Deep Koopman Learning using the Noisy Data

Anonymous authors
Paper under double-blind review

Abstract

This paper proposes a data-driven framework to learn a finite-dimensional approximation of a Koopman operator for approximating the state evolution of a dynamical system under noisy observations. To this end, our proposed solution has two main advantages. First, the proposed method only requires the measurement noise to be bounded. Second, the proposed method modifies the existing deep Koopman operator formulations by characterizing the effect of the measurement noise on the Koopman operator learning and then mitigating it by updating the tunable parameter of the observable functions of the Koopman operator, making it easy to implement. The performance of the proposed method is demonstrated on several standard benchmarks. We then compare the presented method with similar methods proposed in the latest literature on Koopman learning.

1 Introduction

Directly dealing with complex nonlinear dynamical systems for model-based control design has remained a challenge for the control community. One long-standing solution to this problem has been to use linearized models and the associated vast body of knowledge for linear analysis. Linear control theory is a very rich and well-developed field that provides rigorous control development with methods for stability and robustness guarantees. Lyapunov showed that for a linearized system that is stable around an equilibrium point, there exists a region of stability around this equilibrium point for which the original nonlinear system is also stable A.Lyapunov (1992). Recent advances in data-driven methods have spurred new and increased research interest in machine learning (ML) based methods for deriving reduced-order models (ROM) as surrogates for complex nonlinear systems. This has also led to the adoption of these methods for developing control and autonomy/automation solutions for robotic and unmanned systems. Examples include learning dynamics using deep neural networks (DNNs) Murphy (2002); Gillespie et al. (2018), physics-informed neural networks (PINNs) Raissi et al. (2019), and lifting linearization methods such as Koopman operator methods Mezić (2015); Proctor et al. (2018); Mauroy & Goncalves (2016). Lifting linearization allows one to represent a nonlinear system with an equivalent linear system in a lifted, higher-dimensional, space. It is, however, typically difficult to find an exact finite-dimensional linear representation for most nonlinear systems. Further, the Koopman operator fails on non-autonomous systems:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t). \tag{1}$$

This poses a challenge for the control design of dynamical systems in the choice of a sufficient basis function necessary for the lifted system to be linear and exact. Extensions to such systems require truncation-based approximations, and the finite-dimensional representation is no longer exact. To this end, various eigen-decomposition-based truncations are proposed. In Lusch et al. (2017) the authors proposed using deep learning methods to discover the eigenfunctions of the approximated Koopman operator, and Yeung et al. (2019); Lusch et al. (2018); Han et al. (2020); Bevanda et al. (2021) employed deep neural networks (DNNs) as observable functions of the Koopman operator, which are tuned based on collected state-control pairs by minimizing an appropriately defined loss function which is also referred as the deep Koopman operator method (DKO). Recent work such as Hao et al. (2024) has extended the DKO method to approximate nonlinear time-varying systems. Similar to the Koopman operator Koopman (1931); Koopman & Neumann (1932), two other popular methods, dynamic mode decomposition (DMD) Schmid (2010) and extending

dynamic mode decomposition (EDMD) lift the state space to a higher-dimensional space, for which the temporal evolution is approximately linear Korda & Mezić (2018). These methods rely on a set of measured output variables that collectively define some nonlinear representation of the independent state variables. Establishing a sufficient set of these observable functions remains an active area of research. Further, real-world noisy measurements impose additional challenges. Additionally, for most practical systems, it is also critical to find computationally feasible approximation methods for extracting finite dimensional representations.

Related work. While Koopman-based methods have been proven to be effective in learning dynamics from a system's input-output (state) data pairs. However, in practical real-world applications, the output measurements are noisy and can result in biased estimates of the linear system. Even if the noise of the state variables is assumed to be uncorrelated, the nonlinear transformations in the observables may lead to complex noise-influence correlations between the noise-free states and the transformed observables. Several methods are proposed to solve the measurement noise issue. One solution Sotiropoulos. (2021) is to directly measure the states and the observables, this, however, may not always be possible. Noisy measurements are also shown to further complicate the anti-causal observable problem when dealing with the lifting of controlled systems Selby (2021). In other approaches, authors in Dawson et al. (2016); Hemati et al. (2017) introduce total least square (TLS) methods in DMD, in Haseli & Cortés (2019) the authors propose a combination of the EDMD and TLS methods to account for the measurement noise, and in Sinha et al. (2023) the authors propose to solve the EDMD with measurement noise as a robust Koopman operator problem which is a min-max optimization problem.

This paper extends the DKO method to the scenario where the system state data is corrupted by unknown but bounded measurement noise. As already discussed, this creates the challenge of generating additional noise transformations impacted by the DNN-derived basis functions of the deep Koopman operator. This leads to distortion of the noise, and the properties of the measurement noise and associated correlations may not remain the same after lifting. The contributions of this work are that we first propose a data-driven framework to learn the deep Koopman operator from the system states-inputs data pairs under unknown and bounded measurement noise, and then we provide numerical evidence that our proposed method can approximate the system dynamics with reasonable accuracy adequate for control applications.

This paper is organized as follows. Section 2 states the problem. Section 3 presents the proposed algorithm and its theoretical development. The numerical simulations and comparison of the algorithms are shown in Section 4. Finally, Section 5 concludes the paper.

Notations. We denote $\|\cdot\|$ as the Euclidean norm. For a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\| \boldsymbol{A} \|_F$ denotes its Frobenius norm, $\boldsymbol{A}'$ denotes its transpose, and $\boldsymbol{A}^\dagger$ denotes its Moore-Penrose pseudoinverse.

2 Problem formulation

Consider the following discrete time-invariant system:

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{2}$$

$$\boldsymbol{y}_t = \boldsymbol{x}_t + \boldsymbol{w}_t, \tag{3}$$

where $t = 0, 1, 2, \cdots$ denotes the time index, $\boldsymbol{x}_t \in \mathbb{R}^n$ and $\boldsymbol{u}_t \in \mathbb{R}^m$ denote the system state and control input, respectively, $\boldsymbol{y}_t \in \mathbb{R}^n$ denotes the measured state, $\boldsymbol{w}_t \in \mathbb{R}^n$ corresponds to the unknown measurement noise, which is assumed to be bounded (i.e., $\| \boldsymbol{w}_t \| \leq w_{max}$), and $\boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ denotes the nonlinear dynamics mapping which is unknown.

Suppose an observed system states-inputs trajectory from time 0 to T denoted as:

$$\boldsymbol{\xi} = \{(\boldsymbol{y}_t, \boldsymbol{u}_t), t = 0, 1, 2, \cdots, T\}. \tag{4}$$

One way to approximate the unknown dynamics $\boldsymbol{f}$ in 2 using ξ is by employing the so-called deep Koopman operator learning. Namely, one first introduces an estimate dynamics $\hat{\boldsymbol{x}}_{t+1} = \hat{\boldsymbol{f}}(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t, \boldsymbol{\theta})$, where $\hat{\boldsymbol{x}}_t \in \mathbb{R}^n$ and $\hat{\boldsymbol{u}}_t \in \mathbb{R}^m$ denote the introduced system states and inputs, respectively, and $\hat{\boldsymbol{f}}$ is constructed by following

the Koopman operator theory described by:

$$\boldsymbol{g}(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{\theta}) = \boldsymbol{A}\boldsymbol{g}(\hat{\boldsymbol{x}}_t, \boldsymbol{\theta}) + \boldsymbol{B}\hat{\boldsymbol{u}}_t, \tag{5}$$

$$\hat{\boldsymbol{x}}_{t+1} = \boldsymbol{C}\boldsymbol{g}(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{\theta}), \tag{6}$$

where $\boldsymbol{g}(\cdot, \boldsymbol{\theta}) : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^r$ is usually represented by a deep neural network (DNN) with a known structure and tunable by $\boldsymbol{\theta} \in \mathbb{R}^p$, and $\boldsymbol{A} \in \mathbb{R}^{r\times r}$, $\boldsymbol{B} \in \mathbb{R}^{r\times m}$, $\boldsymbol{C} \in \mathbb{R}^{n\times r}$ are constant matrices. Here, 5 with $r \geq n$ represents the dynamics evolution in the lifted space, $\mathbb{R}^r$ and 6 denotes the mapping between the lifted space, $\mathbb{R}^r$, and the original space, $\mathbb{R}^n$. We assume that the introduced dynamics 5-6 are observable and the $\boldsymbol{g}(\cdot, \boldsymbol{\theta})$ is Lipschitz continuous with Lipschitz constant L_g. By combining 5-6, one can denote the deep Koopman operator dynamics as follows:

$$\hat{\boldsymbol{x}}_{t+1} = \hat{\boldsymbol{f}}(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t, \boldsymbol{\theta}) = \boldsymbol{C}(\boldsymbol{A}\boldsymbol{g}(\hat{\boldsymbol{x}}_t, \boldsymbol{\theta}) + \boldsymbol{B}\hat{\boldsymbol{u}}_t). \tag{7}$$

The **problem of interest** is to find constant matrices $\boldsymbol{A}^*$, $\boldsymbol{B}^*$, $\boldsymbol{C}^*$ and parameter $\boldsymbol{\theta}^*$ using the noisy trajectory $\boldsymbol{\xi}$ in 4 such that for any $0 \leq k \leq T-1$, the following holds approximately:

$$\boldsymbol{g}(\boldsymbol{y}_{k+1}, \boldsymbol{\theta}^*) = \boldsymbol{A}^*\boldsymbol{g}(\boldsymbol{y}_k, \boldsymbol{\theta}^*) + \boldsymbol{B}^*\boldsymbol{u}_k, \tag{8}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{C}^*\boldsymbol{g}(\boldsymbol{y}_{k+1}, \boldsymbol{\theta}^*). \tag{9}$$

For notation brevity, we define $\boldsymbol{A}^*$, $\boldsymbol{B}^*$, $\boldsymbol{C}^*$, $\boldsymbol{\theta}^*$ satisfying 8-9 as the following set, which will be referred to as the deep Koopman representation (DKR) throughout this paper.

$$\mathcal{K} = \{\boldsymbol{A}^*, \boldsymbol{B}^*, \boldsymbol{C}^*, \boldsymbol{\theta}^*\}. \tag{10}$$

3 Main Results

In this section, we propose an algorithm to achieve the DKR in 10.

Challenges. To achieve the above DKR, one intuitive way is to minimize the following estimation errors using $\boldsymbol{\xi}$ in 4:

$$\boldsymbol{A}^*, \boldsymbol{B}^*, \boldsymbol{C}^*, \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{\theta}} \frac{1}{2T}\sum_{k=0}^{T-1} \|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{f}}(\boldsymbol{y}_k, \boldsymbol{u}_k, \boldsymbol{\theta})\|^2. \tag{11}$$

Here, one immediate challenge of solving 11 is that $\boldsymbol{x}_k$ is unknown in our problem setting.

Key ideas. To overcome the above challenge, we propose an alternative minimization problem of 11. To proceed, for any $0 \leq k \leq T-1$, we first introduce the notation

$$\boldsymbol{x}_{k+1} = \tilde{\boldsymbol{f}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \tilde{\boldsymbol{\theta}}^*) + \tilde{\boldsymbol{\epsilon}}_k = \tilde{\boldsymbol{C}}^*(\tilde{\boldsymbol{A}}^*\boldsymbol{g}(\boldsymbol{x}_k, \tilde{\boldsymbol{\theta}}^*) + \tilde{\boldsymbol{B}}^*\boldsymbol{u}_k) + \tilde{\boldsymbol{\epsilon}}_k$$

and

$$\boldsymbol{y}_{k+1} = \bar{\boldsymbol{f}}(\boldsymbol{y}_k, \boldsymbol{u}_k, \bar{\boldsymbol{\theta}}^*) + \bar{\boldsymbol{\epsilon}}_k = \bar{\boldsymbol{C}}^*(\bar{\boldsymbol{A}}^*\boldsymbol{g}(\boldsymbol{x}_k, \bar{\boldsymbol{\theta}}^*) + \bar{\boldsymbol{B}}^*\boldsymbol{u}_k) + \bar{\boldsymbol{\epsilon}}_k,$$

where $\tilde{\boldsymbol{f}}$ and $\bar{\boldsymbol{f}}$ are obtained by solving the following DKO problem using the noise-free and noisy trajectories, respectively, while $\tilde{\boldsymbol{\epsilon}}_k$ and $\bar{\boldsymbol{\epsilon}}_k$ represent the estimation errors arising from solving the following optimization problems:

$$\tilde{\boldsymbol{A}}^*, \tilde{\boldsymbol{B}}^*, \tilde{\boldsymbol{C}}^*, \tilde{\boldsymbol{\theta}}^* = \arg\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{\theta}} \frac{1}{2T}\sum_{k=0}^{T-1} \|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{f}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta})\|^2 \tag{12}$$

and

$$\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*, \bar{\boldsymbol{C}}^*, \bar{\boldsymbol{\theta}}^* = \arg\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{\theta}} \frac{1}{2T}\sum_{k=0}^{T-1} \|\boldsymbol{y}_{k+1} - \hat{\boldsymbol{f}}(\boldsymbol{y}_k, \boldsymbol{u}_k, \boldsymbol{\theta})\|^2. \tag{13}$$

Then by expanding the error function in 11 using $\tilde{\boldsymbol{f}}$ and $\bar{\boldsymbol{f}}$, and applying the triangle inequality, it follows that:

$$\begin{aligned}&\|\boldsymbol{x}_{k+1} - \tilde{\boldsymbol{f}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \tilde{\boldsymbol{\theta}}^*) + \tilde{\boldsymbol{f}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \tilde{\boldsymbol{\theta}}^*) - \bar{\boldsymbol{f}}(\boldsymbol{y}_k, \boldsymbol{u}_k, \bar{\boldsymbol{\theta}}^*) + \bar{\boldsymbol{f}}(\boldsymbol{y}_k, \boldsymbol{u}_k, \bar{\boldsymbol{\theta}}^*) - \boldsymbol{y}_{k+1} + \boldsymbol{y}_{k+1} - \hat{\boldsymbol{f}}(\boldsymbol{y}_k, \boldsymbol{u}_k, \boldsymbol{\theta})\|^2 \\ \leq&\|\tilde{\boldsymbol{f}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \tilde{\boldsymbol{\theta}}^*) - \bar{\boldsymbol{f}}(\boldsymbol{y}_k, \boldsymbol{u}_k, \bar{\boldsymbol{\theta}}^*)\|^2 + \|\boldsymbol{y}_{k+1} - \hat{\boldsymbol{f}}(\boldsymbol{y}_k, \boldsymbol{u}_k, \boldsymbol{\theta})\|^2 + \|\tilde{\boldsymbol{\epsilon}}_k\|^2 + \|\bar{\boldsymbol{\epsilon}}_k\|^2.\end{aligned} \tag{14}$$

Note that $\tilde{\boldsymbol{\epsilon}}_k$ and $\bar{\boldsymbol{\epsilon}}_k$ are typically small positive constants and we refer to the existing work Hao et al. (2024) for more details on the analysis of above estimation errors. Removing the constant terms of the upper bound derived in 14, we define the following minimization problem to achieve the DKR in 10:

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{\theta}}\{\mathbf{L}_f = \frac{1}{2T}\sum_{k=0}^{T-1}(\parallel \boldsymbol{y}_{k+1} - \hat{\boldsymbol{f}}(\boldsymbol{y}_k,\boldsymbol{u}_k,\boldsymbol{\theta})\|^2 + \|\tilde{\boldsymbol{f}}(\boldsymbol{x}_k,\boldsymbol{u}_k,\tilde{\boldsymbol{\theta}}^*) - \bar{\boldsymbol{f}}(\boldsymbol{y}_k,\boldsymbol{u}_k,\bar{\boldsymbol{\theta}}^*)\parallel^2)\}. \tag{15}$$

Here, 15 says that instead of minimizing the error function in 11, this paper aims to develop an algorithm to minimize the upper bound of 11. The minimization problem 15 can be viewed as a combination of two components: suppose a noisy trajectory in 13, the first part of 15 aims to find a dynamical model $\bar{\boldsymbol{f}}(\boldsymbol{y}_k,\boldsymbol{u}_k,\bar{\boldsymbol{\theta}}^*)$ to approximate the mapping between $\boldsymbol{y}_k,\boldsymbol{u}_k$ and $\boldsymbol{y}_{k+1}$ and its second part minimizes the norm difference between the achieved $\bar{\boldsymbol{f}}(\boldsymbol{y}_k,\boldsymbol{u}_k,\bar{\boldsymbol{\theta}}^*)$ from 13 and $\tilde{\boldsymbol{f}}(\boldsymbol{x}_k,\boldsymbol{u}_k,\tilde{\boldsymbol{\theta}}^*)$ from 12. The key idea of solving 15 is to characterize difference between $\tilde{\boldsymbol{f}}(\boldsymbol{x}_k,\boldsymbol{u}_k,\tilde{\boldsymbol{\theta}}^*)$ and $\bar{\boldsymbol{f}}(\boldsymbol{y}_k,\boldsymbol{u}_k,\bar{\boldsymbol{\theta}}^*)$. Since the above difference is induced by measurement noise $\boldsymbol{w}_t$, to characterize this difference under $\boldsymbol{w}_t$, we introduce the following loss function:

$$\begin{aligned} r(\bar{\boldsymbol{\theta}}^*,\boldsymbol{w}) =& \frac{1}{2T}\sum_{k=0}^{T-1}\max_{\boldsymbol{w}_k}\|\tilde{\boldsymbol{f}}(\boldsymbol{x}_k,\boldsymbol{u}_k,\tilde{\boldsymbol{\theta}}^*) - \bar{\boldsymbol{f}}(\boldsymbol{y}_k,\boldsymbol{u}_k,\bar{\boldsymbol{\theta}}^*)\parallel^2 \\ =& \frac{1}{2T}\max_{\boldsymbol{w}_k}(\sum_{k=0}^{T-1}\|\boldsymbol{g}(\boldsymbol{y}_k,\tilde{\boldsymbol{\theta}}^*) - \boldsymbol{g}(\boldsymbol{x}_k,\bar{\boldsymbol{\theta}}^*)\|^2 + \|[\tilde{\boldsymbol{A}}^*,\tilde{\boldsymbol{B}}^*] - [\bar{\boldsymbol{A}}^*,\bar{\boldsymbol{B}}^*]\|_F^2 + \|\tilde{\boldsymbol{C}}^* - \bar{\boldsymbol{C}}^*\|_F^2). \end{aligned} \tag{16}$$

Here, due to system state $\boldsymbol{x}_k$ in 16 is unknown in our problem setting, we propose the following theorem to minimize $r(\bar{\boldsymbol{\theta}}^*,\boldsymbol{w})$:

Theorem 1 *If the unknown measurement noise $\|\boldsymbol{w}_t\|$ is bounded by w_{max}, then the optimal $\bar{\boldsymbol{\theta}}^*$ minimizing the following loss function also minimizes 16.*

$$\mathbf{L}_w(\bar{\boldsymbol{\theta}}^*) = \frac{1}{2T}(\|[\bar{\boldsymbol{A}}^*,\bar{\boldsymbol{B}}^*]\|_F^2 + \|\bar{\boldsymbol{C}}^*\|_F^2 + \sum_{k=0}^{T-1}\|\boldsymbol{g}(\boldsymbol{y}_k,\bar{\boldsymbol{\theta}}^*)\|^2 + \|\boldsymbol{g}(\boldsymbol{y}_k,\bar{\boldsymbol{\theta}}^*)'\boldsymbol{g}(\boldsymbol{y}_{k+1},\bar{\boldsymbol{\theta}}^*)\|^2). \tag{17}$$

Proof of Theorem 1 is given in Appendix.

3.1 Algorithm

We now present an algorithm that solves 15 using the noisy data in 4. To minimize the first part of 15, we reformulate it as the following double-layer minimization problem, adhering to the deep Koopman dynamics outlined in 5-6 to compute $\bar{\boldsymbol{A}}^*$, $\bar{\boldsymbol{B}}^*$, $\bar{\boldsymbol{C}}^*$, $\bar{\boldsymbol{\theta}}^*$ in 13:

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}}\min_{\boldsymbol{\theta}}\{\mathbf{L}_{DKR} = \frac{1}{2T}(\sum_{k=0}^{T-1}\|\boldsymbol{g}(\boldsymbol{y}_{k+1},\boldsymbol{\theta}) - \boldsymbol{A}\boldsymbol{g}(\boldsymbol{y}_k,\boldsymbol{\theta}) - \boldsymbol{B}\boldsymbol{u}_k\|^2 + \|\boldsymbol{y}_{k+1} - \boldsymbol{C}\boldsymbol{g}(\boldsymbol{y}_{k+1},\boldsymbol{\theta})\|^2)\}. \tag{18}$$

To address the first-layer minimization problem in 18 with respect to the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$, it is necessary to rewrite $\mathbf{L}_{DKR}$ in a compact form. For this purpose, we construct the following data matrices based on $\boldsymbol{\xi}$ in 4.

$$\begin{aligned} \mathbf{Y} = [\boldsymbol{y}_0,\boldsymbol{y}_1,\cdots,\boldsymbol{y}_{T-1}] \in \mathbb{R}^{n\times T}, \quad \bar{\mathbf{Y}} = [\boldsymbol{y}_1,\boldsymbol{y}_2,\cdots,\boldsymbol{y}_T] \in \mathbb{R}^{n\times T}, \mathbf{U} = [\boldsymbol{u}_0,\boldsymbol{u}_1,\cdots,\boldsymbol{u}_{T-1}] \in \mathbb{R}^{m\times T}, \\ \mathbf{G} = [\boldsymbol{g}(\boldsymbol{y}_0,\boldsymbol{\theta}),\boldsymbol{g}(\boldsymbol{y}_1,\boldsymbol{\theta}),\cdots,\boldsymbol{g}(\boldsymbol{y}_{T-1},\boldsymbol{\theta})] \in \mathbb{R}^{r\times T}, \bar{\mathbf{G}} = [\boldsymbol{g}(\boldsymbol{y}_1,\boldsymbol{\theta}),\boldsymbol{g}(\boldsymbol{y}_2,\boldsymbol{\theta}),\cdots,\boldsymbol{g}(\boldsymbol{y}_T,\boldsymbol{\theta})] \in \mathbb{R}^{r\times T}. \end{aligned} \tag{19}$$

It leads $\mathbf{L}_{DKR}$ in 18 to

$$\mathbf{L}_{DKR} = \frac{1}{2T}(\parallel \bar{\mathbf{G}} - (\boldsymbol{A}\mathbf{G} + \boldsymbol{B}\mathbf{U})\parallel_F^2 + \parallel \mathbf{Y} - \boldsymbol{C}\mathbf{G}\parallel_F^2). \tag{20}$$

Here, to ensure there exists a solution of 20, the following assumption is introduced.

Assumption 1 *The matrices* $\mathbf{G} \in \mathbb{R}^{r\times T}$ *and* $\begin{bmatrix}\mathbf{G}\\ \mathbf{U}\end{bmatrix} \in \mathbb{R}^{(r+m)\times T}$ *in 20 are with full row ranks.*

Remark 1 *Assumption 1 is to ensure the matrices* $\mathbf{G} \in \mathbb{R}^{r\times T}$ *and* $\begin{bmatrix}\mathbf{G}\\ \mathbf{U}\end{bmatrix} \in \mathbb{R}^{(r+m)\times T}$ *are invertible and it naturally requires* $T \geq r+m$.

The solution of 20 is given by

$$[\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*] = \bar{\mathbf{G}} \begin{bmatrix}\mathbf{G}\\ \mathbf{U}\end{bmatrix}^{\dagger}, \tag{21}$$

$$\bar{\boldsymbol{C}}^* = \mathbf{Y}\mathbf{G}^{\dagger}. \tag{22}$$

By substituting $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{C}$ in 20 with their corresponding expressions in 21 and 22 respectively, the second-layer minimization of $\mathbf{L}_{DKR}$ with respect to the parameter $\boldsymbol{\theta}$ is formulated as follows:

$$\mathbf{L}_{DKR}(\boldsymbol{\theta}) = \frac{1}{2T}(\parallel \bar{\mathbf{G}} - \begin{bmatrix}\bar{\boldsymbol{A}}^* \bar{\boldsymbol{B}}^*\end{bmatrix} \begin{bmatrix}\mathbf{G}\\ \mathbf{U}\end{bmatrix} \parallel_F^2 + \parallel \mathbf{Y} - \bar{\boldsymbol{C}}^*\mathbf{G} \parallel_F^2). \tag{23}$$

Based on $\mathbf{L}_w$ in 17 and $\mathbf{L}_{DKR}$ in 23, one can reformulate the loss function $\mathbf{L}_f(\boldsymbol{\theta}) = \mathbf{L}_{DKR}(\boldsymbol{\theta}) + \mathbf{L}_w(\boldsymbol{\theta})$ in 15 with respect to $\boldsymbol{\theta}$ as follows:

$$\mathbf{L}_f(\boldsymbol{\theta}) = \frac{1}{2T}(w_1 \parallel \bar{\mathbf{G}} - \begin{bmatrix}\bar{\boldsymbol{A}}^* \bar{\boldsymbol{B}}^*\end{bmatrix} \begin{bmatrix}\mathbf{G}\\ \mathbf{U}\end{bmatrix} \parallel_F^2 + w_2 \parallel \mathbf{Y} - \bar{\boldsymbol{C}}^*\mathbf{G} \parallel_F^2 + w_3 \parallel \bar{\mathbf{G}} \begin{bmatrix}\mathbf{G}\\ \mathbf{U}\end{bmatrix}^{\dagger} \parallel_F^2 + w_4 \parallel \mathbf{Y}\mathbf{G}^{\dagger} \parallel_F^2 + w_5 \parallel \mathbf{G} \parallel_F^2 + w_6 \parallel \mathbf{G}\bar{\mathbf{G}}' \parallel_F^2), \tag{24}$$

where $w_1 \geq 0, w_2 \geq 0, w_3 \geq 0, w_4 \geq 0, w_5 \geq 0, w_6 \geq 0$ represent constant weights satisfying $w_1 + w_2 + w_3 + w_4 + w_5 + w_6 = 1$. At each iteration i, the proposed algorithm begins by computing the matrices $[\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*]$ and $\bar{\boldsymbol{C}}^*$ using 21-22. Subsequently, gradient descent is employed to update $\boldsymbol{\theta}$ in order to minimize the objective function in 24.

Remark 2 *Note that by following the definition of Moore–Penrose inverse (i.e., for any* $\boldsymbol{D} \in \mathbb{R}^{m\times n}$ *with full row rank,* $\boldsymbol{D}^{\dagger} = \boldsymbol{D}'(\boldsymbol{D}\boldsymbol{D}')^{-1}$*), the inverse terms* $(\begin{bmatrix}\mathbf{G}\\ \mathbf{U}\end{bmatrix}\begin{bmatrix}\mathbf{G}\\ \mathbf{U}\end{bmatrix}')^{-1}$ *and* $(\mathbf{G}\mathbf{G}')^{-1}$ *may pose challenges in computing the gradient of* $\mathbf{L}_f(\boldsymbol{\theta})$*. One way to address this issue is to utilize the relation* $\partial_\theta \boldsymbol{K}^{-1} = -\boldsymbol{K}^{-1}(\partial_\theta \boldsymbol{K})\boldsymbol{K}^{-1}$*, where* $\boldsymbol{K} \in \mathbb{R}^{n\times n}$*. This approach enables gradient computation involving matrix inverses in a more manageable form.*

To sum up, we have the following Algorithm 1 which is named as *deep Koopman learning with the noisy data* (DKND) in the rest of this paper.

4 Numerical Simulations

In this subsection, we first demonstrate the performance of the proposed algorithm by showing the estimation errors between the predicted system states and true noise-free states using four benchmark dynamics: one 2D simple linear discrete time-invariant dynamics of $\boldsymbol{x}_{t+1} = \begin{bmatrix}0.9 & -0.1\\ 0 & 0.8\end{bmatrix}\boldsymbol{x}_t + \begin{bmatrix}0\\ 1\end{bmatrix}\boldsymbol{u}_t$ with $\boldsymbol{x}_0 = [1, 0]'$, cartpole ($\boldsymbol{x}_t \in \mathbb{R}^4, \boldsymbol{u}_t \in \mathbb{R}$) and lunar lander ($\boldsymbol{x}_t \in \mathbb{R}^6, \boldsymbol{u}_t \in \mathbb{R}^2$) examples from the Openai gym Brockman et al. (2016), and one unmanned surface vehicles ($\boldsymbol{x}_t \in \mathbb{R}^6, \boldsymbol{u}_t \in \mathbb{R}^2$) example from Bingham et al. (2019). Then we compare the proposed DKND with its related methods.

Experiment setup. To experiment, we first collect the noise-free system states-inputs pairs $\mathcal{D} = \{(\boldsymbol{x}_t, \boldsymbol{u}_t)\}_{t=0}^{T}$ from the above four examples, where $\boldsymbol{u}_t$ is randomly generated control inputs following the uniform distribution and bounded between -1 and 1. Then we add three types of bounded measurement noise following the Gaussian distribution ($\boldsymbol{w}_t^G$ with mean $\mu = 0$ and standard deviation $\sigma = 2$), Poisson

Algorithm 1: Deep Koopman learning with the noisy data (DKND)

Input: $\mathbf{Y}$, $\bar{\mathbf{Y}}$, $\mathbf{U}$ in 19.
Output: $\boldsymbol{g}(\cdot, \boldsymbol{\theta}^*), \boldsymbol{A}^*, \boldsymbol{B}^*, \boldsymbol{C}^*$.
Initialization: Set the learning rate sequences $\{\alpha_i\}_{i=0}^S$ and terminal accuracy $\epsilon \geq 0$, build DNN $\boldsymbol{g}(\cdot, \boldsymbol{\theta}) : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^r$ with nonzero $\boldsymbol{\theta} \in \mathbb{R}^p$; initialize $[\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*]$ and $\bar{\boldsymbol{C}}^*$ by solving 21 and 22 respectively, and construct the loss function $\mathbf{L}_f(\boldsymbol{\theta})$ in 24.
for $i = 0, 1, 2, \cdots, S$ **do**
  Update the $\boldsymbol{\theta}$ using following gradient descent.

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \alpha_i \frac{d\mathbf{L}_f(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_i}.$$

  Update $[\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*]$ and $\bar{\boldsymbol{C}}^*$ by computing 21 and 22 respectively with $\boldsymbol{\theta}_{i+1}$.
  Stop if $\mathbf{L}_f(\boldsymbol{\theta}) < \epsilon$ and save the resulting $\bar{\boldsymbol{A}}^*$, $\bar{\boldsymbol{B}}^*$, $\bar{\boldsymbol{C}}^*$, and $\boldsymbol{\theta}$.
end

distribution ($\boldsymbol{w}_t^P$ with an expected separation $\lambda = 3$), and uniform distribution ($\boldsymbol{w}_t^U$ generated from the open interval $[-1, 2)$) on collected system states to achieve noisy measurements. For instance, we denote $\boldsymbol{y}_t^G = \boldsymbol{x}_t + \boldsymbol{w}_t^G$ as the noisy measurement states under Gaussian noise. Correspondingly, $\mathcal{D}^G = \{(\boldsymbol{y}_t^G, \boldsymbol{u}_t)\}_{t=0}^T$ denote the dataset under Gaussian noise. We allocate 80% of $\mathcal{D}^G$ to train DKND (denoted as $\mathcal{D}_{train}^G$), reserving the remaining 20% for testing (denoted as $\mathcal{D}_{test}^G$). To show the algorithm performance, for any dataset $\mathcal{D}_{test}^G$, we denote the following root mean square deviation (RMSD):

$$RMSD(\mathcal{D}_{test}^G) = \sqrt{\frac{1}{|\mathcal{D}_{test}^G|} \sum_{(\boldsymbol{y}_k, \boldsymbol{u}_k) \in \mathcal{D}_{test}^G} \|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{f}}(\boldsymbol{y}_k, \boldsymbol{u}_k, \boldsymbol{\theta}^*)\|^2},$$

where $|\mathcal{D}_{test}^G|$ denote the number of data pairs $(\boldsymbol{y}_k, \boldsymbol{u}_k)$ in $\mathcal{D}_{test}^G$ and $\hat{\boldsymbol{f}}$ denote the estimated dynamics from the proposed DKND method. We further compare DKND against three relative algorithms: DKL that solves 13 using noisy measurements $\boldsymbol{y}_k$, DMDTLS from Dawson et al. (2016), and the multilayer perceptron (MLP) approach. To fairly evaluate the algorithms, we assign the above methods with the same DNNs structure, training parameters (e.g., learning rate, training epochs, etc.), and training and testing datasets. Note that to reduce the effect of randomly initialized DNN parameters, for all gradient-based methods, we run the above process for 10 experimental trials and record the average RMSD errors and their standard variations in Tables over these trials.

RMSD	Methods	2D example	Cartpole	Lunar lander	Surface vehicle
Training data	DKND	0.2093±0.0046	0.5550±0.0059	0.4905±0.0055	0.6399±0.0074
	DKL	0.2149±0.0106	0.5604±0.0037	0.4730±0.0051	0.6355±0.0140
	MLP	0.2118±0.0016	0.3987±0.0006	0.4650±0.0027	0.4338±0.0031
	DMDTLS	0.2483	29.9163	0.6836	5.1183
Testing data	DKND	0.2126±0.0073	0.6113±0.0088	0.6785±0.0816	0.8571±0.1833
	DKL	0.2124±0.0072	0.6190±0.0088	0.8433±0.0737	0.9241±0.2366
	MLP	0.3721±0.0311	0.8757±0.0172	1.9348±0.1598	2.0417±0.3259
	DMDTLS	0.2514	28.3258	0.6551	9.8491
w_{max}	-	0.7773	0.9108	0.9650	0.9512

Table 1: Averaged RSMD of data under Gaussian noise.

RSMD	Methods	2D example	Cartpole	Lunar lander	Surface vehicle
Training data	DKND	0.4647±0.0013	0.7985±0.0045	0.8633±0.0016	0.9320±0.0033
	DKL	0.4707±0.0206	0.7996±0.0075	0.8565±0.0031	0.9373±0.0336
	MLP	0.4644±0.0011	0.6808±0.0017	0.8329±0.0019	0.8047±0.0022
	DMDTLS	0.4709	3.7518	0.9268	24.6653
Testing data	DKND	0.4719±0.0040	0.8288±0.0054	0.9898±0.0398	0.8761±0.1402
	DKL	0.4784±0.0211	0.8281±0.0088	1.0857±0.1158	0.9006±0.1699
	MLP	0.4888±0.0053	1.0596±0.0429	1.8316±0.1631	1.3917±0.2398
	DMDTLS	0.4709	4.4250	0.8958	38.7688
w_{max}	-	1.1180	1.2529	1.4142	1.4933

Table 2: Averaged RSMD of data under Poisson noise.

RSMD	Methods	2D example	Cartpole	Lunar lander	Surface vehicle
Training data	DKND	1.5424±0.0179	2.2417±0.0648	2.5698±0.0227	3.2663±0.0118
	DKL	1.7493±0.1877	2.3839±0.1021	2.6577±0.0422	3.6623±0.4292
	MLP	2.3287±0.0236	4.0224±0.0035	4.8612±0.0037	4.8083±0.0061
	DMDTLS	27.2598	39.2297	286.4929	65.5456
Testing data	DKND	1.7079±0.0114	2.1580±0.0724	2.8520±0.0838	3.8822±0.0992
	DKL	2.0752±0.2770	2.3234±0.1033	3.0809±0.0639	4.9586±0.9784
	MLP	2.6835±0.0831	4.8319±0.0939	6.2894±0.1411	8.0272 ±0.4762
	DMDTLS	26.7608	40.9191	288.2916	68.0840
w_{max}	-	5.2776	7.1579	8.1615	7.5907

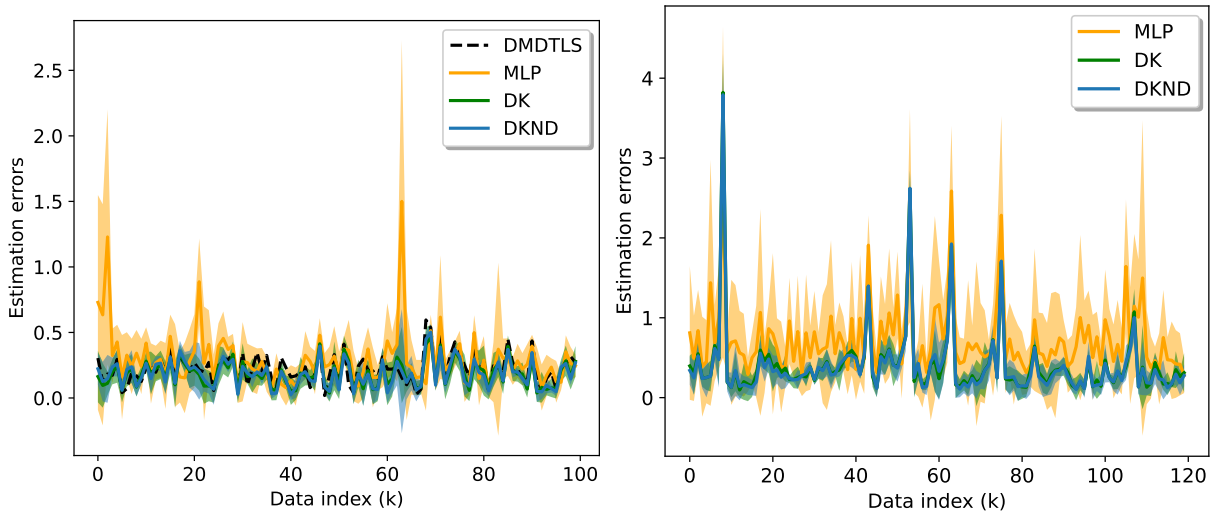
Table 3: Averaged RSMD of data under Uniform noise.

Results analysis. As shown in Tables 1-3, the proposed DKND achieves smaller averaged RSMD and standard deviation on testing data compared to other methods even when w_{max} is increasing. Specifically, when $\boldsymbol{w}_t$ follows a uniform distribution and w_{max} becomes larger, the RSMD gap between the proposed DKND and DKL methods is amplified. Figs. 1-3 provide detailed estimation error plots of $\boldsymbol{\epsilon}_k$ over the testing data, with the shaded regions indicating the variation across 10 trials. Due to space constraints, additional experimental details, including the generation of measurement noise, DNN structures, and training parameters, are included in the Appendix.

5 Discussion and conclusions

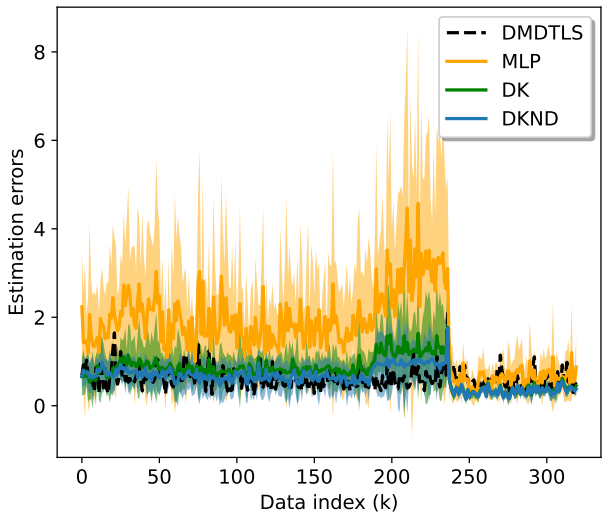
In this paper, we have introduced a data-driven framework called Deep Koopman Learning with Noisy Data (DKND) to address the challenge of learning system dynamics from data affected by measurement noise. By learning dynamics, we refer to estimating dynamics where, given $\boldsymbol{y}_t, \boldsymbol{u}_t$, the output of the estimated dynamics, $\hat{\boldsymbol{x}}_{t+1}$, approximates the true system state $\boldsymbol{x}_{t+1}$ with reasonable accuracy. The key contribution of this work lies in modifying the existing deep Koopman framework by explicitly characterizing the noise effect on the learned representation in 10 and mitigating it through tuning the DNN parameters to minimize 17 requiring only that the measurement noise be bounded. We evaluated the proposed DKND framework on datasets with three different types of measurement noise, using examples including simple 2D dynamics, cartpole, lunar lander, and surface vehicle systems. Our results demonstrate the robustness of DKND under different types of measurement noise compared to related methods.

Limitations. Since the formulation presented in this paper only addresses the scenario where the measurement noise is bounded, the effect of this bound on the performance of the proposed approach is not formally investigated and remains an open question. Due to the non-convex nature of DNN optimization, the DKND framework is inherently limited to achieving local minima. Future research could explore several aspects, including the design of optimal control strategies based on the learned dynamics and the measured system states. A potential direction for improving the algorithm is to integrate matrix computation and gradient descent steps, as suggested in Remark 2 and develop new techniques to determine the optimal weight constants in 24.

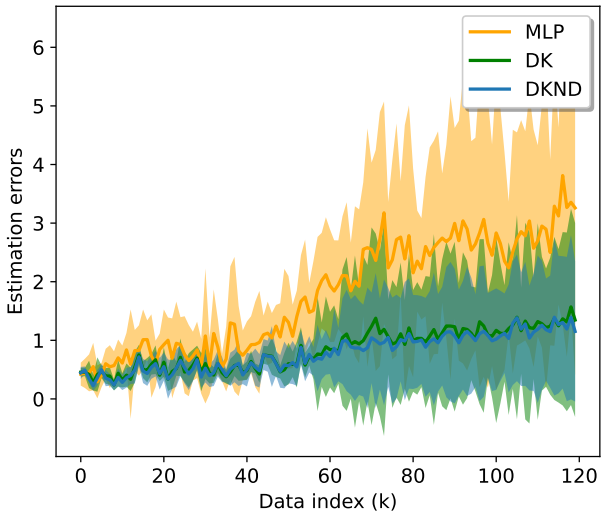


(a) 2D dynamics.

(b) Cartpole.

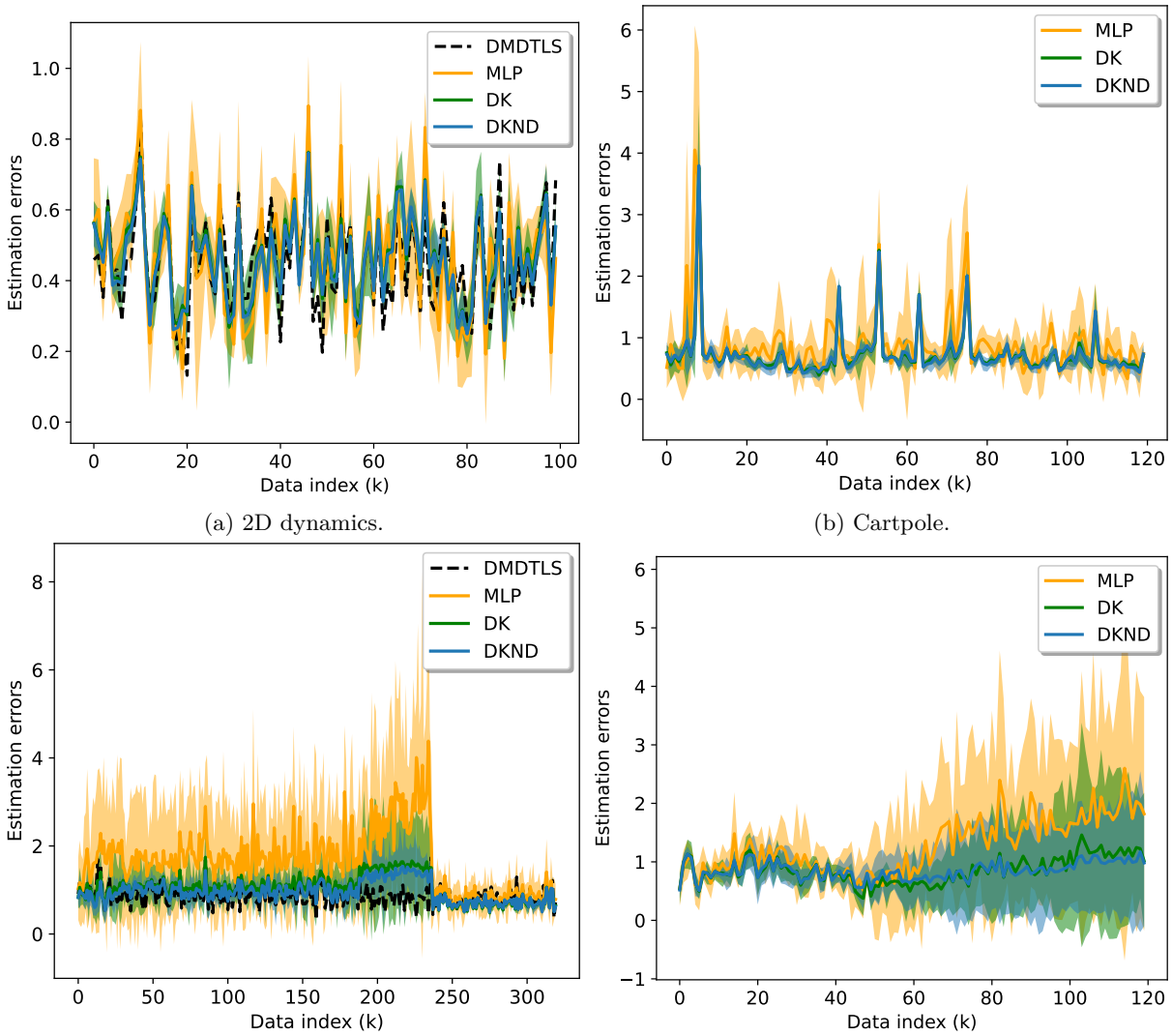


(c) Lunar lander.



(d) Surface vehicle.

Figure 1: Gaussian noise.



(a) 2D dynamics.

(b) Cartpole.

(c) Lunar lander.

(d) Surface vehicle.

Figure 2: Poisson noise.

References

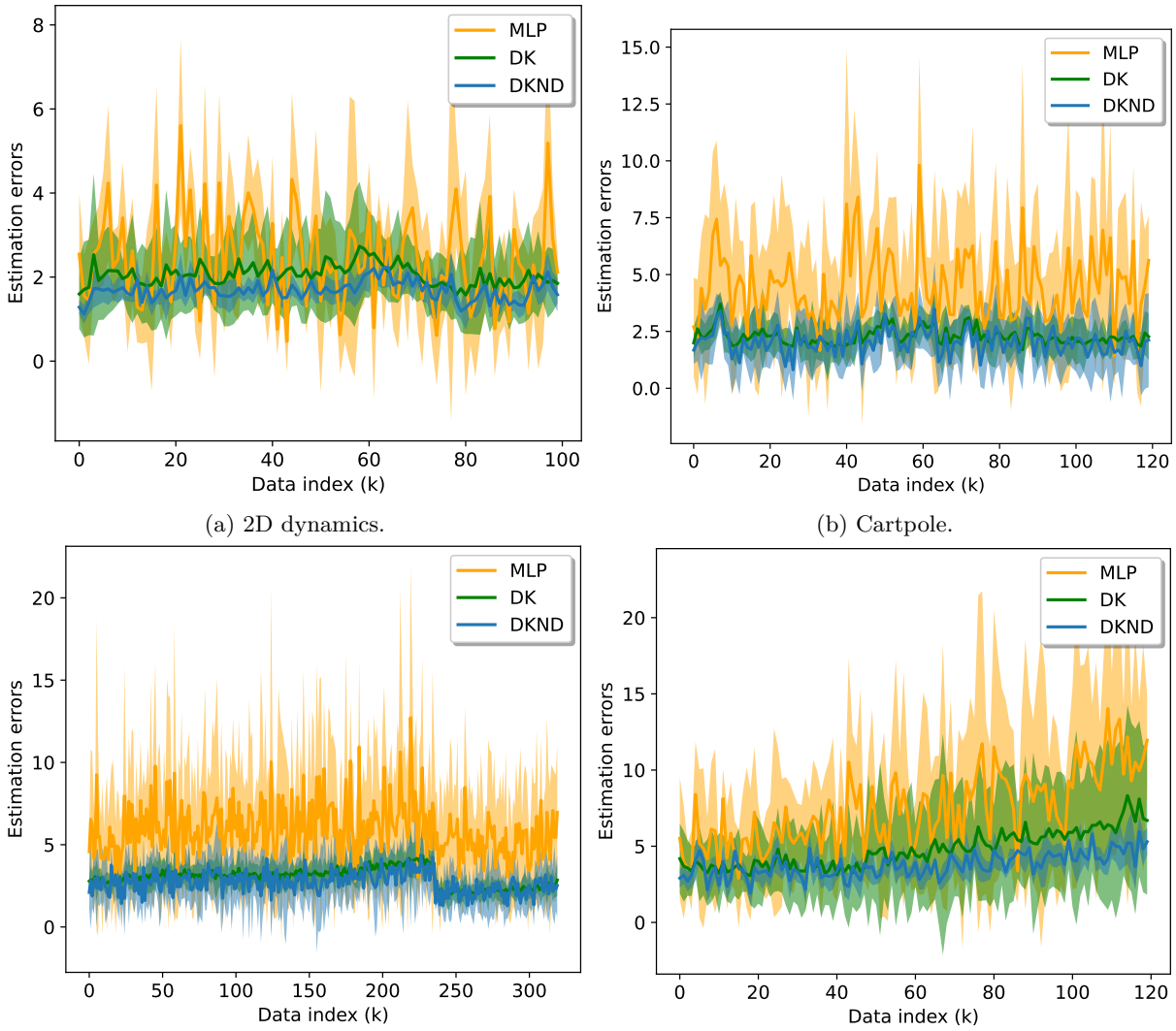
A.Lyapunov. The general problem of the stability of motion. *International Journal of Control*, 55(3), 1992.

Petar Bevanda, Max Beier, Sebastian Kerz, Armin Lederer, Stefan Sosnowski, and Sandra Hirche. Koopmanizingflows: Diffeomorphically learning stable koopman operators. *arXiv preprint arXiv:2112.04085*, 2021.

Brian Bingham, Carlos Agüero, Michael McCarrin, Joseph Klamo, Joshua Malia, Kevin Allen, Tyler Lum, Marshall Rawson, and Rumman Waqar. Toward maritime robotic simulation in gazebo. In *OCEANS 2019 MTS/IEEE SEATTLE*, pp. 1–10. IEEE, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Scott TM Dawson, Maziar S Hemati, Matthew O Williams, and Clarence W Rowley. Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition. *Experiments in Fluids*, 57: 1–19, 2016.



(a) 2D dynamics.

(b) Cartpole.

(c) Lunar lander.

(d) Surface vehicle.

Figure 3: Uniform noise.

Morgan T Gillespie, Charles M Best, Eric C Townsend, David Wingate, and Marc D Killpack. Learning nonlinear dynamic models of soft robots for model predictive control with neural networks. In *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 39–45. IEEE, 2018.

Yiqiang Han, Wenjian Hao, and Umesh Vaidya. Deep learning of koopman representation for control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 1890–1895. IEEE, 2020.

Wenjian Hao, Bowen Huang, Wei Pan, Di Wu, and Shaoshuai Mou. Deep koopman learning of nonlinear time-varying systems. *Automatica*, 159:111372, 2024.

Masih Haseli and Jorge Cortés. Approximating the koopman operator using noisy data: noise-resilient extended dynamic mode decomposition. In *2019 American Control Conference (ACC)*, pp. 5499–5504. IEEE, 2019.

Maziar S Hemati, Clarence W Rowley, Eric A Deem, and Louis N Cattafesta. De-biasing the dynamic mode decomposition for applied koopman spectral analysis of noisy datasets. *Theoretical and Computational Fluid Dynamics*, 31:349–368, 2017.

Bernard O Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the national academy of sciences of the united states of america*, 17(5):315, 1931.

Bernard O Koopman and J v Neumann. Dynamical systems of continuous spectra. *Proceedings of the National Academy of Sciences*, 18(3):255–263, 1932.

Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

Bethany Lusch, Steven L Brunton, and J Nathan Kutz. Data-driven discovery of koopman eigenfunctions using deep learning. *Bulletin of the American Physical Society*, 2017.

Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

Alexandre Mauroy and Jorge Goncalves. Linear identification of nonlinear systems: A lifting technique based on the koopman operator. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 6500–6505. IEEE, 2016.

Igor Mezić. On applications of the spectral theory of the koopman operator in dynamical systems and control theory. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 7034–7041. IEEE, 2015.

Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning*. University of California, Berkeley, 2002.

Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Generalizing koopman theory to allow for inputs and control. *SIAM Journal on Applied Dynamical Systems*, 17(1):909–930, 2018.

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.

Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

Nicholas Stearns Selby. *On the Application of Machine Learning and Physical Modeling Theory to Causal Lifting Linearizations of Nonlinear Dynamical Systems with Exogenous Input and Control*. MIT Dissertation, Cambridge, 2021.

Subhrajit Sinha, Sai P Nandanoori, and DA Barajas-Solano. Online real-time learning of dynamical systems from noisy streaming data. *Scientific Reports*, 13(1):22564, 2023.

Filippos E Sotiropoulos. *Methods for Control in Robotic Excavation*. MIT Dissertation, Cambridge, 2021.

Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pp. 4832–4839. IEEE, 2019.

A Appendix

A.1 Proof of Theorem 1.

Before we present the proof, the following notions are introduced. Let

$$\delta \boldsymbol{g}_t = \boldsymbol{g}(\boldsymbol{y}_t, \boldsymbol{\theta}) - \boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{\theta})$$

and due to the assumption that $\| \boldsymbol{w}_t \| \leq w_{max}$ and $\boldsymbol{g}(\cdot, \boldsymbol{\theta})$ is Lipschitz continuous, one has the immediate consequence of $\| \delta \boldsymbol{g}_t \| \leq L_g w_{max}$. We then introduce the following data matrices.

$$\begin{aligned}
\Delta \mathbf{G} &= [\delta \boldsymbol{g}_0, \delta \boldsymbol{g}_1, \cdots, \delta \boldsymbol{g}_{T-1}] \in \mathbb{R}^{r \times T}, \Delta \bar{\mathbf{G}} = [\delta \boldsymbol{g}_1, \delta \boldsymbol{g}_2, \cdots, \delta \boldsymbol{g}_T] \in \mathbb{R}^{r \times T}, \\
\mathbf{G}_x &= [\boldsymbol{g}(\boldsymbol{x}_0, \boldsymbol{\theta}), \boldsymbol{g}(\boldsymbol{x}_1, \boldsymbol{\theta}), \cdots, \boldsymbol{g}(\boldsymbol{x}_{T-1}, \boldsymbol{\theta})] \in \mathbb{R}^{r \times T}, \\
\bar{\mathbf{G}}_x &= [\boldsymbol{g}(\boldsymbol{x}_1, \boldsymbol{\theta}), \boldsymbol{g}(\boldsymbol{x}_2, \boldsymbol{\theta}), \cdots, \boldsymbol{g}(\boldsymbol{x}_T, \boldsymbol{\theta})] \in \mathbb{R}^{r \times T}, \\
\mathbf{X} &= [\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{T-1}] \in \mathbb{R}^{n \times T}, \bar{\mathbf{X}} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T] \in \mathbb{R}^{n \times T}, \\
\mathbf{W} &= [\boldsymbol{w}_0, \boldsymbol{w}_1, \cdots, \boldsymbol{w}_{T-1}] \in \mathbb{R}^{n \times T}, \bar{\mathbf{W}} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_T] \in \mathbb{R}^{n \times T},
\end{aligned} \tag{25}$$

From 19 and 25, one has

$$\mathbf{Y} = \mathbf{X} + \mathbf{W}, \quad \bar{\mathbf{Y}} = \bar{\mathbf{X}} + \bar{\mathbf{W}}, \quad \mathbf{G} = \mathbf{G}_x + \Delta \mathbf{G}, \quad \bar{\mathbf{G}} = \bar{\mathbf{G}}_x + \Delta \bar{\mathbf{G}}. \tag{26}$$

We now start from the minimization of 12 (over the noise-free trajectory) with respect to dynamics matrices, of which the solution is analogous to 21-22 (over the noisy trajectory). By using the notations in 19, one has the following results by rewriting 21-22:

$$[\tilde{\boldsymbol{A}}^*, \tilde{\boldsymbol{B}}^*] = \bar{\mathbf{G}}_x \begin{bmatrix} \mathbf{G}_x \\ \mathbf{U} \end{bmatrix}^\dagger, \tag{27}$$

$$\tilde{\boldsymbol{C}}^* = \mathbf{X} \mathbf{G}_x^\dagger. \tag{28}$$

We then expand the 27-28 using the notation in 26, it follows that

$$\begin{aligned}
&[\tilde{\boldsymbol{A}}^*, \tilde{\boldsymbol{B}}^*] \\
&= \bar{\mathbf{G}}_x \begin{bmatrix} \mathbf{G}_x \\ \mathbf{U} \end{bmatrix}^\dagger = (\bar{\mathbf{G}} - \Delta \bar{\mathbf{G}}) \begin{bmatrix} \mathbf{G} - \Delta \mathbf{G} \\ \mathbf{U} \end{bmatrix}^\dagger \\
&= (\bar{\mathbf{G}} - \Delta \bar{\mathbf{G}}) \begin{bmatrix} \mathbf{G} - \Delta \mathbf{G} \\ \mathbf{U} \end{bmatrix}' (\begin{bmatrix} \mathbf{G} - \Delta \mathbf{G} \\ \mathbf{U} \end{bmatrix} \begin{bmatrix} \mathbf{G} - \Delta \mathbf{G} \\ \mathbf{U} \end{bmatrix}')^{-1} \\
&= (\bar{\mathbf{G}} \begin{bmatrix} \mathbf{G} \\ \mathbf{U} \end{bmatrix}' - \underbrace{[\bar{\mathbf{G}} - \Delta \bar{\mathbf{G}}, \Delta \bar{\mathbf{G}}]}_{\mathbf{n}} \underbrace{\begin{bmatrix} \Delta \mathbf{G}' & 0 \\ \mathbf{G}' & \mathbf{U}' \end{bmatrix}}_{\mathbf{v}'}) (\underbrace{\begin{bmatrix} \mathbf{G} \\ \mathbf{U} \end{bmatrix} \begin{bmatrix} \mathbf{G} \\ \mathbf{U} \end{bmatrix}'}_{\mathbf{p}} + \underbrace{\begin{bmatrix} \Delta \mathbf{G} - \mathbf{G} & -\Delta \mathbf{G} \\ -\mathbf{U} & 0 \end{bmatrix}}_{\mathbf{u}} \begin{bmatrix} \Delta \mathbf{G}' & 0 \\ \mathbf{G}' & \mathbf{U}' \end{bmatrix})^{-1}
\end{aligned} \tag{29}$$

and

$$\begin{aligned}
\tilde{\boldsymbol{C}}^* &= \mathbf{X} \mathbf{G}_x^\dagger = (\mathbf{Y} - \mathbf{W})(\mathbf{G} - \Delta \mathbf{G})^\dagger, \\
&= (\mathbf{Y} - \mathbf{W})(\mathbf{G} - \Delta \mathbf{G})' ((\mathbf{G} - \Delta \mathbf{G})(\mathbf{G} - \Delta \mathbf{G})')^{-1}, \\
&= (\mathbf{Y} \mathbf{G}' - \underbrace{[\mathbf{Y} + \mathbf{W}, -\mathbf{W}]}_{\bar{\mathbf{n}}} \underbrace{\begin{bmatrix} \Delta \mathbf{G}' \\ \mathbf{G}' \end{bmatrix}}_{\bar{\mathbf{v}}'}) (\underbrace{\mathbf{G} \mathbf{G}'}_{\bar{\mathbf{p}}} + \underbrace{[-\mathbf{G} + \Delta \mathbf{G}, -\Delta \mathbf{G}]}_{\bar{\mathbf{u}}} \begin{bmatrix} \Delta \mathbf{G}' \\ \mathbf{G}' \end{bmatrix})^{-1}.
\end{aligned} \tag{30}$$

Applying Sherman–Morrison formula to 29-30, that is, for given invertible matrix $A \in \mathbb{R}^{n \times n}$ and column vectors $n, v \in \mathbb{R}^n$, if $1 + v' A^{-1} u \neq 0$, the following holds:

$$(A + uv')^{-1} = A^{-1} - \frac{A^{-1} u v' A^{-1}}{1 + v' A^{-1} u}.$$

It leads 29-30 to

$$[\tilde{\boldsymbol{A}}^*, \tilde{\boldsymbol{B}}^*] = [\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*] + (\mathbf{n} \mathbf{v}' \mathbf{p}^{-1} - [\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*]) \mathbf{u} \mathbf{v}' \mathbf{p}^{-1} (\mathbf{I} + \mathbf{v}' \mathbf{p}^{-1} \mathbf{u})^{-1} - \mathbf{n} \mathbf{v}' \mathbf{p}^{-1} \tag{31}$$

and

$$\tilde{\boldsymbol{C}}^* = \bar{\boldsymbol{C}}^* + (\bar{\mathbf{n}} \bar{\mathbf{v}}' \bar{\mathbf{p}}^{-1} - \bar{\boldsymbol{C}}^*) \bar{\mathbf{u}} \bar{\mathbf{v}}' \mathbf{p}^{-1} (\mathbf{I} + \bar{\mathbf{v}}' \bar{\mathbf{p}}^{-1} \bar{\mathbf{u}})^{-1} - \bar{\mathbf{n}} \bar{\mathbf{v}}' \bar{\mathbf{p}}^{-1}, \tag{32}$$

respectively. We then recall the dynamics difference from 16.

$$r(\bar{\boldsymbol{\theta}}^*, \boldsymbol{w}) = \frac{1}{2T} \max_{\boldsymbol{w}_k} (\sum_{k=0}^{T-1} \| \boldsymbol{g}(\boldsymbol{y}_k, \tilde{\boldsymbol{\theta}}^*) - \boldsymbol{g}(\boldsymbol{x}_k, \bar{\boldsymbol{\theta}}^*) \|^2 + \| [\tilde{\boldsymbol{A}}^*, \tilde{\boldsymbol{B}}^*] - [\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*] \|_F^2 + \| \tilde{\boldsymbol{C}}^* - \bar{\boldsymbol{C}}^* \|_F^2).$$

By following 31-32, $r(\bar{\boldsymbol{\theta}}^*, \boldsymbol{w})$ becomes

$$\begin{aligned} r(\bar{\boldsymbol{\theta}}^*, \boldsymbol{w}) =& \frac{1}{2T} \max_{\boldsymbol{w}_k} (\sum_{k=0}^{T-1} \|\delta \boldsymbol{g}_k\|^2 + \| \ (\mathbf{n}\mathbf{v}'\mathbf{p}^{-1} - [\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*])\mathbf{u}\mathbf{v}'\mathbf{p}^{-1}(\mathbf{I} + \mathbf{v}'\mathbf{p}^{-1}\mathbf{u})^{-1} - \mathbf{n}\mathbf{v}'\mathbf{p}^{-1} \ \|_F^2 \\ &+ \| \ (\bar{\mathbf{n}}\bar{\mathbf{v}}'\bar{\mathbf{p}}^{-1} - \bar{\boldsymbol{C}}^*)\bar{\mathbf{u}}\bar{\mathbf{v}}'\mathbf{p}^{-1}(\mathbf{I} + \bar{\mathbf{v}}'\bar{\mathbf{p}}^{-1}\bar{\mathbf{u}})^{-1} - \bar{\mathbf{n}}\bar{\mathbf{v}}'\bar{\mathbf{p}}^{-1} \ \|_F^2). \end{aligned} \tag{33}$$

Note that if $\mathbf{n}\mathbf{v}'\mathbf{p}^{-1} = [\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*]$ and $\bar{\mathbf{n}}\bar{\mathbf{v}}'\bar{\mathbf{p}}^{-1} = \bar{\boldsymbol{C}}^*$, then 33 becomes

$$r(\bar{\boldsymbol{\theta}}^*, \boldsymbol{w}) = \frac{1}{2T} \max_{\boldsymbol{w}_k} (\sum_{k=0}^{T-1} \|\delta \boldsymbol{g}_k\|^2 + \| \ [\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*] \ \|_F^2 + \| \ \bar{\boldsymbol{C}}^* \ \|_F^2). \tag{34}$$

Here to achieve

$$\mathbf{n}\mathbf{v}'\mathbf{p}^{-1} = [\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*] = \bar{\mathbf{G}} \begin{bmatrix} \mathbf{G} \\ \mathbf{U} \end{bmatrix}' \mathbf{p}^{-1}$$

and

$$\bar{\mathbf{n}}\bar{\mathbf{v}}'\bar{\mathbf{p}}^{-1} = \bar{\boldsymbol{C}}^* = \mathbf{Y}\mathbf{G}'\bar{\mathbf{p}}^{-1},$$

one can formulate the following loss function by adding the penalty functions in 33 defined as:

$$\begin{aligned} \hat{r}(\bar{\boldsymbol{\theta}}^*, \boldsymbol{w}) =& \frac{1}{2T} \max_{\boldsymbol{w}_k} (\sum_{k=0}^{T-1} \|\delta \boldsymbol{g}_k\|^2 + \| \ [\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*] \ \|_F^2 + \| \ \bar{\boldsymbol{C}}^* \ \|_F^2 + \| \ \bar{\mathbf{n}}\bar{\mathbf{v}}' - \mathbf{Y}\mathbf{G}' \ \|_F^2 + \| \ \mathbf{n}\mathbf{v}' - \bar{\mathbf{G}} \begin{bmatrix} \mathbf{G} \\ \mathbf{U} \end{bmatrix}' \ \|_F^2) \\ =& \frac{1}{2T} \max_{\boldsymbol{w}_k} (\sum_{k=0}^{T-1} \|\delta \boldsymbol{g}_k\|^2 + \| \ (\mathbf{Y} - \mathbf{W})(\mathbf{G} - \Delta\mathbf{G})' \ \|_F^2 + \| \ (\bar{\mathbf{G}} - \Delta\bar{\mathbf{G}}) \begin{bmatrix} \mathbf{G} - \Delta\mathbf{G} \\ \mathbf{U} \end{bmatrix}' \ \|_F^2 \\ &+ \| \ [\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*] \ \|_F^2 + \| \ \bar{\boldsymbol{C}}^* \ \|_F^2) \\ \leq& \frac{1}{2T} (TL_g^2 w_{max}^2 + \| \ (\mathbf{Y} + Tw_{max})(\mathbf{G} + TL_g w_{max})' \ \|_F^2 + \| \ (\bar{\mathbf{G}} + TL_g w_{max}) (\begin{bmatrix} \mathbf{G} + TL_g w_{max} \\ \mathbf{U} \end{bmatrix})' \ \|_F^2 \\ &+ \| \ [\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*] \ \|_F^2 + \| \ \bar{\boldsymbol{C}}^* \ \|_F^2 \ . \end{aligned} \tag{35}$$

Note here that the minimization of 35 is not equal to the minimization of 33 but they share the same optimal solution. Dropping constant terms in the upper bound of $\hat{r}(\bar{\boldsymbol{\theta}}^*, \boldsymbol{w})$, one has the following reformulated minimization problem of $r(\bar{\boldsymbol{\theta}}^*, \boldsymbol{w})$ regarding to $\bar{\boldsymbol{\theta}}^* \in \mathbb{R}^p$ as

$$\mathbf{L}_w(\bar{\boldsymbol{\theta}}^*) = \frac{1}{2T} (\| \ [\bar{\boldsymbol{A}}^*, \bar{\boldsymbol{B}}^*] \ \|_F^2 + \| \ \bar{\boldsymbol{C}}^* \ \|_F^2 + \| \ \mathbf{G} \ \|_F^2 + \| \ \mathbf{G}\bar{\mathbf{G}}' \ \|_F^2). \tag{36}$$

■

A.2 Simulation details

In this subsection, we provide the simulation details regarding the experiment in Section 4.

A.2.1 Computation resource and training parameters

A.2.2 DNNs architecture

The DNN architectures of method DKND and DKL used in this paper are presented in Table 5. We refer to `https://pytorch.org/docs/stable/nn.html` for the definition of functions $Linear()$, $ReLU()$ and we denote layer^i as the i-th layer of the DNN and $Linear([n, m])$ denotes a linear function with a weight matrix of shape $n \times m$. Since for the DKND and DKL methods, the input of its DNN observable function is the measured state $\boldsymbol{y}_t$ and the input of the MLP method is a stacked vector of $[\boldsymbol{y}_t', \boldsymbol{u}_t']'$ we show the DNNs structure of the MLP method in the following table.

	2D dynamics	Cartpole	Lunar lander	Surface vehicle
Optimizer	Adam			
Accuracy (ϵ)	$1e-4$			
Training epochs (S)	1e4			
Learning rate (α_i)	$1e-5$			
The number of data pairs (T)	500	600	1600	600
Compute device	Apple M2, 16GB RAM			

Table 4: Training parameters.

	2D dynamics	Cartpole	Lunar lander	Surface vehicle
layer^1 type	$Linear([2, 512])$	$Linear([4, 512])$	$Linear([6, 512])$	$Linear([6, 512])$
layer^2 type	$ReLU()$	$ReLU()$	$ReLU()$	$ReLU()$
layer^3 type	$Linear([512, 128])$	$Linear([512, 128])$	$Linear([512, 128])$	$Linear([512, 128])$
layer^4 type	$ReLU()$	$ReLU()$	$ReLU()$	$ReLU()$
layer^5 type	$Linear([128, 4])$	$Linear([128, 6])$	$Linear([128, 4])$	$Linear([128, 10])$

Table 5: DNN structures of DKND and DKL.

	2D dynamics	Cartpole	Lunar lander	Surface vehicle
layer^1 type	$Linear([3, 512])$	$Linear([5, 512])$	$Linear([8, 512])$	$Linear([8, 512])$
layer^2 type	$ReLU()$	$ReLU()$	$ReLU()$	$ReLU()$
layer^3 type	$Linear([512, 128])$	$Linear([512, 128])$	$Linear([512, 128])$	$Linear([512, 128])$
layer^4 type	$ReLU()$	$ReLU()$	$ReLU()$	$ReLU()$
layer^5 type	$Linear([128, 4])$	$Linear([128, 6])$	$Linear([128, 4])$	$Linear([128, 10])$

Table 6: DNN structures of MLP.