the following assumptions need to be made:

Assumption 1. *The matrices $\boldsymbol{\Psi}_i^x$ in (12) and $\boldsymbol{Z}_i$ in (15) are of full row rank.*

Remark 1. *Assumption 1 ensures that the matrices $\boldsymbol{\Psi}_i^x$ and $\boldsymbol{Z}_i$ are invertible by using Moore-Penrose pseudo inverse.*

Assumption 2. *For any $\boldsymbol{\zeta}_i$ in (3), let Δt denotes the observation interval between each states-inputs pair $\{\boldsymbol{x}_t, \boldsymbol{u}_t\}$. The observation interval Δt is sufficiently small such that for some constant $\mu_x \geq 0, \mu_u \geq 0$, $\| \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \| < \mu_x < \infty$ and $\| \boldsymbol{u}_{t+1} - \boldsymbol{u}_t \| < \mu_u < \infty$.*

Remark 2. *If the observation interval Δt goes to zero, the constant μ_x and μ_u also go to zero.*

Assumption 3. *The deep neural network observable function $\boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{\theta})$ is Lipschitz continuous on the system state space with Lipschitz constant μ_g.*

With assumptions 1-3 are made, the lemma of the update law is introduced:

Lemma 1. *[23] If assumption 1-3 hold, given $\boldsymbol{\psi}(\cdot, \boldsymbol{\theta})$, $\mathcal{K}$ and $\mathcal{C}$, with a new batch of data denoted as $\boldsymbol{\zeta}_{i+1}$, the Koopman operator $\mathcal{K}$ and matrix $\mathcal{C}$ are updated as follows:*

$$\mathcal{K} = (\boldsymbol{\Psi}_{i+1}^{x+1} - \mathcal{K} \boldsymbol{Z}_{i+1}) \gamma_{i+1} \boldsymbol{Z}_{i+1}' (\boldsymbol{Z}_i \boldsymbol{Z}_i')^{-1} + \mathcal{K}, \tag{18}$$

$$\mathcal{C} = (\boldsymbol{Z}_{i+1} - \mathcal{C} \boldsymbol{\Psi}_{i+1}^x) \bar{\gamma}_{i+1} (\boldsymbol{\Psi}_{i+1}^x)' (\boldsymbol{\Psi}_i^x (\boldsymbol{\Psi}_i^x)')^{-1} + \mathcal{C}, \tag{19}$$

where

$$\gamma_{i+1} = (I_\tau + \boldsymbol{Z}_{i+1}' (\boldsymbol{Z}_i \boldsymbol{Z}_i')^{-1} \boldsymbol{Z}_{i+1})^{-1} \in \mathbb{R}^{\tau \times \tau}, \tag{20}$$

$$\bar{\gamma}_{i+1} = (I_\tau + (\boldsymbol{\Psi}_{i+1}^x)' (\boldsymbol{\Psi}_i^x (\boldsymbol{\Psi}_i^x)')^{-1} \boldsymbol{\Psi}_{i+1}^x)^{-1} \in \mathbb{R}^{\tau \times \tau}. \tag{21}$$

Once the matrices $\mathcal{K}$ and $\mathcal{C}$ are updated, the parameter $\boldsymbol{\theta}$ is solved by the following optimization problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{j=1}^{i+1} l_{\mathcal{K}}^j + l_{\mathcal{C}}^j. \tag{22}$$

B. Inverse Optimal Control with Deep Koopman Representation

We consider a system trajectory of time horizon T, $\boldsymbol{\xi}_{0:T} = \{\boldsymbol{x}_{0:T}, \boldsymbol{u}_{0:T}\}$, which minimizes the cost function given in (2). Based on Pontryagin's maximum principle [25], [26], there exists a sequence of costates $\boldsymbol{\lambda}_t \in \mathbb{R}^n$ with $t = 0, \cdots, T$, such that the following optimality conditions are satisfied:

$$\begin{aligned} \boldsymbol{\lambda}_t &= \frac{\partial \boldsymbol{\phi}'}{\partial \boldsymbol{x}_t} \boldsymbol{\omega} + \frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{x}_t} \boldsymbol{\lambda}_{t+1}, \\ \boldsymbol{0} &= \frac{\partial \boldsymbol{\phi}'}{\partial \boldsymbol{u}_t} \boldsymbol{\omega} + \frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{u}_t} \boldsymbol{\lambda}_{t+1}, \end{aligned} \tag{23}$$

for $t = 0, 1, \cdots, T-1$, and $\boldsymbol{\lambda}_T = \frac{\partial \boldsymbol{\phi}'}{\partial \boldsymbol{x}_T} \boldsymbol{\omega}$.

Now, we replace $\boldsymbol{f}$ with $\hat{\boldsymbol{f}}$ that is obtained using DKR. According to (9), we have:

$$\begin{aligned} \frac{\partial \hat{\boldsymbol{f}}}{\partial \boldsymbol{x}_t} &= \mathcal{C} \mathcal{K}_x \frac{\partial \boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta})}{\partial \boldsymbol{x}_t}, \\ \frac{\partial \hat{\boldsymbol{f}}}{\partial \boldsymbol{u}_t} &= \mathcal{C} \mathcal{K}_u. \end{aligned} \tag{24}$$

Now, we substitute (24) into (23) leads to

$$\begin{aligned} \boldsymbol{\lambda}_t &= \frac{\partial \boldsymbol{\phi}'}{\partial \boldsymbol{x}_t} \boldsymbol{\omega} + \frac{\partial \boldsymbol{\psi}'}{\partial \boldsymbol{x}_t} \mathcal{K}_x' C' \boldsymbol{\lambda}_{t+1}, \\ \boldsymbol{0} &= \frac{\partial \boldsymbol{\phi}'}{\partial \boldsymbol{u}_t} \boldsymbol{\omega} + \mathcal{K}_u' C' \boldsymbol{\lambda}_{t+1}. \end{aligned} \tag{25}$$

Note that in (25), $\boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta})$ is represented by $\boldsymbol{\psi}$ for simplicity.

Now, we consider a segment of the system trajectory data, say $\boldsymbol{\zeta} = \{\boldsymbol{x}_{\underline{t}:\bar{t}}, \boldsymbol{u}_{\underline{t}:\bar{t}}\} \subseteq \boldsymbol{\xi}_{0:T}$. By writing (25) in matrix form corresponding to the available data $\boldsymbol{\zeta}$, we have the following compact equation

$$\begin{aligned} \boldsymbol{A} \boldsymbol{\lambda}_{\underline{t}+1:\bar{t}} - \boldsymbol{\Phi}_x \boldsymbol{\omega} &= \boldsymbol{V} \boldsymbol{\lambda}_{\bar{t}+1}, \\ \boldsymbol{B} \boldsymbol{\lambda}_{\underline{t}+1:\bar{t}} + \boldsymbol{\Phi}_u \boldsymbol{\omega} &= \boldsymbol{0}, \end{aligned} \tag{26}$$

where

$$\boldsymbol{A} = \begin{bmatrix} I & \frac{-\partial \boldsymbol{\psi}'}{\partial \boldsymbol{x}_{\underline{t}+1}} \mathcal{K}_x' C' & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & I & \frac{-\partial \boldsymbol{\psi}'}{\partial \boldsymbol{x}_{\bar{t}-1}} \mathcal{K}_x' C' \\ 0 & 0 & \cdots & 0 & I \end{bmatrix}, \tag{27}$$

$$\boldsymbol{B} = \begin{bmatrix} \mathcal{K}_u' C' & 0 & \cdots & 0 \\ 0 & \mathcal{K}_u' C' & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{K}_u' C' \end{bmatrix}, \tag{28}$$

$$\boldsymbol{\Phi}_x = \begin{bmatrix} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}_{\underline{t}+1}} & \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}_{\underline{t}+2}} & \cdots & \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}_{\bar{t}-1}} & \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}_{\bar{t}}} \end{bmatrix}', \tag{29}$$

$$\boldsymbol{\Phi}_u = \begin{bmatrix} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{u}_{\underline{t}}} & \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{u}_{\underline{t}+1}} & \cdots & \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{u}_{\bar{t}-2}} & \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{u}_{\bar{t}-1}} \end{bmatrix}', \tag{30}$$

$$\boldsymbol{V} = \begin{bmatrix} 0 & 0 & \cdots & 0 & C\mathcal{K}_x \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{x}_{\bar{t}}} \end{bmatrix}'. \tag{31}$$

The equations in (26) establishes the relationship between data and the unknown weight vector. It can be written as follow:

$$\underbrace{\begin{bmatrix} \boldsymbol{A} & -\boldsymbol{\Phi}_x & \boldsymbol{V} \\ \boldsymbol{B} & \boldsymbol{\Phi}_u & \boldsymbol{0} \end{bmatrix}}_{\hat{\boldsymbol{F}}(\mathcal{K},\mathcal{C},\boldsymbol{\zeta})} \underbrace{\begin{bmatrix} \boldsymbol{\lambda}_{\underline{t}+1:\bar{t}} \\ \boldsymbol{\omega} \\ \boldsymbol{\lambda}_{\bar{t}+1} \end{bmatrix}}_{\boldsymbol{\nu}(\boldsymbol{\lambda},\boldsymbol{\omega})} = \boldsymbol{0} \tag{32}$$

with $\hat{\boldsymbol{F}} \in \mathbb{R}^{(n+m)(\bar{t}-\underline{t}+1)\times(n(\bar{t}-\underline{t}+2)+r)}$ depends on the Koopman operator $\mathcal{K}$, matrix $\mathcal{C}$ and the data segment $\boldsymbol{\zeta}$; $\boldsymbol{\nu} \in \mathbb{R}^{n(\bar{t}-\underline{t}+2)+r}$ depends on the costates $\boldsymbol{\lambda}$ (includes $\boldsymbol{\lambda}_{\underline{t}+1:\bar{t}}$ and $\boldsymbol{\lambda}_{\bar{t}+1}$) and unknown weight vector $\boldsymbol{\omega}$. Note that in (32), the notations $\hat{\boldsymbol{F}}$ and $\boldsymbol{F}$ mean the matrices are generated with the approximated system $\hat{\boldsymbol{f}}$ and true system $\boldsymbol{f}$ in (1) respectively.

For a segment of the system trajectory data $\boldsymbol{\zeta}$, if the Koopman operator $\mathcal{K}$ and matrix $\mathcal{C}$ are given, one can choose to obtain a least square estimate for the weights $\boldsymbol{\omega}$ by solving the following equivalent optimization,

$$\hat{\boldsymbol{\nu}}(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\omega}}) = \arg\min_{\boldsymbol{\nu}} \boldsymbol{\nu}' \hat{\boldsymbol{F}}' \hat{\boldsymbol{F}} \boldsymbol{\nu}. \tag{33}$$

Here, $\hat{\boldsymbol{\nu}}$, $\hat{\boldsymbol{\lambda}}$ and $\hat{\boldsymbol{\omega}}$ are called a least-square estimate to the vector $\boldsymbol{\nu}$, costates $\boldsymbol{\lambda}$ and unknown weights $\boldsymbol{\omega}$ respectively. Also note that to prevent obtaining the trivial solution, a normalization constraint is typically added to the weight variables, e.g., $\sum_{i=1}^{r} \omega_i = 1$.

C. Data-Driven IOC Algorithm

We now develop the data-driven IOC framework to estimate the unknown objective weight with unknown dynamics using the given dataset $\mathcal{D}$ in (3). To solve for the data-driven IOC problem, we propose a method to update the parameter $\boldsymbol{\theta}$ and the unknown weight $\boldsymbol{\omega}$ iteratively. The pseudo code of the proposed method is demonstrated in Algorithm 1.

Algorithm 1: Data-driven IOC Pseudo Code

Input : $\mathcal{D}, \boldsymbol{\phi}$
Output : $\hat{\boldsymbol{\omega}}$
Initialize: $\boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta})$ // Build Koopman observables using DNN with initial guess of $\boldsymbol{\theta}$.

```
Obtain K and C by solving (16) and (17) with ζ_1.
for i = 2 : D do
    Update K and C with ζ_i using (18) and (19).
    Solve (22) to obtain θ.
    Generate matrix F̂ with obtained K, C, θ and all
      of the incorporated trajectory segments ζ_j,
      where j ≤ i.
    Solve (33) to obtain the least-square estimate of
      weight vector ω̂.
```

We now present our main theorem:

Theorem 1. *Given a set of observed sequences of states-inputs pair (3). With the unknown model dynamics (1) approximated by DKR, which has observables $\boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta})$ represented by DNN and parameterized by $\boldsymbol{\theta}$. By using Algorithm 1, the least-square estimate $\hat{\boldsymbol{\omega}}$ of the unknown objective weight in (2) converges to the true weight $\boldsymbol{\omega}$ if assumptions 1-3 hold and the following conditions are fulfilled:*

1) *There are infinite number of hidden layer nodes in the DNN, i.e. $n_h = \infty$.*
2) *The observation interval Δt is sufficiently small such that μ_x and μ_u equal to zero.*
3) *$\max_{\boldsymbol{x}_t \in \mathcal{D}} \parallel \boldsymbol{x}_t - \mathcal{C}\boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta}) \parallel$ is zero.*

The proof of convergence of Algorithm 1 will be shown in the next section.

D. Convergence Analysis

This section provides the convergence analysis for the proposed data-driven IOC algorithm shown in Algorithm 1. First, we denote the estimation error of the approximated system $\hat{\boldsymbol{f}}$ as $\boldsymbol{e}_t \in \mathbb{R}^n$, where

$$\boldsymbol{e}_t = \hat{\boldsymbol{f}}(\boldsymbol{x}_t, \boldsymbol{u}_t) - \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t). \tag{34}$$

By using the update rule stated in Lemma 2, the norm of the estimation error $\boldsymbol{e}_t$ is bounded and can become zero according to the following lemma:

Lemma 2. *[23] If assumptions 1-3 hold, then the supremum of the norm of the estimation error $\boldsymbol{e}_t$ is:*

$$\lim_{n_h \to \infty} \sup \parallel \boldsymbol{e}_t \parallel = (\parallel \mathcal{C}\mathcal{K}_x \parallel \mu_g + 1)\mu_x + \parallel \mathcal{C}\mathcal{K}_u \parallel \mu_u + l_{\mathcal{C}}^{max}.$$

where $l_{\mathcal{C}}^{max} =: \max_{\boldsymbol{x}_t \in \mathcal{D}} \parallel \boldsymbol{x}_t - \mathcal{C}\boldsymbol{\psi}(\boldsymbol{x}_t, \boldsymbol{\theta}) \parallel$. Then, the norm of the estimation error $\boldsymbol{e}_t$ is zero, i.e. $\parallel \boldsymbol{e}_t \parallel = 0$, if the following conditions are satisfied:

1) *There are infinite number of hidden layer nodes n_h.*
2) *The observation interval Δt is sufficiently small such that μ_x and μ_u equal to zero.*
3) *$l_{\mathcal{C}}^{max}$ is zero.*

Lemma 3. *If $\parallel \boldsymbol{e}_t \parallel = 0$, the partial derivative of the estimated dynamics $\hat{\boldsymbol{f}}$ with respect to states and inputs, denoted as $\frac{\partial \hat{\boldsymbol{f}}}{\partial \boldsymbol{x}_t}$ and $\frac{\partial \hat{\boldsymbol{f}}}{\partial \boldsymbol{u}_t}$, are equal to the true partial derivatives $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t}$ and $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_t}$.*

Proof. The estimated and true partial derivatives, $\frac{\partial \hat{\boldsymbol{f}}}{\partial \boldsymbol{x}_t}$ and $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t}$, can be expanded as:

$$\frac{\partial \hat{\boldsymbol{f}}}{\partial \boldsymbol{x}_t} = \lim_{\Delta \boldsymbol{x} \to 0} \frac{\hat{\boldsymbol{f}}(\boldsymbol{x}_{t+1}, \boldsymbol{u}_t) - \hat{\boldsymbol{f}}(\boldsymbol{x}_t, \boldsymbol{u}_t)}{\Delta \boldsymbol{x}},$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t} = \lim_{\Delta \boldsymbol{x} \to 0} \frac{\boldsymbol{f}(\boldsymbol{x}_{t+1}, \boldsymbol{u}_t) - \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t)}{\Delta \boldsymbol{x}}.$$

where $\Delta \boldsymbol{x} = \boldsymbol{x}_{t+1} - \boldsymbol{x}_t$. Denote the difference between estimated and true partial derivatives as:

$$\Delta \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t} = \frac{\partial \hat{\boldsymbol{f}}}{\partial \boldsymbol{x}_t} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t}$$
$$= \lim_{\Delta \boldsymbol{x} \to 0} \frac{(\hat{\boldsymbol{f}}(\boldsymbol{x}_{t+1}, \boldsymbol{u}_t) - \boldsymbol{f}(\boldsymbol{x}_{t+1}, \boldsymbol{u}_t)) - (\hat{\boldsymbol{f}}(\boldsymbol{x}_t, \boldsymbol{u}_t) - \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t))}{\Delta \boldsymbol{x}}.$$

If $\| \boldsymbol{e}_t \| = 0$,

$$\begin{aligned} \hat{\boldsymbol{f}}(\boldsymbol{x}_{t+1}, \boldsymbol{u}_t) - \boldsymbol{f}(\boldsymbol{x}_{t+1}, \boldsymbol{u}_t) = 0, \\ \hat{\boldsymbol{f}}(\boldsymbol{x}_t, \boldsymbol{u}_t) - \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t) = 0. \end{aligned} \tag{35}$$

Therefore,

$$\lim_{\|\boldsymbol{e}_t\| \to 0} \Delta \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t} = 0. \tag{36}$$

Similar proof also leads to:

$$\lim_{\|\boldsymbol{e}_t\| \to 0} \Delta \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_t} = 0, \tag{37}$$

where $\Delta \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_t} = \frac{\partial \hat{\boldsymbol{f}}}{\partial \boldsymbol{u}_t} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_t}$. ■

Proof of Theorem 1. Suppose we know the true dynamics $\boldsymbol{f}$, one can generate a matrix $\boldsymbol{F}$ with data $\boldsymbol{\zeta}$ as in (32), where $\boldsymbol{F} = \hat{\boldsymbol{F}} - \Delta \boldsymbol{F}$, with

$$\Delta \boldsymbol{F} = \begin{bmatrix} \Delta \boldsymbol{A} & \boldsymbol{0} & \Delta \boldsymbol{V} \\ \Delta \boldsymbol{B} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}. \tag{38}$$

where

$$\Delta \boldsymbol{A} = \begin{bmatrix} 0 & \Delta \frac{-\partial \boldsymbol{f}'}{\partial \boldsymbol{x}_{t+1}} & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \Delta \frac{-\partial \boldsymbol{f}'}{\partial \boldsymbol{x}_{\bar{t}-1}} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}, \tag{39}$$

$$\Delta \boldsymbol{B} = \begin{bmatrix} \Delta \frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{u}_{\underline{t}}} & 0 & \cdots & 0 \\ 0 & \Delta \frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{u}_{\underline{t}+1}} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \Delta \frac{\partial \boldsymbol{f}'}{\partial \boldsymbol{u}_{\bar{t}-1}} \end{bmatrix}, \tag{40}$$

$$\Delta \boldsymbol{V} = \begin{bmatrix} 0 & 0 & \cdots & 0 & \Delta \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_{\bar{t}}} \end{bmatrix}'. \tag{41}$$

Equation (32) can be written as:

$$0 = \hat{\boldsymbol{F}} \hat{\boldsymbol{\nu}} = (\boldsymbol{F} + \Delta \boldsymbol{F}) \hat{\boldsymbol{\nu}} = \boldsymbol{F} \hat{\boldsymbol{\nu}} + \Delta \boldsymbol{F} \hat{\boldsymbol{\nu}}. \tag{42}$$

According to Lemma 2, the norm of the estimation error $\boldsymbol{e}_t$ is zero if

1) There is an infinite number of hidden layer nodes in DNN.
2) The observation interval Δt is sufficiently small, which makes μ_x and μ_u equal to zero.
3) $l_{\mathcal{C}}^{max}$ equals to zero.

Once the above conditions are fulfilled, we then employ Lemma 3. According to Lemma 3, if the norm of the estimation error is zero, $\Delta \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ and $\Delta \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}$ are zero. Therefore, $\Delta \boldsymbol{A}, \Delta \boldsymbol{B}, \Delta \boldsymbol{V}$ become zero matrices, which means $\Delta \boldsymbol{F}$ becomes a zero matrix. As a result, $\boldsymbol{F} = \hat{\boldsymbol{F}}$, $\hat{\boldsymbol{\nu}}$ converges to $\boldsymbol{\nu}$ which is obtained using the true $\boldsymbol{F}$. Thus, $\hat{\boldsymbol{\omega}}$ converges to $\boldsymbol{\omega}$. ■

IV. Numerical Experiments

The proposed data-driven inverse optimal control algorithm is evaluated by a simulation of a pendulum model.

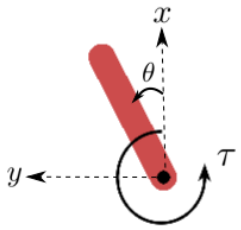

Fig. 2: A simulated pendulum.

As shown in Figure 2, we consider that a pendulum system moves in the vertical plane with continuous dynamics given by [27]

$$ml^2 \ddot{\theta} + mgl \sin\theta = \tau, \tag{43}$$

where θ is the angle of the pendulum, m is the mass of the pendulum, g is the gravitational acceleration, l is the length of the pendulum and τ here is the torque applied. The parameters used are $g = 10m/s^2$, the length $l = 10m$, and the pendulum mass $m = 1kg$. By defining the states and control inputs of the pendulum:

$$\boldsymbol{x} \triangleq \begin{bmatrix} \theta & \dot{\theta} \end{bmatrix}' \quad \text{and} \quad \boldsymbol{u} \triangleq \boldsymbol{\tau}, \tag{44}$$

respectively, one could write (43) in state-space representation $\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})$ and further approximate it by the following discrete-time form

$$\boldsymbol{x}_{t+1} \approx \boldsymbol{x}_t + \Delta \cdot \boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{u}_t) \triangleq \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{45}$$

where $\Delta = 0.001$s is the discretization interval. The motion of the pendulum is controlled to minimize the objective function (2), which here is set as a weighted distance to the goal state $\boldsymbol{x}^{\mathrm{g}} = [\theta^{\mathrm{g}}, \dot{\theta}^{\mathrm{g}}]' = [\pi, 0]'$ plus the control effort $\|\boldsymbol{u}\|^2$. Here, the corresponding features and weights defined are as follows.

$$\boldsymbol{\phi} = \begin{bmatrix} (\theta - \theta^{\mathrm{g}})^2 \\ (\dot{\theta} - \dot{\theta}^{\mathrm{g}})^2 \\ ||\boldsymbol{u}||^2 \end{bmatrix}, \qquad \boldsymbol{\omega} = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, \tag{46}$$

The initial condition of the robot arm is set as $x_0 = [0, 0]'$, and the time horizon is set as $T = 10$. We set the ground-truth weights as in (46).

In the data-driven IOC task, we learn the weight vector $\boldsymbol{\omega}$ and the system $\boldsymbol{f}$ from the segment data of the optimal trajectory. The effectiveness of the algorithm is demonstrated in Table I and II. We tested our algorithm with different results of $l_{\mathcal{C}}^{max}$ and different n_h. In our case, we consider the observation interval Δt to be the same as the sampling instance interval of the unknown discrete-time system (1). Therefore, the constants μ_x and μ_u are kept unchanged.

In Table I and Table II, it is shown that as $l_{\mathcal{C}}^{max}$ decreases, or the number of hidden layer nodes n_h increases, the IOC result gets closer to the ground truth, i.e. the 2-norm error

TABLE I: IOC results with different $l_{\mathcal{C}}^{max}$.

$l_{\mathcal{C}}^{max}$	$\hat{\boldsymbol{\omega}}$	Error, Weight	Error, Traj.
1e-4	$[1.75, 1.37, 0.87]$	0.468	6.13e-2
1e-5	$[1.89, 1.16, 0.94]$	0.207	1.25e-3
1e-6	$[1.92, 1.12, 0.96]$	0.146	5.22e-4

TABLE II: IOC results with different number of hidden layer nodes n_h in DNN .

n_h	$\hat{\boldsymbol{\omega}}$	Error, Weight	Error, Traj.
1254	$[1.71, 1.24, 1.05]$	0.382	6.83e-2
1670	$[1.93, 1.15, 0.92]$	0.183	1.70e-2
2086	$[1.96, 1.05, 0.98]$	0.070	2.24e-4

between $\hat{\boldsymbol{\omega}}$ and the true $\boldsymbol{\omega}$ decreases. Optimal trajectories are also generated using the estimate $\hat{\boldsymbol{\omega}}$. We can see that as the error of weight decreases, the 2-norm of the error between the ground truth trajectory and trajectories produced using $\hat{\boldsymbol{\omega}}$ decreases.

V. Conclusions

This paper has developed a data-driven method to solve the inverse optimal control problem with unknown system dynamics. The unknown system dynamics is approximated by the finite-dimensional Koopman operator, with the observables represented by a deep neural network. We proved that if certain conditions are fulfilled, the approximated system, as well as the approximated system derivatives with respect to states and inputs, will converge to the true system and the true derivatives. As a result, an iterative scheme to update the least-square estimate of the weight vector in the unknown system dynamics is proposed.

For future research, we will extend the proposed method to an IOC method with noisy data. The motivation here is that the assumption of perfect data is sometimes challenging to fulfill since there are always errors when obtaining data using sensors in reality. Another potential research direction would be performing IOC with system output rather than the states.

References

[1] W. Jin, D. Kulić, J. F.-S. Lin, S. Mou, and S. Hirche, "Inverse optimal control for multiphase cost functions," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1387–1398, 2019.

[2] P. Englert, N. A. Vien, and M. Toussaint, "Inverse kkt: Learning cost functions of manipulation tasks from demonstrations," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1474–1488, 2017.

[3] J. Mainprice, R. Hayne, and D. Berenson, "Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces," *IEEE Transactions on Robotics*, vol. 32, no. 4, pp. 897–908, 2016.

[4] *Learning driving styles for autonomous vehicles from demonstration*, 2015.

[5] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning.," in *International Conference of Machine Learning*, vol. 1, p. 2, 2000.

[6] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotion—an inverse optimal control approach," *Autonomous Robots*, vol. 28, no. 3, pp. 369–383, 2010.

[7] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *International Conference on Machine Learning*, p. 1, 2004.

[8] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *International Conference on Machine Learning*, pp. 729–736, 2006.

[9] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *AAAI*, vol. 8, pp. 1433–1438, 2008.

[10] A. Keshavarz, Y. Wang, and S. Boyd, "Imputing a convex objective function," in *IEEE International Symposium on Intelligent Control*, pp. 613–619, IEEE, 2011.

[11] A.-S. Puydupin-Jamin, M. Johnson, and T. Bretl, "A convex approach to inverse optimal control and its application to modeling human locomotion," in *International Conference on Robotics and Automation*, pp. 531–536, 2012.

[12] Z. Liang, W. Jin, and S. Mou, "An iterative method for inverse optimal control," in *2022 13th Asian Control Conference (ASCC)*, pp. 959–964, IEEE, 2022.

[13] W. Jin, D. Kulic, S. Mou, and S. Hirche, "Inverse optimal control from incomplete trajectory observations," *The International Journal of Robotics Research*, pp. 1–18, 2021.

[14] W. Jin and S. Mou, "Distributed inverse optimal control," *Automatica*, vol. 129, p. 109658, 2021.

[15] T. B. Schön, A. Wills, and B. Ninness, "System identification of nonlinear state-space models," *Automatica*, vol. 47, no. 1, pp. 39–49, 2011.

[16] O. Nelles and O. Nelles, *Nonlinear dynamic system identification*. Springer, 2001.

[17] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015.

[18] M. Budišić, R. Mohr, and I. Mezić, "Applied koopmanism," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 22, no. 4, p. 047510, 2012.

[19] M. O. Williams, M. S. Hemati, S. T. Dawson, I. G. Kevrekidis, and C. W. Rowley, "Extending data-driven koopman analysis to actuated systems," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 704–709, 2016.

[20] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Generalizing koopman theory to allow for inputs and control," *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 1, pp. 909–930, 2018.

[21] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.

[22] Y. Han, W. Hao, and U. Vaidya, "Deep learning of koopman representation for control," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 1890–1895, 2020.

[23] W. Hao, B. Huang, W. Pan, D. Wu, and S. Mou, "Deep koopman representation of nonlinear time varying systems," *arXiv preprint arXiv:2210.06272*, 2022.

[24] I. Abraham and T. D. Murphey, "Active learning of dynamics for data-driven control using koopman operators," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1071–1083, 2019.

[25] L. S. Pontryagin, V. G. Boltyanskiy, R. V. Gamkrelidze, and E. F. Mishchenko, *The Mathematical Theory of Optimal Processes*. John Wiley & Sons, Inc., 1962.

[26] E. Todorov *et al.*, "Optimal control theory," *Bayesian brain: probabilistic approaches to neural coding*, pp. 268–298, 2006.

[27] Y. Han, W. Hao, and U. Vaidya, "Deep learning of koopman representation for control," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 1890–1895, IEEE, 2020.